

System Security Configuration Guide for Cisco 8000 Series Routers, IOS XR Releases

Cisco 8000 Series Routers
Updated September 18, 2026

Topics included

1 What's changed in this book....14
Whats changed in this book....15

2 YANG Data Models for System Security Features....16
Using YANG Data Models....17

3 Trustworthy Systems....18
Trustworthy systems....19
How the trustworthy systems ecosystem works....19
Hardware trust....20
Hardware-anchored root of trust....20
How chip guard hardware integrity checks work....22
Software trust components....25
Secure boot....25
How secure iPXE servers work....28
Verify the secure boot status of your router....29
How trustworthiness works....30
Verify trust implementation boot status....31
Key security terms for trustworthy systems....31
TCG-compliant trusted platform module....32
Guidelines for TCG-compliant Trusted Platform Module....34
Benefits of TCG-compliant Trusted Platform Module....34
Verify TPM status and certificates....35
ECC P256-based Secure Unique Device Identifiers....40
ECC P256 cryptographic mechanism comparison....41
ECC P256 error messages....41
Verify ECC P256 configuration in the Trusted Anchor Module....42
Monitor XR shell commands....47
Export shell history before an install operation....48

4 Maintaining Trust....50
Steady-state trust....51
SELinux....51
Secure install mechanisms....53
SSD encryption....54
Runtime defenses....59
Boot integrity and trust visibility....61
Secure gRPC sessions....68

Integrity Measurement Architecture....68

5 AAA Access Model and Local User Administration 70

AAA service capabilities....71
Requirement: Prepare for AAA service configuration....71
Requirement: Verify AAA server compatibility and interoperability....71
User identities....72
Router user categories for AAA administrative access....72
User groups for AAA services....73
Task groups for AAA services....74
How group inheritance works....74
AAA administrative models....75
Caution: Protect administrative access during AAA configuration....75
AAA databases....76
Remote AAA configuration elements....77
AAA method lists and server groups....77
How AAA authentication works....78
Configure the first router user....80
Configure a task group....81
Configure a user group....82

6 AAA Password Security and Authorization Policies 88

AAA password types....89
Type 8 and Type 9 password encryption....89
Type 10 password encryption....92
Deprecated Type 7 password and Type 5 secret behavior....93
Type 7 password masking....99
AAA password security for FIPS compliance....102
AAA password security configuration attributes....103
Requirement: Use a six-character password for the first user....105
User secret policies....105
Configure an AAA password policy....106
Configure a password policy for user secrets and passwords....108
Failed-authentication username display....110
Consecutive-character password restrictions....113
Task-based authorization....119
Task IDs for authorization....120
Methods for assigning task permissions to external AAA users....121
Categories of AAA XML schema for configuration and monitoring....124
AAA method lists....125
Authentication method-list options and application 125
Configure an authentication method-list 128
Authorization method-list features and operation....129
Create an authorization method-list....131

Supported accounting methods and behavior for AAA method-lists 132
Create an accounting method-list 136
Configure interim accounting records 137
Application of AAA method-lists to console and vty access 139
Configure AAA authorization on lines 139
Enable accounting services 140
Configure the login response timeout for a line template 141
Command accounting methods 143
Configure command accounting 143
Local command authorizations with user accounts 144
How command authorization works 146
Configure command authorization with a local user account 147
Command authorization behaviors with TACACS+ and local fallback 148
Configuration patterns for AAA services 151

7 RADIUS Server Configuration and Secure Transport 154
RADIUS protocol 155
RADIUS unsuitable network security situations 156
How RADIUS authentication works 156
RADIUS with DTLS protection 157
Security and deployment benefits of RADIUS DTLS protection 159
How RADIUS with DTLS protection works 159
Requirement: Use supported RADIUS DTLS behavior 160
Configure RADIUS with DTLS protection 161
RADIUS with TLS protection 162
How RADIUS with TLS protection works 163
Requirement: Use supported RADIUS TLS behavior 165
Supported RADIUS TLS cipher suites 165
Configure RADIUS with TLS protection 166
Router-to-RADIUS server communications 168
Configure router to communicate with a RADIUS server 169
RADIUS dead-server detection features 172
Configure RADIUS dead-server detection 172
RADIUS server groups 174
Configure RADIUS server groups 174
Per-VRF AAA for RADIUS 176
Cisco RADIUS vendor-specific attributes for per-VRF AAA 176
Configure per-VRF AAA server groups 178
SSH remote public key authentication using TACACS+ 180
Key features and operational details 180
Support and restrictions of SSH remote public key authentication using TACACS+ 181
How SSH remote public key authentication using TACACS+ servers works 182
Configure SSH remote public key authentication using TACACS+ 184

8 TACACS+ Services for AAA....186
Differentiated Services Code Point (DSCP) values for TACACS+ packets....187
TACACS+ servers....187
TACACS+ server parameters....188
Configure TACACS+ servers....188
TACACS+ host, server group, and AAA configuration patterns....191
Configure TACACS+ server authorization....192
Configure TACACS+ server authentication....193
Configure TACACS+ server accounting....194
TACACS+ server groups....195
TACACS+ server group behavior....195
Configure TACACS+ server groups....196
Per-VRF TACACS+ server groups....197
Behavior and prerequisites for TACACS+ server groups in per-VRF configurations....198
Configure per-VRF TACACS+ server groups....198
TACACS+ operational statistics....200
TACACS+ operational command output....201
TACACS+ with TLS protection....205
Security benefits provided by TACACS+ with TLS protection....205
How TACACS+ with TLS protection works....206
Requirement: You must use supported TACACS+ TLS practices....207
Requirement: Avoid unsupported TACACS+ TLS behavior....207
Configure TACACS+ with TLS protection....207
SSH remote public key authentication using TACACS+....209
Key features and operational details....209
Support and restrictions of SSH remote public key authentication using TACACS+....210
How SSH remote public key authentication using TACACS+ servers works....211
Configure SSH remote public key authentication using TACACS+....212

9 CA Interoperability and Certificate Enrollment....216
Certification authority interoperability....217
Requirement: You must meet certification authority prerequisites....217
How CA interoperability works....217
Certification authority interoperability facts....238
Certification authority interoperability standards....239
Certification authorities....239

10 CA Trust Pools and PKI Certificate Lifecycle....242
Certificate authority trust pools....243
CA certificate bundles in trust pools....243
Requirement: CA trust pool prerequisites....243
Requirement: Follow CA trust pool restrictions....244

How CA trust pools are updated 244
CRL retrievals through HTTP proxy servers....246
Configure optional trust pool policy parameters....249
How CA certificates work in trust pools and trustpoints 250
PKI certificate expiry notifications 251
PKI alert notifications....251
Requirement: Follow PKI credential expiry alert restrictions....252
Regenerate PKI certificates....253
Automatic PKI certificate renewal....254
How automatic PKI certificate renewal works....256
Requirement: Meet automatic PKI certificate renewal prerequisites....257
Requirement: Automatic PKI certificate renewal guidelines....258
Configure automatic PKI certificate renewal....258

11 Crosswork Trust Insights and Public-Key Systems....260
How Cisco IOS XR and Crosswork Trust Insights integrate....261
Configure Cisco IOS XR and Crosswork Trust Insights integration....262
Ed25519 public-key signatures....279
Configure Ed25519 crypto keys....281
Configure Ed25519 Crosswork Trust Insights integration....282
Public key-pairs....283
Requirement: Follow public key-pair usage guidelines....285
Configure public key-pairs in XR config mode....286
System logs and error messages for public key-pair operations....287

12 Keychain Management 290
Keychains....291
Requirement: Account for system-clock changes....291
Keychain implementations....291
How keychain management works....292
Keychain management configuration examples....301

13 MACsec using EAP-TLS Authentication 304
MACsec EAP-TLS authentication sessions....305
Requirement: Follow EAP-TLS authentication limits....305
IEEE 802.1X device roles....305
Requirement: Meet MACsec MKA EAP-TLS prerequisites....306
MACsec local EAP-TLS authentication models....306
How MACsec EAP-TLS encryption works....307

14 uRPF Source Address Validation....320
uRPF source validation mechanisms....321
uRPF validation modes....323

uRPF loose mode....323
uRPF strict mode behaviors....326
uRPF VRF source address validation....329

15 Type 6 Password Encryption....334
Type 6 password encryption....335
Requirement: Enable the Type 6 primary key after an iPXE boot....335
How Type 6 password encryption works....335

16 Management Plane Protection 342
Management plane protection features....343
Requirement: Verify management-plane task permissions....343
Restriction: Account for management plane protection restrictions....343
Management plane protection....343
How management plane protection works....346
Configure an inband management plane protection interface....346
Configure an out-of-band management plane protection interface....347
Examples of management plane protection configurations....348
Additional references....350

17 Secure Shell Fundamentals and Configuration....352
Secure Shell fundamentals....353
SSH clients and servers....354
How secure file transfers work....354
SSH authentication methods....355
SSH and SFTP software packages....355
CiscoSSH....356
Post-quantum cryptography key exchange for CiscoSSH....356
CiscoSSH and Cisco IOS XR SSH differences....358
Restrictions for CiscoSSH....359
Recommendations for using CiscoSSH....359
CiscoSSH session event messages....361
Requirements for Secure Shell....362
Secure Shell restrictions....362
Configure an SSH server....364
NETCONF access controls....365
How NETCONF access controls work....366
Guidelines for NETCONF access control....366
NETCONF access control restrictions....366
Configure NETCONF access control....367
Automatic generation of SSH host keys....367
How SSH host-key pairs are generated....368
Configure allowed SSH host-key algorithms....369

Ed25519 public-key signature algorithm support for SSH....370
Guidelines for Ed25519 public-key signature algorithm usage....372
Generate an Ed25519 public key....372
Configure an SSH client....373
Troubleshoot SSH client connections....374
SSH client strict host key checks....375
Restrictions for SSH client strict host key check....376
SSH client knownhosts commands....376
Configure SSH client strict host key checking....377
SSH client authentication order....379
Set the SSH client authentication order....379
SSH multichannel connections....380
How SSH multichannel connections work....380
SSH multichannel connection restrictions....381
Configure an OpenSSH client for multiplexing....381
SSH cipher and HMAC controls....382
Supported SSH encryption algorithms....382
Caution: Avoid deprecated SSH algorithms....383
Disable an SSH HMAC algorithm....383
Configure SSH cipher algorithms....383

18 Certificate and Public-Key Authentication for SSH....386
SSH certificate and public-key authentication....387
X.509v3 certificate authentication....387
How X.509v3 certificate authentication works....389
X.509v3 certificate authentication restrictions....391
Configure X.509v3 certificate authentication....391
X.509v3 certificate extension validation over mTLS....393
OpenSSH certificate authentication....395
How OpenSSH certificate authentication works....396
Prerequisites for OpenSSH certificate authentication....398
Configure OpenSSH certificate authentication....398
TACACS+ authorization for OpenSSH certificate users....401
Authorize OpenSSH certificate users through TACACS+....402
Public-key authentication for SSH clients....403
How SSH client public-key authentication works....404
Guidelines for public-key authentication of SSH clients....405
Enable public-key authentication for an SSH client....406
Delete SSH client authentication keys....407
Public-key authentication to SSH servers....407
How multiple public keys authenticate a user....409
Guidelines for public-key authentication to SSH servers....410
Configure public-key authentication to a router....411
Delete user public keys from a router....412

19 SSH Access and Connection Management....414
SSH access and connection controls....415
FIDO2 authentication for SSH....415
FIDO2 authentication guidelines....415
FIDO2 authentication restrictions....416
Configure FIDO2 authentication for SSH....416
SSH authentication attempt limits....418
SSH authentication attempt restrictions....419
Set the maximum SSH authentication attempts....419
Multifactor authentication for SSH....420
How Duo multifactor authentication works....421
Set up multifactor authentication for SSH....422
Selective SSH server authentication methods....425
Disable SSH server authentication methods....427
SSH port forwarding....427
How SSH port forwarding works....429
SSH port forwarding restrictions....430
Enable SSH port forwarding....430
DSCP marking for SSH packets....431
Set DSCP marking during SSH connection establishment....432
SSH server timeouts....433
Unused SSH connection timeouts....433
SSH channel timeouts....436

20 FIPS Mode and Cryptographic Services....440
FIPS mode....441
Guidelines and restrictions for FIPS mode....441
Enable FIPS mode....442
FIPS-compliant cryptographic keys....443
Supported cryptographic key types and sizes in FIPS mode....444
Generate and verify FIPS-compliant cryptographic keys....444
Cryptographic services in FIPS mode....446
Configure a FIPS-compliant key chain....446
Configure FIPS-compliant certificates....447
Configure FIPS-compliant OSPFv3....447
Configure a FIPS-compliant SNMPv3 server....448
SSH clients and servers in FIPS mode....449
Connect to an SSH server with FIPS-approved ciphers....449
Configure a FIPS-compliant SSH server....449

21 Implementing Secure Logging....452
System logging over Transport Layer Security (TLS)....453

How the TLS handshake establishes secure logging....453
Restrictions for syslogs over TLS....454
Configure syslogs over TLS....455
Security template framework for TLS applications....457
Restrictions for security template....458
Configuration guidelines for security template framework....458
How the security template framework works....459
Configure a security template....460
TLS RFC 5289 compliance for security template....463

22 802.1X and MAC-Based Port Authentication....464
Port-based authentication methods....465
802.1X port-based authentication....465
MAC Authentication Bypass....475
802.1X authentication with MAC Authentication Bypass fallback....483
How 802.1X authentication falls back to MAC Authentication Bypass....484
802.1X and MAC Authentication Bypass failure behavior....485
Configure 802.1X authentication with MAB fallback....485
RADIUS Change of Authorization for 802.1X and MAB sessions....487
How RADIUS Change of Authorization works....488
RADIUS Change of Authorization response codes and session attributes....489
Guidelines for RADIUS CoA for 802.1X and MAB sessions....490
Enable RADIUS CoA for 802.1X and MAB sessions....490

23 Device Ownership and Authorized Operations....494
Trusted ownership and authorized operations....495
Device ownership....495
How device ownership is established....496
Install and verify device ownership....497
Clear device ownership....498
Cisco MASA service....500
How MASA authenticates routers....502
Ownership voucher use cases....504
MASA entities and permissions....504
MASA REST APIs....509
MASA gRPC integration....510
How routers are provisioned using ownership vouchers....512
Third-party key packages....513
Key package versions and Cisco IOS XR releases....514
Restrictions for key packages....514
How key packages are processed....515
Install key packages on the router....515
How key rotation works....517
Consent tokens....517

Cisco-signed and customer consent tokens....518
How Cisco-signed consent tokens authorize restricted operations....518
Provision Cisco-signed consent tokens....519
Provision customer consent tokens....520

24 Gated shell access....522
Key features of gated shell access....523
How gated shell access works....523
Restrict shell access using a Cisco-signed Consent Token....524
Unrestrict shell access using a Cisco-signed Consent Token....525
Restrict shell access using an owner Consent Token....526
Unrestrict shell access using an owner Consent Token....527
Terminate a pending Consent Token authorization request....528
Access a restricted shell....529

25 Implementing Lawful Intercept....532
Lawful intercept....533
Lawful intercept benefits....534
Lawful intercept prerequisites....534
Lawful intercept topology....535
Lawful intercept restrictions....536
LI package installation and removal....537
Install and activate the LI package....537
Uninstall the LI package....539
SNMPv3 access for lawful intercept....543
Disable SNMP-based lawful intercept....543
Configure inband management plane protection for LI....543
Enable the lawful intercept SNMP server configuration....544
Additional lawful intercept information....545
Interception mode....545
How lawful intercept data flows....545
Lawful intercept scale and performance values....546
Intercept IPv4 and IPv6 packets....546
How high availability preserves lawful intercept flows....547

26 EST Protocol for Automated Certificate Provisioning....550
EST protocol for automated certificate provisioning....551
EST protocol features and benefits....552
EST protocol components and terminology....552
EST client authentication options....552
EST protocol configuration guidelines and limitations....553
Supported key types....553
Client authentication methods....553

TLS validation and authentication requirements....553
EST re-enrollment....553
EST client support for multiple trustpoints....553
Certificate types and their uses in the EST protocol....553
Trust anchor databases and their uses in the EST protocol....554
EST message types....554
Configure the EST protocol....555

27 Upgrade CURL libraries for HTTP client....558
Key benefits for libcurl upgrades....559
Support and restrictions for libcurl upgrade....559
How libcurl upgrade works....560

1 What's changed in this book

Topics:

- Whats changed in this book

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Whats changed in this book

This cumulative guide provides a single, continuously updated version that includes all the latest IOS XR features and release updates. It simplifies your experience by letting you bookmark one link and access the complete guide, instead of navigating through multiple release-specific versions.

Specific changes or updates tied to individual releases are clearly called out within the relevant sections. For a list of features introduced in a specific release, refer to the Release Notes.

The table lists the release numbers for which this document has been updated since its initial publication.

Table 1: Changes to this document

Date	Summary
September 2026	First published for Release 26.3.1.

2 YANG Data Models for System Security Features

Topics:

- Using YANG Data Models

Lists the YANG data models available for implementing and managing System Security features on Cisco devices.

This chapter provides information about the YANG data models for System Security features.

Using YANG Data Models

Lists Cisco IOS XR YANG data model repositories, tools for accessing and exploring models, and tips for locating specific models, containers, and definitions.

Cisco IOS XR supports a programmatic way of configuring and collecting operational data of a network device using YANG data models. Although configurations using CLIs are easier and human-readable, automating the configuration using model-driven programmability results in scalability.

The data models are available in the release image, and are also published in the Github repository. Navigate to the release folder of interest to view the list of supported data models and their definitions. Each data model defines a complete and cohesive model, or augments an existing data model with additional XPaths. To view a comprehensive list of the data models supported in a release, navigate to the *Available-Content.md* file in the repository.

You can also view the data model definitions using the YANG Data Models Navigator tool. This GUI-based and easy-to-use tool helps you explore the nuances of the data model and view the dependencies between various containers in the model. You can view the list of models supported across Cisco IOS XR releases and platforms, locate a specific model, view the containers and their respective lists, leaves, and leaf lists presented visually in a tree structure. This visual tree form helps you get insights into nodes that can help you automate your network.

To get started with using the data models, see the *Programmability Configuration Guide*.

3 Trustworthy Systems

Topics:

- Trustworthy systems
- Hardware trust
- Software trust components
- How trustworthiness works
- Key security terms for trustworthy systems
- TCG-compliant trusted platform module
- ECC P256-based Secure Unique Device Identifiers
- Monitor XR shell commands

Short description: Outlines core concepts, hardware and software components, operational processes, security mechanisms, and verification procedures that establish trustworthy systems, highlighting key technologies including roots of trust, secure boot, TPM, ECC P256, and essential security terms.

Trustworthy systems

Introduces the importance of trustworthy systems, describing their foundational role and explaining how the trustworthy systems ecosystem operates to ensure the reliability and security of modern technology infrastructures.

A trustworthy system is a network infrastructure system that

- does what it is expected to do in a verifiable way
- prevents tampering and reports system integrity, and
- makes security a primary design consideration.

Global service providers, enterprises, and government networks rely on the unimpeded operation of complex computing and communications networks. The integrity of the data and IT infrastructure is foundational to maintaining the security of these networks and user trust. With the evolution to anywhere, anytime access to personal data, users expect the same level of access and security on every network. The threat landscape is also changing, with adversaries becoming more aggressive. Protecting networks from attacks by malicious actors and from counterfeit and tampered products becomes even more crucial.

Routers are the critical components of the network infrastructure and must be able to protect the network and report on system integrity. A “trustworthy solution” is one that does what it is expected to do in a verifiable way. Building trustworthy solutions requires that security is a primary design consideration. Routers that constitute trustworthy systems are a function of security, and trust is about preventing as well as knowing whether systems have been tampered with.

How the trustworthy systems ecosystem works

Describes how trust originates in hardware, moves through the boot process, and culminates in runtime security controls within trustworthy systems. This enables secure measurement and attestation of hardware, firmware, and software components.

In trustworthy systems, the chain of trust starts at the lowest level in hardware and propagates through firmware, the boot process, the operating system (OS) kernel, and runtime applications. Each stage relies on secure methods to measure and attest to the integrity of preceding layers.

Summary

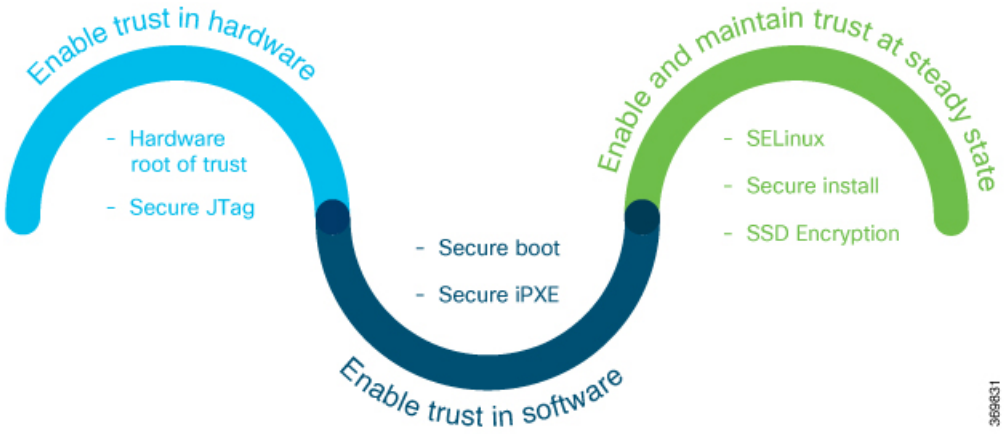
The key components involved in the process are:

- Hardware trust mechanisms: Hardware root-of-trust and secure JTAG provide foundational trust by enabling secure measurement and attestation.
- Software trust mechanisms: Secure boot and secure iPXE ensure the integrity of the software as it loads during the boot process.
- Steady-state trust controls: Security-Enhanced Linux (SELinux), secure install procedures, and SSD encryption maintain trust once the system is operational.

Trustworthy systems establish and maintain trust across hardware, software, and runtime environments by implementing processes that verify and protect each layer of the stack.

Workflow

Figure 1: Ecosystem of trustworthy systems



Trustworthy systems must have methods to securely measure hardware, firmware, and software components and to securely attest to these secure measurements.

These stages describe how the trustworthy systems ecosystem works.

1. Enabling trust in hardware: The system initializes with hardware root-of-trust and secure JTAG, measuring and attesting hardware components.
2. Enabling trust in software during boot: Secure boot and secure iPXE verify the authenticity and integrity of firmware and software loaded during the boot process.
3. Maintaining trust at runtime: Once the OS is running, security controls such as SELinux, secure install, and SSD encryption ensure continued integrity and protection of the system.
4. Secure measurement and attestation: At every stage, the system measures hardware, firmware, and software, and securely attests to those measurements, enabling external validation of trustworthiness.

Result

The router establishes enduring trust across hardware, software, and runtime controls, ensuring that each component is securely measured and verified, creating a trustworthy operational ecosystem.

Hardware trust

Details hardware trust principles, including hardware-anchored roots of trust, integrity checks with chip guard, attestation processes, and required service requests for hardware validation failures, with instructions for verifying hardware trust and PCR quotes.

Hardware trust is a trustworthy systems foundation that

- anchors system integrity in hardware
- uses immutable trust components, and
- supports later software and runtime trust validation.

Additional reference information

Trust in the hardware is enabled through specific components within the Cisco 8000 Series Routers that ensure the integrity and reliability of the system.

Hardware-anchored root of trust

Explains how anti-counterfeit chips, SUDI, and secure JTAG establish hardware trust.

A hardware-anchored root of trust is an immutable hardware trust component that

- starts the chain of trust at boot time
- verifies each boot stage before execution, and
- helps prove that device hardware and software are authentic.

The first component in implementing a trustworthy system is to enable trust in hardware.

Because software alone cannot prove a system's integrity, truly establishing trust must also be done in the hardware using a hardware-anchored root of trust. Without a hardware root of trust, no amount of software signatures or secure software development can protect the underlying system if the underlying system is compromised. To be effective, this root of trust must be based on an immutable hardware component that establishes a chain of trust at boot time. Each piece of code in the boot process measures and checks the signature of the next stage of the boot process before the software boots.

A hardware-anchored root of trust is achieved through:

- Anti-counterfeit chip: All modules that include a CPU, as well as the chassis, are fitted with an anti-counterfeit chip. This chip supports co-signed secure boot, secure storage, and boot-integrity-visibility. It ensures the device's software and hardware are authentic and prevents tampering or unauthorized modifications. It also enforces strong authentication and access control policies to protect sensitive data.
- Secure Unique Device Identifier (SUDI): An X.509 SUDI certificate installed at manufacturing provides a unique device identifier, enabling anti-counterfeit checks, authentication, and remote provisioning. The SUDI is generated using the device's unique hardware identifier—such as its serial number or MAC address—and a securely stored private key. This ensures that each SUDI is unique and cannot be easily duplicated or forged. When a device attempts to connect to a network, the network uses the SUDI to authenticate the device, ensuring only trusted devices are permitted access.
- Secure JTAG: A secure JTAG interface is used for debugging and downloading firmware, protected by asymmetric-key-based authentication and verification protocols. This prevents attackers from modifying firmware or stealing confidential information. Secure JTAG includes hardware and software measures, such as encryption and authentication protocols for securing communications, and access control policies to restrict interface access to authorized users.

Additional reference information

Hardware-anchored roots of trust are enabled by default on Cisco 8000 Series Routers.

Verify hardware trust status

Verify hardware trust status in the Cisco 8000 Series Router.

Confirm whether hardware-anchored root of trust is enabled and integrity is verified in the router.

Hardware-anchored root of trust is enabled by default on Cisco 8000 Series Routers. Verifying the hardware trust status ensures device integrity and compliance.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

Display hardware integrity status.

Example

```
Router# show platform security integrity hardware
Wed Apr 17 11:19:03.202 UTC
```

```
+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 52050
Known-good-digests:
Index   value
  0     hh4jzFBlxSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
observed-digests:
Index   value
  0     hh4jzFBlxSGHZ4hKqnC2FEjqHg4tpx/chZ7YcTwLCco=
PCRs:
Index   value
  15    Dl1BGskyzeJ1LNYKuZK8Qqllwkth0ru+0xWydL9YMdc=
```

The output provides information about the node location, TPM name, uptime, known-good digests, observed digests, and PCRs. Verify that the values match and trust is enabled for the hardware.

The command output confirms the hardware integrity status for the node, indicating whether the hardware trust is enabled.

How chip guard hardware integrity checks work

Describes how chip guard detects counterfeit ASIC or CPU components during secure boot.

Chip guard is designed to address hardware supply chain risks where attackers might replace ASIC or CPU chips in a Cisco router with counterfeit or malicious units. Compromised chips threaten performance, reliability, and security, potentially introducing hidden vulnerabilities. By validating chips before deployment, chip guard helps preserve hardware integrity.

Summary

The key components involved in the process are:

- Application Specific Integrated Circuit (ASIC): Executes critical hardware operations, such as scanning the egress queue for error causes.
- CPU chip: Runs the router's operating system and manages core processing.
- TAm chip with Imprint and Observed databases: Stores and compares SHA-256 chip ID hashes to verify authenticity during boot.

Chip guard hardware integrity checks help ensure Cisco routers contain authentic ASIC and CPU components during secure boot, protecting against supply chain attacks and counterfeit chips.

Workflow

These stages describe how the chip guard hardware integrity checks work.

1. Router manufacturing: The TAm chip is programmed with SHA-256 hashes (digital fingerprints) of the CPU and ASIC chip IDs, stored in the Imprint database (Imprint DB). These values remain immutable for the router's lifetime.
2. Router deployed and powered up: During the secure boot process, the chip guard feature recomputes the SHA-256 hashes of the CPU and ASIC chip IDs, creating the Observed database (Observed DB) within the TAm chip.
3. Comparison of Imprint and Observed databases:
    - If the Imprint DB and Observed DB match, chip guard validates the hardware as authentic and allows the router to continue booting.
    - If the databases do not match, chip guard identifies a counterfeit chip and halts secure boot. A warning message is displayed, indicating that supply chain integrity has been compromised.

Result

Chip guard ensures that only genuine ASIC and CPU components are used in the Cisco router. If counterfeit chips are detected, secure boot is halted, preventing compromised devices from entering the network.

Stages of chip guard implementation

The table shows the various stages through which chip guard is implemented on the router.

Table 2: Table

Stage	Process/Action	Result
1. Router Manufacturing	SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC are programmed in the TAm chip and stored in a database known as Imprint DB.	The Imprint DB inside the TAm chip contains the SHA 256 hashes, which cannot be modified during the router's lifetime.
2. Router Deployed in the Field and Powered Up	During the secure boot process, the chip guard feature recomputes the SHA 256 hashes of the electronic chip IDs of both the CPU and ASIC and creates a database known as Observed DB.	The Observed DB values are stored inside the TAm chip.
3. Comparison of Imprint DB and Observed DB	DBs match	The router continues to boot. Depending on the capability of the underlying router, the chip guard feature validates either the CPU, ASIC, or both.
	DBs do not match	The router notifies that either the CPU or ASIC is counterfeit, and the secure boot process halts. A message is displayed on the console about the chip guard validation failure.

Attestations

Explains how external verifiers use TAm-signed quotes and audit logs to validate device integrity.

Attestations are integrity validation mechanisms that

- let external verifiers request device integrity quotes
- use TAm-signed PCR quotes and audit logs, and
- protect verification using nonces and router-specific attestation keys.

Attestations enable external verifiers to check the integrity of the software running on the host. Implementing this feature on Cisco hardware helps you validate the trustworthiness of hardware and software for network devices.

Once a router is up and running, you can send a request to an external verifier. The external verifier requests an attestation quote from the router. The TAm chip outputs the PCR quote and audit log, signing the quote using an attestation private key unique to that router, and responds to the verifier. The verifier uses Cisco-provided KGV hashes and the Attestation Public Certificate to verify the attested PCR quotes and audit logs. This verification is protected against repeat attacks using a nonce. Additionally, verification ensures attestation is specific to a particular router by using attestation key pairs. These attestation key pairs are unique to each router and ensure attackers cannot tamper with router hardware, boot keys, boot configuration, or running software.

Verify attestation PCR quotes

Verify the attestation PCR quote to confirm hardware integrity.

Ensure that hardware integrity is maintained by checking the PCR quote and its associated signature.

Use this task to verify the hardware integrity proof exposed by Cisco IOS XR platforms via attestation PCR quotes. This check is typically performed as part of security audits or compliance initiatives.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

Use the **show platform security attest pcr 0 trustpoint ciscoaik nonce 4567 location 0/RP0/CPU0** command to display the attestation PCR quote.

Example

```
Router# show platform security attest pcr 0 trustpoint ciscoaik nonce 4567
location 0/RP0/CPU0
Thu Apr 11 05:44:10.808 UTC
Nonce: 4567

+--------------------------------------+
   Node location: node0_RP0_CPU0
+--------------------------------------+
Uptime: 1198700
pcr-quote:
/1RPAVACCsyBEFKzBNunfDQMEIg6plehKFWOkpOFQARwAAAAB5fDQAA97/////AQAACQAAAAAAAAAQAAAAAwAAAAgE798lOkf6lkyT5OG/V46lQuuSvGCiwj3Gq+
pcr-quote-signature:
mC8oPWYzgSIge3lDXCs/Ez7BRKsZyDb4auuhIagWha3aHSa9eMf34Y/EM1ItjeAncs--<truncated>--dJYpsEMGkcrolIquIhaDlgKIH+Gh4QBewdNky3Igiw==
pcr-index       pcr-value
  0             sL3H+Em2xzxXrNUoDF+kC47IXxN4V/d/7hYUXApXNoY=
```

Proof of hardware integrity is recorded in the TAm as part of Chip Guard.

> **Note**
>
> The same data is also available through NETCONF for a remote attestation server: Cisco-IOS-XR-remote-attestation-act.yang.

Hardware integrity is verified. You have obtained the nonce, PCR quote, quote signature, and PCR value, confirming the platform's attested state.

Requirement: hardware validation failure service request

Outlines the required steps to take when chip guard reports a hardware validation failure.

If you receive a chip guard warning about integrity check failure, you must create a service request on the Products Returns & Replacement (RMA) website.

Software trust components

Explains software trust mechanisms comprising secure boot, trusted software boot processes, secure iPXE server operations, and provides procedures to verify router secure boot status for comprehensive software integrity.

Software trust is a trustworthy systems component that

- verifies Cisco-signed boot artifacts
- authenticates network boot sources, and
- prevents untrusted software from reaching runtime.

The second component in implementing a trustworthy system is to enable trust in software.

In Cisco IOS XR7, trust in the software is enabled through:

- Secure Boot
- Secure iPXE

Secure boot

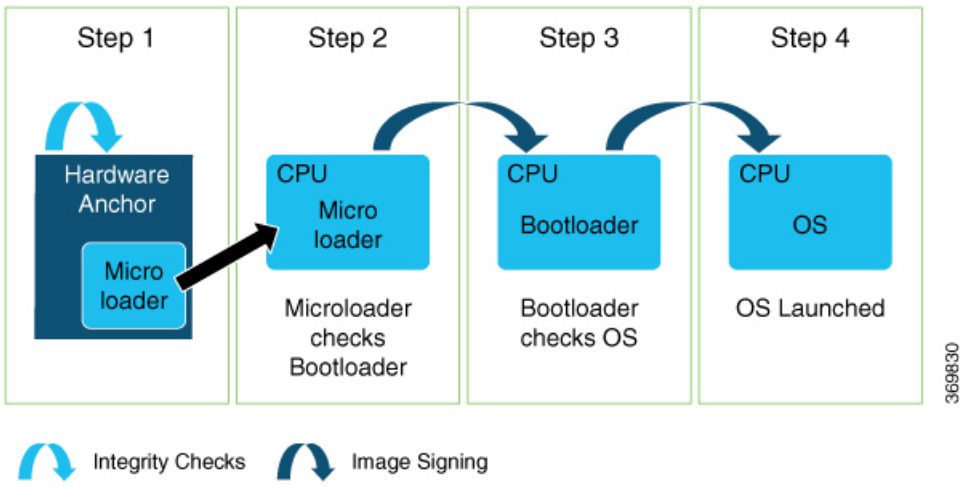
Explains how Cisco Secure Boot verifies boot artifacts and Cisco IOS XR software authenticity.

Secure boot is a software integrity mechanism that

- uses the TAm as the hardware trust anchor
- verifies Cisco-signed boot artifacts before execution, and
- rejects software images when signature verification fails.

Cisco Secure Boot helps to ensure that the code that executes as part of the software image boot sequence on Cisco routers is authentic and unmodified. Cisco IOS XR7 platforms support the hardware-anchored secure boot which is based on the standard Unified Extensible Firmware Interface (UEFI). This UEFI-based secure boot protects the microloader (the first piece of code that boots) in tamper-resistant hardware, establishing a root of trust that helps prevent Cisco network devices from executing tainted network software.

Figure 2: Secure boot



The intent of Secure Boot is to have a trust anchor module (TAm) in hardware that verifies the bootloader code. A fundamental feature of secure boot is the barrier it provides that makes it that it is extremely difficult or nearly impossible to bypass these hardware protections.

Secure boot ensures that the bootloader code is a genuine, unmodified Cisco piece of code and that code is capable of verifying the next piece of code that is loaded onto the system. It is enabled by default.

When secure boot authenticates the software as genuine Cisco in a Cisco device with the TAm, the operating system then queries the TAm to verify whether the hardware is authentic. It verifies by cryptographically checking the TAm for a secure unique device identifier (SUDI) that comes only from Cisco.

The SUDI is permanently programmed into the TAm and logged by Cisco during Cisco's closed, secured, and audited manufacturing processes.

Feature history

The feature history table lists release support for this feature.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
Secure Boot Status	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Secure Boot Status	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Secure Boot Status	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100])(select variants only*); Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Secure Boot Status	Release 24.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100](select variants only*) *This feature is supported on Cisco 8202-32FH-M routers.
Secure Boot Status	Release 7.8.1	You can now verify whether the router is securely booted up with an authentic Cisco software image. We have introduced a **show** command to verify the secure boot status of the router. If the software image was tampered with, then the secure boot fails, and the router does not boot up. Before this release, there was no provision on the router to verify the secure boot status. The feature introduces these: • CLI: **show platform security integrity log secure-boot status** command. • YANG Data Model: `Cisco-IOS-XR-attestation-agent-oper.yang` Cisco native model (see GitHub)
Secure Boot on NCS-57D2-18DD-SYS	Release 7.8.1	You can ensure that the code that executes on Cisco routers is authentic and unmodified. Cisco hardware-anchored secure boot feature protects the microloader, the first piece of code that boots up, in a tamper-resistant hardware. This functionality thereby establishes a root of trust that helps to prevent Cisco routers from executing tainted network software. This feature is now extended to the following variant of Cisco NCS 5700 Series Router: • NCS-57D2-18DD-SYS
Support for Secure Boot	Release 7.4.1	Support for Secure Boot is now extended to the following Cisco NCS 540 router variant: • N540-24Q8L2DD-SYS

How trusted software boots

Describes the secure boot sequence for Cisco-signed boot artifacts, explaining how digital signature verification ensures only trusted software components are executed during system initialization.

Booting the System with Trusted Software

Summary

The key components involved in the process are:

- Trusted Anchoring Module (TAm): Provides certificates used to verify signatures on boot artifacts.

- Bootloader and BIOS/UEFI: Initiates the hardware and verifies initial signatures on boot artifacts.
- Kernel and Initrd: Loads operating system components and imports certificate keys for runtime verification.
- Signature verification infrastructure: Validates digital signatures using Cisco RSA 2048-bit keys and SHA-256 hashes.

Trusted software boot ensures system integrity by verifying each boot component against a Cisco-signed digital signature. This process prevents unauthorized software from running during the boot sequence, safeguarding the device and its runtime environment.

Workflow

The process involves these stages.

1. Verification of boot artifacts: At boot time, the system checks all boot artifacts using certificates stored in Cisco's dbCisco or dbxCisco UEFI stores, anchored in TAm.
2. Bootloader and package verification: The bootloader (including GRUB, PXE, netboot) and initial RAM disk (Initrd) are verified and executed if their signatures are valid.
3. Kernel launch and service startup: The kernel launches, followed by the init process, then all required services. RPM packages are installed only after signature verification.
4. Signature verification on installation: When installing Cisco IOS XR software (from disk or via GISO), each RPM package containing system boot components undergoes signature verification. BIOS updates use capsule updates for authenticated changes.
5. Detailed boot verification sequence:
   - TAm verifies signatures on the bootloader (BIOS/PXE) using hash-based methods like LDWM and SHA-256.
   - The bootloader verifies GRUB and configuration files using Cisco RSA 2048-bit keys and SHA-256.
   - GRUB checks detached signatures (SHA-256) on configuration files.
   - GRUB verifies the kernel and Initrd signatures.
   - The kernel and Initrd fetch and import certificates from TAm into their respective keyrings.
   - The kernel verifies module signatures and IMA signatures when binaries are loaded or executed.
   - Digital signatures for grub.cfg, kernel modules, and IMA files are validated using Cisco RSA 2048-bit public keys.

Result

The device boots only after all artifact verifications succeed, ensuring that only Cisco-signed and authenticated software components run in the Cisco IOS XR runtime environment. If any signature verification fails, the boot process is halted and the image is rejected.

How secure iPXE servers work

Describes how the router authenticates a secure iPXE server before downloading a network boot image.

Before downloading a network boot image from the iPXE server, the Cisco router must verify the authenticity and security of the server using certificates.

Summary

The key components involved in the process are:

- iPXE server: Acts as an HTTP image repository, discovered via DHCP, and supports HTTPS with self-signed certificates.
- Cisco router: Discovers and authenticates the iPXE server prior to downloading boot images.
- Certificate infrastructure: Includes the root certificate chain and the Simple Certificate Enrollment Protocol (SCEP) for exchanging certificates.

Secure iPXE servers enable routers to perform certificate-based authentication before downloading network boot images, ensuring a trusted boot process.

Workflow

These stages describe how secure iPXE servers work.

1. Discovery: The Cisco router locates the iPXE server via DHCP.
2. Authentication preparation: The router downloads the iPXE server's self-signed certificates.
3. Certificate verification: Using SCEP, the router acquires the root certificate chain and checks if it is self-signed to validate trust.
4. Authentication: The root certificate chain is used to authenticate the iPXE server.
5. Channel establishment: Once authentication succeeds, a secure HTTPS channel is established between the router and the iPXE server.
6. Image download: The router can now download Bootstrapper protocol (Bootp), ISO, binaries, and scripts across the secure channel.

Result

The router downloads the network boot image only after successfully authenticating the iPXE server using certificate-based verification.

Verify the secure boot status of your router

Verify that your Cisco router booted securely with an authentic Cisco software image and confirm system integrity.

Ensure the router's security integrity by checking secure boot status and verifying that only authorized Cisco software images are running.

Secure boot validation confirms that the router's startup process is protected and the system has not been tampered with or corrupted. Only supported Cisco IOS XR platforms offer secure boot verification.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

Use the **show platform security integrity log secure-boot status** to verify the secure boot status.

Example

```
Router# show  platform security integrity log secure-boot status
Wed Aug 10 15:39:17.871 UTC

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Secure Boot Status: Enabled
Router#
```

Use the **show platform security integrity log secure-boot status** command to verify the secure boot status of the router. If the router boots securely, then the **show** command output displays the status as "Secure Boot Status: Enabled". If the router does not support this secure boot verification functionality, then the status is displayed as "Not Supported".

If the software image was tampered with, then the secure boot fails and the router does not start. The system displays corresponding error logs at various stages of boot process. For example,

Example

```
Bad signature file...
/initrd.img verification using Pkcs7 signature failed.
error: Security Violation: /initrd.img failed to load.
System halting...
```

The router's secure boot status is confirmed. If "Enabled" is shown, your device booted securely with an authentic Cisco software image. If secure boot failed, the router does not start, and error logs indicate security violations.

How trustworthiness works

Outlines the implementation of trustworthiness within systems, describing operational processes and providing verification steps for trust enforcement status at boot.

The following sequence of events takes place automatically when the Cisco routers that support the IOS XR7 operating system are powered up:

Summary

The key components involved in the process are:

- Micro-loader and Trusted Anchor module (TAm): Begin hardware-level signature verification for BIOS and store keys for subsequent checks.
- BIOS and Bootloader: Verify firmware and operating system components, log measurements, and extend integrity checks.
- Integrity Measurement Architecture (IMA) kernel hooks: Validate executable and file signatures using certificates, ensuring runtime integrity.

Trustworthiness on Cisco routers is established using a chain of cryptographic verifications and secure storage mechanisms, starting at hardware boot and continuing throughout device runtime.

Workflow

These stages describe how trustworthiness works.

1. At power-up, the micro-loader in the chip verifies the digital signature of the BIOS using keys stored in the Trusted Anchor module (TAm). The verification is logged, and the measurement is extended into a Platform Configuration Register (PCR).
2. The BIOS verifies the bootloader signature using TAm keys, logs verification, and extends the measurement into the PCR.
3. Upon successful validation, the BIOS launches the bootloader. The bootloader uses keys provided by BIOS to verify the kernel, initial RAM disk (initrd) file system, and grub-config file. All verification operations are logged, and PCR is extended.
4. The bootloader loads initrd to create the initial file system.
5. The kernel launches and populates kernel keyrings with keys from TAm.
6. The init process launches. Whenever an executable or shared library is run, the IMA kernel hook validates the signature using the certificates in IMA keyring, confirming file integrity.
7. The Cisco IOS XR7 RPM is installed with signed verification. RPM verification results are logged. Cisco IOS XR7 processes launch with IMA measurement.
8. TAm services are launched.
9. Cisco IOS XR7 application runs initial admin user configuration and stores credentials in TAm secure storage. Manual provisioning of user credentials concludes the process.

Result

Trust is established from hardware boot, maintained through network operating system verification, and upheld throughout runtime, ensuring the integrity of Cisco routers and their operating environments.

Verify trust implementation boot status

Verify the secure boot integrity log to confirm trust implementation after system boot.

Ensure that the router's secure boot process has completed successfully and trust mechanisms remain enabled.

Before you begin

Use this task to confirm that Cisco routers maintain hardware and software integrity from boot through runtime by validating the secure boot status. This verification helps confirm compliance with trust requirements and ensures system integrity.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

Display the secure boot integrity log.

Example

```
Router# show platform security integrity log secure-boot
Fri Apr 12 17:13:43.867 UTC

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Secure Boot Status: Enabled
```

The output displays "Secure Boot Status: Enabled," confirming that secure boot is active and trust has been implemented correctly.

Key security terms for trustworthy systems

Provides key security terms and definitions crucial for understanding trustworthy systems, serving as a reference for foundational concepts and terminology used throughout the chapter.

The following terms are commonly used in the trustworthy systems:

- Attestation: Mechanism used to confirm the software's integrity. The verifier trusts the attested data because it is signed by a TPM whose key is certified by the Certificate Authority (CA).
- Attestation Identity Key (AIK): A restricted key used for signing attestation requests.
- Bootloader: Code that runs before any operating system, capable of loading the OS kernel and providing commands for debugging or modifying the kernel environment.
- Certificates and Keys in Trust Anchor module (TAm):
  - Image signing certificate: X.509 certificate for validating signatures of GRUB, initrd, kernel, and kernel modules.
  - IOS-XR Key: Public key certificate signed by the Key Enrollment Key (KEK), common to all Cisco 8000 Series routers, used for signing GRUB, initrd, kernel, and kernel modules.
  - RPM key: Used for signing RPM packages.

  - IMA public key certificate: Used to validate Integrity Measurement Architecture (IMA) signatures of files.
  - BIOS or firmware capsule update key: Used to sign the outer capsule for BIOS or firmware updates; same as the secure boot key.
  - Platform key (PK) and Key Enrollment Key (KEK): Public keys and certificates for managing other keys in the TAm.
  - LDWM Key: Stored in the hardware trust anchor module in Cisco IOS XR7, used for validating BIOS.
- Golden ISO (GISO): Image containing the base Linux distribution artifact (ISO), packages, and configuration files serving as a baseline across all servers. The Cisco IOS XR7 GISO image includes IOS XR RPMs, third-party RPMs, ztp.ini, and secure ZTP certificates.
- GRand Unified Bootloader (GRUB): Boot loader package that loads the kernel, supports multiple operating systems, and starts at system boot.
- Hash Function: Function used to map data of arbitrary size onto data of fixed size.
- Initramfs: Set of directories bundled into a compressed cpio archive, loaded at boot time by the boot loader along with the kernel.
- initrd (initial RAM disk): Initial root file system mounted before the real root file system becomes available; loaded as part of the kernel boot procedure.
- JTAG: Hardware interface enabling communication with chips on a board for debugging, programming, and testing embedded devices.
- Nonce Value: Arbitrary number used only once in cryptographic communications, preventing replay attacks by ensuring uniqueness.
- Platform Configuration Register (PCR): Shielded register/memory region holding hash operations; initialized to a known value at power-up and typically cannot be reset.
- PCR Extend: Operation for updating PCR value:

  PCR[x]new = hash (PCR[x]old || hash (measurement value))
- Trust Anchor module (TAm): Cisco hardware module that verifies device authenticity and provides additional security services.
- Trusted Platform Module (TPM): Specialized chip storing RSA encryption keys; used for hardware authentication and key generation based on Endorsement Key and owner-specified password.
- Root of Trust for Storage: TPM 2.0-compliant PCRs form the Root of Trust for Storage.

TCG-compliant trusted platform module

Describes the TCG-compliant trusted platform module, presenting operational guidelines, associated error messages, benefits, and step-by-step instructions for verifying TPM status and certificates.

A TCG-compliant trusted platform module is a hardware security feature that

- adheres to Trusted Computing Group (TCG) specifications
- uses strong cryptographic algorithms such as ECC P384 algorithm for secure identification and attestation, and
- protects device boot integrity and secure communication through hardware-backed security features.

Additional reference information

These hardware-backed security features enable the creation of Initial Device Identifier (IDevID) and Initial Attestation Keys (IAK), supporting compliance with modern security standards such as TLS 1.3. They also provide resilient fallback mechanisms, including AIKIDO-based keys, if ECC is not provisioned in the TPM or TPM hardware is unavailable. Attestation capabilities are enhanced through new PCR 0-8 measurements. TPM support now includes a SHA-384 bank of PCRs for measured boot, further strengthening trust in early boot stages.

IDevID is a factory-installed digital certificate that uniquely and securely identifies a device, typically using its public key and manufacturer-signed credentials. It authenticates devices when they first connect to a network, ensuring that only genuine, trusted hardware is allowed access. This mechanism is commonly applied in secure onboarding, supply chain security, and zero-touch provisioning scenarios.

Table 4: Feature History Table

Feature Name	Release Information	Feature Description
TCG-compliant Trusted Platform Module	Release 25.4.1	Introduced in this release on: Modular Systems (8800 [LC ASIC: Q100, Q200, P100])* Device security and cryptographic identity are enhanced through new support for a TCG-compliant Trusted Platform Module (TPM) with ECC P-384 key pairs. This feature enables TLS 1.3-based secure communication for critical onboarding workflows such as BootZ and sZTP. Enrolment and attestation are strengthened through TPM PCR 0-8 measurements, establishing a hardware-rooted chain of trust from early boot stage. CLI: • **show platform security tpm** • **show platform security tpm PCR** • **show platform security tpm integrity log boot** • **show platform security tpm info ECC-sudi-certs** • **show platform security tpm attest certificate** *This feature is only supported on Cisco 8800-RP2-S hardware variants.

Starting Cisco IOS XR Software Release these features are migrated to TPM:

- gRPC Extensible Manageability Services Daemon (EMSD)
- Attestation APP
- EnrollZ/AttestZ

Guidelines for TCG-compliant Trusted Platform Module

Outlines platform support and automatic enablement guidelines for TPM.

Follow these guidelines to ensure secure and reliable use of TCG-compliant Trusted Platform Module feature:

- The TCG-compliant Trusted Platform Module (TPM) feature is only supported on Cisco 8000 Series RP2-S variants.
- TPM enablement occurs automatically on supported Cisco 8000 Series RP2-S hardware variants; there are no manual steps required to activate core TPM functionality.
- This feature remains permanently enabled and cannot be disabled on future RP2-S hardware that supports TPM.
- The system gracefully falls back to the AIKIDO module if the TCG-compliant IDevID is not present on Generic TPM, maintaining the operation of critical features such as AttestZ and EnrollZ.
- BIOS and PCR measurements follow TCG 2.0 specifications to meet industry-standard trusted computing requirements.
- Applications including EMSD, EnrollZ, and AttestZ are configured to automatically use TPM ECC-based keys and certificates whenever TPM is available.
- The implementation adheres to Cisco's Security Development Lifecycle (CSDL) standards, encompassing threat modeling, secure coding practices, static analysis, and vulnerability testing.
- For measurement and PCR extension within TPM, the SHA-384 algorithm is used through the dedicated SHA-384 PCR bank.

TPM usage error messages

Lists TPM usage error messages and reasons.

Review these error messages during TPM usage.

Table 5: Error Messages during TPM

Error Messages	Reason
No TCG ECC IDevID certificate	The certificate is not provisioned.
No TCG ECC IDevID certificate chain	The certificate chain is missing.
No TCG ECC HW CAPABILITY	The TPM lacks ECC capability.

Benefits of TCG-compliant Trusted Platform Module

Lists benefits of migrating from RSA to TCG-compliant ECC P384 with TPM.

TCG-compliant Trusted Platform Module provide several benefits for device identity and attestation.

- When you migrate from RSA to TCG-compliant ECC P384 algorithm with TPM, you improve protection against cryptographic attacks and tampering. This process significantly strengthens device security.
- Using TCG-compliant ECC P384 algorithm for device identity helps you create robust and unique cryptographic identification. Detailed PCR measurements improve attestation, ensuring trusted onboarding and secure operations.
- You can use ECC keys from TPM to adopt TLS 1.3 for security features such as EMSD, and gRPC. TLS 1.3 provides advanced security and faster communications.
- If the TPM is unavailable, the system uses AIKIDO to maintain operation and ensure service continuity for security-critical functions. However, if TAM supports only RSA-based keys, the device can negotiate only TLS 1.2.

- Use new **show** commands to inspect the TPMs state, PCR values, attestation data, and certificate chains. These actions enhance security transparency for your audit and troubleshooting procedures.

Verify TPM status and certificates

Verify that the Trusted Platform Module (TPM) is enabled, functioning as expected, and properly configured on your device to ensure platform integrity and compliance.

Ensure that TPM is active on your device and has the expected cryptographic information for secure operation.

TPM provides hardware-based security and attestation for Cisco routers. Verifying its status, certificates, and integrity measurements is required for compliance and troubleshooting.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

1. Display TPM security information.

Example

```
Router# show platform security tpm info all location 0/RP0/CPU0
-------------------------------------------
Node - 0/RP0/CPU0
-------------------------------------------
Device Type            -     TPM
Device PID             -     8800-RP2-S
Device Serial Number   -     FOC2845N1BJ
Device Firmware Version-     0x01.0200
Server Version         -     3
Server Package Version -     13.1.1
Client Package Version -     13.1.1

HA Sudi Root Cert:
-------------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: ecdsa-with-SHA384
        Issuer: O=Cisco, CN=Cisco ECC Root CA
        Validity
            Not Before: Apr  4 08:15:44 2013 GMT
            Not After : Sep  7 16:24:07 2099 GMT
        Subject: O=Cisco, CN=Cisco ECC Root CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:7e:c0:c3:bc:d8:1b:c6:f3:67:91:4d:d6:78:8c:
                    e6:b5:75:39:04:7f:2f:fe:60:d0:ac:77:2a:d3:6d:
                    02:41:45:54:67:b0:58:b7:19:bf:cc:bd:4b:36:5c:
                    7b:5b:83:38:ec:a6:d7:4d:30:26:61:b3:4b:8b:ab:
                    5e:0e:15:26:3b:4c:88:ab:02:70:c9:22:37:02:50:
                    75:c0:d5:d4:48:34:c7:bf:58:53:fe:ae:cb:8f:73:
                    20:f5:06:5b:12:87:ca
                ASN1 OID: secp384r1
                NIST CURVE: P-384
```

```
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                A4:45:B6:2F:A3:31:B1:76:15:B0:0A:18:33:CA:F6:AD:4F:3D:28:04
    Signature Algorithm: ecdsa-with-SHA384
    Signature Value:
        30:65:02:31:00:f0:9b:71:dc:7b:40:56:9f:ff:61:a9:3a:ba:
        6d:f1:19:10:44:4f:88:f9:80:ff:a5:86:6f:13:89:c8:19:75:
        39:9e:0e:b6:e8:85:bb:9f:06:0d:07:31:11:ba:80:57:2b:02:
        30:08:b9:e0:52:29:8f:89:14:84:28:c7:27:80:1d:98:73:ea:
        97:2c:6b:31:e3:84:b7:ac:48:b1:d6:54:d5:49:35:54:ca:66:
        27:8f:7a:f6:e2:b1:1e:38:ab:8a:a5:f3:86
```

The command output is truncated.

2. View the TPM integrity boot log.

Example

```
Router# show platform security tpm integrity log boot location 0/rp0/CPU0

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Uptime: 504480
Event Number: 1
Event Type: 03
PCR Index: 0
Event Digest Hash Algorithm: SHA1
Event Digest: AAAAAAAAAAAAAAAAAAAAAAAAAAA=
Event Data: U3BlYyBJRCBFdmVudDAzAAAAAAAAAgACAgAAAAsAIAAMADAAAA==
Event Number: 2
Event Type: 80000008
PCR Index: 0
Event Digest Hash Algorithm: SHA256
Event Digest: 8emdTrXnHLbHk7wXjWIMkF3/rKTUx7aJ7bFbqw5IEv4=
Event Digest Hash Algorithm: SHA384
Event Digest: 6OuqsUTGphX/NRZc82MBOefq1uLmac7i+kCjzmF5emFth2Fvd8NIAY4JruOOQMLM
Event Data: MrVWeEdXauXu7dTksv0FWh/3qy9lCrxrKWRy/2iR+xc=
Event Number: 3
Event Type: 80000008
PCR Index: 0
Event Digest Hash Algorithm: SHA256
Event Digest: mlrPXdkLmp++U11lPwBJZpnJ/qjOpVTxQljh8548Qxs=
Event Digest Hash Algorithm: SHA384
Event Digest: ioW380j8Ujk+EbQCX2Rz3OpJrjkh0ICSOCaJcm6cPdp0m05yma6tAqOk9FXCH145
Event Data: AEtoSI8ICqnAz53G1/XMD4qHH1qkuXmEAhLEUpw5ON4=
Event Number: 4
Event Type: 08
```

The command output is truncated.

3. Check TPM status and take necessary actions.

Table 6: TPM verification decisions and actions

If...	Then...
TPM is shown as enabled and active	no further action is required.

If...	Then...
TPM is not present or status is not enabled	initiate TPM troubleshooting procedures.
Digest algorithm output is not SHA384 as expected	verify configuration and consult security policies.
Integrity log shows unexpected entries	investigate potential security incidents.

4. Display Platform Configuration Register (PCR) information.

Example

```
Router# show platform security tpm pcr 0,1,2,3,4,5,6,7 location 0/rp0/CPU0
nonce 1234
Nonce: 1234

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Uptime: 503962
pcr-quote:
/[illegible]
pcr-quote-signature:
MGUEyTef32ivqM9y3RhtjG5fNVlgwIgvAOnVa1cRrK6Z4FqxiLFU5wfZQIxARXhkpE/OsiUvN6/Eyr4jrU8a69a+6ei3V33mUzqCv9/wojxBKRmRQ==
pcr-index       pcr-value
  0     jkkHIFe0ce11YQouajSuN0jharqmSlK8dIr4j9TxXDb0PNVE9DjqO8vAh4tqHrl2
  1     wqoQkOVvOepm0+UwHGMmZwOYQl+B1R3TAaCnM3N5spA47EMWEI8cJDv77BvU02Ep
  2     UYkjsPlV0I2gd8lqq6Uiud7O3mHFmc6mxBiJz76krk1QUp2W/k0a/a+2Xn+VvyPE
  3     UYkjsPlV0I2gd8lqq6Uiud7O3mHFmc6mxBiJz76krk1QUp2W/k0a/a+2Xn+VvyPE
  4     lPFWW2vHNA6NX8Iv0J8wWWaT3rAXnNZeRKLtc7LAoEjjNNgX7IxPdyn6WjKBkVP1
  5     TLhFkU36DYM3c75lQ6xSF3C60ebOb4fFGpbaDUuO796OhhJ8dCM69rwNeGxGvGhO
  6     UYkjsPlV0I2gd8lqq6Uiud7O3mHFmc6mxBiJz76krk1QUp2W/k0a/a+2Xn+VvyPE
  7     OoL7f6vUtdwPenkABQg01O0wmynHogAEhtV/YkxizN4Po0CZrfkQfXJf9P6C6h+Y
```

The PCR bank is based on SHA384.

Confirm PCR values reflect expected integrity measurements.

5. Display the TPM Cisco IdevID certificate.

Example

```
Router# show platform security tpm attest certificate CiscoECCSUDI location
+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Certificate name: Cisco ECC SUDI Root
-----BEGIN CERTIFICATE-----
MIIByDCCAU6gAwIBAgIBAzAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgxNTQ0WhgPMjA5
OTA5MDcxNjI0MDdaMCwxDjAMBgNVBAoTBUNpc2NvMRowGAYDVQQDExFDaXNjbyBF
Q0MgUm9vdCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABH7Aw7zYG8bzZ5FN1niM
5rV1OQR/L/5g0Kx3KtNtAkFFVGewWLcZv8y9SzZce1uDOOym100wJmGzS4urXg4V
JjtMiKsCcMkiNwJQdcDV1Eg0x79YU/6uy49zIPUGWxKHyqNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFKRFti+jMbF2FbAKGDPK
9q1PPSgEMAoGCCqGSM49BAMDA2gAMGUCMQDwm3Hce0BWn/9hqTq6bfEZEERPiPmA
/6WGbxOJyBl1OZ4OtuiFu58GDQcxEbqAVysCMAi54FIpj4kUhCjHJ4AdmHPqlyxr
MeOEt6xIsdZU1Uk1VMpmJ4969uKxHjiriqXzhg==
-----END CERTIFICATE-----
-----------------Truncated for brevity-----------------
```

Validate certificate information as required.

6. Display TPM ECC-sudi certs information.

Example

```
Router# show platform security tpm attest certificate CiscoECCSUDI location
0/rp0/CPU0 nonce 1234
+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Nonce: 1234
Certificate name: Cisco ECC SUDI Root
-----BEGIN CERTIFICATE-----
MIIByDCCAU6gAwIBAgIBAzAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgxNTQ0WhgPMjA5
OTA5MDcxNjI0MDdaMCwxDjAMBgNVBAoTBUNpc2NvMRowGAYDVQQDExFDaXNjbyBF
Q0MgUm9vdCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABH7Aw7zYG8bzZ5FN1niM
5rV1OQR/L/5g0Kx3KtNtAkFFVGewWLcZv8y9SzZce1uDOOym100wJmGzS4urXg4V
JjtMiKsCcMkiNwJQdcDV1Eg0x79YU/6uy49zIPUGWxKHyqNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFKRFti+jMbF2FbAKGDPK
9q1PPSgEMAoGCCqGSM49BAMDA2gAMGUCMQDwm3Hce0BWn/9hqTq6bfEZEERPiPmA
/6WGbxOJyBl1OZ4OtuiFu58GDQcxEbqAVysCMAi54FIpj4kUhCjHJ4AdmHPqlyxr
MeOEt6xIsdZU1Uk1VMpmJ4969uKxHjiriqXzhg==
-----END CERTIFICATE-----

Certificate name: Cisco ECC SUDI CA
-----BEGIN CERTIFICATE-----
MIIDETCCApagAwIBAgIBBTAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgyNjEzWhgPMjA5
OTA5MDcxNjI0MDZaMCsxDjAMBgNVBAoTBUNpc2NvMRkwFwYDVQQDExBBQ1QyIEVD
QyBTVURJIENBMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEjGHc+nnSMQr4zuARzDxH
F6TDi97f7eGqSdDK5misVpoBP0Nwwcmazs+GTiT+Us52/BKuo4JlaZ74cnlzzBIn
kPwtuag2bXSCeYVJhcB3sVuVvYiSKVEi1OIgx85dtXdwo4IBiTCCAYUwDgYDVR0P
AQH/BAQDAgEGMBIGA1UdEwEB/wQIMAYBAf8CAQAwfQYIKwYBBQUHAQEEcTBvMD8G
CCsGAQUFBzAChjNodHRwOi8vd3d3LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kvY2Vy
dHMvZWNjcm9vdC5jZXIwLAYIKwYBBQUHMAGGIGh0dHA6Ly9wa2ljdnMuY2lzY28u
Y29tL3BraS9vY3NwMB8GA1UdIwQYMBaAFKRFti+jMbF2FbAKGDPK9q1PPSgEMFwG
A1UdIARVMFMwUQYKKwYBBAEJFQETADBDMEEGCCsGAQUFBwIBFjVodHRwOi8vd3d3
LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kvcG9saWNpZXMvaW5kZXguaHRtbDBCBgNV
HR8EOzA5MDegNaAzhjFodHRwOi8vd3d3LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kv
Y3JsL2VjY3Jvb3QuY3JsMB0GA1UdDgQWBBSWhzrYiYGRQRUzv+A0jyCPwrvDljAK
BggqhkjOPQQDAwNpADBmAjEAzye9v1h5m/1bAUSU0MQwShDDCRJHXsc+1TtbqOJR
1cZMmhsIzNNy/AskkBoIgNNtAjEAk3DAKxkBx3DYyZS4Uz3B50MpbrZwpxPTUCqW
m7qExCM7m1FjIg98ltfdltPHgqP/
-----END CERTIFICATE-----

Certificate name: Cisco ECC SUDI
-----BEGIN CERTIFICATE-----
MIIDYDCCAuagAwIBAgIKB3VSGUcUMEFBaTAKBggqhkjOPQQDAzArMQ4wDAYDVQQK
EwVDaXNjbzEZMBcGA1UEAxMQQUNUMiBFQ0MgU1VESSBDQTAgFw0yNTA1MjMxNjMy
MjJaGA8yMDk5MDkwNzE2MjQwNVowgaAxJjAkBgNVBAUTHVBJRDo4ODAwLVJQMi1T
IFNOOkZPQzI4NDVOMUJKMQ4wDAYDVQQKEwVDaXNjbzERMA8GA1UECxMIVFBNIFNV
REkxUzBRBgNVBAMTSkNpc2NvIDg4MDAgUm91dGUgUHJvY2Vzc29yIDIgKHNaVFAg
ZGVmYXVsdCkgTUFDOmM0YWI0ZGZmODNjMC0wMDA4ODAwLVJQMi1TMHYwEAYHKoZI
zj0CAQYFK4EEACIDYgAEDXJ5EgjisuoVF9OmqmdPkdSDhm48bwqcsnv4Yb+LTHDE
s84Mdj44a1VIKUEZyVaMemNCu/IdO7TgNcLoizCulb4BxiVIulv4oeL2kh23SxK1
BfW5I5FoxkLz4QGr9fTxo4IBWzCCAVcwDgYDVR0PAQH/BAQDAgXgMAwGA1UdEwEB
/wQCMAAwHwYDVR0jBBgwFoAUloc62ImBkUEVM7/gNI8gj8K7w5YwHQYDVR0gBBYw
FDAIBgZngQULAQEwCAYGZ4EFCwECMIHXBgNVHREEgc8wgcygcQYIKwYBBQUHCASg
ZTBjBgVngQUBAgRaNTM1NDREMjA6NTNkZWM1MmE0YjgwYTczYzViOWJjMDY5ZWM0
MDc5MTZiZjNiMzM4MDoxNmI5ZjllYTIxMTQxZmY4NWMwMjAzY2ZhODEwYzg5NzEw
```

```
YjRiOWQ0oFcGCCsGAQUFBwgDoEswSQxAODc5ZWYwMjUxNTlkYTVlMzUzZWEyZTNi
MDhjZGM0MjAzZDUyNzAxNTJjMThlYmRlYjA4Yjc5MzMxYTA3ODQ0ZAYFZ4EFDAEw
HQYDVR0OBBYEFEUV02BniQxfiRTdQVzgfjZh0aD/MAoGCCqGSM49BAMDA2gAMGUC
MQC23df2vMVL9J20W22NzeeBJvU2+awcI10kfcZYSr+SLA+BTHU/iLyUIatZpqcv
Y7UCMA3xchuT66IPNfkrjZoPSQARUP0376dirxOiFSwcedUGVDxzMxObGb3Lh68H
hpBtQA==
-----END CERTIFICATE-----
```

Review details for completeness and correctness.

7. Display the TPM IAK certificate.

Example

```
Router# show platform security tpm attest certificate CiscoECCIAK location
0/rp0/CPU0 nonce 1234
+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Nonce: 1234
Certificate name: Cisco ECC IAK Root
-----BEGIN CERTIFICATE-----
MIIByDCCAU6gAwIBAgIBAzAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgxNTQ0WhgPMjA5
OTA5MDcxNjI0MDdaMCwxDjAMBgNVBAoTBUNpc2NvMRowGAYDVQQDExFDaXNjbyBF
Q0MgUm9vdCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABH7Aw7zYG8bzZ5FN1niM
5rV1OQR/L/5g0Kx3KtNtAkFFVGewWLcZv8y9SzZce1uDOOym100wJmGzS4urXg4V
JjtMiKsCcMkiNwJQdcDV1Eg0x79YU/6uy49zIPUGWxKHyqNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFKRFti+jMbF2FbAKGDPK
9q1PPSgEMAoGCCqGSM49BAMDA2gAMGUCMQDwm3Hce0BWn/9hqTq6bfEZEERPiPmA
/6WGbxOJyBl1OZ4OtuiFu58GDQcxEbqAVysCMAi54FIpj4kUhCjHJ4AdmHPqlyxr
MeOEt6xIsdZU1Uk1VMpmJ4969uKxHjiriqXzhg==
-----END CERTIFICATE-----

Certificate name: Cisco ECC IAK CA
-----BEGIN CERTIFICATE-----
MIIDIzCCAqmgAwIBAgIJBEje0ku4AXhYMAoGCCqGSM49BAMDMCwxDjAMBgNVBAoT
BUNpc2NvMRowGAYDVQQDExFDaXNjbyBFQ0MgUm9vdCBDQTAgFw0yNDAxMTAxOTM3
NDdaGA8yMDk5MDExMDE5Mzc0N1owQTEOMAwGA1UEChMFQ2lzY28xDjAMBgNVBAsT
BVJvYnVyMR8wHQYDVQQDExZFQyBQMzg0IEF0dGVzdGF0aW9uIENBMHYwEAYHKoZI
zj0CAQYFK4EEACIDYgAEOyud77njuBar/GPcZtH7TIai4XAY7B6hftiadq3oj7+e
xGeAqes3K3IMVE17TNDkTLVTNoqIKTV50g6noezuXsOD/9zEwll2fsbzuzt1KFVr
bop5Pyq7HK5PI+dmRFjgo4IBfjCCAXowDgYDVR0PAQH/BAQDAgEGMBIGA1UdEwEB
/wQIMAYBAf8CAQAwfQYIKwYBBQUHAQEEcTBvMD8GCCsGAQUFBzAChjNodHRwOi8v
d3d3LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kvY2VydHMvZWNjcm9vdC5jZXIwLAYI
KwYBBQUHMAGGIGh0dHA6Ly9wa2ljdnMuY2lzY28uY29tL3BraS9vY3NwMB8GA1Ud
IwQYMBaAFKRFti+jMbF2FbAKGDPK9q1PPSgEMFEGA1UdIARKMEgwRgYJKwYBBAEJ
FQFVMDkwNwYIKwYBBQUHAgEWK2h0dHA6Ly93d3cuY2lzY28uY29tL3NlY3VyaXR5
L3BraS9wb2xpY2llcy8wQgYDVR0fBDswOTA3oDWgM4YxaHR0cDovL3d3dy5jaXNj
by5jb20vc2VjdXJpdHkvcGtpL2NybC9lY2Nyb290LmNybDAdBgNVHQ4EFgQUQuOx
5As0KbV+3dYMwPGomM/QrKEwCgYIKoZIzj0EAwMDaAAwZQIwWlteUTHSxAe6d3Gp
T9E1HZGX+oAQSijo/dPWSkhdbchO+Fe6M9bzjb+gvaUt/EBDAjEAoDLC0H4/JW3i
6jElqnEYqD6GWiiQALvz0Zop6JI897nD/WJvsjnPGgMfxtNDfq3H
-----END CERTIFICATE-----

Certificate name: Cisco ECC IAK
-----BEGIN CERTIFICATE-----
MIIDdzCCAvygAwIBAgIKBWSFFBERQYNRaTAKBggqhkjOPQQDAzBBMQ4wDAYDVQQK
EwVDaXNjbzEOMAwGA1UECxMFUm9idXIxHzAdBgNVBAMTFkVDIFAzODQgQXR0ZXN0
YXRpb24gQ0EwIBcNMjUwNTIzMTYzMjIyWhgPMjA5OTAxMTAxOTM3NDZaMIGgMSYw
JAYDVQQFEx1QSUQ6ODgwMC1SUDItUyBTTjpGT0MyODQ1TjFCSjEOMAwGA1UEChMF
```

```
Q2lzY28xETAPBgNVBAsTCFRQTSBTVURJMVMwUQYDVQQDE0pDaXNjbyA4ODAwIFJv
dXRlIFByb2Nlc3NvciAyIChzWlRQIGRlZmF1bHQpIE1BQzpjNGFiNGRmZjgzYzAt
MDAwODgwMC1SUDItUzB2MBAGByqGSM49AgEGBSuBBAAiA2IABInu07M6N/F10hLw
A4KsafCGzqkeft/5nR3awykGZfzRvOAZC3U6mnXniJezgVoEATTnGFO6sQZZSMFO
xrBDdLmNkMlY7Neq/8pC/rqDoFGJi+oD4jOSdTD+Lgj+3XfRi6OCAVswggFXMA4G
A1UdDwEB/wQEAwIF4DAMBgNVHRMBAf8EAjAAMB8GA1UdIwQYMBaAFELjseQLNCm1
ft3WDMDxqJjP0KyhMB0GA1UdIAQWMBQwCAYGZ4EFCwEBMAgGBmeBBQsBAzCB1wYD
VR0RBIHPMIHMoHEGCCsGAQUFBwgEoGUwYwYFZ4EFAQIEWjUzNTQ0RDIwOjUzZGVj
NTJhNGI4MGE3M2M1YjliYzA2OWVjNDA3OTE2YmYzYjMzODA6MTZiOWY5ZWEyMTE0
MWZmODVjMDIwM2NmYTgxMGM4OTcxMGI0YjlkNKBXBggrBgEFBQcIA6BLMEkMQDg3
OWVmMDI1MTU5ZGE1ZTM1M2VhMmUzYjA4Y2RjNDIwM2Q1MjcwMTUyYzE4ZWJkZWIw
OGI3OTMzMWEwNzg0NGQGBWeBBQwBMB0GA1UdDgQWBBT4wBA1PZL4Nz6AtQ4p/tXV
ZM5MHTAKBggqhkjOPQQDAwNpADBmAjEAlvOCAK8orDACly8G67vmtkPpt3tNL47F
1p6OoHNh0DSn7CNhbonVuslq8cRpZTVbAjEA/YC0I0EGTxRkVquPYtByiJVXHLOX
pUDdcb4/1BilpLgq3Ypaec4lF2v0wqPNZvw3
-----END CERTIFICATE-----
```

TPM is verified as enabled and correctly configured. All relevant certificate and integrity information is confirmed and available for compliance purposes.

ECC P256-based Secure Unique Device Identifiers

Details ECC P256-based Secure Unique Device Identifiers, including cryptographic mechanism comparisons, related error messages, and provides verification procedures for ECC P256 configurations in the Trusted Anchor Module.

A Secure Unique Device Identifier (SUDI) is a hardware-based cryptographic identity that

- leverages the ECC P256 curve for enhanced security,
- enables migration from legacy RSA-based authentication to stronger ECC-based protocols, and
- supports secure services such as TLS 1.3 for BootZ, secure Zero-Touch Provisioning (sZTP), and EMSD.

The router automatically checks for ECC p256 capability during TAM initialization and enables the API if available. If ECC p256 is not provisioned or TAM is unreachable, the system falls back to RSA-based keys from the AIKIDO module, maintaining continuity for critical features.

Feature history

The feature history table lists release support for this feature.

Table 7: Feature History Table

Feature Name	Release info	Description
ECC P256-based Secure Unique Device Identifiers	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: , Q200], (select variants only*) ECC256 enablement enhances device security by migrating identity and attestation workflows to ECCp256-based SUDI via the Trust Anchor Module (TAM). This transition enables the adoption of TLS 1.3 for critical services such as BootZ, secure Zero-Touch Provisioning (sZTP), and EMSD. Operational continuity is ensured through a resilient fallback to legacy RSA-based keys if needed. ECC p256 is automatically enabled when certificates are present; no additional configuration steps are required. *This feature is applicable on: • 8202-32FH-M • 8202-32FH-MO

ECC P256 cryptographic mechanism comparison

Provides a comparison of RSA 2048 and ECC P256 mechanisms, including their security strength, supported protocols, and operational continuity.

The following table lists and compares the key features of RSA 2048 and ECC P256 cryptographic mechanisms, highlighting their security strengths, protocol support, and continuity solutions.

Table 8: Comparison of cryptographic mechanisms and protocol support

Crypto Mechanism	Security Strength	Protocol Support	Continuity
RSA 2048	Moderate	TLS 1.2 (legacy)	Provided by AIKIDO
ECC p256	High	TLS 1.3 (advanced)	Provided by TAM

ECC P256 error messages

Lists ECC P256 certificate and TLS negotiation error messages.

The following error messages may occur when working with ECC P256 certificates or TLS negotiations:

- ECCp256 certificate chain missing: The ECC p256 certificate is not provisioned in TAM.
- ECCp256 capability not enabled: TAM lacks ECC p256 capability or is unreachable.
- TLS 1.3 negotiation failed: Fallback to TLS 1.2 initiated due to RSA-only configuration.

Verify ECC P256 configuration in the Trusted Anchor Module

Verify the presence and validity of ECC P256 SUDI certificates in the Trusted Anchor Module (TAM) to ensure strong authentication and attestation capabilities on your device.

Ensure that the Trusted Anchor Module supports and properly configures ECC P256 SUDI certificates for secure device authentication and compliance with security standards.

ECC P256 SUDI certificates help enable secure device attestation and encrypted communications. Verifying their correct configuration in the Trusted Anchor Module is essential to meet security requirements and to ensure your device operates securely in the network.

Before you begin

- Confirm that your device supports ECC P256 SUDI.

Procedure

1. Display TAM security information.

Example

```
Router# show platform security tam all location 0/RP0/CPU0
Node - 0/RP0/CPU0
-----------------------------------------------
Device Type             -     AIKIDO Extended
Device PID              -     8202-32FH-M
Device Serial Number    -     FLM275101MJ
Device Firmware Version-      0x27.0016
Server Version          -     3
Server Package Version -      13.1.1
Client Package Version -      13.1.1

...
...
...

Sudi Sub CA Cert:
-----------------------------------------------
...
...
...

Sudi Cert:
-----------------------------------------------
Certificate:
...
...
...

HA Sudi Root Cert:
-----------------------------------------------
Certificate:
...
...
...

HA Sudi Cert:
-----------------------------------------------
Certificate:
```

```
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:57:34:23:36:75:41:16:24:a5
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O=Cisco, CN=ACT2 ECC SUDI CA
        Validity
            Not Before: Nov 11 01:08:36 2025 GMT
            Not After : Apr  4 08:15:42 2053 GMT
       Subject: serialNumber=PID:8202-32FH-M SN:FLM275101MJ, O=Cisco, OU=ACT-2
 Lite SUDI, CN=Cisco 8200 2RU 32x400G QSFP56-DD w\/IOS XR HBM MACsec
MAC:70da48df0800-0200
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:08:cb:4c:7c:df:5e:7b:55:d9:50:76:7e:1d:e8:
                    3f:c4:63:86:f2:80:f9:26:65:f0:ef:85:61:d7:d2:
                    7e:1a:a7:8c:06:3f:3a:2d:2b:b7:5d:5c:08:51:c6:
                    b7:3a:72:40:f7:c1:2e:24:54:c6:a0:bb:27:50:b6:
                    78:23:0f:79:a1
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                96:87:3A:D8:89:81:91:41:15:33:BF:E0:34:8F:20:8F:C2:BB:C3:96
            X509v3 Subject Alternative Name:
                othername: 1.3.6.1.4.1.9.21.3.4.2::<unsupported>, othername:
 1.3.6.1.4.1.9.21.2.3::<unsupported>
            X509v3 Subject Key Identifier:
                E7:B7:21:95:3F:0F:F5:BF:F7:05:F5:21:CD:F7:B9:D4:0F:5D:ED:2D
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:65:02:31:00:8f:77:87:7d:3d:e8:b3:72:86:47:04:a2:41:
        ab:52:35:d1:44:9e:1d:ae:cd:e2:b3:8b:13:4c:26:9a:41:f2:
        d4:bb:fc:4b:67:bb:66:77:72:d8:c5:df:9a:f3:71:c7:51:02:
        30:00:bd:cd:7d:1e:fb:c7:2a:e7:21:60:28:5e:de:d8:58:45:
        7a:06:05:e0:54:13:b7:2b:40:34:d5:e1:7e:2c:fd:02:b0:ba:
        ad:62:47:77:26:a5:60:75:b8:14:02:82:af
```

Review the output to ensure TAM is ECC p256 capable.

2. Verify ECC p256 certificate details.

Example

```
Router# show platform security tam ECCp256-sudi-certs location 0/RP0/CPU0
Node - 0/RP0/CPU0
-----------------------------------------------

HA Sudi Root Cert:
-----------------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: ecdsa-with-SHA384
```

```
        Issuer: O=Cisco, CN=Cisco ECC Root CA
        Validity
            Not Before: Apr  4 08:15:44 2013 GMT
            Not After : Sep  7 16:24:07 2099 GMT
        Subject: O=Cisco, CN=Cisco ECC Root CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:7e:c0:c3:bc:d8:1b:c6:f3:67:91:4d:d6:78:8c:
                    e6:b5:75:39:04:7f:2f:fe:60:d0:ac:77:2a:d3:6d:
                    02:41:45:54:67:b0:58:b7:19:bf:cc:bd:4b:36:5c:
                    7b:5b:83:38:ec:a6:d7:4d:30:26:61:b3:4b:8b:ab:
                    5e:0e:15:26:3b:4c:88:ab:02:70:c9:22:37:02:50:
                    75:c0:d5:d4:48:34:c7:bf:58:53:fe:ae:cb:8f:73:
                    20:f5:06:5b:12:87:ca
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                A4:45:B6:2F:A3:31:B1:76:15:B0:0A:18:33:CA:F6:AD:4F:3D:28:04
    Signature Algorithm: ecdsa-with-SHA384
    Signature Value:
        30:65:02:31:00:f0:9b:71:dc:7b:40:56:9f:ff:61:a9:3a:ba:
        6d:f1:19:10:44:4f:88:f9:80:ff:a5:86:6f:13:89:c8:19:75:
        39:9e:0e:b6:e8:85:bb:9f:06:0d:07:31:11:ba:80:57:2b:02:
        30:08:b9:e0:52:29:8f:89:14:84:28:c7:27:80:1d:98:73:ea:
        97:2c:6b:31:e3:84:b7:ac:48:b1:d6:54:d5:49:35:54:ca:66:
        27:8f:7a:f6:e2:b1:1e:38:ab:8a:a5:f3:86

HA Sudi Sub CA Cert:
--------------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 5 (0x5)
        Signature Algorithm: ecdsa-with-SHA384
        Issuer: O=Cisco, CN=Cisco ECC Root CA
        Validity
            Not Before: Apr  4 08:26:13 2013 GMT
            Not After : Sep  7 16:24:06 2099 GMT
        Subject: O=Cisco, CN=ACT2 ECC SUDI CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:8c:61:dc:fa:79:d2:31:0a:f8:ce:e0:11:cc:3c:
                    47:17:a4:c3:8b:de:df:ed:e1:aa:49:d0:ca:e6:68:
                    ac:56:9a:01:3f:43:70:c1:c9:9a:ce:cf:86:4e:24:
                    fe:52:ce:76:fc:12:ae:a3:82:65:69:9e:f8:72:79:
                    73:cc:12:27:90:fc:2d:b9:a8:36:6d:74:82:79:85:
                    49:85:c0:77:b1:5b:95:bd:88:92:29:51:22:d4:e2:
                    20:c7:ce:5d:b5:77:70
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
```

```
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            Authority Information Access:
                CA Issuers -
URI:http://www.cisco.com/security/pki/certs/eccroot.cer
                OCSP - URI:http://pkicvs.cisco.com/pki/ocsp
            X509v3 Authority Key Identifier:
                A4:45:B6:2F:A3:31:B1:76:15:B0:0A:18:33:CA:F6:AD:4F:3D:28:04
            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.9.21.1.19.0
                  CPS: http://www.cisco.com/security/pki/policies/index.html

            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://www.cisco.com/security/pki/crl/eccroot.crl
            X509v3 Subject Key Identifier:
                96:87:3A:D8:89:81:91:41:15:33:BF:E0:34:8F:20:8F:C2:BB:C3:96
    Signature Algorithm: ecdsa-with-SHA384
    Signature Value:
        30:66:02:31:00:cf:27:bd:bf:58:79:9b:fd:5b:01:44:94:d0:
        c4:30:4a:10:c3:09:12:47:5e:c7:3e:d5:3b:5b:a8:e2:51:d5:
        c6:4c:9a:1b:08:cc:d3:72:fc:0b:24:90:1a:08:80:d3:6d:02:
        31:00:93:70:c0:2b:19:01:c7:70:d8:c9:94:b8:53:3d:c1:e7:
        43:29:6e:b6:70:a7:13:d3:50:2a:96:9b:ba:84:c4:23:3b:9b:
        51:63:22:0f:7c:96:d7:dd:96:d3:c7:82:a3:ff

HA Sudi Cert:
---------------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:57:34:23:36:75:41:16:24:a5
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: O=Cisco, CN=ACT2 ECC SUDI CA
        Validity
            Not Before: Nov 11 01:08:36 2025 GMT
            Not After : Apr  4 08:15:42 2053 GMT
       Subject: serialNumber=PID:8202-32FH-M SN:FLM275101MJ, O=Cisco, OU=ACT-2
 Lite SUDI, CN=Cisco 8200 2RU 32x400G QSFP56-DD w\/IOS XR HBM MACsec
MAC:70da48df0800-0200
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:08:cb:4c:7c:df:5e:7b:55:d9:50:76:7e:1d:e8:
                    3f:c4:63:86:f2:80:f9:26:65:f0:ef:85:61:d7:d2:
                    7e:1a:a7:8c:06:3f:3a:2d:2b:b7:5d:5c:08:51:c6:
                    b7:3a:72:40:f7:c1:2e:24:54:c6:a0:bb:27:50:b6:
                    78:23:0f:79:a1
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                96:87:3A:D8:89:81:91:41:15:33:BF:E0:34:8F:20:8F:C2:BB:C3:96
            X509v3 Subject Alternative Name:
                othername: 1.3.6.1.4.1.9.21.3.4.2::<unsupported>, othername:
 1.3.6.1.4.1.9.21.2.3::<unsupported>
```

```
            X509v3 Subject Key Identifier:
                E7:B7:21:95:3F:0F:F5:BF:F7:05:F5:21:CD:F7:B9:D4:0F:5D:ED:2D
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:65:02:31:00:8f:77:87:7d:3d:e8:b3:72:86:47:04:a2:41:
        ab:52:35:d1:44:9e:1d:ae:cd:e2:b3:8b:13:4c:26:9a:41:f2:
        d4:bb:fc:4b:67:bb:66:77:72:d8:c5:df:9a:f3:71:c7:51:02:
        30:00:bd:cd:7d:1e:fb:c7:2a:e7:21:60:28:5e:de:d8:58:45:
        7a:06:05:e0:54:13:b7:2b:40:34:d5:e1:7e:2c:fd:02:b0:ba:
        ad:62:47:77:26:a5:60:75:b8:14:02:82:af
```

Confirm that the leaf certificate and certificate chain are correctly displayed.

3. Perform attestation with ECC p256.

Example

```
Router# show platform security attest certificate CiscoECCp256SUDI nonce 1234
 location 0/RP0/CPU0
 Node location: node0_RP0_CPU0
+-------------------------------------+
Nonce: 1234
Certificate name: Cisco ECC256 SUDI Root
-----BEGIN CERTIFICATE-----
MIIByDCCAU6gAwIBAgIBAzAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgxNTQ0WhgPMjA5
OTA5MDcxNjI0MDdaMCwxDjAMBgNVBAoTBUNpc2NvMRowGAYDVQQDExFDaXNjbyBF
Q0MgUm9vdCBDQTB2MBAGByqGSM49AgEGBSuBBAAiA2IABH7Aw7zYG8bzZ5FN1niM
5rV1OQR/L/5g0Kx3KtNtAkFFVGewWLcZv8y9SzZce1uDOOym100wJmGzS4urXg4V
JjtMiKsCcMkiNwJQdcDV1Eg0x79YU/6uy49zIPUGWxKHyqNCMEAwDgYDVR0PAQH/
BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFKRFti+jMbF2FbAKGDPK
9q1PPSgEMAoGCCqGSM49BAMDA2gAMGUCMQDwm3Hce0BWn/9hqTq6bfEZEERPiPmA
/6WGbxOJyBl1OZ4OtuiFu58GDQcxEbqAVysCMAi54FIpj4kUhCjHJ4AdmHPqlyxr
MeOEt6xIsdZU1Uk1VMpmJ4969uKxHjiriqXzhg==
-----END CERTIFICATE-----

Certificate name: Cisco ECC256 SUDI CA
-----BEGIN CERTIFICATE-----
MIIDETCCApagAwIBAgIBBTAKBggqhkjOPQQDAzAsMQ4wDAYDVQQKEwVDaXNjbzEa
MBgGA1UEAxMRQ2lzY28gRUNDIFJvb3QgQ0EwIBcNMTMwNDA0MDgyNjEzWhgPMjA5
OTA5MDcxNjI0MDZaMCsxDjAMBgNVBAoTBUNpc2NvMRkwFwYDVQQDExBBQ1QyIEVD
QyBTVURJIENBMHYwEAYHKoZIzj0CAQYFK4EEACIDYgAEjGHc+nnSMQr4zuARzDxH
F6TDi97f7eGqSdDK5misVpoBP0Nwwcmazs+GTiT+Us52/BKuo4JlaZ74cnlzzBIn
kPwtuag2bXSCeYVJhcB3sVuVvYiSKVEi1OIgx85dtXdwo4IBiTCCAYUwDgYDVR0P
AQH/BAQDAgEGMBIGA1UdEwEB/wQIMAYBAf8CAQAwfQYIKwYBBQUHAQEEcTBvMD8G
CCsGAQUFBzAChjNodHRwOi8vd3d3LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kvY2Vy
dHMvZWNjcm9vdC5jZXIwLAYIKwYBBQUHMAGGIGh0dHA6Ly9wa2ljdnMuY2lzY28u
Y29tL3BraS9vY3NwMB8GA1UdIwQYMBaAFKRFti+jMbF2FbAKGDPK9q1PPSgEMFwG
A1UdIARVMFMwUQYKKwYBBAEJFQETADBDMEEGCCsGAQUFBwIBFjVodHRwOi8vd3d3
LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kvcG9saWNpZXMvaW5kZXguaHRtbDBCBgNV
HR8EOzA5MDegNaAzhjFodHRwOi8vd3d3LmNpc2NvLmNvbS9zZWN1cml0eS9wa2kv
Y3JsL2VjY3Jvb3QuY3JsMB0GA1UdDgQWBBSWhzrYiYGRQRUzv+A0jyCPwrvDljAK
BggqhkjOPQQDAwNpADBmAjEAzye9v1h5m/1bAUSU0MQwShDDCRJHXsc+1TtbqOJR
1cZMmhsIzNNy/AskkBoIgNNtAjEAk3DAKxkBx3DYyZS4Uz3B50MpbrZwpxPTUCqW
m7qExCM7m1FjIg98ltfdltPHgqP/
-----END CERTIFICATE-----

Certificate name: Cisco ECC256 SUDI
-----BEGIN CERTIFICATE-----
MIIC9jCCAnygAwIBAgIKAVc0IzZ1QRYkpTAKBggqhkjOPQQDAjArMQ4wDAYDVQQK
EwVDaXNjbzEZMBcGA1UEAxMQQUNUMiBFQ0MgU1VESSBDQTAgFw0yNTExMTEwMTA4
```

```
MzZaGA8yMDUzMDQwNDA4MTU0MlowgagxJzAlBgNVBAUTHlBJRDo4MjAyLTMyRkgt
TSBTTjpGTE0yNzUxMDFNSjEOMAwGA1UEChMFQ2lzY28xGDAWBgNVBAsTD0FDVC0y
IExpdGUgU1VESTFTMFEGA1UEAxNKQ2lzY28gODIwMCAyUlUgMzJ4NDAwRyBRU0ZQ
NTYtREQgdy9JT1MgWFIgSEJNIE1BQ3NlYyBNQUM6NzBkYTQ4ZGYwODAwLTAyMDAw
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQIy0x83157VdlQdn4d6D/EY4bygPkm
ZfDvhWHX0n4ap4wGPzotK7ddXAhRxrc6ckD3wS4kVMaguydQtngjD3mho4IBBjCC
AQIwDgYDVR0PAQH/BAQDAgXgMAwGA1UdEwEB/wQCMAAwHwYDVR0jBBgwFoAUloc6
2ImBkUEVM7/gNI8gj8K7w5YwgaEGA1UdEQSBmTCBlqBQBgorBgEEAQkVAwQCoEIT
QDAzNzU3NzJGNDM4OTUxRTJCMEM0QTAwMUZCMDhCOTFFMjI5RTg0QTI0NDI1Mjgx
NTFBMTA0QkQ4OEI1RDdEOTSgQgYJKwYBBAEJFQIDoDUTM0NoaXBJRD1uZVJmTEpa
Wk85cnJSc3BiREV1ZW1BQUFBQUFBQUFBQUFBQUFBQUFBQUFBPTAdBgNVHQ4EFgQU
57chlT8P9b/3BfUhzfe51A9d7S0wCgYIKoZIzj0EAwIDaAAwZQIxAI93h3096LNy
hkcEokGrUjXRRJ4drs3is4sTTCaaQfLUu/xLZ7tmd3LYxd+a83HHUQIwAL3NfR77
xyrnIWAoXt7YWEV6BgXgVBO3K0A01eF+LP0CsLqtYkd3JqVgdbgUAoKv
-----END CERTIFICATE-----

signature:
MEQCIFIi1NZvNCcsO2/50eLx/AoypWd4KS8OSX4jS0kw1StuAiAmeAJk9wIw1p5tCzGEOI9mpTk/chDP82GEKERuuvOmgQ==
signature version: 2
```

Ensure the PEM-formatted certificate chain and explicit signature using the ECC p256 private key are generated successfully.

4. Check TAM status and take necessary actions based on these conditions.

Table 9: Table

If	Then
TAM is shown as ECC p256 capable	No further action is required
ECC p256 certificates are missing	Initiate TAM provisioning or troubleshooting procedures
Signature verification fails	Verify the nonce and consult security policies.

The Trusted Anchor Module confirms ECC P256 capability and displays valid certificate information, demonstrating that secure device authentication and attestation are enabled.

Monitor XR shell commands

Explains how monitoring XR shell captures, associates, and protects audit records of shell commands for enhanced security and compliance.

Monitor XR shell commands is an IOS XR security capability that

- audits **run** and **bash** sessions on the active Route Processor (RP) but excludes shell built-in commands
- associates each command with the original authenticated user, and
- sends execution events to a logging mechanism called the shell-logger.

The Shell-Logger uses Tetragon to provide command accountability for monitored IOS XR run and bash sessions on the active Route Processor (RP). It creates protected audit records that associate shell activity with authenticated IOS XR users, supporting security monitoring, compliance, and troubleshooting. Run and bash sessions on the standby RP or other nodes are not monitored.

Tetragon runtime observability

Tetragon, an extended Berkeley Packet Filter (eBPF)-based runtime observability engine, uses eBPF programs to track kernel-level process execution events. Each tracked process execution—including shell commands, script-based commands, and commands executed in nested shells—generates an event. Each event records the timestamp, username, executed command, and whether the command originated from an IOS XR **run** or **bash** session.

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
Monitor XR shell commands	Release 26.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now trace **run** and **bash** commands to specific users and identify potential system compromises. By implementing shell-level user activity tracking, the router records each action, the user who performed it, and its timestamp. You can use this data to monitor user behavior, detect potentially malicious activity, and respond rapidly to security incidents. This feature introduces these changes: **CLI:** • show shell history location

Export shell history before an install operation

Export shell audit history from the router before performing a package installation or image upgrade that clears the local shell-logger directory.

Ensure that shell audit history is preserved by exporting it from volatile storage before executing installation or upgrade operations.

Files under `/var/log/xr-shell-logger/` do not persist across package installations or image upgrades. Exporting history allows you to retain records for compliance and troubleshooting.

Before you begin

- Identify the route-processor location containing the shell history.
- Determine a persistent destination for backup, such as `harddisk:`

Follow these steps to export shell history before an install operation:

Procedure

1. Display the history for the required route processor.

Example

```
Router#show shell history location 0/RP0/CPU0
Mon Jul 13 12:27:24.451 UTC

Time                          User       Origin         Command
----------------------------  ---------- ----------
-------------------------------
Aug  3 2026 17:41:23.541 UTC  user       run        /bin/sh -c -- "cat
/proc/57392/cmdline"
Aug  3 2026 17:41:23.546 UTC  user       run        /usr/bin/cat
/proc/57392/cmdline
Aug  3 2026 17:41:23.548 UTC  user       run       /bin/sh -c -- "ps -p 57392
 -o ppid="
Aug  3 2026 17:41:23.551 UTC  user       run        /usr/bin/ps -p 57392 -o
ppid=
Aug  3 2026 17:41:23.579 UTC  user       run        /bin/sh -c -- "cat
/proc/57365/cmdline"
Aug  3 2026 17:41:23.583 UTC  user       run        /usr/bin/cat
/proc/57365/cmdline
Aug  3 2026 17:41:23.585 UTC  user       run       /bin/sh -c -- "ps -p 57365
 -o ppid="
Aug  3 2026 17:12:35.222 UTC  user       bash       /bin/hostname
Aug  3 2026 17:12:35.223 UTC  user       bash       /bin/less -V
Aug  3 2026 17:12:35.225 UTC  user       bash       /bin/tty
Aug  3 2026 17:12:40.733 UTC  user       bash       /bin/ls -lrt
Aug  3 2026 17:12:43.637 UTC  user       bash       /bin/uname -a
Aug  3 2026 17:12:46.309 UTC  user       bash       /bin/whoami
```

This output displays a table of commands run, including timestamp, user, origin, and command details.

2. Export the history to persistent storage.

Example

```
RP/0/RP0/CPU0:SFDR1#show shell history location 0/RP0/CPU0 | file harddisk:
```

This saves the shell history to a location that is not cleared during install operations. You can also collect diagnostic data for the shell-logger process and save it to a specific storage location by using the **show tech-support shell-logger file** command. By adding **file**, you direct the output to a file instead of displaying it on the console.

The shell history is successfully exported to persistent storage and preserved prior to the install operation.

What to do next

Proceed with the package installation or image upgrade after exporting shell history.

4 Maintaining Trust

Topics:

- Steady-state trust

Outlines comprehensive measures for maintaining trust in secure systems, including steady-state trust models, SELinux, secure installation, SSD encryption, runtime defenses, boot integrity, secure communications, and integrity verification mechanisms.

Steady-state trust

Describes steady-state trust concepts, encompassing SELinux implementation and policy, secure installation mechanisms such as RPM signature validation, SSD encryption with DM-Crypt and data zeroization, runtime defenses, boot integrity and trust visibility, secure gRPC session practices, and the Integrity Measurement Architecture.

Steady-state trust is a runtime trustworthy systems state that

- maintains protection after boot
- uses operating system and storage controls, and
- provides visibility into software integrity.

The third component in implementing a trustworthy system is to maintain trust in the steady or runtime state.

Attackers are seeking long-term compromise of systems and using effective techniques to compromise and persist within critical infrastructure devices. Hence, it is critical to establish and maintain trust within the network infrastructure devices at all points during the system runtime.

Additional reference information

In Cisco IOS XR7, trust is established and maintained in a steady state through:

- SELinux
  - SELinux Policy
  - SELinux Mode
- Secure Install
  - RPM Signing and Validation
  - Third-Party RPMs
- SSD Encryption

SELinux

Explains how SELinux enforces mandatory access controls, confines processes to limited domains, and reduces the impact of compromised applications on Linux systems. SELinux policies, especially targeted ones, provide enhanced security by ensuring third-party applications are restricted in what they can access.

SELinux is a Linux kernel security module that

- supports mandatory access control policies
- confines targeted processes to limited domains, and
- reduces the impact of compromised applications.

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including mandatory access controls (MAC).

A kernel integrating SELinux enforces MAC policies that confine user programs and system servers to the minimum amount of privileges they require to do their jobs. This reduces or eliminates the ability of these programs and daemons to cause harm when compromised, for example through buffer overflows or misconfigurations. This confinement mechanism operates independently of the traditional Linux access control mechanisms. SELinux does not use a "root" super-user and avoids the well-known weaknesses of traditional Linux security models, such as reliance on setuid/setgid binaries.

On Cisco IOS XR7 software, only Targeted SELinux policies are used, so that only third-party applications are affected; all Cisco IOS XR programs can run with full root permission.

With Targeted SELinux, processes that are targeted run in a confined domain. For example, the httpd process runs in the httpd_t domain. If a confined process is compromised by an attacker, SELinux policy configuration limits the attacker's access to resources and reduces the potential damage.

Additional reference information

Processes running in unconfined domains fall back to using discretionary access control (DAC) rules.

DAC is a type of access control defined as a means of restricting access to objects based on the identity of the subjects or groups to which they belong.

SELinux policy

Explains how SELinux policies map users and process domains to enforce restrictions.

An SELinux policy is an access-control policy that

- maps Linux users to SELinux users
- defines domain transitions for applications, and
- applies restrictions to confined users and processes.

Each Linux user is mapped to an SELinux user through an SELinux policy. This allows Linux users to inherit the restrictions placed on SELinux users.

If an unconfined Linux user executes an application, which an SELinux policy defines as an application that can transition from the unconfined_t domain to its own confined domain, the unconfined Linux user is subject to the restrictions of that confined domain. The security benefit is that, even though a Linux user is running in unconfined mode, the application remains confined. Therefore, the exploitation of a flaw in the application is limited by the policy.

A confined Linux user is restricted by a confined user domain against the unconfined_t domain. The SELinux policy can also define a transition from a confined user domain to its own target confined domain. In such a case, confined Linux users are subject to the restrictions of that target confined domain.

SELinux operating modes

Lists and describes the enforcing, permissive, and disabled operating modes of SELinux, including how each mode impacts policy enforcement and system behavior.

The SELinux security system operates in one of three modes, each controlling how SELinux policies are applied:

- Enforcing: SELinux actively applies the security policy and denies access to resources based on defined SELinux policy rules.
- Permissive: SELinux does not enforce policies but logs any access denials. This mode is used primarily for debugging and policy development. Permissive is the default mode.
- Disabled: SELinux policy is not loaded, and no enforcement or logging occurs. You can change the mode through the boot loader, SELinux configuration file, or at runtime using the **setenforce** command.

How SELinux policy works during boot

Describes the role of SELinux policy during system startup.

At system startup, SELinux must ensure every process is assigned the proper domain label so that access controls are enforced from the earliest moment. This prevents unauthorized actions and maintains system integrity.

Summary

The key components involved in the process are:

- SELinux policy: Governs how processes and files are labeled and what actions are permitted.
- init process: Responsible for orchestrating system startup and enforcing SELinux domain labeling.
- System processes: Require correct domain labels to operate within policy restrictions.

SELinux policy ensures proper security labeling and enforcement during system startup, providing a secure environment for all system processes from the moment the system boots.

Workflow

These stages describe how SELinux policy works during boot.

1. Initialization: The system boots, and the init process begins execution as the first userspace process.
2. Policy loading: The SELinux policy is loaded, defining the rules for labeling and permitted actions.
3. Domain assignment: The init process labels itself and other startup processes with their respective domains according to the SELinux policy.
4. Policy enforcement: SELinux continuously monitors and enforces domain labeling as new processes launch, ensuring all processes adhere to security restrictions.

Result

System startup proceeds with all processes labeled in their proper domains, allowing SELinux to enforce security policies and maintain system integrity from boot.

Secure install mechanisms

Explains how Cisco IOS XR ensures the integrity of install operations by validating digitally signed RPM packages and enforcing strict controls on package types during software installation. This mechanism prevents unauthorized or unsigned packages from being installed and protects system reliability.

A secure install mechanism is a software package integrity solution that

- uses digitally signed Cisco RPMs
- validates package signatures during install actions, and
- controls the package types that can be installed.

The Cisco IOS XR software is shipped as RPMs. Each RPM consists of one or more processes, libraries, and other files. An RPM represents a collection of software that performs a similar functionality, such as network protocol packages (e.g., BGP, OSPF) and Cisco IOS XR infrastructure libraries and processes.

RPMs can also be installed into the base Linux system outside the Cisco IOS XR domain; however, those RPMs must also be appropriately signed. All RPMs shipped from Cisco are secured using digitally signed Cisco private keys.

There are three types of packages that can be installed:

- Packages shipped by Cisco (open source or proprietary)
- Customer packages that replace Cisco provided packages
- Customer packages that do not replace Cisco provided packages

RPM signature validation

Explains how Cisco IOS XR validates signed RPMs during install operations.

A RPM signature validation is a package-integrity mechanism that

- uses Cisco keys during the build process

- verifies package signatures during install actions, and
- rejects packages when signature validation fails.

- RPMs are signed using Cisco keys during the build process.
- The install component of Cisco IOS XR automatically performs actions such as verification, activation, deactivation, and removal on RPMs. Many of these actions invoke the underlying DNF installer, which verifies RPM signatures to ensure only legitimate packages are processed.
- Cisco RPMs are signed with GPG keys. The RPM format includes a signature area for the header and payload, which DNF package managers validate during install operations.

Feature history

The feature history table lists release support for this feature.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
RPM Signing and Validation	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Validate third-party RPM signatures

Outlines the mandatory signature validation process for third-party RPMs during installation.

You must ensure that all third-party RPM packages have valid signatures when installing them using XR Install. XR Install uses the 'gpgcheck' option of DNF to enforce signature validation. Any third-party RPM installation through XR Install fails if the package does not pass signature validation.

SSD encryption

Explains disk-level encryption methods for protecting sensitive data on persistent storage media, binding encrypted data to unique system keys, and providing secure access through specific cryptographic tools and hardware.

SSD encryption is a disk-level data protection capability that

- encrypts sensitive data on persistent storage media
- binds encrypted data to a system-specific key, and
- uses DM-Crypt, AES-NI support, and Cryptsetup.

Customers are concerned about the security of sensitive data present on persistent storage media. User passwords are limited in their capability to protect data against attackers who bypass software systems and directly access the storage media.

In this case, only encryption can guarantee data confidentiality.

Cisco IOS XR Software introduces SSD encryption that encrypts data at the disk level. SSD encryption also ensures that the encrypted data is specific to a system and is accessible only with a specific key to decrypt it.

Data that can be encrypted includes sensitive information such as topology data, configuration data, and similar items.

Encryption is an automatic process and can be achieved through the following:

- DM-Crypt
- CPU with AES-NI support
- CryptSetup

Feature history

The feature history table lists release support for this feature.

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
SSD Encryption for Addiitonal PIDs	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH 8011-12G12X4Y-A/D
SSD Encryption for Addiitonal PIDs	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SSD Encryption for Addiitonal PIDs	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
SSD Encryption for Addiitonal PIDs	Release 7.5.1	This feature enables trust and security in the system's steady state by encrypting data at the disk level. The encrypted data can be accessed only with a specific key stored in the TAm.

Feature Name	Release Information	Feature Description
Solid State Drive (SSD) Encryption	Release 7.3.1	This feature enables trust and security in the system's steady state by encrypting data at the disk level. The encrypted data can be accessed only with a specific key stored in the TAm.

DM-Crypt

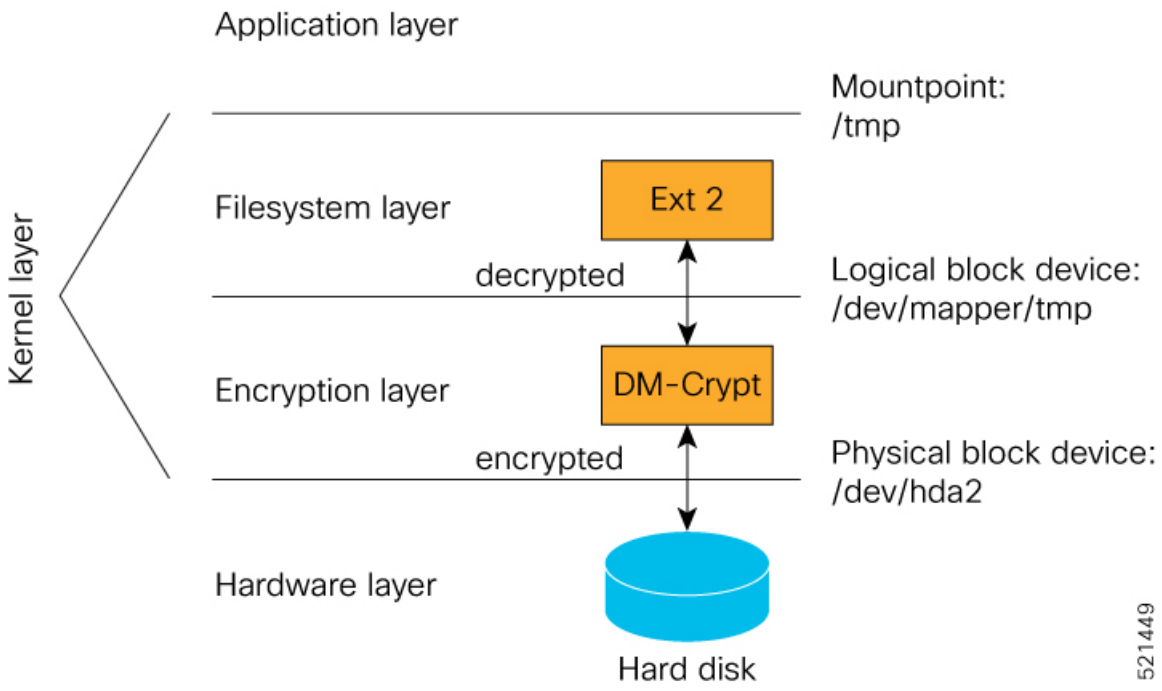
Explains how DM-Crypt provides transparent block-device encryption, enabling secure storage by encrypting and decrypting data blocks automatically as they are written to and read from block devices on Linux systems.

DM-Crypt is a Linux kernel module that

- uses device-mapper infrastructure
- encrypts data written to a block device, and
- decrypts data read from a block device.

DM-Crypt is a Linux kernel module that provides disk encryption. The module takes advantage of the Linux kernel's device-mapper (DM) infrastructure, which offers a way to create virtual layers of block devices. DM-Crypt is a device-mapper target that provides transparent encryption for block devices using the kernel crypto API. When data is written to the block device, it is encrypted; when data is read from the block device, it is decrypted.

Figure 3: DM-Crypt encryption



How AES-NI support works

Explains how AES-NI accelerates encryption and decryption for DM-Crypt operations, enabling high-speed hardware cryptography and freeing CPU resources for other tasks.

Advanced Encryption Standard New Instructions (AES-NI) is utilized in systems requiring encrypted block devices. By offloading cryptographic operations to hardware, AES-NI enables faster and more efficient encryption processes, freeing up the CPU for other tasks.

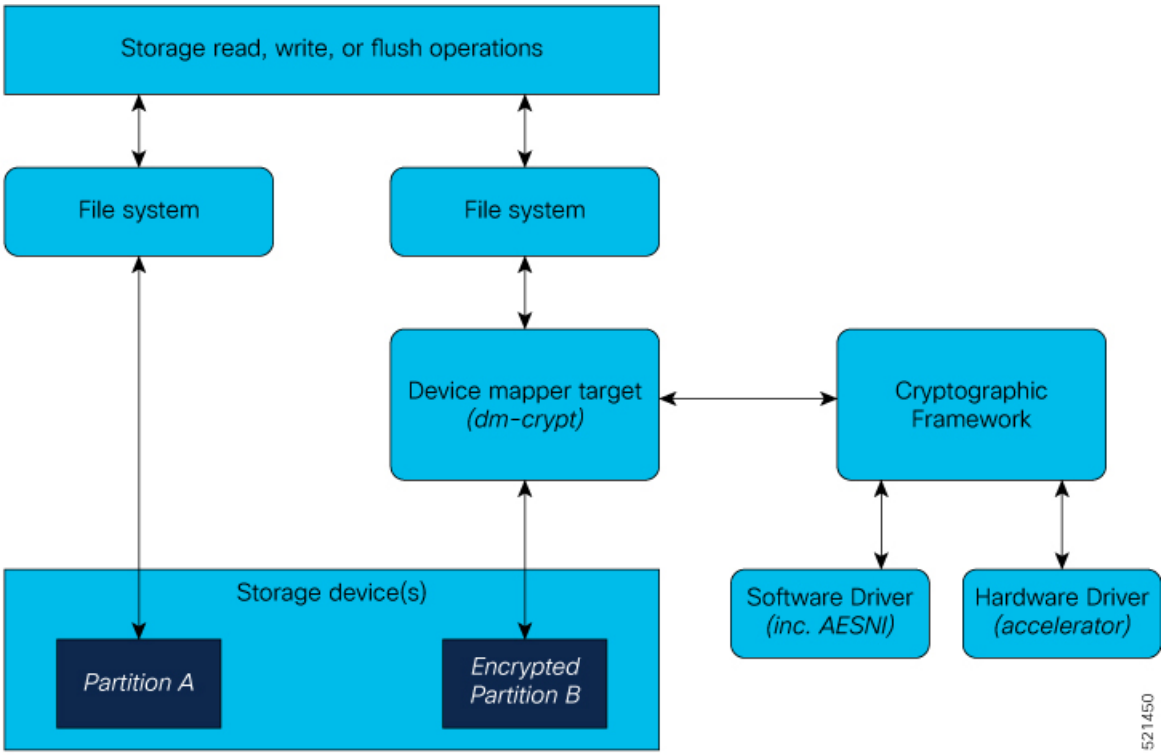
Summary

The key components involved in the process are:

- AES-NI hardware engine: Provides hardware-based encryption and decryption for faster data processing.
- DM-Crypt: Passes read-write requests to the crypto framework for encryption and decryption.
- Cryptographic Framework: Utilizes off-chip hardware accelerators for cryptography and provides software implementations as needed.

AES-NI is a hardware-assisted engine from Intel that accelerates encryption and decryption for DM-Crypt operations, allowing high-speed data processing and optimized CPU usage.

Workflow



The process describes how AES-NI support works.

1. Input-output operations initiate read-write requests to the encrypted block device.
2. DM-Crypt receives these requests and passes multiple cryptographic requests to the Cryptographic Framework.
3. The Cryptographic Framework leverages hardware accelerators (when available) to perform encryption and decryption, or falls back on software implementations if accelerators are not available.
4. After processing, the crypto framework returns the encrypted or decrypted data to DM-Crypt.

Result

Encrypted or decrypted data is efficiently returned to DM-Crypt following hardware-accelerated processing, improving overall system performance and freeing CPU resources for additional tasks.

Cryptsetup

Explains the user-space tool that manages DM-Crypt encrypted devices.

A cryptsetup tool is a command-line interface utility that

- interacts with DM-Crypt to manage encrypted volumes
- enables users to create, access, and manage encrypted devices, and
- provides a user-space mechanism for setting up cryptographic disks.

DM-Crypt is a kernel-level encryption subsystem that relies on user-space tools such as cryptsetup to configure and manage cryptographic volumes.

Encrypted logical volumes

Explains activation, deactivation, and migration behaviors for encrypted logical volumes used to securely store sensitive files.

An encrypted logical volume is a storage volume that

- can be created during software installation
- can be activated or deactivated on demand, and
- stores sensitive files on encrypted disk partitions.

- An encrypted logical volume (LV) is created during software installation.
- You can activate or deactivate the encrypted disk partition as needed. When activated, all sensitive files are migrated from the unencrypted disk partition to the encrypted disk partition; they can be migrated back during deactivation.
- The capacity of the encrypted logical volume is 150 MB of disk space and is accessible as */var/xr/enc* for applications.

Additional reference information

Applications may use this space for storage. However, if the software image is downgraded to a version that does not support encryption, this data is not included in migration.

Activate data encryption for an encrypted logical volume

Configure data encryption on an encrypted logical volume by running the disk-encryption activation command and verifying the output.

Enable data encryption and verify successful completion of the backup and encryption activation process for a specific logical volume.

Use the **disk-encryption activate location** command to activate encryption on a logical volume. This ensures the data is protected by encrypting the specified disk. Reviewing the sample output confirms that the process has completed successfully.

Before you begin

Confirm that the router is running a supported Cisco IOS XR release and platform variant.

Procedure

Activate data encryption.

Example

```
Router# disk-encryption activate location 0/RP0/CPU0
Tue Apr 16 14:35:00.939 UTC
```

```
Preparing system for backup. This may take a few minutes especially for large
 configurations.
 Status report: node0_RP0_CPU0: START TO BACKUP
Router# Status report: node0_RP0_CPU0: BACKUP HAS COMPLETED SUCCESSFULLY
[Done]
```

The router completes the backup and activates data encryption on the specified logical volume. You can confirm successful activation by reviewing the output for the completion message.

SSD binding

Explains how encryption binds the SSD to the system-specific key stored in the TAm.

A SSD binding is an encryption behavior that

- stores an encryption key in the Trust Anchor module (TAm)
- unlocks an encrypted logical volume only with the system-specific key, and
- prevents an encrypted SSD from being unlocked on another line card.

When encryption is activated on a system, each card generates a random encryption key and stores it in its own secure storage—the Trust Anchor module (TAm). During successive reboots, the encryption key is read from the TAm and applied to unlock the encrypted device. Since each card stores its encryption key locally on the TAm, an SSD that is removed from one card and inserted into another cannot be unlocked by the key stored on that card, thereby making the SSD unusable.

If encryption is activated, the encrypted LV can only be unlocked by using the key stored in the TAm. So, if an encrypted SSD is removed and moved to another line card, the SSD cannot be unlocked. In other words, when you activate encryption, the SSD is bound to the card it is inserted in.

Data zeroization

Explains the process by which zeroization deletes sensitive data from cryptographic modules.

Data zeroization is a cryptographic-module operation that

- deletes sensitive data
- can be performed from the XR prompt, and
- uses the factory reset location command.

Zeroization refers to the process of deleting sensitive data from a cryptographic module.

You can perform zeroization by using the **factory reset location** command from the XR prompt.

Caution: data zeroization on encrypted systems

Outlines factory reset requirements and encryption-key deletion risks during data zeroization on encrypted systems.

- In case of a Return Material Authorization (RMA), you must perform a factory reset to zeroize the data.
- Running the **zeroization** command while encryption is activated deletes the master encryption key from the TAm and renders the motherboard unusable after the subsequent reload.

Runtime defenses

Describes runtime defense tools that mitigate common security vulnerabilities.

A runtime defense is a runtime security tool that

- mitigates common vulnerabilities
- protects memory and object access patterns, and
- monitors runtime defense functionality on the router.

Run Time Defenses (RTD) are a collection of tools used at runtime to mitigate common security vulnerabilities. RTD is classified into three groups:

Address space layout randomization

Explains how address space layout randomization (ASLR) helps prevent attackers from exploiting vulnerable functions by randomizing memory addresses within a process, making it difficult to locate target functions and reducing the likelihood of successful attacks.

Address space layout randomization (ASLR) is a runtime defense technique that

- randomizes memory addresses
- makes vulnerable function locations harder to predict, and
- reduces exploit reliability.

ASLR is a technique that stops an attacker from accessing a vulnerable program function by preventing them from discovering its memory address within a running process. To achieve this, ASLR randomly distributes fundamental parts of the process (such as the executable base, stack pointers, libraries, and other components) within the memory address space assigned by the operating system. As a result, attackers cannot reliably determine the memory address of a target function, making exploitation nearly impossible. If an attacker attempts to brute-force the address, the process typically crashes, thwarting the attack.

Kernel address space layout randomization

Explains how kernel address space layout randomization (KASLR) randomizes memory locations within the Linux kernel to make exploit prediction more difficult. This runtime defense technique protects fundamental kernel process components by introducing unpredictability into memory allocation.

A kernel address space layout randomization (KASLR) is a runtime defense technique that

- applies address randomization to the Linux kernel
- protects fundamental kernel process parts, and
- reduces kernel exploit predictability.

Kernel Address Space Layout Randomization (KASLR) is a technique similar to Address Space Layout Randomization (ASLR) but applies specifically at the Linux kernel level. KASLR protects the Linux kernel by randomizing the fundamental structure and locations of kernel processes within kernel memory during system boot. This prevents attackers from easily predicting or obtaining pointers to kernel memory distribution tables, hindering exploitation attempts.

Built-in object size checker

Explains how built-in object size checkers (BOSC) mitigate buffer-overflow vulnerabilities in SafeC library functions.

A built-in object size checker (BOSC) is a compiler-provided protection mechanism that

- checks object size constraints
- helps avoid buffer-overflow vulnerabilities, and
- protects common SafeC functions.

BOSC is a limited buffer overflow protection mechanism provided by the compiler. BOSC helps in avoiding buffer-overflow related security vulnerabilities on common SafeC library functions such as, strcpy_s and memcpy_s.

Executable space protection

Explains how XSpace protects executable and data memory regions.

Executable space protection is a runtime defense technique that

- protects data and code portions of program memory
- blocks illegal execution attempts in data memory, and
- blocks write attempts in code memory.

XSpace mitigates malicious code injection attacks by protecting the data and code portions of the program memory. The basic approach of such attacks is to overflow a buffer in the program stack and cause the transfer of control to injected code. Once in the injected code, the application behaves in an unexpected manner.

X-space ensures that the area of the memory where the authentic code exists is write-protected and the area of memory where the data is present is execution-protected. Illegal execution-attempts in the data-portion or write-attempts in the code-portion results in the application crashing.

Requirement: keep executable space protection enabled

Outlines the need to keep the XSpace functionality enabled to maintain executable space protection at all times.

You must keep the XSpace functionality enabled at all times. Do not disable executable space protection under any circumstances.

RTD monitors for runtime defense functionality

Identifies the monitors that track runtime defense features and status on the router.

The RTD monitor tracks various aspects of runtime defense on the router, including:

- Real-time monitoring of all RTD features enabled on the device.
- Reporting the operational status of RTD defense mechanisms.
- Maintaining logs for security-related runtime events.

Features

- Provides consolidated security status for all runtime defense features.
- Enables administrators to quickly identify which defense mechanisms are active or inactive.
- Supports log retrieval for further analysis of runtime events.

Benefits

- Simplifies security management by centralizing defense monitoring.
- Enhances visibility into router protection mechanisms.
- Assists in troubleshooting by providing detailed runtime security information.

Boot integrity and trust visibility

Explains how boot integrity and trust visibility features expose platform identity, record boot measurements, and support software attestation.

Boot integrity and trust visibility is an observability capability that

- exposes platform identity information

- records boot integrity measurements, and
- supports verification of Cisco IOS XR software integrity.

Boot integrity and trust visibility allows Cisco's platform identity and software integrity information to be visible and actionable. Platform identity provides the platform’s manufacturing installed identity. Software integrity exposes boot integrity measurements that can be used to assess whether the platform has booted trusted code.

During the boot process, each stage of the firmware and software calculates and records the runtime checksum of the next stage boot entity in the Trusted Attestation Module (TAm). This record can be retrieved and compared with a Cisco-certified reference to verify if the software image is genuine. If checksum values differ, the software image may not be Cisco-certified or could have been altered by an unauthorized party.

Boot integrity and trust visibility provide customers with attestation capabilities to help ensure that Cisco IOS XR software has not been tampered with.

Verify boot integrity and trust visibility on the router

Verify the boot integrity, platform configuration register (PCR) values, and trust anchor module (TAm) output to ensure that your router is operating securely and has not been tampered with.

Ensure the router’s boot integrity and trust status are validated by reviewing PCR measurements, integrity logs, and TAm device details.

Boot integrity verification uses cryptographic measurements stored in Platform Configuration Registers (PCRs) and device trust anchor modules (TAm) to provide assurance that hardware and software components have not been altered. You can use IOS XR CLI or NETCONF protocol to obtain these values.

Before you begin

- Confirm that the router is running a supported Cisco IOS XR release and platform variant.
- Ensure that you have access to the router CLI or NETCONF protocol.

Procedure

---

1. Collect PCR and trust measurements.

   Example

   ```
   Router# show platform security attest pcr 15 trustpoint CiscoAIK nonce 4567
   ```

   This command displays integrity log values and confirms measurement authenticity.

   - Parameters measured include:
     - Micro loader hash
     - Boot loader hash
     - Image signing and management key hashes
     - Operating system image hash
   - Platform Configuration Register (PCR) 0-9 are used for secure boot.
     - Signature version designates the format of the signed data. The signature digest is SHA256, and the signing key is TCG-compliant.
     - (Optional) Use the Cisco-IOS-XR-remote-attestation-act.yang model to fetch boot integrity via NETCONF protocol.

2. Review hardware digest and compare PCR values.
   - These commands display "known-good" and "observed" digest values and PCRs for hardware integrity.
   - Ensure that observed digests match known-good digests. PCRs should match platform expectations.

Example

```
Router# show platform security attest PCR 15 trustpoint CiscoAIK nonce 4567
location 0/RP0/CPU0

Sun Jun 21 03:07:18.394 UTC
Nonce: 4567

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
Uptime: 495270
pcr-quote:
/1RDR4AYACCBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hUwAAAEf/////
AQAAACQAAAALAAAAAQALAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3bqeVdpW7CObwBt+g=
pcr-quote-signature:
EZbzSUge89jSjH8ZqTgKJrZJBopEbd818C+h1Ec780qi7Li1WfCZQPIP6KCDV6HsRCVzLoFijgmlMLoZE2rakQq+/
1TgZOWSLjMY7RbjSFr8z/zbpVI+YLnOG+wytVYWuY33uKHBn/
YWokHwo+qVf7u9aLGhnrXKvRUaFknBiZtQGiyAdis6GbPTToqn0WSN1y6DPh4UHZj1vLVwJsI48mbQUrAyCZrz/
XBHLM38tVJjqSrC0jw/6LF2DDoT5ks0VUFT7sqbysw4F56y+z/IlDBrrRW3GFOY46MOxDxLwSl1/
n6zdoVjiKKeqKOnmhpBh72bJQAdeu/GVOYTrOSy4Q==
pcr-index       pcr-value
  15    oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=

Router# show platform security integrity hardware digest-algorithm SHA1
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:14.594 UTC
Nonce: 4567

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495385
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=

Router# show platform security integrity hardware digest-algorithm SHA256
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:31.110 UTC
Nonce: 4567

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495401
```

```
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=

Router# show platform security integrity hardware digest-algorithm SHA256
trustpoint CiscoAIK nonce 4567 location 0/RP0/CPU0

Sun Jun 21 03:09:43.782 UTC
Nonce: 4567

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495414
Known-good-digests:
Index   value
  0     y3n/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
observed-digests:
Index   value
  0     y3n/SsvyNb8g3o7FFRGCZwfbs8EGxvMZg/PeN0NA71k=
PCRs:
Index   value
  15    oYk8yqrzudIpGB4H++SaV0wMv6ugDSUIuUfeSqbJvbY=
Cisco AIK Certificate used for signing PCR
pcr-quote:
/1RDR4AYACCBG/wltf4TEwfdUjtjun7S3rXC90eAb0G0ytrYRv3ExwACRWcAAAAAAD8hywAAAEf////
/AQAAACQAAAALAAAAAQALAwCAAAAgae1J8QIYe06nS2RUx0JYeoG8tM3bqeVdpW7CObwBt+g=
pcr-quote-signature:
qyKbK7ndJbrgxeVnOodLWQzT7++NzrxJ9ERRvJzvTe4+8r6p0HGSepHUhZHzYkXw4DbniHAK0Cs3dwg/
hGKGe4M8Lz+/k682yIjaFYYip0DHMaV2ny/lT7RSqM/6u3j/JZrZv39MaeHa3MyjjonzRf9oe7EBSFAKsa/D54eTR0eFtaxFy/
XdtM0VVQe2JRdoBVxnIBLGiVmGRlVVlmHvwwgX1lAN6e3/soC1Vk3I5gjLldPHUYuJ/
7PIGyAwZsbdeigx8d4ViUUUjSMzK7JISwXa8k4GiPQVLBHtqqR+RA9scmMZTbKLsG3luIWKQeyCtXMYE1VOeW8WQlAvioMICw==
RP/0/RP0/CPU0:ios#show platform security integrity hardware digest-algorith$
Sun Jun 21 03:09:56.794 UTC
Nonce: 4567

+-------------------------------------+
   Node location: node0_RP0_CPU0
+-------------------------------------+
TPM Name: node0_RP0_CPU0_aikido
Uptime: 495427
Known-good-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
observed-digests:
Index   value
  0     3TDUS9iUDCFX3VkICcOnySOQTPA=
PCRs:
Index   value
  15    1Y3uKqNv1UJQUNZQxmZkiuG4blk=
```

3. Inspect TAm device details and certificates.
   - Review the output for device and firmware information, serial numbers, and certificate details.
   - Check SUDI Root, SUDI Sub CA, and SUDI certificates for validity, issuer, and expiration dates.

Example

```
Router# show platform security tam all location all
Mon Apr 15 14:42:34.649 UTC
-----------------------------------------
Node - node0_RP0_CPU0
-----------------------------------------
Device Type            -     AIKIDO Extended
Device PID             -     N540X-12Z16G-SYS-A
Device Serial Number   -     FOC2333NJ0J
Device Firmware Version-     0x24.000b
Server Version         -     3
Server Package Version -     9.4.1
Client Package Version -     9.4.1

Sudi Root Cert:
-----------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            01:9a:33:58:78:ce:16:c1:c1
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=Cisco, CN=Cisco Root CA 2099
        Validity
            Not Before: Aug  9 20:58:28 2016 GMT
            Not After : Aug  9 20:58:28 2099 GMT
        Subject: O=Cisco, CN=Cisco Root CA 2099
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d3:b6:e3:35:7e:0d:3e:f4:67:e5:8a:4e:1a:c6:
        Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Subject Key Identifier:
                38:95:57:0F:34:23:4E:F3:A1:26:20:BA:14:91:C7:41:88:1D:A3:5B
    Signature Algorithm: sha256WithRSAEncryption
         8d:e2:99:a3:ee:31:77:4e:53:16:da:bd:f6:72:a7:58:0d:09:

Sudi Sub CA Cert:
-----------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            0a:64:75:52:4c:d8:61:7c:62
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O=Cisco, CN=Cisco Root CA 2099
        Validity
            Not Before: Aug 11 20:28:08 2016 GMT
            Not After : Aug  9 20:58:27 2099 GMT
        Subject: CN=High Assurance SUDI CA, O=Cisco
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:bd:dc:de:49:67:43:23:a9:51:64:36:11:bc:0e:
```

```
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            Authority Information Access:
                CA Issuers -
URI:https://www.cisco.com/security/pki/certs/crca2099.cer
                OCSP - URI:http://pkicvs.cisco.com/pki/ocsp

            X509v3 Authority Key Identifier:

keyid:38:95:57:0F:34:23:4E:F3:A1:26:20:BA:14:91:C7:41:88:1D:A3:5B

            X509v3 Certificate Policies:
                Policy: 1.3.6.1.4.1.9.21.1.30.0
                  CPS: http://www.cisco.com/security/pki/policies/

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://www.cisco.com/security/pki/crl/crca2099.crl

            X509v3 Subject Key Identifier:
                EA:6B:A3:B9:C1:13:97:7E:1B:FB:3A:8D:68:60:07:39:5F:87:48:FA
    Signature Algorithm: sha256WithRSAEncryption
         5c:a9:81:0e:80:01:e1:19:62:a7:77:03:3d:d3:55:d7:d8:49:

Sudi Cert:
-----------------------------------------------
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 29200071 (0x1bd8ec7)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=High Assurance SUDI CA, O=Cisco
        Validity
            Not Before: Sep  5 03:39:36 2019 GMT
            Not After : Aug  9 20:58:26 2099 GMT
       Subject: serialNumber=PID:N540X-12Z16G-SYS-A SN:FOC2333NJ0J, O=Cisco,
 OU=ACT-2 Lite SUDI, CN=Cisco NCS 540 System with 12x10G+4x1G Cu+12x1G AC
Chassis
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:ca:2a:8a:b4:87:8b:43:68:17:d3:b2:43:44:ca:

                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Alternative Name:
                0...N.
+.........@917C927B4B340B908703945A7A0A6D14D0207ADB2FB622DFA8C83538FD7E63B5.
B..+.........5.3ChipID=QvZQd9q9psveoAz6QJQeNAAAAAAAAAAAAAAAAAAAAAA=
    Signature Algorithm: sha256WithRSAEncryption
         5b:67:da:2e:e5:d4:07:f2:ff:9c:17:c9:54:78:8b:da:16:df:
```

4. Analyze event logs and PCR contents.

   Review the event logs for ordered, TCG-defined events extended to PCRs. Confirm that event hashes and PCR register contents align with expected values.

   Example

```
platform-pid string Platform ID
Event log [key: event_number]: Ordered list of TCG described event log
                               that extended the PCRs in the order they
                               were logged
    +-- event_number   uint32 Unique event number of this even
    +-- event_type     uint32 log event type
    +-- PCR_index      uint16 PCR index that this event extended
    +-- digest         hex-string The hash of the event data
    +-- event_size     uint32 Size of the event data
    +-- event_data     uint8[] event data, size determined by event_size
PCR [index] - List of relevant PCR contents
    +-- index    uint16 PCR register number
    +-- value   uint8[] 32 bytes - PCR register content
PCR Quote binary TPM 2.0 PCR Quote
PCR Quote Signature binary Signature of the PCR quote using TAM-held ECC  or
 RSA restricted key with the optional nonce if supplied
```

The router displays boot integrity measurements, PCR values, and TAm device details. If digest values and certificates are verified and match expectations, your router's integrity and trust status are validated.

How boot integrity verification works

Describes the automatic BIOS and PCR verification sequence performed during system startup.

The boot integrity verification is automatic and the BIOS reports the values to the PCR.

Summary

The key components involved in the process are:

- BIOS: Reports system versions and communicates with Platform Configuration Register (PCR).
- Platform Configuration Register (PCR): Receives and stores integrity measurements during boot.
- Integrity database: Provides expected integrity values for comparison during verification.

Boot integrity verification is an automated process that checks the system's startup components to ensure the platform's integrity and detect any compromise.

Workflow

These stages describe how boot integrity verification works.

1. Report Boot 0 version and look up the expected integrity value for this platform and version.
2. Report bootloader version and look up the expected integrity value for this platform and version.
3. Report OS version and look up the expected integrity value for this platform and version.
4. Compute the expected PCR 0 and PCR 8 values using integrity values obtained from steps 1–3.
5. Compare the expected PCR values against the actual PCR values.
6. Verify the nonced signature to ensure the liveliness of the response data (this step is performed only after platform identity verification using SUDI).
7. (Optional) Verify that the software image (IOS XR) version matches what is expected to be installed on the platform.

Result

A failure in any step indicates either a compromised system or an incomplete integrity value database.

Secure gRPC sessions

Explains gRPC and default TLS behavior for secure client and server connections.

A secure gRPC session is a remote procedure call communication capability that

- uses TLS by default
- provides a secure connection between client and server, and
- prevents eavesdropping, tampering, and message forgery.

- gRPC remote procedure calls: An open source remote procedure call (RPC) system that provides features such as authentication, bidirectional streaming and flow control, blocking or non-blocking bindings, cancellation, and timeouts.

  For more information, see https://opensource.google.com/projects/grpc.
- Transport Layer Security (TLS): A cryptographic protocol that provides end-to-end communications security over networks. TLS prevents eavesdropping, tampering, and message forgery.
- Default TLS behavior in Cisco IOS XR7: In Cisco IOS XR7, TLS is enabled by default in gRPC to provide a secure connection between the client and server.

Recommendation: disable TLS 1.0 for gRPC

Outlines why you should disable TLS version 1.0 for gRPC to reduce security risk.

We recommend disabling TLS version 1.0 for gRPC. Although TLS provides secure communication between servers and clients, TLS version 1.0 may pose a security threat. You can disable TLS version 1.0 using the **grpc tlsv1-disable** command.

Integrity Measurement Architecture

Explains the components and goals of the Linux kernel integrity subsystem, focusing on file measurement, appraisal, and auditing for maintaining system trustworthiness.

Integrity Measurement Architecture is a Linux kernel integrity subsystem that

- detects accidental or malicious file alteration
- measures and appraises file integrity values, and
- audits integrity information.

The goals of the Linux kernel integrity subsystem are to:

- detect whether files are accidentally or maliciously altered, both remotely and locally
- measure the file by calculating the hash of the file content
- appraise a file's measurement against a known good value stored as an extended attribute
- enforce local file integrity

There are three components in the Linux kernel integrity subsystem:

- IMA measurement: Maintains a runtime measurement list and an aggregate integrity value anchored in the hardware Trusted Anchor module (TAm). This design prevents software attacks from compromising the measurement list without detection and supports attesting to runtime integrity in trusted boot systems.

- IMA appraisal: Appraises file measurements against known good values to assess integrity.
- IMA audit: Audits integrity information for analysis and historical tracking.

> **Note**
>
> These goals are complementary to the Mandatory Access Control (MAC) protections provided by SELinux.

For more information about Integrity Measurement Architecture, see An Overview of The Linux Integrity Subsystem.

IMA signatures

Describes IMA appraisal signatures and signature fields stored with RPM-installed files.

The IMA appraisal provides local integrity, validation, and enforcement of the measurement against a known good value stored as an extended attribute—security.ima. The method for validating file data integrity is based on a digital signature, which in addition to providing file data integrity also provides authenticity. Each file (RPM) shipped in the image is signed by Cisco during the build and packaging process and validated at runtime using the IMA public certificate stored in the TAm.

All RPMs contain Cisco IMA signatures of the files packaged in the RPM, which are embedded in the RPM header. The IMA signature of the individual file is stored in its extended attribute during RPM installation. This protects against modification of the Cisco RPMs.

The IMA signature format used for IMA can have multiple lines and every line has comma-separated fields. Each line entry will have the filename, hash, and signature as illustrated below.

- File – Filename with the full path of the file hashed and signed
- Hash – SHA256 hash of the file
- Signature – RSA2048 key-based signature

5 AAA Access Model and Local User Administration

Topics:

- AAA service capabilities
- User identities
- AAA administrative models
- Configure the first router user
- Configure a task group

Outlines AAA access configuration, user identity management, administrative models, and local user setup, guiding administrators through requirements, group structures, database integration, method lists, and step-by-step procedures for secure AAA implementation on network devices.

AAA service capabilities

Introduces the primary functions of AAA services, outlining essential preparation steps and requirements for AAA configuration, including server compatibility and interoperability considerations to ensure successful deployment.

AAA service capabilities are Cisco IOS XR software features that

- control administrative access to the router
- use task-based authorization as the administrative model to manage user access, and
- verify user identity and permissions through authentication and authorization.

AAA services availability

AAA is included in the Cisco IOS XR software base package by default. Major tasks for implementing task-based authorization include configuring task groups and user groups.

Before configuring user groups, task groups, or external authentication servers (such as RADIUS or TACACS+), review the conceptual information to understand the purpose and significance of AAA.

To configure AAA services, follow the appropriate configuration tasks for this topic.

Requirement: Prepare for AAA service configuration

Outlines the access, task-permission, initial-user, and fallback-database requirements that you must meet before configuring AAA services on Cisco IOS XR routers.

To prepare for AAA service configuration, ensure that you meet the following requirements:

- You must belong to a user group associated with a task group that includes the necessary task IDs for AAA command access. Refer to the command reference guides for task ID requirements for each command. If you cannot use a command, contact your AAA administrator to confirm your user group assignment.
- Establish a root system user during initial setup. Administrators may configure local users without any specific AAA configuration, but use an external AAA security server and database if user accounts are shared across multiple routers in an administrative domain. Configure the local database as a fallback to ensure access if the external server is unreachable.

Requirement: Verify AAA server compatibility and interoperability

Outlines the compatibility, interoperability, and task-ID support requirements you must consider before connecting Cisco IOS XR routers to external AAA servers.

To ensure proper compatibility and interoperability when connecting Cisco IOS XR routers to external AAA servers, follow these requirements:

- Verify compatibility only with the Cisco freeware TACACS+ server and FreeRADIUS. Cisco supports these servers for AAA integration.
- When choosing external TACACS+ or RADIUS server software for AAA services, ensure it follows Cisco's compatibility guidelines.
- Use the same AAA server software and database (such as CiscoSecure ACS) for your router and other Cisco equipment not running Cisco software to simplify interoperability.
- If the external TACACS+ server does not support task IDs, follow Cisco's task-ID interoperability guidance provided in the *AAA password security and authorization policies* chapter.
- The router will preserve task-ID behavior only where the external AAA server supports it.

User identities

Details the structure of user identities for AAA, describing user categories, group management, task group organization, predefined groups, and the group inheritance process to support flexible administrative access control.

A user identity is a user record that

- assigns each router user a unique ID across the administrative domain
- limits passwords and one-way encrypted secrets to a maximum of 253 characters, and
- associates each user with at least one user group, enabling attributes such as task IDs.

Additional reference information

User identities are central to managing access and authorization in Cisco IOS XR. Each identity ties the user to permission sets, ensuring secure and auditable access control.

- A user named “admin” may belong to the “network-admins” group, granting permission to perform high-level tasks.
- A user with a unique ID "operator1" may only access operational commands, based on assigned attributes.

Counter-examples

- Guest accounts without group assignment cannot perform authorized tasks.
- Users with passwords exceeding 253 characters cannot be created.

Router user categories for AAA administrative access

Lists the router user categories and the root system user behavior that Cisco IOS XR AAA uses for administrative access.

Router users are classified by Cisco IOS XR AAA as follows:

- Root system user: Owns the entire router chassis, has the highest privileges for all router components, and can monitor all secure domain routers in the system.
- Root Secure Domain Router (SDR) user: Has administrative authority for a specific SDR.
- SDR user: Has user access for a specific SDR.

Table 13: User categories

User category	Access scope
Root system user	Owns the entire router chassis, has the highest privileges over all router components, and can monitor all secure domain routers in the system.
Root Secure Domain Router (SDR) user	Has administrative authority for a specific SDR.
SDR user	Has user access for a specific SDR.

At least one root system user account must be created during router setup. Multiple root system users can exist.

Use the user category to determine the administrative scope that applies after AAA authenticates the user.

User groups for AAA services

Explains local and external user group behavior, including how user groups connect router users to task permissions and remain independent from external server groups.

A user group is a group object that

- provides predefined user groups with defined attributes in Cisco IOS XR
- allows administrators to configure user-defined user groups to meet operational needs, and
- maintains independence between external TACACS+ or RADIUS user groups and local AAA database user groups on the router.

External user group behavior

User groups created in external servers are not related to the user group concept used in local AAA database configuration on the router. The management of external TACACS+ or RADIUS server user groups is independent, and the router does not recognize the user group structure. Remote user or group profiles may contain attributes specifying the groups (defined on the router) to which users belong, as well as individual task IDs. For more information, see the *AAA password security and authorization policies* chapter.

Configuration of user groups in external servers depends on the design of individual server products. Refer to the appropriate server product documentation for details.

Predefined user groups

Lists the predefined Cisco IOS XR user groups and the operational privileges that each group provides for common administrative roles.

Cisco IOS XR software provides predefined user groups whose attributes are already defined. Each group fulfills a particular administrative function.

Table 14: Predefined user groups

User group	Description
cisco-support	Used by the Cisco support team.
maintenance	Displays, configures, and executes commands for network, file, and user-related entities.
netadmin	Controls and monitors all system and network parameters.
provisioning	Displays and configures network, file, and user-related entities.
read-only-tg	Performs any show command but has no configuration ability.
retrieve	Displays network, file, and user-related information.
root-lr	Controls and monitors the specific secure domain router.
sysadmin	Controls and monitors all system parameters but cannot configure network protocols.
serviceadmin	Performs service administration tasks, for example Session Border Controller (SBC) tasks.

To verify the individual permissions of a user group, assign the group to a user and run the **show user tasks** command.

Task groups for AAA services

Explains how Cisco IOS XR task groups collect permitted task IDs for read, write, execute, and debug actions across local and external AAA configurations.

A task group is a task-permission object that

- defines a collection of permitted task IDs for each action type
- supports read, write, execute, and debug permissions on task IDs, and
- allows configuration of task IDs both locally and from external AAA servers.

Additional reference information

Users can configure their own task groups to meet particular needs. Task IDs are defined in the router system and may also be configured in external TACACS+ or RADIUS servers. Task ID definitions may need to be supported before task groups are configured in external software.

Predefined task groups

Lists the predefined Cisco IOS XR task groups that administrators typically use during initial AAA configuration, role setup, and first-stage authorization design.

Cisco IOS XR provides predefined task groups that administrators can use, typically for initial configuration.

Table 15: Predefined task groups

Task group	Description
cisco-support	Cisco support personnel tasks.
maintenance	Maintenance team tasks.
netadmin	Network administrator tasks.
operator	Operator day-to-day tasks, for demonstration purposes.
provisioning	Provisioning team tasks.
retrieve	Retrieve team tasks.
root-lr	Secure domain router administrator tasks.
sysadmin	System administrator tasks.
serviceadmin	Service administration tasks, for example SBC tasks.

Use the predefined task group as a starting authorization profile when its task set matches the intended operational role.

How group inheritance works

Describes how Cisco IOS XR user groups and task groups derive attributes from other groups and create a union of inherited permissions.

User groups and task groups can derive attributes from other groups of the same type. Inheritance is dynamic, meaning changes in the inherited group affect the inheriting group even when no explicit re-inheritance occurs.

Summary

The key components involved in the process are:

- Administrator: Configures inheritance relationships between groups.
- User group or task group: Receives attributes from inherited groups and maintains its own set of permissions.
- AAA system: Calculates and updates the union of attributes for each group based on inheritance rules.

Group inheritance in Cisco IOS XR enables user groups and task groups to derive attributes dynamically from other groups, resulting in a union of permissions and roles.

Workflow

These stages describe how group inheritance works.

1. The administrator configures group A to inherit from group B.
2. The AAA system forms the new set of attributes for group A as the union of group A's attributes and group B's attributes.
3. Any change in group B automatically affects group A through the inheritance relationship, even if group A is not explicitly re-inherited.

Result

The inheriting group has its original attributes as well as those derived from the inherited group, ensuring all appropriate permissions and roles are included.

AAA administrative models

Explains AAA administrative models, emphasizing secure practices, database integration, remote configuration elements, server groups, authentication workflows, and essential guidelines for protecting administrative access and handling system reload scenarios.

A AAA administrative model is a secure-domain access model that

- assigns AAA configuration, local user, group, TACACS+, and RADIUS settings to each secure domain router
- provides isolation so users created in one secure domain router cannot access others unless configured in those domains, and
- ensures each router operates within its own secure domain plane for granular control and security.

Caution: Protect administrative access during AAA configuration

Outlines administrative access lockout conditions and safeguards to consider before enabling console authorization or remote server rollout.

To avoid loss of administrative access during AAA configuration, follow these cautionary measures:

- Do not configure authentication using remote AAA servers that may be unavailable, especially for console access.
- The "none" option without any other method list is not supported.
- Avoid configuring command authorization or XR EXEC mode authorization on the console without careful planning. If TACACS+ servers are unreachable or deny all commands, you may be locked out. This is especially true if authentication uses a user not recognized by the TACACS+ server or if command authorization restricts most commands.
- Ensure you log in with appropriate user permissions in the TACACS+ profile before enabling TACACS+ command authorization or XR EXEC mode authorization on the console.

- If permitted by your site's security policy, use the "none" option for command authorization or XR EXEC mode authorization so that the AAA configuration automatically falls back to this method if TACACS+ servers become unreachable.
- Always enable local fallback when configuring AAA. For details, see the *AAA password security and authorization policies* chapter.
- When committing configuration changes, use the "commit confirmed" command instead of "commit" to allow a timed trial and reduce the risk of accidental lockout.

AAA databases

Explains the local and remote AAA database model that stores users, groups, and task information for router access control and fallback behavior.

AAA databases are repositories that

- store users, groups, and task information that control access to the system
- can be configured as either local or remote databases, and
- support different operational and fallback behaviors based on their configuration.

Additional reference information

Local AAA databases reside on the device itself and provide access control directly. Remote AAA databases use external servers, such as RADIUS or TACACS+, to authenticate, authorize, and account for users and tasks. The AAA configuration determines which database is leveraged for a particular authentication or authorization request.

AAA database sources

Provides the local and remote AAA database storage behavior, including SDR scope, encrypted passwords, setup-dialog behavior, and remote server access requirements.

AAA data can be stored locally in a secure domain router or remotely in an external security server.

Table 16: AAA database source behavior

Source	Behavior
Local database	Stores users, user groups, and task groups locally within an SDR. Data is stored in the in-memory database and persists in the configuration file. The stored passwords are encrypted.
Local database SDR scope	Data is local to the SDR where it is stored. Defined users or groups are not visible to other SDRs in the same system.
Local database setup dialog	If all users are deleted, the setup dialog appears and prompts for a new username and password at the next console login. The setup dialog appears only for console login.
Remote database	Stores AAA data in an external security server, such as CiscoSecure ACS. A client such as a network access server can use the data when it has the server IP address and shared secret.

Use the source behavior details to choose the primary AAA database and appropriate local fallback behavior for your deployment.

Remote AAA configuration elements

Provides the foundational remote AAA configuration facts for shared external databases, security protocol use, client configuration, user groups, and task groups.

Products such as CiscoSecure ACS can administer the shared or external AAA database.

Table 17: Remote AAA configuration elements

Element	Requirement or behavior
Remote server protocol	The router communicates with the remote AAA server by using a standard IP-based security protocol, such as TACACS+ or RADIUS.
Client configuration	Configure the security server with the secret key shared with the router and with the client IP addresses.
External user groups	External TACACS+ or RADIUS user groups are managed independently. The router does not recognize the external user group structure.
Remote profiles	Remote user or group profiles can contain attributes that specify router-defined groups and individual task IDs.
External task groups	Task ID definitions may need external software support before external task groups are configured. Task IDs can also be configured in external TACACS+ or RADIUS servers.

Additional information

Use the server product documentation for external user group configuration, as this configuration is specific to each server product design.

AAA method lists and server groups

Provides the foundational facts for AAA method lists, default database selection, server grouping, and TACACS+ fallback behavior for login and other AAA-enabled applications.

The following table describes the core elements of AAA configuration:

Table 18: AAA configuration elements

Element	Behavior
Method lists	Method lists define the preferred order for AAA data sources. AAA can define more than one method list, and applications such as login can select a list.
Default method list behavior	Console ports can use one method list while vty ports use another. If a method list is not specified, the application tries the default method list. If no default method list exists, AAA uses the local database.

Element	Behavior
Server grouping	Administrators can form server groups for AAA protocols such as RADIUS and TACACS+ and associate them with AAA applications such as PPP and XR EXEC mode.
TACACS+ server-group fallback	If one TACACS+ server in a group is unreachable, the system uses the next server in that group. If the shared secret does not match, the router bypasses that group and uses the next configured method.

Use method lists and server groups to keep different access applications aligned with the intended AAA source order.

How AAA rollover works

Describes how Cisco IOS XR uses prioritized AAA database methods and when it moves to the next method or rejects a request.

AAA can use a prioritized list of database options for authentication, authorization, and accounting requests.

Summary

The key components involved in the process are:

- AAA subsystem: Handles authentication, authorization, and accounting requests.
- Database methods: Includes local, TACACS+, RADIUS, line, and none as options for processing requests.
- Prioritized method list: Determines the order in which methods are attempted.

AAA uses a prioritized list of database methods for authentication, authorization, and accounting. If a method is unavailable, AAA automatically rolls over to the next method. If any method rejects a request, AAA stops the rollover and rejects the request.

Workflow

These stages describe how AAA rollover works.

1. AAA receives an authentication, authorization, or accounting request.
2. AAA tries the first configured method in the prioritized method list.
3. If the method is unavailable, AAA rolls over to the next configured method.
4. If a method rejects the request, AAA does not roll over and rejects the request.
5. Available methods include local, TACACS+, RADIUS, line, and none. Local does not apply to accounting and certain authorization types. Line applies only to authentication. None does not apply to authentication.

Result

When authorization rejection locks out a user, you may need to roll back the previous configuration or remove the problematic AAA configuration through the auxiliary port. Use the local username and password for auxiliary-port login, and run the **config_rollback -n 0x1** command to revert to the previous configuration. If auxiliary-port access is unavailable, a router reload might be required.

How AAA authentication works

Describes how Cisco IOS XR authenticates users, determines their role, obtains task IDs, and applies changed attributes to later sessions.

Authentication is the security process by which a principal, such as a user or application, obtains access to the system.

Summary

The key components involved in the process are:

- User: Provides credentials to request access and is assigned a role.
- AAA system: Authenticates credentials and determines user roles and task group membership.
- SDR user groups: Define user roles and access permissions within the system.
- Clients (XR EXEC mode or external API clients): Use task IDs to optimize operations and enforce available commands or GUI menus.

AAA authentication is the security mechanism by which a principal, such as a user or application, obtains access to the system. AAA verifies credentials, determines roles and access levels, and associates users with permitted tasks.

Workflow

These stages describe how AAA authentication works.

1. A user requests authentication by providing a username and password or secret.
2. AAA verifies the user's password and rejects the user if the password does not match the database value.
3. AAA determines whether the user role is root SDR user or SDR user.
4. If the user is configured as a member of an owner SDR user group, AAA authenticates the user as an owner SDR user.
5. If the user is not configured as a member of an owner SDR user group, AAA authenticates the user as an SDR user.
6. Clients can obtain the user's permitted task IDs by forming a union of all task group definitions in the user's user groups.
7. XR EXEC mode and external API clients can use the task-ID set to optimize operations, such as hiding unavailable commands or disabling unavailable GUI menus.

Result

When user attributes or task permissions change, modifications do not affect the user's current active session. The changes take effect in the user's next session.

Log in to the auxiliary port after system reload with remote AAA authentication

Outlines auxiliary-port login behavior and recovery steps after a reload when TACACS+ or RADIUS authentication is configured for AAA.

If you configure TACACS+ or RADIUS authentication for AAA and reload the system, your first login attempt to the RP auxiliary port may fail—even when you enter a valid TACACS+ username and password.

This happens because, immediately after a reload, the auxiliary port rejects TACACS+ login attempts. To recover, follow these steps:

- Log in with a valid locally configured username and password for your initial login after reload.
- After successfully logging in locally, you can use your TACACS+ username and password for subsequent logins.
- If you access the auxiliary port via a terminal server, clear the terminal server line first. Then log in using your TACACS+ username and password.

Sample AAA configuration:

```
aaa authentication login default group tacacs+ group radius local
line template aux
login authentication default
```

Configure the first router user

Provides instructions for creating the initial local user account, enabling administrators to establish secure access before further AAA configuration.

Create the first root-system user to establish administrative access for the initial management session.

When a Cisco router boots for the first time, you must configure a root-system username and password. This step is necessary to access IOS XR and manage the router.

Before you begin

Use the console port for the initial login dialog.

Procedure

1. Establish a connection to the console port.

   Example

   ```
   Enter root-system username
   ```

   This connection initiates communication with the router. When the connection succeeds, the router displays the prompt.

2. Type the username for the root-system login and press Enter.

   The router sets the root-system username that you use to log in to the router.

3. Type the password for the root-system login and press Enter.

   Example

   ```
   Enter secret
   ```

   The router creates an encrypted password for the root-system username. The password must be at least six characters long.

4. Retype the password for the root-system login and press Enter.

   Example

   ```
   Enter secret again
   ```

   The router verifies that both password entries match.

   > **Note**
   >
   > If the passwords do not match, the router prompts you to repeat the process.

5. Log in to the router.

   The login establishes your access rights for the router management session.

> **Note**
>
> If the router reloads and no username and password are stored, you must create a new username and password.

You can log in to the router and reach the IOS XR prompt. The first-user configuration is complete.

The following example shows the root-system username and password configuration for a new router, and it shows the initial login:

```
Administrative User Dialog
Enter root-system username: cisco
Enter secret:
Enter secret again:

Router#:Jan 10 12:50:53.105 : exec[65652]: %MGBL-CONFIG-6-DB_COMMIT :
'Administration configuration committed by system'.
Use 'show configuration commit changes 2000000009' to view the changes. Use
the 'admin' mode 'configure' command to modify this configuration.
```

```
User Access Verification
Username: cisco
Password:
```

The "secret" line in the configuration script indicates that the password is encrypted. When you type the password during configuration and login, it is hidden for security.

Configure a task group

Guides users through the process of setting up task groups for AAA services, including configuring associated user groups, avoiding reserved usernames, and creating local AAA users to streamline administrative task assignment.

Create a task group to define permissions for AAA operations by associating task IDs.

AAA task groups let you assign specific permissions (read, write, execute, debug) to users through task IDs. Task IDs define which operations a user can perform. Users are placed in user groups that reference task groups, giving them the set of permissions you configure.

Before you begin

- Review the list and purpose of available AAA task IDs on your router.
- Use the **show aaa task supported** command to display all supported task IDs.
- Only users with write permission for the AAA task ID can configure task groups

Procedure

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Create a task group.

Example

```
Router(config)# taskgroup beta
```

Creates a name for a particular task group and enters task group configuration submode.

- Specific task groups can be removed from the system by specifying the **no** form of the **taskgroup** command.

3. Add a description for the task group.

Example

```
Router(config-tg)# description this is a sample task group description
```

(Optional) Creates a description of the task group named in Step 2.

4. Add task IDs with the required permission to the task group.

Example

```
Router(config-tg)# task read bgp
```

Specifies a task ID to be associated with the task group named in Step 2.

- Assigns read permission for any CLI or API invocations associated with that task ID and performed by a member of the task group.
- Specific task IDs can be removed from a task group by specifying the **no** prefix for the **task** command.

After configuring all required task groups, proceed to set up user groups as described in the "Configuring User Groups" section.

Configure a user group

Configure a user group and associate it with task groups on Cisco IOS XR.

Create a user group with specified permissions by associating it with one or more task groups.

User groups allow you to manage permissions for multiple users efficiently. Task groups define sets of allowed actions. Only users with the WRITE:AAA task ID can configure user groups. User groups cannot inherit properties from predefined groups such as owner-sdr. Deleting a referenced user group triggers a warning.

Before you begin

- Ensure you have WRITE:AAA privileges.
- Review task groups to determine which permissions should be inherited.

Procedure

1. Enter global configuration mode

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create a user group.

   Example

   ```
   Router(config)# usergroup beta
   ```

   Creates a name for a particular user group and enters user group configuration submode.

   - Specific user groups can be removed from the system by specifying the **no** form of the **usergroup** command.

3. (Optional) Add a description to the user group.

   Example

   ```
   Router(config-ug)# description this is a sample user group description
   ```

   (Optional) Creates a description of the user group named in Step 2.

4. (Optional) Inherit permissions from another user group.

   Example

   ```
   Router(config-ug)# inherit usergroup sales
   ```

   - Explicitly defines permissions for the user group.

5. Associate the user group with one or more task groups

   Example

   ```
   Router(config-ug)# taskgroup beta
   ```

   Associates the user group named in Step 2 with the task group named in this step.

   - The user group takes on the configuration attributes (task ID list and permissions) already defined for the entered task group.
   - Repeat this step for each task group to assign.

The user group configuration is completed and committed. The group inherits permissions from specified task groups and user groups as configured.

Avoid reserved local usernames

Outlines the reserved local username values you must avoid when configuring Cisco IOS XR AAA users that synchronize to Linux infrastructure.

Do not use the following reserved words as local usernames:

- backup
- bin
- bind
- daemon
- dhcp
- games
- gnat
- irc
- ip
- list
- mail
- man
- messagebus
- news
- nobody
- proxy
- rpc
- root
- sys
- sync
- systemd-timesync
- systemd-network
- systemd-bus-proxy
- sshd
- uucp
- www-data

Configure a local AAA user

Configure a local AAA user, assign them to at least one user group, and commit the changes to Cisco IOS XR.

Create or update a local user who can authenticate to the router and receive task permissions based on user group membership.

Each user is uniquely identified by a username in the administrative domain. Membership in a user group provides task permissions. From Cisco IOS XR Software Release 24.3.1 and later, the router can synchronize up to 100 valid Linux-compatible users to the Linux infrastructure and up to 20 users to the standby route processor in dual-RP routers.

Before you begin

Create the task groups and user groups you want to assign to the user before configuring the user account.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Specify the username for the new user or update an existing user.

   Example

   ```
   Router(config)# username user1
   ```

   Creates a name for a new user (or identifies a current user) and enters username configuration submode.

   - The username must be a single word—no spaces or quotation marks are allowed.

3. Set a password or secret for the user.

   Example

   ```
   Router(config-un)# password 0 pwd1
   ```

   Example

   ```
   Router(config-un)# secret 0 sec1
   ```

   - **password {0 | 7} password**
   - **secret {0 | 5 | 8 | 9 | 10} secret**

   Specifies a password for the user named in step 2.

   - Use the **secret** command to create a secure login password for the user names specified in step 2.
   - Entering **0** following the **password** command specifies that an unencrypted (clear-text) password follows. Entering **7** following the **password** command specifies that an encrypted password follows.
   - Entering **0** following the **secret** command specifies that a secure unencrypted (clear-text) password follows. Entering **5** following the **secret** command specifies that a secure encrypted password follows.
   - Type 0 is the default for the **password** and **secret** commands.

4. Assign the user to at least one user group.

Example

```
Router(config-un)# group sysadmin
```

- The user inherits all attributes of the specified user group. The user takes on all attributes of the user group, as defined by that user group's association to various task groups.
- Each user must be assigned to at least one user group. A user may belong to multiple user groups.
- Repeat step 4 for each user group to be associated with the user specified in step 2.

The user configuration is complete. The user account is created or updated with a password or secret, assigned to at least one user group, and changes are successfully committed.

6 AAA Password Security and Authorization Policies

Topics:

- AAA password types
- AAA password security for FIPS compliance
- Task-based authorization
- AAA method lists
- Command accounting methods
- Local command authorizations with user accounts
- Configuration patterns for AAA services

Outlines AAA password security, FIPS compliance measures, method lists, task-based authorization, command accounting, and configuration patterns to enhance access control and policy enforcement in network environments.

AAA password types

Introduces different AAA password types, covering Type 8, Type 9, and Type 10 password encryption, configuration steps for each, Type 7 password masking, and the behavior and deprecation of older password types in secure environments.

An AAA password type is a credential storage method that determines whether Cisco IOS XR can decrypt or only verify a stored user password.

- Encrypted password types protect stored local user credentials.
- One-way encrypted secrets are preferred for user login accounts because the original password string cannot be deduced from the stored secret.
- Two-way passwords remain necessary for applications such as PPP that must decrypt the stored password for protocol operation.
- When both a secret and a password exist for a user, the secret takes precedence for operations that do not require a decryptable password.

Password type behavior

In configuring a user and that user's group membership, you can specify two types of passwords: encrypted or clear text.

The router supports both two-way and one-way (secret) encrypted user passwords. Secret passwords are ideal for user login accounts because the original unencrypted password string cannot be deduced on the basis of the encrypted secret. Some applications (PPP, for example) require only two-way passwords because they must decrypt the stored password for their own function, such as sending the password in a packet. For a login user, both types of passwords may be configured, but a warning message is displayed if one type of password is configured while the other is already present.

If both secret and password are configured for a user, the secret takes precedence for all operations that do not require a password that can be decrypted, such as login. For applications such as PPP, the two-way encrypted password is used even if a secret is present.

Type 8 and Type 9 password encryption

Explains the Type 8 SHA256 and Type 9 sCrypt hashing methods that Cisco IOS XR supports for local user secret configuration.

A Type 8 or Type 9 password encryption method is a one-way hashing method that stores a local user secret without preserving the original cleartext value.

- Type 8 uses SHA256 hashing.
- Type 9 uses sCrypt hashing.
- The router can store an already encrypted value or internally encrypt a cleartext value by using the selected method.

Encryption method behavior

This feature provides the options for Type 8 and Type 9 encryption methods in AAA security services. The Type 8 and Type 9 encryption methods enable more secure and robust support for saving passwords with respect to each username. Thus, in scenarios where a lot of confidential data need to be maintained, these encryption methods ensure that the admin and other user passwords are strongly protected.

The implementation of Type 8 encryption method uses SHA256 hashing algorithm, and the Type 9 encryption method uses scrypt hashing algorithm.

Configure Type 8 and Type 9 passwords

Configure Type 8 and Type 9 user secrets directly or by entering cleartext that Cisco IOS XR encrypts using your selected hashing method. This enables secure storage and compliance with password requirements.

Configure user secrets with Type 8 (SHA256) or Type 9 (sCrypt) hashing when these algorithms meet your local password-storage requirements.

When configuring a password, you can:

- Provide an already encrypted value, which is stored directly in the system without further encryption.
- Provide a cleartext password, which Cisco IOS XR internally encrypts and stores using your chosen method.

Type 5, Type 8, Type 9, and Type 10 encryption methods all support these options.

For more information about configuring users with Type 8 and Type 9 encryption methods, see *Configure Users* section.

Before you begin

Create the local user profile before you configure the secret value.

Procedure

1. Configure Type 8 and Type 9 passwords directly, or enter cleartext passwords for the router to encrypt.

   To configure a user with a pre-encrypted Type 8 password:

   Example

   ```
   Router(config)# username demo8
   Router(config-un)# secret 8
   $8$dsYGNam3K1SIJO$7nv/35M/qr6t.dVc7UY9zrJDWRVqncHub1PE9UlMQFs
   ```

   The router stores your encrypted secret without further processing.

   To configure a user with a cleartext password (Type 8) for internal encryption:

   Example

   ```
   Router(config)# username demo8
   Router(config-un)# secret 0 enc-type 8 PASSWORD
   ```

   The password is encrypted using Type 8 and stored.

   To configure a user with a pre-encrypted Type 9 password:

   Example

   ```
   Router(config)# username demo9
   Router(config-un)# secret 9
   $9$nhEmQVczB7dqsO$X.HsgL6x1il0RxkOSSvyQYwucySCt7qFm4v7pqCxkKM
   ```

   The router stores your encrypted secret without further processing.

   To configure a user with a cleartext password (Type 9) for internal encryption:

Example

```
Router(config)# username demo9
Router(config-un)# secret 0 enc-type 9 PASSWORD
```

The password is encrypted using Type 9 and stored.

2. Configure masked-secret password entry for Type 5, Type 8, Type 9, or Type 10 authentication.

   To enter a masked-secret password (which hides your entry for security), use:

   Example

   ```
   Router(config)# username us6 masked-secret 0 enc-type 8

   Enter secret:
   Re-enter secret:

   Router(config)# commit
   ```

   The entered secret is encrypted and stored securely; if your entries do not match, an error is shown. *View the encrypted secret:*

   Example

   ```
   Router# show running-config aaa
   ..
   username us6
    secret 8 $8$mlcSk/Ae5Qu/5k$RJdI3SQ8B4iP7rdxxQvVlJVeRHSubZzcgcaLYxjg36s
   ```

   To use masked-secret for Type 8 secret authentication (encrypted password from earlier):

   Example

   ```
   Router(config)# username us6 masked-secret 8

   Enter secret:
   Re-enter secret:

   Router(config)# commit
   ```

   When you key in a password, to ensure that it is not displayed on the screen, use the **masked-secret** option. Steps:

   Use the **username** command as shown below, and enter the password.

   The following command contains the username us6, 0 to specify a cleartext password, and the encryption type (5, 8, 9, or 10).

   View the encrypted secret:

   Enter the username, 8 to specify Type 8 secret authentication, and enter the Type 8 secret. You can also use a secret encrypted earlier.

   If there is a password mismatch between the two entries, an error message is displayed.

The user secret is stored by using the selected Type 8 or Type 9 hashing method.

Type 10 password encryption

Explains Type 10 SHA512 password encryption for local user management and the synchronization behavior between XR VM and System Admin VM.

A Type 10 password encryption method is a SHA512-based hashing method that stores local user secrets for Cisco IOS XR user management.

- Type 10 is the default encryption method when a user secret is configured with a cleartext value.
- Type 10 secrets synchronize from XR VM to System Admin VM when the System Admin VM has the same user.
- The Type 10 password is applied on the System Admin VM only when the username exists there.

Type 10 behavior

The Cisco IOS XR 64 bit software supports Type 10 (**SHA512**) encryption algorithm for passwords used in user management. With this feature, **SHA512** is used by default for the passwords in user name configuration. This is applicable even for the first user creation. The **SHA512** encryption algorithm provides improved security to the user passwords compared to the older algorithms such as **MD5** and **SHA256** .

Restrictions for type 10 password encryption usage

The usage of Type 10 password encryption is subjected to this restriction:

- In a first user configuration scenario or when a user is reconfigured, only the Type 5 and Type 10 encryption are synced from XR VM to System Admin VM and Host VM; Type 8 and Type 9 are not synced.

Configure Type 10 password encryption

Configure Type 10 SHA512 user secret encryption explicitly or let Cisco IOS XR apply Type 10 encryption to a cleartext user secret by default.

Configure user secrets with the default Type 10 SHA512 hashing method to ensure stronger local credential storage.

Use these options to configure Type 10 **sha512** password encryption for a user on a Cisco IOS XR device.

Before you begin

Review the Type 10 password encryption behavior before you configure the user secret.

Procedure

1. Configure the Type 10 user secret.

Example

```
Router#configure
Router(config)#username user10 secret testpassword
Router(config-un)#commit
```

Example

```
Router#configure
Router(config)#username root secret 10
$6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
Router(config-un)#commit
```

Example

```
Router#configure
Router(config)#username user10 secret 0 enc-type 10 testpassword
Router(config-un)#commit
```

Example

```
Router#show run aaa
!
username user10
secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
!
```

Example

```
Router(config)#username user10 secret 10
$6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
Router(config-un)#commit
```

The Type 10 encryption is applied by default when you create a user with a clear-text password.

Also, a new parameter '10' is available for the **secret** option under the **username** command to explicitly configure Type 10 encryption.

In scenarios where you have to enter the clear-text password, you can specify the encryption algorithm to be used by using the **enc-type** keyword and the clear-text password as follows:

The above configuration returns the encrypted password using Type10 algorithm (use the **show run username** command to verify that) which can then be configured for the user as follows:

2. Review the Type 10 running configuration.

Example

```
Router#show run username user10
!
username user10
secret 10 $6$9UvJidvsTEqgkAPU$3CL1Ei/F.E4v/Hi.UaqPrvJWf1
!
```

Deprecated Type 7 password and Type 5 secret behavior

Provides deprecation behavior for Type 7 passwords and Type 5 secrets, including CLI option changes, upgrade behavior, downgrade behavior, and special-case handling.

Password configuration options before release 24.4.1

Until Release 24.4.1, there were two options for configuring a password:

- Password: Uses Type 7 encryption to store the password.
- Secret: Supports Type 5, 8, 9, or 10 hashing algorithms to store the password securely.

Deprecation notice

Starting from the Release 24.4.1, the use of Type 7 password and Type 5 secret are deprecated due to security concerns. The deprecation process commences from the Release 24.4.1. We expect the full deprecation in a future release. We recommend using the default option, which is Type 10 secret.

Note

With the deprecation of Type 7 password encryption in Cisco IOS XR Release 24.4.1, any configuration that used Type 7 passwords will be automatically converted and saved as Type 10 secrets during the upgrade to version 24.4.1.

If you have usernames that include both a password and a secret, then:

For the first 3000 users, the router will retain the original secret and discard the password.

For users beyond the first 3000, the router will convert the password as Type 10 secrets by overwriting the original secret.

Password

The password options available in the router from the Release 24.4.1:

```
RP/0/RP0/CPU0:ios(config-un)#password ?
LINE The type 7 password followed by '7 ' OR SHA512-based password (deprecated,
 use 'secret')
```

Changes:

- All the options that were present until the Release 24.4.1 are removed except LINE (to accept cleartext).
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.

  Post-upgrade: You can still use the Type 7 password configurations option after new commits, but the password will be stored as Type 10 secret.
- New syslog has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is
deprecated.
 Converting it to a Type 10 secret for user <user name>.
```

- **show running configuration** command output before upgrade:

```
username example
password 7 106D000A0618
!
```

show running configuration command output post-upgrade:

```
username example

secret 10
$6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iC1OryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqp8st.
!
```

Masked-password

The masked-password options available in CLI from the Release 24.4.1:

```
Router(config-un)# masked-password ?
0 Specifies a cleartext password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 7 and encrypted that were present until the Release 24.4.1 are removed.
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.
- Post-upgrade: Masked-password is an alternate method of configuring the password. You can still use the masked-password keyword with a clear string after new commits, but the password will be stored as Type 10 secret.
- New syslog has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is
deprecated.
 Converting it to a Type 10 secret for user <user name>.
```

- **show running configuration** command output before upgrade:

```
username example
password 7 106D000A0618
!
```

show running configuration command output post-upgrade:

```
username example

secret 10
$6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iC1OryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqq8st.
!
```

Password-policy

The password-policy options available in CLI from the Release 24.4.1:

```
Router(config-un)# password-policy ?
WORD Specify the password policy name

Router(config-un)# password-policy abcd password ?
0 Specifies an UNENCRYPTED password will follow
7 Specifies that an encrypted password will follow
LINE The UNENCRYPTED (cleartext) user password
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
encrypted Config deprecated. Will be removed in 7.7.1. Specify '7' instead.
```

Changes:

- All the options that were present until 24.4.1are removed except LINE (to accept cleartext).
- During upgrade: Any configuration using the Type 7 password configuration is automatically converted to Type 10 secret.

Post-upgrade: You can still use the password-policy configurations option after new commits, but the it will be stored as Type 10 secret.

- New syslog has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration is
deprecated.
 Converting it to a Type 10 secret for user <username>.
```

- **show running configuration** f command output before upgrade:

```
username example
password-policy abcd password 7 106D000A0618
!
```

show running configuration command output post-upgrade:

```
username example
secret 10
$6$P53pb/FFxNIT4b/.$yVakako4fp9PZiIYYh1xS0.W6b/yPrSyC8j4gLs6xli57iC1OryPXyN9y8yojRD2nhAWb9pjr/WAIhbXqp8st.
!
!
```

AAA password-policy

The aaa password-policy options available in CLI from the Release 24.4.1:

```
Router(config)# aaa password-policy abcd
RP/0/RP0/CPU0:ios(config-pp)#?
min-char-change Number of characters change required between old and new
passwords (deprecated, will be removed in 25.3.1)
restrict-password-advanced Advanced restrictions on new password (deprecated,
 will be removed in 25.3.1)
restrict-password-reverse Restricts the password to be same as reversed old
password (deprecated, will be removed in 25.3.1)
```

Changes:

- The options min-char-change, restrict-password-advanced, and restrict-password-reverse that were present until the Release 24.4.1 are deprecated.
- During upgrade: These deprecated configurations do not go through any change during upgrade.

  Post-upgrade: These deprecated keywords do not take effect when configured post-upgrade.
- New syslog have been added to indicate the deprecation process:

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy
option 'min-char-change' is deprecated.
Password/Secret will not be checked against this option now.
```

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy
option 'restrict-password-reverse' is deprecated.
 Password/Secret will not be checked against this option now.
```

```
%SECURITY-LOCALD-4-DEPRECATED_PASSWORD_POLICY_OPTION : The password policy
option 'restrict-password-advanced' is deprecated.
Password/Secret will not be checked against this option now.
```

- **show running configuration** command output before upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

show running configuration command output post-upgrade:

```
aaa password-policy abcd
lower-case 3
min-char-change 1
restrict-password-reverse
restrict-password-advanced
!
```

Secret

The secret options available in CLI from the Release 24.4.1:

```
Router(config-un)# secret ?
0 Specifies a cleartext password will follow
10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
LINE The cleartext user password
```

```
Router(config-un)# secret 0 enc-type ?
<8-10> Specifies which algorithm to use. Only 8,9,10 supported [Note: Option
'5' is not available to use from 24.4]
```

Changes:

- The options 5 and encrypted are removed.
- During upgrade: Configurations using Type 5 secret will remain unchanged.

  Post-upgrade: Though the keyword 5 has been deprecated, you can still apply the existing configurations using Type 5 secret.
- New syslog has been added to indicate the deprecation process:

```
%SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
Please use the 'secret' keyword with option type 10 for user.
```

- **show running configuration** command output before upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

show running configuration

command output post-upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
```

```
!
!
```

Masked-secret

The masked-secret options available in CLI from the Release 24.4.1:

```
Router(config-un)# masked-secret ?
0 Specifies a cleartext password will follow

10 Specifies that SHA512-based password will follow
8 Specifies that SHA256-based password will follow
9 Specifies that Scrypt-based password will follow
clear Config deprecated. Will be removed in 7.7.1. Specify '0' instead.
<cr> The cleartext user password
```

Changes:

- The options 5 and encrypted are removed.
- During upgrade: Configurations using masked-secret with Type 5 will remain unchanged.

  Post-upgrade: Though the keyword 5 has been deprecated, you can still apply the existing configurations using Type 5 masked secret.
- New syslog has been added to indicate the deprecation process:

```
%SECURITY-LOCALD-2-DEPRECATED_SECRET_TYPE : Type 5 secret is deprecated.
Please use the 'secret' keyword with option type 10 for user.
```

- **show running configuration** command output before upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

  show running configuration command output post-upgrade:

```
username example
secret 5 $1$kACo$2RtpcwyiRuRB/DhWzabfU1
!
!
```

Special use cases

Use case 1: Configurations using both Type 7 password and secret with 8, 9, or 10 hashing, for the same user

- During upgrade:
  - For the first 3000 username configurations, the password configuration will be rejected, and the secret configuration will remain unchanged.
  - For the rest of the username configurations, the original secret configuration will be rejected, and the password will be converted to Type 10 secret.
- Post-upgrade:
  - For a new username configured, or the username that is already present before the upgrade, the password configuration will be rejected.

- New syslog has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-SECRET_CONFIG_PRESENT : The password configuration is
deprecated.
Once secret is configured, cannot use password config for user <user name>
 at index <x> now.
```

  where 'x' is a number representing the index.

Use case 2: Configurations using both Type 7 password and Type 5 secret, for the same user

- During upgrade:
  - For any username configuration, the original Type 5 secret configuration will be rejected, and the password will be converted to Type 10 secret.
- Post-upgrade:
  - For a new username configured, or the username that is already present before the upgrade, the password configuration will be converted to Type 10 secret.
  - New syslog has been added to indicate the deprecation process:

```
%SECURITY-PSLIB-4-DEPRECATED_PASSWORD_TYPE : The password configuration
is deprecated.
 Converting it to a Type 10 secret for user <username>.
```

Type 7 password masking

Explains the masked-password option that hides entered password characters during Type 7 password authentication configuration and shows how the encrypted value is stored.

A Type 7 password masking method is a password-entry method that hides password text from the terminal during Type 7 password authentication configuration.

- The masked-password option prevents the entered password from appearing on the screen.
- The running configuration stores the resulting password in encrypted Type 7 form.
- The router displays an error message if the two password entries do not match.

Feature history

The feature history table lists release support for this feature.

Table 19: Feature History Table

Feature Name	Release Information	Feature Description
Password Masking	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) * This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Password Masking	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) * This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Password Masking	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) * This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Password Masking	Release 7.3.1	With this feature, when you key in a password or secret, it is not displayed on the screen. This enhances security. The feature is enabled by default. The following options are added to the **username** command: • **masked-password** • **masked-secret**

Type 7 password masking behavior

When you key in a password, to ensure that it is not displayed on the screen, use the **masked-password** option. Details:

Use the **username** command as shown below, and enter the password.

The following command contains the username us3, and 0 to specify a cleartext password.

```
Router(config)# username us3 masked-password 0

Enter password:
Re-enter password:

Router(config)#commit
```

View the encrypted password:

```
Router# show run aaa
..

username us3
 password 7 105A1D0D
```

Enable Type 7 password authentication and enter the encrypted password 105A1D0D. You can also use a password encrypted earlier.

```
Router(config)# username us3 masked-password 7

Enter password:
Re-enter password:


Router(config)#commit
```

If there is a password mismatch between the two entries, an error message is displayed.

Configure Type 7 password masking

Configure Type 7 password masking so that entered password characters do not appear on the terminal screen during password authentication setup.

Hide typed password characters while configuring a Type 7 password for local user authentication.

Use the masked-password option when you enter a password that must not be displayed on the screen.

Before you begin

Review the deprecation guidance for Type 7 password configuration before you use this task.

Procedure

1. Configure a masked cleartext password for the user.

   Example

   ```
   Router(config)# username us3 masked-password 0

   Enter password:
   Re-enter password:

   Router(config)#commit
   ```

   - When you enter the password, use the masked-password option to ensure it is not displayed on the screen.
   - Use the username command as shown and enter the password. The example above uses username "us3" and 0 to specify a cleartext password.

2. Review the encrypted password in the running configuration.

   Example

   ```
   Router# show run aaa
   ..
   ```

```
username us3
 password 7 105A1D0D
```

Use the running configuration to confirm the password is stored as an encrypted Type 7 value.

3. Configure Type 7 password authentication with the encrypted password.

Example

```
Router(config)# username us3 masked-password 7

Enter password:
Re-enter password:


Router(config)#commit
```

You can use the encrypted password previously generated by the router or another encrypted password.

4. Confirm password entry behavior.

View the encrypted password in the running configuration.

The user's password is configured, and the running configuration stores the encrypted Type 7 value.

AAA password security for FIPS compliance

Explains password security requirements for FIPS compliance, highlighting configuration attributes, mandatory password lengths, secret policies, enforcement order, user authentication feedback, and consecutive-character restrictions, with guidance for configuring compliant password policies.

A AAA password security mechanism is a local user password policy that

- stores, retrieves, and validates passwords against administrator-defined security rules
- applies only to local users, not to remote users whose profiles are maintained in third-party AAA servers, and
- supports Cisco IOS XR 64-bit platforms only on XR VM.

Additional reference information

High availability behavior

Cisco IOS XR Software introduces advanced AAA password strengthening policies and security mechanisms to store, retrieve, and provide policy rules for user passwords. These policies are applicable only to local users, not to remote users whose profile information is stored in a third-party AAA server. The policy does not apply to user secrets unless user secret policy support is configured separately. If both a secret and a password are configured for a user, the secret takes precedence, and password security policy does not affect authentication or password changes for such users.

The AAA password policy and username configurations persist across RP failovers or process restarts. However, operational data such as password lifetime and user lockout time are not stored in the system database or on disk, so these settings are not restored after failovers or restarts. As a result, users who were previously locked out can log in immediately after an RP failover or process restart.

To configure AAA password policy, see *Configure AAA password policy*.

AAA password security configuration attributes

Provides the password composition, length, lifetime, expiry, change, re-authentication, lockout, and policy lifecycle behavior used by AAA password security configuration.

- AAA password security for FIPS compliance consists of policies that control password composition, length, lifetime, expiry, change behavior, service provisioning, re-authentication, lockout, and policy lifecycle operations.
- Use this reference to choose the policy attributes that a local user password must satisfy and to understand how policy changes affect users.

Here are the key policy attributes and behaviors for AAA password security (FIPS compliance):

Password composition policy

- Passwords can include any combination of uppercase and lowercase letters, numbers, and special characters: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")".
- Administrators may specify the types and the number of required characters, allowing flexible composition rules.
- The default minimum character change between passwords is 4.
- No restriction applies to the maximum number of uppercase, lowercase, numeric, or special characters.

Password length policy

- Administrators can set minimum and maximum password length.
- The minimum length is 2 characters; the maximum is 253 characters.

Password lifetime policy

- Password maximum lifetime can be specified in years, months, days, hours, minutes, or seconds.
- If no lifetime is configured, the password never expires.
- Configuration persists after system reload; password creation time updates upon reboot.
- If a password's lifetime expires, action is taken per the expiry policy.

Password expiry policy

If a user's password has expired:

- The user is prompted to set a new password after successfully entering the expired password.
- The new password is validated against the security policy.
- If compliant, the AAA database updates, and authentication completes with the new password
- If not compliant, the authentication fails and the user is prompted again to set a compliant password; maximum attempts are controlled by login clients (AAA has no restrictions).
- If a lifetime is newly configured for an existing user, it is set in the database and checked during the user's next authentication.
- Expiry is checked only during authentication.

  If a password expires after login, no immediate action is taken; user is prompted at next authentication.
- Debug logs and syslog entries appear only when a login attempt is made.

Example syslog (password expired):

```
Router: Jun 21 09:13:34.241 : locald_DSC[308]:
%SECURITY-LOCALD-5-USER_PASSWD_EXPIRED :
Password for user 'user12' has expired.
```

Password change policy

Users cannot change passwords at will; changes occur when:

- The security administrator changes the password.
- The user is authenticating with an expired profile password.
- The administrator modifies a user's associated password policy, but does not immediately change the password.
- The command **show configuration failed** displays error messages for passwords that do not comply with policy configurations.
- If the policy changes and an existing profile doesn't meet rules, no action is taken for logged in users; the user is prompted to change password at next authentication.
- When a password changes, the new password's lifetime matches the previously set lifetime.
- Password expiry for non-interactive clients (e.g., dot1x) triggers error messages; such clients must contact the security administrator for renewal.

Service provision after authentication

No service is performed before authenticating a user (basic AAA local authentication feature).

User re-authentication policy

When changing the password (especially on expiry), the user is authenticated with the new password. The authentication uses the previous credential, and the new password updates in the database.

User authentication lockout policy

- The **authen-max-attempts** option restricts the maximum permissible failed authentication attempts.
- If the limit is exceeded, the user is locked out until the lockout timer (**lockout-time**) expires.
- Attempted logins during lockout extend the lockout timer from the last attempted login.

Example syslog (user locked out):

```
Router: Jun 21 09:21:28.226 : locald_DSC[308]:
%SECURITY-LOCALD-5-USER_PASSWD_LOCKED :
User 'user12' is temporarily locked out for exceeding maximum unsuccessful
logins.
```

Example syslog (user unlocked):

```
Router: Jun 21 09:14:24.633 : locald_DSC[308]:
%SECURITY-LOCALD-5-USER_PASSWD_UNLOCKED :
User 'user12' is unlocked for authentications.
```

Password policy lifecycle: creation, modification, and deletion

- Administrators with AAA write permission can create password policies.
- Modification can occur at any time, even when the policy is associated with a user.
- Deletion is only permitted if the policy is un-configured from all users.
- After modifying an associated password policy:
  - If the administrator configures the password, the user is not prompted to change password at next login.
  - If not, the user is prompted to change password at next login.
  - In all cases, at each password expiry interval, users are prompted to change passwords at next login.
  - Debug messages are printed for policy creation, modification, and deletion.

Requirement: Use a six-character password for the first user

Outlines the first-user password length requirement that applies during first boot, after reload without username configuration, or after deletion of all username configurations.

You must configure a password with at least six characters when creating the first user on a Cisco router. This applies in the following situations:

- When the Cisco Router is booted for the very first time.
- When the router is reloaded with no username configuration.
- When the already existing username configurations are deleted.

By default, the minimum length for passwords in a Cisco router is limited to two characters. Due to noise on the console, there is a possibility of the router being blocked out. Therefore, the minimum length for password has been increased to six characters for a first user created on the box, in each of the situations described above. This reduces the probability of the router being blocked out. It avoids the security risks that are caused due to very small password length. For all other users created after the first one, the default minimum length for password is still two characters.

For more information about how to configure a first user, see *Configure First User on Cisco Routers*.

User secret policies

Explains how Cisco IOS XR extends password policy support to Type 5, Type 8, Type 9, and Type 10 user secrets for local credential validation.

A user secret policy is a user-authentication policy that

- validates both local and remote user secrets against the same policy used for the user's password
- supports a range of secret types including Type 5 (MD5), Type 8 (SHA256), Type 9 (sCrypt), and Type 10 (SHA512), and
- uses irreversible hashed secrets to prevent modules on the device from retrieving cleartext secrets.

Additional reference information

The Cisco IOS XR Software extends password policy support beyond Type 7 passwords, now applying the same rules to Type 5 (**MD5**), 8 (**SHA256**), 9 (**sCrypt**) and 10 (**SHA512** user secrets for credential validation. The policy is common to both the password and the secret for each user. By using irreversible hashed secrets, the system ensures that other modules within the device cannot access the cleartext version of these secrets, enhancing the security of user credentials. This policy applies to both local and remote users.

Classic Cisco IOS XR platforms support the policy for secrets on the XR and Admin planes. 64-bit Cisco IOS XR platforms support it only on the XR VM, not on the System Admin VM.

To configure a password policy for user secret, see *Configure password policy for user secret and password*.

Requirement: Configure user secret policies in the supported order

Outlines the ordering, hashing, compatibility, downgrade, and model restrictions that apply when you configure password policy for user secrets in username configurations.

You must follow these requirements when configuring password policy for user secrets in username configurations:

- If you have not already configured a policy, the system will not perform any policy validation for the secret. Ensure you configure the policy first and apply it to the username before configuring the secret—especially when copying and pasting username configurations.
- When you change the user secret at login, the system applies the same hashing type as used in the username configuration. For example, if Type 5 hashing was applied, the system uses Type 5 when the secret is modified.
- Password and secret are distinct entities. If the `restrict-old-count` parameter is set while changing a password, the system checks compliance only with the history of old passwords, not old secrets.
- The system checks only old secret history when changing secrets, not old password history. If the same clear-text secret was previously used as a password, the system allows it for secret configuration, and vice versa for password configuration.
- The **restrict-old-count** applies to both secrets and passwords. The new secret or password overwrites the old one in FIFO order.
- If you try to assign a different policy to a username that already has a password or secret associated with a policy, the system rejects the configuration. You must remove the existing password or secret to apply the new policy.
- The router does not allow configuration that requires validating the secret against the previous composition of the clear-text secret, since you cannot retrieve the clear-text secret after it has been hashed. Therefore, these parameters have no effect on secret configuration:
  - **max-char-repetition**
  - **min-char-change**
  - **restrict-password-reverse**
  - **restrict-password-advanced**
- Because the new **policy** configuration for the user is common to passwords and secrets, the existing **password-policy** becomes redundant. These configurations must be mutually exclusive. If one is already present and you try to configure the other, the system rejects the request and reports that **password-policy** and **policy** cannot be used together.

Configure an AAA password policy

Configure an AAA password policy, apply the policy to a local user, and verify the resulting policy attributes on the router.

Create an AAA password security policy so that local user passwords follow the configured length, lifetime, character-change, and lockout rules.

Use the **aaa password-policy** command in global configuration mode to set up password policies for local users.

Before you begin

You must have write permission for AAA tasks.

Procedure

1. Configure the AAA password policy and apply it to a user.

   Example

```
Router(config)#aaa password-policy test-policy
Router(config-aaa)#min-length 8
Router(config-aaa)#max-length 15
Router(config-aaa)#lifetime months 3
Router(config-aaa)#min-char-change 5
Router(config-aaa)#authen-max-attempts 3
Router(config-aaa)#lockout-time days 1
Router(config-aaa)#commit

Router(config)#username user1 password-policy test-policy password 0 pwd1
```

   This configures a password policy named **test-policy** and applies it to the user **user1** using the **username** command with the **password-policy** option.

2. Review the AAA password-policy running configuration.

   Example

```
aaa password-policy test-policy
 min-length 8
 max-length 15
 lifetime months 3
 min-char-change 5
 authen-max-attempts 3
 lockout-time days 1
 !
```

3. Verify the AAA password-policy details.

   Example

```
Router#show aaa password-policy

Password Policy Name : test-policy
  Number of Users : 1
  Minimum Length : 8
  Maximum Length : 15
  Special Character Len : 0
  Uppercase Character Len : 0
  Lowercase Character Len : 1
  Numeric Character Len : 0
  Policy Life Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 3
    years : 0
  Lockout Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 1
```

```
    months : 0
    years : 0
  Character Change Len : 5
  Maximum Failure Attempts : 3
```

Use this command to get details of the AAA password policy configured in the router:

4. Configure password masking for AAA password policies.

   To mask passwords, use the following commands:

   Example

```
Router(config)# aaa password-policy security
Router(config)# username us6 password-policy security masked-password 0

Enter password:
Re-enter password:

Router(config)#commit
```

   To view the encrypted password:

   Example

```
Router# show run aaa
..

aaa password-policy security
..
username us6
 password-policy security password 7 0835585A
```

   You can also use a password encrypted earlier:

   Example

```
Router(config)# username us6 password-policy test-policy masked-password 7

Enter password:
Re-enter password:


Router(config)#commit
```

   When entering a password, use the masked-password option to ensure it is not displayed on the screen. If the two entries do not match, an error message appears.

The password policy is created, associated with the intended user, and confirmed in the password-policy verification output.

Configure a password policy for user secrets and passwords

Configure a password policy that simultaneously governs both passwords and secrets for local user profiles.

Apply a password policy to a user profile to ensure Cisco IOS XR verifies and enforces compliance for password and secret changes.

Cisco IOS XR allows you to configure a password policy that applies to both the user's password and secret. By adding the **policy** option to the **username** command, you ensure that any changes to the password or secret are validated against the specified policy. On Cisco IOS XR 64-bit platforms, password policy enforcement exists only in XR VM; System Admin VM does not enforce password policy. The AAA configuration on System Admin VM uses a YANG model, which may lead to configuration inconsistencies during upgrades or downgrades.

Before you begin

Review *Requirement: Configure user secret policies in the supported order* before you apply the policy to a username configuration.

Procedure

1. Apply the password policy to the user profile.

   Example

   ```
   Router#configure
   Router(config)#username user1
   Router(config-un)#policy test-policy1
   Router(config-un)#secret 10
   $6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZIhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
   Router(config-un)#commit
   ```

   This example shows how to configure a password policy for the user, that applies to both the password and the secret of the user.

2. Review password and secret compliance examples.

   Example

   ```
   username user1
   policy test-policy1
   secret 10
   $6$dmwuW0Ajicf98W0.$y/vzynWF1/OcGxwBwHs79VAy5ZZIhoHd7TicR4mOo8IIVriYCGAKW0A.w1JvTPO7IbZry.DxHrE3SN2BBzBJe0
   !
   ```

   Example

   ```
   username user2
   policy test-policy1
   password 7 09604F0B
   !
   username user3
   policy test-policy1
   secret 10
   $6$U3GZl1VINwJ4Dl1.$8X6av2kQ.AWvMKGEz5ILvZO7OXj6DgeOqLoQKIf7XJxKayViFJNateZ0no6gO6DbbXn4bBo/Dlqitro3jlsS40
   password 7 080D4D4C
   !
   username user4
   secret 10
   $6$mA465X/m/UQ5....$rSKRw9B/SBYC/N.f7A9NCntPkrHXL6F4V26/NIjWXnrSnna03FxW3bcyfDAyveOexJz7/oak0XB6tjLF5CO981
   password-policy test-policy1 password 7 0723204E
   !
   username user5
   password-policy test-policy1 password 7 09604F0B
   !
   ```

Example

```
(1)aaa password-policy pol1
 lifetime minutes 1
 upper-case 1
 restrict-old-count 2
!

username lab2
 group root-lr
(2) policy pol1
(3) secret 10
$6$qphqA0RfEXOn6A0.$wRwWGll0TIpHPdVQ66fUiIM5P46ggoGMGgFuaZd0LD2DLFYDlDPaRyXQLi8Izjb49tC7H7tkTLrc1.GELFpiK.

  password 7 1533292F200F2D
!
```

The below examples show different possible combinations to check for password or secret compliance against the policy:

The compliance check for password or secret in the above examples works as described below:

- When you change the secret for user1, the system checks the secret compliance against the policy, test-policy1.
- When you change the password for user2, the system checks the password compliance against the policy, test-policy1.
- When you change the password or secret for user3, the system checks the password or secret compliance against the policy, test-policy1.
- When you change the secret for user4, the system does not check for compliance against any policy. Whereas, when you change the password for user4, the system checks the password compliance against the policy, test-policy1.
- When you change the password for user5, the system checks the password compliance against the policy, test-policy1.

If you perform configurations in a single commit (for example, via copy-and-paste), the system checks the secret compliance against the assigned policy, but not password compliance during the commit. Configurations can be entered in any order in a single commit.

Failed-authentication username display

Explains how Cisco IOS XR can display the username in failed-authentication system logs for unsuccessful SSH and Telnet login attempts after the feature is enabled.

A failed-authentication username display feature is a logging feature that

- records the entered username in failed-authentication system logs for SSH and Telnet login attempts
- is disabled by default, causing the system to log the username as unknown for these attempts, and
- when enabled, displays the entered username for unsuccessful SSH and Telnet login attempts in the logs.

Feature history
Table 20: Feature History Table

Feature Name	Release Information	Feature Description
Display Username for Failed Authentication for Telnet Protocols	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Display Username for Failed Authentication for Telnet Protocols	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Display Username for Failed Authentication for Telnet Protocols	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Display Username for Failed Authentication for Telnet Protocols	Release 7.10.1	With this feature, we have enhanced the security of the routers and introduced better tracking functionality to the router. The failed authentication sys log now displays the details of users who tried to log in but failed due to authentication failure. With this feature provisioned, the router can now display the user ID of both SSH and Telnet protocols. In earlier releases, this feature was available only for SSH protocols. This feature introduces the following change: **CLI**: **aaa display-login-failed-users**. **YANG DATA Model**: New XPaths for `Cisco-IOS-XR-um-aaa-task-user-cfg` (see Github, YANG Data Models Navigator)

Additional reference information

Effective with Cisco IOS XR Software Release 7.10.1, you can track the username of users whose login attempts failed in the failed authentication system logs. Prior to this release, the feature was only available for SSH clients; now, it is available for both SSH and Telnet clients.

Default behavior: By default, the feature is disabled. When disabled, failed authentication sys logs display the username as **unknown** for both SSH and Telnet. When enabled, failed authentication sys logs display the username of the users who attempted but failed to log in.

To enable this feature, use the **aaa display-login-failed-users** command in XR Config mode.

Enable display of username for failed authentication

Configure failed-authentication system logs to display the username for unsuccessful SSH and Telnet login attempts.

Record the username in system logs when SSH or Telnet authentication fails, allowing you to identify which user attempted to log in unsuccessfully.

By default, failed-authentication logs display the username as unknown for SSH and Telnet clients.

Before you begin

Use XR Config mode to enable the username display feature.

Procedure

1. Enable username display for failed authentication.

Example

```
Router#conf
Router(config)#aaa display-login-failed-users
Router(config)#commit
```

2. Review the running configuration.

Example

```
Router# show run aaa display-login-failed-users
!
aaa display-login-failed-users
!
```

3. Verify the failed-authentication system logs.

Example

```
RP/0/RP0/CPU0:Jul 18 14:46:31.590 UTC: exec[66608]:
%SECURITY-LOGIN-4-AUTHEN_FAILED : Failed authentication attempt by
user lab  from 'console' on 'con0_RP0_CPU0'
```

Example

```
RP/0/RP0/CPU0:Jul 18 14:47:51.590 UTC: ssh_syslog_proxy[1216]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[13519]: Failed
authentication/pam
for lab  from 192.168.122.1 port 44822 ssh2
```

This section shows example from sys logs where the user name is displayed for failed authentication after the configuration of this feature.

```
RP/0/RP0/CPU0:Jul 18 14:46:31.590 UTC: exec[66608]:
%SECURITY-LOGIN-4-AUTHEN_FAILED : Failed authentication attempt by
user lab  from 'console' on 'con0_RP0_CPU0'
```

```
RP/0/RP0/CPU0:Jul 18 14:47:51.590 UTC: ssh_syslog_proxy[1216]:
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[13519]: Failed
authentication/pam
for lab  from 192.168.122.1 port 44822 ssh2
```

Failed-authentication system logs now display the username that was entered during unsuccessful login attempts.

Consecutive-character password restrictions

Explains the AAA password policy rule that restricts consecutive English-alphabet, QWERTY-keyboard, and number sequences in local user passwords and secrets.

A consecutive-character password policy is a AAA password policy rule that

- rejects local user passwords and secrets containing a configured sequence length from alphabets, QWERTY keyboard characters, or numbers
- applies only to locally configured users and not to users on remote AAA servers, and

- can restrict cyclic wrapping sequences such as "yzab", "opqw", or "9012".

Feature history
Table 21: Feature History Table

Feature Name	Release Information	Feature Description
Password Policy to Restrict Consecutive Characters	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Password Policy to Restrict Consecutive Characters	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Password Policy to Restrict Consecutive Characters	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Password Policy to Restrict Consecutive Characters	Release 7.7.1	We have enhanced the router security by enforcing a strong password policy for all users configured on the router. You can now specify a new password policy for the user that restricts the usage of a specific number of consecutive characters for the login passwords. These characters include English alphabets, the sequence of QWERTY keyboard layout, and numbers, such as, 'abcd', 'qwer', '1234', and so on. Apart from *passwords*, the feature is also applicable for *secrets*–the one-way encrypted secure login passwords that are not easy to decrypt to retrieve the original unencrypted password text. The password policy is applicable only for the users configured on the local AAA server on the router; not those configured on the remote AAA server. The feature introduces the **restrict-consecutive-characters** command.

Valid and invalid consecutive-character examples

Most users tend to create passwords or secrets using consecutive characters, making them easier to remember yet susceptible to security attacks. From Cisco IOS XR Software Release 7.7.1 and later, you can enhance user password security by defining a policy that restricts sequences of consecutive characters from English alphabets, QWERTY keyboard layout, and numbers (examples: "abcd", "qwer", "zyxw", "1234"). Cyclic wrapping, such as "yzab" or "9012," can also be restricted. You can specify the number of consecutive characters to restrict.

Key aspects:

- The feature is disabled, by default.
- The security administrator must have **write** permission for AAA tasks to create password policies.
- Password policies are applicable only to locally configured users, not remote AAA server users.

This table depicts the examples of valid and invalid passwords and secrets when the password policy to restrict consecutive characters (say, 4 in this example) is in place.

Valid and invalid passwords

Table 22: Table

Use Case	Examples of Invalid Password or Secret	Examples of Valid Password or Secret
4 consecutive English alphabets	AbcD, ABCD, TestPQRS, DcbA, TestZYxW123, DCBA, ihgf	AbcPqR, Xyzdef, Yzab, zabC

Use Case	Examples of Invalid Password or Secret	Examples of Valid Password or Secret
4 consecutive English alphabets and decimal numbers from QWERTY keyboard layout	Qwer, QWER, Mnbv, aQwerm, Test1234, TestT7890, 5678, fghj	Opas, xzLk, sapo, saqw3210, Test9012
Restrict 4 consecutive English alphabets along with cyclic wrapping	Yzab, TestYZAB, zabc	1234, Qwer, QWER, Mnbv, aQwerm, Test1234, TestT0987
Restrict 4 consecutive English alphabets and numbers from QWERTY keyboard layout along with cyclic wrapping	9012, 8901, Test3210, TestT0987, Opqw, klas, dsal, Cxzm, nmzx	AbcD, ABCD, Yzab, TestYZAB, zabc

Configure consecutive-character restrictions for user passwords and secrets

Configure an AAA password policy that rejects user passwords and secrets containing consecutive English-alphabet, QWERTY-keyboard, or number sequences during local credential validation.

Enhance local password security by preventing passwords and secrets from using easily guessed consecutive character sequences.

Use the **restrict-consecutive-characters** command in AAA password policy configuration mode to restrict consecutive English-alphabet, QWERTY-keyboard, or number sequences. Optionally, include the **cyclic-wrap** keyword to prevent cyclic wrapping. After creating password policies, apply them explicitly to user profiles.

Before you begin

Create the password policy before you apply it to a user profile.

Procedure

1. Configure the consecutive-character restrictions in the AAA password policy.

   Example

   ```
   Router(config)#aaa password-policy test-policy
   Router(config-pp)#restrict-consecutive-characters english-alphabet 4
   Router(config-pp)#restrict-consecutive-characters qwerty-keyboard 5
   ```

   (Optional) Restrict cyclic wrapping of characters and numbers.

   Example

   ```
   Router(config-pp)#restrict-consecutive-characters english-alphabet 4
   cyclic-wrap
   Router(config-pp)#restrict-consecutive-characters qwerty-keyboard 5 cyclic-wrap
   ```

2. Apply the password policy to a user profile.

Example

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#commit
```

3. Review the consecutive-character restrictions in the running configuration.

Example

```
Router(config-pp)#show running-config aaa password-policy
Tue May 17 10:53:16.532 UTC

!
aaa password-policy test-policy
 restrict-consecutive-characters qwerty-keyboard 6
 restrict-consecutive-characters english-alphabet 4 cyclic-wrap
!
```

This is a sample running configuration that shows that you have configured a AAA password policy that restricts six consecutive characters from the QWERTY keyboard, and cyclic wrapping of four consecutive English alphabets.

4. Verify the consecutive-character restriction.

Example

```
Router#show aaa password-policy test-policy
Tue May 17 10:54:24.064 UTC
Password Policy Name : test-policy
  Number of Users : 0
  Minimum Length : 2
  Maximum Length : 253
  Special Character Len : 0
  Uppercase Character Len : 0
  Lowercase Character Len : 0
  Numeric Character Len : 0
  Policy Life Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 0
    years : 0
  Warning Interval :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 0
    years : 0
  Lockout Time :
    seconds : 0
    minutes : 0
    hours : 0
    days : 0
    months : 0
    years : 0
  Restrict Old Time :
    days : 0
    months : 0
```

```
    years : 0
  Character Change Len : 2
  Maximum Failure Attempts : 0
  Reference Count : 0
  Error Count : 0
  Lockout Count Attempts : 0
  Maximum char repetition : 0
  Restrict Old count : 0
  Restrict Username : 0
  Restrict Username Reverse  : 0
  Restrict Password Reverse  : 0
  Restrict Password Advanced : 0
  Restrict Consecutive Character :
    English Alphabet characters: 4
    English Alphabet Cyclic Wrap: True
    Qwerty Keyboard characters: 6
    Qwerty Keyboard Cyclic Wrap: False
Router#
```

Example

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#password DEFg
Router(config-un)#commit
Tue Dec  7 10:17:56.843 UTC

% Failed to commit and rollback one or more configuration items. Please issue
 'show configuration failed [inheritance]' from this session to view the
errors
Router(config-un)#show  configuration failed
username user1
 password 7 03205E0D01
!!% 'LOCALD' detected the 'fatal' condition 'Password contains consecutive
characters from qwerty keyboard or English alphabet'
!
End
```

Example

```
Router(config)#username user1
RP/0/RP0/CPU0:ios(config-un)#masked-secret
Fri Dec  3 12:33:44.354 UTC

Enter secret:
Re-enter secret:

secret is not compliant with policy to restrict consecutive letters or numbers
RP/0/RP0/CPU0:ios(config-un)#
```

Example

```
Router(config)#username user1
Router(config-un)#policy test-policy
Router(config-un)#secret qwerty
                             ^
% Invalid input detected at '^' marker.
Router(config-un)#
```

You can use the **show aaa password-policy** command to know if the feature to restrict consecutive characters for user passwords and secrets is applied on the password policy.

Password or Secret Configuration Failure Scenarios:

You notice these logs or error messages on the router console when password or secret configuration fails because of the policy violation to restrict consecutive characters or numbers:

5. (Optional) Configure the consecutive-character restriction with the YANG data model.

Example

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
 <target>
<candidate/>
</target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
 <aaa xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-lib-cfg">
<password-policies
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg">
<password-policy>
     <name>test-policy</name>
     <restrict-consecutive-characters>
      <qwerty-keyboard>
       <characters>4</characters>
 <cyclic-wrap></cyclic-wrap>
      </qwerty-keyboard>
      <english-alphabet>
       <characters>4</characters>
       <cyclic-wrap></cyclic-wrap>
      </english-alphabet>
</restrict-consecutive-characters>
</password-policy>
</password-policies>
</aaa>
</config>
</edit-config>
</rpc>
##
```

You can use the **Cisco-IOS-XR-aaa-locald-cfg** native YANG data model to restrict consecutive characters for user passwords and secrets. **Cisco-IOS-XR-um-aaa-locald-cfg** is the corresponding unified model (UM).

The following is a sample format to enable the feature using the native YANG data model.

To learn more about the data models and to put them to use, see the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

The password policy rejects any password or secret values that contain the configured consecutive-character sequence, helping protect user credentials from predictable patterns.

Task-based authorization

Describes task-based authorization, detailing supported task-ID behaviors, methods for assigning permissions to external AAA users, privilege level mapping, task map structure, and configuration using XML schema and protocols such as NETCONF and RESTCONF.

A task-based authorization model is an access control model that

- authorizes CLI and API operations based on assigned task permissions
- controls configuration, monitoring, and operational access through AAA task permissions, and
- replaces the privilege-level concept used previously in Cisco IOS with a task-based approach in Cisco IOS XR software.

Additional reference information

AAA employs task permissions for any control, configuration, or monitoring operation through CLI or API. The Cisco IOS software concept of privilege levels has been replaced by a task-based authorization system.

Task IDs for authorization

Explains how task IDs represent operational tasks and connect users, user groups, task groups, and commands to authorized router access.

A task ID is a permission identifier that

- defines the permission to run an operation for a Cisco IOS XR command or API request
- represents control, configuration, and monitoring operations, and
- allows users associated with the task ID to execute operations that require that task ID.

Additional reference information

Task IDs are assigned to users through membership in user groups and task groups. Each user is associated with one or more user groups. Every user group is associated with one or more task groups, and each task group is defined by a set of task IDs. Thus, a user's membership in a user group links that user to a specific set of task IDs, which determine the scope of their authorized access to the router.

The operational tasks that enable users to control, configure, and monitor Cisco IOS XR software are represented by task IDs. A task ID defines the permission for a command. Users associated with the appropriate task IDs can execute any operation permitted by those IDs.

Requirement: Use supported task-ID behavior

Outlines the task-ID usage rules for Cisco IOS XR CLI commands, XML API invocations, operation classes, ROM monitor commands, and configuration submodes.

You must follow these requirements when using task IDs for CLI commands, XML API invocations, and other operations in Cisco IOS XR:

- Most router control, configuration, or monitoring operations using the CLI or XML API are associated with a specific set of task IDs. Each command or API invocation typically is associated with at least one task ID.
- The **config** and **commit** commands, as well as command aliases, do not require any specific task ID permissions. To perform a configuration replace operation, you must have root-lr permissions. To restrict access to configuration mode, use TACACS+ command authorization to deny the **config** command.
- Task ID associations are hard-coded within the router and cannot be modified. Task IDs grant permission to perform certain tasks but do not deny permission. Permissions can be read, write, execute, or debug.
- Task ID classes include:
  - Read: Permit only read operations.
  - Write: Permit change operations (and implicitly allow read).
  - Execute: Permit access operations (e.g., ping, Telnet).

- Debug: Permit debug operations.

- Multiple operations separated by a slash (e.g., read/write) or comma (e.g., read/write, execute) apply to the specified task IDs accordingly.
- If no task ID or operation is specified for a command, then you may use it without a prior association.
- ROM monitor commands do not require task ID association.
- In specific configuration submodes, you may need additional task ID permissions to use some commands. For example, to execute **show redundancy**, you must be associated with the system (read) task ID.
- If you experience issues using a CLI command, contact your system administrator.

Methods for assigning task permissions to external AAA users

Lists the methods that Cisco IOS XR AAA supports for assigning task permissions to users authenticated through TACACS+ or RADIUS external servers.

Cisco IOS XR AAA supports several methods for assigning task permissions to users authenticated through TACACS+ or RADIUS external servers:

- Text task map: Specify the text version of the task map directly in the configuration file of the external TACACS+ or RADIUS server.
- Privilege level: Set the privilege level within the external server's configuration file for users.
- Matching local username: Create a local user with the same username as the externally authenticated user; this user inherits the local task permissions.
- Default task group: Configure a default task group whose permissions are applied to any user authenticating with TACACS+ or RADIUS.

Use the assignment method that matches the external AAA server's task-ID or privilege-level support.

Task maps for external AAA users

Explains the remote task-map method that defines Cisco IOS XR task IDs for users authenticated by TACACS+ or RADIUS servers outside the local database.

A task map is a remote task-permission definition that

- lets external AAA servers define router task IDs remotely
- can include specific task IDs and user group references, and
- is represented by task strings in external server configuration.

Additional reference information

For users who are authenticated using an external TACACS+ server and RADIUS server, Cisco IOS XR software AAA supports a method to define task IDs remotely.

External AAA task string syntax and token rules

Provides the task string syntax and task-ID token rules used in external TACACS+ or RADIUS server configurations for remote task permissions.

The task string used for configuring remote task permissions in external TACACS+ or RADIUS servers consists of comma-delimited tokens. Each token contains either a task ID and its permissions or a user group to include for a specific user.

Task String Structure

The syntax format is as follows:

task = "permissions:taskid_name, #usergroup_name, ..."

Permissions are represented by letters:

- r: read
- w: write
- x: execute
- d: debug

The pound sign (#) indicates that a user group follows.

```
user = user1{
member = some-tac-server-group
opap = cleartext "lab"
service = exec {
task = "rwx:bgp,#operator"
}
}
```

In this example, user1 receives BGP read, write, and execute permissions, and belongs to the user group "operator".

Usage notes

- Task strings should be defined without blank spaces or quotation marks within the custom attribute field.
- Cisco IOS XR software allows you to specify task IDs as attributes in the external server configuration.
- If the server is used with systems other than Cisco IOS XR, attributes may be marked as optional, as described in server documentation (for instance, CiscoSecure ACS and Cisco freeware TACACS+ server require an asterisk * instead of an equal sign = for optional attributes).

Optional attributes

- To enable interoperability with Cisco IOS-based systems, add the optional keyword in front of "task".
- When configuring optional attributes, consult your TACACS+ server documentation to determine the required syntax.

Task string example for CiscoSecure ACS

To specify task IDs and user groups in CiscoSecure ACS:

task = "permissions:taskid name, #usergroup name, ..."

task=rwx:bgp,#netadmin

Configure task permissions for external AAA users in CiscoSecure ACS

Configure custom task permissions in CiscoSecure ACS so that an externally authenticated user receives Cisco IOS XR task access after login.

Assign Cisco IOS XR task IDs or user groups to an external AAA user via a shell custom attribute, enabling explicit control of user task permissions at login.

Task strings in the TACACS+ server configuration file consist of comma-delimited tokens representing task IDs with associated permissions and user group membership. This mechanism allows precise control over what commands and features the user can access on Cisco IOS XR devices after authenticating through CiscoSecure ACS.

Before you begin

- Create or select the external AAA user group to be assigned task permissions.

- Ensure you have access to CiscoSecure ACS and the relevant user accounts.
- Review TACACS+ server documentation if you're assigning optional attributes.

Procedure

1. Log in to CiscoSecure ACS using your administrator credentials.
2. Navigate to Group Setup.
3. From the Group drop-down list, select the user group you want to update.
4. Click Edit Settings to modify the group configuration.
5. Locate and select the Shell (exec) check box to enable custom attribute configuration.
6. Check the Custom attributes option.
7. In the custom attributes field, enter the task string to assign task IDs and user groups. For example:

   Example

   ```
   task=rwx:bgp,#netadmin
   ```

8. Click Submit + Restart to apply changes and restart the server.

Assigning multiple tasks and groups in RADIUS Vendor-Specific Attributes:

- shell:tasks=#sysadmin,rwx:bgp,r:ospf

After login, the user may verify their assigned tasks using the **show user tasks** command.

How privilege level mapping works

Describes how Cisco IOS XR maps TACACS+ privilege levels from external server configuration to local user groups and task maps.

Privilege level mapping provides compatibility with TACACS+ daemons that do not support task IDs.

Summary

The key components involved in the process are:

- External TACACS+ server: Authenticates users and returns a privilege level based on the user's configuration.
- AAA system: Receives the returned privilege level and maps it to a corresponding local user group.
- Local user groups (priv1–priv13, owner-sdr): Supply task maps that define the permissions assigned to authenticated users.

Privilege level mapping provides compatibility with TACACS+ daemons that do not support task IDs. AAA maps the returned external privilege level to a local user group and assigns that group's task map to the authenticated user.

Workflow

These stages describe how privilege level mapping works.

1. The external TACACS+ server authenticates the user and returns a privilege level.
2. AAA maps privilege levels 1 through 13 to local user groups named priv1 through priv13. Privilege level 14 maps to the owner-sdr user group.
3. AAA assigns the task map of the mapped local user group to the authenticated user.

Suppose the TACACS+ server returns privilege level 5 for a user. AAA maps this to the local user group priv5 and assigns the corresponding task map:

```
user = sampleuser1{
    member = bar
    service = exec-ext {
        priv_lvl = 5
    }
}
```

Categories of AAA XML schema for configuration and monitoring

Lists the AAA XML schema categories that Cisco IOS XR publishes for XML-based configuration and monitoring tools used by management applications.

Cisco IOS XR publishes various XML schema categories to support XML-based configuration and monitoring of AAA (Authentication, Authorization, and Accounting) services through management tools and applications. These schema define the structure of XML requests and responses used to communicate with the XML agent for AAA.

AAA XML schema categories:

- Authentication, Authorization and Accounting configuration
- User, user group, and task group configuration
- TACACS+ server and server group configuration
- RADIUS server and server group configuration

Use these schema categories to guide XML-based communication between configuration tools or management applications and the XML agent when performing AAA-related operations.

Support for username and password control in NETCONF and RESTCONF AAA services

Provides details about protocol support for username and password control using local or AAA services in management workflows, including RESTCONF support limitations.

NETCONF and RESTCONF manage username and password access in management workflows by using either local or AAA (Authentication, Authorization, Accounting) services. This approach parallels the control methods used in XML schema-based access.

Key reference information

- Both protocols utilize local or AAA services to manage credentials.
- RESTCONF support for AAA control will be added in a future release.
- Use this reference to understand management protocol scope and RESTCONF support limitations.

Summary of protocol support

- NETCONF currently supports username and password control using local or AAA services.
- RESTCONF support for AAA-based credential control is planned for future releases; it presently relies on local authentication.

AAA method lists

Details configuration and application of AAA method lists, including authentication, authorization, and accounting methods, their behaviors and creation steps, application to access lines, interim accounting, response timeouts, and enabling relevant AAA services.

An AAA method list is a source-selection policy that

- specifies which AAA data sources to use for authentication, authorization, or accounting requests
- allows applications to select either a named or default method list, and
- defaults to the local database if no method list exists or is specified.

Additional reference information

AAA data may reside in several data sources. Method lists are used in AAA configuration to define the preference order for these sources. Applications such as "login" can select a specific method list—console ports may use one method list, while vty ports may use another. If a method list is not specified for an application, it attempts to use the default method list. If no default method list exists, AAA uses the local database by default.

Console ports may use a dedicated named method list, while vty ports use another, providing granular control over AAA behaviors.

Authentication method-list options and application

Lists key facts about authentication method lists, including default behavior, explicit application requirements, and use of authentication for router access.

Authentication method lists specify the order of AAA data sources used to verify users or principals. They determine which authentication sources—such as local databases, server groups, or line passwords—are considered for access control.

Usage and application

- You use authentication method lists to align console, vty, PPP, or other login applications with your intended authentication source order.
- If a method list is not specified, the application uses a default method list.
- Applications should explicitly reference defined method lists for predictable authentication behavior.

Line authentication

- Authentication can be applied to tty lines using the **login authentication** <**method-list-name**> command in line configuration mode.
- When using RADIUS or TACACS+ servers directly (not groups), the server is chosen from the global pool of configured servers, following the order of configuration. Named server groups allow finer selection.
- Subsequent methods from a list are used only if the initial method encounters an error (not rejection).

Common authentication methods

- group tacacs+: Use a TACACS+ server group or TACACS+ servers.
- group radius: Use a RADIUS server group or RADIUS servers.
- group named-group: Use a defined subset of TACACS+ or RADIUS servers.

- local: Use the local username or password database.
- line: Use line password or user group for authentication.

Table 23: Authentication method-list command details

Task instruction	Command syntax	Source details
Enter global configuration mode.	No command syntax in this source step.	Enters global configuration mode.

Task instruction	Command syntax	Source details

Create the authentication method list.	`aaa authentication {login} {default | list-name} method-list`	Creates a series of authentication methods, or a method list.
		• Using the **login** keyword sets authentication for login. Using the **ppp** keyword sets authentication for Point-to-Point Protocol.
		• Entering the **default** keyword causes the listed authentication methods that follow this keyword to be the default list of methods for authentication.
		• Entering a *list-name* character string identifies the authentication method list.
		• Entering a
		method-list
		argument following the method list type. Method list types are entered in the preferred sequence. The listed method types are any one of the following options:
		  • group tacacs+ - Use a server group or TACACS+ servers for authentication
		  • group radius - Use a server group or RADIUS servers for authentication
		  • group *named-group* - Use a named subset of TACACS+ or RADIUS servers for authentication
		  • local - Use a local username or password database for authentication
		  • line - Use line password or user group for authentication
		• The example specifies the **default** method list to be used for authentication.

Save or discard the configuration changes.	No command syntax in this source step.	**commit** - Saves the configuration changes and remains within the configuration session. end • **Yes** - Saves configuration changes and exits the configuration session. • **No** - Exits the configuration session without committing the configuration changes. • **Cancel** - Remains in the configuration session, without committing the configuration changes.

Key points

- Explicit reference by applications to method lists ensures intended authentication flow.

- Default lists apply when no explicit list is referenced.
- Only methods that return errors (not rejections) trigger use of alternative methods in the list.

Configure an authentication method-list

Configure an authentication method-list for login or PPP.

Define the order of authentication methods that Cisco IOS XR uses for the selected authentication application.

Use an authentication method list to specify the sequence of methods (such as TACACS+, RADIUS, or local) that Cisco IOS XR attempts during login or PPP authentication. By configuring distinct method lists, you can customize authentication on a per-interface or per-application basis.

Before you begin

- The default method list is applied for all interfaces unless a non-default named method list is explicitly configured; in that case, the named method list is applied.
- Plan the method order, including any TACACS+, RADIUS, local, or fallback methods, before you configure the list.
- To refer to sets of RADIUS or TACACS+ servers:
  - Use the **radius server-host** or **tacacs-server host** command to configure the host servers.
- Use the **aaa group server radius** or **aaa group server tacacs+** command to create named groups of servers.
- The **group radius**, **group tacacs+**, and **group group-name** forms of the `aaa authentication` command reference these predefined server groups.

Plan the method order, including any TACACS+, RADIUS, local, or other fallback methods, before you configure the list.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Create the authentication method list.

   Example

   ```
   Router(config)# aaa authentication login default group tacacs+
   ```

3. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - commit: Saves the configuration changes and remains within the configuration session.
   - end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.

- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

4. Repeat Step 1 through Step 3 for every authentication method list to be configured.

The authentication method list is available for applications that explicitly reference it or for default-list use.

Authorization method-list features and operation

Provides a reference to authorization method-list features, supported authorization types, fallback behavior, lockout prevention guidance, and command authorization scope in AAA deployments.

Authorization method lists define the sequence and method by which Cisco IOS XR attempts to authorize users for various services. Method lists support fallback logic and enable administrators to specify and control authorization protocols.

Supported authorization types

- Command authorization: Controls XR EXEC mode commands (distinct from task-based authorization).

  Command authorization applies only after configuration and association with a line template; task-based authorization operates automatically.
- XR EXEC mode authorization: Authorizes the start of XR EXEC mode sessions.
- Network authorization: Authorizes network services, such as IKE.
- Event manager authorization: Authorizes event manager operations (supports TACACS+ or local; RADIUS is not allowed for event manager authorization).

Fallback behavior

- The software attempts authorization with the next listed method only if the previous method returns no response or an error (not a failure).
- If any method returns a failure (denial of service), the process stops; no other methods are attempted.

Lockout prevention guidance

- To avoid user lockouts, configure local fallback as an option for authorization, for example, **aaa authorization** commands default tacacs+ local
- Always ensure that server groups are appropriately configured when using TACACS+ or RADIUS (use the **aaa group server tacacs+** command).

Command authorization details

Table 24: Authorization method-list command details

Task instruction	Command syntax	Source details
Enter global configuration mode.	No command syntax in this source step.	Enters global configuration mode.

Task instruction	Command syntax	Source details
Create the authorization method list.	`aaa authorization {commands | eventmanager | exec | network} {default | list-name} {none | local | group {tacacs+ | radius | group-name}}`	Creates a series of authorization methods, or a method list. • The **commands** keyword configures authorization for all XR EXEC mode shell commands. Command authorization applies to the EXEC mode commands issued by a user. Command authorization attempts authorization for all XR EXEC mode commands. • The **eventmanager** keyword applies an authorization method for authorizing an event manager (fault manager). • The **exec** keyword configures authorization for an interactive (XR EXEC mode) session. • The **network** keyword configures authorization for network services like PPP or IKE. • The **default** keyword causes the listed authorization methods that follow this keyword to be the default list of methods for authorization. • A list-name character string identifies the authorization method list. The method list itself follows the method list name. Method list types are entered in the preferred sequence. The listed method list types can be any one of the following: • **none** - The network access server (NAS) does not request authorization information. Authorization always succeeds. No subsequent authorization methods will be attempted. However, the task ID authorization is always required and cannot be disabled. • local - Uses local database for authorization. • **group tacacs+** - Uses the list of all configured TACACS+ servers for authorization. The NAS exchanges authorization information with the TACACS+ security daemon. TACACS+ authorization defines specific rights for users by associating AV pairs, which are stored in a database on the TACACS+ security server, with the appropriate user. • **group radius** - Uses the list of all configured RADIUS servers for authorization. • **group** *group-name* - Uses a named server group, a subset of TACACS+ or RADIUS servers for authorization as defined by the **aaa group server tacacs+** or **aaa group server radius** command.

Task instruction	Command syntax	Source details
Save or discard the configuration changes.	No command syntax in this source step.	**commit** - Saves the configuration changes and remains within the configuration session. end • **Yes** - Saves configuration changes and exits the configuration session. • **No** - Exits the configuration session without committing the configuration changes. • **Cancel** - Remains in the configuration session, without committing the configuration changes.

Additional notes

- Method lists must be applied to specific lines or interfaces to be active.
- Do not use method names (such as "TACACS+") as method-list names.
- The **aaa authorization** command sends attribute-value pairs to the TACACS+ daemon for approval or refusal.

Create an authorization method-list

Configure an authorization method-list to specify the desired order of authorization methods for router command, event manager, EXEC, or network authorization.

Define the order of authorization methods that Cisco IOS XR uses for different types of authorization (commands, event manager, EXEC, or network).

Authorization method lists help you control how and in which order authorization is applied for different router actions. Use this task to configure and commit the ordered methods.

Before you begin

- Review the available authorization methods and plan the desired order for your list.
- Determine if a local fallback method is needed.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Configure the authorization method list for the desired type (commands, event manager, EXEC, or network) and specify the order and group.

   Example

   ```
   Router(config)# aaa authorization commands listname1 group tacacs+
   ```

3. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The authorization method list becomes available for lines or applications that explicitly reference it.

Supported accounting methods and behavior for AAA method-lists

Provides accounting method-list behavior, supported accounting methods, accounting record types, storage behavior, and server-side accounting log expectations for AAA services.

Accounting method lists define how Cisco IOS XR reports user activity to TACACS+ or RADIUS security servers. They enable selection of accounting types (stop-only, start-stop, or update records) for specific lines and interfaces, and control what accounting logs are generated.

Supported accounting methods

- TACACS+
- RADIUS

The router sends accounting records to the configured TACACS+ or RADIUS security server. Each record contains accounting AV pairs and is stored on the security server.

Accounting record types

- stop-only: Sends a "stop accounting" notice at the end of the user process.
- start-stop: Sends a "start accounting" notice at the beginning and a "stop accounting" notice at the end of the process.
- update: Periodically sends update records with accumulated information.
- none: No accounting is performed.

Storage behavior

Accounting records are stored only on the TACACS+ or RADIUS server. When AAA accounting is activated, the router reports account attributes, which are stored in the accounting log on the security server.

Method-list command syntax

Table 25: Accounting method-list command details

Task instruction	Command syntax	Source details
Enter global configuration mode.	No command syntax in this source step.	Enters global configuration mode.

Task instruction	Command syntax	Source details

Choose the accounting method-list command.	`aaa accounting {commands | exec | network} {default | list-name} {start-stop | stop-only}`
	
	`{none | method}`	

Task instruction	Command syntax	Source details

		Note
		
		Command accounting is not supported on RADIUS, but supported on TACACS.
		
		Creates a series of accounting methods, or a method list.
		• The **commands** keyword enables accounting for XR EXEC mode shell commands.
		• The **exec** keyword enables accounting for an interactive (XR EXEC mode) session.
		• The **network** keyword enables accounting for all network-related service requests, such as Point-to-Point Protocol (PPP).
		• The **default** keyword causes the listed accounting methods that follow this keyword to be the default list of methods for accounting.
		• A *list-name* character string identifies the accounting method list.
		• The **start-stop** keyword sends a "start accounting" notice at the beginning of a process and a "stop accounting" notice at the end of a process. The requested user process begins regardless of whether the "start accounting" notice was received by the accounting server.
		• The **stop-only** keyword sends a "stop accounting" notice at the end of the requested user process.
		• The **none** keyword states that no accounting is performed.
		• The method list itself follows the
		start-stop
		keyword. Method list types are entered in the preferred sequence. The method argument lists the following types:
		  • **group tacacs+** - Use the list of all configured TACACS+ servers for accounting.
		  • **group radius** - Use the list of all configured RADIUS servers for accounting.
		  • **group** *group-name* - Use a named server group, a subset of TACACS+ or RADIUS servers for accounting as defined by the **aaa group server tacacs+** or **aaa group server radius** command.

Task instruction	Command syntax	Source details
		• The example defines a **default** command accounting method list, in which accounting services are provided by a TACACS+ security server, with a stop-only restriction.
Save or discard the configuration changes.	No command syntax in this source step.	**commit** - Saves the configuration changes and remains within the configuration session. end • **Yes** - Saves configuration changes and exits the configuration session. • **No** - Exits the configuration session without committing the configuration changes. • **Cancel** - Remains in the configuration session, without committing the configuration changes.

Server-side accounting log expectations

- AAA accounting logs are stored on the security server.
- Method lists control what logs are generated for each service (commands, sessions, network requests).
- Naming a method list should avoid the names of accounting methods (e.g., TACACS+) to prevent confusion.

Create an accounting method-list

Configure an accounting method list for commands, EXEC, or network accounting, select the record mode, and commit the ordered accounting methods.

Define an accounting method list to control how accounting records are generated and which methods are used for a specific accounting type.

Use accounting method lists to specify accounting types (commands, EXEC, or network), select the record mode, and order accounting methods such as TACACS+ or RADIUS.

Before you begin

- Decide which accounting type (commands, EXEC, or network) you need.
- Select the record mode (start-stop or stop-only).
- Plan which server method to use (TACACS+ or RADIUS).

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Specify the accounting method-list.

Example

```
Router(config)# aaa accounting commands default stop-only group tacacs+
```

> **Note**
>
> Command accounting is supported on TACACS+, but not on RADIUS.

3. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The accounting method list is now available for lines or interfaces where it is applied.

Configure interim accounting records

Configure interim accounting records to send accounting updates when new information appears or at specified intervals for active accounting sessions.

Send interim accounting updates to the accounting server before the user process ends, enabling more frequent reporting of accounting information.

This task enables periodic interim accounting records to be sent to the accounting server. When the **aaa accounting update** command is activated, software issues interim accounting records for all users on the system.

> **Note**
>
> Interim accounting records are generated only for network sessions, such as Internet Key Exchange (IKE) accounting, which is controlled by the aaa accounting command with the network keyword. System, command, or EXEC accounting sessions cannot have interim records generated.

Before you begin

Review the congestion caution before you use periodic accounting updates in a deployment with many logged-in users.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Enable interim accounting records.

   Example

   ```
   Router(config)# aaa accounting update periodic 30
   ```

   Enables periodic interim accounting records to be sent to the accounting server.

   - If you use the **newinfo** keyword, interim accounting records are sent to the accounting server whenever new accounting information appears, such as IP address negotiation completion.
   - If you use the **periodic** keyword, updates are sent at specified intervals; the record contains all accounting information recorded for the user up to that time.

   > **Note**
   >
   > Using the **periodic** keyword can cause heavy network congestion if many users are logged in.

3. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - commit: Saves the configuration changes and remains within the configuration session.
   - end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

The accounting server receives interim accounting records according to the selected update mode (new information or periodic interval), improving tracking for active accounting sessions.

Application of AAA method-lists to console and vty access

Provides details about how configured authorization and accounting method lists affect applications such as console and vty access after creation.

After you configure method lists for authorization and accounting services, you can apply those lists to applications (for example, console and vty access) by enabling AAA authorization and accounting on the relevant lines.

Key facts

- AAA authorization and accounting method lists take effect for applications (such as console and vty) only after AAA authorization or accounting is enabled for the associated lines or services.
- Applying a method list to a specific line or group of lines requires you to enable AAA authorization for those lines.
- Use this information to determine where and when to apply authorization and accounting lists after creating them.

Configure AAA authorization on lines

Configure AAA authorization for commands or EXEC sessions on a selected line template or group of lines that uses method-lists.

Apply an authorization method list to a line so that Cisco IOS XR performs authorization for commands or interactive EXEC sessions.

After you use the **aaa authorization** command to define a named authorization method list (or use the default method list) for a particular type of authorization, you must apply the defined lists to the appropriate lines in order for authorization to take place. Use the **authorization** command to apply the specified method lists (or, if none is specified, the default method list) to the selected line or group of lines.

Before you begin

Create the authorization method list before you apply it to a line template.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Enter line template configuration mode.

   Example

   ```
   Router(config)# line console
   ```

   Enters line template configuration mode.

3. Enable AAA authorization on the line template.

   Example

   ```
   Router(config-line)# authorization commands listname5
   ```

Enables AAA authorization for a specific line or group of lines.

- The **commands** keyword enables authorization on the selected lines for all commands.
- The **exec** keyword enables authorization for an interactive (XR EXEC mode) session.
- Enter the **default** keyword to apply the name of the default method list, as defined with the **aaa authorization** command.
- Enter the name of a list of authorization methods to use. If no list name is specified, the system uses the default. The list is created with the **aaa authorization** command.
- The example enables command authorization using the method list named listname5.

4. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The selected line or line template uses the configured authorization method-list.

Enable accounting services

Enable AAA accounting for commands or EXEC sessions on a selected line template or group of lines using an accounting method-list.

Enable AAA accounting to ensure Cisco IOS XR keeps a record of command activity or session events.

Use this task to enable accounting services for a particular line or group of lines, helping you track user actions and activity.

Before you begin

Create the accounting method list before you apply it to a line template.

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Enter line template configuration mode.

Example

```
Router(config)# line console
```

Enters line template configuration mode.

3. Enable AAA accounting on the line template.

Example

```
Router(config-line)# accounting commands listname7
```

Enables AAA accounting for a specific line or group of lines.

- The **commands** keyword enables accounting on the selected lines for all XR EXEC mode shell commands.
- The **exec** keyword enables accounting for an interactive (XR EXEC mode) session.
- Enter the **default** keyword to apply the name of the default method list, as defined with the **aaa accounting** command.
- Enter the name of a list of accounting methods to use. If no list name is specified, the system uses the default. The list is created with the **aaa accounting** command.
- The example enables command accounting using the method list named listname7.

4. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The selected line or line template now uses the configured accounting method-list, and accounting services are enabled for commands or EXEC sessions.

Configure the login response timeout for a line template

Configure the interval that a line waits for a login response before the login attempt times out for a line template.

Control how long the server waits for a reply to a login prompt.

Use this task to set the timeout interval for login attempts on a line template. Adjusting this value determines how long the server waits for a response before timing out unsuccessful login attempts.

Before you begin

Choose a timeout value from 0 to 300 seconds. The default is 30 seconds.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Enter line template configuration mode.

   Example

   ```
   Router(config)# line template alpha
   ```

   Specifies a line to configure and enters line template configuration mode.

3. Set the login response timeout.

   Syntax: timeout login response seconds

   Example

   ```
   Router(config-line)# timeout login response 20
   ```

   Sets the interval that the server waits for reply to a login.

   - The *seconds* argument specifies the timeout interval (in seconds) from 0 to 300. The default is 30 seconds.
   - The example shows how to change the interval timer to 20 seconds.

4. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - commit: Saves the configuration changes and remains within the configuration session.
   - end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

The line template uses the configured login response timeout.

Command accounting methods

Outlines command accounting methods, providing configuration procedures for recording and auditing user command activities within network devices.

A command accounting method is a logging method that

- records commands executed by users as syslog messages
- allows only users with AAA write permissions to enable or disable accounting, and
- does not support commands executed via NETCONF, XML, or gRPC.

Additional reference information

Command accounting enables logging of all commands executed by any user as syslog messages, provided the accounting method is set to local. Only users with AAA write permissions can manage the feature. Commands recorded via command accounting can be viewed using the **show logging** command. Command accounting functions as an additional accounting method and remains active alongside other configured accounting methods; it is not a failover method.

Configure command accounting

Configure command accounting locally or alongside other accounting methods so that Cisco IOS XR logs user commands as syslog messages.

Enable command accounting to log the commands that users execute on the router.

Command accounting can be configured alone or in combination with other accounting methods to ensure commands are logged as syslog messages.

Before you begin

You must have AAA write permissions to enable or disable command accounting.

Procedure

Determine whether you want to configure command accounting alone or alongside other accounting methods.

To configure command accounting alone, enter the following command in configuration mode.

Example

```
Router(config)# aaa accounting commands default start-stop local none
Router(config)# commit
```

Example

```
Router(config)# aaa accounting commands default start-stop group tacacs+ local
 none
Router(config)# commit
```

To configure command accounting together with another accounting method (such as TACACS+):

Example

```
Router(config)# aaa accounting commands default start-stop group tacacs+ local
 none
Router(config)# commit
```

Command accounting is enabled, and user command execution can now be reviewed from the syslog output.

Local command authorizations with user accounts

Explains local command authorization, illustrating its operation, configuration for user accounts, and integration with TACACS+ including behaviors for local fallback scenarios.

A local command authorization feature with user accounts is a local authorization mechanism that

- permits command execution for locally authenticated users when TACACS+ authorization is unavailable or not required
- prevents local users from being locked out when a TACACS+ server is unreachable, and
- ensures that remote users cannot bypass TACACS+ authorization by using the local fallback method; the local option can be used as a fallback after TACACS+ or as the only command authorization method.

Feature history
Table 26: Feature History Table

Feature Name	Release Information	Feature Description
Command Authorization Using Local User Account	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Command Authorization Using Local User Account	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Command Authorization Using Local User Account	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Command Authorization Using Local User Account	Release 7.5.1	This feature allows locally authenticated users—authenticated by the AAA server internal to the router—to run all XR VM commands even if a remote TACACS+ AAA server is not reachable for authorization. It prevents a complete router lockdown. The feature also prevents remotely authenticated users—authenticated using a remote AAA server (say, TACACS+ server)—from running any non-permitted commands on the router, and thus prevents misuse of user privileges. This feature modifies the aaa authorization commands default command to include the **local** option for XR VM command authorization.

Local command authorization behavior

Currently, when a user tries to execute a command on XR VM, the router checks to see whether the user has required permissions to execute it. The router does this authorization process in two steps. First, the system compares the task-IDs of the user with the required task-IDs for the command. If the user has all required task-IDs, and if AAA authorization is configured, then the system sends an authorization request to the local or remote AAA server, based on that configuration. Based on the response from the AAA server, the system allows or rejects the command execution. If authorization is not configured or if it configured with option none, then the system bypasses authorization check and allows user to execute the command.

Similarly, the existing remote authorization process using TACACS+ server has two options - remote authorization using tacacs+ and none. The authorization process using TACACS+ option uses an external TACACS+ server for authorization. The authorization using none option allows the user to execute the command without any authorization check. TACACS+ authorization has the advantage of fine-tuning authorization rules and providing more control on system access that cannot be otherwise done locally. However, if the remote server is not reachable, a user who leverages TACACS+ authorization might get into an unpredictable state of router, as mentioned in these scenarios:

- Remote authorization using TACACS+ with failover option as none (that is, with the **aaa authorization commands default group tacacs+ none** configuration)

  If TACACS+ server is not reachable, then the system bypasses the authorization check and allows user to execute the command. A user who does not have permission to execute certain commands due to additional authorization rules on the TACACS+ server, then gets permission to execute those commands in this scenario. This action introduces a privilege escalation.

- Remote authorization using TACACS+ without any failover option (that is, with the **aaa authorization commands default group tacacs+** configuration)

  If TACACS+ server is not reachable, then the system does not authorize the command at all. Because the user then cannot execute any command, the router gets locked out.

With the introduction of command authorization using local user account feature in Cisco IOS XR Software Release 7.5.1, locally authenticated users can execute commands even if a TACACS+ server is not reachable. This behavior is similar to the behavior with the failover option none, with the only difference that only locally authenticated users can execute commands in this case. This functionality thereby prevents a complete lockdown of the router as mentioned in one of the previously existing scenarios mentioned earlier. At the same time, the feature also prevents users who are authenticated remotely (that is, TACACS+ authenticated users) from executing any non-permitted command on the router. This behavior in turn helps to prevent any sort of misuse of user privileges on the router.

How command authorization works

Describes the command-authorization flow for remotely authenticated and locally authenticated users when TACACS+ authorization is configured during command execution on the router.

Command authorization uses user task IDs and the configured AAA authorization method to decide whether a user can execute a command.

Summary

The key components involved in the process are:

- User: Initiates commands that require authorization.
- AAA Authorization method: Determines whether the user is authorized to execute a command based on the configured method.
- TACACS+ server: Performs external authorization checks for remotely authenticated users.

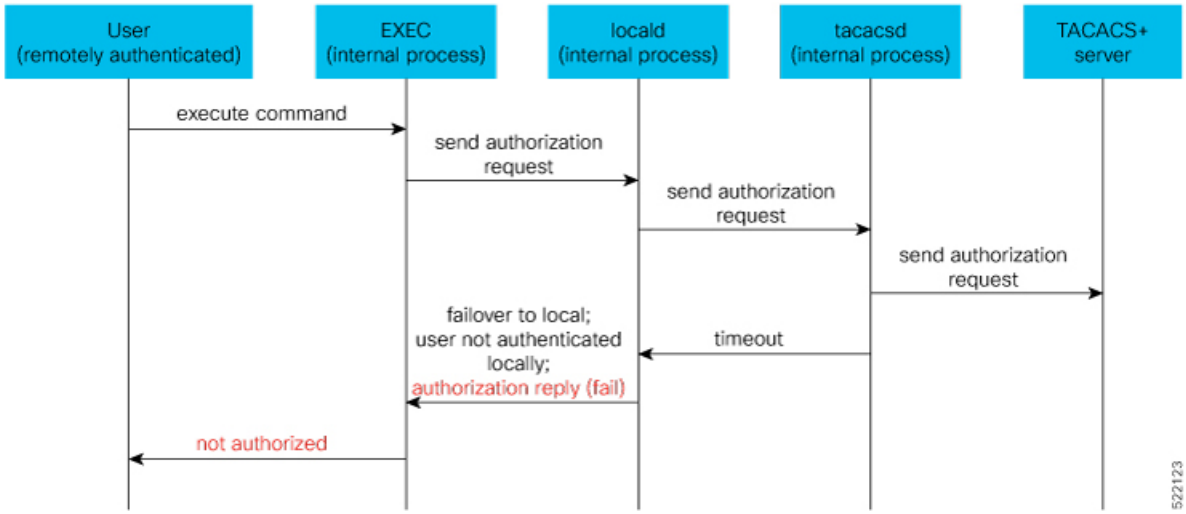
Command authorization ensures that users are permitted to execute commands on a router according to authentication and authorization settings.

Workflow

These stages describe how command authorization works.

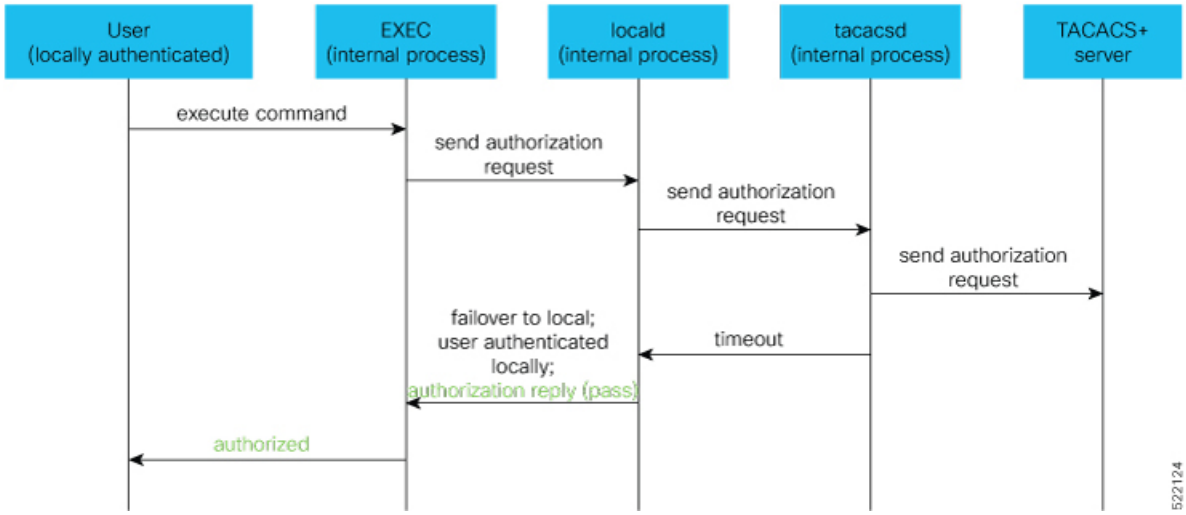
1. Remotely authenticated user:
   - If the TACACS+ server is available, the server authorizes command execution.
   - If there is a timeout and the TACACS+ server is unreachable, command authorization fails. The user cannot execute any commands until the TACACS+ server becomes reachable again, preventing misuse of user privileges.

Figure 4: Call Flow of Command Authorization for Remotely Authenticated Users



2. Locally authenticated user:
   - Even if the authorization request to the TACACS+ server times out, local command authorization succeeds.
   - The local AAA component in the router does not perform an additional check. The user can execute the command regardless of TACACS+ server availability, preventing a complete lockdown of the router.

Figure 5: Call Flow of Command Authorization for Locally Authenticated Users



Result

The router either authorizes the command, denies the command, or blocks command execution until the required TACACS+ server is reachable, depending on the user's authentication method.

Configure command authorization with a local user account

Configure command authorization so locally authenticated users can execute commands when the TACACS+ server is unreachable or local-only command authorization is required.

Preserve command execution capabilities for locally authenticated users during TACACS+ server unreachability or when local-only command authorization is necessary.

Use this task to enable local command authorization or TACACS+ command authorization with local failover, ensuring command execution remains possible when remote authorization fails.

Before you begin

- Be cautious to prevent potential lockout situations caused by misconfiguration.
- If local is the only method specified for authorization, remotely authenticated users may lose access to execute further commands after configuring command authorization using the local user account feature.

Procedure

1. Configure command authorization with local failover or local authorization.

   Example

   ```
   Router#configure
   Router(config)#aaa authorization commands default group tacacs+ local
   ```

   Example

   ```
   Router(config)#aaa authorization commands default local
   ```

   To configure command authorization using local user account, use the **local** option in the **aaa authorization** command in any of these formats:

   Or

2. Review the command-authorization running configuration.

   Example

   ```
   Router#show run aaa
   !
   aaa authorization commands default group tacacs+ local
   !
   ```

   Example

   ```
   Router#show run aaa
   !
   aaa authorization commands default local
   !
   ```

3. Verify the user authentication method.

   Example

   ```
   Router#show user authentication method
   local
   ```

The router authorizes commands by using TACACS+ with local failover, or by using local authorization only, ensuring command execution for locally authenticated users in relevant scenarios.

Command authorization behaviors with TACACS+ and local fallback

Provides feature behaviors and use-case scenarios for TACACS+ command authorization with local command authorization fallback during local and remote login.

Feature behavior with various local command authorization options

This table lists command authorization feature behaviors depending on the AAA authorization configuration.

Table 27: Feature behavior with various local command authorization options

AAA configuration	Expected behavior
aaa authorization commands default group tacacs+ local	If TACACS+ server is not reachable, system allows locally authenticated users to execute the command. If TACACS+ server is reachable and if it returns an authorization failure, then the system does not perform any failover to local authentication with this configuration.
aaa authorization commands default local	This configuration allows only locally authenticated users to execute commands. System completely blocks remote users from executing any command.
aaa authorization commands default local group tacacs+	In this scenario, system chooses local authorization first and grants access if the user is locally authenticated. If not, the request fails over to TACACS+ server. This combination of command options is useful when both local and remote authenticated users want to execute commands when TACACS+ server is reachable.
aaa authorization commands default local none	Although configurable, this combination of command options does not provide any additional security with respect to user access. It is equivalent to having no authorization.

Use case scenarios of command authorization

In these scenarios:

- “Local user” refers to users authenticated and profiled locally (not on the TACACS+ server).
- “Remote user” refers to users authenticated and profiled remotely (on the TACACS+ server, not locally).
- Both types of users are assumed to have permission to execute root-lr commands.

Table 28: Use case scenarios of command authorization

Type of user (local or remote)	AAA configuration summary	Use case scenario	Expected behavior
Local and remote user	No command authorization configured	Execute a command	Command authorization succeeds if the required task-IDs are available
Local user	Only tacacs+ command authorization configured.	Execute a command when TACACS+ server is reachable	Command authorization fails
Execute a command when TACACS+ server is not reachable	Command authorization fails		

Type of user (local or remote)	AAA configuration summary	Use case scenario	Expected behavior
Remote user	Only tacacs+ command authorization configured	Execute a command when TACACS+ server is reachable	Command authorization succeeds `Router#show run aaa authorization aaa authorization commands default group tacacs+`
Execute a command when TACACS+ server is not reachable	Command authorization fails		
Local user	Only tacacs+ command authorization configured with failover option as none.	Execute a command when TACACS+ server is reachable	Command authorization fails
Execute a command when TACACS+ server is not reachable	Command authorization succeeds `Router#show user authentication method local`		
Remote user	Only tacacs+ command authorization configured with failover option as none.	Execute a command that is restricted only to that user when TACACS+ server is reachable	Command authorization fails
Execute a command that is restricted only to that user when TACACS+ server is not reachable	Command authorization succeeds		
Local user	Only local command authorization configured.	Execute a command	Command authorization succeeds `Router#show run aaa authentication aaa authentication login default group tacacs+ local`
Remote user	Only local command authorization configured.	Execute a command	Command authorization fails
Local user	Only tacacs+ command authorization configured with failover option as local.	Execute a command when TACACS+ server is reachable	Command authorization fails

Type of user (local or remote)	AAA configuration summary	Use case scenario	Expected behavior
Execute a command when TACACS+ server is not reachable	Command authorization succeeds `Router#show run aaa authorization aaa authorization commands default group tacacs+ local`		
Remote user	Only tacacs+ command authorization configured with failover option as local.	Execute a command when TACACS+ server is reachable	Command authorization succeeds `Router#show run aaa authorization aaa authorization commands default group tacacs+ local`
Execute a command when TACACS+ server is not reachable	Command authorization fails		

Additional reference information

Use these tables and scenarios to compare:

- TACACS+ only authorization
- TACACS+ with local fallback
- Local-only authorization
- Authentication method combinations

Refer to the examples to predict command execution results for locally authenticated and remotely authenticated users when the TACACS+ server is reachable or unreachable.

Configuration patterns for AAA services

Provides example configuration patterns for deploying AAA services in various network scenarios.

The following examples show how to configure AAA services for various authentication, authorization, and accounting scenarios.

Authentication method lists

An authentication method list named **vty-authen** is configured to use all TACACS+ servers for authentication. If TACACS+ fails, authentication falls back to the local username database.

```
configure
aaa authentication login vty-authen group tacacs+ local
```

The default method list for PPP uses the local method.

```
aaa authentication ppp default local
```

Local user configuration

Configure users with secure passwords and specify user groups.

```
username user1
secret lab
group root-lr
exit

username user2
secret lab
exit
```

Task groups and user groups

Create a task group, assign tasks, and create a user group that inherits permissions from the task group. Add a description to the user group.

```
taskgroup tga
task read bgp
task write ospf
exit

usergroup uga
taskgroup tga
description usergroup uga
exit
```

Configure user2 to inherit from user group **uga**:

```
username user2
group uga
exit
```

TACACS+ server configuration

Add three TACACS servers with specified keywords.

```
tacacs-server host 10.1.1.1 port 1 key abc
tacacs-server host 10.2.2.2 port 2 key def
tacacs-server host 10.3.3.3 port 3 key ghi
```

User group for privilege levels

Create a user group named **priv5** for users authenticated via TACACS+ with privilege level 5:

```
usergroup priv5
taskgroup operator
exit
```

Authorization and accounting method lists

Configure authorization and accounting method lists using TACACS+ servers.

```
aaa authorization commands vty-author group tacacs+
aaa accounting commands vty-acct start-stop group tacacs+
```

Line template configuration

Configure line template **vty** with line password, group assignment, authentication, authorization, and accounting.

```
line template vty
password lab
users group uga
login authentication vty-authen
authorization commands vty-author
accounting commands vty-acct
exit
```

TACACS+ server group

Create a TACACS+ server group named `abc` and add a configured server.

```
aaa group server tacacs+ abc
server 10.3.3.3
exit
```

7 RADIUS Server Configuration and Secure Transport

Topics:

- RADIUS protocol
- RADIUS with DTLS protection
- RADIUS with TLS protection
- Router-to-RADIUS server communications
- RADIUS dead-server detection features
- RADIUS server groups
- Per-VRF AAA for RADIUS
- SSH remote public key authentication using TACACS+

Outlines RADIUS server functionality, secure transport methods with DTLS and TLS, router integration, dead-server detection, server group configuration, and per-VRF AAA techniques to enhance authentication, authorization, and security for network infrastructures.

RADIUS protocol

Explains how the RADIUS client/server protocol supports Cisco IOS XR AAA by centralizing authentication and accounting requests for remote network access.

A RADIUS protocol is a distributed client/server protocol that

- secures network access by sending authentication and accounting requests from Cisco IOS XR routers to a central RADIUS server
- enables RADIUS clients on Cisco routers to forward user authentication and network-service access information to a central server, and
- operates with other AAA security protocols, including TACACS+, Kerberos, and local username lookup.

Additional reference information

RADIUS is a distributed client/server system that secures networks against unauthorized access. In the Cisco implementation, RADIUS clients run on Cisco routers and send authentication and accounting requests to a central RADIUS server that contains all user authentication and network service access information.

RADIUS is a fully open protocol, distributed in source code format, that can be modified to work with any security system currently available on the market.

Cisco supports RADIUS under its AAA security paradigm. RADIUS can be used with other AAA security protocols, such as TACACS+, Kerberos, and local username lookup.

> **Note**
>
> RADIUS is supported on all Cisco platforms, but some RADIUS-supported features run only on specified platforms.

RADIUS has been implemented in a variety of network environments that require high levels of security while maintaining network access for remote users.

Use RADIUS in the following network environments that require access security:

- Networks with multiple-vendor access servers, each supporting RADIUS. For example, access servers from several vendors use a single RADIUS server-based security database. In an IP-based network with multiple vendors' access servers, dial-in users are authenticated through a RADIUS server that has been customized to work with the Kerberos security system.
- Turnkey network security environments in which applications support the RADIUS protocol, such as in an access environment that uses a "smart card" access control system. In one case, RADIUS has been used with Enigma security cards to validate users and grant access to network resources.
- Networks already using RADIUS. You can add a Cisco router with RADIUS to the network. This might be the first step when you make a transition to a Terminal Access Controller Access Control System Plus (TACACS+) server.
- Networks in which a user must access only a single service. Using RADIUS, you can control user access to a single host, utility such as Telnet, or protocol such as Point-to-Point Protocol (PPP). For example, when a user logs in, RADIUS identifies this user as having authorization to run PPP using IP address 10.2.3.4 and the defined access list is started.
- Networks that require resource accounting. You can use RADIUS accounting independent of RADIUS authentication or authorization. The RADIUS accounting functions allow data to be sent at the start and end of services, indicating the amount of resources (such as time, packets, bytes, and so on) used during the session. An Internet service

provider (ISP) might use a freeware-based version of RADIUS access control and accounting software to meet special security and billing needs.

- Networks that support preauthentication. Using the RADIUS server in your network, you can configure AAA preauthentication and set up the preauthentication profiles. Preauthentication enables service providers to better manage ports using their existing RADIUS solutions and to efficiently manage the use of shared resources to offer differing service-level agreements.

RADIUS unsuitable network security situations

Lists the network security situations where RADIUS is not the suitable AAA protocol because the environment requires stronger protocol coverage, router authentication, or multi-service authorization.

RADIUS is not suitable in the following network security situations:

- Multiprotocol access environments. RADIUS does not support the following protocols:
  - AppleTalk Remote Access (ARA)
  - NetBIOS Frame Control Protocol (NBFCP)
  - NetWare Asynchronous Services Interface (NASI)
  - X.25 PAD connections
- Router-to-router situations: RADIUS does not provide two-way authentication. RADIUS can be used to authenticate from one router to a router other than a Cisco router if that router requires RADIUS authentication.
- Networks using a variety of services: RADIUS generally binds a user to one service model.

Use this reference to identify environments where RADIUS limitations affect protocol support, router-to-router authentication, or multi-service authorization.

Choose an alternate AAA design when the environment needs two-way router authentication, unsupported access protocols, or service models that RADIUS cannot represent.

How RADIUS authentication works

Describes the RADIUS authentication flow from user login to server response, including handling of Access-Accept, Access-Reject, challenge, password-change, and authorization-data stages.

RADIUS authentication begins when a user logs in to an access server by providing a username and password. The access server encodes the credentials and forwards them to the RADIUS server.

Summary

The key components involved in the process are:

- User: Provides a username and password to initiate authentication.
- Access server: Sends the submitted credentials to the RADIUS server.
- RADIUS server: Validates the credentials and determines access and service authorization.

RADIUS authentication is a network access control process in which a user logs in to an access server, submits credentials, and receives authorization from a RADIUS server based on those credentials and additional data returned.

Workflow

These stages describe how RADIUS authentication works.

1. Credential Entry: The user is prompted for and enters a username and password.
2. Transmission: The access server sends the username and encrypted password to the RADIUS server.

3. Server Response: The RADIUS server responds with one of the following:
   - Access-Accept: The user is authenticated.
   - Access-Reject: The user is denied access or prompted to re-enter credentials.
   - Challenge: The server issues a challenge to collect additional information from the user.
   - Change Password: The server asks the user to select a new password.
4. Authorization Data: Along with the Accept or Reject response, additional data for XR EXEC mode or network authorization is included:
   - Services the user can access (Telnet, rlogin, local-area transport, PPP, SLIP, or XR EXEC mode).
   - Connection parameters (host or client IP address, access list, user timeouts).
5. Completion: RADIUS authentication must succeed before RADIUS authorization is used.

Result

After RADIUS authentication succeeds, additional response data can authorize XR EXEC mode or network services.

RADIUS with DTLS protection

Explains how Cisco IOS XR uses Datagram Transport Layer Security as the RADIUS transport protocol to protect UDP-based RADIUS communication with encryption and peer authentication.

A RADIUS DTLS protection feature is a secure transport feature that

- uses Datagram Transport Layer Security to protect RADIUS packets exchanged between the Cisco IOS XR RADIUS client and the RADIUS server
- allows RADIUS to continue operating over UDP while adding encryption and peer authentication, and
- is configured per RADIUS server host by using the DTLS server option and a trustpoint.

Feature history
Table 29: Feature History Table

Feature Name	Release Information	Feature Description
RADIUS with DTLS Protection	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
RADIUS with DTLS Protection	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
RADIUS with DTLS Protection	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
RADIUS with DTLS Protection	Release 24.2.11	You can now secure communication for RADIUS packets by using Datagram Transport Layer Security (DTLS) as the transport layer for the RADIUS protocol. The RADIUS protocol continues to operate over UDP but now benefits from the added security provided by DTLS. Utilizing DTLS enables the manual distribution of long-term proof of peer identity through TLS-PSK cipher suites and the option to use X509 certificates in a PKI infrastructure. In the absence of DTLS, RADIUS packets may be subject to potential security vulnerabilities, including data exposure, replay attacks, weak authentication, and encryption vulnerabilities, especially when transmitted across untrusted networks. The feature introduces these changes: **CLI:** • The keyword **dtls-server** is introduced in the **radius-server host** command. **YANG Data Models:** • New Xpath for `Cisco-IOS-XR-um-aaa-cfg.yang` • New Xpath for `Cisco-IOS-XR-aaa-lib-cfg.yang` (see GitHub, YANG Data Models Navigator)

RADIUS DTLS behavior

Traditionally, RADIUS has been used for Authentication, Authorization, and Accounting (AAA). To meet modern security demands, it is important to enhance its encryption and authentication. By addressing these areas, RADIUS's resilience against threats and overall network security are improved.

Datagram Transport Layer Security (DTLS) is now utilized as the transport protocol for RADIUS to enhance security. This modification allows RADIUS to function over UDP while benefiting from DTLS's encryption and peer authentication features.

Security and deployment benefits of RADIUS DTLS protection

Lists the security and deployment benefits of protecting RADIUS communication with DTLS, including cipher-suite support, certificate-based identity, replay protection, and transport reliability.

The security and deployment benefits of RADIUS DTLS protection include:

- TLS-PSK cipher suites: Enable the secure distribution of long-term proof of peer identity by manually sharing a pre-shared key (PSK) between peers. This establishes a secure TLS connection without relying on traditional Public Key Infrastructure (PKI), making it ideal for environments where certificates are impractical.
- X.509 certificate support: Provides robust identity verification using PKI and X.509 certificates. This approach standardizes certificate management, increases security, and facilitates trust across diverse, distributed environments.
- Replay protection: Prevents retransmission attacks by ensuring each packet is uniquely authenticated and cannot be reused maliciously.
- Transport reliability: Enhances data encryption and ensures reliable delivery of RADIUS messages, reducing the risk of dropped or corrupted packets during transport.

How to use this reference

Use this information to compare the advantages of TLS-PSK cipher suites, X.509 certificates, replay protection, and failover behavior for RADIUS over DTLS. Apply these benefits when selecting a RADIUS transport mode in environments requiring strong peer identity and secure, encrypted packet exchange.

How RADIUS with DTLS protection works

Describes the RADIUS over DTLS workflow, including DTLS session establishment, context storage, packet handling, secure transmission, and optimized session control.

RADIUS over DTLS enhances the security of traditional RADIUS communications, especially when traffic traverses untrusted networks or roaming environments. By using DTLS (Datagram Transport Layer Security), session confidentiality, message integrity, and improved session control are achieved over standard UDP transport.

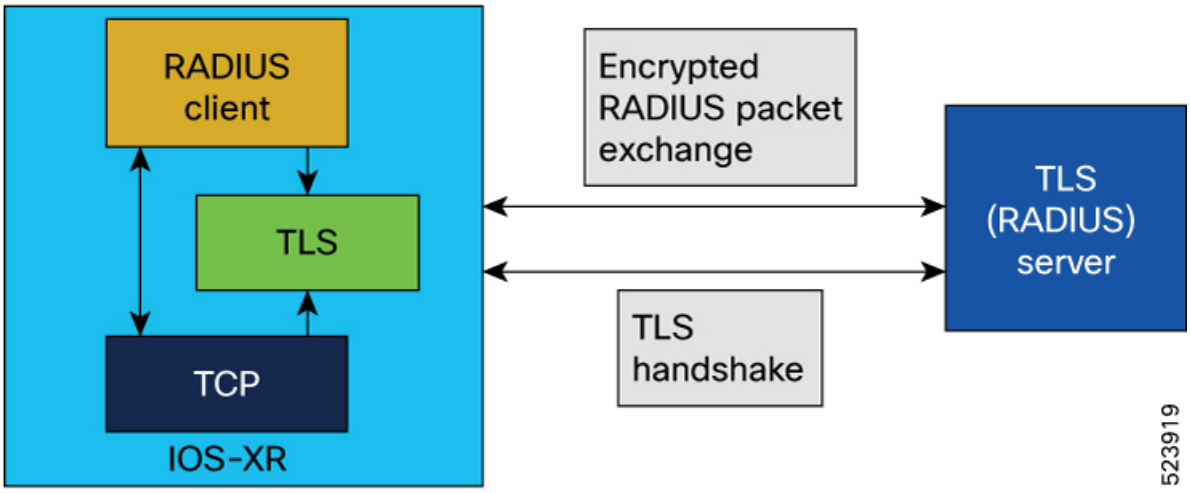
Summary

The key components involved in the process are:

- RADIUS client: Initiates RADIUS communications, establishes DTLS sessions, handles RADIUS packet creation, and manages session monitoring.
- RADIUS server: Accepts DTLS session requests, processes authentication and accounting packets, and maintains session security.
- DTLS layer: Provides encryption, decryption, and integrity for RADIUS packets, and manages session health via heartbeats and watchdogs.

RADIUS with DTLS protection secures RADIUS communications by establishing a DTLS session for encrypted transport, maintaining session context, and optimizing packet handling and session monitoring.

Workflow



These stages describe how RADIUS with DTLS protection works.

1. Establish DTLS session: The RADIUS client initiates a secure DTLS session with the RADIUS server, using a UDP socket that it creates for communication.
2. Store DTLS context: Once the DTLS connection is successful, the DTLS context is preserved within the connection and then stored in the RADIUS context for the specified server, ensuring a secure channel for subsequent traffic.
3. RADIUS packet handling: The RADIUS client constructs standard RADIUS packets. Instead of sending these via UDP, it hands them to the DTLS layer, which encapsulates them for secure transport. The standard packet format remains unchanged.
4. Secure data transmission: RADIUS packets are transmitted over DTLS, ensuring data confidentiality and integrity through encryption and decryption.
5. Optimized RADIUS session control: The RADIUS client performs Path MTU discovery before sending traffic; it uses different sockets for RADIUS/UDP and RADIUS/DTLS traffic. DTLS heartbeats and application-layer watchdogs track session health. The client proactively closes idle or unresponsive sessions and terminates sessions if packet validation fails or invalid authenticators are detected.

Result

RADIUS with DTLS protection enables secure, reliable authentication and accounting by retaining the standard RADIUS packet format while using DTLS for confidentiality, integrity, and advanced session monitoring.

Requirement: Use supported RADIUS DTLS behavior

Outlines the supported RADIUS DTLS port, IPv4, fallback, and server-group behavior that must guide a RADIUS DTLS deployment on Cisco IOS XR routers.

Follow these requirements when deploying RADIUS over DTLS on Cisco IOS XR routers:

- RADIUS/DTLS is supported only for IPv4.
- The default destination port for RADIUS/DTLS is UDP/2083. There are no separate ports for authentication, accounting, or dynamic authorization changes. The source port can be arbitrary.
- Use DTLS as a transport protocol only when you configure it administratively. If the server is unresponsive and DTLS is enabled, the client does not fall back to RADIUS/UDP.
- Create separate AAA server groups for DTLS-capable and non-DTLS servers, as server group failover will only process packets on DTLS-capable servers within a group. If a server group mixes DTLS and non-DTLS servers, processing stops when a non-DTLS server is encountered, and the client starts over with another group.
- Do not use RADIUS over DTLS for BNG use cases.

- Ensure that RADIUS over DTLS uses only the following ciphers:
  - ECDHE-ECDSA-AES256-GCM-SHA384
  - ECDHE-ECDSA-AES128-GCM-SHA256
  - ECDHE-ECDSA-AES256-SHA384
  - ECDHE-ECDSA-AES128-SHA256
  - ECDHE-RSA-AES256-GCM-SHA384
  - ECDHE-RSA-AES128-GCM-SHA256
  - ECDHE-RSA-AES256-SHA384
  - ECDHE-RSA-AES128-SHA256
  - AES256-SHA256 and AES128-SHA256

Configure RADIUS with DTLS protection

Configure a RADIUS server host with DTLS protection using a trustpoint, and verify that DTLS is enabled for secure communication.

Protect RADIUS packets over DTLS by configuring the RADIUS server host to use DTLS and a specified trustpoint.

Use this task when your RADIUS server supports DTLS and the router must send RADIUS packets through the DTLS transport to enhance security.

Before you begin

Ensure the trustpoint is configured before enabling DTLS for the RADIUS server host.

Procedure

1. Configure the RADIUS server host with DTLS protection.

   Example

```
Router# configure
Router(config)#radius-server host 209.165.201.1 auth-port 2083 acct-port 2083
Router(config-radius-host)#dtls-server trustpoint test
Router(config-radius-host)#commit
```

2. Review the RADIUS DTLS running configuration.

   Example

```
Router# show running-config
radius-server host 209.165.201.1 auth-port 2083 acct-port 2083
 dtls-server trustpoint test
!
```

3. Verify that DTLS is enabled for the RADIUS server.

Example

```
Router#show radius
Tue May 28 09:00:45.207 UTC
Global dead time: 0 minute(s)
Number of Servers: 1

Server: 209.165.201.1/2083/2083  is UP
  Address family: IPv4
  Total Deadtime: 0s Last Deadtime: 0s
  Timeout: 5 sec, Retransmit limit: 3
  Quarantined: No
  Authentication:
    0 requests, 0 pending, 0 retransmits
    0 accepts, 0 rejects, 0 challenges
    0 timeouts, 0 bad responses, 0 bad authenticators
    0 unknown types, 0 dropped, 0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Access Transactions: 0
    Maximum Throttled Access Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
 dtls:enabled
  Accounting:
    0 requests, 0 pending, 0 retransmits
    0 responses, 0 timeouts, 0 bad responses
    0 bad authenticators, 0 unknown types, 0 dropped
    0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Accounting Transactions: 0
    Maximum Throttled Accounting Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
```

Verify that DTLS is enabled using the **show radius** command.

The RADIUS server host is configured with DTLS protection. The **show radius** output indicates that DTLS is enabled, ensuring secure communication for RADIUS packets.

RADIUS with TLS protection

Explains how Cisco IOS XR redirects RADIUS packets to a remote server over TLS so AAA traffic receives encrypted transport and peer protection.

A RADIUS TLS protection is a secure transport mechanism that

- protects RADIUS packets by using TLS between the Cisco IOS XR RADIUS client and a remote RADIUS server
- reduces exposure to data disclosure, replay attacks, weak authentication, and encryption weaknesses, and
- supports TLS version 1.3.

Feature history
Table 30: Feature History Table

Feature Name	Release Information	Feature Description
RADIUS with TLS protection	Release 26.2.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

RADIUS with TLS protection	Release 24.4.1	Remote Authentication Dial-In User Service (RADIUS) packets are now less vulnerable to security risks, including data exposure, replay attacks, weak authentication, and encryption weaknesses. This is because we have enabled support for RADIUS with TLS protection.
		
		You can configure the RADIUS protocol on the router to redirect RADIUS packets to a remote server over TLS for Authentication, Authorization, and Accounting (AAA) services.
		
		The feature introduces these changes:
		
		CLI:
		
		• The keyword **radsec-server** is introduced in the **radius-server host** command.
		
		YANG Data Models:
		
		• New Xpath for
		
		`Cisco-IOS-XR-um-aaa-cfg.yang`
		
		• New Xpath for
		
		`Cisco-IOS-XR-aaa-lib-cfg.yang`
		
		(see GitHub, YANG Data Models Navigator)

How RADIUS with TLS protection works

Describes the RADIUS over TLS workflow, including TLS session establishment, context storage, packet handling, secure transmission, and optimized session control.

Traditionally, RADIUS operated over UDP or TCP, which lacked robust security. With modern security demands, transporting RADIUS over TLS offers enhanced protection and session monitoring. This feature supports TLS version 1.3.

Let's understand how you can establish RADIUS communication with TLS.

Summary

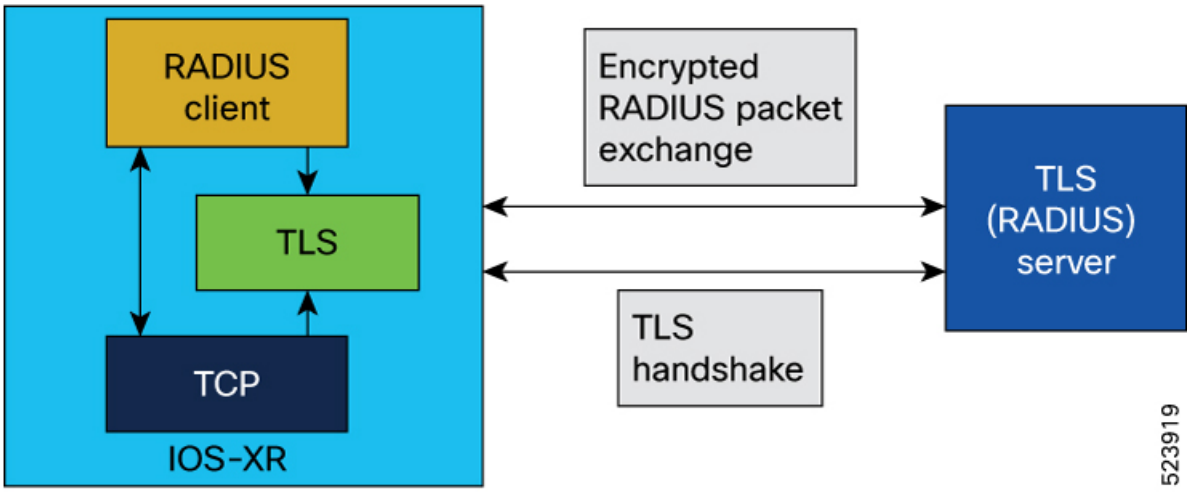
The key components involved in the process are:

- RADIUS client: Initiates the TLS session and manages session monitoring.
- RADIUS server: Acts as the TLS server, receiving and processing RADIUS packets over TLS.
- TLS layer: Provides secure transport, data encryption, integrity, and connection monitoring for RADIUS packets.

RADIUS with TLS protection enhances the traditional Authentication, Authorization, and Accounting (AAA) protocol by implementing TLS version 1.3 as the transport layer. This provides improved encryption, authentication, and session control, increasing resilience against threats and ensuring a secure network environment.

Workflow

Figure 6: Topology for RADIUS with TLS protection



These stages describe how RADIUS with TLS protection works.

1. Establish TLS session: The RADIUS client initiates the process to establish a secure TLS session with the RADIUS server, which acts as the TLS server. This session is built upon a TCP socket that the RADIUS client creates.
2. Store TLS context: After establishing the TLS connection, the TLS context is preserved within the TLS connection context and stored in the RADIUS context for the server, ensuring a persistent secure environment.
3. RADIUS packet handling: The client constructs standard RADIUS packets, which are handed to the TLS layer for secure encapsulation and transmission, rather than sending them directly through a TCP socket.
4. Secure data transmission: RADIUS packets are transmitted securely over TLS, leveraging data encryption and decryption.
5. Optimized RADIUS session control:
   - The client employs Path MTU discovery before sending traffic.
   - Different source sockets are used for RADIUS/UDP and RADIUS/TLS traffic to different servers.
   - TLS heartbeats are used for ongoing connectivity monitoring.
   - An application-layer watchdog checks server responsiveness; idle or non-responsive sessions are proactively closed.
   - Sessions are terminated if packets fail validation or contain invalid authenticators. Failovers are initiated as needed.

Result

RADIUS over TLS maintains the standard packet format while providing confidentiality, integrity, and robust session monitoring—especially in roaming environments or across untrusted networks.

Requirement: Use supported RADIUS TLS behavior

Outlines the supported RADIUS TLS restrictions that must guide secure RADIUS over TLS deployments.

To ensure secure and compliant RADIUS over TLS deployments, you must adhere to these requirements:

- Do not use Broadband Network Gateway (BNG) applications with RADIUS over TLS.
- Use TCP port 2083 as the destination for RADIUS over TLS authentication and accounting; custom ports are not supported.
- Limit the maximum number of concurrent TLS sessions to 50.
- Use only IPv4 addresses; IPv6 is not supported for RADIUS over TLS.
- Do not configure server groups that combine TLS, UDP, and DTLS server types under one server group for RADIUS.

Supported RADIUS TLS cipher suites

Lists the TLS cipher suites that Cisco IOS XR supports for securing RADIUS traffic between the router and the external RADIUS TLS environment.

TLS cipher suites are encryption algorithms that secure RADIUS traffic between Cisco IOS XR routers and external RADIUS TLS servers.

- TLS_AES_256_CGM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_AES_128_CGM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CGM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CGM_SHA384
- TLS_DHE_RSA_WITH_AES_256_CGM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV

The cipher suite negotiated between the client and the server when both support TLS 1.3 is:

- TLS_AES_256_GCM_SHA384

Using this reference

- Use this reference to determine which TLS cipher suites you can select for securing RADIUS traffic between Cisco IOS XR routers and external RADIUS TLS environments.
- When configuring the external RADIUS TLS server, select a cipher suite from this list to ensure compatibility with Cisco IOS XR.

Configure RADIUS with TLS protection

Configure a RADIUS server host to use TLS protection with the radsec-server option and a trustpoint, then verify with operational and running-configuration commands.

Protect RADIUS packets using TLS by configuring the server host with the radsec-server option and a trustpoint.

To secure RADIUS communication with TLS, use the **radius-server host** command with the **radsec-server** keyword.

Before you begin

- Configure a trustpoint on the router.
- Import the CA certificate.
- Enroll the trustpoint and generate a client certificate from the Certification Authority (CA).
- Import the client certificate.

Procedure

1. Enter the hostname or IP address of the RADIUS server.

   Example

   ```
   Router(Config)# radius-server host 209.165.201.1 auth-port 2083 acct-port
   2083 radsec-server
   ```

2. Enter the name of the trustpoint so that the router can verify certificates issued to peers.

Example

```
Router(config-radius-host)# trustpoint test
```

Your router does not need to enroll with the CA that issued the certificates to the peers.

3. Commit the configuration changes.

Example

```
Router(config-radius-host)# commit
```

4. Use the **show radius** command to verify that TLS is enabled.

Confirm that the server-type is shown as TLS.

Example

```
Router#show radius
Thu Jun 20 11:43:40.863 UTC
Global dead time: 0 minute(s)
Number of Servers: 3

Server: 209.165.201.1/2083/2083  is UP
  Address family: IPv4
  Total Deadtime: 0s Last Deadtime: 0s
  Timeout: 5 sec, Retransmit limit: 3
  Quarantined: No
  Authentication:
    0 requests, 0 pending, 0 retransmits
    0 accepts, 0 rejects, 0 challenges
    0 timeouts, 0 bad responses, 0 bad authenticators
    0 unknown types, 0 dropped, 0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Access Transactions: 0
    Maximum Throttled Access Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
  Server-type: TLS
  Accounting:
    0 requests, 0 pending, 0 retransmits
    0 responses, 0 timeouts, 0 bad responses
    0 bad authenticators, 0 unknown types, 0 dropped
    0 ms latest rtt
    Throttled: 0 transactions, 0 timeout, 0 failures
    Estimated Throttled Accounting Transactions: 0
    Maximum Throttled Accounting Transactions: 0

    Automated TEST Stats:
        0 requests, 0 timeouts, 0 response, 0 pending
```

5. Verify the configuration settings by using the show running-configuration command.

Example

```
Router# show running-configuration radius-server
Fri Jun 21 02:59:40.238 UTC
radius-server host 209.165.201.1 auth-port 2083 acct-port 2083
```

```
radsec-server trustpoint test
!
```

The RADIUS server host is configured to use TLS protection. The output from the operational and running-configuration commands confirms the settings.

Router-to-RADIUS server communications

Explains the RADIUS server host communication parameters that Cisco IOS XR uses to reach external RADIUS servers for authentication and accounting services.

A router-to-RADIUS server communication configuration is a RADIUS client configuration that

- identifies external RADIUS server hosts by hostname, IP address, and authentication or accounting UDP port numbers
- allows per-server timeout, retransmission, and key values to override global RADIUS values when both are configured, and
- enables the source-interface setting to force outgoing RADIUS packets to use a specific interface or VRF-specific interface.

Additional reference information

Router-to-RADIUS server communications are configured to define how the router interacts with external RADIUS server hosts. The RADIUS host is typically a multiuser system running RADIUS server software from providers such as Cisco (CiscoSecure ACS), Livingston, Merit, or Microsoft. Key configurable components include:

- Hostname or IP address
- Authentication destination port
- Accounting destination port
- Retransmission value
- Timeout period
- Key string

RADIUS security servers are uniquely identified using either hostname or IP address, TCP/UDP port numbers, or combinations thereof. This enables RADIUS requests to be directed to multiple ports on the same server. If multiple host entries exist for the same service (such as accounting), backup switching is automatic–the second entry is used if the first fails. Host entries are tried in configuration order.

A router and RADIUS server use a shared secret (key string) to encrypt passwords and exchange responses. Global parameters can be set for timeout, retransmission, and encryption key values; these can also be specified per server and per global/per-server combinations. To set parameters globally, use the commands: **radius-server timeout**, **radius-server retransmit**, and r**adius-server key**. For per-server settings, use the **radius-server host** command.

You can configure a maximum of 30 global RADIUS servers.

> **Note**
>
> You can configure both global and per-server timeout, retransmission, and key value commands at the same time. If both are configured, the per-server settings override the global settings.

RADIUS summary example

```
radius source-interface Mgm0/rp0/cpu0/0 vrf default
radius-server timeout 10
radius-server retransmit 2
!
! OOB RADIUS
radius-server host 192.0.2.10 auth-port 1812 acct-port 1813
key cisco123
timeout 10
retransmit 2
!
radius-server host 192.0.2.11 auth-port 1812 acct-port 1813
key cisco123
timeout 10
retransmit 2
!
aaa group server radius radgrp
server 192.0.2.10 auth-port 1812 acct-port 1813
server 192.0.2.11 auth-port 1812 acct-port 1813
!
aaa authorization exec radauthen group radgrp local
aaa authentication login radlogin group radgrp local
!
line template vty
authorization exec radauthen
login authentication radlogin
timestamp disable
exec-timeout 0 0
!
vty-pool default 0 99 line-template vty
```

Configure router to communicate with a RADIUS server

Configure global RADIUS server communication values, define server-specific parameters, and optionally display RADIUS server information for verification.

Set up router communication with one or more external RADIUS servers by specifying global and server-specific values for authentication, accounting, timing, security, and interface sources.

Use this task to establish proper communication between the router and external RADIUS servers. You'll define global defaults, configure specific server hosts, and verify operation as needed.

Before you begin

- Ensure the external RADIUS server is reachable from the router.
- Gather required information such as server IP addresses or hostnames, authentication and accounting port numbers, retransmission attempts, timeout intervals, and shared secrets.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Configure the RADIUS server host.

   Example

   ```
   Router(config)# radius-server host host1
   ```

   Specifies the hostname or IP address of the remote RADIUS server host.

   - Use the auth-port *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for authentication.
   - Use the acct-port *port-number* option to configure a specific UDP port on this RADIUS server to be used solely for accounting.
   - To configure the network access server to recognize more than one host entry associated with a single IP address, simply repeat this command as many times as necessary, making sure that each UDP port number is different. Set the timeout, retransmit, and encryption key values to use with the specific RADIUS host.
   - If no timeout is set, the global value is used; otherwise, enter a value in the range 1 to 1000. If no retransmit value is set, the global value is used; otherwise enter a value in the range 1 to 100. If no key string is specified, the global value is used.

   > **Note**
   >
   > The key is a text string that must match the encryption key used on the RADIUS server. Always configure the key as the last item in the radius-server hostcommand syntax because the leading spaces are ignored, but spaces within and at the end of the key are used. If you use spaces in your key, do not enclose the key in quotation marks unless the quotation marks themselves are part of the key.

3. Set the global RADIUS retransmission count.

   Example

   ```
   Router(config)# radius-server retransmit 5
   ```

   Specifies the number of times the software searches the list of RADIUS server hosts before giving up.

   - In the example, the number of retransmission attempts is set to 5.

4. Set the global RADIUS timeout interval.

Example

```
Router(config)# radius-server timeout 10
```

Sets the number of seconds a router waits for a server host to reply before timing out.

- In the example, the interval timer is set to 10 seconds.

5. Set the RADIUS shared secret key.

Example

```
Router(config)# radius-server key 0 samplekey
```

Sets the authentication and encryption key for all RADIUS communications between the router and the RADIUS daemon.

6. Configure the RADIUS source interface.

Example

```
Router(config)# radius source-interface 0/3/0/1
```

(Optional) Forces RADIUS to use the IP address of a specified interface or subinterface for all outgoing RADIUS packets.

- The specified interface or subinterface must have an IP address associated with it. If the specified interface or subinterface does not have an IP address or is in the down state, then RADIUS reverts to the default. To avoid this, add an IP address to the interface or subinterface or bring the interface to the up state.

The **vrf** keyword enables the specification on a per-VRF basis.

7. Repeat step 2 through step 6 for each external server to be configured.
8. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

9. Display RADIUS server information.

Example

```
Router# show radius
```

(Optional) Displays information about the RADIUS servers that are configured in the system.

The router can now communicate with the configured RADIUS server hosts, and you can view the active RADIUS server configuration.

RADIUS dead-server detection features

Explains the time and tries criteria that Cisco IOS XR uses to mark a nonresponsive RADIUS server as dead and improve availability.

A RADIUS dead-server detection feature is a RADIUS availability feature that

- marks a RADIUS server as dead when configured or dynamically computed time and tries criteria are met
- computes criteria dynamically when no explicit criteria are configured, and
- reduces deadtime and improves packet processing by promptly detecting nonresponding servers.

Additional reference information

The RADIUS dead-server detection feature lets you configure and determine the criteria used to mark a RADIUS server as dead. If no criteria are explicitly configured, the system computes the criteria dynamically based on the number of outstanding transactions.

You can configure two main criteria:

- Time criterion: The minimum elapsed time in seconds from when the router last received a valid packet from the RADIUS server to when the server is marked as dead. If no packet has been received since the router booted and a timeout occurs, the time criterion is treated as met.
- Tries criterion: The number of consecutive timeouts on the router before the server is considered dead. If the server performs both authentication and accounting, both types of packets count. Improperly constructed packets count as timeouts. Only retransmissions are counted (each timeout causes one retransmission).

Both the time criterion and tries criterion must be met before a server is marked as dead.

The **radius-server deadtime** command specifies the duration, in minutes, for which a server remains marked as dead. After this period, the server is automatically marked alive even if no responses were received. Servers are not monitored for deadtime unless this command is configured.

For example, if you configure a deadtime interval and set criteria for timeouts and elapsed time, the feature will mark a RADIUS server as dead only when both conditions are satisfied. This ensures that brief connectivity issues do not incorrectly mark servers as dead.

Configure RADIUS dead-server detection

Configure RADIUS deadtime and dead criteria, then display dead-server detection information to verify how Cisco IOS XR tracks unresponsive RADIUS servers.

Set the conditions that Cisco IOS XR uses to mark a RADIUS server as dead and ensure prompt detection of nonresponsive servers.

The RADIUS Dead-Server Detection feature allows you to configure criteria to determine when a RADIUS server is considered "dead." If criteria are not explicitly configured, Cisco IOS XR computes them dynamically based on the number of outstanding transactions. Prompt detection avoids delays caused by swamped or failed servers, resulting in fast packet processing. You can set a minimum interval (in seconds) between receiving packets and marking a server as dead, and configure the number of consecutive timeouts required before a server is marked as dead. If the server handles both authentication and accounting, both packet types are counted. Incorrect packets also count as timeouts, and only

retransmissions are considered. Both the time and tries criteria must be met for the server to be marked as dead. The **radius-server deadtime** command specifies how long (in minutes) a server remains dead before being marked alive again. Servers are not monitored unless this command is configured.

Before you begin

Configure the RADIUS server host before configuring dead-server detection.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Configure the RADIUS server deadtime.

   Example

   ```
   Router(config)# radius-server deadtime 5
   ```

   Improves RADIUS response times when some servers might be unavailable and causes the unavailable servers to be skipped immediately.

3. Configure the RADIUS dead-server time criterion.

   Example

   ```
   Router(config)# radius-server dead-criteria time 5
   ```

   Establishes the time for the dead-criteria conditions for a RADIUS server to be marked as dead.

4. Configure the RADIUS dead-server tries criterion.

   Example

   ```
   Router(config)# radius-server dead-criteria tries 4
   ```

   Establishes the number of tries for the dead-criteria conditions for a RADIUS server to be marked as dead.

5. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - commit: Saves the configuration changes and remains within the configuration session.
   - end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.

- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

6. Display RADIUS dead-server detection information.

Example

```
Router# show radius dead-criteria host 172.19.192.80
```

(Optional) Displays dead-server-detection information that has been requested for a RADIUS server at the specified IP address.

Cisco IOS XR uses the configured time and tries criteria to mark nonresponsive RADIUS servers as dead, ensuring responsive RADIUS operation and reliable packet processing.

RADIUS server groups

Explains how Cisco IOS XR groups external RADIUS server hosts so AAA method lists can use named server groups for AAA service selection.

A RADIUS server group is a network authentication mechanism that

- consists of a named list of external RADIUS server hosts
- allows AAA method lists to reference the group for authentication, authorization, or accounting, and
- supports up to 30 servers and 30 private servers per group in Cisco IOS XR.

Additional reference information

When configuring RADIUS server groups, you can enter one or more server commands. Each server command specifies the hostname or IP address of an external RADIUS server, and can include optional port numbers. Once configured, a RADIUS server group can be referenced in AAA method lists to select the appropriate group for authentication, authorization, or accounting tasks.

You can configure a maximum of:

- 30 servers per RADIUS server group
- 30 private servers per RADIUS server group

Configure RADIUS server groups

Configure a named RADIUS server group, add external RADIUS servers, set optional group-level deadtime, and display server-group information for verification.

Create a named RADIUS server group that can be referenced in AAA method lists.

This task configures RADIUS server groups by grouping external RADIUS servers under a named group. The group can be used within AAA method lists for authentication, authorization, or accounting. You can configure up to 30 servers and 30 private servers per RADIUS server group.

Before you begin

Ensure the external RADIUS servers are accessible at the time of configuration.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create the RADIUS server group.

   Example

   ```
   Router(config)# aaa group server radius radgroup1
   ```

   Groups different server hosts into distinct lists and enters the server group configuration mode.

3. Add the RADIUS server to the server group.

   Example

   ```
   Router(config-sg-radius)# server 192.168.20.0
   ```

   Specifies the hostname or IP address of an external RADIUS server.

   - After the server group is configured, it can be referenced from the AAA method lists (used while configuring authentication, authorization, or accounting).

4. Repeat step 4 for every external server to be added to the server group named in step 3.
5. Configure the RADIUS server-group deadtime.

   Example

   ```
   Router(config-sg-radius)# deadtime 1
   ```

   Configures the deadtime value at the RADIUS server group level.

   - The *minutes* argument specifies the length of time, in minutes, for which a RADIUS server is skipped over by transaction requests, up to a maximum of 1440 (24 hours). The range is from 1 to 1440.

   The example specifies a one-minute deadtime for RADIUS server group radgroup1 when it has failed to respond to authentication requests for the **deadtime** command

   > **Note**
   >
   > You can configure the group-level deadtime after the group is created.

6. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

7. Display RADIUS server group information.

Example

```
Router# show radius server-groups
```

(Optional) Displays information about each RADIUS server group that is configured in the system.

The RADIUS server group is configured and available for use in AAA method lists.

Per-VRF AAA for RADIUS

Explains how Cisco IOS XR utilizes VPN routing and forwarding instances to provide AAA services, enabling RADIUS communications to leverage VRF-specific server reachability.

A Per-VRF AAA service is a service model that

- enables provider edge or virtual home gateways to communicate directly with a RADIUS server associated with a VPN
- removes the need to route RADIUS communication through a RADIUS proxy, and
- gives VPN owners more flexibility in how their RADIUS servers are reached.

Additional reference information

The Per-VRF AAA functionality enables AAA services to be based on VPN routing and forwarding (VRF) instances. The Provider Edge (PE) or Virtual Home Gateway (VHG) communicates directly with the customer's RADIUS server, which is associated with the customer's VPN, without the need for a RADIUS proxy. Internet service providers (ISPs) can scale their VPN offerings more efficiently and provide customers with greater flexibility by eliminating reliance on RADIUS proxies.

Cisco RADIUS vendor-specific attributes for per-VRF AAA

Provides the Cisco RADIUS vendor-specific attribute format and supported VSA names that per-VRF AAA uses for server, source-interface, and VRF information.

The Internet Engineering Task Force (IETF) draft standard specifies a method for communicating vendor-specific information between the network access server and the RADIUS server by using the vendor-specific attribute (attribute 26). Attribute

26 encapsulates vendor-specific attributes, allowing vendors to support their own extended attributes not suitable for general use.

Cisco IOS XR RADIUS implementation supports the vendor-specific option using the format recommended in the specification. Cisco's vendor-ID is 9, and the supported option has vendor-type 1, named *cisco-avpair*. The value follows this format:

```
protocol: attribute sep value
```

- Protocol: Cisco protocol value for a particular type of authorization.
- Attribute and value: An attribute-value pair defined in the Cisco RADIUS specification.
- Sep: Use "=" for mandatory attributes and "*" for optional attributes.

Supported Cisco VSAs for Per VRF AAA

All RADIUS VSAs listed below–rad-serv, rad-serv-source-if, and rad-serv-vrf–require the prefix **aaa**: before the VSA name.

Table 31: Supported VSAs for per VRF AAA

VSA Name	Value Type	Description
Note The RADIUS VSAs - rad-serv, rad-serv-source-if, and rad-serv-vrf - must have the prefix "aaa:" before the VSA name.		
Note	The RADIUS VSAs - rad-serv, rad-serv-source-if, and rad-serv-vrf - must have the prefix "aaa:" before the VSA name.	

VSA Name	Value Type	Description

rad-serv	string	Indicates the IP address in IPv4 or IPv6 format, key, timeout, and retransmit number of a server and the group of the server.
		The VSA syntax follows:
		`rad-serv=a.b.c.d [key SomeKey] [auth-port X] [acct-port Y]`
		`[retransmit V] [timeout W].`
		Other than the IP address, all parameters are optional and are issued in any order. If the optional parameters are not specified, their default values are used.
		The key cannot contain any spaces; for "retransmit V," "V" can range from 1 to 100; for "timeout W," the "W" can range from 1 to 1000.

rad-serv-vrf	string	Specifies the name of the VRF that is used to transmit RADIUS packets. The VRF name matches the name that was specified through the **vrf** command.

Additional Information

- Use this reference to look up Cisco vendor-specific attribute 26, Cisco vendor ID 9, vendor type 1, cisco-avpair formatting, and the supported per-VRF AAA VSA names.
- To configure the external RADIUS server correctly, ensure it returns the correct `aaa:`-prefixed VSA names and values for per-VRF AAA server, source-interface, and VRF information.

This task configures RADIUS server groups per VRF. For information about configuring TACACS+ server groups per VRF, refer *Configure TACACS+ Server Groups* section.

Configure per-VRF AAA server groups

Configure a per-VRF RADIUS server group with private RADIUS servers and associate it with a VRF reference so AAA services can use VRF-specific reachability.

Create a RADIUS server group that uses private server addresses and associates the group with a specific VRF.

Use per-VRF RADIUS server groups to direct AAA traffic over certain VRFs, allowing network segmentation and improved security by restricting authentication requests to servers reachable within the specified VRF.

Before you begin

Review the supported vendor-specific attributes for per-VRF AAA in your environment before configuring the server group.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create the RADIUS server group.

   Example

   ```
   Router(config)# aaa group server radius radgroup1
   ```

   Groups different server hosts into distinct lists and enters the server group configuration mode.

3. Configure the private RADIUS server for the group.

   Example

   ```
   Router(config-sg-radius)# server-private 10.1.1.1 timeout 5
   Router(config-sg-radius)# server-private 10.2.2.2 retransmit 3
   ```

   Example

   ```
   Router(config-sg-radius)# server-private 2001:db8:a0b:12f0::1/64 timeout 5
   Router(config-sg-radius)# server-private 10.2.2.2 retransmit 3
   ```

   Configures the IP address of the private RADIUS server for the group.

   If private server parameters are not specified, global configurations are used. If global configurations are not specified, default values are used.

   Both **auth-port** and **acct-port** keywords enter RADIUS server-group private configuration mode.

   You can configure a maximum of 30 private servers per RADIUS server group.

4. Configure the VRF reference for the RADIUS server group.

   Example

   ```
   Router(config-sg-radius)# vrf v2.44.com
   ```

   Configures the VRF reference of an AAA RADIUS server group.

   > **Note**
   >
   > Private server IP addresses can overlap with those configured globally and the VRF definitions can help to distinguish them.

5. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- commit: Saves the configuration changes and remains within the configuration session.
- end: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The RADIUS server group is configured to use the specified private RADIUS servers and VRF reference for per-VRF AAA communication.

SSH remote public key authentication using TACACS+

Describes SSH remote public key authentication with TACACS+ integration and explains centralized SSH key management and secure access benefits.

A SSH remote public key authentications using TACACS+ is a feature that

- enables centralized retrieval and validation of SSH public keys through TACACS+ AAA servers,
- supports fallback to `authorized_keys` files based on configuration settings, and
- provides real-time access control and unified auditing across managed network devices.

Table 32: Feature History Table

Feature Name	Release Information	Feature Description
SSH remote public key authentication using TACACS+	Release 26.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*). You can centrally manage SSH public keys on IOS XR by enabling devices to retrieve and validate user public keys from TACACS+ servers at login. This feature simplifies key management, streamlines access control, and supports fallback to local keys for continued access if the server is unavailable.

Key features and operational details

Describes the key behavior, platform requirements, and fallback conditions for SSH remote public key authentication using TACACS+.

When a user attempts SSH login with public key authentication, the device queries the TACACS+ server for the user's authorized public keys. If the server returns keys, the device compares the returned keys and validates the SSH signature locally.

If fallback is enabled in the AAA method list, the device can check the local `authorized_keys` file when the TACACS+ server is unreachable or when no public key is available for the user. If the TACACS+ server explicitly denies the request, authentication fails and local fallback is not used.

Table 33: Platform and server requirements

Item	Requirement	Description
Platform	Supported platforms	Feature is platform and release dependent
TACACS+ server	SSH public key attribute support	Required for remote authentication and key query handling

Note

- The TACACS+ server must support SSH public key attributes for this feature to work.
- Always validate platform, release, and TACACS+ server compatibility before deployment. Centralized SSH key management enhances security, but fallback configuration must be designed to meet operational requirements.

Support and restrictions of SSH remote public key authentication using TACACS+

Lists platform support, server requirements, configuration restrictions, and compatibility notes for SSH remote public key authentication using TACACS+.

Provides platform compatibility details, server requirements, configuration restrictions, and compatibility notes for SSH remote public key authentication using TACACS+ on devices.

Table 34: Platform and server compatibility

Requirement	Description	Platform or server
TACACS+ server requirements	Server must support SSH public key attributes	• Cisco ISE • tac_plus_ng
Cisco ISE support	Provides SSH public key attribute services as a TACACS+ server	Cisco ISE
Open-source server usage	Validated only for testing and lab environments	• tac_plus • tac_plus_ng
SSH feature configuration	Requires the `ssh server enable remote-pubkey-auth` command	Supported platforms

Table 35: Configuration restrictions and compatibility notes

Restriction	Allowed configuration	Behavior
Feature state	Enable the feature with `ssh server enable remote-pubkey-auth`	Feature is disabled by default
Fallback to local `authorized_keys`	local must be present in the AAA method list	Local fallback is allowed if configured in AAA
Explicit denial from TACACS+ server	No fallback permitted	Access is denied if the TACACS+ server explicitly denies the request
Unreachable TACACS+ servers	Fallback allowed only if AAA configuration permits	Device uses local `authorized_keys` if permitted
SSH public key query caching	No caching is supported	Device always queries the server

Note

- The feature is disabled by default. You must enable it with command `ssh server enable remote-pubkey-auth`.
- Fallback to local `authorized_keys` is permitted only if local is included in the AAA method list.
- If the TACACS+ server explicitly denies access, fallback to local authentication is not allowed.
- If all TACACS+ servers are unreachable, fallback to local Key occurs only if the AAA configuration permits it.
- There is no caching of SSH public key queries. The device queries the configured server for current access control.

How SSH remote public key authentication using TACACS+ servers works

Describes how a device uses TACACS+ servers to perform SSH remote public key authentication, including fallback and authorization handling.

This process begins when a user attempts SSH public key authentication on a device that is configured to use TACACS+ for remote public key lookup. The device retrieves the user's authorized public keys from the TACACS+ server, validates the SSH signature locally, and applies AAA policy to determine whether access is granted.

Summary

The key components involved in the process are:

- SSH client initiates a connection using the user’s private key to generate a signature for authentication.
- CiscoSSHd (SSH server) receives SSH login attempts, manages the authentication workflow, and calls the AAA subsystem.
- The AAA method list determines whether local fallback is allowed.
- The TACACS+ server stores user public keys and processes requests for SSH key-based authentication, returning authorization or fallback instructions as appropriate.

Workflow

The process involves the following stages:

1. SSH login starts
   - The user starts an SSH client, enters a *username*, and initiates login with the public key authentication.
   - The SSH client sends a public key authentication request and signature to the SSH server.
   - The SSH server receives the authentication request, and the session waits for authentication to complete.
   - No access is granted at this stage.
2. Device queries TACACS+
   - The device checks the AAA method list to determine whether TACACS+ remote public key authentication is enabled.
   - If TACACS+ remote public key authentication enabled, the AAA subsystem sends an SSH key query to the TACACS+ server to retrieve the user's authorized public keys.
   - Only the *username* is sent. The user's public key and signature are not sent to the TACACS+ server.
3. TACACS+ returns key information
   - The TACACS+ server receives the query, looks up the user's authorized public keys, and forms a response.
   - If public keys are found, the server returns them to the device.
   - If `no public key` is found, or if the server cannot complete the request, the response can lead to local fallback if the AAA method list allows it.
   - If the server explicitly denies the request, authentication fails and local fallback is not attempted.
4. Device verifies the signature
   - CiscoSSHd compares the user’s provided key and signature against the public keys returned by the TACACS+ server.
   - If a valid match and signature are found, authentication is allowed and the SSH session is established.
   - If there is no match or a "no public key" response, fallback logic may activate based on the AAA method list.
   - If the server explicitly denies access, no fallback occurs and authentication is denied immediately.

   This stage provides both centralized policy enforcement and local cryptographic verification.

   If authentication fails here and fallback is not permitted, the session is terminated.
5. Device checks local keys if fallback is allowed
   - If the AAA method list allows local fallback, the device checks the local `authorized_keys` file when TACACS+ does not provide a usable key result, such as when `no public key` is found or the server is unreachable.
   - If a matching local key is found and the signature is valid, authentication succeeds.
   - If no matching local key is found, authentication fails.
   - If the TACACS+ server explicitly denies access, no fallback occurs and access is denied immediately.

Result

SSH access is granted only if authentication succeeds through TACACS+ or, when permitted, through the local `authorized_keys` file. SSH access is denied if authentication fails or if the TACACS+ server explicitly denies the request.

What's next

To troubleshoot authentication, use `show tacacs counters` and `show tacacs trace` commands. Adjust the AAA method list to manage fallback or enforce stricter authentication policies as required.

Configure SSH remote public key authentication using TACACS+

Configure SSH remote public key authentication by using a TACACS+ server to centrally manage user public keys.

This task configures centralized SSH remote public key authentication on the devices using TACACS+ as the external AAA method.

SSH remote public key authentication with TACACS+ allows the device to retrieve user public keys from an external AAA server during SSH login and validate signatures locally. If configured in the AAA method list, the device can fall back to local authentication when TACACS+ is unavailable.

Before you begin

- Ensure that the device can reach an operational TACACS+ server for authentication requests.
- Verify that user accounts on the TACACS+ server include SSH public key attributes in the supported format.
- Confirm that SSH server is enabled and operational on the device.

Follow these steps to configure SSH remote public key authentication using TACACS+.

Procedure

1. Enter global configuration mode.

   Example

   ```
   configure terminal
   ```

   Use this mode to apply configuration changes to the device. All subsequent commands are entered in configuration mode.

   You are now in global configuration mode.

2. Enable SSH remote public key authentication with TACACS+.

   Example

   ```
   ssh server enable remote-pubkey-auth
   ```

   This command enables the device to retrieve user SSH public keys from a TACACS+ server for SSH login authentication. The device validates SSH signatures locally.

   > **Tip**
   >
   > Only new SSH login attempts will use the remote public key feature.

3. Configure the AAA method list to use TACACS+ for SSH authentication, with local fallback if required.

Example

```
aaa authentication login default group tacacs+ local
```

This line makes TACACS+ the default method for SSH authentication and falls back to local credentials (including local authorized keys) if all TACACS+ servers fail or are unreachable. If `local` is omitted, local fallback is not available.

> **Tip**
>
> You can define custom method lists for specific lines or groups as needed.

AAA method list for SSH authentication is configured with TACACS+ as primary and local fallback as required.

4. Define the TACACS+ server group and add the server entries.

Example

```
tacacs-server host 10.10.10.10
 key 0 <TACACS_SHARED_SECRET>
!
aaa group server tacacs+ TACXRGRP
 server 10.10.10.10
!
```

Configure each TACACS+ server with address and shared secret, then create a group for redundancy. Reference the group in the AAA method list if required. For more information about configuring TACACS+ server groups, see Configure TACACS+ Server Groups.

Repeat the server configuration for each TACACS+ server you use.

5. Commit the configuration to apply changes.

Example

```
commit
```

This saves and activates all changes, including enabling SSH remote public key authentication and configuring AAA and TACACS+ server properties.

- Check configuration for errors before committing.

Configuration is saved, activating remote SSH public key authentication with TACACS+.

New SSH authentication attempts use user public keys retrieved from the configured TACACS+ server, aligning access control and audits with centralized AAA policy. Local key authentication occurs only if fallback is defined in the AAA method list.

8 TACACS+ Services for AAA

Topics:

- Differentiated Services Code Point (DSCP) values for TACACS+ packets
- TACACS+ servers
- TACACS+ server groups
- Per-VRF TACACS+ server groups
- TACACS+ operational statistics
- TACACS+ with TLS protection
- SSH remote public key authentication using TACACS+

Outlines TACACS+ integration for AAA, including DSCP marking, server and server group configuration, per-VRF deployment, operational statistics, and securing TACACS+ with TLS protection to enhance access control, authorization, authentication, and accounting in network environments.

Differentiated Services Code Point (DSCP) values for TACACS+ packets

Describes DSCP values applied to TACACS+ packets, explaining how traffic marking optimizes packet handling and prioritization for TACACS+ communication in managed networks.

A Differentiated Services Code Point (DSCP) marking feature is a Quality of Service traffic-classification feature that sets DSCP values for outgoing TACACS+ packets.

- Differentiated Services divides traffic into classes and forwards packets to the corresponding classes.
- DSCP is a 6-bit field in the Type of Service byte in the IP header.
- For a single connection, the DSCP value is set on the socket while connecting to the server.
- For multiple connections, the DSCP value is set on the available open sockets.
- Use the tacacs-server ipv4 command to set the DSCP value.

TACACS+ servers

Details TACACS+ server concepts, including server parameters, configuration procedures for hosts, group, and AAA integration, and step-by-step tasks for server authorization, authentication, and accounting.

A TACACS+ server is a remote AAA endpoint that

- exchanges authentication, authorization, and accounting (AAA) data with Cisco IOS XR over TCP
- supports configuration of parameters such as port, timeout, shared key, and single-connection options, and
- is subject to global server-count limits and specific behaviors related to source interfaces.

Feature history
Table 36: Feature History Table

Feature Name	Release Information	
TACACS+ Server	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-12G12X4Y-A • 8011-12G12X4Y-D

Additional reference information

- By default, the TACACS+ port is standard port 49 if no other port is specified.
- Timeout and key parameters can be configured globally or on a per-server basis.
- The single-connection option allows all TACACS+ requests to one server to be multiplexed over a single TCP connection.
- Cisco IOS XR supports up to 30 global TACACS+ servers.

TACACS+ server parameters

Provides source TACACS+ server parameter details for port, timeout, shared key, single-connection, idle-timeout, source-interface selection, server limits, and per-server override behavior.

TACACS+ server parameters enable fine-tuned configuration of AAA servers for secure authentication, authorization, and accounting. Use this reference to look up parameter details before configuring individual external servers.

Port

If no port is specified, the system defaults to port 49.

Timeout and key

These parameters can be specified globally for all TACACS+ servers.

- Timeout: Specifies how long the AAA server waits for a response from the TACACS+ server.
- Key: Defines the authentication and encryption key shared between the AAA server and the TACACS+ server.

Single-connection

Multiplexes all TACACS+ requests to the TACACS+ server over a single TCP connection.

Single-connection idle-timeout

Specifies the timeout value for the single TCP connection.

Source-interface selection

Allows selection of the network interface for communication with the TACACS+ server.

Server limits

You can configure a maximum of 30 global TACACS+ servers.

Per-server override behavior

Parameter settings, such as shared keys, connection mode, and source-interface, can be overridden on a per-server basis.

Usage notes

Apply parameter behaviors when choosing per-server overrides, shared keys, connection mode, and source-interface settings. This ensures optimal performance and security in your AAA deployment.

Configure TACACS+ servers

Configure a TACACS+ server host, per-server overrides, source-interface behavior, and optional verification output for external AAA communication with the selected server.

Set up TACACS+ server host entries and parameters for Cisco IOS XR external AAA communication.

Use this task to configure TACACS+ server hosts, customize per-server parameters, set source-interface behavior, and verify configuration for secure external AAA communication.

Before you begin

- Confirm network reachability to the TACACS+ server.
- Ensure the shared authentication key matches your TACACS+ daemon.

Procedure

1. Configure the TACACS+ server host and port.

   Example

   ```
   Router# configure
   Router(config)# tacacs-server host 209.165.200.226 port 51
   Router(config-tacacs-host)#
   ```

   Specifies a TACACS+ host server and optionally specifies a server port number.

   By default, port 49 is used. Valid port numbers range from 1 to 65535.

2. Configure the server-specific TACACS+ timeout.

   Example

   ```
   Router(config-tacacs-host)# tacacs-server host 209.165.200.226 timeout 30
   ```

   Specifies a TACACS+ host server and optionally specifies a timeout value.

   The timeout value sets the length of time the AAA server waits to receive a response from the TACACS+ server.

   This option overrides the global timeout value set with the **tacacs-server timeout** command for only this server.

   The timeout value is expressed as an integer in terms of timeout interval seconds.

   The range is from 1 to 1000.

3. Configure the shared key for the TACACS+ server.

   Example

   ```
   Router(config)# tacacs-server host 209.165.200.226 key 0 a_secret
   ```

   Specifies a TACACS+ host server and optionally specifies an authentication and encryption key shared between the AAA server and the TACACS+ server.

   - TACACS+ packets use this key for encryption.

     This key must match the key used by the TACACS+ daemon.

     Specifying this key overrides the global key set by the **tacacs-server key** command for only this server.

   - (Optional) Entering **0** indicates that an unencrypted (clear-text) key follows.
   - (Optional) Entering 7 indicates that an encrypted key follows.
   - The *auth-key* argument specifies the encrypted or unencrypted key to be shared between the AAA server and the TACACS+ server.

4. Configure single-connection multiplexing for the TACACS+ server.

   Example

   ```
   Router(config)# tacacs-server host 209.165.200.226 single-connection
   ```

   Prompts the router to multiplex all TACACS+ requests to this server over a single TCP connection. By default, a separate connection is used for each session.

5. Configure the TACACS+ single-connection idle timeout.

   Example

   ```
   Router:hostname(config)# tacacs-server host 209.165.200.226
   single-connection-idle-timeout 60
   ```

   Sets the timeout value, in seconds, for the single TCP connection to the TACACS+ server.

   The **single-connection** command creates the single TCP connection.

   For releases before Cisco IOS XR Software Release 7.3.2, the range is 500 to 7200 seconds.

   For Cisco IOS XR Software Release 7.3.2 and later, the range is 5 to 7200 seconds.

6. Configure the TACACS+ source interface.

   Example

   ```
   Router(config)# tacacs source-interface GigabitEthernet 0/4/0/0 vrf abc
   ```

   (Optional) Specifies the source IP address of a selected interface for all outgoing TACACS+ packets.

   - The specified interface or subinterface must have an IP address associated with it.

     If the specified interface or subinterface does not have an IP address or is in the down state, TACACS+ reverts to the default interface.

     Add an IP address to the interface or subinterface, or bring the interface to the up state to prevent the fallback behavior.

   - The **vrf** option specifies the Virtual Private Network (VPN) routing and forwarding (VRF) reference of an AAA TACACS+ server group.

7. Repeat step 2 through step 6 for each external server to be configured.
8. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.
   - End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

9. Display TACACS+ server information.

   Example

   ```
   Router# show tacacs
   ```

(Optional) Displays information about the TACACS+ servers that are configured in the system.

The TACACS+ server host entries are configured. You can verify the configuration and view information about the servers using the **show tacacs** command.

TACACS+ host, server group, and AAA configuration patterns

Provides examples of TACACS+ host, server group, AAA authorization, line-template, and vty-pool configurations. Adapt these examples to suit your deployment requirements.

TACACS+ configuration reference

Host and server configuration

```
! Out-of-band TAC
tacacs-server host 192.0.2.10 port 49
key lm51
tacacs-server host 192.0.2.11 port 49
key lm51
```

Server groups

```
aaa group server tacacs+ tacgrp
 server 192.0.2.10
 server 192.0.2.11

aaa group server tacacs+ eem
 server 192.0.2.10
 server 192.0.2.11
```

AAA authorization and authentication

```
aaa authorization exec tacauthen group tacgrp local
aaa authentication login taclogin group tacgrp local
```

Console line settings

```
line console
 authorization exec tacauthen
 login authentication taclogin
 timeout login response 30
 timestamp
 exec-timeout 0 0
 session-timeout 15
```

VTY pool settings

```
vty-pool default 0 99 line-template console
```

Usage guidelines

- Use these configuration patterns as a compact reference for setting up TACACS+ hosts, server groups, AAA authorization, login authentication, and line/vty-pool settings.

- Adapt addresses, keys, method-list names, and line settings to match your deployment.
- Refer to this reference for quick access to common TACACS+ setup components.

Configure TACACS+ server authorization

Configure remote command authorization through TACACS+, enable ConfD authorization and callbacks, and commit the settings that control command authorization checks.

Configure TACACS+ authorization so that an authenticated user or principal is checked for permission to perform a task.

Configure remote command authorization through TACACS+, enable ConfD authorization and callbacks, and commit the settings that control command authorization checks.

Before you begin

Configure the TACACS+ server and AAA server-group design before enabling remote authorization.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Configure TACACS+ command authorization.

   Example

   ```
   Router(config)# aaa authorization command group tacacs
   ```

   Example

   ```
   Router(config)# aaa authorization command group none
   ```

   Example

   ```
   Router(config)# aaa authorization command group tacacs none
   ```

   Configure the AAA system to perform remote authorization using TACACS+ protocol.

   Configure the AAA system to not perform any authorization.

   Configure the AAA system to first perform TACACS+ authorization and if it fails, no authorization should be performed.

3. Enable remote authorization in ConfD.

   Example

   ```
   Router(config)# confdConfig aaa authorization enabled
   ```

   Configure ConfD to perform remote authorization.

4. Enable authorization callbacks in ConfD.

Example

```
Router(config)# confdConfig aaa authorization callback enabled
```

Configure ConfD to invoke application callbacks for authorization.

5. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- Commit: Saves the configuration changes and remains within the configuration session.
- End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The AAA system and ConfD are configured for TACACS+ authorization behavior.

Configure TACACS+ server authentication

Configure ConfD external authentication, authentication order, and local executable behavior so TACACS+ can verify users or principals through the external authentication path.

Set up external authentication in ConfD, allowing TACACS+ to verify user or principal identities.

Use this task to configure ConfD for external authentication via TACACS+, control the authentication order, and specify the executable for authentication on your Cisco device.

Before you begin

- Ensure TACACS+ server connectivity is established.
- Confirm the local login proxy executable path on the device.

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Enable external authentication in ConfD.

Example

```
Router(config)# confdConfig aaa externalAuthentication enabled
```

Configure ConfD to perform external authentication.

3. Configure the ConfD authentication order.

Example

```
Router(config)# confdConfig aaa authOrder externalAuthentication
localAuthentication
```

Configure the AAA subsystem to perform external authentication first and then local authentication.

4. Configure the external authentication executable in ConfD.

Example

```
Router(config)# confdConfig aaa externalAuthentication executable chvrf 0
/opt/cisco/calvados/bin/calvados_login_aaa_proxy
```

Configure the AAA system to perform external authentication using login executable configured on local host.

5. Commit or discard the configuration changes.

Example

```
commit
end
```

Use one of these options:

- Commit: Saves the configuration changes and remains within the configuration session.
- End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

ConfD is configured to use TACACS+ server authentication according to your specified authentication order and settings.

Configure TACACS+ server accounting

Configure TACACS+ command accounting to log command activity and create an audit trail in Cisco IOS XR.

Enable remote command accounting so TACACS+ records command activity on the router for audit and operational review.

TACACS+ command accounting allows Cisco IOS XR to log both user-generated and system-generated commands, providing a comprehensive record for security audits and troubleshooting.

Before you begin

Ensure the TACACS+ server is configured before enabling command accounting.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Configure TACACS+ command accounting.

   Example

   ```
   Router(config)# aaa accounting command tacacs
   ```

   Configure remote accounting commands.

3. Save or discard the configuration changes.

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.
   - End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

TACACS+ command accounting is configured and committed on the router.

TACACS+ server groups

Explains the organization and behavior of TACACS+ server groups, covering their operational characteristics and providing instructions for configuring server groups within AAA deployments.

A TACACS+ server group is a named list of external TACACS+ server hosts that

- includes one or more server commands
- specifies the hostname or IP address of each external TACACS+ server through individual server commands, and
- supports up to 10 TACACS+ servers per server group and 10 private TACACS+ servers in Cisco IOS XR.

TACACS+ server group behavior

Provides source details about TACACS+ server group membership, AAA method-list references, VRF use, server-count limits, and private-server support for planning deployments.

You can enter one or more server commands to specify the hostname or IP address of an external TACACS+ server. AAA method lists can reference the configured server group for authentication, authorization, or accounting. Separate VRF contexts are supported if needed.

You can configure these maximums:

- 10 TACACS+ servers per server group
- 10 private TACACS+ servers

Use this reference to look up TACACS+ server-group behavior and limits before configuring the server group. After adding the intended external TACACS+ servers, use named TACACS+ server groups in AAA method lists for authentication, authorization, or accounting.

Configure TACACS+ server groups

Configure a named TACACS+ server group and specify external servers for AAA use.

Create a TACACS+ server group to enable AAA method lists to reference a group of external servers for authentication, authorization, or accounting.

A TACACS+ server group allows you to organize multiple external TACACS+ servers into a single, named group. This simplifies AAA method list configuration and enables flexible referencing of external servers. You can also define an optional VRF for the group. After configuration, you can verify server group information as needed.

Before you begin

- Ensure the external TACACS+ server(s) are accessible from the device.
- If configuring the same IP address for both global and VRF-specific configurations, server-private parameters are required.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create the TACACS+ server group.

   Example

   ```
   Router(config)# aaa group server tacacs+ tacgroup1
   ```

   Groups different server hosts into distinct lists and enters the server group configuration mode.

3. Add the TACACS+ server to the server group.

   Example

   ```
   Router(config-sg-tacacs+)# server 192.168.100.0
   ```

   Specifies the hostname or IP address of an external TACACS+ server.

   AAA method lists can reference the configured server group for authentication, authorization, or accounting.

4. Configure the VRF reference for the TACACS+ server group.

Example

```
Router(config-sg-tacacs+)# vrf vrf-id
```

The vrf option specifies the Virtual Private Network (VPN) routing and forwarding (VRF) reference of an AAA TACACS+ server group.

5. Repeat step 3 for every external server to be added to the server group named in step 2.
6. Configure the VRF reference for the TACACS+ server group.

   The vrf option specifies the Virtual Private Network (VPN) routing and forwarding (VRF) reference of an AAA TACACS+ server group.

7. Commit or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.
   - End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

8. Display TACACS+ server group information.

   Example

   ```
   Router# show tacacs server-groups
   ```

   (Optional) Displays information about each TACACS+ server group that is configured in the system.

The TACACS+ server group is configured and available for use in AAA method lists.

Per-VRF TACACS+ server groups

Outlines per-VRF TACACS+ server group functionality, highlighting specific behavior, prerequisites, and task-based procedures for configuring server groups within Virtual Routing and Forwarding (VRF) environments.

A per-VRF TACACS+ server group is a TACACS+ server-group configuration that

- associates private TACACS+ server entries with a VRF instance for AAA reachability
- uses server-private and VRF commands for feature configuration, and
- allows server status, statistics, and keys to differ by VRF context.

Additional reference information

Global server definitions can be referenced by multiple server groups, but all references use the same server instance.

Behavior and prerequisites for TACACS+ server groups in per-VRF configurations

Provides source behavior and prerequisites for configuring TACACS+ server groups with VRF-specific private server entries, global-server reuse, and migration planning.

Cisco IOS XR software supports per-VRF AAA for TACACS+ server groups, enabling flexible configuration scenarios for both private and global server definitions. You can use the **server-private** and **vrf** commands to configure this feature.

Key behaviors and features

- You can refer to global server definitions from multiple server groups. All references use the same server instance and connect to the same server.
- For VRF configurations, global server configuration is not strictly required, because server status, statistics, and keys can differ across VRFs.
- Use private server configuration when each VRF requires the same TACACS+ server to have different status, statistics, or keys.
- If the same server is used in different groups with different VRFs, ensure that the server is reachable from all those VRFs.
- When migrating servers to a VRF, it is safe to remove the global server configuration for that server, since per-VRF configuration takes precedence.

Prerequisites

Before configuring per-VRF TACACS+ server groups, verify these requirements:

- Be familiar with configuring TACACS+, AAA, per VRF AAA, and group servers.
- Ensure you have access to the TACACS+ server itself.
- Configure the VRF instance prior to setting up the specific VRF for a TACACS+ server and ensure the VRF is reachable.

Usage considerations

- Global server definitions can be reused across server groups unless distinct status or authentication keys are required per VRF.
- Private server configuration is recommended when VRF-specific differences are essential.
- Plan for migration to VRF-specific configuration by ensuring server reachability and removing obsolete global server settings when no longer needed.

Configure per-VRF TACACS+ server groups

Configure private TACACS+ servers and a VRF reference in a TACACS+ server group, then review and verify the per-VRF server-group state.

Configure a TACACS+ server group so that private TACACS+ servers are reached through a specific VRF.

Use this task when TACACS+ server status, statistics, or keys need to differ by VRF and you want to define separate private TACACS+ server entries associated with specific VRFs.

Before you begin

Before you configure per-VRF TACACS+ server groups, verify these requirements

- Be familiar with configuring TACACS+, AAA, per VRF AAA, and group servers.
- Ensure that you have access to the TACACS+ server.
- Configure the VRF instance before configuring the specific VRF for a TACACS+ server and ensure that the VRF is reachable.

Procedure

1. Configure the per-VRF TACACS+ server group.

   Example

   ```
   Router# configure
   Router(config)# aaa group server tacacs+ tacgroup1
   Router(config-sg-tacacs+)# server-private 10.1.1.1 port 49 key a_secret
   Router(config-sg-tacacs+)# vrf test-vrf
   Router(config-sg-tacacs+)# commit
   ```

   Cisco IOS XR software supports per-VRF AAA on TACACS+ server groups.

   Use the **server-private** and **vrf** commands to configure this feature.

   You can refer to global server definitions from multiple server groups.

   All references use the same server instance and connect to the same server.

   For VRF, you do not need the global configuration because the server status, server statistics and the key could be different for different VRFs.

   Use the server-private configuration to configure per-VRF TACACS+ server groups.

   If the same server is used in different groups with different VRFs, ensure that the server is reachable through all those VRFs.

   If you are migrating the servers to a VRF, it is safe to remove the global server configuration for that server.

   The configuration groups different server hosts into distinct lists and enters server-group configuration mode.

   You can enter one or more server commands.

   The server command specifies the hostname or IP address of an external TACACS+ server.

   AAA method lists can reference the configured server group for authentication, authorization, or accounting.

   The server-private command configures the IP address and secret key of the private TACACS+ server that is reachable through a specific VRF.

   You can configure multiple private server entries that are reachable through the same VRF.

   The vrf option specifies the VRF reference for a AAA TACACS+ server group.

2. Review the per-VRF TACACS+ running configuration.

   Example

   ```
   aaa group server tacacs+ tacgroup1
    vrf test-vrf
    server-private 10.1.1.1 port 49
     key 7 0822455D0A16
    !
   ```

```
 server-private 10.1.1.2 port 49
  key 7 05080F1C2243
 !
 server-private 2001:db8:1::1 port 49
  key 7 045802150C2E
 !
 server-private 2001:db8:1::2 port 49
  key 7 13061E010803
 !
!
```

3. Verify per-VRF TACACS+ server-group information.

Example

```
Router# show tacacs
Fri Sep 27 11:14:34.991 UTC

Server: 10.1.1.1/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4

Server: 10.1.1.2/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv4

Server: 2001:db8:1::1/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv6

Server: 2001:db8:1::2/49 vrf=test-vrf [private]
        opens=0 closes=0 aborts=0 errors=0
        packets in=0 packets out=0
        status=up single-connect=false family=IPv6
```

The TACACS+ server group uses private TACACS+ server entries and the configured VRF for server reachability.

TACACS+ operational statistics

Presents TACACS+ operational data collection, including command outputs used to monitor and verify TACACS+ service status and performance on the network.

A TACACS+ operational statistics feature is a monitoring capability that

- displays TACACS+ transaction counters, server details, source interfaces, and counter-clear operations on a router
- shows TCP connection statistics, AAA packet counts, and TACACS+ transaction outcomes, and
- provides information to monitor TACACS+ health and assists in identifying and debugging TACACS+ transaction failures.

Feature history

The feature history table lists release support for this feature.

Table 37: Feature History Table

Feature Name	Release Information	Feature Description
View TACACS+ information in Router	Release 7.5.4	With this feature, you can view TCP connection statistics like failures, timeout, and disconnect in connections, number of AAA packets received from an external server or sent to an external server, and so on during TACACS+ transactions. This information helps you monitor TACACS+ health in the routers. It is also helpful in identifying and debugging TACACS+ transaction failures if any. This feature introduces the following commands: • **show tacacs counters** • **show tacacs details** • **show tacacs source-interface** • **clear tacacs counters**

TACACS+ operational command output

Provides source examples for TACACS+ counters, TACACS+ details, source-interface display, and counter-clear verification output used to monitor router transactions during AAA operations.

This reference section includes examples of various TACACS+ operational command outputs used to monitor and troubleshoot router transactions during AAA operations. Use these examples to identify outputs that report TACACS+ request counts, failures, errors, timeouts, server state, and source-interface details. The outputs can help verify operational health and confirm that the counters reset as expected after using clear commands.

TACACS+ counters output

The TACACS+ counters output records the number of requests, timeouts, failures, errors, and successes for each TACACS+ server across all AAA services.

```
plaintext:Y
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]

 Authentication:
  10 requests, 4 accepts, 3 failure, 2 error, 1 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  6 requests, 6 accepts, 0 denied, 0 error, 0 timeout

 Exec Accounting:
```

```
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  6 requests, 6 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

 Authentication:
  0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Exec Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout
```

TACACS+ details output

The TACACS+ details output includes server group, source interface, individual server statistics, packet counters, connection opens and closes, and TCP connection indicators.

```
plaintext:Y
Router:ios# show tacacs details

TACACS+ Server                                                  :
10.105.236.101/4010 [Global]
    Family                                                      : IPv4
    Timeout(in secs)                                            : 3
    Connection Opens                                            : 8
    Connection Closes                                           : 8
    Requests sent                                               : 6
    Response received                                           : 6
    Packets Abort                                               : 2
    Server State                                                : Down
    Server On-Hold                                              : True
    Tacacs-Single-Connect                                       : False
    Tacacs-Single-Connect-Idle-Timeout(in secs)                 : 0
    Last Connection Attempted                                   : 08:32:43
 UTC Tue Aug 02 2022


TACACS+ Server                                                  :
10.105.236.101/8010 [Private] vrf=default
    Family                                                      : IPv4
    Timeout(in secs)                                            : 3
    Connection Opens                                            : 8
    Connection Closes                                           : 7
    Requests sent                                               : 7
    Response received                                           : 7
    Packets Abort                                               : 0
    Server State                                                : Up
    Server On-Hold                                              : False
    Tacacs-Single-Connect                                       : False
    Tacacs-Single-Connect-Idle-Timeout(in secs)                 : 0
    Last Connection Attempted                                   : 08:32:52
```

```
 UTC Tue Aug 02 2022


TACACS+ Server-groups:

Global list of servers
    Server 10.105.236.101/4010 family=IPv4
Server group 'tac1' has 1 servers
    Servers in this group are under 'default' vrf
    Server 10.105.236.101/8010 [private] family=IPv4

TACACS+ Source-Interface:

Interface                          VRF Id                          IPV4-Address
GigabitEthernet0/0/0/0            0x60000001                          0.0.0.0
MgmtEth0/RP0/CPU0/0               0x60000000                  192.168.122.222

Interface                           VRF Id                          IPV6-Address
GigabitEthernet0/0/0/0             0x60000001                                ::
MgmtEth0/RP0/CPU0/0                0x60000000                                ::
```

TACACS+ source-interface output

The TACACS+ source-interface output displays the source interface and corresponding IP addresses for TACACS+ transactions.

```
plaintext:Y
Router:ios# show tacacs source-interfaces

Interface                                              VRF Id
       IPV4-Address

MgmtEth0/RP0/CPU0/0                                   0x60000000
      192.168.122.222

Interface                                               VRF Id
       IPV6-Address

MgmtEth0/RP0/CPU0/0                                   0x60000000
             ::
```

Clear TACACS+ counters output

To clear all AAA services counters reported by the `show tacacs counters` command for all TACACS+ servers, use the **clear tacacs counters** command:

```
plaintext:Y
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]

 Authentication:
  10 requests, 4 accepts, 3 failure, 2 error, 1 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  6 requests, 6 accepts, 0 denied, 0 error, 0 timeout
```

```
 Exec Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  6 requests, 6 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

 Authentication:
  0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Exec Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

Router:ios# clear tacacs counters
Router:ios# show tacacs counters

TACACS+ Server:   10.105.236.101/4010 [global]

 Authentication:
  0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Exec Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

TACACS+ Server:   10.105.236.101/2201 [private] vrf = default

 Authentication:
  0 requests, 0 accepts, 0 failure, 0 error, 0 timeout

 Exec Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Command Authorization:
  0 requests, 0 accepts, 0 denied, 0 error, 0 timeout

 Exec Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout

 Command Accounting:
  0 requests, 0 accepts, 0 fail, 0 error, 0 timeout
```

How to use these outputs

- Use the counters and details outputs to monitor TACACS+ service health and validate router interaction.
- Check after clearing counters to confirm reset success and verify current request and error activity.

TACACS+ with TLS protection

Explains securing TACACS+ with TLS protection by detailing security benefits, operational workflows, supported and unsupported requirements, and configuration steps for implementing TLS in TACACS+ deployments.

A TACACS+ with TLS protection is a security enhancement that

- encrypts TACACS+ AAA communication between network devices and TACACS+ servers using Transport Layer Security (TLS)
- provides confidentiality and integrity for sensitive AAA data transmitted over potentially insecure networks, and
- supports mutual authentication between the client and server through TLS X.509 certificates and is compatible with TLS versions 1.3 and 1.2.

Feature history

The feature history table lists release support for this feature.

Table 38: Feature History Table

Feature Name	Release Information	Feature Description
TACACS+ with TLS protection	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can significantly enhance security and reduce the risk of attacks on weak encryption by using TACACS+ over TLS. This method ensures the secure transmission of all Authentication, Authorization, and Accounting (AAA) data between the client and server. It provides robust protection for sensitive environments by supporting mutual authentication through a TLS X.509 certificate-based infrastructure. This feature is compatible with both TLS versions 1.3 and 1.2.

Security benefits provided by TACACS+ with TLS protection

Lists the security benefits of TACACS+ with TLS protection, including encrypted TACACS+ communication, protected AAA data, mutual authentication, and TLS standards compliance.

The security benefits of TACACS+ with TLS protection include:

- Encrypted communication: Enhances security by encrypting all TACACS+ traffic between clients and servers.
- Data protection: Protects AAA data with robust encryption, reducing risk of interception or tampering.
- Mutual authentication: Supports mutual authentication between client and server, preventing unauthorized access and impersonation.
- TLS standards compliance: Complies with TLS 1.3 and 1.2 standards to ensure strong, modern cryptographic protocols.

Use this reference to identify the security benefits that TACACS+ with TLS protection provides. Apply these benefits when selecting TACACS+ transport protection for sensitive AAA environments.

How TACACS+ with TLS protection works

Describes how TACACS+ with TLS protection secures AAA communications by establishing an encrypted TLS session during SSH access and exchanging authentication, authorization, and accounting packets between network devices and servers.

The use of TLS protection for TACACS+ is critical in environments where sensitive AAA information must be secured from potential interception or tampering. By wrapping TACACS+ packets in an encrypted TLS tunnel, organizations safeguard both user credentials and authorization events during SSH remote access.

Summary

The key components involved in the process are:

- End user: Initiates an SSH session to access the network device.
- Router as TACACS+ TLS client: Receives the AAA service request and initiates a TLS session with the TACACS+ server.
- TACACS+ server: Uses valid TLS configuration, terminates the TLS session, and processes AAA requests and responses.
- TLS session: Encrypts the TACACS+ traffic between the client and server.
- TACACS+ packets: Carry authentication, authorization, and accounting information over the TLS tunnel.

TACACS+ with TLS protection enhances network security by establishing an encrypted TLS session between a router and a TACACS+ server whenever a user initiates SSH access. This ensures all AAA traffic is protected during communication.

Workflow

These stages describe how TACACS+ with TLS protection works.

1. Session initiation: The end user initiates an SSH session to the network device.
2. AAA request: The router receives the user's AAA service request for TACACS+ with TLS protection.
3. TLS establishment: If the TACACS+ server has valid TLS settings, the router and server establish a TLS session.
4. Secure exchange: The router and TACACS+ server exchange TACACS+ packets over the encrypted TLS session.

Result

The process establishes a secure, encrypted channel for TACACS+ traffic, ensuring AAA communications remain protected during SSH access.

Requirement: You must use supported TACACS+ TLS practices

Outlines mandatory requirements for TACACS+ deployments with TLS protection on routers and servers.

You must follow these requirements when configuring TACACS+ with TLS protection:

- Configure the destination port for TACACS+ with TLS protection; do not use separate ports for authentication, accounting, or authorization.
- Use TLS X.509 certificate-based mutual authentication between client and server.
- Use only TLS version 1.3 (as mandated by RFC 8446) or TLS version 1.2.
- Choose either multi-connect or single connect modes without TLS resumption, as required by your environment.
- Ensure you implement the cipher suites mandated by TLS 1.3 and TLS 1.2.
- Use IPv4 or IPv6 for TACACS+ transactions, based on your network configuration.
- Configure the source interface and non-default Virtual Routing and Forwarding (VRF), if required by your topology.
- Set the connection timer and single-connect idle timeout as needed; these settings work the same as for TACACS+ over TCP.

Requirement: Avoid unsupported TACACS+ TLS behavior

Outlines mandatory restrictions for configuring TACACS+ with TLS protection in Cisco IOS XR. These include avoiding legacy TACACS+ encryption methods, mixing TLS and non-TLS servers in server groups, and using TLS session resumption or PSK cipher suites, which are not supported in current releases. Review and apply these requirements to prevent misconfigurations and ensure secure TACACS+ deployment.

Cisco IOS XR imposes strict requirements concerning unsupported behaviors when configuring TACACS+ with TLS protection. You must comply with the following:

- Do not use the TACACS+ encryption method supported in Cisco IOS XR software releases earlier than 25.3.1–it is unsupported.
- Never mix non-TLS and TLS servers in the same TACACS+ server group.
- Do not enable TLS session resumption or use TLS sessions with PSK cipher suites; these are unsupported.

Configure TACACS+ with TLS protection

Configure TLS for TACACS+ server communication directly on a TACACS+ host or within an AAA server group, then verify the TLS state.

Enable TLS for TACACS+ server communication so that authentication, authorization, and accounting traffic is encrypted.

Use TLS to secure TACACS+ server communication, either directly on a TACACS+ host or within an AAA server group. Afterwards, verify the TLS protection state.

Before you begin

Configure the TACACS+ server and trustpoint information before enabling TLS.

Procedure

1. Enable TLS for TACACS+ server communication.

   TACACS+ server host configuration:

Example

```
Router(config)# tacacs-server host 10.105.236.101 port 4950
Router(config-tacacs-host)# tls
Router(config-tacacs-host-tls)# server-name-indicator aaa.cisco.com
Router(config-tacacs-host-tls)# trustpoint test
Router(config-tacacs-host-tls)# commit
```

Server-group configuration:

Example

```
Router(config)# configure
Router(config)# aaa group server tacacs+ tac1
Router(config-sg-tacacs)# server-private 10.105.236.101 port 2345
Router(config-sg-tacacs-private)# tls
Router(config-sg-tacacs-private-tls)# server-name-indicator aaa.cisco.com
Router(config-sg-tacacs-private-tls)# trustpoint abc
Router(config-sg-tacacs-private-tls)# commit
```

- TACACS+ server host configuration:
- Server-group configuration:

2. Display TACACS+ TLS operational information.

Example

```
Router# show tacacs
Info: Verify that TACACS+ with TLS protection is configured.

Server: 10.105.236.101/2084
.....
FIPS mode : TRUE/FALSE.
TLS:
   Version: TLS 1.3/1.2      // only for active connection
   Cipher: TLS_AES_128_GCM_SHA256  // only for active connection
Statistics:
   Successful connections: 0
   Failed connections         : 0
   SSL  errors :
    Connect error:
    Read error:
    Write error:
    Handshake Failure:
    Protocol Mismatch :
  Certificate Validation Error:
  Cipher Suite Mismatch:
  Session Timeout:
  Revoked Certificate:
  Unsupported TLS Version:
  Untrusted CA:
```

TLS secures TACACS+ communication with the specified server.

SSH remote public key authentication using TACACS+

Describes SSH remote public key authentication with TACACS+ integration and explains centralized SSH key management and secure access benefits.

A SSH remote public key authentications using TACACS+ is a feature that

- enables centralized retrieval and validation of SSH public keys through TACACS+ AAA servers,
- supports fallback to `authorized_keys` files based on configuration settings, and
- provides real-time access control and unified auditing across managed network devices.

Table 39: Feature History Table

Feature Name	Release Information	Feature Description
SSH remote public key authentication using TACACS+	Release 26.3.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100], 8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100]); Modular Systems (8800 [LC ASIC: P100]) (select variants only*). You can centrally manage SSH public keys on IOS XR by enabling devices to retrieve and validate user public keys from TACACS+ servers at login. This feature simplifies key management, streamlines access control, and supports fallback to local keys for continued access if the server is unavailable.

Key features and operational details

Describes the key behavior, platform requirements, and fallback conditions for SSH remote public key authentication using TACACS+.

When a user attempts SSH login with public key authentication, the device queries the TACACS+ server for the user's authorized public keys. If the server returns keys, the device compares the returned keys and validates the SSH signature locally.

If fallback is enabled in the AAA method list, the device can check the local `authorized_keys` file when the TACACS+ server is unreachable or when no public key is available for the user. If the TACACS+ server explicitly denies the request, authentication fails and local fallback is not used.

Table 40: Platform and server requirements

Item	Requirement	Description
Platform	Supported platforms	Feature is platform and release dependent
TACACS+ server	SSH public key attribute support	Required for remote authentication and key query handling

Note

- The TACACS+ server must support SSH public key attributes for this feature to work.
- Always validate platform, release, and TACACS+ server compatibility before deployment. Centralized SSH key management enhances security, but fallback configuration must be designed to meet operational requirements.

Support and restrictions of SSH remote public key authentication using TACACS+

Lists platform support, server requirements, configuration restrictions, and compatibility notes for SSH remote public key authentication using TACACS+.

Provides platform compatibility details, server requirements, configuration restrictions, and compatibility notes for SSH remote public key authentication using TACACS+ on devices.

Table 41: Platform and server compatibility

Requirement	Description	Platform or server
TACACS+ server requirements	Server must support SSH public key attributes	• Cisco ISE • tac_plus_ng
Cisco ISE support	Provides SSH public key attribute services as a TACACS+ server	Cisco ISE
Open-source server usage	Validated only for testing and lab environments	• tac_plus • tac_plus_ng
SSH feature configuration	Requires the `ssh server enable remote-pubkey-auth` command	Supported platforms

Table 42: Configuration restrictions and compatibility notes

Restriction	Allowed configuration	Behavior
Feature state	Enable the feature with `ssh server enable remote-pubkey-auth`	Feature is disabled by default
Fallback to local `authorized_keys`	local must be present in the AAA method list	Local fallback is allowed if configured in AAA
Explicit denial from TACACS+ server	No fallback permitted	Access is denied if the TACACS+ server explicitly denies the request
Unreachable TACACS+ servers	Fallback allowed only if AAA configuration permits	Device uses local `authorized_keys` if permitted
SSH public key query caching	No caching is supported	Device always queries the server

Note

- The feature is disabled by default. You must enable it with command `ssh server enable remote-pubkey-auth`.
- Fallback to local `authorized_keys` is permitted only if local is included in the AAA method list.
- If the TACACS+ server explicitly denies access, fallback to local authentication is not allowed.
- If all TACACS+ servers are unreachable, fallback to local Key occurs only if the AAA configuration permits it.
- There is no caching of SSH public key queries. The device queries the configured server for current access control.

How SSH remote public key authentication using TACACS+ servers works

Describes how a device uses TACACS+ servers to perform SSH remote public key authentication, including fallback and authorization handling.

This process begins when a user attempts SSH public key authentication on a device that is configured to use TACACS+ for remote public key lookup. The device retrieves the user's authorized public keys from the TACACS+ server, validates the SSH signature locally, and applies AAA policy to determine whether access is granted.

Summary

The key components involved in the process are:

- SSH client initiates a connection using the user’s private key to generate a signature for authentication.
- CiscoSSHd (SSH server) receives SSH login attempts, manages the authentication workflow, and calls the AAA subsystem.
- The AAA method list determines whether local fallback is allowed.
- The TACACS+ server stores user public keys and processes requests for SSH key-based authentication, returning authorization or fallback instructions as appropriate.

Workflow

The process involves the following stages:

1. SSH login starts
   - The user starts an SSH client, enters a *username*, and initiates login with the public key authentication.
   - The SSH client sends a public key authentication request and signature to the SSH server.
   - The SSH server receives the authentication request, and the session waits for authentication to complete.
   - No access is granted at this stage.
2. Device queries TACACS+
   - The device checks the AAA method list to determine whether TACACS+ remote public key authentication is enabled.
   - If TACACS+ remote public key authentication enabled, the AAA subsystem sends an SSH key query to the TACACS+ server to retrieve the user's authorized public keys.
   - Only the *username* is sent. The user's public key and signature are not sent to the TACACS+ server.

3. TACACS+ returns key information
   - The TACACS+ server receives the query, looks up the user's authorized public keys, and forms a response.
   - If public keys are found, the server returns them to the device.
   - If `no public key` is found, or if the server cannot complete the request, the response can lead to local fallback if the AAA method list allows it.
   - If the server explicitly denies the request, authentication fails and local fallback is not attempted.
4. Device verifies the signature
   - CiscoSSHd compares the user's provided key and signature against the public keys returned by the TACACS+ server.
   - If a valid match and signature are found, authentication is allowed and the SSH session is established.
   - If there is no match or a "no public key" response, fallback logic may activate based on the AAA method list.
   - If the server explicitly denies access, no fallback occurs and authentication is denied immediately.

   This stage provides both centralized policy enforcement and local cryptographic verification.

   If authentication fails here and fallback is not permitted, the session is terminated.
5. Device checks local keys if fallback is allowed
   - If the AAA method list allows local fallback, the device checks the local `authorized_keys` file when TACACS+ does not provide a usable key result, such as when `no public key` is found or the server is unreachable.
   - If a matching local key is found and the signature is valid, authentication succeeds.
   - If no matching local key is found, authentication fails.
   - If the TACACS+ server explicitly denies access, no fallback occurs and access is denied immediately.

Result

SSH access is granted only if authentication succeeds through TACACS+ or, when permitted, through the local `authorized_keys` file. SSH access is denied if authentication fails or if the TACACS+ server explicitly denies the request.

What's next

To troubleshoot authentication, use `show tacacs counters` and `show tacacs trace` commands. Adjust the AAA method list to manage fallback or enforce stricter authentication policies as required.

Configure SSH remote public key authentication using TACACS+

Configure SSH remote public key authentication by using a TACACS+ server to centrally manage user public keys.

This task configures centralized SSH remote public key authentication on the devices using TACACS+ as the external AAA method.

SSH remote public key authentication with TACACS+ allows the device to retrieve user public keys from an external AAA server during SSH login and validate signatures locally. If configured in the AAA method list, the device can fall back to local authentication when TACACS+ is unavailable.

Before you begin

- Ensure that the device can reach an operational TACACS+ server for authentication requests.
- Verify that user accounts on the TACACS+ server include SSH public key attributes in the supported format.

- Confirm that SSH server is enabled and operational on the device.

Follow these steps to configure SSH remote public key authentication using TACACS+.

Procedure

1. Enter global configuration mode.

   Example

   ```
   configure terminal
   ```

   Use this mode to apply configuration changes to the device. All subsequent commands are entered in configuration mode.

   You are now in global configuration mode.

2. Enable SSH remote public key authentication with TACACS+.

   Example

   ```
   ssh server enable remote-pubkey-auth
   ```

   This command enables the device to retrieve user SSH public keys from a TACACS+ server for SSH login authentication. The device validates SSH signatures locally.

   > **Tip**
   >
   > Only new SSH login attempts will use the remote public key feature.

3. Configure the AAA method list to use TACACS+ for SSH authentication, with local fallback if required.

   Example

   ```
   aaa authentication login default group tacacs+ local
   ```

   This line makes TACACS+ the default method for SSH authentication and falls back to local credentials (including local authorized keys) if all TACACS+ servers fail or are unreachable. If `local` is omitted, local fallback is not available.

   > **Tip**
   >
   > You can define custom method lists for specific lines or groups as needed.

   AAA method list for SSH authentication is configured with TACACS+ as primary and local fallback as required.

4. Define the TACACS+ server group and add the server entries.

Example

```
tacacs-server host 10.10.10.10
 key 0 <TACACS_SHARED_SECRET>
!
aaa group server tacacs+ TACXRGRP
 server 10.10.10.10
!
```

Configure each TACACS+ server with address and shared secret, then create a group for redundancy. Reference the group in the AAA method list if required. For more information about configuring TACACS+ server groups, see Configure TACACS+ server groups on page 196.

Repeat the server configuration for each TACACS+ server you use.

5. Commit the configuration to apply changes.

Example

```
commit
```

This saves and activates all changes, including enabling SSH remote public key authentication and configuring AAA and TACACS+ server properties.

- Check configuration for errors before committing.

Configuration is saved, activating remote SSH public key authentication with TACACS+.

New SSH authentication attempts use user public keys retrieved from the configured TACACS+ server, aligning access control and audits with centralized AAA policy. Local key authentication occurs only if fallback is defined in the AAA method list.

9 CA Interoperability and Certificate Enrollment

Topics:

- Certification authority interoperability
- Certification authority interoperability facts

Explains CA interoperability concepts and certificate enrollment processes, covering configuration requirements, enrollment methods, and operational workflows for managing digital certificates across integrated systems.

Certification authority interoperability

Describes how CA interoperability enables communication between certification authorities, outlining configuration steps, supported protocols, and trust establishment to facilitate secure certificate management in multi-CA environments.

A certification authority interoperability capability is a security feature that

- enables Cisco IOS XR devices to use digital certificates for secure communication
- supports certificate-based trust across IPSec, SSL, and SSH protocols, and
- allows devices and certification authorities (CAs) to interact for efficient certificate management and scalable security.

Certification authority (CA) interoperability allows devices and CAs to communicate, permitting retrieval and use of digital certificates in network deployments. While IPSec can operate without a CA, using a CA improves manageability and scalability for IPSec.

> **Note**
>
> IPSec is not currently supported.

Requirement: You must meet certification authority prerequisites

Outlines the prerequisites you must meet to ensure certification authority interoperability and supported deployment.

To implement certificate authority (CA) interoperability successfully, you must meet the following requirements:

- You must belong to a user group associated with a task group that includes the required task IDs. Command reference guides list the task IDs for each command. If your user group assignment prevents you from using a command, contact your AAA administrator for assistance.
- You need to have a CA available to your network before you configure this interoperability feature. The CA must support Cisco Systems PKI protocol and the Simple Certificate Enrollment Protocol (SCEP), formerly called Certificate Enrollment Protocol (CEP).

How CA interoperability works

Describes how CA interoperability works in Cisco IOS XR, outlining the participants, key stages, and outcome of the certificate authority workflow.

CA interoperability is essential for establishing secure identities and encrypted communications between network devices and external systems. By integrating with certificate authorities, Cisco IOS XR devices can obtain, validate, and manage certificates required for security protocols such as IPsec and SSL. Administrators must follow a defined process to ensure each device is properly authenticated and authorized to participate in secure networking.

Summary

The key components involved in the process are:

- Router: Initiates CA operations and interacts with certificate authorities to request and manage certificates.
- Certificate Authority (CA): Issues and manages digital certificates, validating requests and ensuring authenticity.
- Administrator: Configures device settings for CA interactions, including certificate requests and trustpoint configuration.

Certificate authority (CA) interoperability in Cisco IOS XR allows network devices to manage secure communications by interacting with external CAs to issue, authenticate, and manage digital certificates.

Workflow

These stages describe how CA interoperability works.

1. Configure the router hostname and IP domain name: The administrator sets the router identity required for certificate requests.
2. Generate RSA key pairs: The router creates cryptographic keys for secure certificate transactions.
3. Import public key to the router: The administrator loads the public key from the CA onto the router.
4. Declare certification authority and configure trustpoint: The administrator defines the CA as a trusted source.
5. Authenticate the CA: The router validates the CA credentials.
6. Request device certificates: The router sends a certificate signing request (CSR) to the CA to obtain its own certificate.
7. Configure certificate enrollment using cut-and-paste: The administrator can manually enroll certificates if automated enrollment is unavailable.
8. Save or discard configuration changes:
   - The administrator can:
     - Save and commit changes to exit the configuration session.
     - Exit without saving changes.
     - Cancel and remain in the session without committing changes.
9. Digitally sign IKE key management messages: The router uses issued certificates to authenticate and ensure the integrity of IKE communications.

Result

The CA interoperability process enables secure certificate management and authentication in Cisco IOS XR, ensuring trusted communication across network devices.

Configure the router hostname and IP domain name

Configure the router hostname and IP domain name to enable certificate authority interoperability for IPSec.

Assign a hostname and IP domain name to the router, enabling it to generate a fully qualified domain name (FQDN) for keys and certificates used in IPSec.

Routers require both a hostname and an IP domain name to generate FQDNs for IPSec keys and certificates. For example, a certificate named *router20.example.com* is created when the router's hostname is set to *router20* and its IP domain name is set to *example.com*.

Before you begin

Review certificate authority prerequisites.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

Enters global configuration mode.

2. Configure the router hostname.

   Example

   ```
   Router(config)# hostname myhost
   ```

   Configures the hostname of the router.

3. Configure the IP domain name.

   Example

   ```
   Router(config)# domain name mydomain.com
   ```

   Configures the IP domain name of the router.

4. Save or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.
   - End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

The router's hostname and IP domain name are configured. You can verify successful configuration by checking the output against the provided examples and confirming intended certificate authority interoperability behavior.

RSA key pairs

Explains RSA key pairs for Cisco IOS XR certificate-based trust, focusing on their role, constraints, and operational details for certification authority interoperability.

A set of RSA key pairs is a certificate authority interoperability component that

- enables both digital signing and encryption, securing IKE key management messages
- acts as a required prerequisite for obtaining a digital certificate for the router, and
- supports compliance with current cryptographic security standards by enforcing minimum key sizes and removal of weak or deprecated keys.

RSA key pairs play a crucial role in securing IKE key management messages by enabling digital signing and encryption. Their use is required before a certificate can be obtained for your router.

Specifically, RSA key pairs:

- digitally sign IKE key management messages, ensuring their authenticity and integrity,
- encrypt IKE key management messages, protecting sensitive information from unauthorized access, and
- satisfy prerequisites for obtaining a digital certificate for the router, which further enhances security in network communications.

Feature history

Table 43: Feature History Table

Feature Name	Release Information	Feature Description
Syslog warnings for RSA keys and DSA keys	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This enhancement ensures Cisco IOS XR devices remain compliant with evolving security standards. During system reboot, the device now sends syslog warnings if weak SSH host keys are detected–specifically, RSA keys less than 3072 bits or any DSA keys. Additionally, the default RSA key size has been increased from 2048 to 3072 bits to further strengthen security. This feature introduces these changes: CLI: . • The **crypto key generate rsa** command has been modified. • The **crypto key generate dsa** command has been modified. • The **show crypto key mypubkey rsa** command has been modified.
RSA and DSA Keys Available in Running Configuration	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
RSA and DSA Keys Available in Running Configuration	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
RSA and DSA Keys Available in Running Configuration	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
RSA and DSA Keys Available in Running Configuration	Release 7.3.4	You can now view the RSA and DSA keys in the running configuration by using the **show running-configuration** command.

Syslog warnings for RSA keys and DSA key

Starting with Cisco IOS XR Release 26.1.1, the default RSA key size is 3072 bits. The system triggers a syslog warning during boot if a SSH host RSA key is less than 3072 bits. DSA keys are no longer auto-generated at boot, and their presence prompts a syslog warning for removal. These changes ensure compliance with cryptographic security standards and bolster device protection against new threats.

Configure RSA key pairs

Configure RSA key pairs to support certificate authority interoperability on Cisco routers.

Enable secure communications by generating, labeling, and verifying RSA key pairs as required by certificate authority interoperability.

RSA key pairs are used for authentication and encryption. Cisco routers often auto-generate RSA keys during boot-up. However, manual configuration is required when specific key pairs are needed or existing keys must be changed or deleted. RSA key details are visible in the running configuration.

Before you begin

- Review certificate authority prerequisites relevant to your environment.
- Confirm you understand the implications of deleting or regenerating keys, especially if keys have been compromised.

Procedure

1. Renerate an RSA key pair.

   This step is requied only iff the RSA key pair is missing after router boot-up.

   Example

   ```
   Router# crypto key generate rsa general-keys
   ```

   From Cisco IOS XR Release 7.3.2 onwards, you can configure this command from XR Config mode.

To delete the RSA keys, use the no form: no crypto key generate rsa

2. Delete a specific RSA key pair or all RSA keys.

   For example, if key compromise is suspected or the keys should no longer be used).

   Example

```
Router# crypto key zeroize rsa key1
```

   The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.

   You can run the **crypto key zeroize** command only in the **exec** mode.

   - In some situations, where you believe that the RSA keys were compromised in some way or the RSA keys should no longer be used, you should delete all the RSA keys from you router using the **rypto key zeroize rsa** command.
   - To remove a specific RSA key pair, use the *keypair-label* argument.
   - From Cisco IOS XR Release 7.3.2 onwards, you can delete key-pairs with the no form of the command from XR Config mode.

3. Use the **show crypto key mypubkey rsa** command to view the key details.

   Keys generated in config mode are visible in the running configuration.

   Example

```
Router# show crypto key mypubkey rsa
Fri Mar 27 14:00:20.954 IST
Key label: the_default
Type : RSA General purpose
Size : 3072
Created : 01:13:10 IST Thu Feb 06 2025
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001
```

   The keys in this example are in OpenSSL format. Only those keys that are generated in the config mode are visible in the running configuration.

Your router is configured with the required RSA key pairs. You can confirm RSA key presence and validity using the verification command output.

Import a public key to a router

Configure public-key import on your router to enable certificate authority interoperability and support secure authentication.

Import a public RSA key to your router for user authentication and certificate authority interoperability.

This task demonstrates how to import a public key to a router to authenticate users and establish a secure relationship with a certificate authority.

Before you begin

- Review all prerequisites required by your certificate authority.
- Make sure you have the public RSA key to be imported.

Procedure

1. Import the public key to the router.

   Example

   ```
   Router# crypto key import authentication rsa general-keys
   ```

   Generates RSA key pairs.

   - Use the **usage keys** keyword to specify special usage keys; use the **general-keys** keyword to specify general-purpose RSA keys.
   - The *keypair-label* argument is the RSA key pair label that names the RSA key pairs.

2. Verify the imported key configuration.

   Example

   ```
   Router# show crypto key mypubkey rsa
   Fri Mar 27 14:00:20.954 IST
   Key label: system-root-key
   Type : RSA General purpose
   Size : 2048
   Created : 01:13:10 IST Thu Feb 06 2020
   Data :
    30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
    00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
    B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
    F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
    39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
    AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
    9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
    EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
    22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
    11020301 0001
   Key label: system-enroll-key
   Type : RSA General purpose
   Size : 2048
   Created : 01:13:16 IST Thu Feb 06 2020
   Data :
    30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
    009DBC14 C83604E4 EB3D3CF8 5BA7FDDB 80F7E85B 427332D8 BBF80148 F0A9C281
    49F87D5C 0CEBA532 EBE797C5 7F174C69 0735D13A 493670CB 63B04A12 4BCA7134
    EE0031E9 047CAA1E 802030C5 6071E8C2 F8ECE002 CC3B54E7 5FD24E5C 61B7B7B0
    68FA2EFA 0B83799F 77AE4621 435D9DFF 1D713108 37B614D3 255020F9 09CD32E8
    82B07CD7 01A53896 6DD92B5D 5119597C 98D394E9 DBD1ABAF 6DE949FE 4A8BF1E7
    851EB3F4 60B1114A 1456723E 063E50C4 2D410906 BDB7590B F1D58480 F3FA911A
    6C9CD02A 58E68D04 E94C098F 0F0E81DB 76B40C55 64603499 2AC0547A D652412A
    BCBBF69F 76B351EE 9B2DF79D E490C0F6 92D1BB97 B905F33B FAB53C20 DDE2BB22
    C7020301 0001
   ```

   (Optional) Displays the RSA public keys for your router.

The **show running-config** command also displays the RSA keys. The keys in the following example are in OpenSSL format.

> **Note**
>
> Only those keys that are generated in the config mode are visible in the running configuration.

The router will have the imported public RSA key available for authentication and certificate authority interoperability. Successful configuration is confirmed when the verification output matches the example data.

Configure CA trustpoints

Configure CA trustpoints for certificate authority interoperability by using CLI commands and verifying completion via command output.

Set up a certificate authority (CA) trustpoint on the router so it can verify peer certificates for secure communication.

This task establishes a trusted CA by configuring trustpoint parameters, setting enrollment methods, and defining certificate request behavior.

> **Warning**
>
> If a `commit replace` operation removes a crypto CA trustpoint or certificate configuration, the system can clear the installed certificates when it processes the delete event. If the operation later fails and the configuration is restored, the certificates might not be restored. Services that use the certificates can become unavailable. After a failed `commit replace`, verify that the required certificates are still installed with **show crypto ca certificates** comamnd and reinstall any certificates that were cleared.

Before you begin

- Review the prerequisites for your specific certificate authority system.
- Ensure you have access to all required URLs and configuration information.

Procedure

1. Create the CA trustpoint with a specific name.

   Example

   ```
   Router# configure
   Router(config)# crypto ca trustpoint myca
   ```

   Declares a CA.

   - Configures a trusted point with a selected name so that your router can verify certificates issued to peers.
   - Enters trustpoint configuration mode.

2. Specify the enrollment method by setting the CA URL.

   Example

   ```
   Router(config-trustp)# enrollment url
   http://ca.domain.com/certsrv/mscep/mscep.dll
   ```

   Specifies the URL of the CA.

   - The URL should include any nonstandard cgi-bin script location.

3. (Optional) Configure an LDAP query URL

   Example

   ```
   Router(config-trustp)# query url ldap://my-ldap.domain.com
   ```

   Specifies the location of the LDAP server if your CA system supports the LDAP protocol.

4. (Optional) Set certificate request retry parameters.

   Example

   ```
   Router(config-trustp)# enrollment retry period 2
   ```

   Example

   ```
   Router(config-trustp)# enrollment retry count 10
   ```

   Specifies a retry period and retry count.

   - After requesting a certificate, the router waits to receive a certificate from the CA. If the router does not receive a certificate within a period of time (the retry period) the router will send another certificate request.
   - Range is from 1 to 60 minutes. Default is 1 minute.

5. (Optional) Specify a named RSA key pair for this trustpoint.

   Example

   ```
   Router(config-trustp)# rsakeypair mykey
   ```

   Specifies a named RSA key pair generated using the **crypto key generate rsa** command for this trustpoint.

   - Not setting this key pair means that the trustpoint uses the default RSA key in the current configuration.

6. Save or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.

- End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
- Yes: Saves configuration changes and exits the configuration session.
- No: Exits the configuration session without committing the configuration changes.
- Cancel: Remains in the configuration session without committing the configuration changes.

The CA trustpoint is configured when your verification output matches the source example, confirming certificate authority interoperability.

Configure CA authentication

Configure CA authentication on your router to securely validate the certificate authority and establish trust for certificate operations.

Establish trust between your router and the certificate authority by authenticating the CA certificate.

This task enables your router to authenticate the certificate authority (CA) by obtaining and verifying the CA's self-signed certificate, which contains the CA's public key. Manual authentication is required by comparing the fingerprint of the CA certificate with information provided by the CA administrator.

Before you begin

Review the certificate authority prerequisites for your router.

Procedure

---

1. Authenticate the certification authority.

   Example

   ```
   Router# crypto ca authenticate myca
   ```

   Authenticates the CA to your router by obtaining a CA certificate, which contains the public key for the CA.

2. Verify the configuration or operational state.

   Example

   ```
   Router# show crypto ca certificates
   ```

   (Optional) Displays information about the CA certificate.

The task is complete when the configuration and verification output matches the expected source examples.

Multi-tier certification authorities

Explains multi-tier certification authorities for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A multi-tier certification authority is a certificate authority interoperability component that

- supports certificate-based trust in Cisco IOS XR deployments
- allows importing a complete CA hierarchy (from the Root CA to subordinate CAs) as part of terminal-based enrollment, and

- provides flexibility and security by enabling use of a certificate chain (including Root and subordinate CAs) for trustpoint authentication.
- Terminal-based enrollment: During terminal-based enrollment of a CA trustpoint, Cisco IOS XR network devices initially accepted only Root CA certificates. However, some network topologies use a multi-tier CA hierarchy, offering more flexibility and security. From Cisco IOS XR Software Release 7.10.1 and later, you can import a complete CA hierarchy—spanning the Root CA to the subordinate CA that issues the certificate—in a single authentication request. This includes providing the certificate chain (Root CA and all intermediate subordinate CAs) as part of the terminal-based enrollment process. This is especially useful when an existing CA hierarchy issues router certificates from a subordinate CA rather than the root.
- CA tiering limit: You can have a maximum of 8 tiers—a chain of CA consisting of one Root CA and up to seven subordinate CAs—for trustpoint authentication.

Feature history

Table 44: Feature History Table

Feature Name	Release Information	Feature Description
Multi-Tier Certificate Authority for Trustpoint Authentication	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Multi-Tier Certificate Authority for Trustpoint Authentication	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Multi-Tier Certificate Authority for Trustpoint Authentication	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Multi-Tier Certificate Authority for Trustpoint Authentication	Release 7.10.1	Apart from the root certificate authority (CA), you can now use a subordinate CA to issue certificates and authenticate your network devices. This feature is beneficial when you have an existing CA hierarchy where it is not the root CA but the subordinate CA that issues the leaf or router certificates. In earlier releases, you could associate only a single CA, not a multi-tier CA, to a trustpoint. And, you could use only the root CA certificate to enroll the router certificates. This feature modifies the **show crypto ca certificates** command to display the **Trusted Certificate Chain** field.

Configure multi-tier CA trustpoint authentication

Configure multi-tier CA trustpoint authentication using PEM-encoded certificates and verify successful trustpoint enrollment.

Establish trustpoint authentication with a multi-tier certificate authority (CA) by configuring the required PEM-encoded certificates.

Multi-tier CA trustpoint authentication allows routers to establish secure connections using multiple certificates in a CA hierarchy. Only PEM-encoded certificates are accepted for this process.

Before you begin

- Generate a key pair on the router.
- Import the public key.
- Configure a trustpoint as detailed in prior sections.

Procedure

1. Authenticate the trustpoint using multi-tier CA certificates.

Example

```
Router# crypto ca authenticate test-ca
Mon Feb  6 08:17:48.943 UTC


Enter the base 64/PEM encoded certificate/certificates.
Please note: for multiple certificates use only PEM
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIF5TCCA82gAwIBAgICEAEwDQYJKoZIhvcNAQELBQAwXTELMAkGA1UEBhMCSU4x
CzAJBgNVBAgMAktBMQwwCgYDVQQHDANCR0wxDTALBgNVBAoMBENTQ08xDTALBgNV
.
.
.
```

```
/4UzeeX6ll0gGJVbDwGeIZTH00artqxHquKQ2P7eXQ1pg0PRNRqWN90SvT5yE33N
eHgbtvdHg1K6K6IAj/NGnd7xUrA1TQ4bdmouCNkgbXM/G9DwgkOOvZ8KYRP9JW57
LYIv2ZcRS2vdnZRD9JPGVig2EgcfVPtj+Q==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIF9TCCA92gAwIBAgIUD6AGesleqedhorkrJ9HWjz1RQzswDQYJKoZIhvcNAQEL
BQAwXTELMAkGA1UEBhMCSU4xCzAJBgNVBAgMAktBMQwwCgYDVQQHDANCR0wxDTAL
.
.
.
+6rMWd6BmfSy2PT3Qz5AjO2+3N1dd67qRRrX7skklkX4JXY42n5/l9PQtSp0wTBh
uy5yUAagynu0z07GczE7E9V+tJHRmNTbnd8pxLk4lTwqtiCIXwQLZA75SkwCS5wh
fn7OrV7uFjMaggNkvj0kSSOkWxqJ+j/KqMAA2zQMUV+qdvT6i+ZV44U=
-----END CERTIFICATE-----
Serial Number  : 10:01
  Subject:
 CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 12:31:40 UTC Sun Jun 14 2020
  Validity End   : 12:31:40 UTC Wed Jun 12 2030

  CRL Distribution Point
 http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
  D8E0C11ECED96F67FDBC800DB6A126676A76BD62
  Serial Number  : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B

  Subject:
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
 CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 13:12:32 UTC Sun Jun 07 2020
  Validity End   : 13:12:32 UTC Sat Jun 02 2040

  CRL Distribution Point
 http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
  08E71248FB7578614442E713AC87C461D173952F

CA Certificate validated using issuer certificate.
Router#
```

2. Verify the configuration.

Example

```
Router# show crypto ca certificates test-ca
Mon Feb  6 09:03:53.019 UTC

Trustpoint       : test-ca
==================================================
CA certificate
  Serial Number  : 10:01
  Subject:
            CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
            CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 12:31:40 UTC Sun Jun 14 2020
  Validity End   : 12:31:40 UTC Wed Jun 12 2030
```

```
  CRL Distribution Point
           http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
            D8E0C11ECED96F67FDBC800DB6A126676A76BD62
Trusted Certificate Chain
  Serial Number  : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B

  Subject:
           CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
           CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 13:12:32 UTC Sun Jun 07 2020
  Validity End   : 13:12:32 UTC Sat Jun 02 2040

  CRL Distribution Point
           http://10.105.236.78/crl_akshath_two_level_ca/crl.der
  SHA1 Fingerprint:
            08E71248FB7578614442E713AC87C461D173952F
Router certificate
  Key usage      : General Purpose
  Status         : Available
  Serial Number  : 28:E5
  Subject:
           CN=test
  Issued By      :
           CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 08:49:54 UTC Mon Feb 06 2023
  Validity End   : 08:49:54 UTC Wed Mar 08 2023
  SHA1 Fingerprint:
            6C8644FA67D9CEBC7C5665C35838265F578835AB
Associated Trustpoint: test-ca
```

Use the **show crypto ca certificates** *trustpoint-name* command to view the CA certificate chain. The command output displays the **Trusted Certificate Chain** field if there is one or more subordinate CAs involved in the hierarchy.

The router successfully authenticates the multi-tier CA trustpoint and validates the certificate chain. You can view the CA certificate chain using the **show crypto ca certificates trustpoint-name** command. The process is complete when all certificates are validated and associated with the configured trustpoint.

Obtain router certificates for your router

Obtain router certificates required for secure network operation.

Request and verify certificates from a certificate authority (CA) for each router RSA key pair.

You must obtain a signed certificate for each RSA key pair on your router. If using general-purpose keys, only one certificate is needed. If using special usage keys, two certificates are required. These certificates establish trust between your router and the CA for secure communication.

Before you begin

- Review all prerequisites outlined by your certificate authority.
- Prepare the challenge password you will use for certificate revocation.

Procedure

---

1. Request the router certificate from the CA.

Example

```
Router# crypto ca enroll myca
```

Requests certificates for all of your RSA key pairs.

- This command requests certificates for all existing RSA key pairs. Perform this command only once, even if you have multiple key pairs.
- You will be prompted to create a challenge password. This password is not saved with your configuration and is required if the certificate needs to be revoked. Remember it securely.
- A certificate may be issued immediately. If not, the router will resend the request every minute until the retry period expires. If a timeout occurs, contact your system administrator to approve your request and repeat the command.

2. Verify the configuration or operational state to ensure certificates were obtained.

Example

```
Router# show crypto ca certificates
```

(Optional) Displays information about the CA certificate.

Your router certificates are successfully obtained and verified when the output matches the expected configuration example.

CA enrollment URLs

Explains CA enrollment URLs for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A CA enrollment URL is a certificate authority interoperability component that

- serves as a web address allowing routers to connect with a Certification Authority (CA) for certificate enrollment and renewal
- automates the process of obtaining digital certificates, minimizing manual intervention, and
- reduces the risk of potential errors associated with manual certificate management.

Feature history
Table 45: Feature History Table

Feature Name	Release Information	Description
IPv6 support for CA enrollment URL	Release 26.2.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is now supported on Cisco 8404-SYS-D routers.

Feature Name	Release Information	Description
IPv6 support for CA enrollment URL	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*) *This feature is supported on Cisco 8712-MOD-M routers.
IPv6 support for CA enrollment URL	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can enhance network compatibility and simplify management in modern network environments using IPv6 addresses and CA server URLs that resolve to IPv6 addresses as enrollment URLs for the CA hosted on IPv6-based servers. This improvement addresses previous limitations that caused configuration issues and failures when using IPv6 CA enrollment URL.

How CA enrollment URLs work

Describes how CA enrollment URLs operate in Cisco IOS XR, including the enrollment sequence, key participants, and the resulting certificate authority interoperability workflow.

- When a router needs to enroll for a new certificate or renew an existing one, the router sends a request to the enrollment URL. The requests sent to the enrollment URL typically include the device's identity information and its public key.
- Upon receiving a request at the enrollment URL, the CA processes it to verify the identity of the requester. The verification involves checking the authenticity and validity of the information provided in the enrollment request.
- After the validation is successful, the CA issues a digital certificate. This certificate is then sent back to the router through the enrollment URL.
- The router then installs and use this certificate for secure communications.

Summary

The key components involved in the process are:

- Router: Initiates a request to the enrollment URL to obtain or renew a digital certificate, providing device identity and a public key.
- Certificate Authority (CA): Receives and processes enrollment requests, verifies identity, and issues digital certificates.
- Enrollment URL: Serves as the endpoint through which routers communicate with the CA to exchange requests and certificates.

Certificate Authority (CA) enrollment URLs in Cisco IOS XR facilitate secure certificate acquisition and renewal for routers by automating communication with the certificate authority.

Workflow

These stages describe how CA enrollment URLs work.

1. Request submission: When a router needs a new certificate or must renew an existing one, it sends a request to the enrollment URL containing its identity information and public key.
2. Identity verification: Upon receiving the request, the CA verifies the authenticity and validity of the router's information.
3. Certificate issuance: If verification succeeds, the CA issues a digital certificate and sends it to the router via the enrollment URL.
4. Certificate installation: The router installs the received certificate, enabling secure communications.

Result

The process enables routers and certificate authorities to achieve certificate interoperability, securing communications through validated digital certificates.

IPv4 address interoperability for CA enrollment URLs

Lists supported IPv4 address formats and facts for certificate authority (CA) enrollment URLs so you can verify interoperability or plan configuration.

Routers support IPv4 addresses or web addresses that resolve to IPv4 addresses as valid CA server addresses when you specify the CA URL using the **enrollment url** command.

Key facts

- You can specify either a direct IPv4 address or a domain name that resolves to an IPv4 address as the CA server address in the enrollment URL.
- This configuration enables the router to enroll for certificates with certificate authorities that use IPv4 addressing.
- Refer to these facts when planning, configuring, or validating CA interoperability for device certificate enrollment.

Reference purpose

Use this reference to quickly look up supported IPv4 formats or behaviors for CA enrollment URLs when planning certificate authority integration or troubleshooting certificate enrollment.

IPv6 address interoperability for CA enrollment URLs

Provides details on configuring IPv6 addresses as Certificate Authority (CA) enrollment URLs, command usage, and interoperability in Cisco IOS XR Release 24.4.1 and later.

Starting from Cisco IOS XR Release 24.4.1, you can configure IPv6 addresses or web addresses that resolve to an IPv6 address as the certificate authority (CA) server address when specifying the CA enrollment URL using the **enrollment url** command. This update removes previous limitations that prevented successful CA declaration using IPv6 in the **crypto ca trust point** command.

Use this reference to:

- Look up configuration details and command usage for IPv6 CA enrollment URLs.
- Verify certificate authority interoperability behavior when using IPv6 addresses.

Apply these facts when planning, configuring, or validating CA interoperability:

- You can specify IPv6 addresses directly or use hostnames that resolve to IPv6.
- The **enrollment url** command now supports IPv6, ensuring compatibility for certificate enrollment tasks.
- Restrictions from earlier releases have been eliminated, so IPv6-based CAs can be configured successfully.

Requirement: You must follow CA enrollment URL guidelines

Outlines the requirement for following CA enrollment URL guidelines with specific prerequisites, restrictions, and supported deployment conditions.

To ensure certificate authority deployment remains supported, you must follow these CA enrollment URL guidelines:

- The enrollment URL string must start with *http://CA_name*, where *CA_name* is the host Domain Name System (DNS) name or IP address of the CA (for example, *http://ca-server*).
- If the CA CGI-bin script location is not */CGI-bin/pkiclient.exe* at the CA (the default), include the nonstandard script location in the enrollment URL as *http://CA_name/script-location*, where *script-location* is the full path to the CA scripts.

Configure CA enrollment URLs

Configure CA enrollment URLs to enable certificate authority

Set up CA enrollment URLs to ensure your device can interact with a certificate authority for secure certificate enrollment.

Use this task when you need to establish a secure enrollment process between your device and a certificate authority by specifying the correct enrollment URL.

Before you begin

- Review related certificate authority prerequisites.
- Ensure your device is properly configured for certificate management.

Procedure

1. Configure the device for certificate authority trustpoint

   Example

   ```
   Router# configure
   Router(config)# crypto ca trustpoint myca
   ```

2. Set the CA enrollment URL

   Example

   ```
   Router(config-trustp)# enrollment url
   http://ca.domain.com/certsrv/mscep/mscep.dll
   ```

3. Verify the enrollment URL configuration.

   Example

   ```
   Router# show running-config crypto ca trustpoint myca
   crypto ca trustpoint myca
     enrollment url http://ca.domain.com/certsrv/mscep/mscep.dll
    !
   ```

The CA enrollment URL is configured and verified; your device is ready to interact with the certificate authority using the specified URL.

Configure cut-and-paste certificate enrollment on your router

Configure cut-and-paste certificate enrollment on your router to enable secure manual certificate exchange with a certificate authority (CA).

Set up and verify manual certificate enrollment with a certificate authority using cut-and-paste procedures.

This task lets you configure your router to manually enroll and import certificates from a certificate authority, enabling secure communication using trusted certificates.

Before you begin

- Review the certificate authority prerequisites for your environment.
- Ensure your router is prepared for certificate operations.

Procedure

1. Create the certificate authority (CA) trustpoint.

   Example

   ```
   Router# crypto ca trustpoint myca
   ```

   Declares the CA that your router should use and enters trustpoint configuration mode.

   - Use the *ca-name* argument to specify the name of the CA.

2. Configure the manual enrollment method.

   Example

   ```
   Router(config-trustp)# enrollment terminal
   ```

   Specifies manual cut-and-paste certificate enrollment.

3. Save or discard the configuration changes.

   Example

   ```
   commit
   end
   ```

   Use one of these options:

   - Commit: Saves the configuration changes and remains within the configuration session.
   - End: Prompts you to save, discard, or cancel the configuration changes before leaving the configuration session.
   - Yes: Saves configuration changes and exits the configuration session.
   - No: Exits the configuration session without committing the configuration changes.
   - Cancel: Remains in the configuration session without committing the configuration changes.

4. Authenticate the certificate authority.

Example

```
Router# crypto ca authenticate myca
```

Authenticates the CA by obtaining the certificate of the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered previously.

5. Request the router certificate from the CA.

Example

```
Router# crypto ca enroll myca
```

Obtains the certificates for your router from the CA.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered previously.

6. Import the certificate through cut-and-paste.

Example

```
Router# crypto ca import myca certificate
```

Imports a certificate manually at the terminal.

- Use the *ca-name* argument to specify the name of the CA. Use the same name that you entered previously.

> **Note**
>
> You must enter the crypto ca import command twice if usage keys (signature and encryption keys) are used. The first time the command is entered, one of the certificates is pasted into the router; the second time the command is entered, the other certificate is pasted into the router. (It does not matter which certificate is pasted first.

7. Verify the configuration and certificates.

Example

```
Router# show crypto ca certificates
```

Displays information about your certificate and the CA certificate.

Cut-and-paste certificate enrollment is complete when verification output matches the expected example and your router is properly enrolled with the CA.

You can configure CA interoperability with commands like:

```
configure
hostname myrouter
domain name mydomain.com
end
```

```
Uncommitted changes found, commit them? [yes]:yes

crypto key generate rsa mykey

The name for the keys will be:mykey
Choose the size of the key modulus in the range of 360 to 2048 for your General
 Purpose Keypair
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus [1024]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]

show crypto key mypubkey rsa

Key label:mykey
Type     :RSA General purpose
Size     :1024
Created  :17:33:23 UTC Thu Sep 18 2003
Data     :
 30819F30 0D06092A 864886F7 0D010101 05000381 8D003081 89028181 00CB8D86
 BF6707AA FD7E4F08 A1F70080 B9E6016B 8128004C B477817B BCF35106 BC60B06E
 07A417FD 7979D262 B35465A6 1D3B70D1 36ACAFBD 7F91D5A0 CFB0EE91 B9D52C69
 7CAF89ED F66A6A58 89EEF776 A03916CB 3663FB17 B7DBEBF8 1C54AF7F 293F3004
 C15B08A8 C6965F1E 289DD724 BD40AF59 E90E44D5 7D590000 5C4BEA9D B5020301
 0001

! The following commands declare a CA and configure a trusted point.

configure
crypto ca trustpoint myca
enrollment url http://xyz-ultra5
enrollment retry count 25
enrollment retry period 2
rsakeypair mykey
end

Uncommitted changes found, commit them? [yes]:yes

! The following command authenticates the CA to your router.

crypto ca authenticate myca

Serial Number  :01
Subject Name   :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San
Jose,st=CA,c=US
Issued By      :
cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San
Jose,st=CA,c=US
Validity Start :07:00:00 UTC Tue Aug 19 2003
Validity End   :07:00:00 UTC Wed Aug 19 2020
Fingerprint:58 71 FB 94 55 65 D4 64 38 91 2B 00 61 E9 F8 05
Do you accept this certificate?? [yes/no]:yes

! The following command requests certificates for all of your RSA key pairs.

crypto ca enroll myca

% Start certificate enrollment ...
% Create a challenge password. You will need to verbally provide this
  password to the CA Administrator in order to revoke your certificate.
```

```
% For security reasons your password will not be saved in the configuration.
% Please make a note of it.

Password:
Re-enter Password:
    Fingerprint: 17D8B38D ED2BDF2E DF8ADBF7 A7DBE35A

! The following command displays information about your certificate and the
CA certificate.

show crypto ca certificates

Trustpoint       :myca
==========================================================
CA certificate
  Serial Number  :01
  Subject Name   :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San
Jose,st=CA,c=US
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San
Jose,st=CA,c=US
  Validity Start :07:00:00 UTC Tue Aug 19 2003
  Validity End   :07:00:00 UTC Wed Aug 19 2020
Router certificate
  Key usage      :General Purpose
  Status         :Available
  Serial Number  :6E
  Subject Name   :
        unstructuredName=myrouter.mydomain.com,o=Cisco Systems
  Issued By      :
        cn=Root coax-u10 Certificate Manager,ou=HFR,o=Cisco Systems,l=San
Jose,st=CA,c=US
  Validity Start :21:43:14 UTC Mon Sep 22 2003
  Validity End   :21:43:14 UTC Mon Sep 29 2003
  CRL Distribution Point
       ldap://coax-u10.cisco.com/CN=Root coax-u10 Certificate Manager,O=Cisco
 Systems
```

Certification authority interoperability facts

Lists key facts, examples, and lookup details for certification authority interoperability to help you verify and plan certificate deployments.

he following reference information is provided to support certification authority interoperability:

- Supported certificate authority types:
  - Root CA
  - Intermediate CA
  - External CA
  - Self-signed CA
- Interoperability examples:
  - Certificate signing requests (CSRs) generated by device or software are compatible with root and intermediate certificate authorities using PEM encoding.

- Import and validation of external CA certificates may require conversion to DER format.

- Command outputs for validation:
  - Use the **show crypto pki certificates** to verify certificate installation.
  - Use the **show crypto pki trustpoints** to display certificate authority configuration
  - Use the **show crypto pki** and **debug crypto pki events** commands to check interoperability status.

Planning considerations

- Ensure certificate chain includes both root and intermediate authorities for proper trust validation.
- When configuring certificate authorities, align key usage and extended key usage attributes with interoperability requirements.
- Validate certificate authority profiles against the latest security policy requirements.

Certification authority interoperability standards

Provides certification authority interoperability standards facts, examples, command output, or lookup details from the source chapter so the reader can verify certificate authority interoperability behavior.

Cisco supports the following standards:

- Public-Key Cryptography Standard #7 (PKCS #7): Encrypts and signs certificate enrollment messages.
- Public-Key Cryptography Standard #10 (PKCS #10): Provides syntax for certificate requests.
- RSA keys: Developed by Ron Rivest, Adi Shamir, and Leonard Adelman; RSA keys come in pairs (one public key and one private key).
- SSL: Secure Socket Layer protocol.
- X.509v3 certificates: Enables IPSec-protected networks to scale by providing digital ID cards for devices. When devices communicate, they exchange digital certificates to prove identity, eliminating manual public key exchange. Certificates are obtained from a Certificate Authority (CA); X.509 is part of the X.500 standard of the ITU.

> **Note**
>
> The Internet Key Exchange (IKE) standard is not supported.

Certification authorities

Explains certification authorities for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A certification authority is a trust management entity that

- issues digital certificates to validate the identity of devices and users
- verifies certificate authenticity and manages certificate revocation, and
- enables interoperability for certificate-based trust across Cisco IOS XR deployments.

Additional reference information

Certification authorities (CAs) play a critical role in establishing and maintaining secure communication within network environments. They handle certificate issuance, manage registration authorities, and define operational constraints for interoperability.

A typical deployment might use multiple certification authorities to ensure secure device authentication between routers running Cisco IOS XR. CA registration authorities may serve as intermediaries to facilitate certificate enrollment and lifecycle management.

Certification authority purposes

Explains certification authority purposes for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

Certification authority purposes are certificate authority interoperability components that

- manage certificate requests and issue certificates to participating IPSec network devices to provide centralized key management
- simplify the administration of IPSec network devices across multiple IPSec-compliant devices, such as routers, and
- enable digital signatures, using public key cryptography, to authenticate devices and individual users.
- Digital signatures are enabled by public key cryptography. Each user possesses a key pair: a public and a private key. The keys complement each other—data encrypted with one key can be decrypted with the other. A signature is formed when data is encrypted with a user's private key; the receiver verifies the signature by decrypting the message with the sender's public key. Successful decryption indicates the sender is the holder of the private key, enabling trust.
- Digital certificates provide the link. A digital certificate contains identifying information (such as name, serial number, company, department, or IP address) and a copy of the entity's public key. The certificate is signed by a certification authority (CA), a trusted third party. To validate the CA's signature, the receiver must know the CA's public key, usually set during installation or via configuration (for example, browsers preconfigured with several CAs, or IKE authenticating peer devices).
- Without digital signatures, manual exchange of public keys or secrets is required between device pairs. Adding new devices necessitates changes across all devices. With certificates, each device enrolls with a CA; when two devices communicate, they exchange certificates and digital signatures to authenticate each other. Adding a new device requires only CA enrollment, no updates to other devices. Certificates are exchanged automatically, allowing authenticated secure connections.

CA registration authorities

Explains CA registration authorities for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A CA registration authority is a certificate authority interoperability component that

- acts as a proxy server to handle certification authority (CA) functions when the CA is offline
- supports certificate enrollment, validation, or trust management on Cisco IOS XR routers, and
- facilitates interoperability and operational continuity for certificate-based trust deployments.

Some CAs include a registration authority (RA) as part of their implementation. The RA serves as a proxy for the CA, allowing CA functions to continue even when the CA itself is offline.

Additional reference information

Registration authorities play a crucial role in managing certificates in environments that require high availability and redundancy. By acting as intermediaries, they ensure that certificate lifecycle operations such as enrollment and validation are not disrupted during CA downtime.

10 CA Trust Pools and PKI Certificate Lifecycle

Topics:

- Certificate authority trust pools
- PKI certificate expiry notifications
- Automatic PKI certificate renewal

Outlines the principles, configuration, and management of CA trust pools and the PKI certificate lifecycle, emphasizing trust models, certificate operations, and key management to ensure secure authentication and communication within a network environment.

Certificate authority trust pools

Introduces the concept of CA trust pools, describing trust models, foundational principles, and their significance in establishing trusted entities for secure network communications.

A certificate authority trust pool is a certificate authority interoperability component that

- supports certificate-based trust in Cisco IOS XR deployments
- authenticates sessions—such as HTTPS—between devices using commonly recognized trusted agents (certificate authorities or CAs), and
- provisions, stores, and manages a pool of certificates from known CAs, including both built-in and downloaded certificates.

Additional reference information

The trust pool feature is enabled by default in Cisco IOS XR software to create a scheme similar to a browser's service for securing sessions. A special trusted point called a trust pool is designated, containing multiple known CA certificates from Cisco and potentially other vendors. The trust pool includes both built-in and downloaded CA certificates, supporting flexible interoperability across certification authorities. The feature also includes the child topic "CA Certificate Bundling in the Trust Pool."

CA certificate bundles in trust pools

Explains CA certificate bundles in trust pools for Cisco IOS XR certificate-based trust, including source behavior, constraints, and operational details that support certification authority interoperability.

A CA certificate bundle is a certificate authority interoperability component that

- supports certificate-based trust in Cisco IOS XR deployments
- contains certificates packaged into a baseline image in a dedicated store called a CA trust pool, and
- is updated automatically by Cisco and recognized by other vendors.

Additional reference information

The CA certificate bundle is available in different formats to enhance interoperability:

- Privilege Management Infrastructure (PMI) certificates in Distinguished Encoding Rules (DER) binary format, encapsulated in Public-Key Cryptographic Message Syntax Standard 7 (PKCS7).
- Files with concatenated X.509 certificates in Privacy Enhanced Mail (PEM) format using PEM headers.

Consider a router running Cisco IOS XR:

The router uses a built-in CA certificate bundle stored in a special trust pool. This bundle is automatically updated to maintain interoperability with multiple certification authorities, ensuring secure communications.

Requirement: CA trust pool prerequisites

Outlines the prerequisites required to deploy a certificate authority and maintain support for CA trust pools.

To use the certification authority, ensure that the software image includes a crypto subsystem. Crypto is Cisco's proprietary encryption mechanism, available in the baseline software image, and is required for secure certificate authority operations.

Requirement: Follow CA trust pool restrictions

Outlines the requirements for maintaining proper CA trust pool restrictions and keeping certificate authority deployment supported.

Ensure you follow these requirements and restrictions for CA trust pools:

- Device certificates that use CA certificates cannot be enrolled in a CA trust pool.
- Starting with Cisco IOS XR software version 7.3.3, server certificates (leaf certificates) on the router must include a Fully Qualified Domain Name (FQDN) in the Common Name (CN) field.
- To add an IP address in the Subject Alternate Name (SAN) field of server certificates, add the extension type as IP address in the certificate. If the IP address extension type configuration isn't available, use the **crypto ca fqdn-check ip-address allow** command for the router to validate the IP address in the SAN field successfully.

How CA trust pools are updated

Describes how CA trust pools are updated in Cisco IOS XR, including sources, participants, and the update sequence that achieves certificate authority interoperability.

Cisco IOS XR considers the CA trust pool as a single entity. Any update replaces the entire trust pool. Built-in certificates in the trust pool cannot be physically replaced, but if their X.509 subject-name attribute matches a certificate in the updated bundle, the built-in certificate is deactivated after an update.

The CA trust pool is considered as a single entity, As such, any update you perform will replace the entire trustpool.

Summary

The key components involved in the process are:

- CA trust pool: The consolidated set of trusted certificate authority certificates maintained on the device.
- Certificate bundle source: The URL or location from which updated certificate bundles are obtained for replacement or augmentation.
- Administrator: Manages manual updates and reviews certificate warnings or errors presented by the system.

CA trust pools are updated in Cisco IOS XR to maintain interoperability with certificate authorities and ensure secure communications. Updates are triggered by certificate expiration, reissuance, the addition of new trusted certificates, or corruption of the trust pool configuration.

Workflow

These stages describe how CA trust pools are updated.

1. Trigger conditions: The process begins when:
   - A certificate in the trust pool expires or is reissued.
   - The published CA certificate bundle contains additional certificates required by specific applications.
   - The device configuration is found to be corrupted.
2. Update method selection: The system or administrator chooses either of the following methods:
   - Automatic update:
     - A timer is established based on the certificate with the earliest expiration time.
     - If a bundle location is not configured or not explicitly disabled, syslog warnings are issued at intervals to notify the administrator.

    - Automatic updates use the configured bundle URL. Upon trust pool expiration, the system reads the policy, loads the bundle, and replaces the PKI trust pool.
    - If automatic update encounters problems, the system retries with the following schedule: 20 days, 15 days, 10 days, 5 days, 4 days, 3 days, 2 days, 1 day, then once every hour until the download is successful.
  - Manual update:
    - The administrator manually initiates an update using provided commands or procedures.

3. Trust pool replacement: The CA trust pool is replaced as a single unit, making sure all necessary certificates are active. Built-in certificates that match those in the published bundle are deactivated.
4. Result confirmation: The system confirms the update. The administrator reviews the new trust pool and resolves any syslog warnings or errors.

Result

The process ensures that the CA trust pools remain up to date, certificates are valid and trusted, and interoperability with certificate authorities is successfully maintained.

Configure manual trust pool certificate updates

Configure manual updates for the trust pool certificate bundle to ensure proper interoperability with certificate authorities.

Manually update the trust pool certificate bundle when automatic updates are unavailable or certain certificates require replacement.

The CA trust pool feature is enabled by default and uses a built-in CA certificate bundle, which receives automatic updates from Cisco. Manual updates are necessary if certificates are outdated, corrupt, or need specific replacement.

Before you begin

Review certificate authority prerequisites for your environment.

Procedure

1. (Optional) Remove all downloaded CA certificates.

   Example

   ```
   Router# crypto ca trustpool import url clean
   ```

   (Optional) Manually removes all downloaded CA certificates. This command is run in the EXEC mode.

2. Import the updated CA certificate bundle.

   Example

   ```
   Router# crypto ca trustpool import url
   http://www.cisco.com/security/pki/trs/ios.p7b
   ```

   Specify the URL from which the CA trust pool certificate bundle must be downloaded. This manually imports (downloads) the CA certificate bundle into the CA trust pool to update or replace the existing CA certificate bundle.

3. Verify the trust pool configuration and operational state.

Example

Example

```
Router# show crypto ca trustpool

Trustpool: Built-In
================================================
CA certificate
  Serial Number  : 5F:F8:7B:28:2B:54:DC:8D:42:A3:15:B5:68:C9:AD:FF
  Subject:
        CN=Cisco Root CA 2048,O=Cisco Systems
Issued By      :
        CN=Cisco Root CA 2048,O=Cisco Systems
  Validity Start : 20:17:12 UTC Fri May 14 2004
  Validity End   : 20:25:42 UTC Mon May 14 2029
  SHA1 Fingerprint:
         DE990CED99E0431F60EDC3937E7CD5BF0ED9E5FA

Trustpool: Built-In
================================================
CA certificate
  Serial Number  : 2E:D2:0E:73:47:D3:33:83:4B:4F:DD:0D:D7:B6:96:7E
Subject:
        CN=Cisco Root CA M1,O=Cisco
  Issued By      :
        CN=Cisco Root CA M1,O=Cisco
  Validity Start : 20:50:24 UTC Tue Nov 18 2008
  Validity End   : 21:59:46 UTC Fri Nov 18 2033
  SHA1 Fingerprint:
         45AD6BB499011BB4E84E84316A81C27D89EE5CE7
```

Displays the CA trust pool certificates of the router in a verbose format.

Manual trust pool certificate updates are complete when the configuration and verification outputs confirm that the certificate bundle has been successfully updated and matches the intended interoperability behavior.

CRL retrievals through HTTP proxy servers

Explains how CRL retrievals through HTTP proxy servers enable certificate-based trust in Cisco IOS XR, highlighting operational behaviors, constraints, and interoperability.

A CRL retrieval through HTTP proxy servers is a certificate authority interoperability mechanism that

- allows Cisco IOS XR routers to download Certificate Revocation Lists (CRLs) from certification authorities as part of certificate validation
- enables reuse of a CRL with the same certificate multiple times until the CRL expires, and
- retrieves new CRLs automatically when a certificate is received and the previous CRL has expired.

If the CRL Distribution Point (CDP) is not directly reachable, you can obtain the CRL through an HTTP proxy server using this feature.

Feature history
Table 46: Feature History Table

Feature Name	Release Information	Feature Description
Retrieve CRL through the HTTP Proxy Server	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Retrieve CRL through the HTTP Proxy Server	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Retrieve CRL through the HTTP Proxy Server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Retrieve CRL through the HTTP Proxy Server	Release 7.3.1	CRL contains the serial numbers of the third-party certificates that are invalidated by the issuing Certificate Authority. In the event that the CRL Distribution point (CDP) is not directly reachable, you can fetch the CRL through the http proxy server using the newly introduced crypto ca http-proxy command. Command modified for this feature: crypto ca crl request

Examples

Configuration example: To retrieve a CRL through the HTTP proxy server for smart licensing, use the following commands:

```
Router# config
Router(config)# crypto ca HTTP-proxy 10.10.10.1 port 1
Router(config)# commit


Router# license smart register idtoken NWRkMTJjZjYtMzJhNi00YzYxLWI3M$
Router# commit
```

Verification example:

Smart licensing registration is validated by fetching the CRL from the CDP, through the HTTP proxy server. If the validation is successful, the **show crypto ca crls** command displays the CRLs. If validation fails, no output is shown.

```
Router# show crypto ca crls
Thu Jun  6 13:43:00.763 UTC
CRL Entry
==========================================
  Issuer : CN=xyz-w2k Root CA 2,O=xyz Limited,C=BM
  Last Update : Dec 17 18:18:14 2018 GMT
  Next Update : Jun 15 18:18:14 2019 GMT
  CRL Distribution Point :
        http://xyz-w2k.cisco.com/CertEnroll/xyz-w2k-root.crl
CRL Entry
==========================================
  Issuer : CN=zxy-w2k SSL ICA G2,O=zxy,C=US
  Last Update : Jun  6 12:57:04 2019 GMT
  Next Update : Jun  9 12:57:04 2019 GMT
  CRL Distribution Point :
        http://zxy-w2k.cisco.com/CertEnroll/zxy-w2k-root.crl
RP/0/RP0/CPU0:ios#


Router# show license status
Smart Licensing is ENABLED
Utility:
  Status: DISABLED
Data Privacy:
  Sending Hostname: yes
    Callhome hostname privacy: DISABLED
    Smart Licensing hostname privacy: DISABLED
  Version privacy: DISABLED
Transport:
  Type: Callhome
Registration:
  Status: REGISTERED
  Smart Account: BU Production Test 1
  Virtual Account:
  Export-Controlled Functionality: ALLOWED
  Initial Registration: SUCCEEDED on Jun 06 2019 13:42:46 UTC
  Last Renewal Attempt: None
  Next Renewal Attempt: Dec 03 2019 13:42:46 UTC
  Registration Expires: Jun 05 2020 13:37:45 UTC
License Authorization:
  Status: AUTHORIZED on Jun 06 2019 13:42:55 UTC
  Last Communication Attempt: SUCCEEDED on Jun 06 2019 13:42:55 UTC
  Next Communication Attempt: Jul 06 2019 13:42:54 UTC
  Communication Deadline: Sep 04 2019 13:37:55 UTC
```

```
Export Authorization Key:
  Features Authorized:
    <none>
```

Note

To fetch the latest CRL from a specific CDP, use the crypto ca crl request <cdp-url> [HTTP-proxy <ip-address> port <port-number>] command.

Configure optional trust pool policy parameters

Configure optional trust pool policy parameters to adjust certificate authority interoperability settings for your router.

Allow you to customize certificate authority trust pool policy parameters, such as CA bundle URL, revocation list checking, and policy description, for enhanced certificate management and interoperability.

Adjusting optional trust pool policy parameters fine-tunes how your router interacts with certificate authorities. Use this task when you need to customize the CA trust pool policy for improved interoperability or compliance.

Before you begin

Review the related certificate authority prerequisites for your deployment.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Enter ca-trust pool configuration mode where commands can be accessed to configure CA trust pool policy parameters.

   Example

   ```
   Router(config)# crypto ca trust pool policy
   ```

   Enters ca-trustpool configuration mode where commands can be accessed to configure CA trust pool policy parameters.

3. Configure the CA bundle URL.

   Example

   ```
   Router(config-trustpool)# cabundle url
   http://www.cisco.com/security/pki/crl/crca2048.crl
   ```

   Specifies the URL from which the CA trust pool certificate bundle is downloaded.

4. Configure certificate revocation list (CRL) checking.

Example

```
Router(config-trustpool)# crl optional
```

Disables revocation checking when the trust pool policy is being used. By default, the router enforces a check of the revocation status of the certificate by querying the certificate revocation list (CRL).

5. Set the trust pool policy description.

Example

```
Router(config-trustpool)# description Trustpool for Test.
```

The optional trust pool policy parameters are correctly configured when your configuration and verification outputs match the examples provided.

How CA certificates work in trust pools and trustpoints

Describes the workflow of CA certificate handling in trust pools and trustpoints in Cisco IOS XR, outlining the sequence, participants, and result for certificate authority interoperability.

There may be cases where a Certificate Authority (CA) resides in both a trust pool and a trustpoint; for example, a trustpoint uses a CA and then a CA bundle is downloaded containing the same CA. In such scenarios, the router considers the CA and its policy in the trustpoint before considering the CA in the trust pool or its policy. This approach ensures that the implementation of the trust pool feature does not alter the router's existing behavior or authentication mechanisms.

The policy defines how the security appliance obtains the CA certificate and the authentication policies enforced for user certificates issued by the CA.

Summary

The key components involved in the process are:

- CA certificate: Provides authority for user certificate authentication.
- Trustpoint: Stores specific CA certificates and associated policies for authentication.
- Trust pool: Contains bundles of CA certificates and associated policies downloaded to the router.

CA certificates can reside in both trust pools and trustpoints within Cisco IOS XR. The workflow determines which certificate and policy are used when there are overlaps, ensuring current behaviors remain consistent as the trust pool feature is implemented.

Workflow

These stages describe how CA certificates work in trust pools and trustpoints.

1. When a CA certificate exists in both the trust pool and a trustpoint, Cisco IOS XR first evaluates the certificate and policy in the trustpoint.
2. The trustpoint's policy determines how the security appliance obtains the CA certificate and enforces authentication for user certificates issued by that CA.
3. If no relevant certificate or policy exists in the trustpoint, the system evaluates the trust pool and its associated policy.
4. This order ensures that existing authentication behaviors are not changed when the trust pool feature is implemented.

Result

The router attains certificate authority interoperability, prioritizing trustpoint configuration and policies over the trust pool, thereby preserving established authentication behaviors during trust pool feature integration.

PKI certificate expiry notifications

Details the entire PKI certificate lifecycle, including certificate enrollment, renewal, revocation, and validation processes, with guidance on ensuring continuous authentication and robust network security.

A PKI certificate expiry notification is a trust infrastructure mechanism that

- alerts administrators when a public key infrastructure (PKI) certificate is approaching its expiry date
- uses SNMP traps and syslog messages to deliver notification events, and
- supports certificate authority interoperability in Cisco IOS XR deployments.

Additional reference information

PKI certificate expiry notifications help ensure proactive renewal of certificates, reducing the risk of service interruption or trust failures. Notifications are triggered based on approaching expiry dates and can be customized or integrated to support operational workflows.

When a PKI certificate is about to expire, the Cisco IOS XR device sends an SNMP trap and a syslog message to the configured monitoring system.

Counter-example

If the certificate is revoked or disabled, no expiry notification is sent; the expiring certificate must still be valid for notification to be triggered.

PKI alert notifications

Explains PKI alert notifications for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A PKI alert notification is a certificate authority interoperability component that

- provides timely alerts when certificates used by Cisco IOS XR applications are nearing expiry
- supports certificate-based authentication and trust for routers, and
- enhances operational awareness by notifying administrators through syslog and SNMP traps, ensuring seamless service continuity.
- PKI traps retrieve certificate information from devices in the network and send SNMP traps at regular intervals to the network management system, based on thresholds configured on the device.
- An SNMP trap (certificate expiry notification) is sent to the SNMP server at intervals starting from 60 days to one week before the certificate end date.

Additional reference information

Certificates are essential for authenticating routers in Cisco IOS XR deployments. If a certificate expires, it becomes invalid and can disrupt services such as Crosswork Trust Insights, Internet Key Exchange version 2 (IKEv2), dot1x, and others. PKI alert notifications ensure administrators are informed of impending certificate expiry, reducing risk to critical network functions.

Alert notifications are sent in the following modes and intervals:

- First notification: Sent 60 days before certificate expiry; warning mode.
- Repeated notifications: Sent every week until one week before expiry; warning mode.
- Last notification: Sent every day within the final week before expiry; alert mode.

The notifications include the following information:

- Certificate serial number
- Certificate issuer name
- Trustpoint name
- Certificate type
- Number of days remaining for the certificate to expire
- Certificate subject name

Sample syslog message

```
%SECURITY-CEPKI-1-CERT_EXPIRING_ALERT : Certificate expiring WITHIN A WEEK.
Trustpoint Name= check, Certificate Type= ID, Serial Number= 02:EC,
Issuer Name= CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN, Subject name=
CN=cisco.com,
Time Left= 1 days, 23 hours, 59 minutes, 41 seconds
```

Notification Intervals and Modes

Interval	Description	Notification Mode
First notification	Sent 60 days before certificate expiry	Warning
Repeated notifications	Sent weekly until one week before expiry	Warning
Last notification	Sent daily within the final week before expiry	Alert

PKI credentials expiry alerts

PKI alert notifications are not sent for:

- Secure Unique Device Identifier (SUDI) certificates
- Certificates in a trustpool (trustpools have their own expiry alerts mechanism)
- Trustpoint clones
- Certificate authority (CA) certificates without an associated router certificate
- Certificates with key usage keys

Requirement: Follow PKI credential expiry alert restrictions

Outlines the requirements and restrictions for PKI credential expiry alerts to keep certificate authority deployments supported.

To ensure PKI credential expiry alert restrictions remain supported:

- You cannot disable this feature, and it requires no additional configuration tasks.

- To enable PKI traps, use the command **snmp-server traps pki**. If SNMP is already configured, the SNMP trap will use the same PKI expiry timer.
- Verification: Run **show runn snmp-server traps** to confirm that PKI traps are enabled.

Regenerate PKI certificates

Configure PKI certificate regeneration to ensure continued certificate authority interoperability and security after certificate expiry.

Regenerate PKI certificates to maintain secure authentication and enable certificate authority interoperability when an existing certificate expires.

When a PKI certificate expires, it becomes invalid and must be regenerated to keep secure communications functioning. You will usually receive an expiry notification, prompting this task. Regenerating certificates ensures ongoing trust between network devices and certificate authorities.

Before you begin

- Review certificate authority prerequisites and ensure you understand any site-specific requirements.
- Confirm you have access to your router and know the trustpoint name used for PKI certificates.

Procedure

1. Clear the existing certificate connected to the trustpoint.

   Example

   ```
   Router# clear crypto ca certificates [trustpoint-name]
   ```

   Example

   ```
   Router# clear crypto ca certificates myca
   ```

   Example

   ```
   Clear the existing certificate using the following command:
   ```

   For example,

2. (Recommended) Regenerate a new keypair for the configured trustpoint label.

   Example

   ```
   Router# crypto key generate rsa [keypair-label]
   ```

   Example

   ```
   Router# crypto key generate rsa mykey
   The name for the keys will be: mykey
   % You already have keys defined for mykey
   Do you really want to replace them? [yes/no]: yes
     Choose the size of the key modulus in the range of 512 to 4096 for your
   General Purpose Keypair. Choosing a key modulus greater than 512 may take a
   ```

```
few minutes.

How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]The name for the keys will be: mykey
% You already have keys defined for mykey
Do you really want to replace them? [yes/no]: yes
  Choose the size of the key modulus in the range of 512 to 4096 for your
General Purpose Keypair. Choosing a key modulus greater than 512 may take a
few minutes.

How many bits in the modulus [2048]:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

If prompted to replace existing keys or select key size, accept defaults or enter the required values.

For example,

3. Reenroll the certificate.

Example

```
Router# crypto ca authenticate [trustpoint-name]
Router# crypto ca enroll [trustpoint-name]
```

Example

```
Router# crypto ca authenticate myca
Router# crypto ca enroll myca
```

Reenroll the certificate using the following command. For more information, see *Obtain router certificates for your router* section.

PKI certificates are successfully regenerated and the device resumes secure connectivity with the certificate authority.

Automatic PKI certificate renewal

Explains automatic PKI certificate renewal and auto-enroll behavior for Cisco IOS XR trustpoint certificates.

Automatic PKI certificate renewal is a Cisco IOS XR capability that

- enables a router to request a new PKI certificate before the current certificate expires
- eliminates the need for manual replacement of certificates, and
- helps avoid interruptions to encrypted communications such as MACsec session flaps due to certificate expiry.
- Public key infrastructure (PKI) controls the digital certificates used to authenticate a router and protect sensitive information flowing through a network.
- PKI certificates can have a short validity period and otherwise require manual replacement before expiration.

Feature history
Table 47: Feature History Table

Feature Name	Release Information	Feature Description
Automatic renewal of Public Key Infrastructure (PKI) certificate	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Automatic renewal of Public Key Infrastructure (PKI) certificate	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Automatic renewal of Public Key Infrastructure (PKI) certificate	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Automatic renewal of Public Key Infrastructure (PKI) certificate	Release 7.5.3	You can now enable the router to renew the PKI certificate from the Certificate Authority (CA) by configuring the percentage of the certificate validity, after which the router requests a new certificate from the CA, and the CA authorizes it before certification expiration. This feature eliminates the previously needed manual efforts of certification renewal and avoids interruptions such as MACsec session flaps due to certificate expiry and so on. This feature introduces the following commands: • **auto-enroll** • **renewal-message-type**

Additional reference information

- The auto-enroll setting defines the certificate-validity percentage after which the router requests a new certificate from the certification authority (CA).
- PKI encrypts and decrypts data using a public key and a private key pair that it generates. A PKI digital certificate authenticates the identity of a router.
- You can configure a timeline for PKI certificate renewal by specifying the percentage of certificate validity after which the router requests a new certificate from the CA server. This timeline for automatic PKI certificate renewal is called auto-enroll.

How automatic PKI certificate renewal works

Describes the auto-enroll renewal workflow that lets a Cisco IOS XR router request and install a new PKI certificate before the current certificate expires.

Automatic PKI certificate renewal starts when a router has auto-enroll configured for a trustpoint and the configured percentage of certificate validity has elapsed.

Summary

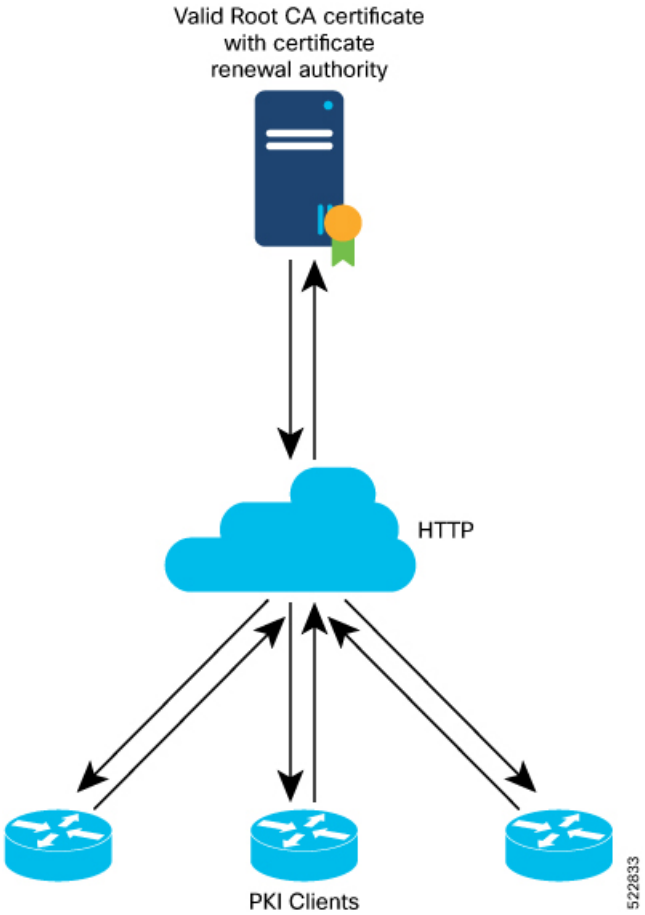
The key components involved in the process are:

- Router: Monitors the auto-enroll timeline and generates certificate signing requests when renewal is needed.
- Certification authority server: Receives renewal requests, verifies authorization, and creates the newly signed certificate.
- Auto-enroll configuration: Specifies the percentage of certificate validity after which renewal begins.

Automatic PKI certificate renewal ensures that a Cisco IOS XR router with auto-enroll configured can obtain and install a new PKI certificate before the existing certificate expires, maintaining uninterrupted secure network operations.

Workflow

Figure 7: Automatic renewal of Public Key Infrastructure (PKI) certificate



These stages describe how automatic PKI certificate renewal works.

1. The router reaches the configured auto-enroll point as the PKI certificate approaches its expiry date.
2. The router generates a certificate signing request.
3. The router sends the certificate signing request to the certification authority (CA) server using Simple Certificate Management Protocol (SCMP).
4. The CA server creates and signs the new certificate, authorizing it before the current certificate expires.
5. The router installs the new certificate, replacing the old one.
6. Automatic renewal ensures there is no disruption to data flows in the network.

Result

The router successfully installs a newly signed PKI certificate before the previous certificate expires, maintaining secure communication and avoiding service interruption.

Requirement: Meet automatic PKI certificate renewal prerequisites

Outlines the requirements and restrictions needed to support automatic PKI certificate renewal for certificate authority deployments.

To ensure automatic PKI certificate renewal is supported, meet the following requirements:

- Ensure that the certificate authority (CA) has a valid certificate.
- Confirm that the CA supports certificate renewal.

- Configure a trustpoint in the router using the **crypto ca trustpoint** command.
- A trustpoint must be authenticated before enrollment. It is authenticated when it has a CA certificate and enrolled when it has a router certificate.
- Ensure communications between the PKI client and the CA server use the HTTP protocol; the enrollment URL must be an HTTP URL.

Requirement: Automatic PKI certificate renewal guidelines

Outlines the requirements, restrictions, and guidelines you must follow to keep your PKI certificate authority deployment supported.

To maintain a supported PKI certificate authority deployment, follow these requirements:

- PKI certificates must be signed using the RSA algorithm only.
- If you configure the auto-enroll option under the trustpoint after the renewal timer for a PKI certificate has started, this configuration takes effect for the next renewal cycle—not the current one. The same condition applies to configuring the no auto-enroll option.
- The auto-enroll percentage can range between 1 and 99.
- The certificate renewal process requires the serial number and IP address values in the trustpoint. If these values are available in the trustpoint, the renewal process obtains them from there; otherwise, the router CLI prompts you to configure them during trustpoint enrollment.
- By default, PKI uses PKCS requests for automatic certificate renewal. You can also configure the router to use a Renewal request by executing the **renewal-message-type renewalreq** command.
- If the CA server cannot address certificate renewal requests, it instructs the router to poll the renewal request. In this case, the router retries for 10 minutes with a gap of 1 minute between each request if a certificate renewal attempt fails. You can configure these values with the **enrollment retry count** and **enrollment retry period** commands.

Configure automatic PKI certificate renewal

Configure automatic PKI certificate renewal for a trustpoint to ensure certificates are renewed automatically, improving interoperability with certificate authorities.

Enable automatic renewal of PKI certificates so your device maintains valid credentials for secure communication.

Use automatic PKI certificate renewal to avoid manual certificate management and ensure the device consistently has valid certificates from the certificate authority (CA).

Before you begin

- Review CA requirements and prerequisites for certificate renewal.
- Confirm you have an existing trustpoint configured.
- Have access to the device CLI.

Procedure

1. Configure the trustpoint for certificate enrollment.

Example

```
Router# configure
Router(config)# crypto ca trustpoint test
Router(config-trustp)# enrollment url http://frog.phoobin.com
Router(config-trustp)# subject-name OU=Spiral Dept., O=tiedye.com
Router(config-trustp)# auto-enroll 30
Router(config-trustp)# commit
```

2. (Optional) To disable automatic renewal.

Example

```
Router# configure
Router(config)# no auto-enroll
Router(config-trustp)# commit
```

3. Review the running configuration.

Example

```
Router# show running-config crypto ca trustpoint test
crypto ca trustpoint test
 enrollment url http://frog.phoobin.com
 auto-enroll 30
!
```

PKI certificate renewal is automated. The device will renew certificates at the specified interval without manual intervention.

11 Crosswork Trust Insights and Public-Key Systems

Topics:

- How Cisco IOS XR and Crosswork Trust Insights integrate
- Ed25519 public-key signatures
- Public key-pairs

Outlines the use of Crosswork Trust Insights and the implementation of public-key systems, summarizing best practices, configuration steps, and verification processes to enhance secure network operations.

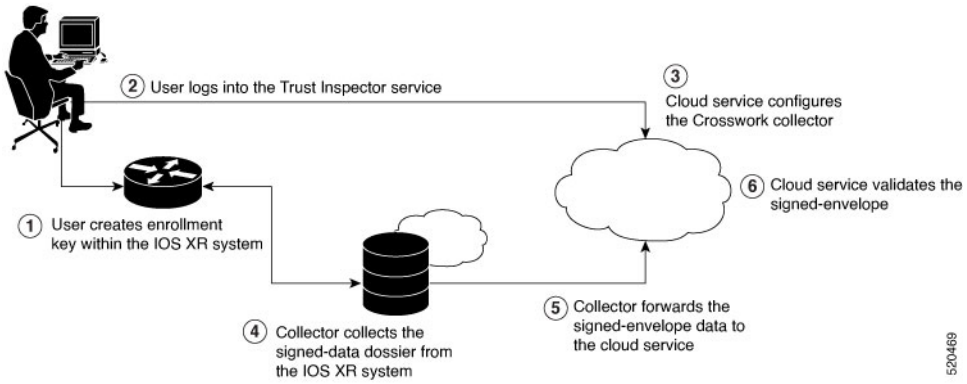
How Cisco IOS XR and Crosswork Trust Insights integrate

Introduces the key features and concepts of Crosswork Trust Insights and public-key systems, outlining their roles in securing network infrastructure, and detailing configuration, management, and verification procedures for effective deployment.

Cisco IOS XR Software provides a framework to securely enroll devices and share system integrity information with Cisco Crosswork Trust Insights, a cloud-based SaaS solution that tracks trust posture and validates network hardware and software integrity.

For more details, see:

- Cisco Crosswork Trust Insights
- Cisco Crosswork Trust Insights Data Sheet



Summary

The key components involved in the process are:

- Cisco IOS XR platform: Generates key pairs and trust roots for secure communication.
- Trust Inspector service: Manages device enrollment and receives required credentials for integration.
- Crosswork collector: Collects signed-data dossiers from IOS XR instances and forwards them for validation.
- Crosswork cloud service: Validates signed-data, ensuring trust posture and certificate authority interoperability.

Integrating Cisco IOS XR and Crosswork Trust Insights helps ensure certificate authority interoperability between network hardware and cloud services by enabling secure enrollment and signed-data sharing.

Workflow

These stages describe integrating Cisco IOS XR and Crosswork Trust Insights.

1. The Cisco IOS XR platform generates a new key pair and trust root using IOS XR commands.
2. The user logs into Trust Inspector, initiates the enrollment workflow, and provides the device's management IP, credentials, and certificate root.
3. Trust Inspector configures the Crosswork collector to log in to the router and pull integration data.
4. The Crosswork collector begins periodic polling and generates signed-information dossiers from each IOS XR instance.
5. The collector forwards signed-envelope data to the Crosswork cloud service for validation.
6. The cloud service validates the signed-envelope against the enrolled certificate or trust chain.

Result

The integration achieves certificate authority interoperability, allowing network devices and cloud services to securely verify system integrity and trust posture.

Configure Cisco IOS XR and Crosswork Trust Insights integration

Configure Cisco IOS XR and Crosswork Trust Insights integration by following key setup, enrollment, and verification procedures to ensure certificate authority interoperability.

Set up interoperability between Cisco IOS XR devices and Crosswork Trust Insights by enrolling systems, configuring authorization, and verifying data-signing.

Integration requires enrolling systems, generating key pairs, configuring certificate trustpoints, and collecting data dossiers. Proper setup enables authorized access and confirms certificate authority behavior.

Before you begin

- Check for IOS XR Software Maintenance Updates (SMUs) related to Crosswork Trust Insights. See, Cisco Software Downloads.
- Review Cisco IOS XR Release Notes.
- Ensure the following configurations are present on the IOS XR device:
  - User authorization required to collect the signed-data dossier
  - SSH server configuration
  - Netconf server configuration
  - Domain name configuration, which is required for certification enrollment

Procedure

1. Review configuration example for user authorization.

Example

```
Router# configure
Router(config)# taskgroup alltasks-dossier
Router(config-tg)# task read sysmgr
Router(config-tg)# task read system
Router(config-tg)# task read pkg-mgmt
Router(config-tg)# task read basic-services
Router(config-tg)# task read config-services
Router(config-tg)# task execute dossier
Router(config-tg)# commit
```

Example

```
Router# configure
Router(config)# usergroup dossier-group
Router(config-ug)# taskgroup alltasks-dossier
Router(config-ug)# commit
```

Example

```
Router# configure
Router(config)# username dossier-user
Router(config-un)#group dossier-group
Router(config-un)# commit
```

You must have the required user access privileges in order to collect the data dossier from the system. This is defined in terms of IOS XR Task IDs for each command.

For the respective Task ID applicable for each data dossier option and for the signed-envelope, see the Task ID section in the Command Reference page of **show platform security integrity dossier** command and **utility sign** command.

> **Note**
>
> We recommend that you use the task execute dossierto configure a CTI (customer-define) user, who collects dossier from the system.

Listed below are the configurations to set up a user with sufficient authorization to collect all the signed-data dossier. You can configure customized task groups, then associate those task groups with user groups, and finally associate the user groups with the user.

2. Review configuration example for for ssh and netconf.

Example

```
Router# configure
Router(config)# ssh server v2
Router(config)# ssh server vrf default
Router(config)# ssh server netconf vrf default
Router(config)# netconf-yang agent
Router(config-ncy-agent)# ssh
Router(config-ncy-agent)# exit
Router(config)# domain name example.com
Router(config)# commit
```

3. Review the running configuration.

Example

```
ssh server v2
ssh server vrf default
ssh server netconf vrf default
!
netconf-yang agent
 ssh
!
domain name example.com
```

4. Configure the feature by using the source example.

Example

```
Router# configure
Router(config)# ssh server rate-limit 600
Router(config)#line default
Router(config-line)#exec-timeout 0 0
Router(config-line)#session-timeout 0
Router(config-line)# commit
```

While the dossier is collected from a device through SSH, the SSH session might timeout. Also, multiple ssh sessions to a device can result in the denial of some SSH sessions. To avoid such occurrence, the following configuration is recommended on the device:

5. Review the running configuration.

Example

```
ssh server rate-limit 600
!
line default
 exec-timeout 0 0
 session-timeout 0
!
```

Integration is successful when the device configuration and verification steps produce output matching the required certificate interoperability and user authorization as demonstrated in the configuration examples.

Configure Crosswork Trust Insights key pairs

Configure Crosswork Trust Insights key pairs to enable secure certificate authority interoperability.

Establish secure key pairs used for certificate authority operations within Crosswork Trust Insights.

To enroll a system running Cisco IOS XR, you need to generate keys and certificates for both the leaf and root nodes. The system supports a two-tier self-signed certificate chain for re-keying without re-enrollment, enhancing interoperability and compliance with cryptographic standards.

You can use the **system-root-key** and **system-enroll-key** options in the **crypto key generate** command to generate the root key and the enrollment key respectively, for all the hashing algorithms. You can do this for hashing algorithms such as RSA, DSA or ECDSA (including ECDSA nistp384 and ECDSA nistp521).

To delete the RSA keys, use the no form: no crypto key generate rsa

The details of RSA and DSA keys are displayed in the running configuration.

Starting with Cisco IOS XR Release 26.1.1, the default RSA key size is 3072 bits; any smaller key is considered weak, triggering a syslog warning during system boot. This enhancement ensures compliance with current cryptographic security standards and enhance device protection against emerging threats.

Before you begin

Review certificate authority prerequisites.

Procedure

---

1. Generate the root key pair.

   Choose a key modulus of 3072 bits or higher for security.

Example

```
Router# crypto key generate rsa system-root-key

Sun Oct 20 13:05:26.657 UTC
The name for the keys will be: system-root-key
Choose the size of the key modulus in the range of 512 to 4096 for your General
 Purpose Keypair. Choosing a key modulus greater than 512 may take a few
minutes.
How many bits in the modulus 3072:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

2. Generate the enrollment key pair.

   Select the recommended modulus (3072 bits or higher).

   Example

```
Router# crypto key generate rsa system-enroll-key

Sun Oct 20 13:05:40.370 UTC
The name for the keys will be: system-enroll-key
Choose the size of the key modulus in the range of 512 to 4096 for your General
 Purpose Keypair. Choosing a key modulus greater than 512 may take a few
minutes.
 How many bits in the modulus 3072:
Generating RSA keys ...
Done w/ crypto generate keypair
[OK]
```

3. Verify the key pairs.

   Example

```
Router# show  crypto  key mypubkey rsa | begin system-

Fri Mar 27 14:00:20.954 IST
Key label: system-root-key
Type     : RSA General purpose
Size     : 3072
Created  : 01:13:10 IST Thu Feb 06 2025
Data     :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 00A93DE0 1E485EE3 0E7F0964 C48361D1 B6014BE7 A303D8D6 F7790E92 88E69C4B
 B97B7A9C D1B277E3 1569093C 82BD3258 7F67FB49 94860ECD 34498F1F 59B45757
 F32C8E8F 7CEE23EC C36A43D1 9F85C0D9 B96A14DD DD3BBD4C A1FB0888 EED210A7
 39D9A403 7ACE0F6E 39107226 CA621AD8 6E8102CA 9761B86F D33F2871 9DD16559
 AFCB4729 EFCEDBAF 83DF76E4 9A439844 EE3B1180 4022F575 99E11A2C E25BB23D
 9DD74C81 4E5C1345 D9E3CC79 1B98B1AA 6C06F004 22B901EC 36C099FE 10DE2622
 EB7CE618 9A555769 12D94C90 D9BEE5EA A664E7F6 4DF8D8D4 FE7EAB07 1EF4FEAB
 22D9E55F 62BA66A0 72153CEC 81F2639F B5F2B5C5 25E10364 19387C6B E8DB8990
 11020301 0001

Key label: system-enroll-key
Type     : RSA General purpose
Size     : 3072
Created  : 01:13:16 IST Thu Feb 06 2025
Data     :
```

```
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 009DBC14 C83604E4 EB3D3CF8 5BA7FDDB 80F7E85B 427332D8 BBF80148 F0A9C281
 49F87D5C 0CEBA532 EBE797C5 7F174C69 0735D13A 493670CB 63B04A12 4BCA7134
 EE0031E9 047CAA1E 802030C5 6071E8C2 F8ECE002 CC3B54E7 5FD24E5C 61B7B7B0
 68FA2EFA 0B83799F 77AE4621 435D9DFF 1D713108 37B614D3 255020F9 09CD32E8
 82B07CD7 01A53896 6DD92B5D 5119597C 98D394E9 DBD1ABAF 6DE949FE 4A8BF1E7
 851EB3F4 60B1114A 1456723E 063E50C4 2D410906 BDB7590B F1D58480 F3FA911A
 6C9CD02A 58E68D04 E94C098F 0F0E81DB 76B40C55 64603499 2AC0547A D652412A
 BCBBF69F 76B351EE 9B2DF79D E490C0F6 92D1BB97 B905F33B FAB53C20 DDE2BB22
 C7020301 0001
```

Example

```
Router(config)# crypto key generate rsa test
Router(config)# commit
Thu May 12 08:37:59.894 UTC
Router(config)# end
Router# show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2025 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nIbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOIycjsizI/TLBz9WcQX.rmxVwkNgIKAnRCUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 3072 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

You can use the **show crypto key mypubkey rsa** command to verify the above key pair generation.

You can also view the RSA keys in the running configuration. The keys in the following example are in OpenSSL format:

> **Note**
>
> Only those keys that are generated in the config mode are visible in the running configuration.

Crosswork Trust Insights key pairs are configured. Verification output matches the expected certificate entries, confirming interoperability.

Configure system trustpoint certificates

Configure system trustpoint certificates to establish certificate authority interoperability and enable secure root and leaf certificate functionality.

Set up system trustpoint certificates to ensure certificate authority interoperability and proper cryptographic verification within your network device.

System trustpoint certificates are required for secure authentication between network devices and their certificate authority. This task generates and verifies the root and leaf certificates.

Before you begin

- Review certificate authority requirements for your device.
- Identify the domain name and certificate information for your specific environment.
- Have administrative access to the network device's CLI.

Review the related certificate authority prerequisites before you begin.

Procedure

1. Define the domain name and create the system trustpoint.

   Example

```
Router# configure
Router(config)# domain name domain1
Router(config)# crypto ca trustpoint system-trustpoint
Router(config)# keypair rsa system-enroll-key
Router(config)# ca-keypair rsa system-root-key
Router(config)# subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco
systems,OU=ASR
Router(config)# subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San
Jose,O=cisco systems,OU=ASR
Router(config)# enrollment url self
Router(config)# key-usage certificate digitalsignature keyagreement
dataencipherment
Router(config)# lifetime certificate 300
Router(config)# message-digest sha256
Router(config)# key-usage ca-certificate digitalsignature keycertsign crlsign
Router(config)# lifetime ca-certificate 367
Router(config)# commit
```

2. Generate the root and leaf RSA keypairs.

Example

```
Router(config)# keypair rsa system-enroll-key
Router(config)# ca-keypair rsa system-root-key
Router(config)# subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco
systems,OU=ASR
Router(config)# subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San
Jose,O=cisco systems,OU=ASR
Router(config)# enrollment url self
Router(config)# key-usage certificate digitalsignature keyagreement
dataencipherment
Router(config)# lifetime certificate 300
Router(config)# message-digest sha256
Router(config)# key-usage ca-certificate digitalsignature keycertsign crlsign
Router(config)# lifetime ca-certificate 367
Router(config)# commit
```

3. Set certificate subject names for each.

Example

```
Router(config)# subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco
systems,OU=ASR
Router(config)# subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San
Jose,O=cisco systems,OU=ASR
Router(config)# enrollment url self
Router(config)# key-usage certificate digitalsignature keyagreement
dataencipherment
Router(config)# lifetime certificate 300
Router(config)# message-digest sha256
Router(config)# key-usage ca-certificate digitalsignature keycertsign crlsign
Router(config)# lifetime ca-certificate 367
Router(config)# commit
```

4. Configure enrollment URL and specify certificate attributes.

Example

```
Router(config)# enrollment url self
Router(config)# key-usage certificate digitalsignature keyagreement
dataencipherment
Router(config)# lifetime certificate 300
Router(config)# message-digest sha256
Router(config)# key-usage ca-certificate digitalsignature keycertsign crlsign
Router(config)# lifetime ca-certificate 367
Router(config)# commit
```

5. Review the running configuration.

Example

```
config
domain name domain1
crypto ca trustpoint system-trustpoint
keypair rsa system-enroll-key
ca-keypair rsa system-root-key
subject-name CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
subject-name ca-certificate CN=lab1-ca,C=US,ST=CA,L=San Jose,O=cisco
```

```
systems,OU=ASR
enrollment url self
key-usage certificate digitalsignature keyagreement dataencipherment
lifetime certificate 300
message-digest sha256
key-usage ca-certificate digitalsignature keycertsign crlsign
lifetime ca-certificate 367
!
```

System trustpoint certificates are successfully configured when the **show running-config** output matches your intended settings, confirming certificate authority interoperability.

Configure root and leaf certificates

Configure root and leaf certificates to establish certificate authority interoperability and verify successful configuration.

Generate and verify root and leaf certificates for secure authentication.

Root and leaf certificates are essential for enabling certificate-based authentication in network environments. The root certificate is self-signed and acts as the certificate authority, while the leaf certificate is signed by the root certificate.

The root certificate is self-signed. The root certificate signs the leaf certificate.

Before you begin

- Review all prerequisites relevant to the certificate authority.
- Ensure you have access to the necessary router commands and credentials.

Procedure

1. Generate the root certificate.

   Example

```
Router# crypto ca authenticate system-trustpoint

Sun Oct 20 13:07:24.136 UTC
% The subject name in the certificate will include: CN=lab1
ca,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
% The subject name in the certificate will include: ios.cisco.com
Serial Number  : 0B:62
Subject:
serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco
systems,L=San Jose,ST=CA,C=US,CN=lab1-ca
Issued By      :
       serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco
systems,L=San Jose,ST=CA,C=US,CN=lab1-ca
  Validity Start : 13:07:26 UTC Sun Oct 20 2019
  Validity End   : 13:07:26 UTC Wed Oct 21 2020
  SHA1 Fingerprint:
         9DD50A6B24FEBC1DDEE40CD2B4D99A829F260967
```

2. Generate the leaf certificate.

Example

```
Router# crypto ca enroll system-trustpoint

Sun Oct 20 13:07:45.593 UTC
% The subject name in the certificate will include:
CN=lab1-ads,C=US,ST=CA,L=San Jose,O=cisco systems,OU=ASR
% The subject name in the certificate will include: ios.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: c44a11fc
% Include an IP address in the subject name? [yes/no]: no
Certificate keypair configured Type: 1, Label: system-enroll-key.Leaf cert
key usage string: critical,digitalSignature,keyEncipherment,keyAgreement.
Serial Number  : 0B:63
  Subject:
        serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco
systems,L=San Jose,ST=CA,C=US,CN=lab1-ads
  Issued By      :
        serialNumber=c44a11fc,unstructuredName=ios.cisco.com,OU=ASR,O=cisco
systems,L=San Jose,ST=CA,C=US,CN=lab1-ca
  Validity Start : 13:07:47 UTC Sun Oct 20 2019
  Validity End   : 13:07:47 UTC Sat Aug 15 2020
  SHA1 Fingerprint:
         19D4C40F9EFF8FF25B59DE0161BA6C0706DC9E3A
```

3. Verify the certificates.

Example

```
Router# show crypto ca certificates system-trustpoint
Fri Mar 27 14:00:51.037 IST

Trustpoint       : system-trustpoint
==================================================
CA certificate
  Serial Number  : 10:B5
  Subject:
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
  Issued By      :
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
  Validity Start : 12:30:17 UTC Fri Feb 21 2020
  Validity End   : 12:30:17 UTC Sat Feb 20 2021
  SHA1 Fingerprint:
         9400A30816805219FAAA5B9C86C214E6F34CEF7B
Router certificate
  Key usage      : General Purpose
  Status         : Available
  Serial Number  : 10:B6
  Subject:

serialNumber=7b20faa4,unstructuredAddress=10.1.1.1,unstructuredName=test-sec1.cisco.com,CN=Anetwork,OU=IT,O=Spark
 Network,L=Rotterdam,ST=Zuid Holland,C=NL
  Issued By      :
        serialNumber=7b20faa4,unstructuredName=test-sec1.cisco.com
  Validity Start : 12:30:31 UTC Fri Feb 21 2020
  Validity End   : 12:30:31 UTC Sat Feb 20 2021
  SHA1 Fingerprint:
         21ACDD5EB6E6F4103E02C1BAB107AD86DDCDD1F3
Associated Trustpoint: system-trustpoint
```

You can use the **show crypto ca certificates system-trustpoint [detail]** command to see the details of generated root and leaf certificates:

Root and leaf certificates are correctly configured, and their output details match expected values. Certificate authority interoperability is verified.

System certificate expiry actions

Provides facts and actions for managing system certificate expiry, including notification behavior and command examples for certificate regeneration.

Certificate expiry notification

- IOS XR sends a notification to the syslog server when a system certificate is nearing expiry.
- Certificate Authority (CA) clients use these notifications for timely certificate management.

Certificate regeneration actions

To regenerate a system certificate on receiving an expiry notification:

- Clear the system trustpoint certificate
- Authenticate the system trustpoint
- Enroll the system trustpoint for a new certificate

```
Router# clear crypto ca certificates system-trustpoint
Router# crypto ca authenticate system-trustpoint
Router# crypto ca enroll system-trustpoint
```

Usage notes

- Apply these actions when planning, configuring, or validating certificate authority interoperability.
- For more information on certificate expiry alert notifications, see *PKI alert notification* section.
- For detailed steps on certificate regeneration, refer to *Regenerate the certificate* section.

Data dossier collection

Explains Cisco IOS XR data dossier collections, including command selectors, signed envelopes, filesystem inventory, incremental IMA, associated commands, and feature history.

A data dossier collection is a Cisco IOS XR security-integrity capability that

- collects data from IOS XR components in JSON format using the **show platform security integrity dossier** command
- enables output signing for integrity and authenticity using the **utility sign** command, and
- gathers package information, reboot and rollback history, system-integrity snapshots, system inventory, filesystem inventory, and incremental IMA data.

Signed envelopes

To verify the integrity and authenticity of data dossier output, a signature can be added to the data, which is provided in JSON format. The Secure Unique Device Identifier (SUDI) signature supplies a hardware root of trust to the dossier collected by the system.

Filesystem inventory

The metadata of the filesystem is collected through the data dossier function. The metadata includes timestamps for file creation, access, and modification. Snapshots are taken at each configured interval: the initial snapshot contains all files, and subsequent snapshots record incremental changes.

- The **snapshot-interval** is specified in 15-minute blocks. The interval ranges from 1 to 96. For example, a value of 2 indicates that a snapshot interval is collected every 30 minutes.
- The snapshots are stored in **/misc/scratch/filesysinv**.
- Logs are kept at **/var/log/iosxr/filesysinv/***.

Incremental integrity measurement architecture

Incremental IMA allows you to define the starting IMA sequence to include in a response, after which subsequent events are reported progressively. You can specify the start event, sequence number, and start time to selectively collect IMA events.

Feature history
Table 48: Feature History Table

Feature Name	Release Information	Description
Collect Filesystem Inventory	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Collect Filesystem Inventory	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Collect Filesystem Inventory	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Collect Filesystem Inventory	Release 7.3.1	With this feature, a snapshot of the filesystem metadata such as when the file was created, modified, or accessed is collected at each configured interval. In addition to displaying the changes that the file underwent as compared to the previous snapshot, the inventory helps in maintaining data integrity of all the files in the system.

Collect data dossier output from Cisco IOS XR components

Configure data dossier collection by creating a signed envelope, collecting reboot-history dossier output, collecting filesystem inventory, and retrieving incremental IMA data.

Collect and review data dossier output from Cisco IOS XR components, including reboot-history, filesystem inventory, and incremental IMA data.

Data dossier output is presented in JSON format. For information about data dossier selectors, signed envelopes, filesystem inventory, and incremental IMA commands, see the *Data dossier collection* reference.

Before you begin

Review prerequisites related to certificate authority configuration.

Procedure

1. Create a signed envelope for the data dossier output.

   Example

```
Router# show platform security integrity dossier include packages
reboot-history rollback-history system-integrity-snapshot system-inventory
nonce 1580 | utility sign nonce 1580 include-certificate
```

2. Collect reboot-history data dossier output.

   Example

```
Router# show platform security integrity dossier nonce 1234 include
reboot-history
Thu Feb 27 22:20:57.542 IST
{"collection-start-time":1582225757.60921,"model-name":"http://cisco.com/ns/yang/Cisco-IOS-XR-md","model-revision":"2019-08-05","license-udi":{"result-code":
 "Success", "license-udi": "UDI:
PID:NCS-5501-SE,SN:FOC2107R0ZB\n"},"version":{"result-code": "Success",
"version": "Cisco IOS XR Software, Version 7.0.12\nCopyright (c) 2013-2020
by Cisco Systems, Inc.\n\nBuild Information:\n Built By     : user1\n Built
On     : Mon Jan 27 01:36:26 PST 2020\n Built Host   : iox-lnx-076\n Workspace
    : /auto/srca
rchive15/prod/7.0.12/cisco8000/ws\n Version      : 7.0.12\n Location     :
/opt/cisco/XR/packages/\n Label        : 7.0.12\n\ncisco NCS-5500 ()
processor\nSystem uptime is 4 days 10 hours 12
minutes\n\n"},"platform":{"result-code": "Success", "platform": "Node
       Type                      State             Config
state\n--------------------------------------------------------------------------------\n0/RP0/CPU0
```

```
        NCS-5501-SE(Active)        IOS XR RUN        NSHUT\n0/RP0/NPU0
  Slice
             UP                \n0/FT0            NCS-1RU-FAN-FW
  OPERATIONAL       NSHUT\n0/FT1             NCS-1RU-FAN-FW
OPERATIONAL       NSHUT\n0/PM0             NCS-1100W-ACFW             FAILED
            NSHUT\n0/PM1             NCS-1100W-ACFW             OPERATIONAL

NSHUT\n0/PM1","reboot-history":{"result-code":"Success","model-name":"Cisco-IOS-XR-linux-os-reboot-history-oper","model-revision":"2019-04-05","nod
 "0/RP0/CPU0", "reboot-history": [{"reason": "
User initiated graceful reload", "time": "Wed Feb 19 15:25:11 2020",
"cause-code": 1, "no": 1}, {"reason": "CARD_SHUTDOWN", "time": "Wed Feb 19
16:38:00 2020", "cause-code": 37, "no": 2}, {"reason": "CARD_SHUTDOWN", "time":
 "Wed Feb 19 19:06:27 2020", "cause-code": 37, "no": 3}, {"reason":
"CARD_SHUTDOWN", "time": "Thu Feb 20 11:50:50 2020", "cause-code": 37, "no":
 4}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21 10:54:09 2020",
"cause-code": 37, "no": 5}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21
19:00:10 2020", "cause-code": 37, "no": 6}, {"reason": "CARD_SHUTDOWN", "time":
 "Sun Feb 23 12:05:25 2020", "cause-code": 37, "no": 7}]}, {"node-name":
"0/0/CPU0", "reboot-history": [{"reason": "Reboot triggered by install",
"time": "Tue Feb  4 19:59:23 2020", "cause-code": 36, "no": 1}, {"reason":
"CARD_SHUTDOWN", "time": "Tue Feb  4 20:12:06 2020", "cause-code": 37, "no":
 2}, {"reason": "Headless SDR", "time": "Sun Feb  9 17:45:25 2020",
"cause-code": 671088647, "no": 3}, {"reason": "User initiated graceful reload",
 "time": "Sun Feb  9 17:45:29 2020", "cause-code": 241, "no": 4}, {"reason":
 "CARD_SHUTDOWN", "time": "Sun Feb  9 18:28:25 2020", "cause-code": 37, "no":
 5}, {"reason": "Headless SDR", "time": "Sun Feb  9 19:01:55 2020",
"cause-code": 671088647, "no": 6}, {"reason": "Headless SDR", "time": "Wed
Feb 19 15:25:19 2020", "cause-code": 671088647, "no": 7}, {"reason":
"CARD_SHUTDOWN", "time": "Wed Feb 19 16:37:46 2020", "cause-code": 37, "no":
 8}, {"reason": "CARD_SHUTDOWN", "time": "Wed Feb 19 19:06:1
4 2020", "cause-code": 37, "no": 9}, {"reason": "CARD_SHUTDOWN", "time": "Thu
 Feb 20 11:50:37 2020", "cause-code": 37, "no": 10}, {"reason":
"CARD_SHUTDOWN", "time": "Fri Feb 21 10:54:01 2020", "cause-code": 37, "no":
 11}, {"reason": "CARD_SHUTDOWN", "time": "Fri Feb 21 18:59:57 2020",
"cause-code": 37, "no": 12}, {"reason": "CARD_SHUTDOWN", "time": "Sun Feb 23
 12:05:12 2020", "cause-code": 37, "no":
13}]}]},"collection-end-time":1582822260.296664}
Router#
```

3. Enable filesystem inventory collection.

   Example

   ```
   Router(config)# filesystem-inventory
   Router(config-filesystem-inventory)# snapshot-interval 2
   Router(config-filesystem-inventory)# commit
   ```

   The **snapshot-interval** value is the time interval in 15-minute blocks. The interval ranges from 1 to 96. A value of 2 indicates that a snapshot is collected every 30 minutes.

4. Retrieve filesystem inventory in JSON format.

   Example

   ```
   show platform security integrity dossier include filesystem-inventory | file
    <platform>-parent.json

   {"collection-start-time":1610168028.380901,
   "model-name":"http://cisco.com/ns/yang/Cisco-IOS-XR-ama",
   "model-revision":"2019-08-05","license-udi":{"result-code": "Success",
   ```

```
"license-udi":
"UDI: PID:NCS-55A1-24H,SN:FOC2104R15R\n"},"version":{"result-code": "Success",
"version": "Cisco IOS XR Software, Version 7.3.1
\nCopyright (c) 2013-2020 by Cisco Systems, Inc.\n\nBuild Information:\n
Built By     : <user>\n Built On     : Thu Jan  7 17:16:02 PST 2021\n
Built Host   : <host>\n Workspace    : <ws>
Version      : 7.3.1\n  Location     : /opt/cisco/XR/packages/\n Label
  : 7.3.1\n\ncisco
() processor\nSystem uptime is 8 hours 7
minutes\n\n"},"platform":{"result-code":
"Success", "platform":
"Node               Type                         State             Config state
--------------------------------------------------------------------------------
0/RP0/CPU0        <node-type>(Active)       IOS XR RUN        NSHUT\n
0/RP0/NPU0        Slice                      UP
0/RP0/NPU1        Slice                      UP
0/FT0             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT1             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
0/FT2             <platform>-A1-FAN-RV       OPERATIONAL       NSHUT
PM1               <platform>-1100W-ACRV      OPERATIONAL       NSHUT
"},
----------------------------Output is snipped for brevity
-------------------------------------
```

5. Limit the number of filesystem inventory snapshots.

   Example

   ```
   show platform security integrity dossier include filesystem-inventory
   filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 0, \"count\":
    1}}'
   ```

6. Start filesystem inventory collection from a new block.

   Example

   ```
   show platform security integrity dossier include filesystem-inventory
   filesystem-inventory-options '{\"0/RP0/CPU0\": {\"block_start\": 5}}'
   ```

7. Collect filesystem inventory data from a remote node.

   Example

   ```
   show platform security integrity dossier include filesystem-inventory
   filesystem-inventory-options '{\"0/RP1/CPU0\": {\"block_start\": 0}}' | file
    harddisk:PE1_remote.json
   ```

8. Retrieve incremental IMA data.

   Example

   ```
   show platform security integrity dossier incremental-ima
   "{\"ima_start\":[{\"0/RP0/CPU0\":{\"start_event\":1000,\"start_time\":\"Tue
   Feb 16 09:15:17 2021\"}}]}"
   ```

   The system reports subsequent events from the starting IMA sequence in the response.

The task is complete when the required data dossier output is collected, signed if needed, and stored or reviewed in JSON format.

Configure key generation and data-signing tests

Configure key generation and data-signing tests to verify certificate authority interoperability.

Set up and verify certificate interoperability by generating keys and signing data using the required procedures.

Perform this task to test key generation and data-signing with different key algorithms. Refer to the *How to Integrate Cisco IOS XR and Crosswork Trust Insights* section for specifics on integrating these technologies.

Before you begin

- Review the related certificate authority prerequisites.
- Ensure you have access to the required system trustpoint configuration details.

Review the related certificate authority prerequisites before you begin.

Procedure

1. Unconfigure the trustpoint.

   no crypto ca trustpoint system-trustpoint

2. Clear the certificates that were generated earlier.

   clear crypto ca certificates system-trustpoint

3. Generate new keys.
4. Configure the system trustpoint again.
5. Authenticate and enroll the system trustpoint to generate the certificates.

The task is complete when the configuration and verification output matches the source example, confirming the intended certificate authority interoperability behavior.

RPM package authenticity fingerprints

Explains how Cisco IOS XR verifies RPM package authenticity by comparing package fingerprint values with Known Good Values and by reporting install-time and run-time fingerprints through the YANG model output.

A fingerprint is a unique hex value that

- represents the state of an RPM package at install time
- monitors changes to package files after installation, and
- enables Cisco IOS XR to verify package authenticity against a Known Good Value (KGV).

- Known Good Value (KGV): The KGV is a reference fingerprint generated and published for each package after it is built by Cisco. It is used to determine whether an installed package matches the software released by Cisco.
- Install-time fingerprint: This hex value represents the software state of the package at installation. If the value matches the KGV, the RPM package is genuine and unmodified before installation. A mismatched fingerprint indicates a manipulated RPM.
- Run-time fingerprint: This hex value represents the running state of installed package files. If it matches the install-time fingerprint, no changes have occurred since installation. If the values differ, files in the package have been modified post-installation—either inadvertently or maliciously. A value of "0" means no run-time fingerprint is available and helps monitor changes to running software over time.

In the example, both the install time and run time fingerprints are the same.

The fingerprint generation status is used to indicate how up-to-date the run time fingerprints are. This may indicate that generation is currently in progress and will complete shortly, or generation is awaiting the end of an atomic change.

Feature history
Table 49: Feature History Table

Feature Name	Release Information	Description
Verify Authenticity of RPM Packages Using Fingerprint	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Verify Authenticity of RPM Packages Using Fingerprint	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Verify Authenticity of RPM Packages Using Fingerprint	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Description
Verify Authenticity of RPM Packages Using Fingerprint	Release 7.3.1	This feature helps in verifying the authenticity of an installable package using fingerprint values. The fingerprint value of the package is compared with a point of reference called Known Good Value (KGV). The KGV for an image or package is generated after it is built by Cisco. After installing the package, the associated install time and build time fingerprint values are compared using Yang RPC to determine whether the package is genuine. A match in the fingerprints indicates that the package published on CCO and that installed on router are the same.

Additional reference information

Cisco IOS XR, Release 7.3.1, introduced a fingerprint mechanism to verify the authenticity of packages released by Cisco. This mechanism helps determine if installed software matches what Cisco published. Cisco Crosswork applications monitor the installed software over time to flag potential issues.

Security measures like GPG and IMA signing are used for package installation, but fingerprints provide additional visibility to detect changes and anomalies.

Install-time and run-time fingerprints are reported exclusively in the YANG model output; there are no CLI commands to display these values.

Below is a typical YANG model output showing RPM package fingerprint status and values:

```
Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:97f5bc36-0eb0-4d2f-9c6f-3d34fea14be0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
<data>
    <install xmlns="http://cisco.com/ns/yang/Cisco-IOS
XR-spirit-install-instmgr-oper">
      <packages>
        <active>
         <summary>
         <rpm-fingerprint-status>generation-up-to-date</rpm-fingerprint-status>

            <rpm-fingerprint-timestamp>Mon Jun 15 15:58:22
2020</rpm-fingerprint-timestamp>
            <package>
             <name>asr9k-xr</name>
              <version>7.3.1</version>
              <release>r731</release>
             <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
              <rpm-fingerprint>

<rpm-fingerprint-install-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-install-time>
```

```
<rpm-fingerprint-run-time>2871bf68d3cd764938775afc9e5a69c130f9fbde</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>

            <package>
             <name>asr9k-mcast-x64</name>
             <version>192.0.2.10</version>
             <release>r731</release>
             <gpg-key-id>ddcead3dcb38048d</gpg-key-id>
              <rpm-fingerprint>

<rpm-fingerprint-install-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-install-time>


<rpm-fingerprint-run-time>3ddca55bc00a0ce2c2e52277919d398621616b28</rpm-fingerprint-run-time>

              </rpm-fingerprint>
             </package>
------------------------------ Truncated for brevity
---------------------------------
```

In this example, both install-time and run-time fingerprints are identical, indicating the package files have not been altered since installation. The fingerprint generation status shows how current the values are and may indicate synchronization or pending changes.

Attribute	Install-time fingerprint	Run-time fingerprint
Represents	State at installation	State of running files
Comparison target	Known Good Value (KGV)	Install-time fingerprint
Change detection	None before install	Changes after install
Value when unavailable	Fingerprint value present	"0"
Display location	YANG model output	YANG model output

Counter-example

If the run-time fingerprint differs from the install-time fingerprint (or from the KGV), that signals the RPM package was modified after installation and may require further investigation.

Ed25519 public-key signatures

Explains Ed25519 public-key signature support in Cisco IOS XR certificate-based trust, covering source behavior, constraints, and operational details for certification authority interoperability.

A Ed25519 public-key signature is a cryptographic algorithm that

- uses elliptic curve cryptography to provide enhanced security and faster performance compared to other signature algorithms
- enables certificate-based trust and interoperability with certificate authorities in Cisco IOS XR deployments, and
- allows integration with Cisco Crosswork Trust Insights through key generation and management on 64-bit platforms.

Ed25519 keys can be generated with an empty label or predefined labels, such as **system-root-key** and **system-enroll-key**. When an empty label is used, the system assigns a default label. Predefined labels facilitate integration with Cisco Crosswork Trust Insights.

Feature history
Table 50: Feature History Table

Feature Name	Release Information	Feature Description
Support for Ed25519 Public-Key Signature System	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.
Support for Ed25519 Public-Key Signature System	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M
Support for Ed25519 Public-Key Signature System	Release 7.3.1	This feature allows you to generate and securely store crypto key pair for the Ed25519 public-key signature algorithm on Cisco IOS XR 64-bit platforms. This signature system provides fast signing, fast key generation, fool proof session keys, collision resilience, and small signatures. The feature also facilitates integration of Cisco IOS XR with Cisco Crosswork Trust Insights. Commands introduced for this feature are: • crypto key generate ed25519 • crypto key zeroize ed25519 • show crypto key mypubkey ed25519 Commands modified for this feature are: • ca-keypair • keypair

Configure Ed25519 crypto keys

Configure Ed25519 crypto keys to enable certificate authority interoperability and enhance cryptographic security.

Generate and manage Ed25519 crypto keys on your device to support certificate authority interoperability and ensure cryptographic integrity.

Ed25519 crypto keys provide high-performance digital signature capability and are interoperable with certificate authority requirements. You can generate and delete these keys from XR EXEC or XR Config mode.

Before you begin

- Review the related certificate authority prerequisites for your deployment.
- Ensure you have access to XR EXEC or XR Config mode on your device.

Procedure

1. Generate an Ed25519 crypto key.

   Example

   ```
   Router# crypto key generate ed25519
   ```

   To generate the Ed25519 crypto key, use the **crypto key generate ed25519** command in XR EXEC mode or XR Config mode.

   To delete the Ed25519 crypto key with default label or any predefined label, use the **crypto key zeroize ed25519** command in XR EXEC mode.

   > **Note**
   >
   > From Cisco IOS XR Release 7.3.2 onwards, you can generate and delete key-pairs from XR Config mode, as well. For more details, see Public key-pairs.

2. Verify the configuration.

   Example

   ```
   Router# show crypto key mypubkey  ed25519

   Mon Nov 30 07:05:06.532 UTC
   Key label: the_default
   Type : ED25519
   Size : 256
   Created : 07:03:17 UTC Mon Nov 30 2020
   Data :
   FF0ED4E7 71531B3D 9ED72C48 3F79EC59 9EFECCC3 46A129B2 FAAA12DD EE9D0351
   ```

   Use the **show crypto key mypubkey ed25519** command to view all Ed25519 crypto keys generated on the system.

The Ed25519 crypto keys are successfully generated and verified. You now have cryptographic keys ready for certificate authority interoperability and cryptographic operations.

Configure Ed25519 Crosswork Trust Insights integration

Configure Ed25519 Crosswork Trust Insights integration to enable interoperability between Cisco IOS XR and Cisco Crosswork Trust Insights using certificate authority features.

Enable secure integration between Cisco IOS XR and Crosswork Trust Insights by configuring Ed25519-based trustpoint and certificate management.

Use this task when you need to establish trust between Cisco IOS XR and Crosswork Trust Insights using the Ed25519 signature algorithm for certificates. This capability ensures interoperability with certificate authority requirements.

Before you begin

- Review certificate authority prerequisites and verify access to Cisco IOS XR device configuration.
- Ensure you have permission to enroll and authenticate trustpoints.

Procedure

1. Configure the domain name on the device.

   Example

   ```
   Router# configure
   Router(config)# domain name domain1
   Router(config)# crypto ca trustpoint system-trustpoint
   Router(config-trustp)#keypair ed25519 system-enroll-key
   Router(config-trustp)#ca-keypair ed25519 system-root-key
   Router(config-trustp)# commit

   Router# crypto ca authenticate system-trustpoint
   Router# crypto ca enroll system-trustpoint
   ```

   This section shows how to generate the system trustpoint, and the root and leaf certificates using the Ed25519 signature algorithm, as part of integrating Cisco IOS XR with Cisco Crosswork Trust Insights.

2. Create a crypto CA trustpoint using Ed25519 key pairs.

   Example

   ```
   Router(config)# crypto ca trustpoint system-trustpoint
   Router(config-trustp)#keypair ed25519 system-enroll-key
   Router(config-trustp)#ca-keypair ed25519 system-root-key
   Router(config-trustp)# commit

   Router# crypto ca authenticate system-trustpoint
   Router# crypto ca enroll system-trustpoint
   ```

3. Authenticate the trustpoint with the certificate authority.

   Example

   ```
   Router# crypto ca authenticate system-trustpoint
   ```

4. Enroll the trustpoint with the certificate authority

Example

```
Router# crypto ca enroll system-trustpoint
```

5. Review the running configuration.

Example

```
config
domain name domain1
crypto ca trustpoint system-trustpoint
 keypair ed25519 system-enroll-key
 ca-keypair ed25519 system-root-key
!
```

The device is successfully integrated with Crosswork Trust Insights using Ed25519 trustpoint and certificate authority configuration. You will see matching configuration and command output confirming successful interoperability.

Public key-pairs

Explains public key-pairs for Cisco IOS XR certificate-based trust, including the source behavior, constraints, and operational details that support certification authority interoperability.

A public key-pair is a cryptographic credential that

- consists of a mathematically linked public and private key
- enables secure communication and authentication, and
- supports multiple cryptographic algorithms such as RSA, DSA, ECDSA, and ED25519.

Feature history
Table 51: Feature History Table

Feature Name	Release Information	Feature Description
Syslog alerts on public keys generated in XR config mode	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) This enhancement ensures Cisco IOS XR devices remain compliant with evolving security standards. During system reboot, the device now sends syslog warnings for public keys if weak SSH host keys are detected—specifically, RSA keys less than 3072 bits or any DSA keys. Additionally, the default RSA key size has been increased from 2048 to 3072 bits to further strengthen security. This feature introduces these changes: CLI: . • The **crypto key generate rsa** command has been modified. • The **crypto key generate dsa** command has been modified. • The **show crypto key mypubkey rsa** command has been modified.
Public Key-Pair Generation in XR Config Mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100])(select variants only*) *This feature is supported on the Cisco 8712-MOD-M routers.
Public Key-Pair Generation in XR Config Mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 8212-48FH-M • 8711-32FH-M

Feature Name	Release Information	Feature Description
Public Key-Pair Generation in XR Config Mode	Release 7.3.2	This feature allows you to generate public-key pairs in the XR Config mode, which in turn lets you save configurations. You can then load these saved configurations across different routers to quickly deploy the key-pair configurations. You could generate public-key pairs in earlier releases only in the XR EXEC mode, which does not save configurations. So manually executing the key-pair generation commands on every router was time-consuming. The following commands are available in XR Config mode, in addition to XR EXEC mode: • crypto key generate rsa • crypto key generate dsa • crypto key generate ecdsa • crypto key generate ed25519

Additional reference information

Syslog alerts on public keys generated in XR config mode (Cisco IOS XR Release 26.1.1):

- The default RSA key size is 3072 bits.
- A syslog warning is triggered during system boot if a weak SSH host key is detected (RSA key less than 3072 bits).
- DSA keys are no longer auto-generated at boot. If present, a syslog warning prompts removal.
- These changes align device behavior with current cryptographic security standards and improve protection.

The following table lists the supported key types and key sizes in FIPS and non-FIPS modes:

Key Type	Non-FIPS Mode	FIPS Mode
RSA	Supported for all key sizes from 512-4096	Supported for key sizes 2048, 3072, 4096
DSA	Supported for key sizes 512, 768, 1024	Supported for key size 2048
ECDSA	Supported for nistp256, nistp384, nistp512	Supported for nistp256, nistp384, nistp512
ED25519	Supported	Not Supported

For more details on FIPS, see the *Configuring FIPS Mode* chapter.

Requirement: Follow public key-pair usage guidelines

Outlines the requirements and restrictions for generating public key-pairs in XR config mode to keep certificate authority deployment supported.

Follow these requirements when generating public key-pairs in XR config mode:

- Do not generate**system-root-key** and **system-enroll-key** in XR config mode.
- Generating a key-pair in XR config mode overwrites any key-pair previously generated in XR EXEC mode, except in Cisco IOS XR Release 24.4.1 and later.
- From Release 24.4.1 onwards, generate keys in XR config mode only after zeroizing or removing any existing key using the no generate command.
- You cannot overwrite or delete keys generated in XR config mode from XR EXEC mode.
- When you use the **no** form of the crypto key generate command in XR config mode, only keys generated in config mode are deleted.
- The **show crypto key mypubkey** displays keys generated in XR EXEC mode first, followed by those generated in XR config mode.

Configure public key-pairs in XR config mode

Configure public key-pairs in XR config mode to enable secure device communications with certificate authority interoperability.

Establish cryptographic key-pairs in Cisco IOS XR config mode to secure device communication.

Use cryptographic key-pairs with certificate authorities to support secure operations in Cisco IOS XR devices. XR config mode allows you to generate DSA, RSA, ECDSA, and ED25519 keys of various sizes.

Before you begin

- Review certificate authority prerequisites.
- Ensure you are running Cisco IOS XR Release 26.1.1 or later, as the default RSA key size is 3072 bits.

Procedure

1. Generate desired cryptographic keys.

   Example

   ```
   Router# conf t
   Router(config)# crypto key generate dsa 1024
   Router(config)# crypto key generate rsa user1 general-keys 3072
   Router(config)# crypto key generate rsa user2 usage-keys 3072
   Router(config)# crypto key generate rsa 3072
   Router(config)# crypto key generate ecdsa nistp256
   Router(config)# crypto key generate ecdsa nistp384
   Router(config)# crypto key generate ecdsa nistp521
   Router(config)# crypto key generate ed25519
   Router(config)# commit
   ```

   Use **no** form of the command in XR Configuration mode to delete any of the key-pairs.

   Starting with Cisco IOS XR Release 26.1.1, the default RSA key size is 3072 bits; syslog warning is triggered during system boot for weak SSH host key (RSA key < 3072 bits). Additionally, if you configure the DSA key, a syslog on system boot alerts you to remove the DSA key.

2. Verify the generated keys.

Example

```
Router# show crypto key mypubkey ecdsa
Key label: the_default
Type     : ECDSA General Curve Nistp256
Degree   : 256
Created  : 11:49:22 IST Wed Apr 21 2021
Data     :
 04D6D132 2253ABD0 81449E3F 9D5CEA3A 1107950A 829E9090 8960FBD5 ABA039B7
 24A4E217 7EA47475 91C60AC7 013DBC2E EA8434D9 0BD5B0FC 694913AE 0098A4F5
 77

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 22:44:22 IST Thu Mar 18 2021
Data     :
 04017798 4369F493 8D0E57D1 1975FC46 CDC03A78 03A9F90E B38CA504 17DB9A64
 D1DEA6A6 D23E7E20 4D8D4D31 C7878BDB BF5EEE40 1978A889 70C5D703 BB033B77
 0FFD9201 366A9AC8 35E69BB3 97FF4E91 6B498510 39425971 C5E43858 83286088
 A6A7BF92 0EA2B416 BD4E81CE DCEB65F1 15CC75B5 91204E89 3339A168 2382CAB6
 40170131 8F


----------------------------
 Public keys from config sysdb:
 ----------------------------
 Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 11:51:52 IST Wed Apr 21 2021
Data     :
 045F7C14 1A88C27E 9CED3FF1 7FEDFA03 B49575FA 7AD88370 BC9C7D7F F99C8917
 33620916 758BDEFC 7187E33A 2D3CCD33 14FF3267 9855A5E9 E3BD166C CE838462
 40742231 6198EE12 3E189F42 22A8149A 8E7B186D 88E728D4 7F47D565 53441061
 79
```

The device now has public key-pairs configured as required. The verification command confirms certificate authority interoperability and key presence in XR config mode.

System logs and error messages for public key-pair operations

Provides system log facts and error messages related to public key-pair creation, deletion, and certificate authority interoperability, including command output to help verify behavior.

Use this reference to look up details of public key-pair system logs and error messages generated during key-pair operations. Apply these facts when planning, configuring, or validating certificate authority interoperability.

Log messages for key-pair operations

- Logs on successful key-pair creation:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA generated,
label:the_default, modBits:1024
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key ECDSA_NISTP256 generated,
 label:the_default, modBits:256
```

- Logs on key-pair deletion:

```
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key RSA zeroized, label:user1
cepki[287]: %SECURITY-CEPKI-6-KEY_INFO : crypto key DSA zeroized,
label:the_default
```

- Error messages when overwriting key-pairs generated in XR Config Mode from XR EXEC mode:

```
Router# conf t
Router(config)# crypto key generate ed25519
Router(config)# commit
Router# crypto key generate ed25519
Cannot execute the command : Operation not permitted
ce_cmd[68727]: %SECURITY-CEPKI-6-ERR_2 : Cannot       execute the command :
   Operation not permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR : Key is added as part of config mode,
 key deletion is not allowed , delete key from config mode
```

- Error messages when deleting key-pairs generated in XR Config Mode from XR EXEC mode:

```
Router# conf t
Router(config)# crypto key generate ed25519
Router(config)# commit
Router# crypto key zeroize ed25519
Cannot execute the command : Operation not permitted
ce_cmd[68736]: %SECURITY-CEPKI-6-ERR_2 : Cannot execute the command :
Operation not permitted
```

- Logs and warnings after process restart (examples):

```
Router# process restart cepki
Tue Dec  9 04:50:55.396 UTC
RP/0/RP0/CPU0:Dec  9 04:50:55.445 UTC: sysmgr_control[66932]:
%OS-SYSMGR-4-PROC_RESTART_NAME : User cafyauto (con0_RP0_CPU0) requested a
restart of process cepki at 0/RP0/CPU0
Router#


This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply third-party
authority to import, export, distribute or use encryption. Importers,
exporters, distributors and users are responsible for compliance with
U.S. and local country laws. By using this product you agree to comply
with applicable laws and regulations. If you are unable to comply with
U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be
found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

RP/0/RP0/CPU0:Dec  9 04:50:56.430 UTC: cepki[277]:
%INFRA-WARN_INSECURE-4-INSECURE_FEATURE_WARN : Feature 'Weak RSA SSH host
key present' utilized or configured. This feature is deprecated as it is
known to be insecure; it will be removed in a future release. Consider
upgrading RSA (crypto key generate rsa) to at least 3072 bits

...
```

```
Router# process restart cepki
Tue Dec  9 05:14:49.926 UTC
RP/0/RP0/CPU0:Dec  9 05:14:49.973 UTC: sysmgr_control[68825]:
%OS-SYSMGR-4-PROC_RESTART_NAME : User cafyauto (con0_RP0_CPU0) requested a
restart of process cepki at 0/RP0/CPU0
Router#

This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply third-party
authority to import, export, distribute or use encryption. Importers,
exporters, distributors and users are responsible for compliance with
U.S. and local country laws. By using this product you agree to comply
with applicable laws and regulations. If you are unable to comply with
U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be
found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.

RP/0/RP0/CPU0:Dec  9 05:14:50.969 UTC: cepki[277]:
%INFRA-WARN_INSECURE-4-INSECURE_FEATURE_WARN : Feature 'Weak DSA SSH host
key present' utilized or configured. This feature is deprecated as it is
known to be insecure; it will be removed in a future release. Consider
removing DSA (crypto key zeroize dsa)
```

12 Keychain Management

Topics:

- Keychains

Details keychain management principles, implementation practices, operational processes, and configuration examples to guide secure authentication, key lifetime administration, and compliance with system requirements within network environments.

Keychains

Outlines the fundamentals of keychains, highlighting requirements related to system-clock changes, best practices for keychain implementation, supported applications, proper key lifetime administration, and step-by-step operational procedures for configuring keychains and cryptographic algorithms, with comprehensive configuration examples.

A keychain is a method for configuring shared secrets that

- supports authentication between routers, routing protocols, and network management applications
- protects communication with peers by requiring applications to use keychains, and
- allows multiple keys to be managed together so keys can roll over without interrupting authenticated communication.

Cisco 8000 Series Routers use keychain management to prepare shared secrets before peers establish trust. Routing protocols and network management applications use authentication to protect communication with peers.

Requirement: Account for system-clock changes

Outlines the system clock restriction that you must consider because key validity depends on configured lifetime values before keychain configurations are used

You must be aware that changing the system clock impacts the validity of the keys in the existing configuration.

Keychain implementations

Explains how Cisco IOS XR applications use keychain implementations to authenticate peer communication and support lifetime-based key rollover for routing and management applications.

A keychain implementation is a key-management capability that

- organizes shared keys for applications to authenticate peer communication
- enables secure handling and rollover of cryptographic keys based on configured lifetimes, and
- supports hitless key rollover for routing protocols such as BGP, OSPF, and IS-IS, as well as authentication for RSVP and IP SLAs.

Applications that use keychains for authentication

Lists the Cisco IOS XR routing and management applications that use keychains for peer authentication, and provides details on how each application implements keychain support.

The following Cisco IOS XR applications support keychain-based authentication after a keychain has been configured:

Application	Keychain usage
BGP	Uses TCP authentication and sends a Message Authentication Code (MAC) based on the keychain cryptographic algorithm. Supports hitless key rollover.
OSPF	Uses keychains for peer authentication and hitless key rollover.
IS-IS	Uses keychains for peer authentication and hitless key rollover.
RSVP	Uses keychains for authentication of RSVP sessions.

Application	Keychain usage
IP SLAs	Uses keychains for MD5 authentication of the IP SLA control message.

Key lifetimes

Explains key lifetimes and the configured time values that control when a key can be used for Cisco IOS XR authentication.

A key lifetime is a time interval in a keychain that

- specifies the absolute time when a key becomes active
- defines when the key ends, either by absolute time, relative duration, or an infinite setting, and
- determines when routing update packets are sent using the active key.

Additional reference information

Keys cannot be used during periods when they are not active.

A key with a start-time set for midnight and an end-time set for 24 hours later will be valid only within that interval. Keys outside their configured lifetime will not be used for authentication.

Requirement: Configure valid and overlapping key lifetimes

You must configure valid and overlapping key lifetimes to prevent inactive-key gaps during planned rollover windows.

To ensure keychain authentication remains active during scheduled rollover periods, follow these requirements for key lifetimes:

- Configure a lifetime for every key you add to a keychain.
- A key without a lifetime is invalid and is rejected during configuration.
- Each key definition must specify the time interval during which that key is active.
- Overlap activation times in a keychain to avoid any period when no key is active.
- If no key is active during a time period, neighbor authentication cannot occur and routing updates can fail.
- Use Network Time Protocol (NTP) or another time-synchronization method when you configure time-based lifetimes.

How keychain management works

Describes the Cisco IOS XR keychain management implementation sequence from authorization checks through keychain creation, key attributes, algorithm selection, commit, and verification.

Cisco IOS XR implements keychain management by creating a keychain and adding keys. The process defines key material and lifetimes, selects an algorithm, and verifies the resulting configuration.

Summary

The key components involved in the process are:

- Keychain: Groups keys that authenticate the same peer, peer group, or both.
- Key identifier: Uniquely identifies each key inside a keychain.
- Key string: Stores the shared secret text for the key.
- Send and accept lifetimes: Control when a key can authenticate outbound and inbound traffic.

- Cryptographic algorithm: Defines the algorithm used for authentication digest generation or validation.

Workflow

These stages describe how keychain management works.

1. Verify that the user has the task IDs required to configure the keychain commands.
2. Create or enter the keychain, and optionally configure the accept tolerance used during rollover.
3. Create the key identifier and configure the key string.
4. Configure accept and send lifetimes so inbound and outbound authentication keys are valid during the planned windows.
5. Configure the cryptographic algorithm, commit the changes, and verify keychain operational output.

Result

The process is complete when the keychain has at least one valid key and the application that uses the keychain can authenticate its peer.

Configure a keychain

Configure a network keychain by naming it, setting attributes, committing changes, and verifying its operation.

Establish a secure keychain for authentication purposes within your router configuration.

Keychains are used to manage authentication keys for protocols such as IS-IS. You can create a new keychain or modify an existing one by naming it and configuring one or more authentication keys.

Before you begin

- Ensure you have the required task ID permissions for keychain management.
- Plan your keychain keys and attributes, such as lifetimes or cryptographic algorithms.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create or enter the keychain configuration.

   Example

   ```
   Router(config)# key chain isis-keys
   ```

   Creates a name for the keychain.

   - Specify a unique name for the keychain.
   - For correct operation, configure at least one key identifier with attributes such as lifetime or key string. Just configuring the keychain name is not sufficient.

3. Commit the configuration changes.

Commits the configuration changes and remains within the configuration session.

4. (Optional) Display keychain information.

Example

```
Router# show key chain isis-keys
Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```

If you do not specify a key-chain-name, all configured keychains are displayed.

5. Review the running configuration.

Example

```
key chain isis-keys
  accept-tolerance infinite
  key 8
   key-string mykey91abcd
   cryptographic-algorithm MD5
   send-lifetime 1:00:00 june 29 2006 infinite
   accept-lifetime 1:00:00 june 29 2006 infinite
   !
  !
 !
```

The keychain is successfully configured when the router accepts the commit and displays the expected keychain values.

Configure accept tolerance for keys

Configure accept tolerance for a keychain so received peer keys remain acceptable during rollover, including the use of finite second values or the infinite option.

Set accept tolerance for keys to ensure seamless rollover of keys in your keychain configuration.

Accept tolerance allows a keychain to accept keys during hitless key rollover for routing and management protocols. This prevents disruptions when transitioning between keys.

Before you begin

Ensure you have the necessary task ID permissions for keychain management.

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Create or enter the keychain configuration.

   Example

   ```
   Router(config)# key chain isis-keys
   ```

   Creates a name for the keychain.

3. Configure the accept tolerance for received packets.

   Example

   ```
   Router(config-isis-keys)# accept-tolerance infinite
   ```

   The accept tolerance limit applies to a peer key.

   The limit defines how long an expired or soon-to-be active key can validate received packets.

   - Specify the tolerance value in seconds.

     The range is from 1 to 8640000.

   - Use the **infinite** keyword to specify that an accept key is always acceptable and validated when used by a peer.

4. Commit the configuration changes.

   Commits the configuration changes and remains within the configuration session.

The router now accepts peer keys according to the specified accept tolerance in the keychain configuration. Key rollover will be hitless as configured.

Configure a key identifier for a keychain

Configure a key identifier within a keychain so Cisco IOS XR can attach key-string, lifetime, and algorithm attributes to a unique 48-bit key ID.

Use this task to configure a key identifier for a keychain.

This task configures a key identifier for the keychain. You can create or change the key for the keychain.

Before you begin

Confirm that you have the required task ID permissions for keychain management.

Procedure

---

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create or enter the keychain configuration.

Example

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

3. Create or enter the key configuration.

Example

```
Router(config-isis-keys)# key 8
```

The key ID must be unique within the keychain.

The key ID is a 48-bit integer.

4. Commit the configuration changes.

Commits the configuration changes and remains within the configuration session.

The key identifier exists when the router commits the key ID under the selected keychain.

Configure a key string

Configure text for a key string and choose whether Cisco IOS XR treats the string as cleartext or an encrypted password value.

Use this task to define or update the text value for a key string in a Cisco IOS XR router configuration. You can configure the string as either cleartext or in encrypted form to control how the router stores and uses the key in authentication.

Configure the text of a key string within a Cisco IOS XR keychain, specifying its format and committing the configuration.

Before you begin

Ensure you have already created the keychain and a key identifier.

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Create or enter the keychain configuration.

Example

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

3. Create or enter the key configuration.

Example

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

4. Configure the key-string text.

Example

```
Router(config-isis-keys-0x8)# key-string password 8
```

The key-string value specifies the text string for the key.

The clear keyword specifies the key string in cleartext form.

The password keyword specifies the key in encrypted form.

5. Commit the configuration changes.

Commits the configuration changes and remains within the configuration session.

The key string is configured and committed to the router under the selected key identifier. The specified string is stored as cleartext or encrypted, depending on your selection.

Configure accept lifetimes for valid keys

Configure accept lifetimes for valid keys so local applications can authenticate remote peers only during the configured receive-validity window for each keychain key.

Set up accept lifetimes for keys in a keychain to control the window during which local applications authenticate remote peers.

Use accept lifetimes to ensure authentication happens only within a specific, active receive-validity window for each keychain key.

Before you begin

Create the keychain and key identifier before configuring the key attributes.

Procedure

---

1. Enter global configuration mode.

Example

```
Router# configure
```

Enters global configuration mode.

2. Create or enter the keychain configuration.

Example

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

3. Create or enter the key configuration.

   Example

   ```
   Router(config-isis-keys)# key 8
   Router(config-isis-keys-0x8)#
   ```

   Creates a key for the keychain.

4. Configure the accept lifetime for the key.

   Example

   ```
   Router(config-isis-keys)# key 8
   Router(config-isis-keys-0x8)# accept-lifetime 1:00:00 october 24 2005 infinite
   ```

   (Optional) Specifies the validity of the key lifetime in terms of clock time. You can specify the *start-time* and *end-time* in *hh:mm:ss month DD YYYY* format or *hh:mm:ss DD month YYYY* format.

5. Commit the configuration changes.

   Commits configuration changes and exits the configuration session.

The accept lifetime is set for each key, so local applications authenticate remote peers only during the valid receive window for the configured key.

Configure outbound authentication digest keys

Configure outbound authentication digest keys with send lifetimes to control when each key is valid for outbound authentication.

Configure outbound authentication digest keys with specific send lifetimes, ensuring keys are used only within their defined validity windows for outbound authentication.

Setting send lifetimes allows outbound authentication digest keys to generate authentication digests for application traffic only during defined periods. This enhances security by restricting the validity window and ensures proper timing, which is especially important if using Network Time Protocol (NTP) or other time synchronization methods.

Before you begin

- Create the keychain and key identifier before configuring key attributes.
- Ensure system time is synchronized (for example, with NTP) if using time-based lifetimes.

Procedure

---

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create or enter the keychain configuration.

Example

```
Router(config)# key chain isis-keys
```

Creates a name for the keychain.

3. Create or enter the key configuration.

Example

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)#
```

Creates a key for the keychain.

4. Configure the send lifetime for the key.

Example

```
Router(config-isis-keys)# key 8
Router(config-isis-keys-0x8)# send-lifetime 1:00:00 october 24 2005 infinite
```

(Optional) Specifies the time period during which an authentication key on a keychain is valid for sending.

You can specify the key lifetime in terms of clock time.

You can also configure the start time with one of these lifetime options

- The duration keyword specifies a lifetime in seconds
- The infinite keyword specifies that the key lifetime does not expire
- *end-time* argument

If you intend to set lifetimes on keys, Network Time Protocol (NTP) or some other time synchronization method is recommended.

5. Commit the configuration changes.

Commits the configuration changes and remains within the configuration session.

The key's send lifetime is configured. Outbound authentication digest generation uses the key only during the specified send-validity window.

Requirement: You must use FIPS-approved keychain algorithms

Outlines the FIPS-approved keychain algorithm and key-string length requirements that you must follow when FIPS mode is enabled for authenticated sessions.

To ensure compliance when FIPS mode is enabled for authenticated sessions, follow these requirements:

- Configure keychain sessions with a FIPS-approved cryptographic algorithm when **crypto fips-mode** is enabled.
- Do not use nonapproved FIPS algorithms such as MD5 or HMAC-MD5; sessions using these algorithms will fail.
- Ensure that all applications using keychains—such as OSPF, BGP, RSVP, IS-IS, or any other protocol—use only approved algorithms.
- When using any HMAC-SHA algorithm, configure a key string with a minimum length of 14 characters.
- Never use an HMAC-SHA key string shorter than 14 characters; otherwise, the session will go down.

Configure a cryptographic algorithm

Configure a keychain cryptographic algorithm for authentication digest generation and validation. Review FIPS constraints before using HMAC-SHA algorithms.

Configure a cryptographic algorithm for keychains that enables secure authentication digest generation and validation.

Use this task when you need to set or change the algorithm a keychain uses for authentication. In FIPS mode, you must use only approved algorithms to ensure compliance.

Before you begin

- Confirm that the keychain and key identifier exist.
- If operating in FIPS mode, review the requirement to use only FIPS-approved keychain algorithms before committing the configuration.

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

   Enters global configuration mode.

2. Create or enter the keychain configuration.

   Example

   ```
   Router(config)# key chain isis-keys
   Router(config-isis-keys)#
   ```

   Creates a name for the keychain.

3. Create or enter the key configuration.

   Example

   ```
   Router(config-isis-keys)# key 8
   Router(config-isis-keys-0x8)#
   ```

   Creates a key for the keychain.

4. Configure the keychain cryptographic algorithm.

   Example

   ```
   Router(config-isis-keys-0x8)# cryptographic-algorithm MD5
   ```

   The cryptographic-algorithm command specifies the algorithm choice.

   Supported keychain algorithms include these values

   - HMAC-MD5
   - HMAC-SHA1-12

- HMAC-SHA1-20
- MD5
- SHA-1
- HMAC-SHA-256
- HMAC-SHA1-96
- AES-128-CMAC-96

Protocol-specific algorithm support includes these values

- Border Gateway Protocol (BGP) supports HMAC-MD5, HMAC-SHA1-12, HMAC-SHA1-96 and AES-128-CMAC-96.
- Intermediate System-to-Intermediate System (IS-IS) supports HMAC-MD5, SHA-1, MD5, AES-128-CMAC-96, HMAC-SHA-256, HMAC-SHA1-12, HMAC-SHA1-20, and HMAC-SHA1-96.
- Open Shortest Path First (OSPF) supports MD5, HMAC-MD5, HMAC-SHA-256, HMAC-SHA1-12, HMAC-SHA1-20, and HMAC-SHA1-96.

5. Commit the configuration changes.

   Commits configuration changes and exits the configuration session

---

The selected cryptographic algorithm is applied to the keychain for authentication, and the router will display the algorithm in the keychain output once the configuration is accepted.

Keychain management configuration examples

Provides source-backed keychain management examples featuring complete configuration outputs and operational displays to help validate configured keychain values.

The following reference section includes:

- A complete example of keychain management configuration, enabling you to review keychain setup step by step.
- Matching operational output from show commands, allowing you to validate your configured keychain values against real device output.
- Guidance for comparing keychain attributes displayed in configuration files with their corresponding operational display.

How to use these examples

- Review the configuration section to understand proper keychain setup.
- Use the show command output provided to verify that your keychain configuration appears as expected on your device.
- Compare the attributes and values listed to ensure consistency and correct operation.

Keychain management configuration example

Provides a complete keychain configuration and corresponding verification output for the isis-keys keychain, including accept tolerance, key string, algorithm, and lifetimes.

This reference includes both a configuration example for the isis-keys keychain and the corresponding verification output from the show key chain command.

Keychain configuration example

The following configuration sets up the keychain named isis-keys with infinite accept tolerance. It defines key 8, assigns a key string, specifies the MD5 algorithm, and applies send and accept lifetimes:

```
configure
 key chain isis-keys
  accept-tolerance infinite
  key 8
   key-string mykey91abcd
   cryptographic-algorithm MD5
   send-lifetime 1:00:00 june 29 2006 infinite
   accept-lifetime 1:00:00 june 29 2006 infinite
```

Verification output

The show key chain command displays the keychain details, including accept tolerance, key ID, key text, cryptographic algorithm, and the send and accept lifetimes:

```
Router# show key chain isis-keys

Key-chain: isis-keys/ -

accept-tolerance -- infinite
Key 8 -- text "1104000E120B520005282820"
  cryptographic-algorithm -- MD5
  Send lifetime:   01:00:00, 29 Jun 2006 - Always valid  [Valid now]
  Accept lifetime: 01:00:00, 29 Jun 2006 - Always valid [Valid now]
```

13 MACsec using EAP-TLS Authentication

Topics:

- MACsec EAP-TLS authentication sessions

Outlines MACsec EAP-TLS authentication principles, device roles, prerequisites, local authentication models, configuration workflows, and verification procedures for integrating EAP-TLS with MACsec encryption in network environments.

MACsec EAP-TLS authentication sessions

Introduces MACsec EAP-TLS authentication session concepts, covering authentication limits, device roles, prerequisites, local authentication models with legacy TLS 1.3 KDF, encryption processes, and comprehensive configuration tasks for secure MACsec deployment.

A MACsec EAP-TLS authentication session is a certificate-based exchange that

- uses Extensible Authentication Protocol-Transport Layer Security (EAP-TLS) for peer authentication
- enables mutual authentication between authentication server and client, and
- derives the Master Session Key (MSK) that is used for MKA key material.

Additional reference information

- EAP-TLS utilizes certificates for mutual authentication between routers.
- After successful authentication, EAP-TLS generates the MSK, which is used to derive the Connectivity Association Key (CAK).
- The Connectivity Association Key Name (CKN) is derived from the EAP session ID.

Requirement: Follow EAP-TLS authentication limits

You must follow these EAP-TLS authentication limits before configuring MACsec with 802.1X authentication on physical Ethernet interfaces and supported host modes.

To remain within Cisco IOS XR supported behavior for MACsec EAP-TLS authentication, always follow these requirements:

- Use 802.1X only on physical Ethernet interfaces.
- Use only EAP-TLS as the authentication method, as supported by Cisco IOS XR for this use case.
- Use 802.1X port-based authentication only to derive keys for MACsec Key Agreement (MKA); do not use it for port control.
- Use supported PAE roles only: authenticator and supplicant.
- If Cisco IOS XR acts as the authenticator, use RADIUS as the EAP transport for remote EAP authentication.
- Use only single-host mode; multi-host mode is not supported.

IEEE 802.1X device roles

Lists the IEEE 802.1X supplicant, authenticator, and authentication server roles that participate in MACsec EAP-TLS authentication on point-to-point LAN segments.

Devices in the network have these specific IEEE 802.1X authentication roles.

Table 52: IEEE 802.1X roles

Role	Description
Supplicant	An entity at one end of a point-to-point LAN segment that seeks authentication by an authenticator attached to the other end of that link.
Authenticator	An entity that facilitates authentication of other entities attached to the same LAN.

Role	Description
Authentication server	An entity that provides an authentication service to an authenticator and determines whether the supplicant is authorized to access services.

Each role identifies how a device participates in authentication on a point-to-point LAN segment.

Use these role definitions to map routers, clients, and AAA infrastructure to the MACsec EAP-TLS authentication workflow.

Requirement: Meet MACsec MKA EAP-TLS prerequisites

Outlines the CA, AAA, EAP-TLS, and time-synchronization requirements to satisfy before enabling MACsec MKA with certificate-based EAP-TLS authentication.

You must ensure all critical prerequisites are met before configuring MACsec MKA with EAP-TLS authentication.

- Configure a Certificate Authority (CA) server for the network.
- Ensure that the configured CA certificate is valid.
- Configure Cisco Identity Services Engine (ISE) Release 2.2 or later as the external AAA server.
- Alternatively, configure Cisco Secure Access Control Server Release 5.6 or later as the external AAA server.
- Configure the remote AAA server with the EAP-TLS method.
- Synchronize both routers, the CA server, and the external AAA server with Network Time Protocol (NTP). If time is not synchronized, certificates may not be validated.

Use this requirement before you configure RADIUS, certificates, EAP profiles, 802.1X profiles, or MACsec EAP on an interface.

MACsec local EAP-TLS authentication models

Describes MACsec local EAP-TLS authentication models, providing guidance on enabling legacy TLS 1.3 KDF and implementing secure local authentication procedures.

A MACsec local EAP-TLS authentication model is a type of local EAP authentication configuration that

- co-locates the EAP server with the authenticator on the router
- enables authentication of dot1x (802.1X) clients using the EAP-TLS method, and
- provides mutual authentication and generates an MSK after authentication succeeds.

Interoperability requirement

For local EAP authentication with earlier XR releases, see *Requirement: Enable legacy TLS 1.3 KDF for local EAP*.

Requirement: Enable legacy TLS 1.3 KDF for local EAP

Outlines the legacy KDF command requirement for EAP-based MACsec local EAP authentication interoperability with earlier XR releases.

Configure the **enable-tls1.3-legacy-kdf** command when you use EAP-based MACsec with local EAP authentication.

- This command is required for interoperability with XR releases earlier than Release 25.4.1.
- Use the command when the EAP server is co-located with the authenticator on the router.

How MACsec EAP-TLS encryption works

Explains MACsec EAP-TLS encryption operation, guiding users through configuring RADIUS servers, 802.1X authentication methods, RSA key pairs, domain names, trustpoints, certificate enrollment, EAP profiles, and interface profiles, including verification procedures.

Configuring MACsec encryption with EAP-TLS authentication requires preparing AAA, certificate, EAP, 802.1X, and interface settings before verification. This process ensures the device can securely authenticate, enroll for certificates, and apply the necessary policies to a physical interface.

Summary

The key components in this process are:

- RADIUS server: Provides remote EAP authentication transport for the authenticator role.
- 802.1X authentication method: Selects the default AAA method for dot1x authentication.
- RSA key pair and trustpoint: Provide the certificate enrollment foundation.
- EAP and 802.1X profiles: Bind EAP-TLS authentication behavior to the device role.
- MACsec EAP interface policy: Enables MACsec EAP behavior on the selected interface.

MACsec EAP-TLS encryption uses a multi-step configuration process that prepares authentication, certificate, EAP, and interface settings to establish secure Layer 2 encryption based on EAP-TLS.

Workflow

These stages describe how MACsec EAP-TLS encryption works.

1. Configure the RADIUS server pre-shared key and Vendor-Specific Attribute (VSA) behavior.
2. Configure the default 802.1X authentication method to use the RADIUS group.
3. Generate the RSA key pair required for certificate enrollment.
4. Configure the trustpoint, domain name, and certificate enrollment actions.
5. Configure the EAP profile and 802.1X profile on the device.
6. Attach the 802.1X profile and MACsec EAP policy to the interface.
7. Verify 802.1X authorization and MACsec MKA session state on the interface.

Result

The process is complete when 802.1X shows an authorized port state and MACsec MKA session output shows a secured session.

Configure a RADIUS server for MACsec EAP-TLS authentication

Configure the RADIUS server to support MACsec EAP-TLS authentication by setting pre-shared keys and appropriate VSA handling.

Set up the RADIUS server so MACsec EAP-TLS authentication can use the remote RADIUS server as the EAP transport.

Obtain pre-shared key values for the remote RADIUS server before configuring. You may specify either IPv4 or IPv6 addresses for the RADIUS server host.

Before you begin

Meet the MACsec MKA EAP-TLS prerequisites.

Procedure

1. Configure the RADIUS server pre-shared key and VSA behavior.

   Example

   ```
   Router# configure terminal
   Router(config)# radius-server host 209.165.200.225 key 7 094F471A1A0A57
   Router(config)# radius-server vsa attribute ignore unknown
   Router(config)# commit
   ```

   The example uses a remote RADIUS server host and pre-shared key value.

2. Review the RADIUS server running configuration.

   Example

   ```
   Router# show run radius-server
   radius-server host 209.165.200.225 auth-port 1646
     key 7 094F471A1A0A57
     radius-server vsa attribute ignore unknown
   !
   ```

   The running configuration shows the expected commands.

The RADIUS server is ready for MACsec EAP-TLS authentication when the running configuration shows the correct host address, key, and VSA attribute configuration.

Configure the 802.1X authentication method

Configure the default 802.1X AAA authentication method on the router so MACsec EAP-TLS authentication can use RADIUS as the authentication protocol.

Configure 802.1X authentication on Cisco IOS XR using the default AAA method with RADIUS as the protocol.

Cisco IOS XR supports only the default AAA method for 802.1X authentication. This ensures MACsec EAP-TLS authentication uses RADIUS for secure and standardized access control.

Before you begin

Configure the RADIUS server before you configure the 802.1X authentication method.

Procedure

1. Configure 802.1X authentication to use the default RADIUS method.

   Example

   ```
   Router# configure terminal
   Router(config)# aaa authentication dot1x default group radius
   Router(config)# commit
   ```

   Only the default AAA method is supported for 802.1X authentication.

2. Review the 802.1X AAA running configuration.

Example

```
Router# show run aaa
configure
  aaa authentication dot1x default group radius
```

The running configuration shows the expected commands.

The authentication method is configured when the running configuration shows the default dot1x method with the RADIUS group.

Configure an RSA key pair for MACsec EAP-TLS

Configure an RSA key pair that signs and encrypts key management messages before the router obtains the certificate used for MACsec EAP-TLS authentication.

Generate the RSA key pair required before the node obtains its certificate for MACsec EAP-TLS.

RSA key pairs are used to sign and encrypt key management messages for MACsec EAP-TLS authentication. You must enter the key modulus size when prompted during key generation.

Before you begin

Ensure you meet the Certificate Authority (CA) and Network Time Protocol (NTP) requirements before certificate enrollment.

Procedure

---

1. Generate the RSA key pair and enter the modulus size when prompted.

   Example

   ```
   Router# crypto key generate rsa 8002
   Wed Aug  7 10:25:22.461 UTC
   The name for the keys will be: 8002
   Choose the size of the key modulus in the range of 512 to 4096 for your General
    Purpose Keypair.
   Choosing a key modulus greater than 512 may take a few minutes.

   How many bits in the modulus [2048]: 600
   Generating RSA keys ...
   Done w/ crypto generate keypair
   [OK]
   ```

   The command prompts for a key modulus size in the range of 512 to 4096.

2. Display the generated RSA public key.

   Example

   ```
   Router# show crypto key mypubkey rsa

   Key label: 8002
   Type     : RSA General purpose
   Size     : 600
   Created  : 12:56:29 UTC Wed Aug 07 2019
   Data     :
    3067300D 06092A86 4886F70D 01010105 00035600 3053024C 0096DB0F EE3B3233
    6E5FDA53 0FC504D1 9A056E29 BB703118 C6A8A254 1DC6504B 29CD4DA0 984735C8
   ```

```
 46CD39A1 C379B059 92870F76 693D4A66 D9953F69 450238D4 C57803AF 41160D4F
 C9451945 02030100 01
```

The show command output identifies the key label, key type, size, creation time, and public key data.

3. Review RSA keys that are visible in the running configuration.

Example

```
Router(config)#crypto key generate rsa test
Router(config)#commit
Thu May 12 08:37:59.894 UTC
Router(config)#end
Router#show running-config
Thu May 12 08:38:04.244 UTC
Building configuration...
!! IOS XR Configuration 7.3.4
!! Last configuration change at Thu May 12 08:37:59 2022 by cisco
!
username cisco
 group root-lr
 group cisco-support
 secret 10
$6$8zR0nIbkA7A1n...$0Kn.YxNNmh1cXo9cEvEwLGAFf.rEOIycjsizI/TLBz9WoQX.rmxVwkNgIKAnROUGPtBVlQ/Ndew8gEREXJ7mI0
!
call-home
 service active
 contact smart-licensing
 profile CiscoTAC-1
  active
  destination transport-method http
 !
!
interface MgmtEth0/RSP0/CPU0/0
 shutdown
!
crypto key generate rsa test general-keys 2048 | -----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
AIIlxZ5XfwAAcJb6e1d/AAAA7du+AAAAAI6Qs47BQLhIVQAAAAAAAAAAWQDQVn8A
ANyKXp5XfwAAKAAAAAAAAACaNcWeV38AANyKXp5XfwAAmjXFnld/AADcil6eV38A
AJo1xZ5XfwAAAO3bvgAAAABVAAAAAAAAABBEANBWfwAA3Ipenld/AAAgAAAAAAAA
AI8lxZ5XfwAA3Ipenld/AACPJcWeV38AAHhZANBWfwAAAO3bvgAAAADUTNDpQMWp
UUUAAAAAAAAAkBcA0FZ/AADcil6eV38AABgAAAAAAAAiSXFnld/AADcil6eV38A
AAIBAA==
-----END PUBLIC KEY-----
 |
end
```

Only keys generated in config mode are visible in the running configuration.

The example shows RSA keys in OpenSSL format.

The RSA key pair is available when the show command output displays the generated key label and public key data.

MACsec EAP-TLS trustpoints

Describes trustpoint concepts for MACsec EAP-TLS, including procedures for configuring trustpoints to facilitate secure certificate management.

A MACsec EAP-TLS trustpoint is a certificate authority (CA) and certificate management object that

- includes the identity of the CA
- includes CA-specific configuration parameters, and
- maintains an association with one enrolled identity certificate.

Additional reference information

After you define a trustpoint, you can reference it by name in commands that require a CA for MACsec EAP-TLS enrollment.

Configure a trustpoint for MACsec EAP-TLS

Configure a trustpoint so the router can reference CA identity, enrollment URL, subject name, RSA key pair, and CRL behavior during enrollment.

Set up a trustpoint for certificate enrollment, enabling the router to reference certificate authority (CA) identity and specific enrollment parameters.

Trustpoints manage CA identity, CA-specific parameters, and enrolled identity certificates. For more details, see the *MACsec EAP-TLS trustpoints* section.

Before you begin

Generate the RSA key pair before binding the trustpoint to that key pair.

Procedure

1. Configure the trustpoint, enrollment URL, subject name, RSA key pair, and CRL behavior.

   Example

   ```
   Router# configure terminal
   Router(config)# crypto ca trustpoint test2
   Router(config-trustp)# enrollment url http://caurl.com
   Router(config-trustp)# subject-name
   CN=8000Series,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
   Router(config-trustp)# rsakeypair 8002
   Router(config-trustp)# crl optional
   Router(config-trustp)# commit
   ```

   You can also specify the enrollment URL as an IP address, such as http://10.2.2.2.

2. Review the trustpoint running configuration.

   Example

   ```
   crypto ca trustpoint test2
    crl optional
    subject-name CN=8000Series,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
    enrollment url http://caurl.com
    rsakeypair 8002
      !
   ```

   The show run crypto ca trustpoint test2 sample output displays the configured trustpoint parameters.

The trustpoint is configured when the running configuration shows the trustpoint name, enrollment URL, subject name, RSA key pair, and CRL behavior.

Configure a domain name for certificate enrollment

Configure the router domain name required before certificate enrollment can create the identity certificate for MACsec EAP-TLS authentication.

Set the router's domain name as a prerequisite for certificate enrollment.

The domain name is essential for certificate enrollment preparation. It ensures that the certificate can be correctly generated and associated with the router for secure MACsec EAP-TLS authentication.

Before you begin

Configure the trustpoint before certificate enrollment.

Procedure

1. Configure the domain name.

   Example

   ```
   Router(config)# domain name ca.8000-series.cisco.com
   Router(config)# commit
   ```

   The domain name is required for certificate enrollment.

2. Review the domain name running configuration.

   Example

   ```
   Router# show run domain name
   Thu Mar 29 16:10:42.533 IST
   domain name ca.8000-series.cisco.com
   ```

   The running configuration shows the expected commands.

The domain name is successfully configured when the running configuration displays the expected domain name.

Obtain certificates from the certificate authority

Obtain the CA certificate and enroll the device certificate so MACsec EAP-TLS authentication can use validated certificate material for the router.

Obtain the CA certificate and enroll the device certificate to enable MACsec EAP-TLS authentication with validated certificates.

Certificate enrollment involves two actions:

1. Obtain the CA certificate for the trustpoint.
2. Enroll the device certificate with the certificate authority.

Before you begin

- Configure the CA server.
- Obtain a valid CA certificate.
- Generate an RSA key pair.
- Configure the trustpoint and domain name.

- Set up a remote AAA server.
- Synchronize NTP.

Procedure

1. Obtain the CA certificate for the trustpoint.

   Example

   ```
   Router# crypto ca authenticate test2
   ```

   Use the **crypto ca authenticate tp_name** command.

2. Enroll the device certificate with the CA.

   Example

   ```
   Router# crypto ca enroll test2
   ```

   Use the **crypto ca enroll tp_name** command.

3. Review the enrolled CA and router certificates.

   Example

   ```
   RP/0/RSP0/CPU0:router# show crypto ca certificates
   Trustpoint : test2
   ================================================
   CA certificate
   Serial Number        : E0:18:F3:E4:53:17:3E:28
   Subject             : subject-name CN=8002,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
   Issued By           : subject-name CN=8002,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
   Validity Start       : 08:17:32 UTC Fri Jun 24 2016
   Validity End         : 08:17:32 UTC Mon Jun 22 2026
   SHA1 Fingerprint     : 894ABBFAA3B08E5B7D9E470ECFBBC04576B569F2
   Router certificate
   Key usage            : General Purpose
   Status               : Available
   Serial Number        : 03:18
   Subject              :
   serialNumber=cf302761,unstructuredAddress=209.165.200.225,unstructuredName=8002,
   C=US,ST=NY,L=Newyork,O=Govt,OU=BU,CN=8002
   Issued By            : CN=8000Series,OU=BU,O=Govt,L=Newyork,ST=NY,C=US
   Validity Start       : 13:04:52 UTC Fri Feb 23 2018
   Validity End         : 13:04:52 UTC Sat Feb 23 2019
   SHA1 Fingerprint     :33B50A59C76CCD87D3D0F0271CD5C81F4A1EE9E1
   Associated Trustpoint: test2
   ```

   The sample output shows CA certificate and router certificate details, including validity dates, SHA1 fingerprints, and the associated trustpoint.

Certificate enrollment is complete when the **show crypto ca certificates** command displays the CA certificate, device certificate, SHA1 fingerprints, validity dates, and associated trustpoint.

Configure an EAP profile for MACsec EAP-TLS

Configure an EAP profile that identifies the device and binds EAP-TLS authentication to the PKI trustpoint used for MACsec certificate-based authentication.

Configure the EAP profile for MACsec EAP-TLS authentication.

You can configure multiple EAP profiles to support MACsec EAP-TLS certificate-based authentication for devices.

Before you begin

- Authenticate the certificate authority (CA).
- Request and install certificates for the device.

Procedure

1. Configure the EAP profile identity and TLS trustpoint method.

   Example

   ```
   Router# configure terminal
   Router(config)# eap profile 8002
   Router(config-eap)# identity CE1
   Router(config-eap)# method tls pki-trustpoint test2
   Router(config-eap)# commit
   ```

   You can configure multiple EAP profiles.

2. Review the EAP profile running configuration.

   Example

   ```
   Router# show run eap profile 8002
   eap profile 8002
   method tls pki-trustpoint test2
   !
   identity CE1
   !
   ```

   The running configuration shows the expected commands.

The EAP profile is successfully configured when the running configuration displays the profile identity and TLS PKI trustpoint method.

Configure an 802.1X profile

Configure the 802.1X profile with specific PAE role, reauthentication timer, and EAP profile settings for MACsec EAP-TLS authentication.

Set up an 802.1X profile to enable and control MACsec EAP-TLS authentication on the device.

The 802.1X profile defines PAE behavior, reauthentication timing, and binds the appropriate EAP profile for secure authentication.

Before you begin

Ensure that the EAP profile is already configured before referencing it in the 802.1X profile.

Procedure

1. Configure the 802.1X profile, PAE role, reauthentication timer, and supplicant EAP profile.

   Example

```
Router# configure
Router(config)# dot1x profile 8k_prof
Router(config-dot1x-8k_prof)# pae both
Router(config-dot1x-8k_prof)# authenticator timer reauth-time 3600
Router(config-dot1x-8k_prof)# supplicant eap profile 8002
Router(config-dot1x-8k_prof)# exit
Router(config)# commit
Router(config)# end
```

   The example configures the device to use both PAE roles.

2. Review the 802.1X profile running configuration.

   Example

```
Router# show run dot1x profile 8k_prof

dot1x profile 8k_prof
pae both
authenticator
  timer reauth-time 3600
!
supplicant
  eap profile 8002
!
```

   The running configuration shows the expected commands.

The 802.1X profile is configured and operational, with the running configuration showing the relevant PAE role, authenticator timer, and supplicant EAP profile settings.

Configure MACsec EAP and an 802.1X profile on an interface

Configure an 802.1X profile and MACsec EAP policy on the physical interface that uses MACsec EAP-TLS authentication for MKA key derivation.

Set up MACsec EAP and attach an 802.1X profile to a selected interface.

This task attaches the 802.1X profile to the chosen interface and enables the MACsec EAP policy, providing secure authentication and key derivation.

Before you begin

Ensure that the 802.1X profile is configured before attaching it to an interface.

Procedure

1. Attach the 802.1X profile and MACsec EAP policy to the interface.

Example

```
Router# configure
Router(config)# interface fourHundredGigE 0/0/0/0
Router(config-if)# dot1x profile 8k_prof
Router(config-if)# macsec eap policy macsec-1
Router(config-if)# commit
```

You can attach one 802.1X profile to an interface.

2. Review the interface running configuration.

Example

```
Router# show run interface HundredGigE 0/0/0/0
interface HundredGigE 0/0/0/0
dot1x profile 8k_prof
macsec eap policy macsec-1
!
```

The running configuration shows the expected commands.

The interface is configured when the running configuration displays the 802.1X profile and MACsec EAP policy under the interface.

Verify MACsec EAP and 802.1X configuration on an interface

Verify that the interface is authorized and that MACsec MKA session output shows secured EAP-based MACsec state, peer status, and cipher-suite details.

Ensure that MACsec EAP and 802.1X are properly configured and operational on the selected interface.

Use these steps after configuring the interface profile and MACsec EAP policy.

Before you begin

Complete the MACsec EAP and 802.1X interface configuration before running the verification commands.

Procedure

---

1. Use the **show dot1x interface** command to display detailed 802.1X interface status.

Example

```
Router# show dot1x interface HundredGigE 0/0/0/24 detail

Dot1x info for HundredGigE 0/0/0/24
-----------------------------------------------------------------
Interface short name : Hu0/0/0/24
Interface handle     : 0x800020
Interface MAC        : 0201.9ab0.85af
Ethertype            : 888E
PAE                  : Both
Dot1x Port Status    : AUTHORIZED
Dot1x Profile        : 8k_prof
Supplicant:
Config Dependency    : Resolved
EAP profile          : 8k
```

```
Client List:
Authenticator        : 0257.3fae.5cda
EAP Method           : EAP-TLS
Supp SM State        : Authenticated
Supp Bend SM State   : Idle
Last authen time     : 2018 Mar 01 13:31:03.380
Authenticator:
Config Dependency    : Resolved
ReAuth               : Enabled, 0 day(s), 01:00:00
Client List:
Supplicant           : 0257.3fae.5cda
Auth SM State        : Authenticated
Auth Bend SM State   : Idle
Last authen time     : 2018 Mar 01 13:33:17.852
Time to next reauth  : 0 day(s), 00:59:57
MKA Interface:
Dot1x Tie Break Role : Auth
EAP Based Macsec     : Enabled
MKA Start time       : 2018 Mar 01 13:33:17.852
MKA Stop time        : NA
MKA Response time    : 2018 Mar 01 13:33:18.357
```

The sample output shows the interface state, PAE value, authorized port status, EAP-TLS method, supplicant state, authenticator state, and EAP-based MACsec state.

2. Use the **show macsec mka session interface** command to display the MACsec MKA session summary for the interface.

   Example

```
Router# show macsec mka session interface HundredGigE 0/0/0/24

=========================================================================
Interface Local-TxSCI # Peers Status Key-Server
=========================================================================
Hu0/0/0/24  0201.9ab0.85af/0001 1 Secured YES
```

The sample output shows the local TxSCI, peer count, secured status, and key-server state.

3. Use the **show macsec mka session interface** command to display detailed MACsec MKA session status for the interface.

   Example

```
Router# show macsec mka session interface HundredGigE 0/0/0/24 detail

MKA Detailed Status for MKA Session
===================================
Status                                : SECURED - Secured MKA Session with
 MACsec

Local Tx-SCI                           : 0201.9ab0.85af/0001
Local Tx-SSCI                          : 2
Interface MAC Address                  : 0201.9ab0.85af
MKA Port Identifier                    : 1
Interface Name                         : Hu0/0/0/24
CAK Name (CKN)                         : A94399EE68B2A455F85527A4309485DA
CA Authentication Mode                 : EAP
Keychain                               : NA (EAP mode)
Member Identifier (MI)                 : 3222A4A7678A6BDA553FDB54
Message Number (MN)                    : 114
```

```
Authenticator                              : YES
Key Server                                 : YES
MKA Cipher Suite                           : AES-128-CMAC
Configured MACsec Cipher Suite             : GCM-AES-XPN-256
Latest SAK Status                          : Rx & Tx
Latest SAK AN                              : 1
Latest SAK KI (KN)                        : 3222A4A7678A6BDA553FDB5400000001
(1)
Old SAK Status                             : No Rx, No Tx
Old SAK AN                                 : 0
Old SAK KI (KN)                            : RETIRED (0)
SAK Transmit Wait Time                    : 0s (Not waiting for any peers to
respond)
SAK Retire Time                            : 0s (No Old SAK to retire)
Time to SAK Rekey                          : NA
MKA Policy Name                            : *DEFAULT POLICY*
Key Server Priority                        : 16
Delay Protection                           : FALSE
Replay Window Size                         : 64
Include ICV Indicator                      : FALSE
Confidentiality Offset                     : 0
Algorithm Agility                          : 80C201
SAK Cipher Suite                          : 0080C20001000004 (GCM-AES-XPN-256)
MACsec Capability                          : 3 (MACsec Integrity,
Confidentiality, & Offset)
MACsec Desired                             : YES

# of MACsec Capable Live Peers             : 1
# of MACsec Capable Live Peers Responded : 1

Live Peer List:
MI                        MN     Rx-SCI (Peer)      SSCI    KS-Priority
-----------------------------------------------------------------------------
86B47DE76B42D9D7AB6805F7  113   0257.3fae.5cda/0001   1        16

Potential Peer List:
MI      MN         Rx-SCI (Peer)             SSCI         KS-Priority
-----------------------------------------------------------------------------

Peers Status:
Last Tx MKPDU   : 2018 Mar 01 13:36:56.450
Peer Count      : 1
RxSCI           : 02573FAE5CDA0001
MI              : 86B47DE76B42D9D7AB6805F7
Peer CAK        : Match
Latest Rx MKPDU : 2018 Mar 01 13:36:56.450
```

The sample output shows secured MKA session status, CAK Name (CKN), EAP CA authentication mode, key server state, and MKA cipher suite.

It also shows configured MACsec cipher suite, SAK state, peer list, and Peer CAK match state.

The configuration is verified when the **show dot1x** output shows an authorized port and the MACsec MKA session outputs show a secured session and valid peer status.

14 uRPF Source Address Validation

Topics:

- uRPF source validation mechanisms

Outlines uRPF source address validation methods, detailing validation modes, operational behaviors, configuration tasks, compliance requirements, VRF-specific guidelines, and procedures for verifying uRPF operations across network interfaces.

uRPF source validation mechanisms

Introduces uRPF source validation concepts, describing validation modes, loose and strict mode behaviors, VRF-based source validation, configuration and verification procedures, compliance requirements, and default route handling for comprehensive network security.

A uRPF source validation mechanism is a network security feature that

- checks whether the source IP address of a received packet is listed as reachable in the router's Forwarding Information Base (FIB)
- performs a reverse path lookup in the FIB to ensure the validity of the source, and
- drops packets when the source IP address is not listed in the FIB, reducing the risk of source address spoofing.

Feature history
Table 53: Feature History Table

Feature Name	Release Information	Description
URPF source validation using VRF table	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200]) Modular Systems (8000 [ASIC: Q200])(select variants only*) This feature provides secure traffic validation by supporting high prefix scales and high-speed updates that exceed the capabilities of standard Access Control Lists. It implements a new loose mode Unicast Reverse Path Forwarding (URPF) by using a dedicated URPF Virtual Routing and Forwarding (VRF) table for source lookups. This process enables efficient source validation within existing platform routing limits. This feature introduces these changes: **CLI**: • The **vrf RED urpf-lookup-ipv4** and **vrf RED urpf-lookup-ipv6** keywords are introduced in the **vrf** *vrf-name* command. *This feature is supported on: • Cisco 8202-32FH-M • 88-LC0-36FH • 88-LC0-36FH-M

Feature Name	Release Information	Description
uRPF in Loose Mode	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
uRPF in Loose Mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
uRPF in Loose Mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
uRPF in Loose Mode	Release 7.3.15	When the source IP address of an incoming packet is not present in the Forwarding Information Base (FIB), the router considers it as an invalid packet and drops it. Use the **allow-default** keyword of **ipv4/ipv6 verify unicast source reachable-via** command and configure the default route for the interface so that the router does not drop a packet even when the source IP address is not present in the FIB. The command **ipv4/ipv6 verify unicast source reachable-via** is introduced.

Security behavior

Source address spoofing is a common technique used in Denial-of-Service (DoS) attacks, where forged source IP addresses make attack traffic harder to identify and trace. Attackers may change source IP addresses to avoid detection by service providers, and malware-infected hosts can use multiple forged addresses to flood a device.

uRPF improves security by validating the source IP address of each received packet. If the source is reachable according to the FIB, it is considered valid; otherwise, the router drops the packet.

uRPF validation modes

Provides loose-mode and strict-mode source validation behavior so you can select the correct uRPF validation mode for each router interface design.

The router supports these uRPF validation modes.

Table 54: uRPF validation modes

Mode	Lookup	Packet handling	Use case
Loose mode	The router checks whether a matching FIB entry exists for the source IP address.	The router does not require the source to be learned through the same interface that received the packet.	Use in multihomed service-provider edge networks where legitimate traffic can arrive on alternate interfaces.
Strict mode	The router checks whether the receiving interface is also the interface used to reach the source address.	The router accepts packets only when the source is reachable through the receiving interface; otherwise, it drops the packet.	Use where routing is naturally or explicitly symmetric. Strict mode is supported since Cisco IOS XR Release 7.9.1.

The modes differ by whether the ingress interface must match the route back to the packet source.

Use loose mode for asymmetric paths and strict mode only where source routes are symmetric.

uRPF loose mode

Explains how loose-mode uRPF validates packet sources by checking for a FIB entry while allowing legitimate asymmetric traffic paths in multihomed service-provider edge networks.

A uRPF loose mode is a source validation mode that

- checks whether the source IP address has a matching entry in the FIB
- permits legitimate traffic that uses alternate interfaces to reach the router, and
- validates the source address against the FIB without requiring an ingress-interface match.

Loose-mode packet example

In loose mode, the router receives packets with source address 203.1.113.1 from both HundredGigE0/2/0/2 and HundredGigE0/2/0/3.

When loose mode is configured on the ingress interface, the router checks only whether the source address has a matching FIB entry. The router does not drop the packet when the ingress interface is not listed in the FIB as the outgoing interface for that prefix. This feature is useful in multihomed service-provider edge networks, where legitimate traffic may use alternate paths and interfaces.

Requirement: Comply with interface and address-family configuration for loose-mode uRPF

Outlines the interface and address-family requirements that you must follow before configuring loose-mode uRPF on supported router, subinterface, bundle, and bundle-subinterface connections.

Follow these requirements when you configure uRPF in loose mode:

- Configure loose-mode uRPF only on router interfaces, subinterfaces, bundle interfaces, and bundle subinterfaces.
- Configure both IPv4 and IPv6 commands for uRPF to work.
- Use loose mode for deployments that can have legitimate asymmetric paths, such as multihomed service-provider edge networks.

Configure uRPF loose mode on an interface

Configure loose-mode uRPF on an interface. This allows the router to validate source addresses against the FIB while supporting asymmetric paths in multihomed edge networks.

Enable loose-mode Unicast Reverse Path Forwarding (uRPF) on router interfaces to help prevent IP spoofing while permitting asymmetric routing.

Loose-mode uRPF is typically used in multihomed service-provider edge networks, where traffic paths may be asymmetric. This configuration allows the router to check if the source address exists in the Forwarding Information Base (FIB), even if it was received on a different interface.

Procedure

Configure loose-mode uRPF for IPv4 and IPv6 on the interface.

Example

```
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 verify unicast source reachable-via any
Router(config-if)# ipv6 verify unicast source reachable-via any
Router(config-if)# commit
```

Use the reachable-via any option so the router checks for a source FIB entry without requiring an ingress-interface match.

Loose-mode uRPF is now configured on the selected interface. The router will validate source addresses for both IPv4 and IPv6 against the FIB, allowing for traffic with asymmetric return paths.

Verify uRPF drop counters

Verify CEF drop counters so you can confirm the number of packets dropped because of uRPF source validation on each reported router node.

Confirm that uRPF source validation is dropping packets as expected by checking the CEF drop counters.

Use the show cef drops command after configuring uRPF source validation to see per-node drop statistics.

Before you begin

Ensure you have configured uRPF on the relevant interface.

Procedure

Use the **show cef drops** command to display the uRPF drop counters.

Example

```
Router(config-if)# show cef drops
Node: 0/0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1911
  RPF suppressed drops packets :              0
  RP destined drops    packets :              0
  Discard drops        packets :              0
  GRE lookup drops     packets :              0
  GRE processing drops packets :              0
  LISP punt drops      packets :              0
  LISP encap err drops packets :              0
  LISP decap err drops packets :              0
Node: 0/RP0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1503
```

The output shows RPF drops, RPF suppressed drops, and other CEF drop counters for each node.

The verification is complete when the output displays RPF drop counters for the expected nodes.

Configure default route handling for loose-mode uRPF

Configure loose-mode uRPF with the allow-default option so your router uses a default route during source IP address verification for selected interfaces. This lets packets from sources not listed in the FIB use the default route, supporting more flexible IP verification.

Configure default route handling for loose-mode uRPF to enable the use of a default route during source address verification.

Loose-mode uRPF requires the source IP address of a packet to appear in the FIB. The allow-default option enables the router to use the default route in the source IP address verification process, allowing for greater flexibility in routing.

Before you begin

- Configure a default route for the interface before using the allow-default option.
- Without allow-default, the router drops packets when the source address is not listed in the FIB table.
- With allow-default, the router still drops packets if the default route is not configured for the interface.
- On any VRF interface, loose-mode uRPF with allow-default applies to all interfaces in that router VRF.

Procedure

Configure loose-mode uRPF with default-route source verification.

Example

```
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 verify unicast source reachable-via any allow-default
Router(config-if)# ipv6 verify unicast source reachable-via any allow-default
Router(config-if)# commit
```

Use the allow-default option when the default route must participate in the source IP address verification process.

Default-route handling is configured when the IPv4 and IPv6 reachable-via any allow-default commands are committed under the selected interface.

uRPF strict mode behaviors

Explains how strict-mode uRPF validates that incoming packets arrive on the same interface the router uses to reach the source address.

A uRPF strict mode behavior is a source validation mechanism that

- checks the source address in the Forwarding Information Base (FIB)
- forwards packets arriving on the same interface the router uses to reach the packet's source address, and
- drops packets that arrive on any other interface.

Table 55: Feature History Table

Feature Name	Release Information	Feature Description
uRPF in Strict Mode	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
uRPF in Strict Mode	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
uRPF in Strict Mode	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
uRPF in Strict Mode	Release 7.9.1	You can protect the router against DoS attacks with spoofed source IP addresses by enabling the Strict mode in uRPF. When this feature is enabled, the router accepts the incoming packet only if the source IP address of the packet is present in its routing table and if the source IP address of the input packet is reachable via the interface on which the packet has been received. If not, the router drops the packet.In earlier releases IOS XR supports only loose mode uRPF. This feature introduces the **hw-module profile cef unipath-surpf** command. This feature modifies the **ipv4/ipv6 verify unicast source reachable-via** command.

Use this information before enabling strict mode on supported hardware.

Additional reference information

Strict-mode uRPF is supported on select Cisco platforms, introduced across various releases with specific command changes. Platform and command support details are essential before enabling strict mode.

Use this information before enabling strict mode on supported hardware.

In a strict-mode scenario, R1 uses the interface HundredGigE0/2/0/3 as the egress path for network 203.0.113.0/24. R1 receives packets with source address 203.1.113.1 through interfaces HundredGigE0/2/0/2 and HundredGigE0/2/0/3. R1 accepts the packet received on HundredGigE0/2/0/3 because it is the correct path to the source according to the FIB. Conversely, R1 drops the packet arriving on HundredGigE0/2/0/2 because it is not the FIB route to the source. Where multiple egress interfaces exist for a network, the router checks all entries before dropping the packet.

Requirement: Follow strict-mode uRPF rules

Outlines the key configuration, interface, traffic-family, reload, default-state, and route-symmetry rules you must follow before configuring strict-mode uRPF on supported router interfaces.

Follow these rules when you configure uRPF in strict mode.

- Configure strict mode only on router interfaces, subinterfaces, bundle interfaces, and bundle subinterfaces.
- Do not configure strict mode on tunnel interfaces or BVI interfaces.
- Configure both IPv4 and IPv6 traffic types for uRPF to work.
- The router disables strict mode by default.
- Use strict mode only where routing has natural or configured symmetry.
- Avoid strict mode on internal interfaces where route asymmetry is likely.
- Reload the router after executing the hw-module profile cef unipath-surpf command.
- Use the allow-default option only when traffic is expected to arrive through default routes.

Configure uRPF strict mode on an interface

Configure strict-mode uRPF so the router validates that packets arrive on the same interface used to reach the source address after reload.

Enable strict-mode uRPF on the router interface to ensure packets are only accepted if they arrive through the interface used to reach the packet's source address.

Strict-mode uRPF requires symmetry between the packet's ingress interface and the route back to the source. This helps mitigate spoofed IP addresses by ensuring correct routing paths.

Before you begin

Review and follow the requirement: strict-mode uRPF rules.

Procedure

Enable the strict-mode uRPF profile and configure IPv4 and IPv6 strict source validation on the interface.

Example

```
Router(config)# hw-module profile cef unipath-surpf enable
Router(config)# interface HundredGigE 0/2/0/2
Router(config-if)# ipv4 address 10.0.0.1 255.255.255.0
Router(config-if)# ipv4 verify unicast source reachable-via rx
Router(config-if)# ipv6 address 2001::1/64
Router(config-if)# ipv6 verify unicast source reachable-via rx
Router(config-if)# commit
Router(config-if)# exit
Router(config)# reload
```

The configuration includes the hw-module profile cef unipath-surpf enable command and a router reload.

Strict-mode uRPF is now enforced on the interface. The router only accepts packets whose source address is reachable via the same interface on which they arrive.

Verify uRPF strict mode configuration

Verify strict-mode uRPF running configuration and CEF drop counters after the router reload completes and source validation is active on the interface.

Confirm that strict-mode uRPF is properly configured and operating as intended.

Use these commands after configuring strict mode and reloading the router to ensure source validation is active.

Before you begin

Configure strict-mode uRPF and reload the router before you verify the configuration.

Procedure

Use the **show cef drops** command to isplay the uRPF drop counters.

Example

```
Router(config-if)# show cef drops
Node: 0/0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1911
  RPF suppressed drops packets :              0
  RP destined drops    packets :              0
  Discard drops        packets :              0
  GRE lookup drops     packets :              0
  GRE processing drops packets :              0
  LISP punt drops      packets :              0
  LISP encap err drops packets :              0
  LISP decap err drops packets :              0
Node: 0/RP0/CPU0
  Unresolved drops     packets :              0
  Unsupported drops    packets :              0
  Null0 drops          packets :              0
  No route drops       packets :              2
  No Adjacency drops   packets :              0
  Checksum error drops packets :              0
  RPF drops            packets :              1503
```

Verify that the output shows RPF drops, RPF suppressed drops, and other CEF drop counters for each node.

The strict-mode uRPF configuration is verified when the running configuration displays reachable-via rx and the CEF output displays RPF drop counters.

uRPF VRF source address validation

Explains how uRPF VRF source address validation performs source lookups against a selected VRF table instead of the interface VRF.

A uRPF VRF source address validation is a source verification feature that

- uses a selected uRPF VRF table rather than the interface's own VRF
- centralizes permitted source prefixes in one VRF, and
- supports scalable and flexible network architectures.

Benefits

uRPF VRF source address validation provides these benefits.

- Scales to hundreds of thousands of source prefixes and massive ECMP environments.
- Reduces the need for complex per-interface ACLs and lowers TCAM consumption.
- Simplifies management of prefix lists and source policies in large tenant environments.

Requirement: Follow uRPF VRF source validation guidelines

Outlines the platform, BGP route-leak, interface, VRF, address-family, and verification guidelines that support uRPF VRF source validation on Cisco 8000 systems.

Follow these guidelines when you configure uRPF source address validation using a VRF table:

- Use Cisco 8000 Series routers and line cards based on Cisco Silicon One Q200 ASIC systems: Cisco 8202-32FH-M, 88-LC0-36FH, and 88-LC0-36FH-M.
- Use standard Border Gateway Protocol (BGP) Route Policy Language (RPL) to leak routes into the uRPF VRF.
- Expect the router to perform the source lookup in the uRPF VRF only when the interface has uRPF enabled.
- Use a dedicated VRF for uRPF validation.
- Configure only one uRPF VRF on the system to prevent disruptions and comply with platform limitations.
- Enable both IPv4 and IPv6 uRPF in the VRF, because enabling only one address family is not supported.
- Enable uRPF on interfaces that face untrusted or external sources.
- Verify configuration with operational commands such as **show rsi vrf** *vrf_name*.

Requirement: Follow uRPF VRF source validation restrictions

Outlines the restrictions that apply to uRPF private lookups, OpenConfig, drop counters, connected addresses, BGP, and address families before deployment. Make sure to review these requirements for compliant configuration.

Follow these restrictions when you configure uRPF source address validation using a VRF table:

- Use only one VRF for uRPF private lookups.
- Do not use OpenConfig models for uRPF configuration or counters.
- Do not expect drop counters at the interface level.
- Import all connected interface IP addresses into the uRPF VRF.
- Ensure IP addresses are present in the uRPF VRF for protocols such as BGP on uRPF-enabled connected interfaces.
- Do not enable uRPF for only one address family, because the router rejects configurations that enable only one address family.

How uRPF source address validation works

Describes how RSI components, FIB processing, BGP route imports, and router validation work together to perform uRPF VRF source validation for incoming unicast traffic.

uRPF source address validation centralizes permitted source prefixes in a dedicated VRF. Correct operation requires VRF selection, route leakage, and uRPF enablement on the appropriate interfaces.

Summary

The key components involved in the process are:

- User: Configures the uRPF VRF, manages route imports, and enables uRPF on interfaces.
- Router: Validates packet sources against the uRPF VRF and enforces forwarding or dropping as appropriate.

- RSI control process: Manages VRF configurations on the Route Processor.
- RSI agent: Receives VRF updates, communicates with platform layers, and notifies system clients.

This process enables selective filtering of incoming unicast traffic by validating source addresses against routes in a specific VRF.

Workflow

These stages describe how uRPF source address validation works.

1. The user configures the uRPF VRF, imports routes, and enables uRPF on the relevant interfaces.
2. The RSI control process handles the VRF configuration and assigns a table identifier.
3. The RSI control process communicates VRF information and necessary flags to the RSI agent.
4. The RSI agent notifies the FIB Platform Independent layer about required uRPF lookups.
5. The FIB Platform Independent layer updates the hardware to enable uRPF lookups in the specified VRF.
6. Standard BGP policies populate the uRPF VRF with the necessary routes.
7. The router validates the source address of incoming unicast packets against entries in the uRPF VRF and forwards or drops packets accordingly.

Result

uRPF source address validation using VRF tables enables dynamic, scalable, and flexible filtering of source addresses for unicast traffic.

Configure uRPF source address validation with a VRF table

Configure uRPF lookup on a specific VRF so the router can validate IPv4 and IPv6 packet sources against the dedicated VRF table.

Set up uRPF source address validation using a VRF table to ensure that IPv4 and IPv6 packets are checked against the designated VRF table.

This task enables source address validation for both IPv4 and IPv6 on the specified VRF by activating the uRPF lookup flags.

Before you begin

- Follow the relevant requirements and restrictions for uRPF VRF source validation before starting. See:
  - Requirement: Follow uRPF VRF source validation guidelines
  - Requirement: Follow uRPF VRF source validation restrictions

Procedure

1. Enter global configuration mode.

   Example

   ```
   Router# configure
   ```

2. Enter configuration submode for the target VRF.

Example

```
Router(config)# vrf RED
```

3. Enable IPv4 and IPv6 uRPF lookup flags.

   Example

   ```
   Router(config-vrf)# urpf-lookup-ipv4
   Router(config-vrf)# urpf-lookup-ipv6
   ```

   Both address families must be enabled. A one-address-family configuration is not supported.

4. Commit the configuration.

   Example

   ```
   Router(config-vrf)# commit
   ```

5. Verify the VRF configuration.

   Example

   ```
   Router# show rsi vrf all
   Router# show rsi vrf RED
   ```

   Use these commands to confirm that the uRPF lookup flags are enabled for both IPv4 and IPv6.

---

uRPF source address validation is successfully configured when the VRF shows the IPv4 and IPv6 uRPF lookup flags in the RSI verification output.

15 Type 6 Password Encryption

Topics:

- Type 6 password encryption

Outlines Type 6 password encryption concepts, operational workflow, and implementation requirements, guiding users through key activation after iPXE boot, encryption processes, maintenance best practices, and secure BGP session configuration with appropriate key management.

Type 6 password encryption

Introduces Type 6 password encryption, covering prerequisites such as enabling the primary key after iPXE boot and detailing encryption operation, configuration tasks, and key management principles for secure network authentication.

A Type 6 password encryption mechanism is a security feature that

- uses a reversible 128-bit Advanced Encryption Standard (AES) algorithm to securely store plain-text key strings
- allows devices to decrypt the encrypted strings to their original plain-text form, and
- supports secure storage and authentication for network session protocols.

Additional reference information

Type 6 password encryption supports secure storage of plain-text key strings used to authenticate BGP, IP SLA, IS-IS, MACsec, OSPF, and RSVP sessions.

Requirement: Enable the Type 6 primary key after an iPXE boot

Outlines the requirement to enable the Type 6 primary key following an iPXE boot to ensure encrypted password functionality and maintain network security.

After an iPXE boot, enable the Type 6 primary key using the **key config-key password-encryption** command.

How Type 6 password encryption works

Details the operational process of Type 6 password encryption, including procedures for enabling encryption, configuring key chains for BGP sessions, creating encrypted BGP sessions, and emphasizes best practices for key updates and consistent key usage across routers.

Type 6 password encryption is commonly used to secure routing protocol sessions, such as BGP, by ensuring that authentication keys remain confidential. While this example focuses on BGP between routers R1 and R2, the process also applies to OSPF, IS-IS, and other protocols.

Summary

The key components involved in the process are:

- Type 6 server: Creates and manages the primary key used for encrypting client key strings.
- Type 6 client: Utilizes the primary key to protect authentication key strings in key chains.
- Routing protocol: Uses the configured key chain to authenticate sessions between routers.

Type 6 password encryption provides enhanced session security by encrypting authentication keys used in routing protocol sessions.

For MACsec authentication, refer to the Secure MACsec encryption chapter in the *MACsec Configuration Guide for Cisco 8000 Series Routers, Cisco IOS XR Releases*.

Workflow

The process describes how Type 6 password encryption works.

1. Create the primary key and enable Type 6 password encryption on both routers. The primary key is stored internally and is not displayed in the running configuration.
2. Configure a key chain as a Type 6 client and assign the encrypted key string, cryptographic algorithm, and key lifetimes.
3. Create the BGP session on both routers with the same key-chain name, then verify that the BGP neighbor state is Established.

Result

The process protects protocol-session key strings with Type 6 encryption and enables routers to authenticate the BGP session securely. This improves session confidentiality and applies to other protocol types as well.

Enable Type 6 password encryption

Configure, modify, delete, and verify the Type 6 primary key and encryption feature for secure password management.

Set up and manage Type 6 password encryption on Cisco IOS XR routers, ensuring secure storage and operation of encrypted keys.

Perform this task to enable Type 6 password encryption server functions before configuring Type 6 clients (such as key chains). Type 6 encryption improves security by storing the primary encryption key internally and encrypting all plain-text key strings.

Before you begin

- Ensure that you can access XR EXEC and configuration modes on the routers.
- Schedule a primary-key update during a maintenance window.

Procedure

1. Create the Type 6 primary key.

   Example

   ```
   Router# key config-key password-encryption
   Fri Jul 19 12:22:45.519 UTC
   New password Requirements: Min-length 6, Max-length 64
   Characters restricted to [A-Z][a-z][0-9]
   Enter new key :
   Enter confirm key :
   Master key operation is started in background
   ```

   The primary key is the password or key that encrypts all plain-text key strings in the router configuration. An AES symmetric cipher performs the encryption.

   The primary-key operation runs in the background. Use **show type6 server** to view the operation status.

   The router stores the primary key internally rather than in the router configuration. The running configuration does not display or provide access to the primary key.

2. Enable Type 6 password encryption.

   Example

   ```
   Router(config)# password6 encryption aes
   Routerconfig)# commit
   Fri Jul 19 12:22:45.519 UTC
   ```

3. Modify the Type 6 primary key.

   Example

   ```
   Router# key config-key password-encryption
   New password Requirements: Min-length 6, Max-length 64
   Characters restricted to [A-Z][a-z][0-9]
   ```

```
Enter old key :
Enter new key :
Enter confirm key :
Master key operation is started in background
```

Enter the old key, new key, and confirmation key when prompted. The operation updates the key chain and other Type 6 clients.

4. Delete the Type 6 primary key.

Example

```
Router# configure
Router(config)# no password6 encryption aes
Router(config)# commit
Router(config)# exit
Router# key config-key password-encryption delete
WARNING: All type 6 encrypted keys will become unusable
Continue with master key deletion ? [yes/no]:yes
Master key operation is started in background
```

> **Note**
>
> Deleting the primary key makes all Type 6 encrypted keys unusable.

5. Use the **show type6 server** command to verify the primary-key and Type 6 feature state.

Example

```
Router# show type6 server
Fri Jul 19 12:23:49.154 UTC
Server detail information:
==========================================
AES config State       :       Enabled
Masterkey config State :       Enabled
Type6 feature State    :       Enabled
Master key Inprogress  :       No
Masterkey Last updated/deleted : Mon Nov 4 15:55:11 2024
Masterkey Size (bits) : 48
Masterkey Hash (Algo:sha512,Format:base64) :
ujJTh2rta8ItSm/1PYQGxq2GQZXtFEq1yHYhtsIztUi66uaVbfNG7IwX9eoQ817jy8UUeX7X3dMUVGTioLq0Ew
```

The primary-key configuration and Type 6 feature state display as Enabled.

The operation-status field displays No when the primary-key activity is complete.

The three states display Disabled when you disable the primary key.

6. Use the **show type6 trace server all** command to verify Type 6 server trace details.

Example

```
Router# show type6 trace server all
Fri Jul 19 12:26:05.111 UTC
Client file lib/type6/type6_server_wr
```

```
25 wrapping entries (18496 possible, 64 allocated, 0 filtered, 25 total)
Jul 19 09:59:27.168 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 ***** Type6
server process started Respawn count (1) ****
...
...
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 User has started
 Master key operation (CREATE)
Jul 19 12:22:59.908 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Created Master
 key in TAM successfully
Jul 19 12:23:00.265 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key
Available set to (AVAILABLE)
Jul 19 12:23:00.272 lib/type6/type6_server_wr 0/RP0/CPU0 t7145 Master key
inprogress set to (NOT INPROGRESS)
```

7. Check the primary-key update status.

   Example

   ```
   Router# show type6 masterkey update status
   Thu Sep 17 06:48:56.595 UTC
   Type6 masterkey operation is NOT inprogress
   ```

   Example

   ```
   Router# show type6 masterkey update status
   Thu Sep 17 06:50:07.980 UTC
   Type6 masterkey operation is inprogress

   Masterkey upate status information:
   Client Name                Status
   ===================================
   keychain                   INPROGRESS
   ```

   Starting with Cisco IOS XR Software Release 7.0.14, use the primary-key update-status command to display the update status.

   Before that release, use **show type6 clients** for the same purpose.

8. Clear a Type 6 client state if a primary-key update is stuck.

   If the operation-status field displays YES, use the primary-key update-status command to identify the incomplete client.

   Before Release 7.0.14, use **show type6 clients** for the same purpose. Clear that client with **clear type6 client** in XR EXEC mode.

The Type 6 primary key and encryption feature are configured, and you can verify server and client status to ensure secure operation.

Caution: Schedule primary-key updates during a maintenance window

Outlines the network disruption, persistence, and rollback consequences that you must carefully consider before changing the Type 6 primary key.

Schedule a Type 6 primary-key update during a maintenance window.

Updating the Type 6 primary key changes the key chain and affects other clients using Type 6. If the router fails during the operation, it can disrupt the product network and routing protocol sessions. The primary key is not saved in the running

configuration, but changes persist across reloads. You cannot use the rollback configuration command to restore the previous key.

Configure a Type 6 key chain for BGP sessions

Configure a Type 6 client key chain for BGP session authentication.

Set up a Type 6 client key chain to authenticate BGP sessions using encrypted keys.

Use this procedure after enabling Type 6 password encryption. To ensure consistent authentication, configure the same key string on all routers participating in the BGP session.

Before you begin

- Enable the Type 6 primary key and encryption feature.
- Choose the same key-chain name and key string for all routers in the session.

Procedure

1. Enter the key-chain configuration mode.

   Example

   ```
   Router# configure
   Router(config)# key chain my-test-keychain
   Router(config-type6_password)# key 1
   ```

   The key chain in this example is named **my-test-keychain**, and the key identifier is 1.

2. Configure the encrypted key string, cryptographic algorithm, key lifetimes, and commit the configuration.

   Example

   ```
   Router(config-type6_password-1)# key-string password6
   6664496443695544484a4448674b695e685d56565d676364554b64555f4c5c645b
   Router(config-type6_password-1)# cryptographic-algorithm HMAC-MD5
   Router(config-type6_password-1)# accept-lifetime 1:00:00 october 24 2005
   infinite
   Router(config-type6_password-1)# send-lifetime 1:00:00 october 24 2005 infinite
   Router(config-type6_password-1)# commit
   ```

   Type 6 password encryption encrypts plain-text keys.

   Enter a plain-text key string in alphanumeric form.

   When you enable Type 6 password encryption for MACsec, enter the key string in hexadecimal format.

   Using the key-string command, you can enter the password in cleartext format or Type 6 encrypted format, as in this example.

   BGP supports only HMAC-MD5 and HMAC-SHA1-12.

3. Use the **show key chain trace server both** command to verify the key-chain trace information.

   Example

   ```
   Router# show key chain trace server both
   Sat Jul 20 16:44:08.768 UTC
   Client file lib/kc/kc_srvr_wr
   ```

```
4 wrapping entries (18496 possible, 64 allocated, 0 filtered, 4 total)
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 *********kc_srvr process
 started*********
Jul 20 16:43:26.342 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Cerrno DLL
registration successfull
Jul 20 16:43:26.349 lib/kc/kc_srvr_wr 0/RP0/CPU0 t312 (kc_srvr) Initialised
sysdb connection
Jul 20 16:43:26.612 lib/kc/kc_srvr_wr 0/RP0/CPU0 t317 (kc_srvr_type6_thread)
 Succesfully registered as a type6 client
```

4. Use the **show key chain type6_password** command to verify the Type 6 key-chain configuration.

   Example

```
Router# show key chain type6_password
Sat Jul 20 17:05:12.803 UTC
Key-chain: my-test-keychain -
  Key 1 -- text
"6664496443695544484a4448674b695e685d56565d676364554b64555f4c5c645b"
    Cryptographic-Algorithm -- HMAC_MD5
    Send lifetime --  01:00:00, 24 Oct 2005 - Always valid  [Valid now]
    Accept lifetime --  01:00:00, 24 Oct 2005 - Always valid [Valid now]
```

   Verify Type 6 client information.

   Use these associated commands:

   {ul([syn("key chain"), syn("key-string password6"), syn("show key chain trace server both")])}

The Type 6 client key chain contains the encrypted key string and valid send and accept lifetimes for BGP authentication.

Requirement: You must use the same Type 6 key string on all BGP routers

Outlines the requirement to configure a matching Type 6 key string on every router that participates in the BGP session.

To comply with BGP security requirements, ensure that you configure the same Type 6 key string on every router that participates in the BGP session. Using a matching key string across all routers guarantees secure and consistent authentication throughout the BGP network.

Create a BGP session with Type 6 password encryption

Configure and verify an internal BGP (iBGP) session between two routers using a Type 6 key chain for neighbor authentication.

Set up and secure an iBGP session that uses a Type 6 key chain for password encryption between routers.

This example demonstrates configuring an iBGP session between two routers (Router and Router) with Type 6 password encryption. For details on complete iBGP topologies, refer to Cisco 8000 Series Routers documentation.

Before you begin

- Configure the Type 6 primary key and client key chain on both routers.
- Use the same session parameters and key-chain name on both routers.

Procedure

---

1. Configure the BGP session on Router.

Example

```
Router# configure
Router(config)# router bgp 65537
```

Example

```
Router (config-bgp)# neighbor 10.1.1.11 remote-as 65537
Router(config-bgp)# keychain my-test-keychain
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp)# commit
```

Use the same key-chain name for the BGP session and Type 6 encryption. This example uses **my-test-keychain**.

2. Configure the BGP session on Router.

Example

```
Router(config)# router bgp 65537
Router(config-bgp)# neighbor 10.1.1.1 remote-as 65537
Router(config-bgp)# keychain my-test-keychain
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp)# commit
```

Repeat the session configuration on Router and use the same session and key chain as Router.

3. Use the **show bgp sessions** to verify that the BGP neighbor state is Established.

Example

```
Router# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.11     default  0      65537   0    0     Established  None
```

Example

```
Router# show bgp sessions
Neighbor      VRF      Spk    AS      InQ  OutQ  NBRState     NSRState
10.1.1.1      default  0      65537   0    0     Established  None
```

Use these associated commands:

{ul([syn("session-group"), syn("show bgp sessions")])}

The iBGP session between R1 and R2 successfully establishes using the Type 6 key chain. Both routers display their neighbor state as "Established."

16 Management Plane Protection

Topics:

- Management plane protection features

Outlines management plane protection features, requirements, restrictions, interface types, configuration workflows, and verification examples to guide secure and efficient management plane operation in network environments.

Management plane protection features

Introduces management plane protection features, covering task permission requirements, behavior restrictions, inband and out-of-band interfaces, peer filtering, protection mechanisms, management planes, key benefits, configuration processes, interface setup tasks, example configurations, and supplemental references.

A Management Plane Protection (MPP) feature is a security feature that

- restricts the interfaces on which network management packets can enter a device
- lets an operator designate one or more router interfaces as management interfaces, and
- causes only designated management interfaces to accept management traffic destined for the device.

Requirement: Verify management-plane task permissions

Ensures that your user group and task group include the task IDs required to use management lane protection commands.

Verify that your user group is associated with a task group that includes the required task IDs before you configure management plane protection. This ensures that you have the necessary permissions to use all management plane protection commands.

Restriction: Account for management plane protection restrictions

Outlines the restrictions and traffic-filtering behaviors that apply when Management Plane Protection (MPP) is configured.

Account for the following Management Plane Protection (MPP) restrictions whenever you design, enable, modify, or troubleshoot the management-plane configuration:

- MPP does not track denied or dropped protocol requests.
- MPP configuration does not enable protocol services; it only makes enabled services available on different interfaces. Protocols must be explicitly enabled.
- Management requests received on inband interfaces are not necessarily acknowledged on the same interface.
- Route Processor (RP) interfaces are designated as out-of-band interfaces by default and can be configured under MPP.
- MPP configuration changes do not affect active sessions established before the changes.
- MPP controls only incoming management requests for specific protocols (such as TFTP, Telnet, SNMP, SSH, and HTTP).
- MPP does not support MIB.
- In an MPLS L3VPN, if a VRF interface is attached under MPP, it applies the VRF filter on incoming interfaces through LPTS. Incoming packets from a core interface with a different VRF are not allowed by MPP.

Management plane protection

Explains the key attributes and operational principles of management plane protection before you enable the feature.

A management plane protection feature is a security mechanism that

- distinguishes inband interfaces from out-of-band interfaces
- enables peer filtering by clarifying the relationship between the control and management planes, and
- supports, enables, and filters management protocols for improved network defense.

Additional reference information

Before you enable the MPP feature, review the concepts in the related child topics to fully understand configuration options, interface types, and protocol support.

Inband management interfaces

Explains the role of inband management interfaces in processing management and data-forwarding packets.

An inband management interface is a Cisco IOS XR physical or logical interface that

- processes management packets
- processes data-forwarding packets, and
- processes packets as a shared management interface.

Out-of-band management interfaces

Defines an out-of-band management interface and explains how it isolates management traffic from forwarding traffic.

An out-of-band management interface is an interface that

- allows only management protocol traffic to be forwarded or processed
- prevents forwarding or customer traffic from interfering with router management and significantly reduces the possibility of denial-of-service attacks, and
- forwards traffic between out-of-band interfaces, terminates management packets destined for the router, and participates in dynamic routing protocols.

Additional reference information

The service provider connects to the router's out-of-band interfaces and builds an independent overlay management network with all the routing and policy tools that the router can provide.

Peer filters on interfaces

Explains how peer filters limit management traffic to specific peers or peer ranges.

A peer filter is an interface option that

- allows management traffic from a specific peer
- allows management traffic from a range of peers, and
- provides a more specific source restriction than allowing all peers.

Control plane protection

Explains the structure and function of the control plane and describes how MPP operates within it.

A control plane is a collection of route-processor processes that

- runs at the process level
- provides high-level control for most Cisco IOS XR software functions, and
- handles traffic directly or indirectly destined for the router while providing the Control Plane Infrastructure in which MPP operates.

Management planes

Defines the management plane and identifies the protocols and access activities that it supports.

A management plane is a layer of a communication architecture that

- serves as one of three planes responsible for managing network devices
- coordinates functions among the management, control, and data planes, and
- supports device monitoring and command-line interface (CLI) access through management protocols.

Additional reference information

Examples of protocols processed in the management plane include Simple Network Management Protocol (SNMP), Telnet, HTTP, Secure HTTP (HTTPS), and Secure Shell (SSH). These management protocols are used for device monitoring and CLI access. Restricting access to devices to internal sources (trusted networks) is critical.

Management plane rotection feature

Explains the default state, interface behavior, filtering behavior, and supported protocols of MPP.

The MPP features are management-traffic protection features that

- are disabled by default, along with their management protocols, and are enabled when an interface is configured as inband or out-of-band
- allow management traffic by default only on RP and standby RP Ethernet interfaces and on interfaces manually configured for MPP, while all other interfaces drop management packets destined for the device, and
- support SSHv2, all SNMP versions, Telnet, TFTP, HTTP, and HTTPS, and permit modification or deletion of management interfaces after configuration.

Additional reference information

- If MPP is disabled and a protocol is activated, all interfaces can pass management traffic.
- Logical interfaces, and other interfaces that are not present on the data plane, filter packets based on the ingress physical interface.
- MPP supports the following management protocols:
  - SSHv2
  - SNMP, all versions
  - Telnet
  - TFTP
  - HTTP
  - HTTPS

Key benefits of management plane protection

Lists the benefits of restricting management traffic to designated management plane protection interfaces.

MPP provides these benefits:

- Provides greater access control for managing a device than allowing management protocols on all interfaces.
- Improves performance for data packets on non-management interfaces.

- Supports network scalability.
- Simplifies the task of using per-interface access control lists (ACLs) to restrict management access to the device.
- Reduces the number of ACLs needed to restrict access to the device.
- Prevents packet floods on switching and routing interfaces from reaching the CPU.

How management plane protection works

Describes the stages for configuring and verifying inband and out-of-band MPP interfaces on a Cisco 8000 Series Router.

Configuring MPP helps prevent unauthorized access to the router's management functions by restricting which interfaces and peers can send management traffic. Inband refers to management traffic carried along with regular data, while out-of-band uses dedicated management-only interfaces and VRFs.

Summary

The key components involved in the process are:

- Operator: Designates and configures inband or out-of-band management interfaces.
- MPP: Controls incoming management requests for protocols such as SSHv2, SNMP, Telnet, TFTP, HTTP, and HTTPS.
- Peer filtering mechanism: Limits management traffic to specific peers or peer ranges for enhanced security.

MPP ensures secure access to management functions on a Cisco 8000 Series Router by separating inband and out-of-band management interfaces and limiting which interfaces accept management traffic.

Workflow

These stages describe how management plane protection works.

1. Configure the inband management interface, enable the required protocol, optionally restrict the protocol to a peer or peer range, and commit the configuration.
2. Configure the out-of-band VRF and management interface, enable the required protocol, optionally restrict the protocol to a peer or peer range, and commit the configuration.
3. Verify the resulting MPP interface or VRF state with the appropriate **show mgmt-plane** command.

Result

MPP accepts management packets only on the default or explicitly configured management interfaces. All other interfaces drop management packets destined for the device, enhancing the router’s security.

Configure an inband management plane protection interface

Configure an inband Management Plane Protection interface to permit Telnet management traffic from the IPv4 peer range 10.1.0.0/16.

Enable Telnet management access for a specific IPv4 range on an inband interface.

Use this task on a newly added or already operating device. An inband management interface processes both management and data-forwarding packets. In this example, fourHundredGigE 0/0/0/0 is configured as an inband interface to allow Telnet from the specified peer range.

Before you begin

Ensure that your user group includes the task IDs required for the configuration commands.

If configuring an inband MPP interface in a non-default VRF, perform these additional steps:

- Configure the interface under the non-default inband VRF.

- Configure the global inband VRF.
- For Telnet, configure the Telnet VRF server for the inband VRF.

Procedure

1. Specify the interface to configure as an inband interface and allow Telnet protocol on the inband interface.

   Example

   ```
   Router#configure terminal
   Router(config)#control-plane
   Router(config-ctrl)#management-plane
   Router(config-mpp)#inband
   Router(config-mpp-inband)#interface fourHundredGigE 0/0/0/0
   Router(config-mpp-inband-if)#allow telnet peer
   Router(config-telnet-peer)#address ipv4 10.1.0.0/16
   Router(config-telnet-peer)#commit
   ```

   FourHundredGigE 0/0/0/0 is configured as an inband interface. Use the **interface all** command form to configure all interfaces as inband interfaces.

   Telnet is configured on the inband interface. Use the **allow all** command form to enable all protocols.

2. Verify the inband interface configuration.

   Running Configuration

   The following is a sample output of the **show mgmt-plane** command for the inband interface fourHundredGigE 0/0/0/0.

   Example

   ```
   Router# show mgmt-plane inband interface fourHundredGigE 0/0/0/0

   interface - fourHundredGigE 0/0/0/0
           telnet configured -
                   peer v4 allowed - 10.1.0.0/16
   ```

The specified inband interface accepts Telnet management traffic from the configured IPv4 peer range (10.1.0.0/16).

Configure an out-of-band management plane protection interface

Configure an out-of-band MPP interface under the VRF target, permitting TFTP management traffic from a specified IPv6 peer address. This task helps ensure only authorized management traffic is processed, increasing device security and isolating management functions.

Configure an out-of-band interface in VRF target to accept TFTP management traffic from the IPv6 peer address 33::33.

Use this task for devices that have just been added to the network or are already operational. An out-of-band interface only forwards or processes management protocol traffic, providing isolation from production traffic. In this example, fourHundredGigE 0/0/0/3 is set up for VRF target and permits TFTP traffic from a specific IPv6 peer.

Before you begin

Ensure that your user group includes the task IDs required for the configuration commands.

The following preliminary tasks may be needed:

- Configure the interface under the out-of-band VRF.
- Configure the global out-of-band VRF.
- For a specific protocol, configure the protocol VRF server for the out-of-band VRF.

Procedure

1. Configure out-of-band management plane protection on an interface.

   Example

   ```
   Router#configure terminal
   Router(config)#control-plane
   Router(config-ctrl)#management-plane
   Router(config-mpp)#out-of-band
   Router(config-mpp-outband)#vrf target
   Router(config-mpp-outband)#interface fourHundredGigE 0/0/0/3
   Router(config-mpp-outband-if)#allow tftp peer
   Router(config-tftp-peer)#address ipv6 33::33
   Router(config-tftp-peer)#commit
   ```

   FourHundredGigE 0/0/0/3 is configured as an out-of-band interface for VRF target. Use the **interface all** command form to configure all interfaces as out-of-band interfaces.

   TFTP is configured on the out-of-band interface. Use the **allow all** command form to enable all protocols.

2. Verify the out-of-band VRF configuration.

   Running Configuration

   The following is a sample output of the **show mgmt-plane out-of-band vrf** command.

   Example

   ```
   Router# show mgmt-plane out-of-band vrf
   Management Plane Protection -
           out-of-band VRF - target
   ```

The specified out-of-band interface accepts TFTP management traffic exclusively from the configured IPv6 peer (33::33), enhancing the control and security of network management functions.

Examples of management plane protection configurations

Provides several examples of complete inband and out-of-band management plane protection configurations.

MPP allows administrators to restrict access to the device's management plane, enhancing security by controlling which interfaces can accept management traffic. The following examples illustrate both inband and out-of-band configuration scenarios for MPP.

Management plane protection configuration and verification example

Provides a complete inband and out-of-band MPP configuration and displays verification output for reference.

Use this example as a reference when configuring inband and out-of-band interfaces under Management Plane Protection.

Configuration

The example configures inband and out-of-band interfaces under MPP.

```
configure
 control-plane
  management-plane
   inband
    interface all
     allow SSH
     !
    interface fourHundredGigE 0/0/0/0
     allow all
     allow SSH
     allow Telnet peer
      address ipv4 10.1.0.0/16
     !
    !
    interface fourHundredGigE 0/0/0/0
     allow Telnet peer
      address ipv4 10.1.0.0/16
     !
    !
   !
   out-of-band
    vrf target
    interface fourHundredGigE 0/0/0/3
     allow TFTP peer
      address ipv6 33::33
     !
    !
   !
  !
 !
!
```

Verification output

The following output displays the configured management interfaces, allowed peers, and out-of-band VRF.

```
show mgmt-plane

Management Plane Protection

inband interfaces
-----------------------

interface - fourHundredGigE 0/0/0/0
        ssh configured -
                All peers allowed
        telnet configured -
                peer v4 allowed - 10.1.0.0/16
        all configured -
                All peers allowed
interface - fourHundredGigE 0/0/0/0
        telnet configured -
                peer v4 allowed - 10.1.0.0/16

interface - all
        all configured -
                All peers allowed
outband interfaces
```

```
----------------------
interface - fourHundredGigE 0/0/0/3
        tftp configured -
                peer v6 allowed - 33::33

show mgmt-plane out-of-band vrf

Management Plane Protection -
        out-of-band VRF - target
```

Additional references

Lists related MPP documentation, standards, MIBs, RFCs, and technical assistance resources.

Use these references to find MPP commands, MIB information, standards and RFC status, and Cisco Technical Assistance.

The following sections provide references related to implementing Management Plane Protection.

Related Documents

Related topic	Document title
MPP commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Management Plane Protection Commands* on *System Security Command Reference for Cisco 8000 Series Routers*.

Standards

Standards	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	–

MIBs

MIBs	MIBs link
–	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
No new or modified RFCs are supported by this feature.	–

Technical Assistance

Description	Link
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	http://www.cisco.com/techsupport

17 Secure Shell Fundamentals and Configuration

Topics:

- Secure Shell fundamentals
- CiscoSSH
- Requirements for Secure Shell
- Secure Shell restrictions
- Configure an SSH server
- NETCONF access controls
- Automatic generation of SSH host keys
- Configure an SSH client
- SSH client strict host key checks
- SSH client authentication order
- SSH multichannel connections
- Configure an OpenSSH client for multiplexing

Explains core SSH functions and CiscoSSH behavior, and provides server, client, NETCONF, host-key, multiplexing, cipher, and HMAC configuration guidance.

Use this chapter to understand SSH behavior and configure secure remote-management access on Cisco IOS XR routers.

The chapter provides information in these functional areas:

- SSH clients, servers, secure file transfer, authentication methods, and software packaging
- CiscoSSH behavior, implementation differences, requirements, recommendations, and session messages
- SSH server and client configuration, NETCONF access controls, and host-key management
- SSH authentication order, multichannel connections, cipher selection, and HMAC controls

Secure Shell fundamentals

Explains how SSH secures remote sessions and provides client, server, file-transfer, authentication, and software-package functions on Cisco IOS XR routers.

Secure Shell (SSH) is an application and protocol that

- replaces unencrypted Berkeley remote-access tools
- uses cryptographic mechanisms to authenticate and protect sessions, and
- supports remote command execution and secure file transfer.

Cisco IOS XR supports SSHv1 and SSHv2. SSHv1 uses RSA keys. SSHv2 supports DSA, RSA, and ECDSA keys. CiscoSSH is the OpenSSH-based implementation available from Cisco IOS XR Software Release 7.3.2.

Table 56: Feature History Table

Feature Name	Release Information	Feature Description
Implementing Secure Shell	Release 25.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8711-48Z-M • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Implementing Secure Shell	Release 24.4.1	Introduced in this release on: Fixed Systems (8700 [ASIC:K100])(select variants only*). Secure Shell (SSH) provides encrypted communication for secure remote management of network devices. This feature enhances security by protecting against eavesdropping and unauthorized access during remote sessions. By implementing SSH, administrators can ensure secure command-line interface access, safeguarding sensitive data and maintaining the integrity of network operations. *Previously this feature was supported on Q200 and Q100. It is now extended to Cisco 8712-MOD-M routers.

SSH limitations

- Cisco IOS XR does not support X11 forwarding through an SSH connection.
- References to CiscoSSH apply to the OpenSSH-based implementation unless a topic explicitly identifies the older Cisco IOS XR SSH implementation.

SSH clients and servers

Explains how SSH clients and servers establish encrypted inbound and outbound management connections and identifies supported client authentication and packet-marking behavior.

SSH clients and servers are complementary applications that

- establish encrypted connections across an insecure network
- authenticate the router, remote device, and user, and
- provide functions comparable to inbound and outbound Telnet without sending the session in clear text.

The SSH server accepts connections from publicly and commercially available clients. The SSH client connects the router to Cisco routers and other SSH servers.

SSH client support

- Authentication sources: RADIUS, TACACS+, and locally stored usernames and passwords
- Legacy client algorithms stated by the source: AES, 3DES, and SHA-1
- DSCP range: 0 through 63
- Default client and server DSCP value when it is not configured: 16

Use the **ssh client** command in XR Config mode to configure SSH client options.

The router can execute a command on a remote device as part of an outbound SSH session.

How secure file transfers work

Describes how Secure File Transfer Protocol (SFTP) clients and the SSH server create authenticated file-transfer sessions, exchange protocol requests and responses, and apply transfer-rate controls.

SFTP is an SSHv2 subsystem that securely copies router configuration and image files. The client is always enabled and supports VRF-aware and interactive operation.

> **Note**
>
> The SSH server must be running to accept incoming SFTP connections.

Summary

The process uses these components:

- SFTP client: Sends the file-transfer request.
- SSH server: Authenticates the user and creates a child session.
- SFTP server: Processes file requests through a request-response protocol.
- LPTS policer: Controls the SSH-known packet rate.

Workflow

These stages describe a secure file transfer:

1. The user runs the **sftp** command. The SFTP API creates a child session that interacts with the SSH server.

2. For each incoming request, the SSH server creates an SSH server child process. The child process establishes a secure channel through key exchange and user authentication.
3. If the client requests the SFTP subsystem, the SSH server daemon creates an SFTP server child process and instance. The server authenticates the session, initiates the connection, and sets the client environment and default user directory.
4. The SFTP server waits for an SSH_FXP_INIT message. The client then sends requests, and the server returns status, handle, data, or name responses.
5. The client and server transfer data through the encrypted channel. The SSH-Known LPTS policer applies to the transfer.

Result

The client and server exchange files over an authenticated and encrypted SSH session.

SSH authentication methods

Describes server host, user, password, public-key, and keyboard-interactive authentication behavior for secure remote SSH connections on Cisco IOS XR routers.

Use this information to distinguish server verification from user authentication.

SSH supports these authentication methods:

- Host authentication: The client verifies the public key supplied by the server during key exchange. A changed known-host key causes the client to close the connection.
- RSA user authentication: The user proves possession of a private RSA key by sending a signature. The corresponding public key is stored on the server.
- Password authentication: This method is the default when the server supports it.
- Keyboard-interactive authentication: The client accepts interactive prompts without knowing the underlying authentication mechanism. This method supports interactive applications only.

RSA user keys have these source limits:

- Minimum length: 512 bits
- Maximum length: 4096 bits
- Import format: Base64-encoded binary format

SSH and SFTP software packages

Describes which SSH, SFTP, management-plane, control-plane, and data-plane security components are present in the base and optional security software packages.

Use this information to determine which software package supplies each security component.

The software packages contain these components:

- Base package: SSH, SFTP, management-plane components, and control-plane components such as the IPsec control plane
- Security package: Data-plane components such as MACsec and the IPsec data plane

Both packages allow FIPS operation so that the control plane can negotiate FIPS-approved algorithms.

CiscoSSH

Explains the OpenSSH-based Cisco implementation, including post-quantum key exchange, implementation differences, restrictions, operating recommendations, and session-event messages for secure router management.

CiscoSSH is an OpenSSH-based secure-shell implementation that

- uses the Linux TCP/IP stack on management Ethernet and line-card interfaces
- supports FIPS operation, X.509 certificates, MPP, ACLs, and VRFs, and
- preserves most Cisco IOS XR SSH commands while adding OpenSSH security and interoperability.

CiscoSSH replaces the older Cisco IOS XR SSH implementation beginning with Cisco IOS XR Software Release 7.3.2.

Table 57: Feature History Table

Feature Name	Release Information	Feature Description
CiscoSSH	Release 7.3.2	This release introduces CiscoSSH, a newer implementation of SSH on this platform. CiscoSSH leverages OpenSSH implementation, by using the Linux TCP/IP stack to transmit and receive SSH packets over the management Ethernet interface and line card interfaces on the router. CiscoSSH provides additional security features like FIPS compliance and X.509 digital certification. It supports packet path features like MPP, ACL and VRF support, and ensures interoperability with various existing SSH implementations. **Note** Cisco IOS XR SSH, the SSH implementation that existed prior to this release, is now deprecated.

Post-quantum cryptography key exchange for CiscoSSH

Explains the hybrid post-quantum key exchange algorithms supported by CiscoSSH and identifies their post-quantum and classical cryptographic components for secure sessions.

Post-Quantum Cryptography (PQC) key exchange for CiscoSSH is a hybrid security capability that

- combines a post-quantum algorithm with the classical X25519 key exchange
- mitigates long-term quantum risks to remote-access and data-transfer confidentiality and integrity, and

- is enabled by default beginning with Cisco IOS XR Software Release 26.2.1.

CiscoSSH supports hybrid algorithms based on ML-KEM and Streamlined NTRU Prime.

Table 58: Feature History Table

Feature Name	Release Information	Feature Description
Post-Quantum Cryptography key exchange for CiscoSSH	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) CiscoSSH mitigates quantum threats to ensure long term confidentiality and integrity of remote access and data transfer by introducing PQC (Post-Quantum Cryptography) hybrid key exchange algorithms. Supported algorithms include ML-KEM and NTRU Prime hybrids: • `mlkem768x25519-sha256` • `sntrup761x25519-sha512` • `sntrup761x25519-sha512@openssh.com`
Post-Quantum Cryptography key exchange for CiscoSSH	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) CiscoSSH mitigates quantum threats to ensure long term confidentiality and integrity of remote access and data transfer by introducing PQC (Post-Quantum Cryptography) hybrid key exchange algorithms. Supported algorithms include ML-KEM and NTRU Prime hybrids: • `mlkem768x25519-sha256` • `sntrup761x25519-sha512` • `sntrup761x25519-sha512@openssh.com`

Supported PQC algorithms

Use this table to select a supported PQC hybrid key exchange algorithm.

Table 59: Supported PQC algorithms

Algorithm	Description
`mlkem768x25519-sha256`	Combines NIST-standardized ML-KEM with the classical X25519 elliptic-curve Diffie-Hellman exchange.
`sntrup761x25519-sha512`	Combines Streamlined NTRU Prime with the classical X25519 elliptic-curve Diffie-Hellman exchange.
`sntrup761x25519-sha512@openssh.com`	Provides the older OpenSSH-style name for compatibility with peers that expect that identifier.

PQC algorithm configuration

Configure server and client key exchange lists with the **ssh server algorithms key-exchange** and **ssh client algorithms key-exchange** commands.

CiscoSSH and Cisco IOS XR SSH differences

Compares server ports, username syntax, algorithm handling, timeouts, rekey behavior, port forwarding, and file-transfer behavior between the current and legacy SSH implementations.

Use this comparison when migrating from Cisco IOS XR SSH to CiscoSSH.

Table 60: SSH implementation differences

Function	CiscoSSH	Cisco IOS XR SSH
NETCONF ports	Uses port 830 or SSH port 22; an explicit port can also be configured.	Supports an explicitly configured NETCONF port.
Username syntax	Does not authenticate usernames that contain a colon because the colon is a delimiter.	Does not impose this colon restriction.
Unsupported algorithms	Cannot enable unsupported algorithms.	Can explicitly enable supported legacy options with the **ssh server enable cipher** command.
Unreachable-host timeout	120 seconds	60 seconds
Authentication timeout	Applies the configured value to the total allowed login attempts.	Applies the value to each login attempt.
Time-based rekey	Triggers when a packet arrives after timer expiration.	Triggers immediately after timer expiration.
Port-forwarded channels	Does not state a channel limit, but **show ssh** displays no more than 16 entries.	Supports a maximum of 16 channels.
LPTS policer rate for port-forwarded SSH sessions	When using SSH port forwarding feature, the router considers the traffic flows corresponding to port-forwarded SSH sessions as third party applications. Hence, the LPTS polices those traffic flows at a medium rate.	The LPTS polices the traffic flows corresponding to port-forwarded SSH sessions at a high rate.

Function	CiscoSSH	Cisco IOS XR SSH
File transfer through SCP	The router checks for the presence of system files after authentication.	The router checks for the presence of system files before authentication.
Noninteractive SFTP initiated by the router	Transfers from an external device to the router only.	Transfers in both directions.

Restrictions for CiscoSSH

Provides mandatory operating restrictions for CiscoSSH versions, interfaces, addresses, algorithms, commands, backup servers, and management-interface ACLs on supported routers.

Use only supported CiscoSSH versions, interfaces, algorithms, and commands.

- Do not use SSHv1 or configure a backup SSH server.
- Do not use the standby management Ethernet interface before Release 24.2.1.
- Do not use secondary IPv4 addresses or BVI interfaces for SSH before Release 7.7.1.
- Do not configure AES-CBC, 3DES-CBC, or **diffie-hellman-group1-sha1** for CiscoSSH.
- Do not use these unsupported commands:
  - **show ssh history**
  - **show ssh history details**
  - **clear ssh stale sessions**
- For Cisco IOS XR Software releases earlier than 25.1.1, configure ingress ACLs in SSH server configuration mode. Ingress ACLs on the management interface do not filter SSH or NETCONF traffic in these releases.

Recommendations for using CiscoSSH

Provides operational recommendations for CiscoSSH connections, authentication, algorithms, upgrades, VRFs, ACLs, sessions, processes, routing, verification, and debugging on managed network routers.

Connection and authentication behavior

Account for CiscoSSH connection syntax and authentication behavior when you configure clients and access controls.

- Place the NETCONF or SFTP subsystem name last when you send a subsystem request from an SSH client.

```
ssh username@ipaddress -p 830 -s netconf    Correct usage
ssh username@ipaddress netconf -p 830 -s    Incorrect usage
```

- Expect a disallowed SSH client connection to time out when an ACL or MPP policy blocks it. The router does not send a TCP reset for the blocked connection.
- If the router has no imported public key, CiscoSSH still negotiates public-key authentication. The public-key attempt fails and generates a console syslog, after which the client and server continue with keyboard-interactive or password authentication. Disable public-key authentication on the client when it is not required.
- For Go SSH-based clients, use the option that ignores the window size when you initiate a connection. Adding a line-feed option can prevent session establishment because the client expects a nonzero window size while CiscoSSH sends a window size of 0.

Algorithm and upgrade preparation

Remove unsupported algorithm configurations and recalculate session limits when you migrate from Cisco IOS XR SSH to CiscoSSH. CiscoSSH rejects unsupported algorithm-only configurations and applies session and rate limits differently from Cisco IOS XR SSH.

- Remove configurations that contain only unsupported algorithms, such as `3des-cbc` or `diffie-hellman-group1-sha1`, before an upgrade. A configuration that contains only unsupported algorithms fails to commit.

```
Router(config)# ssh server algorithms cipher 3des-cbc

!!% Operation not permitted: 3des-cbc is not supported in CiscoSSH; SSH cannot
 operate with only this option.
```

- If a cipher list contains supported and unsupported algorithms, the router issues a warning and removes the unsupported algorithms.

```
Router(config)# ssh server algorithms cipher aes128-ctr aes192-ctr 3des-cbc

ssh_conf_proxy[1193]: %SECURITY-SSHD_CONF_PRX-3-ERR_GENERAL: 3des-cbc is not
 supported and will be removed
```

- Reconfigure the SSH session and rate limits after an upgrade from Cisco IOS XR SSH. The configuration applies to every enabled VRF, and CiscoSSH enforces each limit per VRF. The maximum number of VTY sessions remains 200 across all VRFs.

VRF and ingress filtering

Configure VRFs and ingress filtering according to the Cisco IOS XR Software release and the required SSH or NETCONF traffic path.

- Configure ingress ACLs in SSH server configuration mode to filter SSH and NETCONF traffic.

  Use this syntax for SSH:

```
ssh server vrf vrf-name ipv4 access-list ipv4-access-list-name ipv6
access-list ipv6-access-list-name
```

  Use this syntax for NETCONF:

```
ssh server netconf vrf vrf-name ipv4 access-list ipv4-access-list-name ipv6
 access-list ipv6-access-list-name
```

- Beginning with Cisco IOS XR Software Release 25.1.1, use **ssh server packet-flow-netio ingress** in XR Config mode when ingress ACLs remain on the management interface. This configuration can reduce SCP and SFTP performance.
- Use **ssh vrf default** only for SSHv2. CiscoSSH does not support SSHv1 on the default VRF.
- After restarting `ssh_conf_proxy`, expect a delay before a nondefault VRF accepts incoming SSH sessions. Session establishment can time out while the router programs the required LPTS entries.

Session and process behavior

Plan operational procedures around CiscoSSH session state and Linux process behavior.

- Account for the default keepalive interval of 60 seconds and the maximum of three attempts. CiscoSSH detects an unresponsive session after 180 seconds. These parameters are not configurable.
- Do not rely on the configured TCP window-scale value for CiscoSSH. The router accepts the configuration, but Linux TCP uses dynamic window scaling and ignores the application setting.

- After an SSH server process restart, session-limit enforcement does not count sessions that existed before the restart because the active-session count is not persistent.
- Expect any SSH configuration change to restart the SSH server process. Existing SSH, SCP, SFTP, and NETCONF sessions remain active.
- Use **clear ssh all** to clear all incoming SSH sessions.
- After restarting `xlncd` or `ip_smiap`, allow time for the router to restore virtual IP addresses.
- Restart the Linux CiscoSSH server process with the **kill** command in the Linux shell. The **process restart ssh_server** command cannot restart this process.
- Restart `ssh_conf_proxy` and `ssh_syslog_proxy` with the **process restart** command. These processes handle SSH configuration and syslog messages.
- Do not expect an XR-TCP process restart to affect CiscoSSH because CiscoSSH uses Linux TCP.

Routing, verification, and debugging

Account for Linux route selection, command timeouts under stress, and SSH server restarts caused by debug-level changes.

- Do not expect Linux to use a route that is more specific than a connected route in the XR routing table.

```
XR routing table:

10.0.0.0/24   via 10.0.0.2 (connected route)
10.0.0.192/28 via 198.51.100.1 (static route)
```

This table shows the expected next-hop behavior:

Table 61: Expected behavior of more-specific routes with CiscoSSH

Destination IP range	Cisco IOS XR OS next hop	Linux next hop	Match
10.0.0.1–10.0.0.191	10.0.0.2	10.0.0.2	Yes
10.0.0.193–10.0.0.206	198.51.100.1	10.0.0.2	No
10.0.0.207–10.0.0.255	10.0.0.2	10.0.0.2	Yes

- During a router stress test, retry **show ssh**, **show ssh session details**, or **show ssh rekey** if the command times out.

```
Error: Timed out to obtain information about one or more incoming/outgoing
sessions. Please retry.
```

- Use **debug ssh server l1/l2/l3** for three server debug levels and **debug ssh client l1/l2/l3** for client debugging.
- Expect the SSH server process to restart whenever you enable or disable debugging. A debug-level change updates LOGLEVEL in the internal `sshd_config` file.

CiscoSSH session event messages

Describes the CiscoSSH and legacy Cisco IOS XR SSH message patterns for session login, logout, authentication failure, and rekey events.

Use these message identifiers to correlate SSH session events with the active implementation.

Table 62: SSH session event identifiers

Event	CiscoSSH indication	Cisco IOS XR SSH indication
Login	SSHD_SYSLOG_PRX reports accepted authentication and session start.	SSHD reports successful authentication.
Logout	SSHD_SYSLOG_PRX reports the disconnect and user.	SSHD reports that the user logged out from the VTY.
Login failure	SSHD_SYSLOG_PRX reports failed authentication.	SSHD reports the failed user authentication attempt.
Rekey	SSHD_SYSLOG_PRX reports a server-initiated time rekey.	SSHD emits an INFO_REKEY message.

Requirements for Secure Shell

Provides user access-control, VRF, AAA, host-key, and SFTP requirements that must be satisfied before configuring or using secure SSH services.

Satisfy the access, routing, authentication, and key requirements before you enable SSH services.

- Use a user group whose task group includes the task IDs required by each command.
- Provide the default VRF or a specific VRF for an SSHv2 server.
- Configure local or remote user authentication through AAA.
- Configure AAA authentication and authorization for SFTP.
- Keep at least one default host key on a router that accepts incoming SSH sessions. FIPS mode requires a default RSA or ECDSA key.

Secure Shell restrictions

Describes host-key, FIPS, SFTP, authentication, application, cipher, terminal, password, and file-system restrictions that apply to SSH services on Cisco IOS XR routers.

Review these restrictions before configuring SSH, SCP, or SFTP.

SSH host-key requirements

- An external client requires the router to have an RSA, DSA, or ECDSA host-key pair for an incoming SSHv2 connection.
- A local RSA, DSA, or ECDSA host-key pair is not required when the router initiates an SSH client connection to an external device. A local host key is also not required when the router operates as an SFTP client.
- During boot, the router automatically generates missing RSA, DSA, and ECDSA host-key pairs.
- Do not delete every default crypto key identified by the `the_default` label. SSH clients cannot establish sessions with the router when no default key remains.
- Keep at least one default crypto key on the router. In FIPS mode, at least one default RSA or ECDSA key is mandatory.

FIPS algorithm restrictions

These algorithms are supported only when FIPS mode is disabled:

- Key exchange: `diffie-hellman-group1-sha1` and `curve25519`
- Cipher: `3des-cbc` and `chacha20-poly1305@openssh.com`

An SSH session fails if one of these non-FIPS algorithms is configured before FIPS mode is enabled.

SFTP requirements and restrictions

- The remote SSH server must enable the SFTP subsystem.
- For example, an SSHv2 server can enable the subsystem in `/etc/ssh2/sshd2_config` with this entry:

```
subsystem-sftp /usr/local/sbin/sftp-server
```

- Public-domain SSH packages usually include the SFTP server and enable it by default.
- SFTP requires OpenSSH_2.9.9p2 or later on the remote server.

Authentication and application restrictions

- SSH, SFTP, and SCP servers support RSA-based user authentication. The SSH client does not support RSA-based user authentication in this source context.
- Execution shell, SFTP, SCP, and NETCONF are the supported SSH applications.

SSH server cipher preference

The SSH server uses this cipher preference order:

1. AES128
2. AES192
3. AES256
4. aes128-gcm
5. aes256-gcm
6. chacha20-poly1305

The server rejects a client request for an unsupported cipher, and the SSH session does not proceed.

Terminal and password restrictions

- Use terminal type VT100. The router generates a warning when the client uses another terminal type.
- The SSH client does not support a password message of `none`.

File permission and timestamp behavior

- A file created on the router does not retain its original UNIX-like permission information because the router infrastructure does not provide UNIX-like file permissions.
- A file created on the remote file system uses the destination host's umask.
- The modification time and last-access time of a file created on the remote file system are set to the time of the copy.

Configure an SSH server

Configures a router to accept secure SSH connections and optionally sets transfer scaling, authentication timeout, access controls, packet marking, and session verification.

Enable secure inbound management access to the router.

Before you begin

- Meet the access, VRF, AAA, and host-key requirements.
- Identify any IPv4 or IPv6 ACLs that must restrict incoming connections.

Procedure

1. Configure the router identity and enable the SSH server in global configuration mode.

   Example

   ```
   Router# configure
   Router(config)# hostname router1
   Router(config)# domain name example.com
   Router(config)# ssh server v2
   ```

   Use **ssh server vrf** with optional IPv4 and IPv6 ACLs when the server must operate in a specific VRF or restrict clients. The server supports multiple VRFs.

2. Optionally configure transfer scaling, AAA authentication timeout, and packet marking.

   Example

   ```
   Router(config)# ssh server tcp-window-scale 10
   Router(config)# ssh timeout 60
   Router(config)# ssh server dscp 63
   Router(config)# commit
   ```

   - The TCP window scale can improve in-band SCP and SFTP throughput.
   - The authentication timeout range is 5 through 120 seconds; the default is 30 seconds.
   - The DSCP range is 0 through 63; the default is 16.

3. Verify active and historical SSH session information.

   Example

   ```
   Router# show ssh
   Router# show ssh session details
   Router# show tech-support ssh
   ```

   - **show ssh** displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.
   - **show ssh session details** displays a detailed report of the SSHv2 connections to and from the router.

- **show tech-support ssh** automatically runs the show commands that display system information.

The router accepts supported SSH client connections on the configured VRFs and applies the optional controls.

NETCONF access controls

Explains how NETCONF access controls separate management requests from other SSH services by port and ACL while preserving SCP and SFTP access.

A NETCONF access control is an SSH security mechanism that

- blocks NETCONF requests on the SSH port while allowing them on a designated NETCONF port
- restricts NETCONF access with IPv4 or IPv6 ACLs, and
- allows SSH services such as SCP and SFTP to continue on the SSH port.

SSH uses port 22 by default, and NETCONF uses port 830 by default. Without this control, a NETCONF session can be established on the SSH port.

Table 63: Feature History Table

Feature Name	Release Information	Description
NETCONF access controls	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) When this feature is enabled, NETCONF sessions will be blocked on the SSH port. However, SCP and SFTP will continue to function on the SSH port. The feature introduces these changes: CLI: • **ssh server netconf disable ssh-port** YANG Data Models: • New XPaths for `Cisco-IOS-XR-um-ssh-cfg.yang` (see GitHub, YANG Data Models Navigator)

Benefits of NETCONF access control

- Prevents unauthorized NETCONF requests on the SSH port.
- Blocks NETCONF for selected addresses without blocking their SSH access.

- Preserves SCP and SFTP access.

How NETCONF access controls work

Describes how the SSH server identifies a NETCONF channel request after session establishment and permits or rejects the requested service.

The SSH server cannot identify the requested subsystem during the TCP handshake or user authentication. It identifies NETCONF when the client requests a channel.

Summary

The process uses these components:

- SSH server: Establishes the connection and enforces subsystem access.
- SSH and NETCONF ports: Separate general SSH and NETCONF services.
- ACLs: Restrict addresses on the configured service port.
- Client: Requests NETCONF, SCP, SFTP, or another permitted SSH service.

Workflow

These stages describe NETCONF access enforcement:

1. An administrator disables NETCONF on the SSH port and configures any port ACLs.
2. The server completes the TCP handshake and authenticates the SSH connection.
3. At channel-request time, the server identifies the requested subsystem.
4. The server rejects NETCONF on a disabled SSH port and continues to allow permitted services such as SCP and SFTP.

Result

NETCONF is available only through permitted ports and addresses while other allowed SSH services remain available.

Guidelines for NETCONF access control

Provides guidance for separating NETCONF from the SSH port and limiting management access to approved client addresses with ACLs on managed routers.

Restrict NETCONF to approved addresses and ports

Disable NETCONF on the SSH port and apply ACLs to the designated NETCONF service.

The server cannot distinguish NETCONF from another SSH service until the client requests a channel. Port ACLs can reject disallowed addresses during connection establishment.

Apply this practice when NETCONF management must meet address-based or port-separation security requirements.

Approved clients use the NETCONF port while SCP, SFTP, and other permitted SSH services remain available on the SSH port.

NETCONF access control restrictions

Describes when the SSH server can identify NETCONF requests and how port and address restrictions affect NETCONF, SCP, and SFTP traffic.

Account for these restrictions when designing NETCONF access policy.

NETCONF access control has these restrictions:

- The SSH protocol does not identify the requested service during the TCP handshake or authentication.

- The server rejects NETCONF on the SSH port only after it receives the channel request.
- An ACL on the NETCONF port rejects a blocked address during the TCP handshake.
- Disabling NETCONF on the SSH port does not block SCP or SFTP for the same address.

Configure NETCONF access control

Configures the SSH server to reject NETCONF channel requests on the SSH port while preserving other permitted SSH services for authorized clients.

Prevent NETCONF management sessions from using the SSH service port.

Before you begin

- Ensure that you have administrative access.
- Confirm that NETCONF clients can use the designated NETCONF port before blocking NETCONF on the SSH port.

Procedure

1. Disable NETCONF on every configured SSH port and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server netconf disable ssh-port
   Router(config)# commit
   ```

2. Attempt a NETCONF connection on the SSH port and inspect the router log.

   Example

   ```
   Router# show logging
   %SECURITY-SSHD_SYSLOG_PRX-3-ERR_GENERAL: On ssh port invalid netconf channel
    request received
   ```

The router rejects NETCONF channel requests on the SSH port and continues to accept permitted SSH, SCP, and SFTP requests.

Automatic generation of SSH host keys

Explains automatic SSH host-key generation, allowed host-key selection, Ed25519 support, FIPS restrictions, verification, and manual key management on the router.

Automatic generation of SSH host keys is a security feature that

- creates SSH host key for supported algorithms (DSA, ECDSA, and RSA) automatically when the router boots
- eliminates the need for explicit manual key generation after initial setup, and
- ensures SSH clients can connect to the SSH server immediately after bootup with a basic configuration.

An SSH host key is a public-private key pair that

- identifies the SSH server to connecting clients
- participates in secure session establishment, and
- must match an algorithm accepted by both the client and server.

The router automatically creates default DSA, ECDSA, and RSA host-key pairs at boot when they are missing. Release 26.1.1 changes the automatically generated RSA key from 2048 to 3072 bits.

Table 64: Feature History Table

Feature Name	Release info	Description
SSH key strength: 3072-bit by default	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This update enhances device security by automatically generating RSA 3072-bit SSH host keys during system boot, instead of RSA 2048-bit keys. 3072-bit aligns with industry best practices and provides improved cryptographic protection, ensuring secure SSH access and compliance with the latest security requirements.

SSH host-key algorithm priority order

During SSH connection negotiation, the router evaluates host-key algorithms in this order, from highest to lowest priority:

1. `ecdsa-nistp-521`
2. `ecdsa-nistp-384`
3. `ecdsa-nistp-256`
4. RSA
5. DSA

SSH host-key management

- If a host-key pair is not required, use the **crypto key zeroize** command in EXEC mode to remove it.
- If an SSH host-key pair is not present after the router boots, use the **crypto key generate** command in EXEC mode to generate it manually.

How SSH host-key pairs are generated

Describes how the router creates missing host-key pairs during boot, preserves existing keys during upgrades, and makes SSH available for initial provisioning.

Automatic key generation supports initial boot, zero-touch provisioning (ZTP), Golden ISO boot, and software upgrades.

During ZTP, no additional steps are required to configure SSH host-key pairs.

Summary

The process evaluates supported algorithms and the keys already present on the router.

Workflow

These stages describe automatic host-key generation:

1. During initial boot, the router checks for host-key pairs for all supported algorithms: DSA, ECDSA, and RSA.
2. The router creates each missing pair, which enables clients to establish SSH connections after the basic SSH configuration is active. From Release 26.1.1, the default RSA key is 3072 bits.
3. During an upgrade from an earlier software version, the router preserves existing host-key pairs and generates only missing pairs. For example, during an upgrade from version 1 to version 2, the router does not regenerate pairs that were created in version 1. This behavior prevents duplicate key generation.
4. If an SSH host-key pair is not present after boot, use the **crypto key generate** command in EXEC mode to generate it manually.
5. The SSH server can use the generated keys after its basic configuration is active.

Result

SSH clients can authenticate the server without a separate manual key-generation step.

Configure allowed SSH host-key algorithms

Configures the SSH server to negotiate only selected host-key algorithms and verifies that the corresponding generated key pair exists on the router.

Limit server host-key negotiation to algorithms approved for the deployment.

Without an explicit allowed list, the server permits every generated host-key pair for backward compatibility.

Before you begin

Generate or confirm the host-key pair for every algorithm that you plan to allow.

Procedure

1. Allow the selected host-key algorithm and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server algorithms host-key ecdsa-nistp521
   Router(config)# commit
   ```

2. Verify the generated key for the allowed algorithm.

   Example

   ```
   Router# show crypto key mypubkey ecdsa
   Key label: the_default
   Type     : ECDSA General Curve Nistp256
   Degree   : 256
   Created  : 10:59:08 UTC Mon Nov 19 2018
   Data     :
   ```

```
04AC7533 3ABE7874 43F024C1 9C24CC66 490E83BE 76CEF4E2 51BBEF11 170CDB26
14289D03 6625FC4F 3E7F8F45 0DA730C3 31E960FE CF511A05 2B0AA63E 9C022482
6E

Key label: the_default
Type     : ECDSA General Curve Nistp384
Degree   : 384
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04B70BAF C096E2CA D848EE72 6562F3CC 9F12FA40 BE09BFE6 AF0CA179 F29F6407
FEE24A43 84C5A5DE D7912208 CB67EE41 58CB9640 05E9421F 2DCDC41C EED31288
6CACC8DD 861DC887 98E535C4 893CB19F 5ED3F6BC 2C90C39B 10EAED57 87E96F78
B6

Key label: the_default
Type     : ECDSA General Curve Nistp521
Degree   : 521
Created  : 10:59:09 UTC Mon Nov 19 2018
Data     :
0400BA39 E3B35E13 810D8AE5 260B8047 84E8087B 5137319A C2865629 8455928F
D3D9CE39 00E097FF 6CA369C3 EE63BA57 A4C49C02 B408F682 C2153B7F AAE53EF8
A2926001 EF113896 5F1DA056 2D62F292 B860FDFB 0314CE72 F87AA2C9 D5DD29F4
DA85AE4D 1CA453AC 412E911A 419E9B43 0A13DAD3 7B7E88E4 7D96794B 369D6247
E3DA7B8A 5E
```

The output identifies the key label, curve, creation time, and public-key data.

The server selects a host key only from the configured allowed list when a matching generated pair exists.

Ed25519 public-key signature algorithm support for SSH

Explains Ed25519 public-key signature support and its position in the algorithm priority used to negotiate SSH connections on Cisco IOS XR 64-bit platforms.

Ed25519 is a public-key signature algorithm supported when SSH clients and servers establish sessions on Cisco IOS XR 64-bit platforms.

Ed25519 provides better security and faster performance than DSA or ECDSA signature algorithms.

Table 65: Feature History Table

Feature Name	Release Information	Feature Description
Ed25519 public-key signature algorithm support for SSH	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Ed25519 public-key signature algorithm support for SSH	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Ed25519 public-key signature algorithm support for SSH	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Ed25519 public-key signature algorithm support for SSH	Release 7.3.1	This algorithm is now supported on Cisco IOS XR 64-bit platforms when establishing SSH sessions. It is a modern and secure public-key signature algorithm that provides several benefits, particularly resistance against several side-channel attacks. Prior to this release, DSA, ECDSA, and RSA public-key algorithms were supported. This command is modified for this feature: ssh server algorithms host-key

Public-key algorithm priority

During SSH negotiation, the client and server evaluate public-key algorithms in this order, from highest to lowest priority:

1. `ecdsa-sha2-nistp256`
2. `ecdsa-sha2-nistp384`
3. `ecdsa-sha2-nistp521`
4. `ssh-ed25519`
5. `ssh-rsa`
6. `ssh-dsa`

Guidelines for Ed25519 public-key signature algorithm usage

Explains why an SSH server operating in FIPS mode must have a permitted host-key algorithm other than Ed25519 for new connections.

Do not configure Ed25519 as the only allowed SSH host-key algorithm when FIPS mode is enabled.

Ed25519 is not FIPS-certified. The router therefore omits Ed25519 from the public-key algorithm list that it sends during SSH key negotiation for new connections when FIPS mode is enabled.

A new SSH connection fails when the **ssh server algorithms host-key** configuration permits only Ed25519 and FIPS mode is enabled.

This restriction applies only to new connections. An existing SSH session that negotiated Ed25519 remains active until the session disconnects.

Generate an Ed25519 public key

Generates an Ed25519 host key, permits the algorithm for SSH server negotiation, and verifies the resulting public-key information on the router.

Make an Ed25519 host key available to the SSH server.

Before you begin

Ensure that FIPS mode is disabled.

Procedure

1. Generate the Ed25519 key with the platform crypto key-generation procedure.

   The source delegates the exact key-generation syntax to the crypto key procedure.

2. Permit Ed25519 in the SSH server host-key list and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server algorithms host-key ssh-ed25519
   Router(config)# commit
   ```

3. Verify the Ed25519 public key.

   Example

   ```
   Router# show crypto key mypubkey ed25519
   ```

The router has an Ed25519 key that the SSH server can negotiate while FIPS mode is disabled.

What to do next

Use **crypto key zeroize ed25519** in EXEC mode to remove the key when it is no longer required.

Configure an SSH client

Configures server host-key validation and starts an authenticated outbound SSH connection from a Cisco IOS XR router to a remote server.

Connect securely from the router to a remote SSH server.

Before you begin

- Configure local or remote AAA authentication.
- Configure enough VTY lines for the required concurrent connections. The default VTY count is 5, and the source states a default maximum of 64 SSH sessions.
- Use AES-CTR in FIPS mode; 3DES and AES-CBC are not supported in that mode.

Procedure

1. Optionally configure the complete path to a known-host public-key file and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh client knownhost harddisk:/server_pubkey
   Router(config)# commit
   ```

   Include both the colon and slash in the device path.

2. Optionally set the DSCP value for packets sent by the SSH client.

   Example

   ```
   Router# configure
   Router(config)# ssh client dscp 63
   Router(config)# commit
   ```

   Use a value from 0 through 63. If you do not configure a value, the client and server use 16.

3. Start an outbound SSH connection.

   Example

   ```
   Router# ssh 192.0.2.1 username sample-user1
   ```

   - The client first attempts SSHv2. The remote peer can select SSHv1 only when it supports the legacy server.
   - The DES option applies only to an SSHv1 client.
   - When a hostname resolves to IPv4 and IPv6 addresses, the client uses IPv6.

4. Optionally execute a command on the remote router through the SSH session.

   Example

   ```
   Router# ssh 192.0.2.1 username admin command "show redundancy sum"
   Password:
   ```

```
    Active Node    Standby Node
    -----------    ------------
       0/4/CPU0        0/5/CPU0  (Node Ready, NSR: Not Configured)
Router#
```

The remote router returns the command output through the encrypted SSH session.

The router authenticates the remote server when a known-host file is configured, applies the optional packet marking, and opens the outbound SSH session.

Troubleshoot SSH client connections

Provides causes and corrective actions for rejected SSHv1 and SSHv2 connections, key-generation errors, session-capacity problems, FIPS cipher failures, AAA configuration, and PuTTY interoperability.

Use this information to identify and resolve common SSH connection problems.

Connection and key-generation problems

Table 66: SSH connection troubleshooting

Symptom or message	Possible cause	Resolution
An SSHv1 connection is rejected.	• The RSA key pair was zeroized. • The remote SSH server does not accept SSHv1 connections.	Confirm that the remote server accepts SSHv1. On the server, configure a hostname and domain, generate an RSA key pair with the **crypto key generate rsa** command in EXEC mode, and enable the SSH server.
An SSHv2 connection is rejected.	The DSA, RSA, or ECDSA key pair required for the connection was zeroized.	Configure a hostname and domain, generate the required key pair with the applicable **crypto key generate** command in EXEC mode, and enable the SSH server.
`No hostname specified`	The router does not have a configured hostname.	Configure the router hostname with the **hostname** command.
`No domain specified`	The router does not have a configured domain name.	Configure the router domain with the **domain name** command.

SSH connection capacity

Each SSH connection consumes one virtual terminal (VTY) resource. The number of concurrent SSH connections cannot exceed the number of VTY lines configured in the VTY pool.

- The default number of VTY lines is 5.
- The default maximum number of SSH sessions is 64.

Configure enough VTY lines in the VTY pool for the required number of concurrent SSH connections.

FIPS cipher requirements

In FIPS mode, use an AES-CTR cipher. FIPS mode does not support weaker ciphers such as 3DES or AES-CBC.

AAA authentication considerations

SSH uses local authentication or remote authentication configured through authentication, authorization, and accounting (AAA). When configuring AAA, apply the global configuration keyword that disables AAA on the console so that the console does not operate under AAA.

PuTTY interoperability

When using PuTTY 0.63 or later, go to `SSH > Bugs` and set `Chokes on PuTTY's SSH2 winadj request` to `On`. This setting helps prevent the session from breaking down when Cisco IOS XR sends lengthy output to the PuTTY client.

SSH client strict host key checks

Describes SSH client strict host key checking and explains how this feature enhances security and compliance on Cisco IOS XR routers by enforcing validation of server host keys before establishing SSH connections.

A SSH client strict host key checking is a feature that

- enforces validation of server host keys using a system-wide `known_hosts` file,
- restricts SSH client connections to hosts with trusted or previously accepted host keys, and
- supports configurable policies for new or changed keys, such as off, accept-new, ask, or yes.

The knownhosts file is located at `/mnt/rdsfs/ciscossh/known_hosts` and is shared across all users, providing persistent, centralized host key management for SSH client operations.

Table 67: Feature History Table

Feature Name	Release Information	Feature Description
SSH client strict host key checking	Release 26.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200], 8400 [ASIC: K100]); Modular Systems (8800 [LC ASIC: Q200, P100]). You enhance SSH security by enforcing strict host key checking, allowing you to control how the SSH client handles new or changed server keys - accept, reject, or prompt for approval. Trusted host keys are stored system-wide and persist across reloads, ensuring consistent validation for your outbound SSH connections.

> **Note**
>
> SSH client strict host key checking is incompatible with the legacy `ssh client knownhost` configuration. Remove the legacy configuration before configuring strict host key checking or using the `ssh-client knownhosts` exec-mode commands.

Restrictions for SSH client strict host key check

Lists the behaviors, requirements, and security implications of SSH client strict host key checking modes.

Provides reference information about the modes of SSH client strict host key checking, their behaviors, and their effect on host key storage, automation, and system security.

- All SSH client users share the default known hosts file at `/mnt/rdsfs/ciscossh/known_hosts`. Adding, deleting, or accepting a host key in this file affects all SSH client sessions.
- RDSFS must be available to retain host keys across device reloads and switchovers.
- When you use the `ssh-client knownhosts import <file-path>` command, the imported file replaces the contents of the default known-hosts file. The system does not append the imported entries.
- Host keys learned through SSH, SCP, SFTP, or other CLI operations that initiate SSH, SCP, or SFTP sessions are stored only in the default system-wide knownhosts file.
- The following strict host key checking modes are available:
  - `strict-check off`: Automatically accepts new or changed host keys. This mode weakens protection against man-in-the-middle attacks and should be used only for temporary, low-risk troubleshooting.
  - `strict-check accept-new`: Automatically accepts and stores new host keys, but rejects changed host keys.
  - `strict-check ask`: Prompts for approval only when an unknown host key is encountered. After the key is accepted and stored, any later change to that host key is rejected.
  - `strict-check yes`: Rejects both unknown and changed host keys. To avoid connection failures, pre-populate the trusted host list before enabling this policy.

> **Note**
>
> If the legacy `ssh client knownhost` command is present in the global configuration, you cannot configure `strict-hostkey-check`. The exec-mode `ssh-client knownhosts` commands are also unavailable until you remove the legacy configuration.

SSH client knownhosts commands

Use these exec-mode commands to manage the system-wide trusted-host database used by SSH, SCP, and SFTP client sessions.

The trusted-host database is stored in `/mnt/rdsfs/ciscossh/known_hosts` and is shared by all SSH client users.

Use these commands to manage entries in the trusted-host database.

Table 68: SSH client known hosts commands

Command	Description
ssh-client knownhosts add	Adds a trusted-host entry.
ssh-client knownhosts import	Imports a knownhosts file.
ssh-client knownhosts export	Exports the trusted-host database.
ssh-client knownhosts delete hostname	Deletes an entry by hostname.
ssh-client knownhosts delete line	Deletes a numbered entry.
ssh-client knownhosts clear	Removes all entries.
show ssh client knownhosts	Displays the trusted-host entries.

Run these commands in exec mode.

Host keys learned during SSH, SCP, SFTP, or another CLI session are stored only in the system-wide database.

Configure SSH client strict host key checking

Configure SSH client strict host key checking to enforce secure validation of SSH server host keys on routers.

Configure SSH client strict host key checking to control how routers verify SSH server host keys for outbound SSH connections and prevent unauthorized access.

- Enforce trust of only verified SSH servers to meet security policies.
- Minimize operational risk by ensuring host trust decisions are policy-driven.

> **Note**
>
> The global strict host key checking configuration can be overridden for an individual SSH, SCP, or SFTP session using the `strict-hostkey-check` option.

SSH client strict host key checking validates remote server keys before starting an SSH session. All SSH client operations reference a single, system-wide `knownhosts` file at `/mnt/rdsfs/ciscossh/known_hosts`, regardless of the user.

This mechanism ensures that clients only connect to trusted hosts and enables administrators to choose the level of enforcement (**off**, **accept-new**, **ask**, or **yes**) for their environment.

- No per-user host key management; changes are global for all SSH client users.
- The `strict-check` mode controls the degree of host verification and is set according to deployment security requirements.

Before you begin

- Administrative access to the Cisco router.
- Persistent storage is available and RDSFS is mounted at `/mnt/rdsfs`.

- Your organization's SSH security policies for trusted host validation are known.

Follow these steps to configure SSH client strict host key checking.

Procedure

1. Set the SSH client strict host key checking mode.

   Example

   ```
   Router# configure
   Router(config)# ssh client strict-hostkey-check {off | accept-new | ask |
   yes}
   Router(config)# commit
   ```

   - `off`: Automatically accepts new and changed host keys.
   - `accept-new`: Accepts and stores new host keys, but rejects changed host keys.
   - `ask`: Prompts for an unknown host key; after it is accepted and stored, rejects any later key change.
   - `yes`: Rejects both unknown and changed host keys.

2. Exit configuration mode.

   Example

   ```
   Router(config)# end
   ```

3. For an individual node, use exec-mode commands to manage trusted-host entries.

   The following commands operate on the current node. To align trusted-host entries across nodes, copy and import the same baseline known-hosts file on each node.

   a) Import a baseline knownhosts file.

      Example

      ```
      Router# ssh-client knownhosts import <file-path>
      ```

   b) Export the system-wide known-hosts database.

      Example

      ```
      Router# ssh-client knownhosts export <file-path>
      ```

   c) Add, delete, or clear trusted-host entries if required.

      Example

      ```
      Router# ssh-client knownhosts add
      Router# ssh-client knownhosts delete hostname
      Router# ssh-client knownhosts delete line
      Router# ssh-client knownhosts clear
      ```

      To manage the trusted host entries, see SSH client knownhosts commands on page 376.

4. Verify the trusted-host list.

Example

```
Router# show ssh client knownhosts
```

SSH client strict host key checking is configured. The router handles new and changed SSH server keys according to the selected enforcement mode.

What to do next

Review the router's outbound SSH connection logs to confirm that SSH client connections are handled according to the selected strict host-key checking mode and your organization's SSH security policies.

SSH client authentication order

Describes the default order in which Cisco IOS XR SSH and CiscoSSH clients negotiate public-key, password, and keyboard-interactive authentication.

Use this information to compare the default client authentication order before setting an explicit preference.

Table 69: Default authentication order

Implementation	Authentication order
Cisco IOS XR SSH	public-key, password, keyboard-interactive
CiscoSSH	public-key, keyboard-interactive, password

Set the SSH client authentication order

Configures the preferred order in which the SSH client negotiates public-key, keyboard-interactive, and password authentication methods with compatible remote servers.

Apply an explicit authentication preference to outbound SSH connections.

The configuration is available from Cisco IOS XR Software Release 7.9.2 / Release 7.10.1.

Procedure

1. Set the authentication order and commit the configuration.

Example

```
Router# configure
Router(config)# ssh client auth-method public-key keyboard-interactive password
Router(config-ssh)# commit
```

2. Verify the configured method order.

Example

```
Router# show running-config ssh client auth-methods
```

SSH multichannel connections

Explains how SSH multiplexing carries concurrent shell, remote-command, and secure file-transfer channels through one authenticated TCP connection between a client and server.

An SSH multichannel connection is a multiplexed session that

- carries multiple SSH channels through one TCP socket
- authenticates the user once for all channels in the session, and
- supports simultaneous terminal, command-execution, and file-transfer operations.

Server-side multiplexing is enabled by default. A compatible client must explicitly enable multiplexing.

Benefits of SSH multiplexing

- Reduces repeated authentication.
- Avoids establishing a separate TCP connection for each operation.

How SSH multichannel connections work

Describes how a client creates a primary SSH session, opens service channels, and multiplexes their traffic through a shared TCP connection.

The client and server use a single authenticated session for multiple concurrent services.

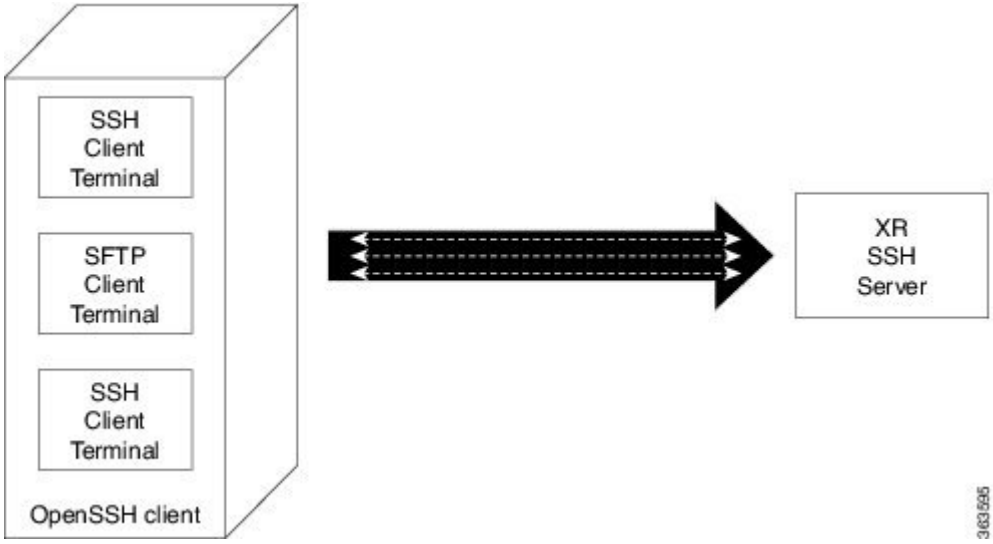
Summary

The process uses these components:

- Primary connection: Owns the TCP socket and authenticated session.
- Client channels: Carry independent shell, command, or file-transfer operations.
- SSH server: Processes each request to open a channel.

Workflow

Figure 8: Client-server interaction over a SSH multichannel connection



These stages describe SSH multiplexing:

1. The client creates a primary SSH connection and authenticates the user.
2. A later client operation locates the existing control socket instead of creating another TCP connection.
3. The client requests a new channel for the operation. The server processes that request as one service.
4. The client multiplexes traffic from all active channels through the primary connection.

Result

Multiple operations share one authenticated TCP connection while their SSH channels remain logically independent.

SSH multichannel connection restrictions

Describes the concurrent-channel, throughput, client-configuration, and server-side support limits that apply when multiple SSH services share one authenticated TCP connection.

Review these limits before enabling client multiplexing.

SSH multiplexing has these restrictions:

- Do not use one multiplexed connection for heavy data transfer because all channels share that connection's TCP speed limit.
- Do not use more than 15 concurrent channels in one session.
- Cisco IOS XR software provides server-side multiplexing; the client must provide compatible client-side support.

Configure an OpenSSH client for multiplexing

Configures an OpenSSH client to reuse a primary authenticated SSH connection for later service sessions to the same remote server.

Reduce connection setup and repeated authentication for concurrent SSH operations.

Before you begin

Use an OpenSSH client and ensure that the client configuration directory exists.

Procedure

1. Open the system or user SSH client configuration file.

   The system file is located under `/etc/ssh/ssh_config`. The user file is located under `~/.ssh/config` or `$HOME/.ssh/config`.

2. Add a host rule that creates or reuses a primary connection.

   Example

   ```
   Host *
       ControlMaster auto
       ControlPath ~/.ssh/tmp/%r@%h:%p
   ```

   ControlMaster auto creates a primary session when one does not exist. **ControlPath** identifies its control socket.

3. Create the directory specified by ControlPath and save the client configuration.

   Protect the directory so that only the intended user can access its control sockets.

Later SSH operations to a matching server reuse the primary connection when its control socket is available.

SSH cipher and HMAC controls

Explains how SSH client and server settings select encryption algorithms, disable HMAC algorithms, and control access to deprecated cipher families.

An SSH algorithm control is a security configuration that

- sets the client or server cipher preference list
- removes selected HMAC algorithms from negotiation, and
- permits deprecated CBC or 3DES ciphers only when explicitly enabled and supported.

Supported SSH encryption algorithms

Lists SSH encryption algorithms in source preference order and identifies which CTR, GCM, CBC, and 3DES algorithms are enabled by default.

Use this list when constructing an SSH client or server cipher preference.

The source lists these algorithms in preference order:

- aes128-ctr
- aes192-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com
- aes128-cbc
- aes192-cbc
- aes256-cbc

- 3des-cbc

CTR and GCM ciphers are enabled by default. CBC and 3DES ciphers are disabled by default and are not supported by CiscoSSH as documented in its implementation restrictions.

Caution: Avoid deprecated SSH algorithms

Warns that enabling CBC, 3DES, or other deprecated SSH algorithms can expose remote management sessions to avoidable cryptographic attacks and weaker protection.

Enable a deprecated SSH algorithm only when a required peer cannot negotiate a supported stronger algorithm.

This caution applies to client and server commands that expose CBC or 3DES cipher families.

Deprecated algorithms provide weaker protection and can increase exposure to cryptographic attacks. CiscoSSH also rejects unsupported algorithm-only configurations.

Prefer CTR or GCM ciphers and remove deprecated algorithms after the compatibility requirement ends.

Disable an SSH HMAC algorithm

Removes a selected HMAC algorithm from SSH server and client negotiation and verifies the saved cryptographic configuration on the router.

Prevent negotiation of an HMAC algorithm that does not meet the deployment security policy.

Procedure

1. Disable the HMAC algorithm on the SSH server and client, and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server disable hmac hmac-sha1
   Router(config)# commit
   Router(config)# ssh client disable hmac hmac-sha1
   Router(config)# commit
   ```

2. Verify the SSH configuration.

   Example

   ```
   Router# show running-config ssh
   ```

The SSH client and server no longer offer the disabled HMAC algorithm.

Configure SSH cipher algorithms

Configures SSH client and server cipher lists, enables supported deprecated cipher families when required, removes configured lists, and verifies the resulting configuration.

Control the ciphers that an SSH client and server can use during connection negotiation.

Before you begin

Confirm that each selected cipher is supported by the SSH implementation and operating mode. Use deprecated CBC and 3DES ciphers only when they are required and supported.

Procedure

1. Enter global configuration mode on Router 1 and Router 2.

   Example

   ```
   Router# configure
   ```

2. Select one of the following cipher configurations and apply the corresponding commands.

   - To enable all listed ciphers on the client and server, configure:

     ```
     Router 1:
     Router(config)# ssh client algorithms cipher aes256-cbc aes256-ctr
     aes192-ctr aes192-cbc aes128-ctr aes128-cbc aes128-gcm@openssh.com
     aes256-gcm@openssh.com 3des-cbc

     Router 2:
     Router(config)# ssh server algorithms cipher aes256-cbc aes256-ctr
     aes192-ctr aes192-cbc aes128-ctr aes128-cbc aes128-gcm@openssh.com
     aes256-gcm@openssh.com 3des-cbc
     ```

   - To enable CTR ciphers on the client and CBC and 3DES ciphers on the server, configure:

     ```
     Router 1:
     Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr
     aes256-ctr

     Router 2:
     Router(config)# ssh server algorithms cipher aes128-cbc aes256-cbc
     aes192-cbc 3des-cbc
     ```

   - To remove the explicitly configured cipher lists from the client and server, configure:

     ```
     Router 1:
     Router(config)# no ssh client algorithms cipher

     Router 2:
     Router(config)# no ssh server algorithms cipher
     ```

   - To configure only deprecated CBC and 3DES ciphers on the client and server, configure:

     ```
     Router 1:
     Router(config)# ssh client algorithms cipher aes128-cbc aes192-cbc
     aes256-cbc 3des-cbc

     Router 2:
     Router(config)# ssh server algorithms cipher aes128-cbc aes192-cbc
     aes256-cbc 3des-cbc
     ```

- To explicitly enable deprecated CBC and 3DES cipher families and configure CTR ciphers on the client and server, configure:

```
Router 1:
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr
aes256-ctr

Router 2:
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
Router(config)# ssh server algorithms cipher aes128-ctr aes192-ctr
aes256-ctr
```

3. Commit the configuration on each router.

Example

```
Router(config)# commit
```

4. Verify the configured cipher lists in the running configuration.

Example

```
Router# show running-config ssh
```

When all listed ciphers are enabled, the running configuration contains:

```
Router 1:
ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!

Router 2:
ssh server algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!
```

The SSH client and server use their configured cipher lists during algorithm negotiation.

18 Certificate and Public-Key Authentication for SSH

Topics:

- SSH certificate and public-key authentication
- X.509v3 certificate authentication
- OpenSSH certificate authentication
- TACACS+ authorization for OpenSSH certificate users
- Public-key authentication for SSH clients
- Public-key authentication to SSH servers

Explains certificate and public-key trust models and provides requirements, configuration, verification, and key-management guidance for SSH clients and servers on Cisco IOS XR routers.

Use this chapter to select and configure X.509v3, OpenSSH certificate, or public-key authentication for SSH connections.

The chapter provides information in these functional areas:

- X.509v3 server and user certificate authentication
- OpenSSH CA, host-certificate, user-certificate, and TACACS+ integration
- Public-key authentication from a router acting as an SSH client
- Public-key authentication to a router acting as an SSH server

SSH certificate and public-key authentication

Explains how certificates, certificate authorities, and public-private key pairs authenticate SSH servers and users while avoiding the transmission of reusable passwords over the network.

SSH certificate and public-key authentication is an identity-verification model that

- uses possession of a private key to prove an identity
- uses a stored public key or a trusted certificate authority to verify that proof, and
- can authenticate a server, a user, or both endpoints of an SSH connection.

X.509v3 authentication uses certificates issued through public-key infrastructure. OpenSSH certificate authentication uses certificates signed with an SSH CA key. Direct public-key authentication stores a user's public key on the SSH server.

Authentication choices

- Use X.509v3 certificates when the deployment uses X.509 trustpoints, identity certificates, and certificate revocation lists.
- Use OpenSSH certificates when an OpenSSH CA signs router and user SSH certificates.
- Use direct public-key authentication when a router acting as a client or server must authenticate locally configured users by key pair.

X.509v3 certificate authentication

Explains how X.509v3 identity certificates and trusted CA signatures support SSH server and user authentication on Cisco IOS XR routers.

X.509v3 certificate authentication is an SSH authentication method that

- binds a server or user public key to a digital identity
- verifies the binding through a chain of signatures from a trusted CA, and
- allows or denies SSH access after certificate validation.

The feature supports server authentication and user authentication on the SSH server.

User authentication supports these public-key algorithms:

- x509v3-ssh-dss
- x509v3-ssh-rsa
- x509v3-ecdsa-sha2-nistp256
- x509v3-ecdsa-sha2-nistp384
- x509v3-ecdsa-sha2-nistp521

Server authentication supports only x509v3-ssh-rsa algorithm.

Table 70: Feature History Table

Feature Name	Release Information	Feature Description
X.509v3 Certificate-based Authentication for SSH	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
X.509v3 Certificate-based Authentication for SSH	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
X.509v3 Certificate-based Authentication for SSH	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
X.509v3 Certificate-based Authentication for SSH	Release 7.3.1	This feature adds new public-key algorithms that use X.509v3 digital certificates for SSH authentication. These certificates use a chain of signatures by a trusted certification authority to bind a public key to the digital identity of the user who is authenticating with the SSH server. These certificates are tough to falsify and are therefore used for identity management and access control across many applications and networks. Commands introduced for this feature are: ssh server certificate ssh server trustpoint This command is modified for this feature: ssh server algorithms host-key

How X.509v3 certificate authentication works

Describes how SSH peers obtain, exchange, validate, and map X.509v3 certificates during server and user authentication on Cisco IOS XR routers.

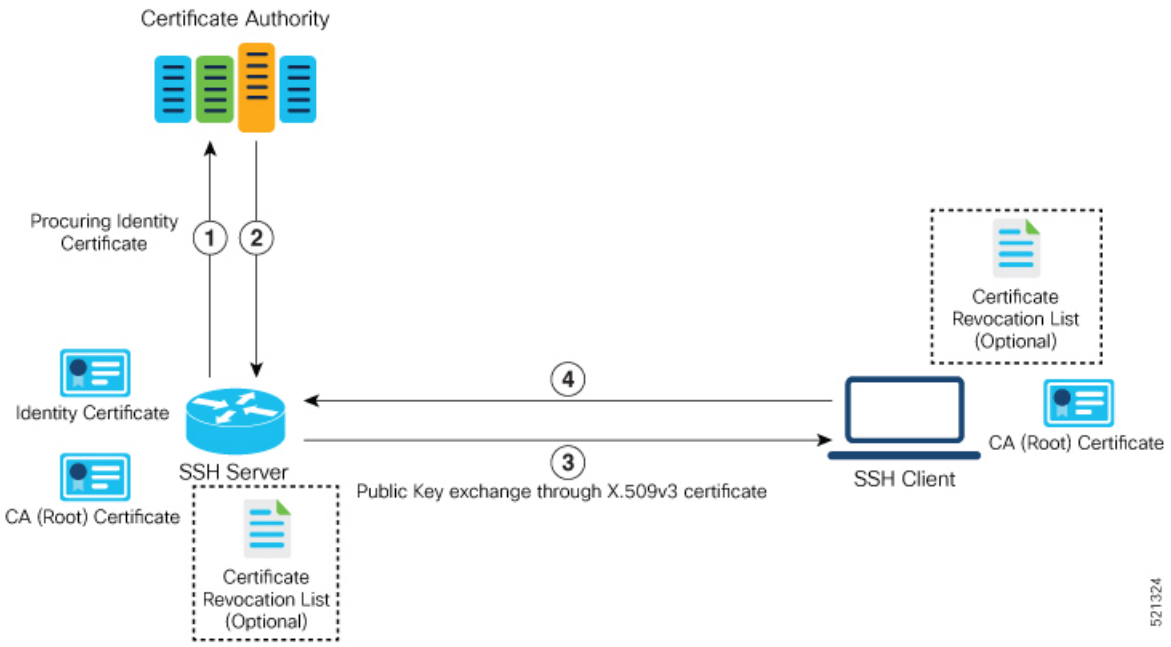
The SSH Transport Layer Protocol (TLP) authenticates the server to the client. The User Authentication Protocol (UAP) authenticates the client user to the server after the transport layer is established.

Summary

A Certificate Authority (CA) issues identity and root certificates. Each SSH peer validates the certificate presented by the other peer before continuing the connection.

Workflow

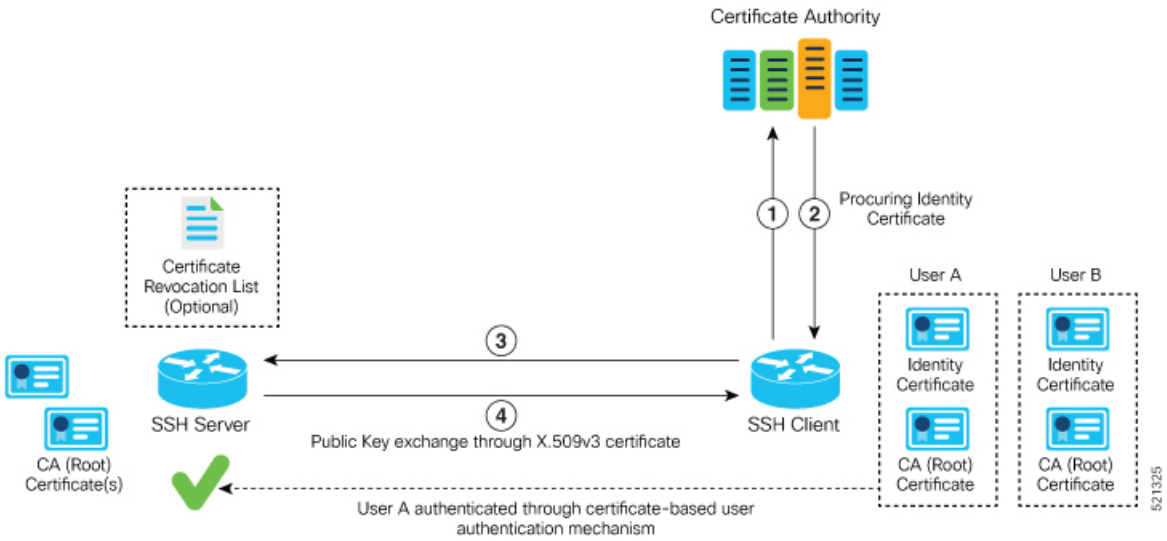
Figure 9: Server authentication using X.509v3 certificate



These stages describe server authentication using an X.509v3 certificate:

1. The SSH server obtains a valid identity certificate from a trusted CA. The server can obtain the certificate manually or through an enrollment protocol such as SCEP.
2. The CA provides the identity certificate and associated root certificates. The SSH server stores these certificates locally.
3. During transport-layer authentication, the SSH server presents its identity certificate to the SSH client.
4. The SSH client validates the server certificate. When validation succeeds, the client continues to the next phase of SSH connection establishment.

Figure 10: User authentication using X.509v3 certificate



These stages describe user authentication using an X.509v3 certificate:

5. After the SSH transport layer is established, the SSH client obtains a valid user identity certificate from a trusted CA.
6. The CA provides the user identity certificate and associated root certificates. The SSH client stores these certificates locally.

7. The SSH client sends a user-authentication request and presents the user certificate to the SSH server.
8. The SSH server validates the user certificate and checks its revocation status against a CRL.
9. The SSH server extracts the username from the certificate subject common name or subject alternative name and presents it to the AAA server for authorization.
10. The SSH server compares the extracted username with the username in the SSH connection request. When the usernames match and authorization succeeds, the server continues the SSH connection.

Result

The SSH connection advances only when certificate validation and username matching succeed.

X.509v3 certificate authentication restrictions

Describes endpoint roles, AAA methods, revocation checks, certificate-chain limits, trustpoint behavior, and failure conditions for X.509v3 SSH authentication on Cisco IOS XR routers.

Review these restrictions before enabling X.509v3 authentication.

X.509v3 SSH authentication has these restrictions:

- Cisco IOS XR supports the feature only when the router acts as the SSH server, not as the SSH client.
- Public-key authentication requires locally configured users because RADIUS and TACACS+ do not authenticate public keys. Include the **local** method in the AAA authentication list. Remote TACACS+ authorization and accounting can still apply.
- Online Certificate Status Protocol (OCSP) certificate verification is not supported. The router checks revocation status with a certificate revocation list (CRL).
- A user certificate chain cannot exceed nine certificates.
- The router supports up to ten user trustpoints. User trustpoints apply to all users and all VRFs.
- The host trustpoint applies to SSH in all VRFs.

Failure conditions

- Without a configured host trustpoint and its router certificate, the server omits **x509v3-ssh-rsa** from the supported host-key methods.
- User authentication fails when the certificate format is incorrect, the certificate chain exceeds nine certificates, or certificate verification fails.

Diagnostic messages

- **SECURITY-PKI-6-LOG_INFO** reports successful peer-certificate verification.
- **SECURITY-PKI-3-ERR_GENERAL** reports that the user certificate issuer was not found in the configured trustpoints.
- **SECURITY-SSHD-4-WARNING_X509** reports that the server could not retrieve the host certificate chain and will not use X.509 host authentication.

Configure X.509v3 certificate authentication

Configures host and user trustpoints, certificate username selection, authentication limits, AAA handling, and verification for X.509v3 SSH server and user authentication.

Enable X.509v3 certificate-based server and user authentication for SSH.

Before you begin

- Create the PKI trustpoints and install the required CA, router, and user certificates.
- Configure locally defined user accounts and include the local method in the AAA authentication list.

Procedure

1. Configure the X.509v3 RSA host-key algorithm and host trustpoint.

   Example

   ```
   Router# configure
   Router(config)# ssh server algorithms host-key x509v3-ssh-rsa
   Router(config)# ssh server certificate trustpoint host sample-host-tp
   Router(config)# commit
   ```

   The server offers host-key algorithms according to the available keys and certificates. Without an explicit host-key configuration, it offers all available algorithms. In addition to **x509v3-ssh-rsa**, you can configure **ecdsa-sha2-nistp521**, **ecdsa-sha2-nistp384**, **ecdsa-sha2-nistp256**, **ssh-rsa**, and **ssh-dsa**. Without the host trustpoint, the server does not offer **x509v3-ssh-rsa** for server verification. The host trustpoint applies to SSH in all VRFs.

2. Configure one or more trustpoints for user authentication.

   Example

   ```
   Router# configure
   Router(config)# ssh server trustpoint user sample-user-tp1
   Router(config)# ssh server trustpoint user sample-user-tp2
   Router(config)# commit
   ```

   You can configure up to ten user trustpoints. The configured trustpoints apply to all users and to SSH in all VRFs. Without a user trustpoint, the SSH server does not use certificates to authenticate users.

3. Select the certificate field that supplies the SSH username.

   Example

   ```
   Router# configure
   Router(config)# ssh server certificate username common-name
   Router(config)# commit
   ```

   The router can use the common name from the certificate subject name, the user principal name from the subject alternative name **otherName** field, or both values. This example uses the subject common name.

4. Optionally set the total SSH authentication-attempt limit.

   Example

   ```
   Router# configure
   Router(config)# ssh server max-auth-limit 5
   Router(config)# commit
   ```

   The supported range is 4 through 20, and the default is 20. The separate password-method limit remains three attempts.

5. Configure local authentication and the required remote authorization method.

   Example

   ```
   Router# configure
   Router(config)# aaa authentication login default group tacacs+ local
   Router(config)# aaa authorization exec default group radius group tacacs+
   Router(config)# commit
   ```

   The AAA authentication configuration for public-key authentication is the same as the configuration for regular or keyboard-interactive authentication, except that the authentication method list must include the local method. The example uses local certificate authentication with remote TACACS+ or RADIUS authorization.

6. Verify the server and active-session certificate settings.

   Example

   ```
   Router# show ssh server
   Certificate Based := Yes
   Host Trustpoint := sample-host-tp
   User Trustpoints := sample-user-tp1 sample-user-tp2
   Router# show ssh session details
   Router# show ssh
   ```

   The server output identifies certificate authentication, the host and user trustpoints, and the negotiated session algorithms. Session output identifies the X.509 public-key authentication method.

The SSH server can present its X.509v3 host certificate and authenticate users whose certificates chain to a configured user trustpoint.

What to do next

Use the **show ssh session details** command to verify the algorithm selected for the SSH session, and use the **show ssh** command to verify that the authentication method is `x509-rsa-pubkey`.

To disable server authentication, remove **x509v3-ssh-rsa** from the allowed host-key algorithms. To disable user authentication, remove the user trustpoint configuration.

X.509v3 certificate extension validation over mTLS

Explains how Cisco IOS XR routers process RFC 5280 extensions with different criticality settings in X.509v3 client certificates during mutual TLS validation.

X.509v3 certificate extension validation over mutual TLS (mTLS) is a capability that

- acknowledges the RFC 5280 extensions in a client certificate
- processes extensions with different criticality settings, and
- allows certificate authentication to continue when the presented extensions are valid.

Before Cisco IOS XR Software Release 7.9.1, the router could fail to process a critical certificate extension, causing authentication to fail. From Release 7.9.1, a client certificate can contain any number of RFC 5280 extensions with different criticality settings.

Table 71: Feature History Table

Feature Name	Release Information	Feature Description
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Validating X.509v3 Certificate Extensions over Mutual Transport Layer Security (mTLS)	Release 7.9.1	With this feature, the router can handle the X.509v3 Certificates Extensions defined in RFC 5280 while validating the client certificate over mTLS. Here, the router acknowledges all extensions in X.509v3 Certificates of the user while validating it. Previously, the router failed to process certification extensions when the severity was critical and resulting in authentication failure. This feature permits users to configure any certificate extensions with different severity in their X.509v3 Certificates.

Related commands

- **ssh server algorithms host-key**

- **ssh server certificate username**
- **ssh server max-auth-limit**
- **ssh server trustpoint host**
- **ssh server trustpoint user**
- **show ssh server**
- **show ssh session details**

OpenSSH certificate authentication

Explains how an OpenSSH CA signs host and user certificates to establish mutual trust between a client and a Cisco IOS XR router.

OpenSSH certificate authentication is a two-way SSH trust model that

- uses an SSH key as a CA to sign host and user certificates
- authenticates the router to the client and the client user to the router, and
- avoids first-connection trust establishment when both certificates chain to the trusted CA.

The certificate records the subject public key, signing CA, type, validity, key ID, and serial number. A user certificate also records principals that the router matches against the login username.

Supported key types

OpenSSH certificates support RSA, DSA, ECDSA, and Ed25519 keys. User certificates can also contain supported critical options and extensions created with **ssh-keygen** utility.

Table 72: Feature History Table

Feature Name	Release Information	Feature Description
OpenSSH Certificate based Authentication for Router	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
OpenSSH Certificate based Authentication for Router	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
OpenSSH Certificate based Authentication for Router	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
OpenSSH Certificate based Authentication for Router	Release 7.5.3	You can now use OpenSSH certificates to authenticate to the remote routers from a client machine. This feature uses the ssh-keygen utility, a standard SSH component to generate and manage authentication keys, available in OpenSSH to create a CA (Certificate Authority) like infrastructure for logging into the router. In this feature, the certificates that are used to authenticate router and client are both signed by the same CA. This automatically establishes trust between router and client, and eliminates the need to establish trust, while using the client for remote logging to router for the first time.

How OpenSSH certificate authentication works

Describes how an OpenSSH CA, router trustpoint, host certificate, and user certificate establish mutually authenticated remote access to a Cisco IOS XR router.

A system with OpenSSH and **ssh-keygen** provides the certificate-signing infrastructure.

Summary

The router and client trust the same CA. The CA signs a router host certificate and a user certificate before the client opens the SSH connection.

Workflow

Figure 11: Router authentication workflow

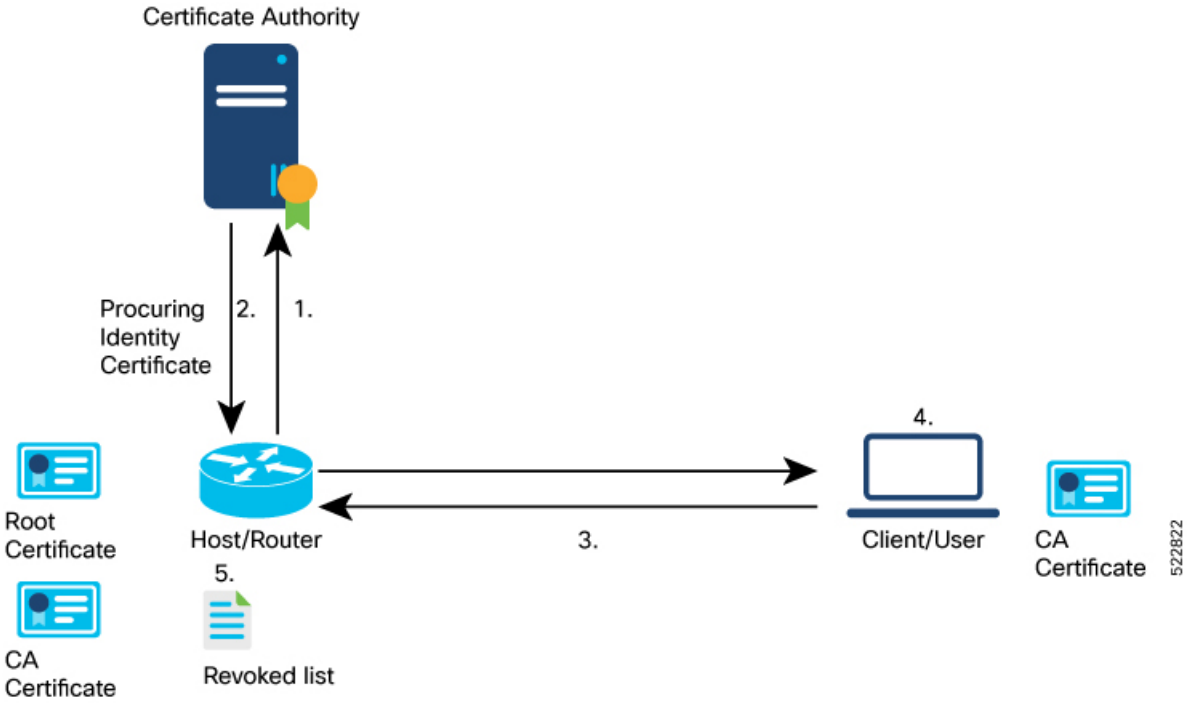
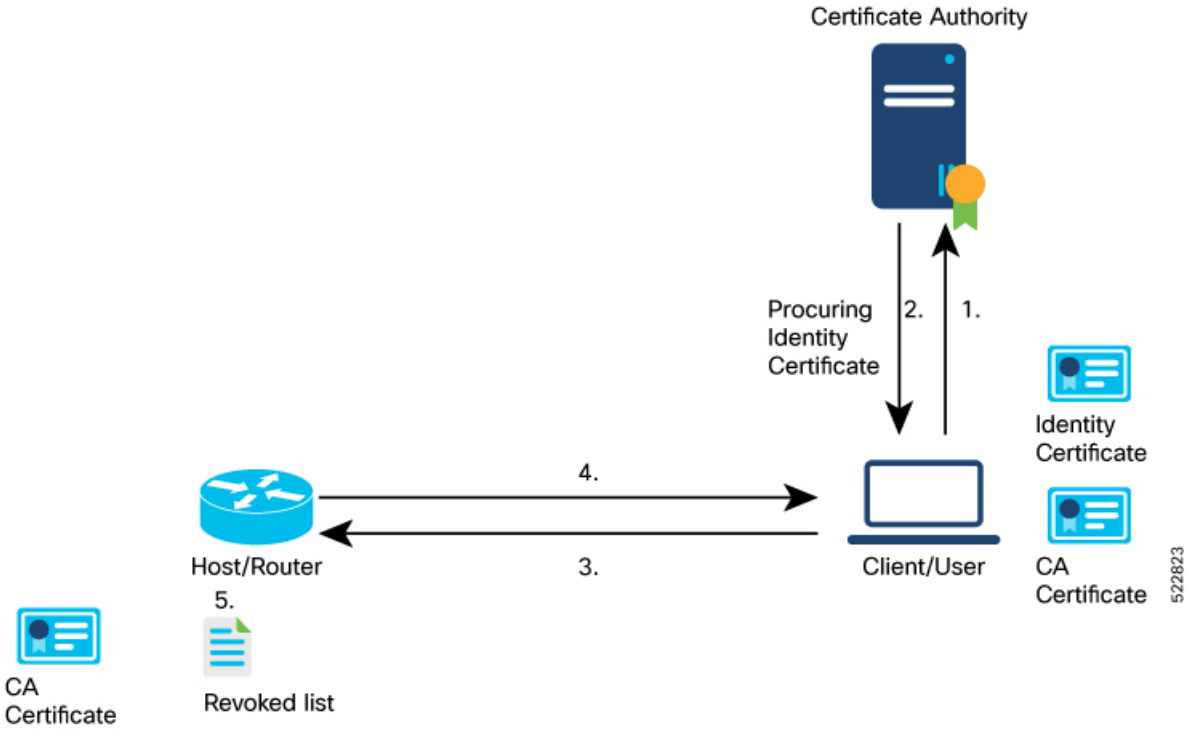


Figure 12: User authentication workflow



These stages describe OpenSSH certificate authentication:

1. An administrator creates an OpenSSH trustpoint on the router and assigns it to host and user authentication.
2. The CA system generates an SSH CA key pair. Its public key is installed in the router trustpoint.
3. The router generates a certificate-signing request. The CA signs it as a host certificate, and the signed certificate is imported into the router.

4. The client generates a user key pair. The CA signs the public key with the login username as a principal, and the signed user certificate is installed on the client.
5. The client trusts the CA for the router host and presents its user certificate during login. The router validates the certificate, matches its principal to the login username, and permits authorized access.

Result

The host and user certificates establish a mutually authenticated SSH channel.

Prerequisites for OpenSSH certificate authentication

Provides requirements for the OpenSSH CA system, router trustpoint, signed host and user certificates, certificate principals, matching private keys, and authorized usernames.

Establish CA trust and valid identities before an OpenSSH certificate login.

- Use a CA system that has OpenSSH and the **ssh-keygen** utility.
- Install the CA public key in an OpenSSH trustpoint on the router.
- Assign a trustpoint for host authentication and up to ten trustpoints for user authentication.
- Import a valid CA-signed host certificate into the router.
- Install the CA-signed user certificate and its matching private key on the client.
- Set a user-certificate principal that matches the SSH login username.
- Configure the username locally on the router unless TACACS+ certificate-user authorization is enabled.

The router and client can authenticate each other only when their certificates are valid, signed by a trusted CA, and associated with the requested identities.

Configure OpenSSH certificate authentication

Configures a router trustpoint, CA-signed host and user certificates, client trust, certificate verification, and an authenticated OpenSSH connection to a Cisco IOS XR router.

Establish certificate-based authentication between an OpenSSH client and a Cisco IOS XR router.

Before you begin

- Provide a CA system and client system with OpenSSH and **ssh-keygen**.
- Provide secure methods to transfer the CA public key, certificate-signing request, signed host certificate, and signed user certificate.

Procedure

1. Establish the OpenSSH trustpoint on the router and generate its CA key pair on the CA host.
   a) On the router, create an OpenSSH trustpoint and assign it to host and user authentication.

Example

```
Router# configure
Router(config)# crypto ca openssh trustpoint sample-openssh-ca
Router(config)# ssh server openssh trustpoint host sample-openssh-ca
Router(config)# ssh server openssh trustpoint user sample-openssh-ca
Router(config)# commit
```

b) On the CA host, generate the CA key pair and display its public key.

Example

```
ca-host$ ssh-keygen -t rsa -f sample-ca-key
ca-host$ cat sample-ca-key.pub
```

Leave the passphrase empty.

2. Create, install, and verify the CA-signed router host certificate.
   a) On the router, install the CA public key in the trustpoint and verify it.

Example

```
Router# crypto ca openssh authenticate sample-openssh-ca
Enter the CA pubkey.
End with a blank line or the word "quit" on a line by itself
Do you accept this certificate? [yes/no]: yes
Router# show crypto ca openssh certificates
```

Paste the complete CA public key at the prompt.

   b) On the router, generate and display the host certificate-signing request.

Example

```
Router# crypto ca openssh enroll sample-openssh-ca
Display Certificate Request to terminal? [yes/no]: yes
```

Copy only the host-key content between the host-key markers to a file named `sample-host.pub` on the CA host.

   c) On the CA host, sign the router host key and display the signed certificate.

Example

```
ca-host$ ssh-keygen -h -s sample-ca-key -I "sample-router" -V +10w -z 10
sample-host.pub
ca-host$ cat sample-host-cert.pub
```

The example creates a host certificate with identity **`sample-router`**, a 10-week validity period, and serial number 10.

   d) On the router, import the signed host certificate and verify its properties.

Example

```
Router# crypto ca openssh import sample-openssh-ca certificate
Enter the OpenSSH certificate.
End with a blank line
Router# show crypto ca openssh certificates
```

The output lists the CA certificate and router host certificate with the host certificate's key ID, serial number, and validity period.

3. Create and install the CA-signed user certificate and authorize its username on the router.
   a) On the client host, generate the user key pair and display its public key.

Example

```
client-host$ ssh-keygen -t rsa -f sample-user-key
client-host$ cat sample-user-key.pub
```

b) On the CA host, sign the user public key with a principal that matches the router username.

Example

```
ca-host$ ssh-keygen -s sample-ca-key -I "sample-user-cert" -V +10w -n
sample-user -z 20 sample-user-key.pub
ca-host$ ssh-keygen -Lf sample-user-key-cert.pub
```

The user-certificate signing command uses these parameters:

Table 73: User-certificate signing parameters

Parameter	Description	Example value
CACert	Specifies the filename of the CA private key that signs the user public key.	`sample-ca-key`
IdentityOfSysReqCert	Specifies the identity assigned to the user certificate.	**`sample-user-cert`**
CertValidity	Specifies the certificate validity period.	**`+10w`**
Username	Specifies the principals added to the certificate. During authentication, a principal must match the SSH login username. Separate multiple principals with commas.	**`sample-user`**
CertSerialNo	Specifies the certificate serial number.	`20`
CopiedUserCertName	Specifies the filename containing the user public key copied from the client host for signing.	`sample-user-key.pub`

Note

In addition to the mandatory user-certificate fields, you can configure supported critical options and extensions. See the **ssh-keygen** documentation for details.

c) On the client host, install the signed user certificate and trust the CA for the router host.

Place `sample-user-key-cert.pub` with its matching private key. Add an **@cert-authority** entry containing the router hostname and CA public key to the client's `known_hosts` file.

d) On the router, configure the username that matches the user-certificate principal.

Example

```
Router# configure
Router(config)# username sample-user
Router(config-un)# group root-lr
Router(config-un)# commit
```

4. On the client host, connect to the router with the signed user certificate and matching private key.

Example

```
client-host$ ssh -o CertificateFile=sample-user-key-cert.pub -i sample-user-key
 sample-user@192.0.2.2 -o StrictHostKeyChecking=yes
```

The client validates the router host certificate, and the router validates and authorizes the signed user certificate.

TACACS+ authorization for OpenSSH certificate users

Explains how a router validates an OpenSSH user certificate and uses centralized TACACS+ profiles to authorize router access.

TACACS+ authorization for OpenSSH certificate users is an access-control method that

- validates the OpenSSH user certificate on the router
- queries an external TACACS+ server for the corresponding user profile, and
- authorizes router access by using the user group returned by the TACACS+ server.

Table 74: Feature History Table

Feature Name	Release Information	Feature Description
Certificate-based user authentication using TACACS+ server	Release 7.5.4	This feature enables the router login for users in the remote TACACS+ server using the certificate-based authentication methods. Here, the router authenticates a user using the OpenSSH certificates and authorizes access according to the configurations available for that user in the external TACACS+ server. This feature provides an option to configure the users in a centralized TACACS+ server and use them across multiple routers in a network. Thus, it helps you overcome the hassles of configuring users in each router individually while authenticating users based on certificates.This feature introduces the aaa enable-cert-authentication command.

Authorization behavior

After certificate validation succeeds, the router uses the configured **aaa authorization exec** method list to query the TACACS+ server.

- If the TACACS+ server finds a matching user profile, it authorizes the user and returns the associated user group to the router.

- If the TACACS+ server does not find a matching profile, the router permits or denies access according to the remaining methods in the authorization list, such as the local database.

Centralized user management

Centralized TACACS+ profiles eliminate the need to configure the same certificate-user profile separately on every router in the network.

Restriction

Certificate-based access for users defined on an external TACACS+ server is supported only by the OpenSSH implementation.

Authorize OpenSSH certificate users through TACACS+

Configures a router to validate OpenSSH user certificates and use centralized TACACS+ profiles for EXEC authorization and access control.

Authorize OpenSSH certificate users through a centralized TACACS+ server instead of defining every user separately on each router.

Before you begin

- Enable OpenSSH certificate authentication on the router.
- Configure the user profiles on the external TACACS+ server.
- Configure the TACACS+ server or server group on the router.
- Use the OpenSSH implementation. External TACACS+ certificate users are not supported by the legacy SSH implementation.

Procedure

1. Enable OpenSSH certificate-user access and TACACS+ EXEC authorization.

   Example

   ```
   Router# configure
   Router(config)# aaa enable-cert-authentication
   Router(config)# aaa authorization exec default group tacacs+ local
   Router(config)# commit
   ```

   The local fallback applies according to the configured authorization method list.

2. Verify the running AAA configuration.

   Example

   ```
   Router# show running-config aaa
   aaa enable-cert-authentication
   aaa authorization exec default group tacacs+ local
   ```

After the router validates an OpenSSH user certificate, it queries TACACS+ for the user profile and applies the returned user group. If no TACACS+ match exists, the remaining methods in the authorization list determine access.

Public-key authentication for SSH clients

Explains how passwordless public-key authentication works when a Cisco IOS XR router acts as an SSH, SFTP, or SCP client for a locally configured user.

SSH client public-key authentication is a passwordless method that

- generates a user-specific RSA key pair on the router acting as the client
- installs the public key on the remote SSH server, and
- uses the private key on the client to prove the user's identity.

The method also supports passwordless SFTP and SCP operations for locally configured users.

Table 75: Feature History Table

Feature Name	Release Information	Feature Description
Public Key-Based Authentication of SSH Clients on Cisco IOS XR Routers	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Public Key-Based Authentication of SSH Clients on Cisco IOS XR Routers	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Public Key-Based Authentication of SSH Clients on Cisco IOS XR Routers	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description

Public Key-Based Authentication of SSH Clients on Cisco IOS XR Routers	Release 7.10.1	You are now assured of cryptographic strength even as you avail of automated password-less login while establishing SSH connections with the server. With the password and keyboard-interactive authentication, Cisco IOS XR routers configured as SSH clients now support public key-based authentication. In this authentication method, passwords need not be sent over the network; hence, it provides an additional layer of security and aids in automation processes. This feature is available only for users locally configured on the router; not those configured on remote servers.
		
		Previous releases supported SSH public key-based authentication only for Cisco IOS XR routers configured as SSH servers.
		
		The feature introduces these changes:
		• CLI:
		  • **crypto key generate authentication-ssh rsa**
		  • **crypto key zeroize authentication-ssh rsa**
		  • **show crypto key mypubkey authentication-ssh rsa**
		• Yang Data Models:
		New Xpaths for:
		  • `Cisco-IOS-XR-crypto-act.yang`
		  • `Cisco-IOS-XR-crypto-epki-newoper.yang`
		
		(see GitHub, YANG Data Models Navigator)

Security characteristics

The private key is not sent over the network. The server challenges the client with the stored public key, and only the client holding the corresponding private key can produce the required response.

How SSH client public-key authentication works

Describes how a router acting as an SSH client proves possession of a private key to a server that stores the matching public key.

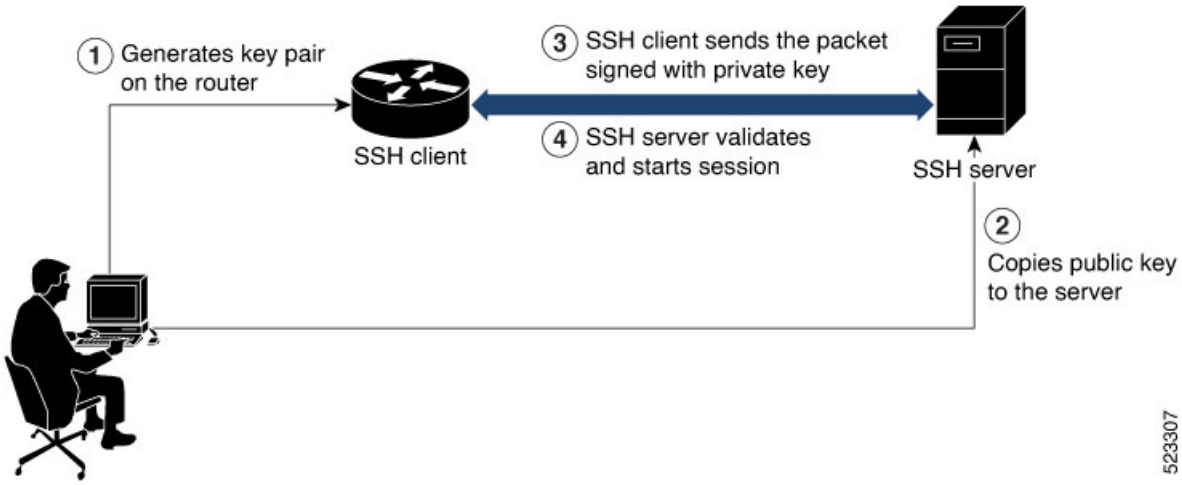
The router generates and retains an RSA authentication key pair for a locally configured user.

Summary

The server stores the public key. The client keeps the private key and uses it to answer the server's authentication challenge.

Workflow

Figure 13: Public key-based authentication of SSH clients: Work flow



These stages describe SSH client public-key authentication:

1. An administrator generates an RSA authentication key pair on the router acting as the SSH client.
2. The administrator installs the public key in the remote user's authorized-key store.
3. The client requests an SSH connection for the locally configured user.
4. The server uses the stored public key to challenge the client. The client uses the corresponding private key to produce the valid response.
5. The server verifies the response and authenticates the user without requesting the account password.

Result

The router establishes a passwordless SSH, SFTP, or SCP connection to the remote server.

Guidelines for public-key authentication of SSH clients

Provides RSA-key, local-user, privilege, public-key installation, and connection requirements for a Cisco IOS XR router acting as an SSH client.

Prepare the client key pair and server authorization before requesting public-key authentication.

- Use an RSA authentication key; the client feature supports only RSA.
- Configure the user locally on the router. RADIUS and TACACS+ do not authenticate direct public keys.
- Use a root-privileged account to create or delete another user's keys.
- Install the correct public key for the user on the SSH server.
- Generate the authentication key before starting the connection. Without a key, the client does not attempt public-key authentication.

A missing client key or an incorrect server-side public key causes user authentication to fail.

Enable public-key authentication for an SSH client

Configures RSA public-key authentication by generating a client key pair, installing the public key on an SSH server, and opening a passwordless connection.

Authenticate a router acting as an SSH client without sending a user password.

Before you begin

- Configure the user locally on the router.
- Obtain administrator access or existing user credentials that permit installation of the public key on the remote server.

Procedure

1. Generate the RSA authentication key pair.

   Example

   ```
   Router# crypto key generate authentication-ssh rsa
   How many bits in the modulus [2048]: 2048
   Generating RSA keys ...
   Done w/ crypto generate keypair
   [OK]
   ```

   The supported modulus range is 512 through 4096 bits.

2. Display and copy the generated public key in OpenSSH format.

   Example

   ```
   Router# show crypto key mypubkey authentication-ssh rsa
   Key label: sample-user
   Type : RSA Authentication
   Size : 2048
   OpenSSH Format:
   ssh-rsa AAAA...sample-public-key
   ```

3. Install the public key for the user on the remote SSH server.
   - On a Cisco IOS XR SSH server, import the key with **crypto key import authentication rsa**.
   - On a Linux SSH server, add the key to the user's `~/.ssh/authorized_keys` file.

4. Connect to the remote SSH server.

   Example

   ```
   Router# ssh sample-user@192.0.2.225
   ```

The router authenticates with its private key and opens the SSH connection without prompting for the remote account password.

Delete SSH client authentication keys

Deletes a locally configured user's RSA authentication key pair from a Cisco IOS XR router acting as an SSH client and prevents its reuse.

Remove client authentication keys that are no longer authorized for outbound SSH connections.

Before you begin

Use root privileges when deleting keys that belong to another user.

Procedure

Delete the RSA authentication keys for the user.

Example

```
Router# crypto key zeroize authentication-ssh rsa username sample-user
```

The SSH client can no longer use the deleted key pair for public-key authentication.

Public-key authentication to SSH servers

Explains how a Cisco IOS XR router stores public keys and authenticates SSH clients that prove possession of the corresponding private keys.

Public-key authentication to an SSH server is a passwordless method that

- associates imported public keys with a router user account
- verifies a client signature with one of those keys, and
- permits access only when the signature and username match.

From Cisco IOS XR Software Release 7.11.1, a router supports up to four public keys per user. A user can therefore connect from as many as four client systems without requiring a separate router username for each system.

Table 76: Feature History Table

Feature Name	Release Information	Feature Description
Multiple Public Keys per User for Public Key-based Authentication	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D

Feature Name	Release Information	Feature Description
Multiple Public Keys per User for Public Key-based Authentication	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Multiple Public Keys per User for Public Key-based Authentication	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Multiple Public Keys per User for Public Key-based Authentication	Release 7.11.1	We provide greater flexibility to access secure routers by allowing four public keys to be used for authentication. With the ability to associate multiple public keys with your user account on the router, we've also simplified the authentication process by eliminating the need to create unique users for each SSH client device. The feature introduces these changes: CLI: • The **second**, **third**, and **fourth** keywords are introduced in the crypto key import authentication rsa command. • The **second**, **third**, and **fourth** keywords are introduced in the crypto key zeroize authentication rsa command. • The **second**, **third**, and **fourth** keywords are introduced in the keystring command.` YANG Data Models: • `Cisco-IOS-XR-crypto-act` • `Cisco-IOS-XR-um-ssh-cfg` (See GitHub, YANG Data Models Navigator)

Security characteristics

The private keys remain on their client systems. Signature verification is less exposed to brute-force password guessing and password theft than reusable password authentication.

How multiple public keys authenticate a user

Describes how a router matches an SSH client's signature against as many as four public keys associated with one user.

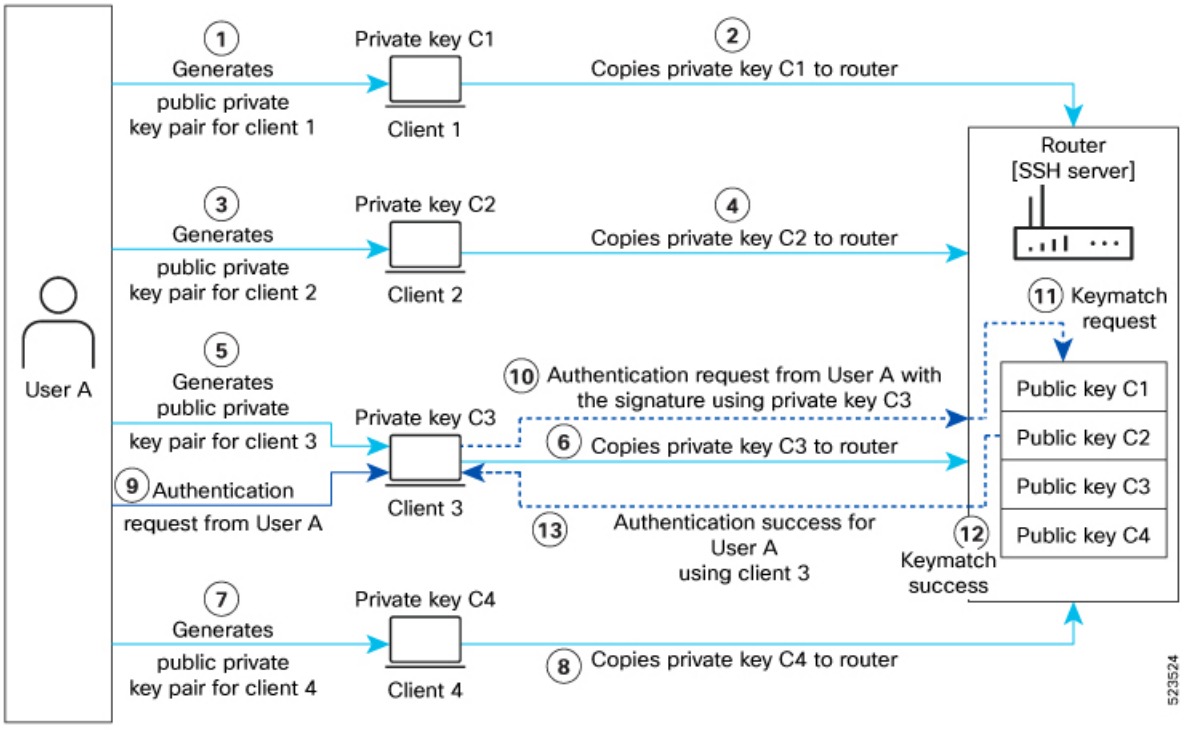
Each SSH client generates its own key pair. The router stores the corresponding public key under a shared user account.

Summary

A client proves possession of its private key by signing authentication data. The router tests the signature with the public keys stored for that user.

Workflow

Figure 14: Multiple public keys per user for public key-based authentication



These stages describe authentication with multiple public keys:

1. Up to four client systems independently generate public-private key pairs.
2. An administrator imports or configures each public key, one at a time, under the same router username.
3. A client requests access for that username and presents a signature created with its private key.
4. The router compares the signature with the public keys associated with the username.
5. If one public key verifies the signature, the router authenticates the user. If none verifies it, the router denies access.

Result

The same router user can authenticate from any client that holds one of the corresponding private keys.

Guidelines for public-key authentication to SSH servers

Provides key-count, import-format, security-key structure, operation-mode, and deletion requirements for user public keys stored on a Cisco IOS XR SSH server.

Add and manage each public key in a supported format and operation mode.

- Associate no more than four public keys with one user.
- Import or configure only one public key at a time.
- Use RSA, Base64, PEM PKCS#1, or PEM PKCS#8 when importing RSA public-key files.
- Import RSA keys with **crypto key import authentication rsa**.
- Import security keys with **ssh-server authorized-keys**. Include the complete security-key structure and application-data field; the router rejects malformed keys.
- Delete a key in the same XR configuration or XR EXEC mode that was used to import it.

Key-count, format, structure, and mode mismatches can prevent import or later key removal.

Configure public-key authentication to a router

Associates as many as four client public keys with one router user by file import or direct SSH key configuration.

Allow one router user to authenticate from multiple SSH client systems without a password.

Use either public-key file import or direct SSH key configuration. Add the keys sequentially.

Before you begin

- Generate the RSA key pair on each SSH client.
- Copy each public-key file to the router when using the import method.

Procedure

1. Create the router user for public-key import.

   Example

   ```
   Router# configure
   Router(config)# username sample-user1
   Router(config)# commit
   ```

2. Import as many as four client public-key files and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# crypto key import authentication rsa username sample-user1
   disk0:/sample-key1.pub
   Router(config)# crypto key import authentication rsa username sample-user1
   second disk0:/sample-key2.pub
   Router(config)# crypto key import authentication rsa username sample-user1
   third disk0:/sample-key3.pub
   Router(config)# crypto key import authentication rsa username sample-user1
   fourth disk0:/sample-key4.pub
   Router(config)# commit
   ```

   You can reference a supported remote file path instead of copying the file to local storage.

3. Alternatively, configure public-key strings directly under an SSH server username.

   Example

   ```
   Router# configure
   Router(config)# ssh server username sample-user2
   Router(config-user-key)# keystring ssh-rsa sample-public-key-1
   Router(config-user-key)# keystring ssh-rsa second sample-public-key-2
   Router(config-user-key)# keystring ssh-rsa third sample-public-key-3
   Router(config-user-key)# keystring ssh-rsa fourth sample-public-key-4
   Router(config-user-key)# commit
   ```

   Replace each sample key value with the complete OpenSSH public-key string from the corresponding client.

4. Connect from a client with the shared username.

Example

```
client-host$ ssh sample-user1@192.0.2.2
```

5. Verify imported keys and active public-key sessions.

Example

```
Router# show crypto key authentication rsa sample-user1 all
Router# show ssh
```

The key output lists the configured key labels, and active SSH sessions report RSA public-key authentication.

The user can access the router from any client that holds a private key corresponding to one of the configured public keys.

Delete user public keys from a router

Deletes all imported authentication keys, one user's imported keys, an SSH username configuration, or a selected configured key from a Cisco IOS XR router.

Revoke public-key access for all users, one user, or one client key.

Use the deletion method that matches the method and operation mode used to add the key.

Procedure

---

1. To revoke every imported RSA authentication key, delete all keys and commit the configuration.

Example

```
Router# configure
Router(config)# crypto key zeroize authentication rsa all
Do you really want to remove all these keys? [yes/no]: yes
Router(config)# commit
```

2. To revoke every imported RSA key for one user, delete that user's keys and commit the configuration.

Example

```
Router# configure
Router(config)# crypto key zeroize authentication rsa username sample-user
all
Do you really want to remove all these keys? [yes/no]: yes
Router(config)# commit
```

3. To remove all directly configured SSH keys for one user, delete the SSH server username configuration.

Example

```
Router# configure
Router(config)# no ssh server username sample-user
Router(config)# commit
```

4. To revoke only the third directly configured key, delete that key from the SSH username configuration.

   Example

```
Router# configure
Router(config)# no ssh server username sample-user keystring third
Router(config)# commit
```

The selected public keys no longer authorize SSH access to the router.

19 SSH Access and Connection Management

Topics:

- SSH access and connection controls
- FIDO2 authentication for SSH
- SSH authentication attempt limits
- Multifactor authentication for SSH
- Selective SSH server authentication methods
- SSH port forwarding
- DSCP marking for SSH packets
- SSH server timeouts

Explains authentication controls, multifactor access, port forwarding, packet marking, and timeout settings that secure and manage SSH connections on the routers.

Use this chapter to strengthen SSH authentication, control permitted connection behavior, and release resources held by inactive connections or channels.

The chapter covers these functional areas:

- FIDO2 and Duo multifactor authentication
- Authentication-attempt limits and permitted server authentication methods
- SSH port forwarding and DSCP marking during connection establishment
- Unused-connection and idle-channel timeouts

SSH access and connection controls

Explains the controls that authenticate SSH users, restrict access methods, protect forwarded application traffic, prioritize connection packets, and end inactive sessions on Cisco IOS XR routers.

SSH access and connection controls are security and resource-management functions that

- strengthen identity verification and limit failed authentication
- govern permitted SSH services and packet handling, and
- close inactive connections and channels before they exhaust router resources.

These controls include FIDO2 and Duo MFA, authentication-method selection, port forwarding, DSCP marking, and separate timeouts for unused connections and idle channels.

FIDO2 authentication for SSH

Explains how FIDO2 hardware security keys provide passwordless SSH authentication while retaining private keys on the authenticator and requiring user presence.

FIDO2 authentication for SSH is an open authentication standard that

- enables passwordless SSH login and multifactor authentication
- stores the private key only on a hardware security key, and
- requires user presence, such as touch or biometric confirmation, during login.

The client sends the registered public key during authentication. The server verifies that the key exists in the authorized-key database for the user, and the hardware device signs the authentication request without exposing its private key.

Table 77: Feature History Table

Feature Name	Release Information	Feature Description
FIDO2 authentication for SSH	Release 26.2.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) FIDO2 support for SSH enables secure, passwordless logins by using hardware security keys to store private keys. This feature requires physical user presence to authenticate sessions and protects against phishing by verifying a signature from the security device during the login process.

Supported algorithms

FIDO2 SSH authentication supports **ed25519-sk** and **ecdsa-sk** public-key types.

FIDO2 authentication guidelines

Provides guidelines for storing, importing, and using FIDO2 public keys for local SSH authentication on Cisco IOS XR routers.

Follow these guidelines when using FIDO2 authentication for SSH:

- Use FIDO2 authentication only for local SSH public-key authentication.
- Store FIDO2 public keys in the `authorized_keys` file.
- Use the **ssh-server authorized-keys** command to import FIDO2 public keys.
- The private key never leaves the FIDO2 device.
- User presence, such as touch or biometric confirmation, is mandatory during SSH authentication.
- You can import multiple FIDO2 public keys for a single user.

Keeping the private key on the FIDO2 device and requiring user presence protects the authentication process from credential theft and unauthorized key use.

The router authenticates the user with a registered FIDO2 public key while the corresponding private key remains on the FIDO2 device.

FIDO2 authentication restrictions

Describes remote authentication, key storage, SSH implementation, key-type, automation, and key-import restrictions for FIDO2 authentication on Cisco IOS XR routers.

Review these restrictions before configuring FIDO2 authentication for SSH.

FIDO2 authentication for SSH has these restrictions:

- Remote authentication through TACACS+ or RADIUS is not supported for FIDO2 keys.
- FIDO2 public keys cannot be stored in the CEPKI system database.
- XRSSH does not support FIDO2 authentication.
- Only **ed25519-sk** and **ecdsa-sk** key types are supported.
- Automation of the complete authentication flow is limited because user presence is mandatory.
- You cannot import FIDO2 keys by using the **crypto key import authentication** command.

Configure FIDO2 authentication for SSH

Set up FIDO2-based SSH authentication using a YubiKey (or other compatible device) for secure client access to Cisco IOS XR routers.

Set up FIDO2-based SSH authentication on Cisco IOS XR routers, enabling secure logins using a hardware security key such as a YubiKey.

FIDO2 authentication strengthens device access security by requiring both the user's credentials and physical presence of a hardware-backed authenticator. This protects against phishing attacks and credential theft. Use this method when you need to enhance SSH security for administrators or automation systems accessing Cisco IOS XR routers, or when organizational policy mandates strong, multi-factor authentication.

Before you begin

- Ensure you have a supported client - macOS, Windows, or any OS running OpenSSH 8.2+.
- Obtain a FIDO2-compatible device such as a YubiKey.
- Access privileges on the target Cisco IOS XR router.

Procedure

1. Connect your FIDO2 device to the client machine.

   Plug in your YubiKey to your client (macOS, Windows, or supported system).

2. Generate FIDO2 key pairs on your client.

   Example

   ```
   ssh-keygen -t ecdsa-sk -f id_ecdsa_sk
   ```

   a) In your terminal, run **ssh-keygen -t ecdsa-sk -f id_ecdsa_sk** command.
   b) When prompted, touch your authenticator of YubiKey to authorize key generation.
   c) Enter a passphrase for the key if desired, then confirm the passphrase.
   d) A public key (`id_ecdsa_sk.pub`) and private key will be generated.

3. Import your public key to the IOS XR router.
   a) Copy the contents of `id_ecdsa_sk.pub` from your client.
   b) On the router, enter EXEC mode and run the **ssh-server authorized-keys** command.
   c) If you do not provide a username, the key is imported for the current logged-in user.
   d) Paste the public key when prompted.

4. Verify the imported SSH key on the IOS XR router.

   Example

   ```
   Router#show ssh server authorized-keys user <username>
   ```

   a) Confirm your key appears in the authorized keys list.

5. Test SSH access using the FIDO2 key.

   Example

   ```
   Router#show ssh
   SSH version : Cisco-2.0
   id       chan pty     location        state           userid    host
            ver authentication connection type
   ----------------------------------------------------------------------------------------
   Incoming sessions
   974       0   vty1    0/RP0/CPU0      SESSION_OPEN    cafyauto  10.189.219.24
            v2  sk-ecdsa-sha2-nistp256@openssh Command-Line-Interface
   ```

   a) On your client, use the command:

      Example

      ```
      ssh -i id_ecdsa_sk <username>@<router_ip>
      ```

   b) When prompted, touch your YubiKey to confirm presence.
   c) Upon successful authentication, you will see a login confirmation.

6. Optional: Import multiple public keys for a user.

You can import additional keys using step 3 for a specific username; the behavior remains as with previous (non-FIDO) public keys.

You have successfully configured FIDO-based SSH authentication for enhanced security. SSH connections now require your FIDO device for login.

SSH authentication attempt limits

Explains the overall SSH authentication-attempt limit and the separate fixed limit for password authentication on Cisco IOS XR SSH servers.

The SSH authentication-attempt limit is a server-wide control that

- counts attempts across public-key, certificate-based, keyboard-interactive, and password authentication
- denies the connection when the overall limit is reached, and
- applies the same configured value to every user.

The configurable range is 3 through 20, with a default of 20. Before Release 7.3.2, the range was 4 through 20. Password authentication retains a separate maximum of three attempts regardless of the overall setting.

Table 78: Feature History Table

Feature Name	Release Information	Feature Description
User Configurable Maximum Authentication Attempts for SSH	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
User Configurable Maximum Authentication Attempts for SSH	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
User Configurable Maximum Authentication Attempts for SSH	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
User Configurable Maximum Authentication Attempts for SSH	Release 7.3.1	This feature allows you to set a limit on the number of user authentication attempts allowed for SSH connection, using the three authentication methods that are supported by Cisco IOS XR. The limit that you set is an overall limit that covers all the authentication methods together. If the user fails to enter the correct login credentials within the configured number of attempts, the connection is denied and the session is terminated. This command is introduced for this feature: ssh server max-auth-limit

SSH authentication attempt restrictions

Describes the server-role, user-scope, and password-specific restrictions that apply when configuring maximum SSH authentication attempts on Cisco IOS XR routers.

Review these restrictions before configuring the maximum number of SSH authentication attempts.

Maximum SSH authentication attempts have these restrictions:

- The feature is available only when the Cisco IOS XR router functions as an SSH server. It does not apply when the router functions as an SSH client.
- The configuration is not user-specific. The configured limit is the same for all users.
- For security reasons, the SSH server permits a maximum of three authentication attempts that explicitly use password authentication. Configuring the maximum SSH authentication-attempt limit does not change this password-specific limit.

Password authentication example

If you configure the overall maximum authentication-attempt limit as five by using **ssh server max-auth-limit 5** , the SSH server still permits only three attempts that explicitly use password authentication.

Set the maximum SSH authentication attempts

Sets and verifies the overall number of authentication attempts permitted for each incoming SSH connection to a Cisco IOS XR server.

Terminate SSH connections whose authentication failures reach the configured overall limit.

The setting applies only when the router acts as an SSH server and uses the same limit for all users. Password authentication remains limited to three attempts.

Procedure

1. Set the maximum authentication attempts and commit the configuration.

Example

```
Router# configure
Router(config)# ssh server max-auth-limit 5
Router(config)# commit
```

2. Verify the configured value.

Example

```
Router# show running-configuration ssh
ssh server max-auth-limit 5
ssh server v2
```

When the overall limit is reached, the router terminates the session and reports **Max authentication tries reached-exiting** in a **SECURITY-SSHD-3-ERR_GENERAL** Syslog message.

Multifactor authentication for SSH

Explains how SSH login combines user credentials with a token, cryptographic device, or mobile approval to reduce access risks caused by compromised or weak passwords.

Multifactor authentication (MFA) for SSH is a multistep identity-verification method that

- requires two or more verification factors
- combines something the user knows with something the user has, and
- reduces access risk from compromised or weak passwords.

Both the SSH client and server must support keyboard-interactive authentication. The default Cisco IOS XR client order is public-key, keyboard-interactive, and password authentication; you can change it with **ssh client auth-method**.

Table 79: Feature History Table

Feature Name	Release Information	Feature Description
Multi-Factor Authentication for SSH	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Multi-Factor Authentication for SSH	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Multi-Factor Authentication for SSH	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Multi-Factor Authentication for SSH	Release 24.1.1	You can now deploy robust authentication mechanisms for SSH connections to your routers and reduce security risks due to compromised or weak passwords. We now support multi-factor authentication (MFA)—a secure access management solution that verifies the identity of a user using multiple verification factors—for SSH login on Cisco IOS XR routers. These verification factors include a combination of login credentials such as username and password and a token, a cryptographic device, or a mobile phone with MFA application installed. No new commands or data models were introduced or modified as part of this feature.

How Duo multifactor authentication works

Describes how an SSH client, router, Cisco ISE, Duo Authentication Proxy, Duo cloud, and a mobile authenticator complete two-factor access.

The sample workflow uses a Cisco IOS XR router as the SSH server and Cisco ISE as the AAA server.

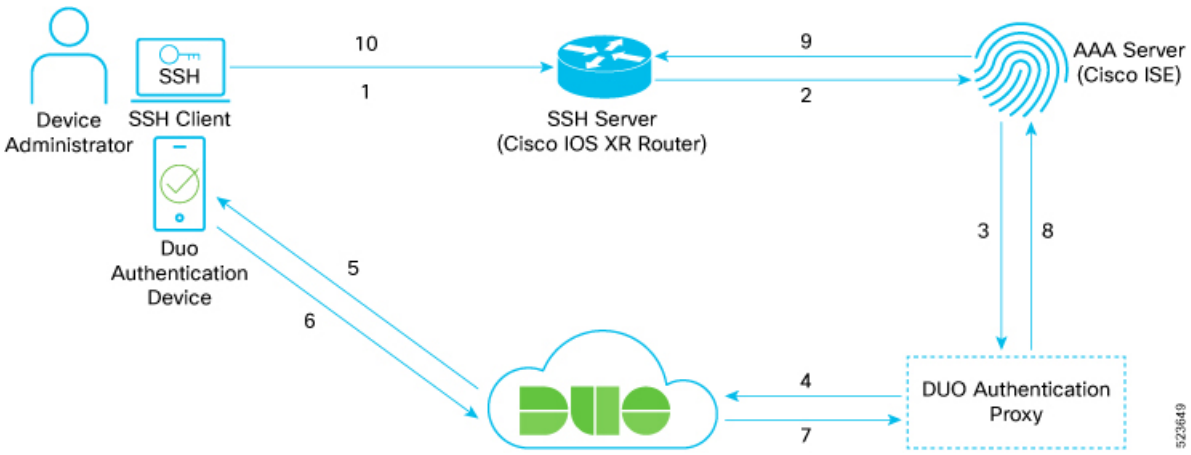
Summary

The workflow uses these components:

- SSH client: Initiates the administrator's connection.
- SSH server: Receives the connection on the router.
- Cisco ISE: Provides RADIUS or TACACS+ AAA.
- Duo Authentication Proxy: Coordinates primary authentication and contacts Duo for secondary authentication.
- Duo cloud and authentication device: Deliver and approve the second-factor request.

Workflow

Figure 15: MFA setup for an SSH connection: Sample topology



These stages describe Duo MFA for SSH:

1. The administrator initiates an SSH connection with credentials for a user configured in ISE.
2. The router forwards the request to Cisco ISE.
3. ISE sends the request to the Duo Authentication Proxy. The proxy returns it to ISE for first-factor authentication, and ISE reports the result.
4. After successful ISE authentication, the proxy requests second-factor authentication from Duo cloud.
5. Duo cloud sends a push notification to the administrator's Duo authentication device.
6. The administrator approves the notification.
7. Duo cloud reports approval to the proxy.
8. The proxy reports successful authentication to ISE.
9. ISE authorizes the administrator.
10. The administrator establishes the SSH connection to the router.

Result

The router grants SSH access after both verification factors and AAA authorization succeed.

Set up multifactor authentication for SSH

Coordinates Duo, Duo Authentication Proxy, Cisco ISE, router RADIUS attributes, and end-to-end verification tasks for multifactor SSH access to Cisco IOS XR routers.

Set up the sample Duo MFA topology for SSH access to a Cisco IOS XR router.

Before you begin

- Use Cisco IOS XR Software Release 24.1.1 or later and configure router AAA with ISE.
- Provide Cisco ISE as the RADIUS or TACACS+ server.
- Install Duo Authentication Proxy on Windows or Linux.
- Install the Duo application on the authentication device.

Procedure

1. Configure the Duo system and activate the mobile device.

2. Install and configure Duo Authentication Proxy.
3. Integrate Cisco ISE with Duo and configure device-administration policies.
4. Configure the router RADIUS attributes for the proxy.
5. Verify SSH login with the primary credentials and Duo second factor.

The SSH login request can traverse the router, ISE, Duo Authentication Proxy, and Duo cloud for multifactor authentication and authorization.

Configure the Duo system

Creates a Duo account, protects the Cisco ISE application, records its application credentials, adds a mobile authentication device, and activates Duo Mobile.

Prepare Duo cloud and the administrator's mobile authenticator for SSH MFA.

Procedure

---

1. Create a Duo account at duo.com.
2. Protect the Cisco ISE application and record its Duo credentials.

   For details, see the *First Steps* listed in https://duo.com/docs/radius.

   a) Sign in to Duo and select **Applications**.
   b) Search for Cisco ISE and select **Protect This Application**.
   c) Copy the Integration Key, Secret Key, and API Hostname to a secure location.
3. Add the Duo mobile device from **Dashboard > Users > username > Add Phone**.
4. Activate Duo Mobile from **Dashboard > 2FA Devices > phone-number > Activate Duo Mobile**.

The Duo application is protected, and the administrator's mobile device can receive second-factor requests.

Configure the Duo authentication proxy

Installs the Duo authentication proxy, configures its primary authenticator and RADIUS clients, and checks proxy startup, firewall access, and connectivity.

Relay primary authentication between ISE and the configured identity source and request the Duo second factor.

For more details, see https://duo.com/docs/authproxy-reference.

Procedure

---

1. Download and install the latest Duo authentication proxy on Windows or Linux machine.

   This example uses Windows Server 2016 for the primary proxy and Ubuntu for the secondary proxy.
2. Configure the proxy for your primary authenticator.

   Edit the Duo authentication proxy configuration file, `authproxy.cfg`, located in the conf subdirectory of the proxy installation path in the server using a text editor. You can add multiple ISE servers as RADIUS clients and multiple router subnets or IP addresses.
3. Start each proxy server and inspect its logs for configuration or connectivity errors.

On Windows, configure Windows Firewall to allow connections for the authentication proxy.

Configure Cisco Identity Services Engine

Integrates Cisco Identity Services Engine (ISE) with Duo authentication proxy, selects the RADIUS token identity source, creates device-administration policies, and onboards authorized users.

Use Cisco ISE for primary authentication and SSH user authorization in the Duo MFA workflow.

For more details, see Configure Duo Two Factor Authentication for ISE Management Access.

Procedure

---

1. Add the Duo authentication proxy as a RADIUS token server.

   Go to **Administration > Identity Management > External Identity Sources > RADIUS Token**, and select **Add**. Use the same shared secret configured in the authentication proxy.

   For details, see step 1 listed under ISE Configuration.

2. Select the configured RADIUS token server as the administrative identity source.

   Go to **Administration > System > Admin Access > Admin Access > Authentication Method**, and select the configured source, such as **RADIUS:DUO**.

   For details, see step 2 listed under ISE Configuration.

3. Create the device-administration policy set.

   Go to **Work Centers > Device Administration > Device Admin Policy Sets**.

   In this example, we created a policy set that matches on both protocols (RADIUS and TACACS+) with the **Allowed Protocols** set to **Default Device Admin**.

4. Configure authentication and authorization policies in the policy set.

   - Authentication policy: In this example, we have set a default rule to check the Identity Source Sequence that we defined in the steps above which contains the RADIUS Token Servers (Duo Authentication Proxies) and Active Directory.
   - Authorization policy: In this example, we have set a rule that checks if the authenticated user belongs either to the Domain Users or NS-ISE-IOS-Admins groups that we have configured in the active directory (AD). If the user belongs to one of these groups, then the system returns the pre-configured command sets and shell profile.

5. Synchronize the active directory users with Duo or add the users manually.

   For details, see Enroll user with Duo.

ISE can authenticate through the Duo proxy and return the authorized command sets and shell profile for approved users.

Configure router RADIUS attributes

Points a Cisco IOS XR router to Duo authentication proxy and configures the RADIUS authentication port, accounting port, and shared secret.

Send router RADIUS requests to the Duo authentication proxy.

For more details, see Configure Your RADIUS Clients.

Before you begin

Use the proxy address and the secret configured in the **radius_server_auto** section of `authproxy.cfg`.

Procedure

Configure the proxy as the RADIUS server and commit the shared attributes.

Example

```
Router# configure
Router(config)# radius-server host 209.165.200.225 auth-port 1812 acct-port
1813
Router(config-radius-host)# key sample-radius-secret
Router(config-radius-host)# commit
```

The router sends RADIUS authentication traffic to the Duo authentication proxy on port 1812 and accounting traffic on port 1813.

Verify multifactor authentication

Verifies an SSH login with Active Directory credentials and Duo approval and checks Cisco ISE RADIUS live logs and Duo Authentication Proxy logs for errors.

Confirm that primary authentication, Duo second-factor authentication, and SSH authorization complete successfully.

Procedure

1. Initiate an SSH connection from a client already added to ISE.
2. Sign in with the administrator's Active Directory credentials.
3. Confirm that the Duo authentication device receives the configured push or passcode request.
4. Approve the push or enter the correct passcode.

   The administrator is authenticated and authorized to access the router through SSH.
5. Check ISE RADIUS live logs for requests sent through the Duo Authentication Proxy.
6. Check the Authentication Proxy log for configuration, authentication, or connectivity errors.

The verified connection demonstrates successful primary authentication, Duo approval, ISE authorization, and router access.

Selective SSH server authentication methods

Explains how a Cisco IOS XR SSH server permits selected password, keyboard-interactive, and public-key authentication methods and rejects attempts that use disabled methods.

Selective SSH server authentication is a control that

- allows password, keyboard-interactive, and public-key methods to be disabled individually
- limits clients to the remaining permitted methods, and
- rejects login attempts that use a disabled method.

Public-key authentication includes certificate-based authentication. Disabling public-key authentication therefore disables certificate-based authentication.

Table 80: Feature History Table

Feature Name	Release Information	Feature Description
Selective Authentication Methods for SSH Server	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Selective Authentication Methods for SSH Server	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Selective Authentication Methods for SSH Server	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Selective Authentication Methods for SSH Server	Release 7.8.1	You now have the flexibility to choose the preferred SSH server authentication methods on the router. These methods include password authentication, keyboard-interactive authentication, and public-key authentication. This feature allows you to selectively disable these authentication methods. By allowing the SSH clients to connect to the server only through these permitted authentication methods, this functionality brings in additional security for router access through SSH. Before this release, by default, the SSH server allowed all these authentication methods for establishing SSH connections. The feature introduces these changes: • CLI: New disable auth-methods command • YANG Data Model: New XPaths for `Cisco-IOS-XR-crypto-ssh-cfg.yang` Cisco native model (see GitHub)

Disable SSH server authentication methods

Disables selected SSH server authentication methods on a Cisco IOS XR router and verifies which permitted methods remain available to connecting clients.

Prevent clients from using an authentication method that is not permitted by the router's access policy.

This example disables keyboard-interactive authentication. You can similarly disable password or public-key authentication.

Procedure

1. Disable keyboard-interactive authentication and commit the SSH server configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server
   Router(config-ssh)# disable auth-methods keyboard-interactive
   Router(config-ssh)# commit
   ```

2. Verify the permitted SSH server authentication methods.

   Example

   ```
   Router# show ssh server
   Authentication Method Supported
   -----------------------------------
   PublicKey := Yes
   Password := Yes
   Keyboard-Interactive := No
   Certificate Based := Yes
   ```

The server rejects keyboard-interactive login attempts and continues to accept the other displayed methods.

SSH port forwarding

Explains how SSH tunneling carries application TCP/IP traffic through an encrypted channel between a local client and a remote application server.

SSH port forwarding is a tunneling method that

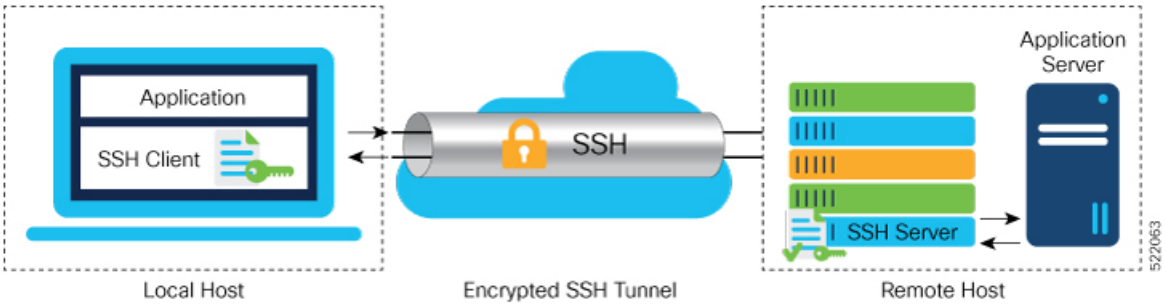
- accepts application traffic on a local client port
- carries that traffic through an encrypted SSH connection, and
- delivers it to a specified host and port through the SSH server.

Port forwarding protects otherwise insecure TCP/IP connections without modifying the application workflow. It can support legacy applications, VPN implementations, and intranet access across firewalls. The feature is disabled by default.

SSH port-forwarding model

An application on a local host can use an SSH client to connect securely to an application server on a remote host. The SSH server and application server can reside on the same router. In a data-center deployment, the SSH server can reside on one router and the application server can reside on another device.

Figure 16: SSH port forwarding



When port forwarding is enabled, the local application connects to a port on which the SSH client listens. The SSH client forwards the application traffic through an encrypted tunnel to the SSH server. The SSH server then connects to the application server, whether it resides on the same router or elsewhere in the same data center. This flow secures the complete application communication without requiring changes to the application or the user's workflow.

Table 81: Feature History Table

Feature Name	Release Information	Feature Description
SSH Port Forwarding with CiscoSSH	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
SSH Port Forwarding with CiscoSSH	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100]) (select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
SSH Port Forwarding with CiscoSSH	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]) (select variants only*); Modular Systems (8800 [LC ASIC: P100]) (select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
SSH Port Forwarding with CiscoSSH	Release 7.3.2	This release introduces SSH port forwarding with CiscoSSH, an OpenSSH-based implementation of SSH. CiscoSSH replaces Cisco IOS XR SSH, which is the older SSH implementation that existed prior to this release.
SSH Port Forwarding with Cisco IOS XR SSH	Release 7.3.15	With this feature enabled, the SSH client on a local host forwards the traffic coming on a given port to the specified host and port on a remote server, through an encrypted SSH channel. Legacy applications that do not otherwise support data encryption can leverage this functionality to ensure network security and confidentiality to the traffic that is sent to remote application servers. This feature introduces the ssh server port-forwarding local command.

How SSH port forwarding works

Describes how an SSH client creates a local forwarded port and how the SSH server relays traffic to a remote application socket.

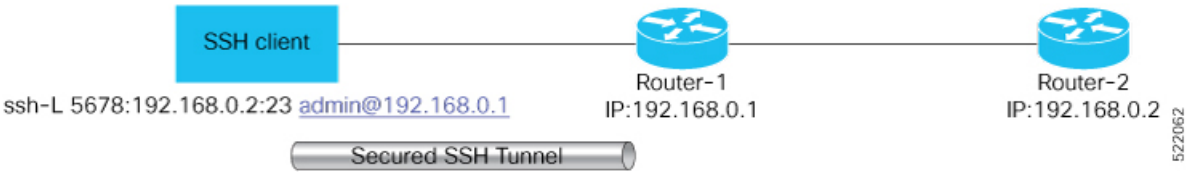
The sample topology forwards local port 5678 through Router-1 to port 23 on Router-2.

Summary

The client command uses **ssh -L local-port:remote-server-hostname:remote-port username@sshserver-hostname**. The local port belongs to the client host; the remote host and port identify the application server; the SSH server hostname identifies the router that receives the tunnel request.

Workflow

Figure 17: Sample topology for SSH port forwarding



These stages describe local SSH port forwarding:

1. The client requests the tunnel using the command

   ```
   ssh -L local-port:remote-server-hostname:remote-port username@sshserver-hostname
   ```

   .
2. The SSH server on Router-1 accepts the TCP/IP packet and opens a socket to Router-2 on port 23.
3. After the SSH connection is established, the server connects the forwarded channel to the new socket.
4. The server forwards incoming client data on that channel to the remote socket.
5. When the client closes the connection, the server closes the socket and forwarded channel.

Result

The local application communicates with the remote application server through an encrypted, port-forwarded local connection.

SSH port forwarding restrictions

Describes the endpoint-role, VRF-reachability, port-mapping, server-configuration, connectivity, and packet-format restrictions that apply when using SSH port forwarding on Cisco IOS XR routers.

Review these restrictions before enabling SSH port forwarding.

SSH port forwarding has these restrictions and guidelines:

- Cisco IOS XR software supports SSH port forwarding only when the router functions as an SSH server, not when it functions as an SSH client. The SSH client on the end host must support SSH port forwarding or tunneling.
- The application server must be reachable in the same VRF in which the current SSH connection between the server and client is established.
- The port numbers do not need to match. You can map any port on the SSH server to any port on the client.
- If an SSH client requests port forwarding when the feature is not enabled on the SSH server, port forwarding fails and the router displays an error message on the console.
- The port-forwarded channel closes if a connectivity problem occurs or if the SSH server receives an improperly formatted SSH packet from the client.

Enable SSH port forwarding

Enables local SSH port forwarding on a Cisco IOS XR server and verifies the forwarded application session, running configuration, and server state.

Allow a compatible end-host SSH client to tunnel application traffic through the router.

Before you begin

Ensure that the remote application server is reachable in the VRF used by the SSH connection.

Procedure

1. Enable local port forwarding and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server port-forwarding local
   Router(config)# commit
   ```

2. Verify the running configuration.

   Example

   ```
   Router# show running-configuration
   ssh server port-forwarding local
   ```

3. Verify the forwarded session and server state.

   Example

   ```
   Router# show ssh
   connection type
   port-forwarded-local
   ```

```
Router# show ssh server
Port Forwarding := local
```

A successful connection also generates a **SECURITY-SSHD-6-PORT_FWD_INFO_GENERAL** Syslog message. A request made while forwarding is disabled generates **SECURITY-SSHD-3-PORT_FWD_ERR_GENERAL** Syslog message.

The server accepts supported local port-forwarding requests and reports them as port-forwarded local sessions.

DSCP marking for SSH packets

Explains when CiscoSSH marks client and server packets and how connection-phase marking prevents initial SSH handshake packets from being filtered.

DSCP marking for SSH packets is a traffic-classification behavior that

- marks CiscoSSH client packets during the TCP connection phase
- can mark server packets before authentication when explicitly enabled, and
- helps prevent transit policies from dropping unmarked handshake packets.

OpenSSH 8.0 marks packets only after authentication. OpenSSH 8.5 and later mark client packets during TCP connection establishment; Cisco IOS XR Software Release 24.1.1 brings this behavior to CiscoSSH. Server packets still require **ssh server set-dscp-connection-phase** for early marking.

Table 82: Feature History Table

Feature Name	Release Information	Feature Description
DSCP Marking from TCP Connection Phase for SSH Packets	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
DSCP Marking from TCP Connection Phase for SSH Packets	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
DSCP Marking from TCP Connection Phase for SSH Packets	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
DSCP Marking from TCP Connection Phase for SSH Packets	Release 24.1.1	We now prevent SSH client packet drops in the TCP connection (initial handshake) phase as they travel across transit routers in the network. This is because you can mark the DSCP values for SSH client packets in the TCP connection phase, which overrides the transit routers' policies to filter and drop packets with no DSCP value marked. Using a new command, you can also set the DSCP value from the TCP connection phase for SSH server packets. The feature introduces these changes: CLI: • ssh server set-dscp-connection-phase YANG Data Model: • New XPath, `set-dscp-connection-phase`, for `Cisco-IOS-XR-crypto-ssh-cfg.yang` (see GitHub, YANG Data Models Navigator)

Set DSCP marking during SSH connection establishment

Enables DSCP marking for CiscoSSH server packets during the initial TCP connection phase and verifies the resulting running SSH server configuration.

Mark server handshake packets before SSH authentication so transit policies do not treat them as unmarked traffic.

The command is available with CiscoSSH and Cisco IOS XR SSH, but the configuration is relevant only to CiscoSSH.

Procedure

1. Enable connection-phase DSCP marking and commit the SSH server configuration.

Example

```
Router# configure
Router(config)# ssh server set-dscp-connection-phase
Router(config-ssh)# commit
```

2. Verify the running SSH configuration.

Example

```
Router# show run ssh
ssh server set-dscp-connection-phase
```

CiscoSSH marks server packets with the configured DSCP value during TCP connection establishment.

SSH server timeouts

Explains how separate connection and channel timers release inactive SSH resources on Cisco IOS XR routers while preserving active connections and independent management operations.

SSH server timeouts are resource-management controls that

- terminate connections that have no active channels
- close individual channels that stop sending or receiving data, and
- prevent inactive SSH state from consuming session resources indefinitely.

A connection is the encrypted client-server session that supports authentication and data exchange. A channel is an independent virtual path inside that connection for an operation such as a shell, SFTP, or NETCONF session.

Timeout comparison
Table 83: SSH timeout behavior

Timeout	Timer starts	Result
Connection	All channels close and no new channel opens.	The router terminates the complete SSH connection.
Channel	A channel has no data activity for the configured period.	The router closes the idle channel while retaining the parent connection.

Unused SSH connection timeouts

Explains how a Cisco IOS XR router detects and disconnects stale SSH connections that remain open without active channels or newly created channels.

An unused SSH connection timeout is a server setting that

- starts when all channels in a connection are closed
- terminates the connection if no new channel opens before expiry, and
- prevents stale automation connections from exhausting session resources.

SSH connections persist after authentication and can host multiple channels for management or file-transfer operations. The connection timer does not run while an active channel remains open.

Table 84: Feature History Table

Feature Name	Release Information	Feature Description
Unused connection timeout for SSH sessions	Release 25.3.1	You can prevent session limit exhaustion and maintain optimal system performance by automatically disconnecting SSH connections with no active channels. The feature introduces a configurable timeout for unused SSH connections, ensuring stale sessions do not occupy resources on your routers. The router monitors each SSH connection and terminates it when all channels remain closed and SSH clients do not create new channels within the configured timeout period. The feature introduces these changes: CLI: • ssh server timeout YANG Data Models: • `Cisco-IOS-XR-crypto-ssh-cfg.yang` data model was modified. • `Cisco-IOS-XR-um-ssh-cfg.yang` data model was modified. (see GitHub)

Set the unused SSH connection timeout

Configures an unused-connection timeout and verifies through session output and system logs that a Cisco IOS XR router terminates SSH connections with no active channels.

Disconnect stale SSH connections automatically to protect session capacity and system resources.

Procedure

1. Set the unused-connection timeout to 600 seconds and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# ssh server timeout connection 600
   Router(config)# commit
   ```

   The timer begins when all channels in the connection close and no new channel opens.

2. Verify the configured connection timeout.

   Example

   ```
   Router# show run ssh
   ssh server timeout
    connection 600
   Router# show ssh server
   Connection Timeout := 600
   ```

3. Monitor the SSH session before and after timeout expiry.

   Example

   Before timeout expiry:

```
Router# show ssh
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host
  ver authentication connection type
-------------------------------------------------------------------------------
Incoming sessions
2         0   vty0    0/RP0/CPU0      SESSION_OPEN    cisco     198.51.100.1
  v2  password       Command-Line-Interface

Outgoing sessions
```

   After timeout expiry:

```
Router# show ssh
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host
  ver authentication connection type
-------------------------------------------------------------------------------
Incoming sessions
2         0   XXXXX   0/RP0/CPU0      SESSION_OPEN    cisco     198.51.100.1
  v2  password

Outgoing sessions
```

   When the connection timeout expires and entire connection closes:

```
Router# show ssh
SSH version : Cisco-2.0

id       chan pty     location        state           userid    host
          ver authentication connection type
-------------------------------------------------------------------------------
Incoming sessions

Outgoing sessions
```

   While an active channel exists, the connection type identifies that channel and the connection timer does not trigger. When no channel is active, the connection type becomes blank. After expiry, the connection no longer appears under incoming sessions.

4. Check the console log for termination of the inactive connection.

   Example

```
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[42522]: terminating inactive
 connection from user sample-user 198.51.100.1 port 25372
```

The router automatically disconnects unused SSH connections after the configured timeout period.

SSH channel timeouts

Explains how a Cisco IOS XR router closes inactive shell, SFTP, NETCONF, and other SSH channels after a configured period while their parent connection remains active.

An SSH channel timeout is a server setting that

- monitors each channel for data inactivity
- closes a channel when its configured idle period expires, and
- retains the parent SSH connection for other active or future channels.

Channels are independent virtual paths within one SSH connection and support concurrent operations such as shell access and file transfer. The timeout applies to all channel types, including Shell, SFTP, and NETCONF.

Table 85: Feature History Table

Feature Name	Release Information	Feature Description
Channel timeout for SSH sessions	Release 25.3.1	You can improve resource efficiency and minimize potential security risks by automatically closing idle SSH channels on the routers after a specific period of inactivity. The feature introduces a configurable timeout for SSH channels which ensures that unused channels do not persist while the parent SSH connection remains active. The router monitors each SSH channel and closes any channel where no data is sent or received within the configured timeout period. The feature introduces these changes: CLI: • ssh server timeout YANG Data Models: • `Cisco-IOS-XR-crypto-ssh-cfg.yang` data model was modified. • `Cisco-IOS-XR-um-ssh-cfg.yang` data model was modified. (see GitHub)

Set the SSH channel timeout

Configures an SSH channel timeout and verifies that a Cisco IOS XR router closes idle channels while retaining their parent connection.

Close inactive SSH channels automatically to improve resource efficiency and reduce security exposure.

Procedure

1. Set the SSH channel timeout to 300 seconds and commit the configuration.

Example

```
Router# configure
Router(config)# ssh server timeout channel 300
Router(config)# commit
```

The timer expires when a channel remains idle for the configured period.

2. Verify the configured channel timeout.

Example

```
Router# show run ssh
ssh server timeout
 channel 300
 connection 600
Router# show ssh server
```

3. Compare the SSH session before and after channel timeout expiry.

Example

Before timer expiry:

```
Router#show ssh

SSH version : Cisco-2.0

id       chan pty     location        state           userid    host
     ver authentication connection type
-------------------------------------------------------------------------------
Incoming sessions
8         0   vty0    0/RP0/CPU0      SESSION_OPEN    user1     198.51.100.1
     v2  password       Command-Line-Interface

Outgoing sessions
Router#
```

After timer expiry:

```
Router#show ssh

SSH version : Cisco-2.0

id       chan pty     location        state           userid    host
     ver authentication connection type
-------------------------------------------------------------------------------
Incoming sessions
8         0   XXXXX   0/RP0/CPU0      SESSION_OPEN    user1     198.51.100.1
     v2  password

Outgoing sessions
Router#
```

Before timer expiry, the output identifies the active pseudoterminal and connection type. After expiry, the **pty** field displays **XXXXX** and the connection type is blank.

4. Check the console log for closure of the idle channel.

Example

```
%SECURITY-SSHD_SYSLOG_PRX-6-INFO_GENERAL : sshd[39648]: Closing channel 0 of
 user sample-user 198.51.100.1 port 34454 after 300 seconds of inactivity
```

The router automatically closes inactive SSH channels after the configured timeout while the parent connection remains available.

20 FIPS Mode and Cryptographic Services

Topics:

- FIPS mode
- FIPS-compliant cryptographic keys
- Cryptographic services in FIPS mode
- SSH clients and servers in FIPS mode

Provides FIPS mode requirements and procedures for compliant cryptographic keys, key chains, certificates, OSPFv3, SNMPv3, and SSH services on Cisco IOS XR routers.

Use this chapter to prepare a Cisco IOS XR router for FIPS mode and configure supported cryptographic services.

The chapter covers these functional areas:

- FIPS mode concepts, requirements, restrictions, enablement, and system reload
- Cryptographic keys, key chains, and certificates
- OSPFv3, SNMPv3, and SSH configurations that use approved algorithms

Review the requirements before enabling FIPS mode. Then configure the cryptographic material and services that the router requires.

FIPS mode

Explains the FIPS 140-2 cryptographic-module standard, the Cisco Common Cryptographic Module, and the Cisco IOS XR applications verified for FIPS compliance.

Federal Information Processing Standard (FIPS) 140-2 is a U.S. and Canadian government certification standard that

- defines requirements for cryptographic modules
- specifies practices for algorithms, key material, data buffers, and operating-system interaction, and
- supports level 1 compliance in Cisco IOS XR software through the Cisco Common Cryptographic Module (C3M).

C3M provides FIPS-validated cryptographic primitives and functions that applications and protocols can use.

Applications verified for FIPS compliance

Cisco IOS XR software verifies these applications for FIPS compliance:

- Secure Shell (SSH)
- Secure Sockets Layer (SSL)
- Transport Layer Security (TLS)
- Internet Protocol Security (IPsec) for Open Shortest Path First version 3 (OSPFv3)
- Simple Network Management Protocol version 3 (SNMPv3)
- AAA password security

C3M also provides cryptographic services for protocols and applications such as RTP and 802.1X.

Guidelines and restrictions for FIPS mode

Outlines access, algorithm, key-string, Telnet, commit-order, SSH-session, and reload conditions that apply when enabling or disabling FIPS mode on Cisco IOS XR routers.

Requirements

Satisfy these requirements before you enable FIPS mode:

- Use a user group whose associated task group includes the task IDs required by each command. Contact the AAA administrator if a user-group assignment prevents command access.
- Configure sessions with FIPS-approved cryptographic algorithms.

  OSPF, BGP, RSVP, and IS-IS sessions do not operate in FIPS mode when they use MD5 or HMAC-MD5.

  The same restriction applies to key-chain applications and other applications that use nonapproved algorithms.
- Use a key string with at least 14 characters for sessions that use an HMAC-SHA algorithm. A session with a shorter key string goes down in FIPS mode.
- Stop new incoming SSH sessions while configuring or removing FIPS mode, and reload the router after the configuration change.

These requirements apply when you configure, remove, or operate Cisco IOS XR software in FIPS mode.

FIPS mode rejects nonapproved cryptographic operations and requires a reload to finalize a mode change.

Restrictions

Do not use these algorithms in a process that must remain FIPS compliant:

- Rivest Cipher 4 (RC4)
- Message Digest 5 (MD5)
- Keyed-Hash Message Authentication Code MD5 (HMAC-MD5)
- Data Encryption Standard (DES)

Telnet behavior

FIPS mode rejects new Telnet configuration and Telnet connections.

- If FIPS mode is already enabled, the system rejects an attempted Telnet configuration.
- If a Telnet configuration exists before FIPS mode is enabled, the system retains the configuration but rejects Telnet connections.

Recommendation: Enable FIPS mode in a separate commit

Configure **crypto fips-mode** first, and commit dependent FIPS configurations separately.

FIPS mode rejects configurations such as these examples:

- **key chain** *sample-keychain* **key 1 cryptographic-algorithm MD5**
- **key chain** *sample-keychain* **key 1 cryptographic-algorithm HMAC-MD5**
- **router ospfv3 1 authentication ipsec spi 256 md5** *sample-md5-value*
- **router ospfv3 1 encryption ipsec spi 256 esp des** *sample-des-value*
- **router ospfv3 1 encryption ipsec spi 256 esp des** *sample-des-value* **authentication md5** *sample-md5-value*
- **snmp-server user** *sample-user sample-group* **v3 auth md5 priv des56**
- **ssh server algorithms key-exchange diffie-hellman-group1-sha1**
- **telnet vrf default ipv4 server max-servers 100**

Enable FIPS mode

Configures FIPS mode, verifies the configuration-change message in the system log, and reloads all locations to finalize the mode change.

Place the router in FIPS mode so that its cryptographic services enforce approved configurations.

Enabling or removing FIPS mode changes system-wide cryptographic behavior and requires a router reload.

Before you begin

- Satisfy the access, algorithm, key-string, and Telnet requirements for FIPS mode.
- Stop new incoming SSH sessions.

Procedure

1. Enable FIPS mode in global configuration mode, and commit the change.

Example

```
RP/0/RP0/CPU0:router# configure
Router(config)# crypto fips-mode
Router(config)# commit
```

2. Verify that the system log reports the FIPS mode change.

Example

```
Router# show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
     Console logging: level debugging, 60 messages logged
     Monitor logging: level debugging, 0 messages logged
     Trap logging: level informational, 0 messages logged
     Buffer logging: level debugging, 3 messages logged

Log Buffer (9000000 bytes):
<output omitted>
The configuration setting for FIPS mode has been modified. The system must
be reloaded to finalize this configuration change.
```

Use **show logging | i fips** to filter FIPS-specific logging messages.

3. Reload all locations.

Example

```
Router# reload location all
```

The router reloads all nodes on the single-chassis or multishelf system.

The router starts in FIPS mode and enforces the applicable cryptographic requirements.

FIPS-compliant cryptographic keys

Explains when cryptographic key pairs require manual generation, how key purpose affects RSA generation, and which generated keys appear in the running configuration.

A FIPS-compliant cryptographic key pair is a collection of key material that

- uses a key type and size permitted by the documented FIPS mode
- supports signing, encryption, or both functions, and
- can be inspected or removed with the corresponding key-management commands.

The router automatically generates cryptographic keys when it boots. Generate keys manually only when required keys are missing.

Key generation in configuration mode

Cisco IOS XR Release 7.3.2 and later supports key-pair generation and deletion in XR configuration mode.

Only keys generated in configuration mode appear in the running configuration.

Supported cryptographic key types and sizes in FIPS mode

Lists the RSA, DSA, and ECDSA key sizes that the source identifies as permitted in FIPS mode and the commands used to remove each key type.

Use this information to select a documented key type and size before generating a key pair.

Table 86: Cryptographic key types and sizes

Key type	Documented FIPS-mode sizes	Key removal command
RSA	2048, 3072, or 4096 bits	**crypto key zeroize rsa** *keypair-label*
DSA	2048 bits	**crypto key zeroize dsa** *keypair-label*
ECDSA	nistp256, nistp384, or nistp512	**crypto key zeroize ecdsa** *keypair-label*

Generate and verify FIPS-compliant cryptographic keys

Configures RSA, DSA, or ECDSA key pairs with documented FIPS-mode sizes and verifies the resulting public keys on the router.

Create missing cryptographic keys for applications that operate in FIPS mode.

The router generates cryptographic keys during startup. Use this task when an application requires a key that is missing.

Before you begin

- Enable FIPS mode.
- Generate keys manually only when the required keys were not generated during router startup.
- Select a supported key type and size.

Procedure

1. Generate an RSA key pair when the application requires RSA.

   Example

   ```
   Router# crypto key generate rsa general-keys sample-rsa-keypair
   ```

   Use a key size of 2048, 3072, or 4096 bits. The **usage-keys** option generates separate signing and encryption keys; **general-keys** generates one key pair for both purposes.

   To delete the RSA key pair, use the **crypto key zeroize rsa** *keypair-label* command.

2. Generate a DSA key pair when the application requires DSA.

   Example

   ```
   Router# crypto key generate dsa
   ```

   The permitted DSA size in FIPS mode is 2048 bits.

   To delete the DSA key pair, use the **crypto key zeroize dsa** *keypair-label* command.

3. Generate an ECDSA key pair when the application requires ECDSA.

Example

```
Router# crypto key generate ecdsa
```

The ECDSA key sizes allowed under FIPS mode are **nistp256**, **nistp384**, and **nistp512**.

To delete the ECDSA key pair, use the **crypto key zeroize ecdsa** *keypair-label* command.

4. Display the generated RSA public keys.

Example

```
Router# show crypto key mypubkey rsa
Key label: system-root-key
Type : RSA General purpose
Size : 2048
Created : 01:13:10 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 <public-key output omitted>
Key label: system-enroll-key
Type : RSA General purpose
Size : 2048
Created : 01:13:16 IST Thu Feb 06 2020
Data :
 30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
 <public-key output omitted>
```

5. Display the generated DSA public keys.

Example

```
Router# show crypto key mypubkey dsa
```

6. Optionally verify keys that were generated in the configuration mode in the running configuration.

Example

```
Router# configure
Router(config)# crypto key generate rsa sample-rsa-key general-keys 2048
Router(config)# commit
Router(config)# end
Router# show running-config
!! IOS XR Configuration 7.3.4
!
username sample-user
 group root-lr
 group cisco-support
 secret 10
 <encrypted-secret omitted>
!
crypto key generate rsa sample-rsa-key general-keys 2048 | -----BEGIN PUBLIC
 KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAgiXFnld/AADcil6eV38A
<public-key output omitted>
-----END PUBLIC KEY-----
 |
end
```

Only keys generated in configuration mode appear in the running configuration. The displayed keys use OpenSSL format.

Cryptographic services in FIPS mode

Explains how key chains, certificates, OSPFv3, and SNMPv3 use documented cryptographic algorithms and key material after FIPS mode is enabled.

Cryptographic services in FIPS mode are router functions that

- use documented cryptographic algorithms and key material
- protect authentication, certificate, routing, and management operations, and
- must meet the applicable FIPS mode requirements before they can operate successfully.

Enable FIPS mode before configuring the service-specific key chain, certificate, OSPFv3, or SNMPv3 settings.

Uses of cryptographic services

Table 87: Service-specific cryptographic uses

Service or component	Cryptographic use
Key chain	Authenticates application sessions with a configured hash or HMAC algorithm.
Certificate	Associates a certificate trustpoint with cryptographic keys and the required signature-hash or encryption type.
OSPFv3	Uses IPsec authentication and encryption to protect routing packets.
SNMPv3	Uses authentication and privacy algorithms to protect management access.

Configure a FIPS-compliant key chain

Configures a key chain and selects SHA-1 or HMAC-SHA1-20 as the documented cryptographic algorithm for operation in FIPS mode on Cisco IOS XR routers.

Create a key chain that uses a documented FIPS-mode algorithm.

Applications such as routing protocols can use key chains to authenticate protocol sessions.

Before you begin

Enable FIPS mode and ensure that an HMAC-SHA key string contains at least 14 characters.

Procedure

Create the key chain and key in global configuration mode and select its cryptographic algorithm.

Example

```
Router# configure
Router(config)# key chain sample-keychain
Router(config-sample-keychain)# key 1
```

```
Router(config-sample-keychain-1)# cryptographic-algorithm HMAC-SHA1-20
Router(config-sample-keychain-1)# commit
```

You may choose from these cryptographic algorithms: **HMAC-SHA1-20** or **SHA-1**.

The key chain uses the selected documented FIPS-mode algorithm.

Configure FIPS-compliant certificates

Configures a trustpoint with an RSA key, commits the certificate configuration, and displays the resulting certificate information for validation on the router.

Configures a trustpoint with an RSA key and verifies the certificate information on the router.

The trustpoint associates certificate authority information with the cryptographic key used by the router.

Before you begin

- Enable FIPS mode.
- Generate the required RSA key pair.
- Ensure that certificates meet the FIPS requirements for key length, signature hash, and encryption type.
- Use a minimum RSA or DSA key length of 1024 bits and the SHA-1-20 hash algorithm.

Procedure

1. Associate the trustpoint with the RSA key in global configuration mode, and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# crypto ca trustpoint sample-trustpoint sample-rsa-keypair
   Router(config)# commit
   ```

2. Display the certificate information.

   Example

   ```
   Router# show crypto ca certificates
   ```

Configure FIPS-compliant OSPFv3

Configures OSPFv3 authentication and IPsec encryption with the algorithms identified by the source for operation in FIPS mode on Cisco IOS XR routers.

Protect OSPFv3 packets with documented authentication and encryption algorithms.

Before you begin

Enable FIPS mode.

Procedure

Configure the OSPFv3 process, area authentication, and process encryption in global configuration mode.

Example

```
Router# configure
Router(config)# router ospfv3 sample-ospfv3
Router(config-ospfv3)# area 1
Router(config-ospfv3-ar)# authentication ipsec spi 256 sha1 password
sample-auth-password
Router(config-ospfv3-ar)# exit
Router(config-ospfv3)# encryption ipsec spi 256 esp 3des password
sample-encryption-password
Router(config-ospfv3)# commit
```

- The area ID can be a decimal value or an IP address.
- OSPFv3 supports only SHA-1 for authentication.
- IPsec support applies only to OSPFv3.
- For encryption, use 3DES or AES with an optional 192-bit or 256-bit key. Select SHA-1 when you enable authentication.

The OSPFv3 process protects routing packets with the configured authentication and encryption settings.

Configure a FIPS-compliant SNMPv3 server

Configures an SNMPv3 user with SHA authentication and AES privacy settings for operation while the router is in FIPS mode.

Configure secure SNMPv3 authentication and privacy for a server user.

SNMPv3 separates user authentication from privacy encryption in the server-user configuration.

Before you begin

Enable FIPS mode.

Procedure

Configure the SNMPv3 user in global configuration mode, and commit the configuration.

Example

```
Router# configure
Router(config)# snmp-server user sample-snmp-user sample-snmp-group v3 auth
sha clear sample-auth-password priv aes 128 clear sample-privacy-password
Router(config)# commit
```

The command supports SHA authentication and 3DES or AES privacy. AES privacy supports 128-bit, 192-bit, or 256-bit keys.

The SNMPv3 server user uses the configured authentication and privacy protection.

SSH clients and servers in FIPS mode

Explains how SSH clients select documented FIPS-approved ciphers and how SSH version 2 servers use supported key-exchange, cipher, and HMAC algorithms.

SSH clients and servers in FIPS mode are secure connection endpoints that

- use documented algorithms during session establishment
- negotiate compatible cipher and integrity protection, and
- operate after FIPS mode and the required SSH settings are enabled.

The SSH client selects an AES-CTR cipher when it starts a connection. The SSH version 2 server negotiates supported key-exchange, cipher, and HMAC algorithms with the client.

Connect to an SSH server with FIPS-approved ciphers

Configures an SSH client session with an AES-CTR cipher and a 128-bit, 192-bit, or 256-bit key length approved by the documented FIPS mode.

Connect the router SSH client to a server with a documented FIPS-approved cipher.

The SSH client selects the cipher and key length when it starts the connection.

Before you begin

Enable FIPS mode.

Procedure

Start the SSH session with the destination address, AES-CTR key length, and username.

Example

```
Router# ssh 192.0.2.1 cipher aes 128-CTR username sample-ssh-user
```

AES in Counter mode with a 128-bit, 192-bit, or 256-bit key is the FIPS-compliant cipher for the SSH client.

The router starts an SSH client session with the selected AES-CTR cipher.

Configure a FIPS-compliant SSH server

Configures an SSH version 2 server while the router operates in FIPS mode and identifies the supported server algorithm reference for connection planning.

Enable inbound SSH version 2 connections while the router operates in FIPS mode.

The server negotiates connection algorithms from the set supported in FIPS mode.

Before you begin

- Enable FIPS mode.
- Review the supported SSH server algorithms for FIPS mode.

Procedure

Enable the SSH version 2 server in global configuration mode, and commit the configuration.

Example

```
Router# configure
Router(config)# ssh server v2
Router(config)# commit
```

Use these values when evaluating an SSH server configuration for FIPS mode.

Table 88: Supported SSH server algorithms

Algorithm category	Supported algorithms
Key exchange	• diffie-hellman-group14-sha1 • ecdh-sha2-nistp256 • ecdh-sha2-nistp384 • ecdh-sha2-nistp521
Cipher	• aes128-ctr • aes192-ctr • aes256-ctr • aes128-gcm • aes256-gcm
HMAC	• hmac-sha2-512 • hmac-sha2-256 • hmac-sha1

The SSH version 2 server negotiates a mutually supported key-exchange algorithm, cipher, and HMAC algorithm with each connecting client.

21 Implementing Secure Logging

Topics:

- System logging over Transport Layer Security (TLS)
- Security template framework for TLS applications
- TLS RFC 5289 compliance for security template

Explains how to send system log messages securely over Transport Layer Security (TLS), configure TLS security templates, and apply RFC 5289-compliant cryptographic controls on Cisco 8000 Series Routers.

Use this chapter to implement secure logging over TLS and to centralize TLS and certificate policies for supported applications.

The chapter covers these areas:

- Secure logging over TLS, including handshake behavior, restrictions, configuration, and verification
- Security templates for reusable certificate, TLS, and compliance policies
- RFC 5289 compliance for security templates

System logging over Transport Layer Security (TLS)

Describes how Cisco 8000 Series Routers use Transport Layer Security (TLS) to send system log messages to a remote syslog server over a trusted channel.

Secure logging over Transport Layer Security (TLS) is a method that

- sends system log messages from the router to a remote syslog server through an encrypted and authenticated channel
- preserves logs outside the router because the router's local logging buffer is limited and does not retain logs across reboots, and
- replaces unsecured User Datagram Protocol (UDP) transport with a channel that authenticates the server and client, encrypts transferred syslog data, and verifies data integrity.

Secure logging components

The router acts as the TLS client, and the remote syslog server acts as the TLS server. TLS runs over Transmission Control Protocol (TCP), so the router completes the TCP handshake before it begins the TLS handshake.

Secure logging uses RFC 5425, Transport Layer Security Transport Mapping for Syslog.

How the TLS handshake establishes secure logging

Shows how the router and remote syslog server use TCP and Transport Layer Security (TLS) handshakes to establish an encrypted logging session.

The router must complete a TCP handshake with the remote syslog server before it can establish the TLS session.

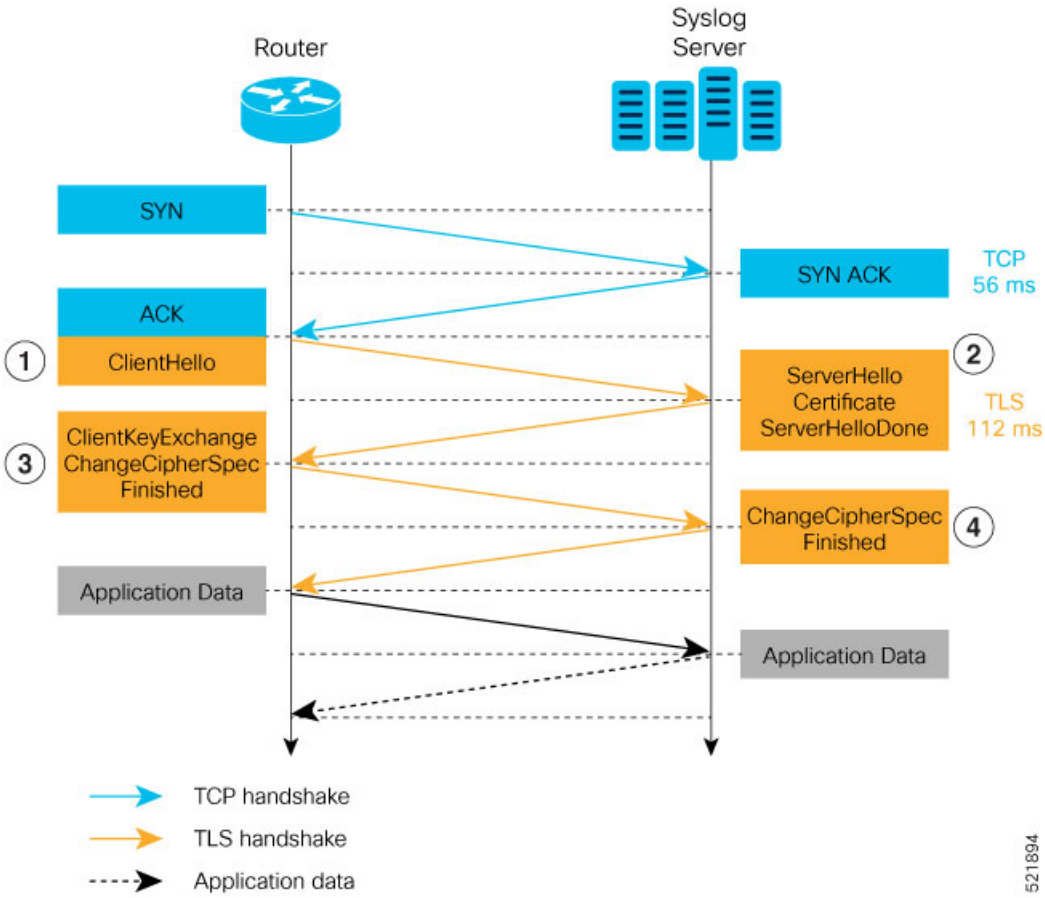
Summary

The secure-logging handshake involves these participants and messages:

- Router: Acts as the TLS client and starts the handshake.
- Syslog server: Acts as the TLS server and provides its certificate.
- Certification authority: Validates the server certificate presented to the router.

Workflow

Figure 18: TLS handshake



These stages describe how the TLS session is established after the TCP handshake completes:

1. The router sends a Client Hello message to begin the TLS handshake.
2. The server sends its TLS certificate, which contains its public key, to support secure-connection establishment.
3. The router validates the server certificate with the certification authority and checks the certificate validity. It then sends a Change Cipher Spec message to indicate that subsequent messages use the negotiated key and algorithm.
4. The server decrypts the message with its private key and sends an encrypted Change Cipher Spec message with the session key.
5. Both endpoints complete the TLS handshake and can exchange encrypted application data.

Result

The router and syslog server establish a trusted TLS session for secure syslog transport.

Restrictions for syslogs over TLS

Identifies the specific server-identifier and certificate-name requirements that apply when the router sends syslog messages over Transport Layer Security (TLS).

- When you configure the remote syslog server on the router, specify only one identifier for a remote syslog server: its hostname or its IPv4 or IPv6 address. A single server identifier gives the TLS configuration one unambiguous endpoint identity.

Configure syslogs over TLS

Configures a trustpoint, a remote syslog server, and hostname resolution so a Cisco 8000 Series Router can send syslog messages over Transport Layer Security (TLS).

Configure secure transport for syslog messages sent to a remote server.

You can identify the remote server by its IPv4 or IPv6 address or by its hostname. Configure only one identifier. The logging severity determines which syslog messages the router sends.

Before you begin

Before you begin, obtain the trustpoint and certification authority enrollment information required for the TLS channel.

- Choose the remote server identifier and logging severity.

Procedure

1. Configure the trustpoint for the TLS channel.

   Example

   ```
   Router# conf t
   Router(config)# crypto ca trustpoint tp
   Router(config-trustp)# subject-name CN=new
   Router(config-trustp)# enrollment terminal
   Router(config-trustp)# rsakeypair k1
   Router(config-trustp)# commit
   ```

   You can use **enrollment url SCEP-url** or **enrollment terminal** for certification authority enrollment.

2. Configure the remote syslog server with its IPv4 or IPv6 address.

   Example

   ```
   Router(config)# logging tls-server TEST
   Router(config-logging-tls-peer)# severity debugging
   Router(config-logging-tls-peer)# trustpoint tp
   Router(config-logging-tls-peer)# address ipv4 10.105.230.83
   Router(config-logging-tls-peer)# commit
   ```

   Replace the address with the IPv4 or IPv6 address of your remote syslog server.

3. Configure the remote syslog server with its hostname instead of an address when hostname-based identification is required.

   Example

   ```
   Router(config)# logging tls-server TEST
   Router(config-logging-tls-peer)# severity debugging
   Router(config-logging-tls-peer)# trustpoint tp
   Router(config-logging-tls-peer)# tls-hostname xyz.cisco.com
   Router(config-logging-tls-peer)# commit
   ```

4. Configure DNS to resolve the hostname specified by `tls-hostname` to the remote syslog server.

Example

```
Router(config)# domain ipv4 host xyz.cisco.com 10.105.230.83
Router(config)# domain name cisco.com
Router(config)# commit
```

5. Verify the TLS connection and certificate information.

Example

```
Router# show lpts bindings brief
@ - Indirect binding; Sc - Scope
Location Clnt Sc L3 L4 VRF-ID Interface Local-Address,Port Remote-Address,Port
---------- ---- -- ---- ------ --------- ------------
-------------------------------------
0/RP0/CPU0 TCP LR IPV4 TCP default any 5.10.18.5,35926 10.105.230.83,6514

Router# show logging
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
Console logging: level debugging, 185 messages logged
Monitor logging: level debugging, 94 messages logged
Trap logging: level informational, 0 messages logged
Logging to TLS server 10.105.230.83, 66 message lines logged
Buffer logging: level debugging, 183 messages logged
Log Buffer (2097152 bytes):
...................

Router# show crypto ca certificates
Trustpoint : tp
================================================
CA certificate
Serial Number : B5:68:C8:96:A4:7C:1A:BA
Subject:
CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
Issued By :
CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
Validity Start : 05:39:51 UTC Tue Aug 13 2019
Validity End : 05:39:51 UTC Mon Aug 08 2039
CRL Distribution Point
http://10.105.236.78/crl_xxx/crl.der
SHA1 Fingerprint:
03BD57E04A2AA4648A84F515A46EF99CCF488387
```

- Port 6514 is the default port for syslog over TLS. In the **show lpts bindings brief** output, confirm that port 6514 is associated with the syslog server address.
- Use **show logging** to confirm the TLS server address and sent-message count.
- Use **show crypto ca certificates** to inspect certification authority certificate details.

The router sends syslog messages to the remote server through a configured TLS channel.

What to do next

When the TLS channel comes up, confirm these console messages:

```
RP/0/RP0/CPU0: syslogd[148]: %SECURITY-XR_SSL-6-CERT_VERIFY_INFO : SSL
Certificate verification: Peer certificate verified successfully
```

```
RP/0/RP0/CPU0: syslogd[148]: %OS-SYSLOG-5-LOG_NOTICE : Secure Logging:
Successfully established TLS session , server :10.105.230.83
```

Security template framework for TLS applications

Describes reusable security templates that centralize certificate authentication, Transport Layer Security (TLS) controls, and compliance settings for multiple Cisco IOS XR applications.

A security template is a named configuration bundle for TLS-enabled applications that

- centralizes and standardizes security policy configuration,
- encapsulates certificate authentication policies, TLS protocol controls, and compliance mode settings, and
- provides a reusable source of truth that multiple applications can reference instead of embedding security settings locally.

The framework supports TLS version and cipher-suite control, elliptic-curve and digital-signature selection, certificate onboarding through SCEP, file-based methods, and Certz profiles, and dynamic change notifications for registered applications.

Table 89: Feature History Table

Feature Name	Release Information	Feature Description
Security template framework for TLS enabled applications	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) Security templates reduce misconfiguration risks and operational overhead by centralizing and standardizing security policy configuration for TLS-enabled applications. A security template bundles certificate authentication policy, TLS controls, and compliance mode settings. It acts as a single source of truth that applications reference, avoiding local embedding of security settings. This template defines how certificates are handled and controls various aspects of the TLS handshake. The feature introduces these changes: CLI: • **security-template**

Key terminologies

Term	Description
Certificate authentication policy	Rules that specify how certificates authenticate servers or clients, including trust-anchor selection and certificate validation settings.
Common Criteria (CC) mode	An enhanced security mode that enforces stricter compliance-focused behavior.
Certz profile	A profile that provides identity certificates, private keys, and certification authority bundles from a centralized certificate management service.

Benefits and use cases

The security template framework provides these benefits:

- Centralized security management for templates used by multiple applications
- Flexibility for Common Criteria mode, certificate authentication policies, and future security policies
- Simplified configuration by reducing application-specific settings
- Consistency and compliance through uniform policy application
- Extensibility for additional fields and configurations
- Operational efficiency through automated notifications and faster updates

Use cases include these activities:

- Managing TLS settings for syslog
- Supporting web-scale environments with many root certification authorities and custom certificate validation needs
- Enforcing advanced cryptographic policies for each application, such as application-specific elliptic curves and signature algorithms

Restrictions for security template

Identifies the only application that can use security templates in the current implementation of secure logging over Transport Layer Security.

Use security templates only with syslog messages sent over TLS. Configure exactly one syslog collector identifier: an IPv4 or IPv6 address, or a hostname using **tls-hostname** *hostname* command.

Configuration guidelines for security template framework

Provides configuration guidelines for scalable, consistent, and secure management of TLS and certificate settings through reusable security templates for Cisco IOS XR applications.

Manage TLS policies centrally

Apply these guidelines when you create, assign, and maintain security templates for TLS-enabled applications:

- Centralize security policy configuration for TLS-enabled applications using reusable security templates. This standardizes certificate authentication, TLS controls, and compliance settings across services, reducing operational overhead and misconfiguration risk.
- Use a unique name for each security template and make it the application's single source of truth instead of embedding security settings locally.
- Enable Common Criteria (CC) mode when enhanced compliance is required.
- Configure certificate authentication policies for identity certificates and peer trust anchors. Prefer a Certz profile when available because it takes precedence over individual trustpoints.
- When both standalone trustpoint and security template are configured under a syslog server, use the trustpoint specified in the security template because it overrides the standalone trustpoint.
- Register applications to receive and handle security-template update notifications so that changes apply dynamically and consistently.

How the security template framework works

Describes how the security template infrastructure provisions reusable TLS policies, applies them to registered applications, and propagates policy changes for Cisco IOS XR applications.

As organizations deploy more TLS-enabled applications, centralized management helps control certificate authorities, protocol versions, cipher suites, and compliance requirements.

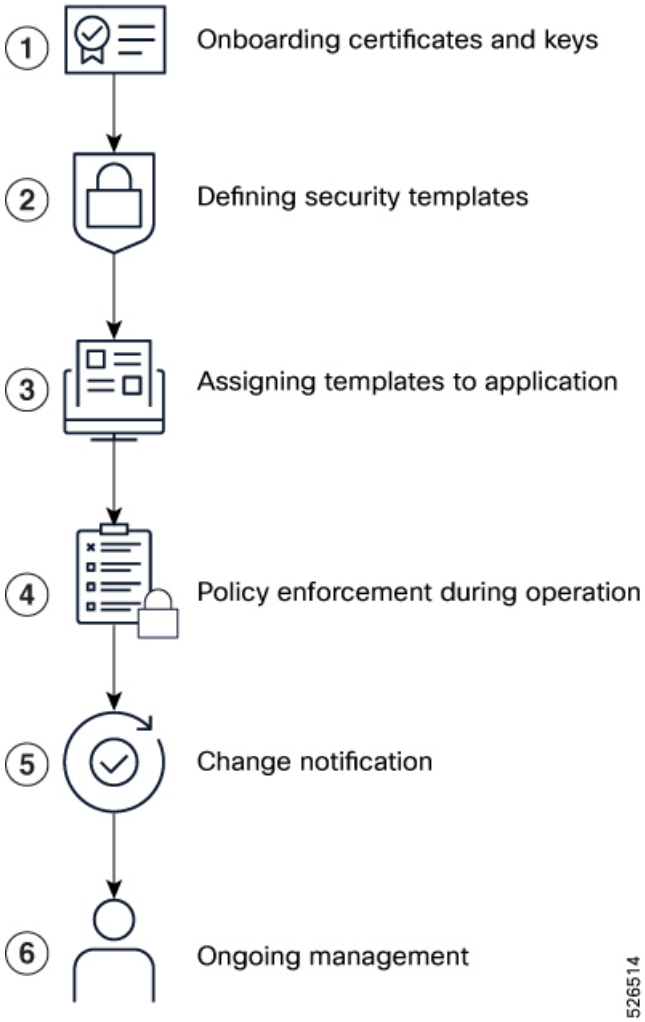
Summary

The framework coordinates these components:

- Security template infrastructure: Stores and manages templates and their application to Cisco IOS XR services.
- Applications such as syslog: Reference and enforce the settings in named security templates.
- Trustpoints and Certz profiles: Provide certificate and key material for authentication and trust validation.
- Notification and change management: Updates registered applications when templates or certificate material changes.

Workflow

Figure 19: Security template process



These stages describe how applications use the security template framework:

1. Administrators onboard certificates and keys by configuring trustpoints with methods such as SCEP or file import. They can also configure Certz profiles for certificate lifecycle management.

2. An administrator defines a named security template through the command-line interface, specifying TLS versions, cipher suites, certificate authentication policies, trust anchors, identity sources, and compliance modes.
3. An administrator assigns the template to applications so that the applications reference the template instead of embedding their own security settings.
4. Each application registers with the security template infrastructure, provides a callback for configuration-change notifications, and initializes its Secure Sockets Layer (SSL) or TLS context through library APIs.
5. During operation, applications use the template parameters for certificate validation, TLS negotiation, and compliance enforcement.
6. When a template or associated certificate material changes, the infrastructure notifies registered applications. Applications can reinitialize sessions or contexts to apply the new configuration.
7. Administrators update templates, rotate certificates, and audit compliance centrally while changes propagate to consuming applications.

Result

The framework provides centralized, consistent, and scalable security-policy management across Cisco IOS XR applications, reducing operational overhead and misconfiguration risk while supporting secure, compliant communications.

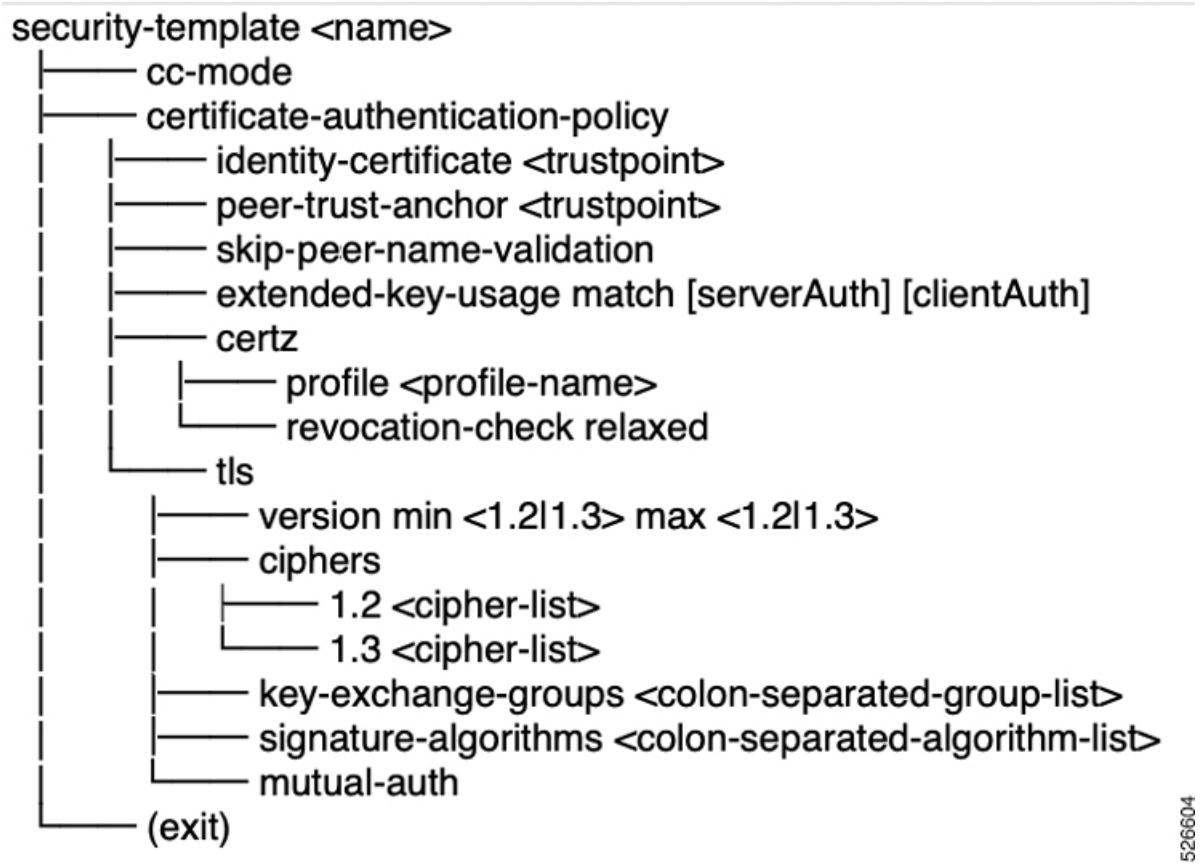
Configure a security template

Configures a reusable security template with certificate authentication, Transport Layer Security (TLS), and compliance settings, then applies it to a supported application.

Centralize and standardize security policies, certificate authentication, and TLS controls for applications running on Cisco routers.

Understand the security-template mode, certificate authentication policy, TLS settings, and mutual authentication before you configure the template.

Figure 20: Security-template CLI structure



- Security template mode: Defines a template with a unique name.
- Common Criteria (CC) mode: Enables stricter, compliance-focused behavior.
- Identity certificate: Specifies the trustpoint for the leaf certificate and its chain.
- Peer trust anchor: Specifies the trustpoint that validates incoming server or client certificates. It takes precedence over a peer identity when both are configured.
- Skip peer name validation: Disables hostname verification against a certificate's DNS Subject Alternative Name (SAN) or subject Common Name (CN). Use it only in a lab or controlled environment.
- Extended key usage match: Validates the **serverAuth** and **clientAuth** certificate extensions.
- Certz profile: Sources the identity certificate, private key, and certification authority bundle from a gNSI Certz profile. A configured Certz profile takes precedence over trustpoints.
- Certz revocation method relaxed: Prevents authentication failure when the certificate revocation list bundle does not contain a revocation list for a certificate in the verified chain.
- TLS settings: Control the minimum and maximum TLS versions, allowed cipher suites, key-exchange groups, signature algorithms, and mutual TLS authentication.

Before you begin

Before you begin, configure the trustpoints or Certz profiles required for certificate management and confirm that the application supports security-template integration.

Procedure

1. Enter security-template mode by defining a unique template name.

Example

```
Router(config)# security-template template1
```

2. Enable Common Criteria mode when enhanced compliance is required.

   Example

   ```
   Router(config-security-template)# cc-mode
   ```

3. Enter certificate-authentication-policy mode and configure certificate settings.

   Example

   ```
   Router(config-security-template)# certificate-authentication-policy
   Router(config-certificate-authentication-polic)# peer-trust-anchor trustpoint1
   Router(config-certificate-authentication-polic)# identity-certificate CA2
   Router(config-certificate-authentication-polic)# extended-key-usage match
   serverAuth
   Router(config-certificate-authentication-polic)# skip-peer-name-validation
   Router(config-certificate-authentication-polic)# certz-profile certz1
   Router(config-certificate-authentication-polic)# commit
   ```

   Configure the identity certificate, peer trust anchor, and optional peer-name validation, extended-key-usage, and Certz settings as required.

4. Enter TLS configuration mode and set protocol and cryptographic parameters.

   Example

   ```
   Router(config-certificate-authentication-polic)# tls
   Router(config-tls)# ciphers 1.2 AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA25
    1.3 TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256
   Router(config-tls)# version min 1.2 max 1.3
   Router(config-tls)# key-exchange-groups P-256:P-384:X25519
   Router(config-tls)# signature-algorithms
   rsa_pss_rsae_sha256:rsa_pss_rsae_sha384:rsa_pss_rsae_sha512:rsa_pkcs1_sha256:rsa_pkcs1_sha384:rsa_pkcs1_sha512
   Router(config-tls)# commit
   ```

   Set the minimum and maximum TLS versions, allowed cipher suites, key-exchange groups, signature algorithms, and optional mutual authentication.

5. Apply the security template to a supported application.

   Example

   ```
   Router(config)# logging tls-server TEST
   Router(config-logging-tls-peer)# severity debugging
   Router(config-logging-tls-peer)# trustpoint tp
   Router(config-logging-tls-peer)# address ipv4 10.105.230.83
   Router(config-logging-tls-peer)# security-template template1
   ```

   When a standalone trustpoint and a security template are both configured under a syslog server, the trustpoint in the security template overrides the standalone trustpoint.

6. Register the application with the security template infrastructure.

Register the application to receive change notifications and apply updates dynamically. The security template then enforces unified security policies and TLS parameters across referenced applications.

The application uses the configured security template for consistent certificate, TLS, and compliance behavior.

What to do next

After configuration, monitor application logs and debug output. Update the security template when you rotate certificates or change policies so registered applications can adapt dynamically.

TLS RFC 5289 compliance for security template

Describes how security-template compliance with RFC 5289 supports Common Criteria mode, newer cipher suites, and stronger Elliptic Curve Cryptography (ECC) algorithms.

TLS RFC 5289 compliance for a security template is a security feature that

- supports Common Criteria (CC) mode
- specifies new cipher suites, and
- provides stronger Elliptic Curve Cryptography (ECC) algorithms.

Table 90: Feature History Table

Feature Name	Release Information	Feature Description
TLS RFC 5289 compliance for security template framework	Release 25.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) The security template framework is based on RFC 5289, which specifies new cipher suites for the Transport Layer Security (TLS) protocol. This feature supports Common Criteria (CC) mode which is an enhanced security mode that enforces stricter compliance-focused behavior. It enhances TLS security by introducing stronger Elliptic Curve Cryptography (ECC) algorithms.

The security template framework for TLS-enabled applications uses RFC 5289 compliance. RFC 5289 moves away from HMAC-SHA1 and uses SHA-256 and SHA-384 for message authentication codes. For more information about the security template framework, see *Security template framework for TLS applications*.

22 802.1X and MAC-Based Port Authentication

Topics:

- Port-based authentication methods
- 802.1X authentication with MAC Authentication Bypass fallback
- RADIUS Change of Authorization for 802.1X and MAB sessions

Describes how Cisco 8000 Series Routers control network access with 802.1X, MAC Authentication Bypass, fallback authentication, and RADIUS Change of Authorization.

Use this chapter to understand, configure, and verify port-based authentication for authorized and non-802.1X-capable client devices.

The chapter covers the following authentication capabilities:

- 802.1X port-based authentication with remote RADIUS or local EAP authentication
- MAC Authentication Bypass (MAB) for devices that cannot use 802.1X
- MAB fallback and authentication-method precedence
- RADIUS Change of Authorization for active 802.1X and MAB sessions

Use the topics in this chapter to:

- Control ingress traffic on physical ports until clients are authenticated
- Configure client, authenticator, supplicant, RADIUS, and certificate settings
- Review authentication status, counters, and system messages
- Reauthenticate active sessions after an authorization policy changes

Port-based authentication methods

Explains how 802.1X and MAB control access to network services according to client capabilities, authentication parameters, and authentication results on Cisco 8000 Series Routers.

Port-based authentication methods are access-control mechanisms that

- authenticate a client before allowing it to use network services
- use an authenticator port to control client traffic, and
- support 802.1X-capable and non-802.1X-capable devices.

802.1X uses EAP-based authentication. MAB uses a client MAC address as an authentication parameter and can support devices that do not support 802.1X.

This table summarizes the authentication methods covered in this chapter.

Table 91: Port-based authentication methods

Method	Authentication parameter	Typical use
802.1X	EAP credentials or certificates	Authenticate devices that support IEEE 802.1X before allowing normal traffic
MAB	Client MAC address	Authenticate devices that do not support 802.1X or use MAB as a fallback method

802.1X port-based authentication

Describes the Layer 2 access-control method that blocks client traffic until the client completes authentication through an authenticator and authentication server.

802.1X port-based authentication is a Layer 2 network-access control method that

- uses a client-server model to authenticate a client network device
- controls a physical port before allowing normal traffic, and
- allows access after the client successfully authenticates.

The client requests network access through an authenticator. The authentication server, typically a RADIUS server, verifies the client credentials and authorizes access.

Table 92: Feature History Table

Feature Name	Release Information	Feature Description
802.1X port-based authentication	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100], Modular Systems (8800 [LC ASIC: Q100, Q200, P100])(select variants only*) *This feature is extended to these hardware variants: • 8201 • 8201-32FH • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 8711-48Z-M • 8011-4G24Y4H-I • 8011-32Y8L2H2FH • 8011-12G12X4Y-A Line cards: • 88-LC0-36FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
802.1X port-based authentication	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100]); Modular Systems (8800 [LC ASIC: P100]) You can now secure network access by requiring client network devices to authenticate with encrypted digital certificates before gaining access. The 802.1X port-based authentication ensures that a port remains closed to all traffic until the connected client successfully completes authentication using the Extensible Authentication Protocol with TLS (EAP-TLS) encryption. This prevents unauthorized access and enforces secure, certificate-based communication, enhancing network security and integrity.

How 802.1X port-based authentication works

Describes how the router changes a port from unauthorized to authorized after the authentication server validates the connected supplicant and permits normal client traffic.

802.1X authentication is configured on a Cisco 8000 Series Router to prevent an unauthorized supplicant from accessing network services.

Summary

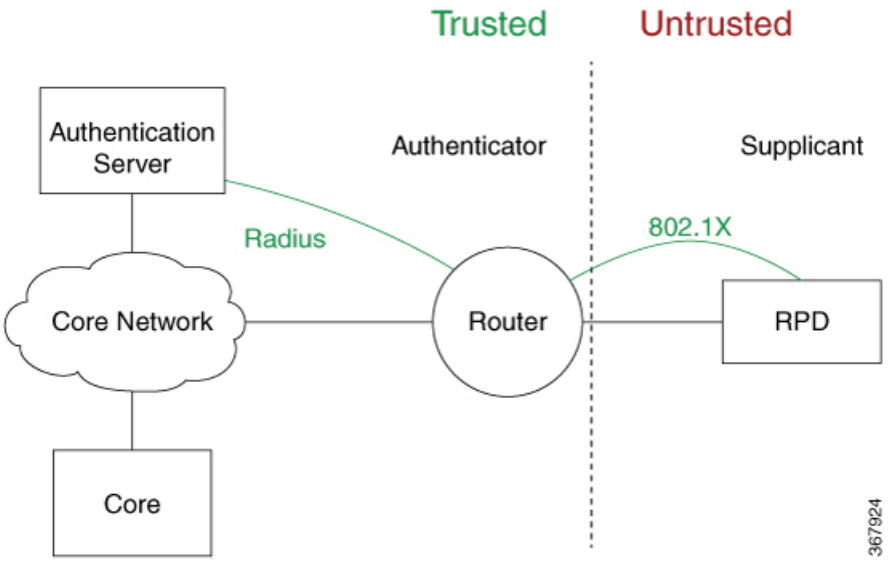
The authentication process includes these components:

- The supplicant requests network access.
- The authenticator controls access on the router port.
- The authentication server validates the supplicant and authorizes access.

While the port is unauthorized, 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) packets. EAPOL frames can use EtherType 0x888E or the Cisco-defined EtherType 0x876F.

Workflow

Figure 21: Topology for IEEE 802.1X port-based authentication



These stages describe how the port processes a client:

1. The router places the port in the Unauthorized state before the supplicant is authenticated.
   - The port blocks normal traffic to and from the supplicant.
   - The port permits EAPOL traffic.
2. The supplicant exchanges authentication information with the authentication server through the authenticator.
3. After successful authentication, the router moves the port to the Authorized state.
   - The router allows normal traffic from the authenticated client.
   - The router can periodically reauthenticate the client.
4. If reauthentication fails, the router blocks the port and returns it to the Unauthorized state.
5. If the link changes from up to down or the router receives an EAPOL-logoff frame, the router returns the port to the Unauthorized state.

Result

Only an authenticated client can send normal traffic through the controlled port.

802.1X protocols, roles, and prerequisites

Lists the protocols, device roles, software, certificate, server, and time-synchronization requirements for configuring 802.1X port-based authentication for remote and local EAP deployments.

Use this reference to identify the components required for 802.1X authentication.

802.1X port-based authentication supports these protocols:

- EAP (Extensible Authentication Protocol) is a flexible authentication framework that supports multiple authentication methods.
- EAP-TLS (Extensible Authentication Protocol with Transport Layer Security) uses TLS for encryption and mutual authentication between the client and server.
- EAPOL (Extensible Authentication Protocol over LAN) carries EAP packets over a wired or wireless LAN.

The devices in an IEEE 802.1X network have these roles:

- Authenticator: A router that facilitates authentication for clients on the same LAN.
- Supplicant: A network device or client that seeks authentication from an authenticator on a point-to-point LAN segment.
- Authentication server: A RADIUS server that verifies client credentials and authorizes access through the authenticator.

Before you configure 802.1X port-based authentication:

- Install the K9sec RPM to enable the feature.
- Configure supported EAP methods on the RADIUS or EAP server and supplicant when remote authentication is used.
- Use EAP-TLS when the router operates as a local EAP server.
- Configure a Certificate Authority (CA) server with a valid certificate.
- Synchronize the supplicant, authenticator, and CA server with Network Time Protocol (NTP). Certificates might not be validated if the devices are not synchronized.

802.1X host modes and traffic behavior

Compares single-host and multi-auth modes and explains how 802.1X handles tagged, untagged, and EAPOL traffic on controlled physical router ports.

Use this reference to select a host mode and understand port traffic before and after authentication.

The supported 802.1X host modes are:

- Single-host: The port allows a single host or client to be authenticated and allows ingress traffic only from the authenticated peer. A security violation is detected if more than one client is present.
- Multi-auth: This is the default host mode. Multiple hosts can independently authenticate through the same port, and ingress traffic is allowed from all authenticated peers. The router supports up to 20 clients using 802.1X in multi-authentication mode.

802.1X applies these traffic rules:

- 802.1X port authentication must be configured on physical ports.
- VLAN sub-interfaces must have preconfigured VLAN IDs.

- All VLAN-tagged traffic is dropped until the port successfully authenticates the client.
- No default VLAN assignment is provided for unauthenticated MAC addresses or untagged traffic.
- Authenticated MAC addresses are validated at the main port, independent of VLAN assignment.
- VLAN-tagged and untagged traffic is allowed only for authenticated MAC addresses.
- Untagged EAPOL traffic is always allowed.
- Port control is enforced only on ingress traffic.

802.1X usage guidelines and restrictions

Lists the platform, physical-port, VLAN, traffic, and hardware restrictions that apply when you implement 802.1X port-based authentication on Cisco 8000 Series Routers.

Consider these restrictions and usage guidelines when implementing 802.1X port-based authentication:

- Configure port authentication only on physical ports.
- Supported modes are single-host and multi-auth.
- Configure VLAN IDs before you use VLAN sub-interfaces.
- All other untagged traffic is dropped until successful authentication.
- 802.1X is not supported on the following hardware variants:
  - 8608-SYS
  - 8404-RSP1-48-EM
  - 8404-RSP1-2FH/4H

Configure 802.1X host modes

Configures the 802.1X host mode under the authenticator configuration to control how many clients can authenticate on a physical port.

Configure single-host or multi-auth mode for an 802.1X-controlled port.

The default host mode is multi-auth. The host mode is configured under the authenticator mode in a dot1x profile.

Before you begin

Configure a dot1x profile before you configure the host mode.

Procedure

Configure the host mode in the dot1x profile.

Example

```
Router# configure terminal
Router(config)# dot1x profile sample-profile
Router(config-dot1x-auth)# pae authenticator
Router(config-dot1x-auth-auth)# host-mode multi-auth
Router(config-dot1x-auth-auth)# commit
```

Replace `multi-auth` with `single-host` when the port must allow only one authenticated client.

The dot1x profile contains the selected 802.1X host mode.

Configure 802.1X with remote RADIUS authentication

Configures centralized 802.1X authentication with a remote RADIUS server, verifies the configuration, and attaches the authenticator profile to a physical interface.

Configure 802.1X port-based authentication with a remote RADIUS server.

The port remains unauthorized until the connected device successfully authenticates through the remote RADIUS server.

Before you begin

- Verify that the RADIUS server is operational and reachable.
- Obtain the pre-shared key for secure communication between the router and RADIUS server.

Procedure

---

1. Configure the RADIUS server.

   Example

   ```
   Router# configure terminal
   Router(config)# radius-server host 209.165.200.225 auth-port 1646 key
   sample-secret007
   Router(config)# radius-server vsa attribute ignore unknown
   Router(config)# commit
   Router# show run radius
   ```

2. Configure the default 802.1X authentication method.

   Example

   ```
   Router(config)# aaa authentication dot1x default group radius
   Router(config)# commit
   ```

3. Verify the configuration using the **show run aaa** command.

   Example

   ```
   Router# show run aaa
   ```

   Only the default AAA method is supported for 802.1X authentication.

4. Configure the 802.1X authenticator profile.

   Example

   ```
   Router(config)# dot1x profile sample-auth
   Router(config-dot1x-auth)# pae authenticator
   Router(config-dot1x-auth)# authenticator
   Router(config-dot1x-auth-auth)# timer reauth-time 3600
   ```

```
Router(config-dot1x-auth-auth)# host-mode multi-auth
Router(config-dot1x-auth-auth)# commit
```

Verify the configuration using the **show run dot1x** command.

5. Attach the 802.1X profile to the interface.

Example

```
Router(config)# interface HundredGigE 0/3/0/0
Router(config-if)# dot1x profile sample-auth
Router(config-if)# commit
```

Verify the configuration using the **show run interface HundredGigE 0/3/0/0** command.

The interface uses 802.1X EAP-TLS authentication through the remote RADIUS server.

Configure 802.1X with local EAP authentication

Configures the router as a local EAP server so that it can mutually authenticate 802.1X clients with EAP-TLS and certificates.

Configure 802.1X port-based authentication with a locally hosted EAP server.

Local EAP authentication uses EAP-TLS with TLS version 1.2. A Master Session Key (MSK) is generated after successful authentication.

Before you begin

Obtain a Certificate Authority (CA), certificate enrollment information, and an EAP profile name.

Procedure

---

1. Generate an RSA key pair.

Example

```
Router# crypto key generate rsa <keypair-label>
```

2. Configure a trustpoint.

Example

```
Router(config)# crypto ca trustpoint <tp_name>
Router(config-trustp)# enrollment url <ca-url>
Router(config-trustp)# subject-name <x.500-name>
Router(config-trustp)# rsakeypair <keypair-label>
Router(config-trustp)# commit
```

A trustpoint identifies the CA, stores CA-specific configuration parameters, and associates an enrolled identity certificate.

3. Configure a domain name.

Example

```
Router(config)# domain name <domain-name>
```

The domain name is required for certificate enrollment.

4. Authenticate and enroll the certificates.

Example

```
Router# crypto ca authenticate <tp_name>
Router# crypto ca enroll <tp_name>
```

5. Ensure that the CA issues the required authentication certificates.
6. Configure an EAP profile.

Example

```
Router(config)# eap profile <profile_name>
Router(config-eap)# identity <user-name>
Router(config-eap)# method tls pki-trustpoint <tp_name>
Router(config-eap)# commit
```

7. Configure the local 802.1X authenticator profile.

Example

```
Router(config)# dot1x profile sample-local-auth
Router(config-dot1x-auth)# pae authenticator
Router(config-dot1x-auth)# authenticator
Router(config-dot1x-auth-auth)# eap profile <profile_name>
Router(config-dot1x-auth-auth)# host-mode multi-auth
Router(config-dot1x-auth-auth)# timer reauth-time 3600
Router(config-dot1x-auth-auth)# commit
```

8. Attach the 802.1X profile to an interface.

Example

```
Router(config)# interface <interface-name>
Router(config-if)# dot1x profile sample-local-auth
Router(config-if)# commit
```

The router uses a local EAP server to authenticate connected devices with EAP-TLS and certificates.

Configure a router as an 802.1X supplicant

Configures a router to authenticate with an upstream authenticator by using an EAP profile, EAP-TLS, certificates, and a supplicant profile.

Configure the router as a supplicant in an 802.1X authentication environment.

Before you begin

Before you configure the router as an 802.1X supplicant:

- Generate an RSA key pair for certificate-based authentication.
- Configure a trustpoint with the appropriate CA.
- Configure a domain name.
- Obtain the CA certificate and enroll the device certificate.
- Configure an EAP profile for authentication.

Procedure

1. Configure the 802.1X supplicant profile.

   Example

```
Router(config)# dot1x profile sample-supp
Router(config-dot1x-supp)# pae supplicant
Router(config-dot1x-supp)# supplicant
Router(config-dot1x-supp-supp)# eap profile <profile_name>
Router(config-dot1x-supp-supp)# commit
```

2. Attach the supplicant profile to an interface.

   Example

```
Router(config)# interface <interface-name>
Router(config-if)# dot1x profile sample-supp
Router(config-if)# commit
```

The router authenticates with an upstream authenticator as an 802.1X supplicant by using EAP-TLS and certificates.

Verify 802.1X port-based authentication

Verifies 802.1X port status, client authentication, programming status, reauthentication timers, and authenticator and supplicant system messages on an interface and resulting port access.

Confirm that 802.1X authentication succeeds and that the port allows traffic for the authenticated client.

Use show command output and syslog messages to verify authenticator and supplicant behavior.

Procedure

1. Display detailed 802.1X information for the interface.

   Example

```
Router# show dot1x interface HundredGigE 0/0/1/0 detail
Dot1x info for HundredGigE 0/0/1/0
-----------------------------------------------------------
Interface short name : Hu 0/0/1/0
Interface handle : 0x4080
Interface MAC : 021a.9eeb.6a59
Ethertype : 888E
PAE : Authenticator
```

```
Dot1x Port Status : AUTHORIZED
Dot1x Profile : sample-test-prof
L2 Transport : FALSE
Authenticator:
Port Control : Enabled
Config Dependency : Resolved
Eap profile : None
ReAuth : Disabled
Client List:
Supplicant : 027e.15f2.cae7
Programming Status : Add Success
Auth SM State : Authenticated
Auth Bend SM State : Idle
Last authen time : 2018 Dec 11 17:00:30.912
Last authen server : 10.77.132.66
Time to next reauth : 0 day(s), 00:51:39
MKA Interface:
Dot1x Tie Break Role : NA (Only applicable for PAE role both)
EAP Based Macsec : Disabled
MKA Start time : NA
MKA Stop time : NA
MKA Response time : NA
```

2. Review authenticator syslog messages.

   Successful authentication displays messages for port control, client authentication, and client access. Authentication failure displays `%L2-DOT1X-5-AUTH_FAIL` and client removal messages. An unreachable authentication server displays `%L2-DOT1X-5-AAA_UNREACHABLE`. An unconfigured authentication method displays `%L2-DOT1X-4-NO_AUTHENTICATION_METHOD`.

   Example

```
%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled
%L2-DOT1X-5-AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client
027E.15F2.CAE7
%L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled
 For Client 027E.15F2.CAE7
%L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0 : Port Control Disabled
%L2-DOT1X-5-AUTH_FAIL : Hu0/0/1/0 : Authentication fail for client
027E.15F2.CAE7
%L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access
Disabled For Client 027E.15F2.CAE7
%L2-DOT1X-5-AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client
027E.15F2.CAE7, Retrying Authentication
%L2-DOT1X-4-NO_AUTHENTICATION_METHOD : Hu0/0/1/0 : No authentication method
configured
```

3. Review supplicant syslog messages.

   Example

```
%L2-DOT1X-5-SUPP_SUCCESS : Hu0/0/1/0 : Authentication successful with
authenticator 008a.96a4.b050
%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with
authenticator 0000.0000.0000.0000
%L2-DOT1X-5-SUPP_FAIL : Hu0/0/1/0 : Authentication successful with
authenticator 008a.96a4.b028
```

The interface is authorized when the output shows an authorized port and an authenticated client.

MAC Authentication Bypass

Describes how MAB authenticates a client by using its MAC address when the client cannot use 802.1X, and how supported platforms extend MAB to multiple clients on one port.

MAC Authentication Bypass is a network fallback authentication method that

- uses the MAC address of a connected client as an authentication parameter
- uses a remote RADIUS server to authorize the client, and
- allows controlled access for devices that do not support 802.1X.

MAB can operate as a standalone method or as a fallback method for 802.1X. Multi-auth MAB can authenticate multiple clients independently on one port on supported platforms.

Table 93: Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Multi-auth MAC Authentication Bypass	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Multi-auth MAC Authentication Bypass	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Multi-auth MAC Authentication Bypass	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Multi-auth MAC Authentication Bypass	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200]) (*select variants only) You can enhance network flexibility by enabling multiple hosts on a single port using MAC Authentication Bypass (MAB). The router now supports up to two clients per port by expanding its MAC learning capability from one to two. It authenticates each MAC address individually, allowing multi-domain authentication and enabling independent management of two endpoints. This feature simplifies network management and increases the connectivity options for devices per port. *This feature is supported on: • 8201-SYS • 8101-32FH
MAC Authentication Bypass	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
MAC Authentication Bypass	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
MAC Authentication Bypass	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
MAC Authentication Bypass	Release 7.3.4 Release 7.5.2	Based on the MAC address of the end device or the client connected to the router port, this feature enables port control functionality for your router. This functionality provides controlled access to network services for end devices that do not support other authentication methods such as IEEE 802.1X port-based authentication. The MAB support is only for the single-host mode. This feature introduces these commands and options: • **mab** option in the **dot1x profile** command • **mab-retry-time** option in the **authenticator** command • **clear mab** • **show mab**

How MAC Authentication Bypass works

Describes how the router learns a client MAC address, sends it to a RADIUS server, and programs the port after authentication succeeds.

MAB provides controlled network access for a client that is connected to a router port and does not use 802.1X authentication.

Summary

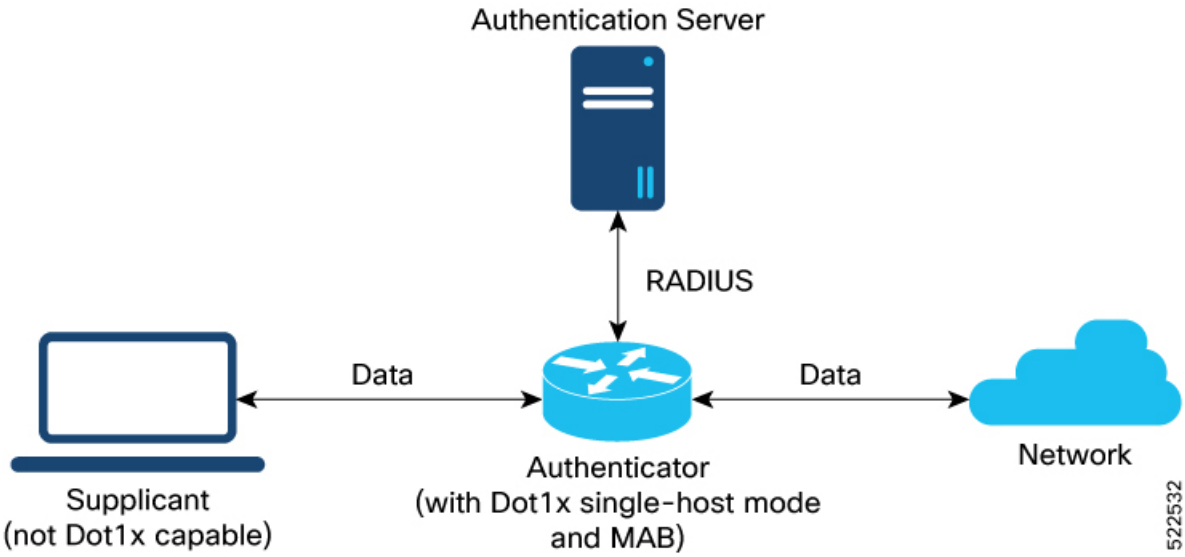
The MAB process includes these components:

- The supplicant or client sends data through the port.
- The authenticator learns and authenticates the client MAC address.
- The RADIUS authentication server maintains the authorized MAC-address database.
- The network receives traffic from an authorized client.

The router uses the client MAC address as the authenticating parameter.

Workflow

Figure 22: MAB topology



These stages describe how MAB controls client access:

1. The router receives an incoming data packet from the connected client and learns the source MAC address.
2. The router sends the learned MAC address to the external RADIUS server for authentication.
3. If the RADIUS server returns an Access-Accept message, the router authenticates the MAC address, programs it on the port, and allows the client traffic to be forwarded to the network.
4. If the RADIUS server returns an Access-Reject message, the router treats the MAC address as unauthenticated and drops further data packets from the client.
5. In multi-auth mode, the router continues MAC learning after authenticating a client until authentication begins for the second client. Each MAC address is authenticated and managed independently.

Result

MAB provides port-controlled access based on the authentication result for each client MAC address.

MAB authentication failure behavior

Lists how MAB handles rejected authentication requests and unreachable RADIUS servers for previously authenticated and unauthenticated clients during standalone operation.

Use this reference to understand the router response when MAB authentication fails.

Table 94: Authentication failure scenarios with MAB

Authentication failure scenario	Expected MAB feature behavior
RADIUS server rejects the authentication request	The router: • deletes the client programming on the port, provided the client was previously authenticated. • clears the client session upon the expiry of the quietWhile timer, which is set to 60 seconds. • switches the port back to MAC learning mode to relearn a new MAC address.

Authentication failure scenario	Expected MAB feature behavior
RADIUS server is not reachable during authentication	With **server dead action auth-retry** command configured: • The router retains the programing of the client that was already authenticated. Else, the router deletes it. • Router retries the authentication process with the RADIUS server at an interval of 60 seconds until the server becomes available. You can configure this interval using the **authenticator timer mab-retry-time** command. • The router does not attempt to learn any new MAC address on the port. • To clear the client session and its programming on the router, you must use the **clear mab session** command. • The router puts the port back in MAC learning mode to relearn a new MAC address. Similarly, for an unauthenticated client, if the authentication does not happen after the retries, the router deletes the client context and puts the port back in MAC learning mode. Without **server dead action auth-retry** command configuration: • The router deletes the programming of the client that was already authenticated and retries authentication. • If the client is still not authenticated, the router automatically clears the client session. • The router puts the port back in MAC learning mode to relearn a new MAC address.

MAC Authentication Bypass requirements and restrictions

Lists the remote-AAA, operating-mode, configuration, and platform restrictions that apply to standalone MAC Authentication Bypass on Cisco 8000 Series Routers.

Review these requirements and restrictions before you configure MAB.

Configure the following items before you configure MAB:

- A remote RADIUS server using the **radius-server** command
- An authentication method using the **aaa authentication dot1x** command
- An 802.1X profile using the **dot1x profile** command in XR configuration mode
- An authenticator with a reauthentication time, host mode, and server-unreachable retry action

MAB has these restrictions:

- User authentication can use only a remote AAA server; the local AAA server on the router is not supported.
- MAB works only as a standalone feature and not as a fallback mechanism for another authentication failure.
- Centralized systems do not support MAB.

Configure MAC Authentication Bypass

Configures MAB in a dot1x profile, sets the client retry timer, attaches the profile to a router interface, and displays the resulting configuration.

Configure MAB to authenticate client MAC addresses through a remote RADIUS server.

Before you begin

- Configure the remote RADIUS server and the default dot1x AAA authentication method.
- Configure the 802.1X profile and authenticator parameters.

Procedure

1. Enable MAB in a dot1x profile.

   Example

   ```
   Router# configure
   Router(config)# dot1x profile sample-test-mab
   Router(dot1x-sample-test-mab)# mab
   Router(dot1x-sample-test-mab)# commit
   ```

2. Configure the MAB authenticator retry timer.

   Example

   ```
   Router# configure
   Router(config)# dot1x profile sample-test-mab
   Router(dot1x-sample-test-mab)# authenticator
   Router(dot1x-sample-test-mab-auth)# timer mab-retry-time 60
   Router(dot1x-sample-test-mab-auth)# commit
   ```

3. Attach the dot1x profile to the interface.

   Example

   ```
   Router(config)# interface GigabitEthernet0/0/0/0
   Router(config-intf)# dot1x profile sample-test-mab
   Router(config-intf)# commit
   ```

4. Review the resulting running configuration.

   Example

   ```
   Router# show running-configuration
   !
   radius-server host <ip-address> auth-port <auth-port-num> acct-port
   <acct-port-num>
   key 7 <key>
   !
   aaa authentication dot1x default group radius
   interface GigabitEthernet0/0/0/0
   dot1x profile sample-test-mab
   !
   dot1x profile sample-test-mab
   ```

```
mab
authenticator
timer reauth-time 60
timer mab-retry-time 60
host-mode single-host
server dead action auth-retry
!
end
```

MAB is enabled on the interface, and the router can authenticate the connected client MAC address through the remote RADIUS server.

Verify MAC Authentication Bypass configuration

Verifies MAB summary, client status, interface details, timers, authentication method, programming status, and interface authentication statistics after client authentication completes.

Confirm that MAB has authorized the expected client and programmed its MAC address on the interface.

Procedure

1. Check the MAB summary.

   Example

```
Router# show mab summary
Fri Apr 1 16:37:32.340 IST
NODE: node0_0_CPU0
==============================================================================
Interface-Name Client Status
==============================================================================
Gi0/0/0/0 1122.3344.5566 Authorized
Router#
```

2. Verify the detailed MAB status.

   Example

```
Router# show mab detail
Fri Apr 1 16:37:37.140 IST
MAB info for GigabitEthernet0/0/0/0
------------------------------------------------------------
InterfaceName : Gi0/0/0/0
InterfaceHandle : 0x00000060
HostMode : single-host
PortControl : Enabled
PuntState : Stop Success
PuntSummary : Punt disabled
Client:
MAC Address : 1122.3344.5566
Status : Authorized
SM State : Terminate
ReauthTimeout : 60s, Remaining 0 day(s), 00:00:46
RetryTimeout : 60s, timer not started yet
AuthMethod : PAP (remote)
LastAuthTime : 2022 Apr 01 16:37:23.634
```

```
ProgrammingStatus : Add Success
Router#
```

3. Verify the MAB interface summary and details.

Example

```
Router# show mab interface gigabitEthernet 0/0/0/0
Fri Apr 1 16:38:27.715 IST
=============================================================================
Interface-Name Client Status
=============================================================================
Gi0/0/0/0 1122.3344.5566 Authorized
Router# show mab interface gigabitEthernet 0/0/0/0 detail
Fri Apr 1 16:38:31.543 IST
MAB info for GigabitEthernet0/0/0/0
-----------------------------------------------------------------
InterfaceName : Gi0/0/0/0
InterfaceHandle : 0x00000060
HostMode : single-host
PortControl : Enabled
PuntState : Stop Success
PuntSummary : Punt disabled
Client:
MAC Address : 1122.3344.5566
Status : Authorized
SM State : Terminate
ReauthTimeout : 60s, Remaining 0 day(s), 00:00:51
RetryTimeout : 60s, timer not started yet
AuthMethod : PAP (remote)
LastAuthTime : 2022 Apr 01 16:38:23.640
ProgrammingStatus : Add Success
Router#
```

4. Verify MAB interface statistics.

Example

```
Router# show mab statistics interface gigabitEthernet 0/0/0/0
Fri Apr 1 16:41:23.011 IST
InterfaceName : GigabitEthernet0/0/0/0
-----------------------------------------------------------------
MAC Learning:
RxTotal : 0
RxNoSrcMac : 0
RxNoIdb : 0
Port Control:
EnableSuccess : 1
EnableFail : 0
UpdateSuccess : 0
UpdateFail : 0
PuntStartSuccess : 0
PuntStartFail : 0
PuntStopSuccess : 1
PuntStopFail : 0
AddClientSuccess : 1
AddClientFail : 0
RemoveClientSuccess : 0
RemoveClientFail : 0
Client :
```

```
MAC Address : 1122.3344.5566
Authentication:
Success : 1406
Fail : 0
Timeout : 0
AAA Unreachable : 0
Router#
```

The output identifies the authorized client, successful MAC programming, and MAB authentication counters.

MAC Authentication Bypass system logs

Lists the system logs that indicate MAB port-control state, client-authentication results, client-access changes, and AAA-unreachable events on Cisco 8000 Series Routers.

Use these messages to identify the result of MAB port control and client authentication.

Table 95: MAB port-control messages

Condition	Message
Port-control enable succeeds	%L2-DOT1X-5-PORT_CONTROL_ENABLE_SUCCESS : Hu0/0/1/0 : Port Control Enabled with Single-Host mode
Port-control enable fails	%L2-DOT1X-5-PORT_CONTROL_ENABLE_FAILURE : Hu0/0/1/0 : Failed to enable port-control
Port-control disable succeeds	%L2-DOT1X-5-PORT_CONTROL_DISABLE_SUCCESS : Hu0/0/1/0 : Port Control Disabled
Port-control disable fails	%L2-DOT1X-5-PORT_CONTROL_DISABLE_FAILURE : Hu0/0/1/0 : Failed to disable port-control
MAB authentication succeeds	%L2-DOT1X-5-MAB_AUTH_SUCCESS : Hu0/0/1/0 : Authentication successful for client <mac-address> %L2-DOT1X-5-PORT_CONTROL_ADD_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Enabled For Client <mac-address>
MAB authentication fails	%L2-DOT1X-5-MAB_AUTH_FAIL : Hu0/0/1/0 : Authentication failed for client <mac-address> %L2-DOT1X-5-PORT_CONTROL_REMOVE_CLIENT_SUCCESS : Hu0/0/1/0 : Port Access Disabled For Client <mac-address>
Authentication server is unreachable	%L2-DOT1X-5-MAB_AAA_UNREACHABLE : Hu0/0/1/0 : AAA server unreachable for client 027E.15F2.CAE7, Retrying Authentication

802.1X authentication with MAC Authentication Bypass fallback

Explains how MAB provides fallback authentication for clients that cannot complete 802.1X while preserving 802.1X as the preferred authentication method.

802.1X authentication with MAC Authentication Bypass (MAB) fallback is a combined access-control mode that

- uses 802.1X as the primary authentication method

- uses MAB when the client does not complete 802.1X authentication, and
- terminates MAB authorization when 802.1X authentication succeeds.

This mode supports networks that contain both 802.1X-capable and non-802.1X-capable devices.

Table 96: Table 3: Feature History Table

Feature Name	Release Information	Feature Description
MAC Authentication Bypass fallback method for 802.1X authentication	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100]); Modular Systems (8800 [LC ASIC: Q200, P100]) (select variants only*) You can use MAC Authentication Bypass (MAB) as a fallback method to enhance network security and flexibility when routers do not support the 802.1X protocol. By default, 802.1X authentication is set as the primary authentication method. In multi-authentication mode, a router supports up to 20 MAB clients simultaneously, in networks with a mix of 802.1X-capable and non-802.1X-capable devices. The feature introduces these changes: CLI: • show dot1x port authentication *This feature is supported on: • 8212-48FH-M • 88-LC0-36FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

How 802.1X authentication falls back to MAC Authentication Bypass

Describes how the router detects an 802.1X-capable client, falls back to MAB when required, and gives precedence to a successful 802.1X session.

MAB fallback supports networks with both 802.1X-capable and non-802.1X-capable devices.

Summary

The router can prioritize 802.1X authentication by using these methods:

- EAPOL detection: The router monitors for EAPOL packets and gives a successful 802.1X session precedence over MAB authorization.
- Change of Authorization (CoA): The AAA server sends a RADIUS CoA request to enforce 802.1X authentication.

Workflow

These stages describe fallback authentication:

1. During initial authentication, the router waits for EAPOL packets for three intervals of 10 seconds each.
2. During reauthentication, the router waits for EAPOL packets for three intervals of 30 seconds each.

3. If no EAPOL packets are detected, the client does not support 802.1X, or the remote server reports an 802.1X failure, the router attempts MAB.
4. If the RADIUS server times out during 802.1X authentication and the dot1x profile explicitly sets **server dead action auth-fail**, the router falls back to MAB.
5. If 802.1X authentication succeeds, the router terminates MAB authorization and gives control to the 802.1X session.

Result

The router authenticates each client with 802.1X when possible and uses MAB when the configured fallback conditions occur.

802.1X and MAC Authentication Bypass failure behavior

Lists common 802.1X and MAB failure conditions and explains when the router retries authentication, deletes client programming, or returns to MAC learning.

Use this reference to understand authentication failure handling during MAB fallback.

802.1X authentication can fail when:

- No EAPOL is received because the client does not support 802.1X or does not respond to the EAPOL request.
- The RADIUS server sends an Access-Reject message.
- The RADIUS server is unreachable while the default server dead action auth-fail command is applied.

MAB authentication can fail when:

- The RADIUS server sends an Access-Reject message.
- The RADIUS server is unreachable while the default server dead action auth-fail command is applied.

This table lists how the router responds to failed authentication.

Table 97: Authentication failure behavior

Failure condition	Router behavior
802.1X does not receive EAPOL	The router proceeds with MAB when fallback is configured.
802.1X authentication fails	The router proceeds with MAB when fallback is configured.
All authentication methods fail	After 60 seconds, the router deletes the client and associated programming.

Configure 802.1X authentication with MAB fallback

Configures 802.1X and MAB authentication on one port, verifies the selected authentication method, and confirms the resulting client authorization state.

Configure 802.1X and MAB authentication methods and verify the result of each attempted method.

802.1X is the primary method. MAB is used when the client does not complete 802.1X authentication.

Before you begin

Configure the RADIUS server and the default dot1x authentication method.

Procedure

1. Configure a dot1x profile with 802.1X and MAB.

   Example

```
Router# configure
Router(config)# dot1x profile sample-auth-mab
Router(config-dot1x-auth-mab)# pae authenticator
Router(config-dot1x-auth-mab)# mab
Router(config-dot1x-auth-mab)# authenticator timer reauth-time 60
Router(config-dot1x-auth-mab)# authenticator server dead action auth-fail
Router(config-dot1x-auth-mab)# commit
```

2. Verify the dot1x profile.

   Example

```
Router# show run dot1x profile sample-auth-mab
dot1x profile sample-auth-mab
mab
pae authenticator
authenticator
timer reauth-time 60
server dead action auth-retry
```

3. Attach the profile to the interface.

   Example

```
Router# configure
Router(config)# interface GigabitEthernet 0/1/0/0
Router(config-if)# dot1x profile sample-auth-mab
Router(config-if)# commit
```

   Run the **show run interface GigabitEthernet 0/1/0/0** command to verify that the port control is configured on the interface.

4. Verify MAB authorization after 802.1X receives no EAPOL.

   Example

```
Router# show dot1x port authentication
Interface Client Method Status
GigabitEthernet 0/1/0/0 ac4a.6730.0620 mab Authorized
Router# show dot1x port authentication detail
Port Authentication Info for GigabitEthernet 0/1/0/0
----------------------------------------------------------------
Interface Handle : 0x80001c0
Interface State : Up
Port Status : Authorized (1/1)
Profile : sample-test-auth-mab
Method List : dot1x, mab
Client :
MAC Address : ac4a.6730.0620
Status : Authorized
Programming Status : Add Success
```

```
Unauthorized Timer : 60s, timer off
Method:
dot1x : Failed (No EAPoL received)
mab : Success
```

5. Verify 802.1X authorization when the client completes 802.1X.

   Example

```
Router# show dot1x port authentication
NODE: node0_2_CPU0
================================================================================
Interface Client Method Status
================================================================================
GigabitEthernet 0/1/0/0 ac4a.6730.0620 dot1x Authorized

Router# show dot1x port authentication detail
Port Authentication Info for GigabitEthernet 0/1/0/0
--------------------------------------------------------------------
Interface Handle : 0x80001c0
Interface State : Up
Port Status : Authorized (1/1)
Profile : sample-test-auth-mab
Method List : dot1x, mab
Client :
MAC Address : ac4a.6730.0620
Status : Authorized
Programming Status : Add Success
Unauthorized Timer : 60s, timer off
Method:
dot1x : Success
mab : Not Run
```

The output identifies whether MAB or 802.1X authorized the client.

RADIUS Change of Authorization for 802.1X and MAB sessions

Explains how RADIUS CoA dynamically changes active 802.1X and MAB session attributes and initiates reauthentication without requiring complete session termination.

RADIUS Change of Authorization is an AAA mechanism that

- changes attributes of active client sessions by using CoA requests
- allows external AAA or policy servers to initiate session reauthentication, and
- supports 802.1X and MAB session updates without complete session termination.

CoA requests are sent from an external server to the router that functions as the CoA server. The router supports the RADIUS CoA extensions defined in RFC 5176.

Table 98: Table 4: Feature History Table

Feature Name	Release Information	Feature Description
RADIUS Change of Authorization for 802.1X and MAB sessions	Release 25.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200], 8700 [ASIC: K100]); 8010 [ASIC: A100]); Modular Systems (8800 [LC ASIC: Q100, Q200]) You can now prioritize dot1x authentication on a dot1x-enabled port that is already running in fallback MAC authentication bypass (MAB) mode on your router. This feature allows Change of Authorization (CoA) for 802.1X- or MAB-authenticated clients through RADIUS CoA requests from the external AAA server or policy server. This capability allows external servers to trigger authentication updates for the client sessions without the need to disconnect and reconnect. The feature introduces these changes: CLI: • The **show dot1x port authentication** command is modified to include the **statistics** keyword. • The **show radius dynamic-author** command is modified to include the **Dot1x COA Statistics** field in the output.

How RADIUS Change of Authorization works

Describes how an external AAA or policy server sends a CoA request to reauthenticate an active 802.1X or MAB client session.

RADIUS CoA uses a push model in which an external server sends a request to the router after a policy changes for a user or user group.

Summary

The process includes these components:

- The external AAA or policy server sends the CoA request.
- The router identifies the client session.
- The router reruns the requested authentication method and applies the updated policy.

Workflow

These stages describe RADIUS CoA processing:

1. The AAA server sends a standard CoA-Request containing the Cisco VSAs for session reauthentication.
2. The router matches the session-identification attributes in the request to the client session.
3. If the request is valid, the router reauthenticates the session by using the last successful method or by rerunning the authentication process.
4. The router returns a CoA-ACK when it processes the request or a CoA-NAK when it rejects the request.

Result

RADIUS CoA changes active session authorization without requiring the client to disconnect and reconnect.

RADIUS Change of Authorization response codes and session attributes

Lists CoA response codes, error strings, rate limits, Cisco VSAs, and session-identification attributes used for 802.1X and MAB session reauthentication.

Use this reference to construct and interpret RADIUS CoA requests.

CoA-Request processing returns one of these response codes:

- CoA acknowledgment (CoA-ACK)
- CoA non-acknowledgment (CoA-NAK)

These error strings can appear in a CoA-NAK message:

- Client not found
- Client not authorized
- Invalid Request
- Invalid NAS port ID
- Resource unavailable at the moment, try after sometime
- Missing mandatory subscriber:command=reauthenticate
- Missing mandatory Calling-Station-Id
- Invalid subscriber command received
- Calling-Station-Id format is invalid

Rate limits specific to CoA-NAK messages are:

- Per-client CoA rate limit: 1 request every 60s
- Global CoA rate limit: 128 requests per minute

Include these Cisco VSAs to initiate session reauthentication:

Table 99: CoA commands and Cisco VSAs

CoA command	Cisco VSA
session reauthenticate	Cisco:Avpair="subscriber:command=reauthenticate"
session reauthenticate-type	Cisco:Avpair="subscriber:reauthenticate-type=last" or Cisco:Avpair="subscriber:reauthenticate-type=rerun"

The **reauthenticate-type** value defines the method used for reauthentication:

- last: Uses the authentication method that last succeeded on the session.
- rerun: Completely reruns the authentication process.

These rules apply to 802.1X and MAB session re-authentication using RADIUS CoA requests:

- The **subscriber:command=reauthenticate** VSA must be present.

- If **subscriber:reauthenticate-type** is omitted, the router uses the last successful authentication method.
- **subscriber:reauthenticate-type** is valid only when included with **subscriber:command=reauthenticate**.

Session-identification attributes must match the router session; otherwise, the router returns a CoA-NAK.

Table 100: Session identification attributes in CoA requests

Attribute	IETF attribute	Importance and content
Calling-Station-Id	Attribute 31	Mandatory; contains the host MAC address.
NAS-Port-ID	Attribute 87	Optional; contains the complete interface name on which the client MAC address is currently served.

Guidelines for RADIUS CoA for 802.1X and MAB sessions

Provides guidelines for securing RADIUS CoA communication between the router and external server by using the supported pre-shared-key authentication method.

Apply the following guideline when you configure RADIUS CoA.

Explicitly configure the CoA server with the IP address of the CoA client and the associated pre-shared key to ensure secure communication.

The router supports only pre-shared key-based RADIUS CoA.

Enable RADIUS CoA for 802.1X and MAB sessions

Configures the router to receive RADIUS CoA requests, reauthenticate active 802.1X and MAB sessions, and verify authorization results in system logs.

Enable RADIUS CoA so an external AAA or policy server can reauthenticate an active 802.1X or MAB session.

Before you begin

Configure the CoA server with the IP address of the CoA client and the associated pre-shared key.

Procedure

1. Enable the RADIUS CoA server on the router.

   Example

```
Router# configure
Router(config)# aaa server radius dynamic-author
Router(config-dynamic-author)# port 1700
Router(config-dynamic-author)# server-key sample-test-password
Router(config-dynamic-author)# client 10.101.130.122 vrf default
Router(config-dynamic-author-client)# server-key sample-test-password
Router(config-dynamic-author-client)# commit
```

2. Set up the remote RADIUS server on the router.

Example

```
Router# configure
Router(config)# radius-server vsa attribute ignore unknown
Router(config)# radius-server host 10.101.130.122 auth-port 1812 acct-port
1813
Router(config-radius-host)# key sample-test-password
Router(config-radius-host)# commit
```

3. Send a CoA request with Cisco VSAs from the CoA client to change authorization for an existing 802.1X or MAB session.

   The CoA client can be an external AAA server or policy server.

4. Check the router system logs to verify the change of authorization.

Table 101: System logs for CoA-ACK scenarios

If the client is...	And re-authentication type is...	Then the router displays..
802.1X-authenticated	**last**	`%L2-DOT1X-5-COA_ACK : Hu0/0/0/5: Processed the CoA to 'reauthenticate(last)' client 008a.96a4.b052` `%L2-DOT1X-5-AUTH_SUCCESS : Hu0/0/0/5: Authentication successful for client 008a.96a4.b052`
802.1X-authenticated	**rerun**	`%L2-DOT1X-5-COA_ACK : Hu0/0/0/5: Processed the CoA to 'reauthenticate(rerun)' client 008a.96a4.b052` `%L2-DOT1X-5-AUTH_SUCCESS : Hu0/0/0/5: Authentication successful for client 008a.96a4.b052`
MAB-authenticated	**last**	`%L2-DOT1X-5-COA_ACK : Hu0/0/0/5: Processed the CoA to 'reauthenticate(last)' client f4db.e62e.c61b` `%L2-DOT1X-5-MAB_AUTH_SUCCESS : Hu0/0/0/5: Authentication successful for client f4db.e62e.c61b`

MAB-authenticated	**rerun**	`%L2-DOT1X-5-COA_ACK : Hu0/0/0/5: Processed the CoA to 'reauthenticate(rerun)' client f4db.e62e.c61b`
		The router processes 802.1X authentication first. If it fails, it proceeds with the re-authentication process using MAB.
		`%L2-DOT1X-5-MAB_AUTH_SUCCESS : Hu0/0/0/5: Authentication successful for client f4db.e62e.c61b`

Table 102: System logs for CoA-NAK scenarios

If the client is...	And re-authentication type is...	Then the router displays..
802.1X- or MAB-unauthorized	**last**	%L2-DOT1X-5-COA_NAK : Hu0/0/0/5:008a.9634.5cd4: Rejected the CoA to 'reauthenticate(last)' because of 'Client not authorized'
802.1X- or MAB-unauthorized	**rerun**	%L2-DOT1X-5-COA_NAK : Hu0/0/0/5:008a.9634.5cd4: Rejected the CoA to 'reauthenticate(rerun)' because of 'Client not authorized'

23 Device Ownership and Authorized Operations

Topics:

- Trusted ownership and authorized operations
- Device ownership
- Cisco MASA service
- How routers are provisioned using ownership vouchers
- Third-party key packages
- Consent tokens

Provides the security concepts and operational references for establishing device ownership, obtaining ownership vouchers, provisioning third-party key packages, and authorizing privileged router operations.

Use this chapter to establish trusted ownership and prepare the certificates, vouchers, keys, and consent tokens required for authorized router operations.

This chapter covers these security topics:

1. Trusted ownership and authorized operations introduces the shared security model.
2. Device ownership explains ownership artifacts, establishment, clearing, and security profiles.
3. Cisco MASA service describes ownership-voucher creation and MASA interactions.
4. Third-party key packages explains package versions and key provisioning.
5. Consent tokens explains authorization for restricted actions.

Trusted ownership and authorized operations

Explains how ownership certificates, ownership vouchers, customer keys, and consent tokens establish trust and authorize protected router operations.

Trusted ownership and authorized operations are security controls that

- establish a trusted relationship between a router and its management network
- validate certificates, vouchers, and third-party software before protected operations, and
- authorize restricted actions through customer-controlled keys and consent tokens.

Device ownership establishment allows the network to validate the router and the router to validate the network. It also helps validate third-party application signatures before installation.

The ownership voucher identifies the owner known to the manufacturer and carries the pinned-domain certificate used to authenticate later network interactions. Customer key packages add, remove, or revoke keys used by third-party applications and consent-token workflows.

Consent tokens authorize restricted actions only after the requester is authenticated as the legitimate device owner or as an authorized customer-controlled operator.

Device ownership

Explains the ownership certificates, ownership vouchers, and serial numbers used to establish trusted relationships between Cisco IOS XR routers and their management networks.

Device ownership establishment (DOE) is a process that

- establishes a device's first trusted connection with the device management service (network) and vice versa
- validates the router to the network and the network to the router, and
- validates third-party application signatures before installation and enables protected operations such as Reimage Protection and customer key-package installation.

Ownership artifacts

Device ownership establishment uses these artifacts:

- **Owner Certificate**: The owner certificate (OC) is an X.509 certificate [RFC5280] that identifies an owner, such as an organization. A certificate authority (CA) can sign the OC. The OC contains the owner certificate and all intermediate certificates leading to the pinned-domain-cert (PDC) specified in the OV.
- **Ownership Voucher**: The ownership voucher (OV) [RFC8366] securely identifies the owner known to the manufacturer. The device manufacturer signs the OV. The OV verifies that the OC has a chain of trust leading to the trusted PDC included in the OV. Cisco's Manufacturer Authorized Signing Authority (MASA) service issues OVs.
- **Serial Number**: The serial number (SN) is typically in the format LLLYYWWSSSS. LLL represents the manufacturing location, YY and WW represent the year and week of manufacture, and SSSS is the unique router code. Find the SN at the bottom of the router or run **show platform security device-info location** *location*.

DOE is required to enable or disable Reimage Protection and to install and enable a customer key package for third-party application onboarding.

How device ownership is established

Describes how the customer, Cisco, and the router exchange and validate ownership artifacts before a trusted relationship is established.

Device ownership establishment (DOE) validates the router to the network and the network to the router.

Summary

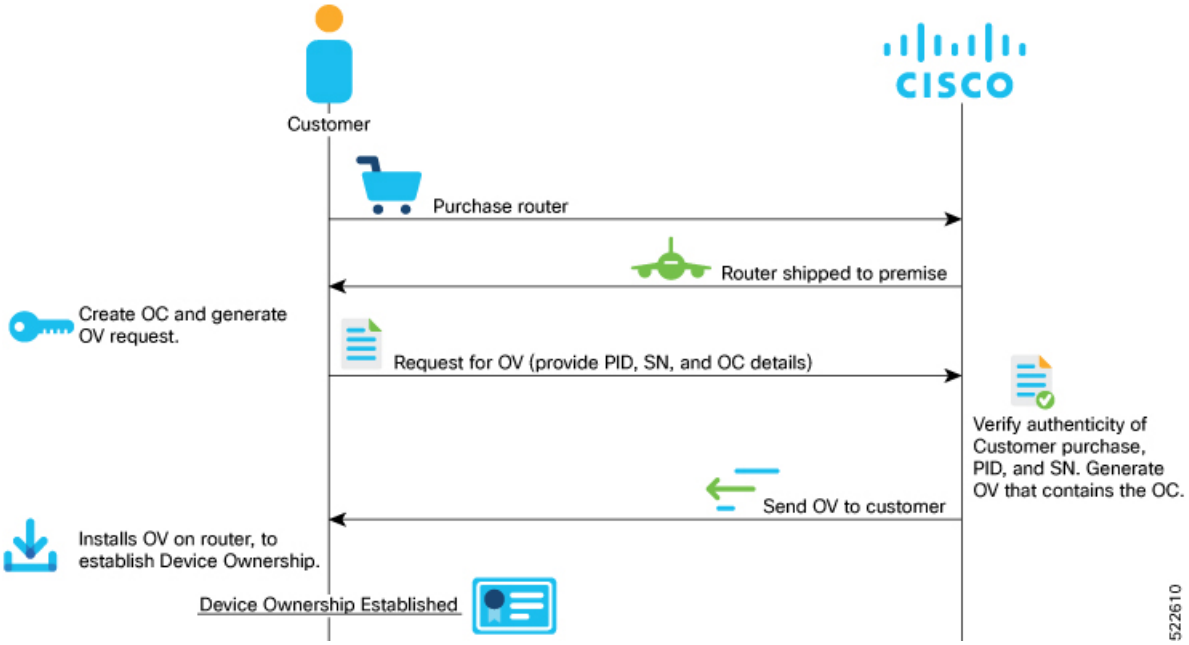
The ownership establishment process uses these artifacts:

- The customer creates an OC and requests an OV for each router serial number.
- Cisco verifies the artifacts and generates the OV.
- The customer installs the artifacts and the router validates the OV and OC chain.

The resulting trust relationship supports secure third-party application and key-package operations.

Workflow

Figure 23: Workflow for device ownership establishment



These are the stages involved in establishing device ownership:

1. The customer creates the OC using OpenSSL commands. The router later verifies that the OC chains to the PDC included in the OV.
2. The customer creates an OV request for each router serial number and sends the request to Cisco with the serial number.
   - Reference scripts for creating OCs are available at GitHub.
3. Cisco verifies the authenticity of the artifacts and generates the OVs.
   - Modifying the .vcj output file can cause processing delays and errors.
   - If duplicate .vcj files are generated for a serial number or card, the first valid file is applied and duplicates are rejected.
   - Multiple .vcj files can be archived in a flat TAR file and provided to the GISO script or XR interfaces.

Result

Cisco generates ownership vouchers that the customer can install on the router for ownership validation.

Install and verify device ownership

Installs the ownership voucher and owner certificate on a router, then verifies that the router has established device ownership.

Install the OV and OC on the router and verify the resulting device-ownership state.

The installation command requires a TAR file of OVs, with each OV representing a route processor. You can include an OV for the chassis or maintain a single TAR file for the route processors you purchased.

The OC must have a trust chain leading to the PDC in the OV, and the command accepts only the latest OV. The OC and OV can also be included in a GISO.

Before you begin

- Have the OC and the OV TAR file available on the router.
- Ensure that the OC trust chain leads to the PDC in the OV.

Procedure

1. Install the OV and OC on the router.

   Example

   ```
   Router# platform security device-ownership
   /disk0:/testing2/deliverable/bulk_ovs.tar.gz /disk0:/testing2/oc-single.cms
   location all
   ```

   The router validates the OC chain and adds the PDC and OC to a special trust point as CA certificates.

2. Verify that device ownership is established.

   Example

   ```
   Router# show platform security device-ownership
   Performing operation on all nodes..
   ========================
   Location : 0/RP0/CPU0
   ========================

   Trustpoint : device_ownership
   ================================================
   CA certificate
   Certificate:
   Data:
   Version: 3 (0x2)
   Serial Number:
   f6:20:61:bd:db:22:30:74
   ...truncated...

   Router# show logging
   ownership_app[66652]: %SECURITY-OWNERSHIP-6-INFO: Device ownership established.
   ```

Device ownership is established, enabling secure validation between the router and network.

The router establishes device ownership and is ready for secure validation with its management network.

Clear device ownership

Clears the ownership artifacts from a router through the customer consent-token challenge-response workflow and verifies the cleared state.

Clear the PDC, OC, and artifacts such as key packages associated with an OV.

Clear device ownership uses the consent token workflow. For more information, see Consent tokens.

Clearing device ownership uses the consent-token workflow and affects every application that depends on the OC.

Before you begin

- Enable the customer consent-token key.
- Ensure that the required consent-token response is available.

Procedure

---

1. Generate a clear-device-ownership challenge for the router location.

   Example

```
Router# clear device-ownership challenge customer location 0/RP1/CPU0
Do you want to clear the device ownership [Y/N]? Y
Tue Feb 10 05:20:07.811 UTC
+-------------------------------------+
Node location: node0_RP1_CPU0
+-------------------------------------+
Challenge string:
2GrKAAAQyBAAQAAAEAgAEAAABAAMAC1b98TiegyWHBAAQul/tEEkuKEBi2va2SsEMgJABAAAAUGARhmNkBwAEWWUjZAgABzg4MDAtUlAIAAtGI0MjMzAyUDhIMw=
+-------------------------------------+
```

2. Submit the challenge response for the router location.

   Example

```
Router# clear device-ownership response location 0/RP1/CPU0
*************************************************************
Please enter challenge response string for node location node0_RP1_CPU0
*************************************************************
```

   The router processes the consent-token response.

3. Verify that device ownership is cleared.

   Example

```
Router# show platform security device-ownership
No platform ownership information found.
```

Device ownership is cleared.

What to do next

After you clear device ownership, applications that depend on the OC do not function.

Security profiles for Cisco IOS XR software

Explains how Cisco IOS XR security profiles support different security requirements and control the security level applied to the system.

Security profiles for Cisco IOS XR software are configurable security settings that

- support classic ZTP, secure ZTP, third-party RPM signature verification, and partner RPM GISO
- provide different security levels for IOS XR system integrity and protection, and
- let you select and override profile settings based on your business needs.

Table 103: Feature History Table

Feature Name	Release Information	Feature Description
Security profiles for Cisco IOS XR software	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This feature supports different security profiles to ensure integrity and protection of the IOS XR system when transitioning between security profiles. The supported security profiles are Strict, Default, and Relaxed.

Each security profile supports user-configurable security parameters or security levels. The profiles are added using the ownership voucher (OV).

Table 104: Cisco IOS XR security profiles

Security profile	Purpose
Strict	Enables all security features. This profile corresponds to the High security level to enable security features. For more information, see *Install Owner and Partner RPMs using IOS XR Install Infrastructure* in *Cisco IOS XR Setup and Upgrade Guide for Cisco 8000 Series Routers*.
Default	All security features have a default value, which exists from releases prior to Cisco IOS XR Release 26.1.1.
Relaxed	Sets the security level of the device to Low where certain security checks are not enforced, such as verifying signatures of owner RPMs. For more information, see *Install Owner and Partner RPMs using IOS XR Install Infrastructure* in *Cisco IOS XR Setup and Upgrade Guide for Cisco 8000 Series Routers*.

> **Note**
>
> From Cisco IOS XR Release 26.1.1, when using the Cisco MASA web interface to create OVs, select the same security profile setting for all cards (serial numbers) that are part of, or are intended to be part of, the same chassis. The MASA interface does not enforce this because the cards and their corresponding OVs may belong to, or be intended for, different chassis with different security requirements.

Cisco MASA service

Explains how the Cisco Manufacturer Authorized Signing Authority service creates ownership vouchers, supports router authentication, and provides web, REST, and gRPC interactions.

The Cisco Manufacturer Authorized Signing Authority (MASA) service is a service that

- creates ownership vouchers (OVs) for a Cisco IOS XR router, which together with the owner certificate (OC) certify that the router belongs to a given customer
- supports secure ZTP workflows and securely booting a device on a 5G cell site over a third-party Ethernet service, and
- allows customers to download OVs and view their logging and audit information.

The MASA service also enables Cisco Account teams to assign device serial numbers to customers and view the logging, verification, and audit details of OVs.

Table 105: Feature History Table

Feature Name	Release Information	Feature Description
Extend Device Ownership	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Extend Device Ownership	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Extend Device Ownership	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Extend Device Ownership	Release 7.10.1	Your router can now run in a dual-ownership mode wherein you can securely migrate the operating system from Cisco IOS XR to third-party software such as SONiC. You can only install the signed SONiC image authorized by Cisco using an ownership voucher (OV) and authenticated variables (AV) on the router. This authorization prevents tampering with the software using unauthorized third-party images.
Cisco MASA Service	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Cisco MASA Service	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Cisco MASA Service	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM

Feature Name	Release Information	Feature Description
Cisco MASA Service	IOS XR 7.8.1	The Cisco Manufacturer Authorized Signing Authority (MASA) service creates ownership vouchers (OVs) for a Cisco IOS XR router. These OVs along with the owner certificate (OC) certify that the router belongs to a given customer. Use cases where OVs and OCs are required include secure ZTP workflows and securely booting up your device on a 5G cell site over a third-party ethernet service. You can use the MASA service to download, and view logging and audit of OVs for the routers you own. This service also enables Cisco's Account teams to assign the serial number of a device to customers and view details of the logging, verification, and audit of OVs.

How MASA authenticates routers

Describes how the ZTP server and MASA service authenticate a router and generate ownership vouchers for secure bootstrapping.

The authentication flow identifies and authenticates the router when it boots and checks whether the network can be trusted.

Summary

The authentication flow involves these components:

- The Cisco device boots and communicates with the customer's ZTP server.
- The ZTP server accesses MASA through the Internet using HTTPS.
- MASA authenticates the user, verifies the request, and generates OVs.

The router validates the received OV and verifies the signature on onboarding information with the OC received during bootstrapping.

Workflow

Figure 24: Components of the authentication flow

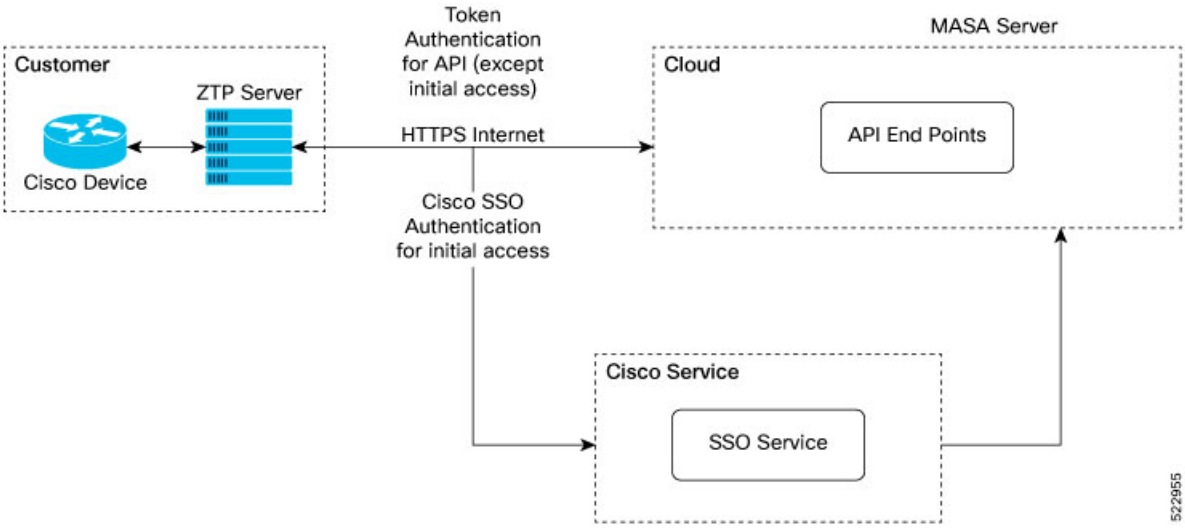
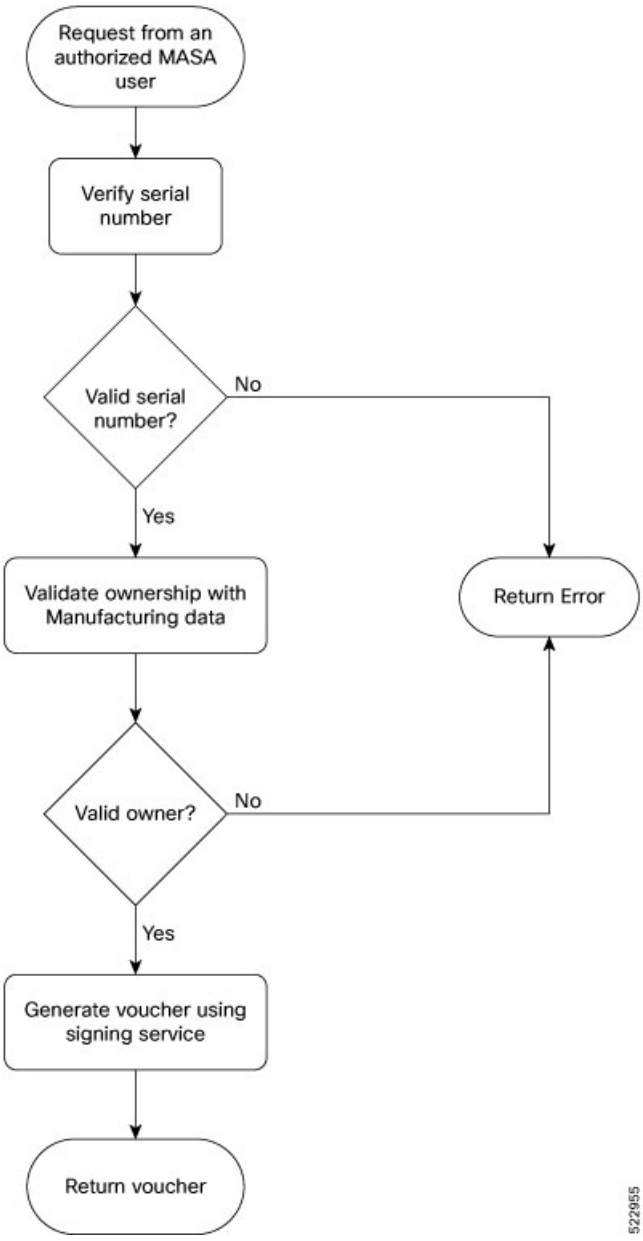


Figure 25: Workflow to obtain ownership vouchers



These stages describe how the service obtains an OV:

1. The customer or ZTP server authenticates to the MASA service. Initial access uses Cisco SSO; subsequent API access can use a token.
2. The ZTP server sends a request to the MASA API through the HTTPS Internet connection.
3. MASA verifies the authenticated user and the router serial number.
4. MASA generates the OV and returns it to the customer or ZTP server.
5. The ZTP server provides the OV to the device, which validates the OV and the onboarding information before provisioning.

Result

The router receives an ownership voucher that supports authentication of the router and its trusted network.

Ownership voucher use cases

Lists the secure bootstrapping, application-hosting, extended-ownership, and Bootz scenarios that use ownership vouchers.

Use this reference to identify when MASA ownership vouchers are required.

These use cases show how ownership vouchers apply:

Secure Zero Touch Provisioning bootstrapping

Secure ZTP bootstraps a router over an untrusted network. MASA provides an OV that authenticates the router and ensures connectivity between the router and network. For more information, see the *Secure Zero Touch Provisioning* chapter in the *System Setup and Software Installation Guide for Cisco 8000 Series Routers*.

> **Note**
>
> MASA can help generate OVs for Cisco routers only.

Application Hosting on XR

Cisco IOS XR Application Hosting provides an IOS XR container for applications that augment XR features.

- Customer Apps are developed by Cisco customers and cannot be signed by Cisco.
- Partner Apps are developed by partners and are signed by Cisco.
- Cisco Apps are developed by Cisco and are signed by Cisco.

MASA can work with the Golden ISO Tool (gisobuild.py) to provide OVs for secure workflows that onboard third-party RPMs on routers running Cisco IOS XR. For more information, see the *Application Hosting Guide for Cisco 8000 Series Routers*.

Extended device ownership

An extended OV moves the state of the Cisco Trusted Platform Module (TPM) and platform keys to allow customized control on the router. It transfers Unified Extensible Firmware Interface (UEFI) database ownership from Cisco generic mode to an extended mode owned by Cisco and the customer, allowing third-party images to be installed. For more information, see the *Migrate from Cisco IOS XR to SONiC on Cisco 8000 Series Routers*.

Deploy router using Bootz

Bootz is a secure zero-touch provisioning solution for data centers. It automates network-device setup, authenticates devices with the Bootz server, and protects remote configuration from unauthorized access. Bootz uses MASA to issue OVs that authenticate network devices. For more information, see the *Deploying Router Using Bootz Protocol* chapter in the *System Setup and Software Installation Guide for Cisco 8000 Series Routers*.

MASA entities and permissions

Describes MASA organizations, administrators, and users, and lists the permissions available for ownership-voucher operations.

Use this reference to identify the MASA role and access requirements for each voucher operation.

These entities interact with the MASA server:

- Organization: A group in MASA specific to a Cisco customer. Data and access are available only to members of that group.
- Admin: One or more initially designated organization members who can invite members, set access restrictions, and adjust organization settings.
- User: A non-admin organization member who can interact with MASA after an Admin invites the user. New users have view-only access by default.

Before interacting with MASA, you must be an authorized user with a Cisco account and an active invitation. Initial authentication uses Cisco SSO at masa.cisco.com. For later sessions, generate access keys called tokens. Tokens can be passed in API headers, can have a custom validity period of up to six months, and can be revoked at any time. Token scope is limited to the user's role.

This example shows a token in the header of a REST API call:

```
Authorization: Bearer
```

The ZTP server must be able to access the Internet. The MASA application is served through HTTPS.

Table 106: User permissions

Type	Regular User	Admin
Invite other People into the organization	Not allowed	Allowed by default
Add or remove permissions for other users	Not allowed	Allowed by default
View all existing vouchers	Allowed by default	Allowed by default
Request new vouchers	Permission can be provided by Admin	Allowed by default
Download vouchers	Permission can be provided by Admin	Allowed by default
Archive vouchers	Permission can be provided by Admin	Allowed by default

MASA web application interactions

Explains how the MASA web application supports authenticated ownership-voucher requests and management for Cisco routers.

MASA web application interactions are web-based voucher-management activities that

- authenticate authorized users through Cisco SSO
- request ownership vouchers by using a PDC and router serial numbers, and
- manage generated ownership vouchers according to assigned user permissions.

Table 107: Feature History Table

Feature Name	Release Information	Feature Description
Pre-upload Pinned-Domain Certificate	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Pre-upload Pinned-Domain Certificate	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I
Pre-upload Pinned-Domain Certificate	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Pre-upload Pinned-Domain Certificate	Release 24.1.1	You can now pre-upload your Pinned-Domain Certificate (PDC) credentials before requesting OVs Ownership Vouchers (OVs) from the MASA server, thus making the voucher request process easier.

Request ownership vouchers through the MASA web application

Requests ownership vouchers through the MASA web application by supplying a pinned-domain certificate and router serial numbers, then manages the generated vouchers.

Request and manage ownership vouchers for Cisco routers through the MASA web application.

The MASA Home page displays recent-request status and links to recently generated ownership vouchers. Your assigned permissions determine which voucher actions are available.

Before you begin

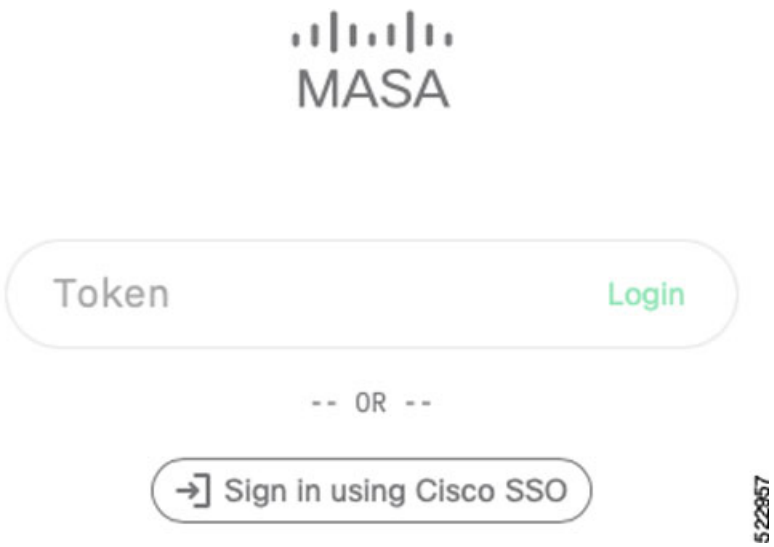
- Have an authorized MASA user account, Cisco account, and invitation.
- Have the router serial number and a pinned-domain certificate (PDC).

Procedure

1. Sign in to the MASA web application.

   Example

   Figure 26: Sign in page—MASA web application



   a. Go to masa.cisco.com.
   b. Click **Sign in using Cisco SSO**.
   c. Enter your username and password.
   d. Accept the End User License Agreement.

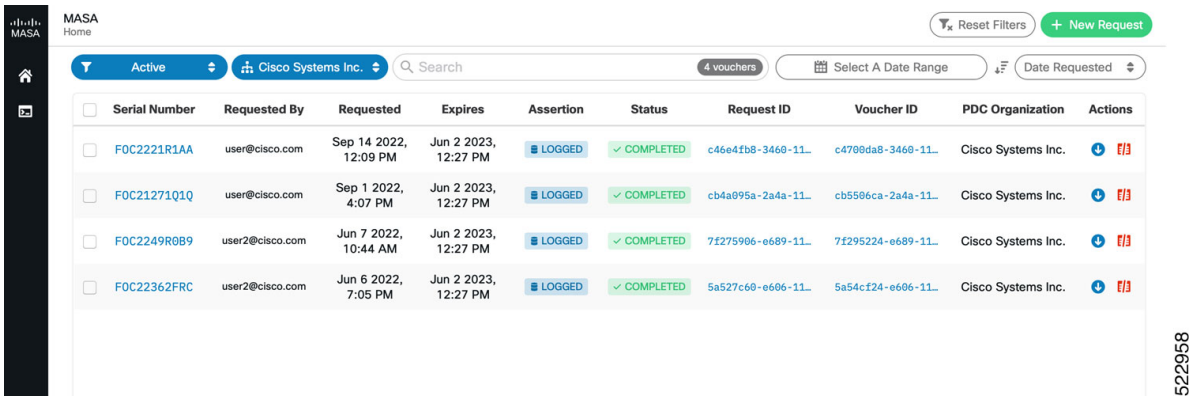
   The MASA Home page displays recent requests and download links.

2. Open the new request form.

   Example

   Click **New Request** on the top right of the Home page.

   Figure 27: Home page—MASA web application

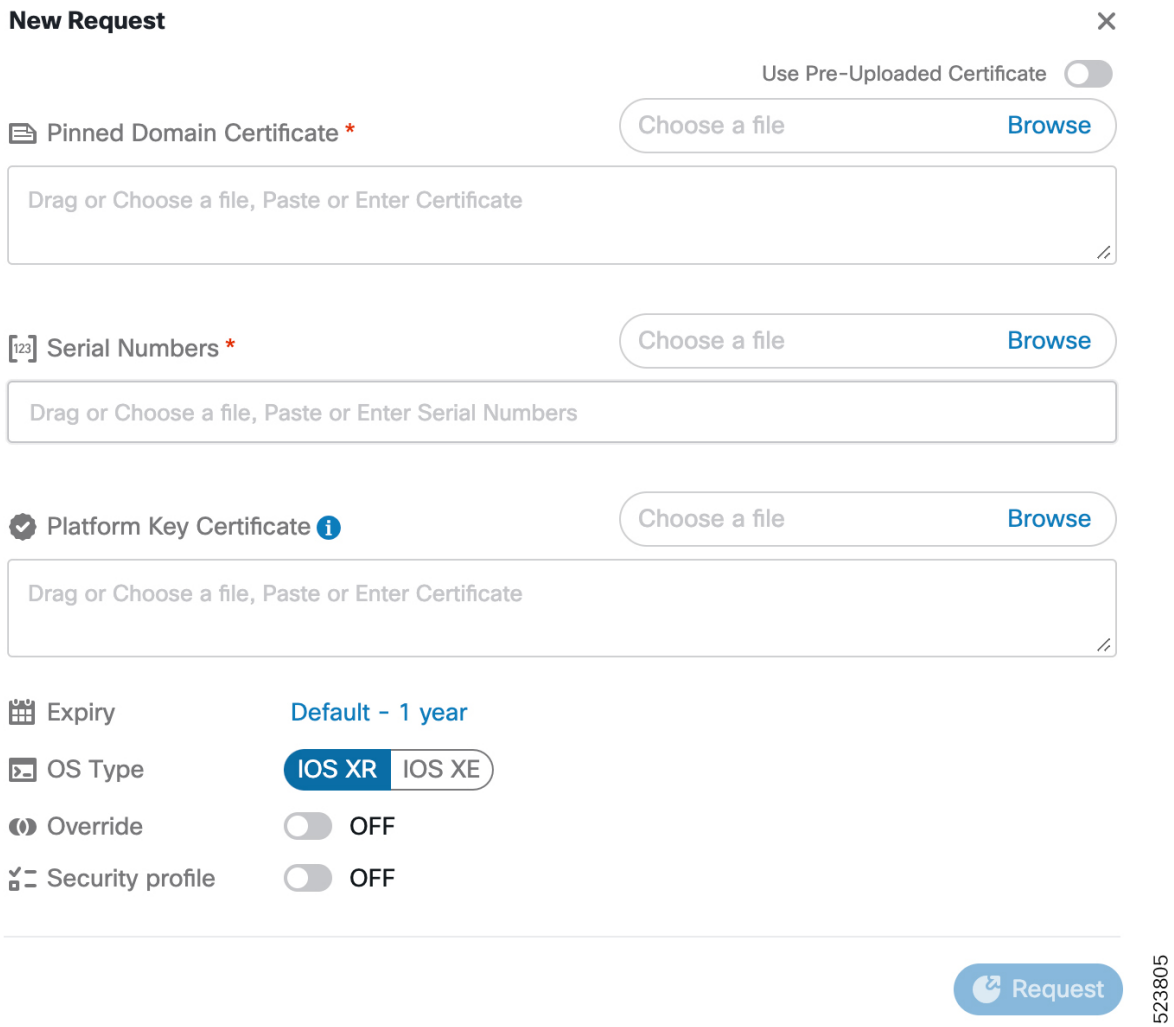


3. Enter the PDC and router serial numbers.

Example

Paste the PDC content or browse to the .pem file. You can pre-upload the certificate before requesting the OV and select it by enabling **use pre-uploaded certificate**. Enter the serial number of one or more routers. Always use the serial number of the RP.

Figure 28: New Request page



The request contains the certificate and router identifiers that MASA uses to generate OVs.

4. Manage the generated ownership vouchers from the Home page.

Example

Depending on your permissions, download or regenerate OVs, view details of past requests, filter, sort, group, and archive requests.

Figure 29: Home page with new OVs displayed



The MASA web application generates and displays the requested ownership vouchers.

MASA REST APIs

Lists the REST endpoints used to request ownership vouchers and retrieve voucher details, including the response fields and serial-number limit.

Use these API references to interact programmatically with the MASA service.

The OpenAPI documentation contains the paths, formats, and structures of the MASA APIs.

Figure 30: MASA API documentation

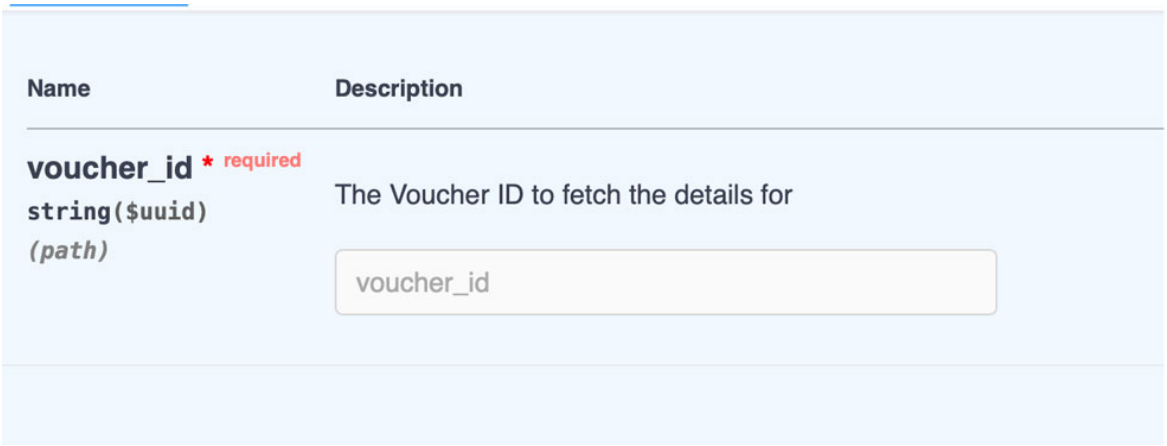


Table 108: Ownership-voucher REST endpoints

Method and path	Purpose
POST /request/ov	Requests an ownership voucher.
GET /voucher/{voucher_id}	Fetches details about an already generated voucher.

This response shows the information returned for a generated voucher:

```
{
  "ok": true,
  "voucher": {
    "req_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "voucher_id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
```

```
    "requested_at": "2022-08-31T09:43:39.719Z",
    "created_at": "2022-08-31T09:43:39.719Z",
    "expires_at": "2022-08-31T09:43:39.719Z",
    "last_renewal_at": "2022-08-31T09:43:39.719Z",
    "assertion": "logged",
    "status": "completed",
    "serial_number": "T8I52J1IKOM",
    "pdc_organization": "Cisco Systems",
    "requested_by": "user1@cisco.com"
  }
}
```

“serial_number” is the serial number of the RP. You can provide up to 20 serial numbers in a single request.

MASA gRPC integration

Explains how gRPC extends MASA API access beyond HTTP and supports efficient, type-safe communication for ownership-voucher and account-management operations.

MASA gRPC integration is an API integration that

- provides gRPC access to MASA APIs in addition to HTTP
- uses Protocol Buffers for efficient, type-safe, and consistent communication between services, and
- supports group, user-role, domain-certificate, and ownership-voucher operations.

Table 109: Feature History Table

Feature Name	Release Information	Feature Description
Interaction with MASA through gRPC	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Interaction with MASA through gRPC	Release 25.1.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100], 8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8712-MOD-M • 8011-4G24Y4H-I

Feature Name	Release Information	Feature Description
Interaction with MASA through gRPC	Release 24.4.1	Introduced in this release on: Fixed Systems (8200 [ASIC: P100], 8700 [ASIC: P100])(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*) *This feature is supported on: • 8212-48FH-M • 8711-32FH-M • 88-LC1-36EH • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM
Interaction with MASA through gRPC	Release 24.1.1	From this release, you can use the gRPC protocol to interact with MASA APIs in addition to the current HTTP protocols. Through structured serialization of data with gRPC's Protocol Buffers, the communication between services is made more efficient, type-safe, and consistent.

MASA gRPC APIs

Lists the gRPC APIs available for MASA group, user-role, domain-certificate, and ownership-voucher operations.

Use this reference to select a MASA gRPC API for a supported operation.

gRPC is available in addition to HTTP. Structured serialization with gRPC Protocol Buffers makes communication between services more efficient, type-safe, and consistent.

Table 110: MASA gRPC APIs

RPC	Description
rpc GetGroup	Returns the domain-certificates (keyed by id), serials, and user/role mappings for that group.
rpc AddUserRole	Assigns a role to a user in a named group. Username is unique to an Org ID.
rpc RemoveUserRole	Removes a role from a user in a named group. Username is unique to an Org ID.
rpc GetUserRole	Returns the roles that the user is assigned in the group. Username is unique to an Org ID. A user can only view roles of another user in the group that it has a role assigned to.
rpc CreateDomainCert	Creates the certificate in the group.
rpc GetDomainCert	Reveals the details of the certificate.
rpc DeleteDomainCert	Deletes the certificate from the database.
rpc GetOwnershipVoucher	Issues an ownership voucher.

For more information on gRPC, see *Use gRPC Protocol to Define Network Operations with Data Models* in the *Programmability Configuration Guide for Cisco 8000 Series Routers*.

How routers are provisioned using ownership vouchers

Describes how Cisco, the customer, the ZTP server, and the device exchange ownership artifacts during secure router provisioning.

The workflow provisions a Cisco IOS XR router by using an ownership voucher obtained through the customer and Cisco MASA service.

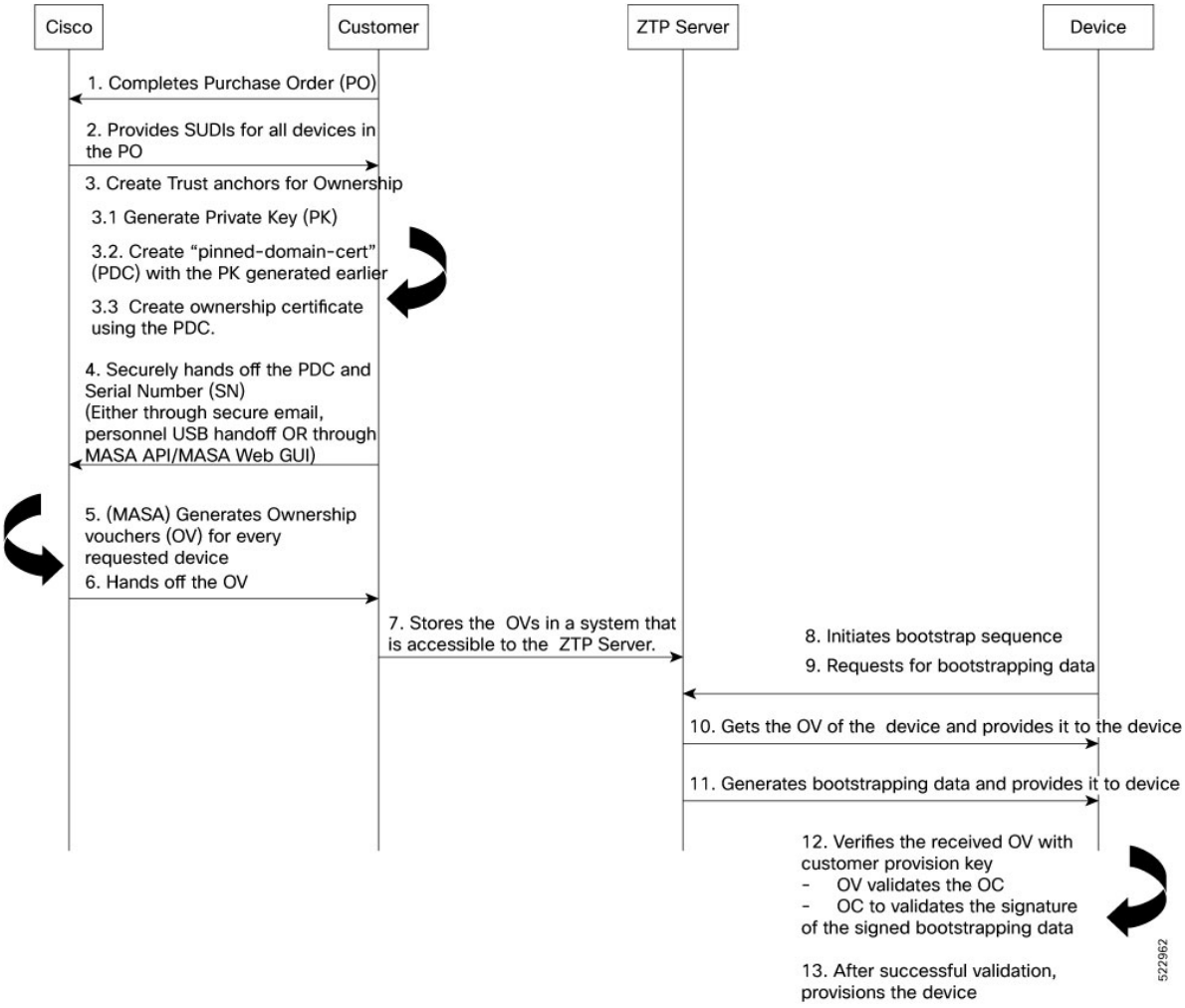
Summary

The workflow includes these actors:

- Cisco: Provides the SUDIs associated with the customer's purchase order.
- Customer: Owns and prepares the trust anchors and ownership artifacts.
- MASA: Issues ownership vouchers for the requested devices.
- ZTP server: Stores and exchanges ownership vouchers and bootstrapping data.
- Device: Validates the ownership artifacts and bootstrapping data before provisioning.

Workflow

Figure 31: Workflow to provision a router using ownership vouchers



These stages describe the provisioning flow:

1. Cisco completes the purchase order and provides SUDIs for all devices in the purchase order.
2. The customer generates a private key, creates a PDC with that key, creates an OC using the PDC, and securely hands off the PDC and serial number through secure email, personal USB handoff, the MASA API, or the MASA web GUI.
3. MASA generates ownership vouchers for every requested device and hands off the OV to the customer.
4. The ZTP server stores the OVs in a system accessible to the ZTP server and the device initiates its bootstrap sequence.
5. The device requests bootstrapping data. The ZTP server gets the device OV and provides it to the device, then generates and provides bootstrapping data.
6. The device verifies the received OV: the OV validates the OC and the OC validates the signature of the signed bootstrapping data.
7. After successful validation, the device is provisioned.

Result

The device is provisioned only after it validates ownership and signed bootstrapping data.

Third-party key packages

Explains how signed key packages onboard owner keys, customer consent tokens, and owner RPM keys on Cisco IOS XR routers.

Key packages are Cryptographic Message Syntax (CMS [RFC5652]) objects that are digitally signed with the private keys of the customer's Ownership Certificate (OC) and can

- add one or more keys to the router
- delete one or more keys from the router, and
- revoke one or more keys from the router.

Key packages are required for onboarding owner keys, customer consent tokens, and owner RPMs. A key package provides a secure mechanism to install owner or third-party public keys (GPG or X.509) on the router.

Table 111: Feature History Table

Feature Name	Release Information	Feature Description
Key package enhancements	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This feature introduces you to the version 3 key package. With the version 3 key package, you can create, validate, sign a key package before the key package is provisioned on the router. Unlike the reserved customer consent token name, CUS-CT, used in version 1 and version 2 key packages, you can now use any name for the customer consent token that you include in the key package. However, to enable the customer consent token workflow, execute the **consent-token customer** command after the key package installation.

Key package versions and Cisco IOS XR releases

Compares key-pkg-ver1, key-pkg-ver2, and key-pkg-ver3 capabilities with the Cisco IOS XR releases that support each version.

Use this reference to select a key package version for the target Cisco IOS XR release.

You can create key packages using the GitHub script for different key package versions and install them on supported Cisco IOS XR releases.

Table 112: Key package versions and IOS XR releases

Key package version	IOS XR release	Purpose
key-pkg-ver1	Cisco IOS XR Release 7.6.x, Cisco IOS XR Release 7.7.x	With key-pkg-ver1, you can create a single key package with signed keys, remove any existing key package, and verify a key package. The key request in the key package supports ADD, DELETE, or REVOKE action.
key-pkg-ver2	Cisco IOS XR Release 7.8.x – Cisco IOS XR Release 25.4.x	With key-pkg-ver2, you can create a super key package with two or more key packages or a single key package containing signed keys, remove any existing key package or super key package, and verify a key package or super key package. The key request in the key package supports ADD, DELETE, or REVOKE action.
key-pkg-ver3	Cisco IOS XR Release 26.1.1 and later	With key-pkg-ver3, you can create a single key package JSON file containing signed keys, validate the key package, and sign the key package. Key requests do not include action (ADD, DELETE, or REVOKE).

Restrictions for key packages

Provides the restrictions that govern key package readiness, timestamp processing, and key removal on the router.

Requirements

Meet these requirements before installing key packages:

- Establish device ownership.
- Ensure that the router is up and running.
- Ensure that no uncommitted install operation is in progress.

Restrictions

These restrictions apply when processing key packages:

- Key packages to add, remove, revoke, or unrevoke customer or Cisco keys must have a timestamp.
- Process key packages in timestamp order. A key package with a timestamp earlier than that of a processed key package is rejected.
- A key package that removes a key fails if installed packages are signed with that key.

How key packages are processed

Describes how signed key packages add, remove, or revoke keys on Cisco IOS XR routers according to the key-package version.

Key packages provide a mechanism for onboarding customer or third-party keys on the router.

Summary

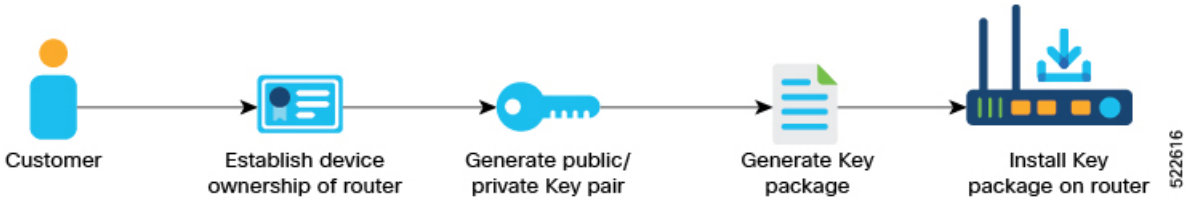
The key-package process involves these components:

- The customer creates a signed key package by using a public-private key pair.
- The router validates and processes the key package.
- The installed keys support third-party application verification and consent-token responses.

Key-package versions 1 and 2 use actions in key requests, while version 3 represents the desired installed-key set.

Workflow

Figure 32: Workflow for installing a key package on the router



These stages describe how the router processes key packages:

1. A customer or third party prepares a public-private key pair and uses the corresponding private key to sign the key package.
2. The router receives a key package and validates its signature and configuration data.
3. For key-package versions 1 and 2, the router applies the ADD, DELETE, or REVOKE action specified in each key request.
4. For key-package version 3, the router uses the keys included in the new request as the desired installed-key set: included keys are installed and omitted existing keys are removed.
5. The router updates its customer allowed list after successful processing. The installed keys can also sign a consent-token response.

Install key packages on the router

Installs a customer or third-party key package on the router and verifies that the package and its keys are available.

Install and verify a customer or third-party key package on the router.

Key package versions 1 and 2 support ADD, DELETE, and REVOKE actions. Starting with Cisco IOS XR Release 26.1.1, include the desired installed-key set in the new key request instead of using those actions.

For example, include K1, K2, and K3 to install K3 when K1 and K2 are installed. Include K1 and K3 to remove K2. You can also use installed keys to sign a consent-token response.

Before you begin

- Ensure that device ownership is established.
- Have access to a Linux machine and the key-package scripts at GitHub.

Procedure

1. Generate an RSA/GPG key pair on a Linux machine.

   Example

   Use the standard OpenSSL commands to generate the key pair. Typically, this key pair is a GPG key, but it can also be an X.509 certificate.

2. Generate a version 1, version 2, or version 3 key package.

   Example

   Use the key-package scripts at GitHub to generate the key package.

3. Install the key package on the router.

   Example

   Copy the key package to the router and run the **platform security key-package customer install** *key-package-file* command.

```
Router# platform security key-package customer install
disk0:/testing2/key-pkg/key_add.kpkg

Key package successfully validated
Config file successfully parsed.
Successfully added key cust-ct.der to TPM
Successfully processed all keys.
Router#
```

   The key package is validated and processed on the router.

4. Verify that the key package is installed.

   Example

```
Router# show platform security key-package customer allowed-list location
0/RP0/CPU0
-----------------------------------------------
Node - node0_RP0_CPU0
-----------------------------------------------

Key Name: D3CUS-CT1
Key:
MIIC7TCCAdUCAQIwDQYJKoZIhvcNAQELBQAwOzELMAkGA1UEBhMCVVMxDDAKBgNV
BAoMA3h6eTEMMAoGA1UECwwDYWJjMRAwDgYDVQQDDAdST09ULUNOMB4XDTIxMDYx
NDE1MjkwOVoXDTI0MDMxMDE1MjkwOVowPjELMAkGA1UEBhMCVVMxDDAKBgNVBAoM
A3h5ejEMMAoGA1UECwwDYWJjMRMwEQYDVQQDDApDVVNULUNULUNOMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyOT2SGTuJcQlAHCsQn4gcoZGK+po1A6g
LPV5AzOBcY0pfXV5eXoxf6S8qbmQP4l4v5MjsHzFTOuouMmijpGYFJv7TORwJ2Xw
weJ5aKbqsYTQlSQSUZ1XxG7AOdHMshVRzy7vIA7LLQJnD0j1F1U2FoRi5NhhY12L
wmYA4aPj1o+LoubAfjF1BVl3vE8rfI0mzsXODJIks+oeJbsq4HmyMbOAzLVdeucp
7bu3S8kDlc1ph4zqm81BkDZgV1++2CoCBWROt9dRZrp+ENw1GEHcXgS659iZpUmj
juG1n0W3Y6br8SE+EqqhMqkAfSbO8vaG02qYtTUNJ5gkMcTljCfDAQIDAQABMA0G
CSqGSIb3DQEBCwUAA4IBAQCDeJ5ov2gG3rj5ttpfibxiakpzl706W9crjIePJka6
CWS7Y3nxt02+PGsBByEcBPV7aU8oH2GfKN4jNZHDChfzGN7rtfRE2CG+ttvTxJLC
Ba+LjzKFSveKgPRG/gAAkZY0hRmTe7FkgmKB4UCi+u0XP3U5VlT5XRP3LGVoX0fC
rY4/GBKkG5eOF+VGD4iyPfOHjrwduO/K2DqDXyUfa1PXZDzatpnin07ShkCJQoT+
u6C1SotJ8mtrFJpePDUsa5W3O2oPROFHd4sGCivt40AbpaWECK+KLpKC+DoqN+46
```

```
tMV79rpQ0mtXo/XfY4UGir4weH9g/e2fct4g+Y2E/BD+

Key Name: D3CUS-CTX
Key:
PNM:APNAM,KNM:AKNAM,
```

> **Note**
>
> Key packages can also be included in a GISO and installed on the router from the GISO. For more information, see *Install signed owner RPMs using GISO* in *System Setup and Software Installation Guide for Cisco 8000 Series Routers*.

The installed key appears in the customer allowed list.

The customer or third-party key package is installed and verified.

How key rotation works

Describes how each key package version installs new keys and removes existing keys during key rotation.

Key rotation installs new keys and deletes installed keys by using one or more key packages based on the key package version.

Summary

The key package versions support rotation in these ways:

- key-pkg-ver-1 and key-pkg-ver-2 use multiple key packages to add and remove keys.
- key-pkg-ver-3 uses one key package that includes keys to install and omits keys to remove.

Workflow

These are the stages in the key rotation process:

1. For key-pkg-ver-1 and key-pkg-ver-2, create two key packages: one to remove an existing key and one to install a new key. For key-pkg-ver-3, create one key package that includes new keys to install and omits existing keys to remove.
2. Install the key package or packages by using **platform security key-package customer install** *pkg-name.kpkg*.

Result

The new key set is installed and the selected existing keys are removed according to the package version.

Consent tokens

Explains how Cisco-signed and customer-signed consent tokens authorize restricted router actions through time-limited, device-specific, single-use challenge-response workflows.

Consent tokens are security tokens that

- are signed by Cisco and used to enable or disable restricted router actions
- are associated with a particular device and must be used within a defined time window, and

- can be used only once.

Cisco verifies that the user requesting a consent token is the legitimate device owner before issuing the token.

Table 113: Feature History Table

Feature Name	Release Information	Feature Description
Customer consent token configuration	Release 26.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This feature allows you to enable the customer consent token workflow so that the key name is linked to the key package. The feature introduces these changes: CLI: **consent-token customer**

Cisco-signed and customer consent tokens

Compares Cisco-signed consent tokens with owner or customer consent tokens and identifies the restricted operations that use them.

Use this reference to select the consent-token workflow for a restricted router operation.

An owner token or customer consent token functions similarly to a consent token. A Cisco-signed token is signed by Cisco. An owner token or customer consent token is signed by a customer key rooted in the customer's OC.

Customer-signed tokens allow customers to control who is authorized to perform restricted actions and enforce stricter authentication requirements. They can be used only when the customer has installed an OV/OC.

Consent tokens are used when you disable secure ZTP or enable or disable lawful intercept. Customer consent tokens are also used to clear device ownership.

How Cisco-signed consent tokens authorize restricted operations

Describes how a router generates a challenge and accepts a Cisco TAC-signed response to enable or disable a restricted feature.

Consent tokens use a challenge-response process. The challenge string generated on the router is valid for a defined time interval.

Summary

The Cisco-signed workflow involves these actors:

- The router generates a challenge containing the device ID, nonce, and requested action.
- The Cisco TAC engineer verifies the requester and signs a response.
- The router validates the signature and matching device ID and nonce.

If you use Cisco's consent-token workflow, contact Cisco TAC for every request to enable or disable certain privileged operations.

Workflow

These are the stages used to provision a Cisco-signed consent token:

1. Generate the challenge string on the router.
2. Submit the challenge to Cisco TAC. After verifying device ownership and authorization, a TAC engineer returns a signed response.
3. The router validates the signature, device ID, and nonce, then enables or disables the requested feature.

Result

The requested restricted feature is enabled or disabled after the signed response is validated.

Provision Cisco-signed consent tokens

Provisions a Cisco-signed consent token by generating a router challenge, obtaining a signed response from Cisco TAC, and accepting the response to enable or disable a restricted feature.

Enable or disable a restricted router feature by using the Cisco-signed consent-token workflow.

The challenge string generated on the router is valid for a defined time interval. The response must be provided within that interval.

Contact Cisco TAC for every request to enable or disable certain privileged operations.

Before you begin

- Identify the supported security feature for which you need a consent token.
- Have access to Cisco TAC to submit the challenge and receive the signed response.

Procedure

1. Generate the challenge string on the router.

   Example

   ```
   Router# request consent-token generate-challenge feature
   ```

   feature can be any supported security feature, such as secure-ztp, lawful-intercept, or factory-reset.

2. Submit the challenge string to a Cisco TAC engineer.

   Example

   Cisco verifies that the requester is the legitimate device owner and is authorized for the requested action. When the request is verified, the TAC engineer provides a signed response string.

3. Accept the signed response on the router.

   Example

   Paste the response string provided by the TAC engineer when prompted.

   ```
   Router# request consent-token accept-response
   ```

4. Confirm that the requested feature is enabled or disabled.

   Example

   The router validates the signature and confirms that the device ID and nonce match its records.

The requested feature is enabled or disabled after the signed response is validated.

The requested restricted feature is enabled or disabled after the Cisco-signed consent-token response is accepted.

Provision customer consent tokens

Describes how an on-premises consent-token server generates signed responses for router challenge strings.

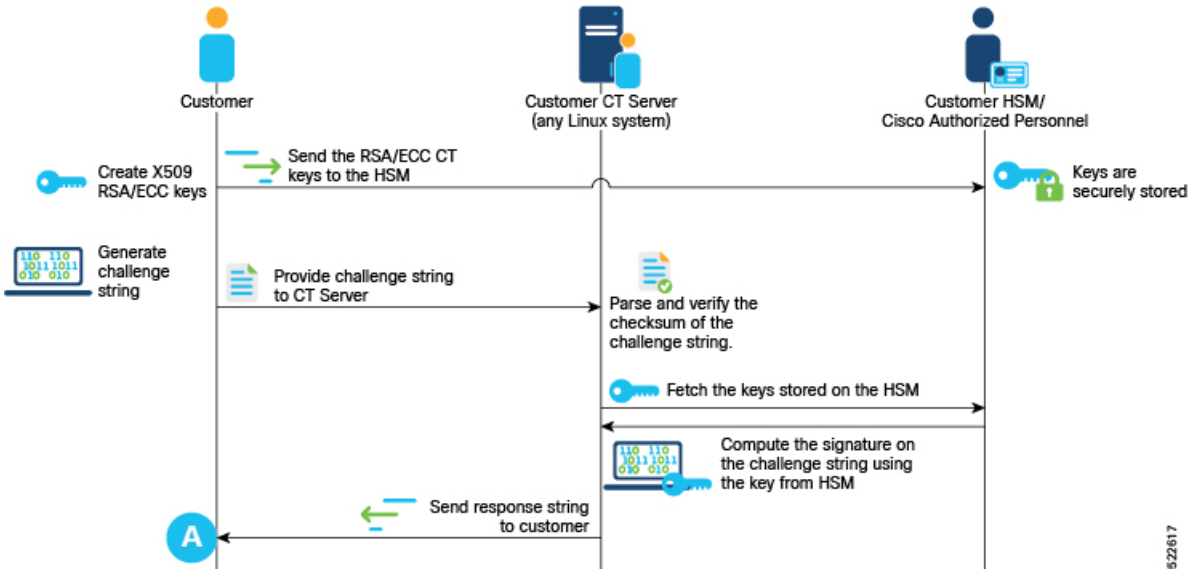
This process sets up a server on your premises to handle customer consent-token challenge responses.

Summary

The customer workflow uses customer keys, a key package, the consent-token configuration, and a signing server.

Workflow

Figure 33: Workflow for the consent token server



These are the stages involved in provisioning customer consent tokens:

1. Generate customer consent-token keys by using the OpenSSL commands.
2. Onboard the customer consent-token keys on the router by using key packages.
3. Enable the customer consent token to link the key name with the key package.

   Enter these commands on the router:

   ```
   Router# configure
   Router(config)# consent-token customer cert-name CT_KEY key-name key1
   product-name prod1
   Router(config)# commit
   ```

   cert-name indicates the consent-token certificate added through a key package. **key-name** indicates the consent-token key, and **product-name** indicates the product name for the consent token.
4. Set up the consent-token server on your premises using the sample `ct_sim.py` script available at GitHub.
5. Generate the challenge for a feature that supports the customer consent-token workflow.

6. Sign the challenge string using the consent-token signing server. The server parses the challenge, verifies the checksum, and computes the signature using the key fetched from the Hardware Security Module (HSM).
7. Accept the response generated by the consent-token signing server.

   The requested feature is enabled or disabled.

24 Gated shell access

Topics:

- Key features of gated shell access
- How gated shell access works
- Restrict shell access using a Cisco-signed Consent Token
- Unrestrict shell access using a Cisco-signed Consent Token
- Restrict shell access using an owner Consent Token
- Unrestrict shell access using an owner Consent Token
- Terminate a pending Consent Token authorization request
- Access a restricted shell

Describes Gated Shell Access, which restricts supported direct, interactive root shell access on Cisco IOS XR platforms.

A gated shell access is a security feature that

- controls direct, interactive root-shell access for supported shell commands,
- requires authorization using a Cisco or customer Consent Token (CT) path before restricting access, and
- stores the selection in HWTAM Secure Object storage.

Consent Token authorization applies to the shell-access restriction operation. This feature uses a separate Consent Token challenge-response workflow for each shell session.

Table 114: Feature History Table

Feature Name	Release Information	Feature Description
Gated shell access	Release 26.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700 [ASIC: P100, K100], 8010 [ASIC: A100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]). Shell access restrictions require Consent Token authorization for direct root shell access initiated through supported shell commands.

Key features of gated shell access

Lists the key features, benefits, direct CLI commands, scope, state behavior, and command interface for gated shell access on Cisco IOS XR platforms.

Use this reference to determine whether a platform supports gated shell access and to identify the associated restrictions and CLI commands.

Key features:

- Gates only direct shell access via supported CLI commands (such as **bash**, **run**, and **attach**). Indirect access through scripts or internal processes is not gated in the initial release.
- It enables restriction or unrestriction of gated shell access, ensuring that only authorized personnel can change shell-access policies.
- Depends on the consent token infrastructure to be present and supported on the platform for operation.
- The restricted-shell selection is stored in an HWTAM secure object.

Table 115: Gated shell access CLI commands

Command	Purpose
platform security shell-access restrict challenge {cisco | owner}	Initiates the Consent Token challenge for the selected Cisco or customer authorization path.
platform security shell-access restrict response <response_string>	Submits the Consent Token response to authorize and complete the shell-access restriction operation.
platform security shell-access restrict terminate-auth	Terminates the pending Consent Token authorization request for shell-access restriction.
platform security shell-access unrestrict challenge {cisco | owner}	Starts the Consent Token authorization flow to remove the restriction.
platform security shell-access unrestrict response <response_string>	Submits the Consent Token response for unrestriction.
platform security shell-access unrestrict terminate-auth	Terminates the pending unrestriction authorization request operation.
show platform security shell-access status	Displays the restricted-shell configuration and handshake status.

The HWTAM Secure Object holds the restricted-shell selection and handshake state. The restricted-shell selection is preserved across reboots and reloads.

How gated shell access works

Describes how restricted shell access invokes the Consent Token workflow and permits a single direct interactive root shell instance on a supported platform.

The process occurs when a user restricts direct interactive shell access on a platform that supports the Consent Token infrastructure.

Summary

When a user restricts shell access, the system records the restriction state in the HWTAM Secure Object. Subsequent supported direct, interactive root shell access attempts check this state and are denied while the restriction is active.

Workflow

These stages describe the gated shell access workflow:

1. A user restricts shell access or requests access to a direct, interactive root shell.
2. The system stores the restricted-shell access state in the HWTAM Secure Object and applies the restriction to supported direct, interactive root shell access.
3. The user must complete the consent token response challenge handshake.
4. The user enters the Consent Token response. When the response succeeds, the system permits one root shell instance for that user session.
5. A user starts the unrestriction authorization flow, submits the response, and the system returns to the default non-gated root shell behavior after successful authorization.

Restrict shell access using a Cisco-signed Consent Token

Restrict direct interactive root shell access on a supported Cisco IOS XR platform and store the restricted-shell selection in the Gated Shell Access HWTAM Secure Object.

Use this task to restrict supported direct, interactive root shell access by using a Cisco-signed Consent Token.

Procedure

1. Generate the challenge string on the router.

   Example

   ```
   RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge cisco
   Thu Aug 20 06:06:33.888 UTC

   +-------------------------------------+
   Node location: node0_RP0_CPU0
   +-------------------------------------+
   Challenge string:
   <challenge_string>
   ```

   This produces a challenge string containing the device ID, a nonce, and the requested action.

2. Submit the challenge string to a Cisco TAC engineer. Cisco verifies that the requester is the legitimate device owner and is authorized to perform the requested action. If verified, the TAC engineer provides a signed response string.
3. Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

   Example

   ```
   RP/0/RP0/CPU0:ios# platform security shell-access restrict response
   Thu Aug 20 06:07:07.090 UTC
   ************************************************************
   Please enter challenge response string for node location node0_RP0_CPU0
   ************************************************************
   <response_string>
   ```

```
Successfully accepted challenge-response for Restrict Shell Access in
node0_RP0_CPU0
```

Example

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, this restricts the direct shell access.

Note

- To access the root shell after restricting shell access, see Access a restricted shell on page 529.

  If required, to terminate a pending Consent Token authorization request for shell-access restriction, see Terminate a pending Consent Token authorization request on page 528.

Unrestrict shell access using a Cisco-signed Consent Token

Remove the restricted-shell setting by completing the Consent Token authorization flow and restore the default non-gated root shell behavior.

Use this task to unrestrict the supported direct, interactive root shell access by using a Cisco-signed Consent Token.

Before you begin

Ensure that gated shell access is restricted. For more information about restricting shell access by using a Cisco-signed Consent Token, see Restrict shell access using a Cisco-signed Consent Token on page 524.

Procedure

1. Generate an unrestrict challenge on the router.

   Example

   ```
   RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge
   Thu Aug 20 06:06:33.888 UTC

   +-------------------------------------+
   Node location: node0_RP0_CPU0
   +-------------------------------------+
   Challenge string:
   <challenge_string>
   ```

2. Submit the challenge string to a Cisco TAC engineer. Cisco verifies that the requester is the legitimate device owner and is authorized to perform the requested action. If verified, the TAC engineer provides a signed response string.
3. Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Unrestrict Shell Access in
node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, the response removes the restriction on direct shell access.

> **Note**
>
> If required, to terminate a pending Consent Token handshake for shell-access restriction, see Terminate a pending Consent Token authorization request on page 528.

4. Verify the shell-access status.

Example

```
RP/0/RP0/CPU0:ios# show platform security shell-access status
Thu Aug 20 05:20:01.690 UTC
Restricted shell access enabled: No
```

Restrict shell access using an owner Consent Token

Restrict direct interactive root shell access on a supported Cisco IOS XR platform and store the restricted-shell selection in the Gated Shell Access HWTAM Secure Object.

Use this task to restrict supported direct, interactive root shell access by using an owner Consent Token.

Procedure

1. Generate the challenge string on the router.

Example

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge owner
Thu Aug 20 06:06:33.888 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
```

This produces a challenge string containing the device ID, a nonce, and the requested action.

2. To generate the response string owner key type , refer to Provisioning customer consent tokens.
3. Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example

```
RP/0/RP0/CPU0:ios# platform security shell-access restrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Restrict Shell Access in
node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, this restricts the direct shell access.

Note

- To access the root shell after restricting shell access, see Access a restricted shell on page 529.

  If required, to terminate a pending Consent Token authorization request for shell-access restriction, see Terminate a pending Consent Token authorization request on page 528.

Unrestrict shell access using an owner Consent Token

Remove the restricted-shell setting by completing the Consent Token authorization flow and restore the default non-gated root shell behavior.

Restore unrestricted root shell access by removing the restricted-shell setting, allowing access to advanced system features when necessary.

Use this task to unrestrict the supported direct, interactive root shell access by using an owner Consent Token.

Before you begin

Ensure that gated shell access is restricted. For more information about restricting shell access by using an owner Consent Token, see Restrict shell access using an owner Consent Token on page 526.

Procedure

1. Generate an unrestrict challenge on the router.

Example

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge
Thu Aug 20 06:06:33.888 UTC
+--------------------------------------+
Node location: node0_RP0_CPU0
+--------------------------------------+
```

```
Challenge string:
<challenge_string>
```

2. To generate the response string owner key type, refer to Provisioning customer consent tokens.
3. Paste the response string provided by the TAC engineer when prompted, to install the signed response on the router.

Example

```
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict response
Thu Aug 20 06:07:07.090 UTC
************************************************************
Please enter challenge response string for node location node0_RP0_CPU0
************************************************************
<response_string>
Successfully accepted challenge-response for Unrestrict Shell Access in
node0_RP0_CPU0
```

The router validates the signature and confirms that the device ID and nonce match its own records. If valid, the response removes the restriction on direct shell access.

> **Note**
>
> If required, to terminate a pending Consent Token handshake for shell-access restriction, see Terminate a pending Consent Token authorization request on page 528.

4. Verify the shell-access status.

Example

```
RP/0/RP0/CPU0:ios# show platform security shell-access status
Thu Aug 20 05:20:01.690 UTC
Restricted shell access enabled: No
```

Terminate a pending Consent Token authorization request

Terminates a pending Consent Token authorization request for restrict or unrestrict shell access.

Use this task to terminate a pending Consent Token authorization request for restrict or unrestrict shell access.

Use this task only after generating a Consent Token challenge and before submitting a response. The terminate command applies to both Cisco and owner Consent Token requests.

Procedure

1. Generate a Consent Token challenge for either restrict or unrestrict shell access, as described in the applicable task.

   Do not submit the response.

2. To terminate a pending Consent Token handshake:

- For restrict shell access, use `platform security shell-access restrict terminate-auth` command.

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios# platform security shell-access restrict challenge cisco
Thu Aug 20 06:08:45.228 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
RP/0/RP0/CPU0:ios# platform security shell-access restrict terminate-auth
Thu Aug 20 06:09:10.334 UTC
Successfully terminated challenge session for Restrict Shell Access in
node0_RP0_CPU0
```

- For unrestrict shell access, use `platform security shell-access unrestrict terminate-auth` command.

```
RP/0/RP0/CPU0:ios#
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict challenge
cisco
Thu Aug 20 06:08:45.228 UTC

+-------------------------------------+
Node location: node0_RP0_CPU0
+-------------------------------------+
Challenge string:
<challenge_string>
RP/0/RP0/CPU0:ios# platform security shell-access unrestrict terminate-auth
Thu Aug 20 06:09:10.334 UTC
Successfully terminated challenge session for Unrestrict Shell Access in
node0_RP0_CPU0
```

The system confirms that it terminated the pending authorization request.

Access a restricted shell

Describes how to access a restricted direct, interactive root shell by completing the Consent Token challenge-response workflow.

Use this task to access a restricted, direct, interactive root shell by completing the Consent Token challenge-response workflow.

This task applies to supported direct, interactive root shell commands, such as `bash`, `run` or `attach`.

Procedure

1. Restrict shell access by using either a Cisco-signed Consent Token or an Owner Consent Token.
2. Enter the `bash`, `run` or `attach` command. When prompted, complete the `cisco` or `owner` Consent Token challenge-response.

   A valid response permits the requested shell instance.

Example

```
RP/0/RP0/CPU0:ios# bash
Thu Aug 20 06:01:15.407 UTC
Restricted shell access is enabled. Please obtain a consent token to proceed.
Consent token challenge: <challenge_string>
Enter challenge response, or Ctrl-C to terminate:
<response_string>
Challenge response accepted successfully.
```

25 Implementing Lawful Intercept

Topics:

- Lawful intercept
- LI package installation and removal
- SNMPv3 access for lawful intercept
- Additional lawful intercept information

Describes the prerequisites, installation tasks, configuration tasks, operational behavior, and limitations for implementing lawful intercept.

Use this chapter to understand, install, configure, and operate lawful intercept on supported Cisco IOS XR systems.

Lawful intercept

Explains how lawful intercept uses SNMPv3 provisioning and mediation devices to deliver authorized communication intercepts to law enforcement agencies.

Lawful intercept is a capability that

- allows service providers to perform authorized surveillance on an individual or target
- uses RFC 3924 architecture and SNMPv3 provisioning to secure communication with the mediation device, and
- replicates intercepted communication for delivery to a law enforcement agency.

Table 116: Feature History Table

Feature Name	Release Information	Feature Description
Lawful intercept	Release 26.2.1	Introduced in this release on: Fixed Systems (8700 [ASIC: K100]) (select variants only*); Modular Systems (8800 [LC ASIC: K100]) (select variants only*) This feature requires egress feature capability on supported platforms. Use the **hw-module profile edge-mode** CLI command to enable the feature. The feature introduces these changes: CLI: • **hw-module profile edge-mode** • **show hw-module profile edge-mode** *This feature is now supported on: • 8711-48Z-M • 8712-MOD-M • 88-LC1-48Y8H-EM
Lawful intercept	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) *This feature is supported on Cisco 8404-SYS-D routers.
Lawful intercept	Release 25.4.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100])(select variants only*) *This feature is supported on: • 8011-32Y8L2H2FH • 8011-12G12X4Y-A/D
Lawful intercept	Release 25.1.1	Introduced in this release on: Fixed Systems (8010 [ASIC: A100]) (select variants only*) *This feature is supported on Cisco 8011-4G24Y4H-I routers.

Feature Name	Release Information	Feature Description
Lawful intercept	Release 24.4.1	Introduced in this release on: Fixed Systems(8200, 8700)(select variants only*); Modular Systems (8800 [LC ASIC: P100])(select variants only*). This feature is now enabled on the following hardware thus allowing service providers to perform surveillance on an individual (or target) as authorized by a judicial or administrative order and share the communication intercepts with law enforcement agencies. *This feature is now supported on: • 8212-48FH-M • 8711-32FH-M • 8712-MOD-M • 88-LC1-12TH24FH-E • 88-LC1-52Y8H-EM • 88-LC1-36EH
Lawful intercept	Release 24.2.1	You can now enable Lawful Intercept (LI) by installing and activating the LI package to enable service providers to perform surveillance on an individual (or target) as authorized by a judicial or administrative order and share the communication intercepts with law enforcement agencies. This feature is supported on Cisco 8800 series routers that have the 88-LC1-36EH line card installed.

Lawful intercept benefits

Describes the operational benefits that lawful intercept provides for authorized monitoring and packet delivery.

Use this topic to understand the operational benefits provided by lawful intercept.

Lawful intercept provides these benefits:

- Allows multiple law enforcement agencies to run a lawful intercept on the same router without each agency knowing about the others.
- Does not affect subscriber services on the router.
- Supports wiretaps in both the input and output directions.
- Supports wiretaps of Layer 3 traffic.
- Cannot be detected by the target.

Lawful intercept prerequisites

Lists the router, mediation device, management-plane, and SNMP prerequisites for lawful intercept.

Use this topic to identify the requirements that must be met before implementing lawful intercept.

Meet these prerequisites before implementing lawful intercept:

- Use the router as the content Intercept Access Point (IAP) router in the lawful interception operation.
- Provision the router before configuring lawful intercept.
- Use a loopback interface for lawful intercept taps when possible. A loopback interface provides advantages over other interface types.
- Configure the management plane to enable SNMPv3 so that the mediation device can communicate with the router through a physical interface, preferably a loopback interface.
- Enable View-based Access Control Model (VACM) views for the SNMP server.
- Provision the mediation device. For detailed mediation device information, see the documentation for the mediation device vendor.
- Use CISCO-TAP2-MIB to set up communication between the router acting as the content IAP and the mediation device.
- Use CISCO-IP-TAP-MIB to set up filters for the IP addresses and port numbers to be intercepted.
- Ensure that the mediation device is reachable from the content IAP router through the global routing table, not through a VRF routing table.

Lawful intercept topology

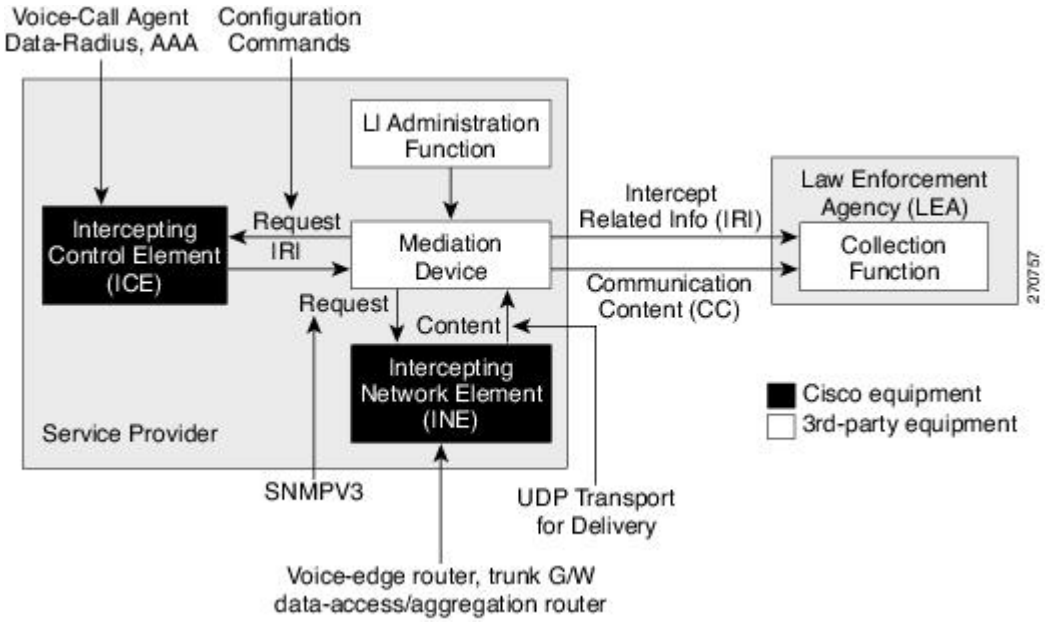
Describes the intercept access points, interfaces, and control elements used for voice and data interception.

Use this topic to understand the topology components used for voice and data interception.

The lawful intercept topology contains intercept access points and interfaces for both voice and data interception.

- The router is used as the content Intercept Access Point (IAP) router, or the Intercepting Network Element (INE), in the lawful interception operation.
- The Intercepting Control Element (ICE) can be Cisco equipment or third-party equipment.

Figure 34: Lawful intercept topology for both voice and data interception



Lawful intercept restrictions

Provides packet, mediation device, tap, interface, fragmentation, and feature limitations for lawful intercept.

Use this topic to identify the limitations that apply to lawful intercept.

These restrictions apply to lawful intercept:

- Deployment and configuration restrictions include:
  - Lawful intercept supports only pure IP over Ethernet packets.
  - A maximum of 512 mediation devices, 1024 IPv4 taps, and 512 IPv6 taps are supported.
  - One tap to multiple mediation devices is not supported.
  - After a route processor reload or failover, re-provision the mediation device and tap configuration.
  - Both IPv4 and IPv6 mediation devices are supported. Resolve ARP for the path to the mediation device and for the mediation device next hop.
  - Remove the SNMP configuration while deactivating the LI RPM.
- Packet and interface restrictions include:
  - LI statistics are not supported.
  - Although original packets can be fragmented, LI packets cannot be fragmented. Ensure that the egress interface MTU to the mediation device supports the captured packet size.
  - LI supports Layer 3 taps for Layer 3 interface types, including physical and bundle interfaces, but LI traffic cannot be transmitted over the management interface.
- Unsupported LI features include:
  - IPv4 or IPv6 multicast tapping.
  - Per-interface tapping.
  - Tagged packet tapping.
  - Replicating one tap to multiple mediation devices.
  - Layer 2 flow tapping.
  - SRv6 traffic.
  - RTP encapsulation.
  - Using LI and SPAN on the same interface.

Restrictions on configuring LI with other features

Describes configuration limits that apply when LI shares ACL-dependent resources with other router features.

Use this topic to understand the restrictions that apply when configuring LI with other ACL-dependent features.

Apply these restrictions when configuring LI with ACL-dependent features such as BGP Flow Specification (BGPFS), QoS-ACL, Security SPAN ACL, QoS Policy Propagation via BGP (QPPB), Policy-Based Routing (PBR), Peering QoS, and Layer 2 ACL for packets with Layer 3 payloads:

- If LI is the first feature configured on the system, add up to three additional ACL-dependent features at the interface or global level. Configuring more features generates an IOS message for features such as QPPB, QoS-ACL, Security-ACL, and BGPFS.

- If LI is already enabled on the router, configuring SPAN with an ACL on an interface causes the configuration to be rejected.
- If three ACL-dependent features are already configured on interfaces when the LI RPM is installed, LI attempts to tap ingress traffic on all interfaces.
- If four ACL-dependent features are configured on specific interfaces while other interfaces have fewer than three, LI taps traffic on the interfaces with fewer than three features and bypasses the interfaces with four features.

LI package installation and removal

Describes the package security controls and directs you to the installation and removal procedures for LI.

As LI is not a part of the Cisco IOS XR image by default, you need to install it separately.

Install and activate the LI package

Installs and activates the optional LI RPM package, verifies each installation operation, and confirms that the package is active.

Install the optional LI feature as it is not included in the Cisco IOS XR image.

Before you begin

Use SCP to transfer the LI RPM files to a folder on the router.

Procedure

1. Use SCP (secure copy) to transfer the RPM files to a folder on the router.
2. Start the LI RPM installation.

   Example

   ```
   Router# install source <location-of-the-RPM-package-file> xr-8000-li xr-li
   noprompt
   ```

   For example:

   ```
   Router# install source /harddisk:/li-rpms/optional-rpms-argon xr-8000-li xr-li
    noprompt
   ```

3. Verify the progress of the RPM installation.

   Example
   RPM Installation (in progress)

   ```
   Router#show install request verbose

   User request: install source file:///harddisk:/li-rpms/optional-rpms-argon
   xr-8000-li xr-li
   Operation ID: 1.1
   State:        In progress since 2023-04-11 17:58:30 UTC

   Current activity:    Verify input and download to internal repository if
   needed
   Next activity:       Veto check
   ```

```
Time started:        2023-04-11 17:58:44 UTC
No per-location information.
```

Example: RPM Installation (complete)

```
Router#show install request verbose

User request: install source file:///harddisk:/li-rpms/optional-rpms-argon
xr-8000-li xr-li
Operation ID: 1.1
State:        Success since 2023-04-11 18:00:46 UTC

Current activity:    Await user input
Time started:        2023-04-11 18:00:46 UTC

The following actions are available:
    install package add
    install package remove
    install package upgrade
    install package downgrade
    install package replace
    install package rollback
    install replace
    install rollback
    install source
    install commit
    install replace reimage
```

The output displays the operation identifier, state, current activity, next activity, and start time.

4. Complete the LI package installation.

Example

```
Router# install commit
Install commit operation 1 has started
Install operation will continue in the background
```

5. Verify the progress of the installation commit.

Example

```
Router# show install request verbose
User request: install commit
Operation ID: 1
State:        In progress since 2023-04-11 18:00:57 UTC

Current activity:    Commit transaction
Next activity:       Transaction complete
Time started:        2023-04-11 18:00:57 UTC

No per-location information.
RP/0/RP0/CPU0:ios#show install request verbose
Tue Apr 11 18:09:57.666 UTC

User request: install commit
Operation ID: 1
State:        Success since 2023-04-11 18:01:36 UTC
```

```
Current activity:    No install operation in progress

The following actions are available:
    install package add
    install package remove
    install package upgrade
    install package downgrade
    install package replace
    install package rollback
    install replace
    install rollback
    install source
    install replace reimage
```

6. Verify that the LI package is active.

Example

```
Router# show install active summary
Active Packages:    XR: 208    All: 1325
Label:              7.11.1.01I
Software Hash:
f73f9988276c2001702066ac8173de2fe5b9501849ca2c98d49f9828b4ab7522

Optional Packages
Version
--------------------------------------------------
---------------------------
xr-8000-li
7.11.1.01Iv1.0.0-1

...
xr-li                                                    7.11.1.01Iv1.0.0-1
..
```

The LI package is installed and listed among the active packages.

Uninstall the LI package

Removes the LI RPM package, applies the least impactful restart method, and verifies the package removal.

Remove the LI package from the router.

Procedure

1. Start the LI package removal.

Example

```
Router# install package remove xr-8000-li xr-li
Install remove operation 2.1.1 has started
Install operation will continue in the background
```

2. Verify the progress of the uninstallation.

Example

```
Uninstallation (in progress)
Router# show install request verbose
User request: install package remove xr-8000-li xr-li
Operation ID: 2.1.1
State:        In progress since 2023-04-11 18:18:05 UTC

Current activity:    Package add or other package operation
Next activity:       Await user input
Time started:        2023-04-11 18:18:44 UTC
Timeout in:          39m 34s
Locations responded: 0/1

Location 0/RP0/CPU0:
  Packaging operation stage: Package operations - completed 6/6
  No client notifications waiting
```

```
Uninstallation (complete)

Router#show install request verbose

User request: install package remove xr-8000-li xr-li
Operation ID: 2.1.1
State:        Success since 2023-04-11 18:19:37 UTC

Current activity:    Await user input
Time started:        2023-04-11 18:19:37 UTC

The following actions are available:
    install package add
    install package remove
    install package upgrade
    install package downgrade
    install package abort latest
    install package abort all-since-apply
    install apply restart
    install apply reload
    install replace reimage

Least impactful apply method: install apply restart
```

3. Apply the LI package uninstallation with the least impactful restart method and verify the progress.

   The available apply methods include restart and reload. The restart method is the least impactful method.

   Example

```
Router# install apply restart
Install apply operation 2.1 has started
Install operation will continue in the background
```

```
Apply restart (in progress)

Router#show install request verbose
Tue Apr 11 18:22:04.298 UTC

User request: install apply restart
Operation ID: 2.1
State:        In progress since 2023-04-11 18:20:44 UTC
```

```
Current activity:    Post-apply operation cleanup
Next activity:       Await user input
Time started:        2023-04-11 18:21:07 UTC

No per-location information.
```

```
Apply restart (complete)

Router#show install request verbose
Tue Apr 11 18:22:21.102 UTC

User request: install apply restart
Operation ID: 2.1
State:        Success since 2023-04-11 18:22:09 UTC

Current activity:    Await user input
Time started:        2023-04-11 18:22:09 UTC

The following actions are available:
    install package add
    install package remove
    install package upgrade
    install package downgrade
    install package replace
    install package rollback
    install replace
    install rollback
    install source
    install commit
    install replace reimage
```

4. Complete the LI package uninstallation.

   Example

```
Router# install commit
Install commit operation 2 has started
Install operation will continue in the background
```

5. Verify the LI package uninstallation.

```
Router# show install request verbose
User request: install commit
Operation ID: 2
State:        Success since 2023-04-11 18:23:35 UTC

Current activity:    No install operation in progress

The following actions are available:
    install package add
    install package remove
    install package upgrade
    install package downgrade
    install package replace
    install package rollback
    install replace
    install rollback
    install source
    install replace reimage
```

```
Router# show install active summary
Active Packages:    XR: 206    All: 1323
Label:              7.11.1.01I
Software Hash:
cdef36e9cf22ba3a1c1217d7b05b41a45b7e9ff6e8f30d871a1c52b91bf16047

Optional Packages
Version
--------------------------------------------------
--------------------------
xr-8000-l2mcast
7.11.1.01Iv1.0.0-1
xr-8000-mcast
7.11.1.01Iv1.0.0-1
xr-8000-netflow
7.11.1.01Iv1.0.0-1
xr-bgp
7.11.1.01Iv1.0.0-1
xr-ipsla
7.11.1.01Iv1.0.0-1
xr-is-is
7.11.1.01Iv1.0.0-1
xr-lldp
7.11.1.01Iv1.0.0-1
xr-mcast
7.11.1.01Iv1.0.0-1
xr-mpls-oam
7.11.1.01Iv1.0.0-1
xr-netflow
7.11.1.01Iv1.0.0-1
xr-ops-script-repo
7.11.1.01Iv1.0.0-1
xr-ospf
7.11.1.01Iv1.0.0-1
xr-perf-meas
7.11.1.01Iv1.0.0-1
xr-perfmgmt
7.11.1.01Iv1.0.0-1
xr-track
7.11.1.01Iv1.0.0-1
```

The LI package is removed and no longer appears in the active package summary.

SNMPv3 access for lawful intercept

Groups the procedures that disable SNMP-based lawful intercept, configure inband management-plane protection, and enable the LI SNMP server configuration.

Use this topic to understand the SNMPv3 procedures that support lawful intercept.

Disable SNMP-based lawful intercept

Disables SNMP-based lawful intercept and drops all SNMP-based taps from the router.

Lawful Intercept is enabled by default on the router after installing and activating the LI RPM package.

Before you begin

The **lawful-intercept disable** command is available only after the LI RPM package is installed and activated.

Procedure

Disable lawful intercept from global configuration mode.

Example

```
Router# configure
Router(config)# lawful-intercept disable
Router(config)# commit
```

SNMP-based lawful intercept is disabled and all SNMP-based taps are dropped. Use the **no** form of the command to re-enable lawful intercept.

Configure inband management plane protection for LI

Configures inband Management Plane Protection to allow SNMP commands from the mediation device on the selected interface.

Configure the Management Plane Protection (MPP) feature for SNMP communication with the mediation device.

MPP is disabled by default. If MPP is not already configured for another protocol, configure it for SNMP communication with the mediation device. Configure an inband interface, such as a loopback interface, to allow SNMP commands to reach the router.

Procedure

1. Configure inband SNMP access on loopback0 inband interface and commit the configuration.

   Example

   ```
   Router# configure
   Router(config)# control-plane
   Router(config-ctrl)# management-plane
   ```

```
Router(config-mpp)# inband
Router(config-mpp-inband)# interface loopback0
Router(config-mpp-inband-Loopback0)# allow snmp
Router(config-mpp-inband-Loopback0)# commit
```

2. Verify the inband SNMP configuration.

Example

```
Router# show mgmt-plane inband interface loopback0
Management Plane Protection - inband interface
interface - Loopback0
snmp configured - All peers allowed
```

The inband management-plane configuration allows SNMP on loopback0. The verification output identifies the interface and confirms that SNMP is configured.

Enable the lawful intercept SNMP server configuration

Enables the SNMP server configuration that allows the mediation device to request interception of data sessions.

Configure the SNMP engine, host, user, views, and group required for lawful intercept.

Before you begin

Perform this task even if you recently upgraded to Cisco IOS XR Software from Cisco IOS and had MPP configured for a protocol.

If you use a loopback interface for SNMP messages, include that interface in the inband management configuration.

Procedure

---

1. Configure the local SNMP engine ID.

Example

```
Router(config)# snmp-server engineID local 00:00:00:09:00:00:00:a1:61:6c:20:56
```

2. Configure the SNMPv3 host.

Example

```
Router(config)# snmp-server host 1.75.55.1 traps version 3 priv
sample-user-name udp-port 4444
```

3. Create the SNMPv3 user.

Example

```
Router(config)# snmp-server user sample-user-name sample-li-group v3 auth md5
 clear lab priv des56 clear lab
```

4. Create the lawful intercept SNMP views.

Example

```
Router(config)# snmp-server view sample-li-view ciscoTap2MIB included
Router(config)# snmp-server view sample-li-view ciscoIpTapMIB included
Router(config)# snmp-server view sample-li-view snmp included
Router(config)# snmp-server view sample-li-view ifMIB included
Router(config)# snmp-server view sample-li-view 1.3.6.1.6.3.1.1.4.1 included
```

5. Configure the SNMPv3 group.

Example

```
Router(config)# snmp-server group sample-li-group v3 auth read sample-li-view
 write sample-li-view notify sample-li-view
Router(config)# commit
```

The SNMP server is configured to support lawful intercept communication with the mediation device.

Additional lawful intercept information

Groups reference information about interception mode, data flow, scale, packet interception, filters, encapsulation, and high availability.

Use this topic to locate operational information about lawful intercept after the feature is installed and configured.

Interception mode

Describes the global LI mode and the tap placement used to intercept traffic regardless of the ingress point.

Lawful intercept operates in Global LI mode.

In Global LI mode, taps are installed on all line cards in the ingress direction. A global tap can intercept target traffic regardless of the ingress point. Only a tap with wildcards in the interface field is supported.

How lawful intercept data flows

Describes how the mediation device requests an intercept, how the router replicates the intercepted data, and how the mediation device delivers it to the collection function.

The router and mediation device exchange lawful intercept information through SNMPv3.

Summary

The process includes these components:

- The mediation device initiates communication content intercept requests.
- The content IAP router intercepts and replicates the communication content.
- The mediation device sends intercepted data to the law enforcement agency collection function.

Workflow

These stages describe how intercepted data flows:

1. The mediation device uses SNMPv3 to initiate communication content intercept requests to the content IAP router.

2. The content IAP router intercepts and replicates the communication content, then sends it to the mediation device in IPv4 or IPv6 UDP format.
3. The mediation device sends intercepted data sessions to the law enforcement agency collection function using a supported lawful intercept delivery standard.

Result

The authorized communication content reaches the mediation device and the law enforcement agency collection function.

Mediation device responsibilities

Lists the mediation device responsibilities for activating, removing, and auditing authorized intercepts.

The mediation device performs these tasks:

- Activates the intercept at the authorized time.
- Removes the intercept when the authorized time period expires.
- Periodically audits network elements to ensure that only authorized intercepts are in place and that all authorized intercepts are in place.

Lawful intercept scale and performance values

Provides the supported lawful intercept scale values for IPv4 and IPv6 intercepts.

The router supports these lawful intercept scalability values:

- A maximum of 1024 IPv4 intercepts.
- A maximum of 512 IPv6 intercepts.

Intercept IPv4 and IPv6 packets

Describes the supported filters and UDP encapsulation used to intercept and forward IPv4 and IPv6 packets.

Use this topic to understand how lawful intercept handles IPv4 and IPv6 packets.

Lawful intercept supports the interception of IPv4 and IPv6 packets. The tap filters classify packets, and the router replicates and encapsulates matching packets before forwarding them to the mediation device.

Use these filters to classify a tap:

- IP address type
- Destination IP address
- Destination mask
- Source IP address
- Source mask
- Type of Service (ToS) and ToS mask
- Layer 4 protocol
- Layer 4 destination port with range
- Layer 4 source port with range

VPN Routing and Forwarding (VRF), flow-id, and interface filters are not supported.

Encapsulation types supported for intercepted packets

Describes the IPv4 and IPv6 UDP encapsulation applied to replicated packets before they are forwarded to the mediation device.

Use this topic to identify the encapsulation types supported for intercepted packets.

The router replicates and encapsulates intercepted packets before forwarding them to the mediation device.

- IPv4 packets use IPv4 UDP encapsulation.
- IPv6 packets use IPv6 UDP encapsulation.
- The replicated packet receives a new UDP header and a new IPv4 or IPv6 header, depending on the packet type.
- The IP header information is derived from the mediation device configuration.
- A 4-byte channel identifier (CCCID) is inserted after the UDP header.
- The router does not support forwarding the same replicated packet to multiple mediation devices.

RTP and RTP-NOR encapsulation types are not supported.

How high availability preserves lawful intercept flows

Describes how lawful intercept high availability reduces information loss during route processor failover and how mediation devices restore the required tables.

High availability preserves operational continuity for tap flows and provisioned mediation device tables during route processor failover (RPFO).

Summary

The high-availability process depends on these actions:

- The mediation device detects loss of taps through the SNMP configuration process.
- The mediation device re-provisions the stream, mediation device, and IP tap table entries.
- The replay timer provides time for re-provisioning while existing tap flows continue.

Workflow

These stages describe lawful intercept behavior during route processor failover:

1. When RPFO is detected, the replay timer starts on the active route processor.
2. The mediation device detects the loss of taps and re-provisions all rows relating to CISCO-TAP2-MIB and CISCO-IP-TAP-MIB to synchronize the database view across the route processor and mediation device.
3. Existing taps continue to flow while the mediation device re-provisions the entries within the replay interval.
4. Interception stops on taps that are not re-provisioned before the replay timer expires.

Result

The mediation device restores the lawful intercept tables and minimizes interruption to existing tap flows.

Preserve TAP and MD tables during route processor failover

Describes how the mediation device restores TAP and mediation device table entries after route processor failover.

After RPFO, the mediation device must re-provision the entries required to synchronize its database view with the route processor.

Summary

The mediation device re-provisions the stream tables, mediation device tables, and IP taps with the same values used before failover.

Workflow

These conditions apply after RPFO:

1. Rows that are not re-provisioned return **NO_SUCH_INSTANCE** for an SNMP Get operation.
2. Create an entire row in one configuration step with the same values used before RPFO and with **rowStatus** set to **CreateAndGo**.
3. Set the **cTap2MediationTimeOut** object to a valid future time.

Result

Existing taps continue to flow when their entries are re-provisioned within the replay interval.

Replay timer

Describes the internal timeout that gives the mediation device time to re-provision tap entries after route processor failover.

The replay timer is an internal timeout that

- provides time for the mediation device to re-provision lawful intercept (LI) tap entries while existing tap flows continue
- starts and resets on the active route processor when route processor failover (RPFO) occurs, and
- has a value based on the number of LI entries on the router, with a minimum value of 10 minutes.

Replay timer behavior after failover

Interception stops on taps that are not re-provisioned before the replay timer expires.

If high availability is not required, the mediation device waits for entries to age out after failover. The mediation device cannot change an entry after the replay timer expires; it can reinstall taps as they are or wait for the entries to age out.

26 EST Protocol for Automated Certificate Provisioning

Topics:

- EST protocol for automated certificate provisioning
- EST protocol configuration guidelines and limitations
- Certificate types and their uses in the EST protocol
- Trust anchor databases and their uses in the EST protocol
- EST message types
- Configure the EST protocol

Describes the EST protocol, its certificate provisioning capabilities, configuration requirements, authentication options, and configuration procedure.

Use this chapter to understand and configure Enrollment over Secure Transport (EST) for automated certificate provisioning.

EST protocol for automated certificate provisioning

Explains how Enrollment over Secure Transport uses TLS to secure certificate provisioning and automate certificate renewal for configured trustpoints.

Enrollment over Secure Transport (EST) is a digital certificate provisioning protocol that

- uses TLS to secure certificate provisioning communication
- supports designated certificate requestors, and
- automates certificate renewal.

Table 117: Feature History Table

Feature Name	Release Information	Feature Description
EST protocol for automated certificate provisioning	Release 26.1.1	Introduced in this release on: Centralized Systems (8400 [ASIC: K100])(select variants only*) * This feature is now supported on Cisco 8404-SYS-D routers.
EST protocol for automated certificate provisioning	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC: Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) This release introduces support for Enrollment Over Secure Transport (EST), a digital certificate provisioning protocol, which enhances certificate management by offering secure transport using TLS and designated certificate requestors. It enables automated certificate renewal. EST is an enhancement of the existing Simple Certificate Enrollment Protocol (SCEP), providing improved security and flexibility for certificate management operations over both IPv4 and IPv6 networks. The feature introduces these changes: CLI: • The **method-est** keyword is introduced in the **crypto ca trustpoint** command. • The **client-authentication** command is introduced. • The **SSL-profile** command is introduced. YANG Data Model: • `Cisco-IOS-XR-crypto-cepki-cfg` • `Cisco-IOS-XR-um-crypto-cfg` (see GitHub, YANG Data Models Navigator)

EST protocol features and benefits

Describes the secure transport, designated requestor, and automated renewal capabilities provided by EST.

Use this topic to understand the security and automation benefits provided by EST.

EST provides these key features and benefits:

- Secure transport: EST uses TLS to secure messages and certificates without additional encapsulation.
- Designated certificate requestors: EST associates a certificate signing request (CSR) with a specific trusted requestor authenticated with TLS.
- Automated certificate renewal: EST supports automatic re-enrollment for seamless certificate renewal.

EST protocol components and terminology

Defines the certificate management protocols, certificate signing request, public key infrastructure, and message syntax terms used with EST.

Identify the components and terms used in the EST protocol.

These terms describe components related to the EST protocol:

- Certificate Management Protocol (CMP): A protocol with comprehensive capabilities for managing digital certificates, including enrollment, renewal, and revocation.
- Certificate signing request (CSR): A message that the client sends to the CA to request a digital certificate. The CSR includes the client public key and identifying information.
- PKCS 7: A standard for cryptographic message syntax that SCEP uses to encapsulate messages.
- Public Key Infrastructure (PKI): The framework that manages digital certificates and keys for certificate provisioning operations through EST.
- Simple Certificate Enrollment Protocol (SCEP): An older certificate provisioning protocol that relies on HTTP and PKCS 7 to secure messages and does not provide some of the modern security features of EST.

EST client authentication options

Describes the TLS certificate-based and HTTP-based methods that an EST client can use for initial enrollment and re-enrollment.

Use this topic to understand the authentication options available to an EST client.

EST supports these client authentication options:

- TLS certificate-based authentication uses a valid certificate chain signed by a CA in the EST server trust store. The server verifies the client during mutual TLS (mTLS) authentication. The bootstrap certificate is used for first enrollment, and the certificate issued during initial enrollment is used for re-enrollment.
- HTTP-based authentication establishes a TLS session and sends HTTP authentication credentials when a bootstrap certificate is unavailable. Type 6 encryption protects passwords configured on the router. Type 6 encryption is disabled by default and must be enabled manually. A master key must also be created and stored in the Trust Anchor Module (TAM).

For HTTP-based authentication, the client sends a username and clear-text password through the TLS-encrypted channel. The device validates the EST server by using the server certificate CA or subordinate CA chain installed locally.

EST protocol configuration guidelines and limitations

Provides key type, authentication, TLS validation, re-enrollment, and trustpoint guidelines for configuring EST.

Review these guidelines and limitations before configuring the EST protocol.

Supported key types

Identifies the key type supported for signing the CSR during EST enrollment.

In Cisco IOS XR release 25.1.1, RSA keys are supported for signing the Certificate Signing Request (CSR) during enrollment. ECDSA keys are not supported.

Client authentication methods

Compares the TLS certificate-based and HTTP-based authentication methods supported by the EST client.

The EST client supports both TLS certificate-based authentication and HTTP-based authentication. Cisco recommends TLS certificate-based authentication according to RFC 7030, although both methods are available.

TLS validation and authentication requirements

Describes how certificate validation failures affect EST enrollment and the absence of fallback to HTTP authentication.

If client or server certificate validation fails during the TLS handshake, EST enrollment fails. EST does not switch to HTTP authentication when TLS certificate-based client authentication fails.

EST re-enrollment

Describes certificate reuse and profile requirements for EST re-enrollment when the server uses different credentials for initial enrollment.

For re-enrollment, the client always uses the previously issued certificate to establish the TLS connection.

A re-enrollment profile is required when the EST server uses a fixed username and password for re-enrollment but uses a one-time password (OTP) for initial enrollment. This behavior is optional and depends on the EST server configuration.

EST client support for multiple trustpoints

Describes why multiple trustpoints are required to support secure EST enrollment for all trustpoints configured on the device.

EST client support requires multiple trustpoints because it enables secure certificate enrollment over TLS for all trustpoints configured on the device. A single trustpoint configuration is insufficient to support this functionality.

Certificate types and their uses in the EST protocol

Maps certificate types and issuers to their uses during EST authentication and certificate provisioning.

Use this topic to identify certificate types, issuers, and uses in the EST protocol.

Table 118: Certificate types and their uses

If the certificate type is...	And the issuer is...	Then it is used for ...
EST server certificate	EST server	authenticating the CA during the TLS handshake when the EST server interacts with clients.
EST client certificate	EST client	authenticating to the EST server during certificate enrollment operations.
End entity (leaf certificate)	EST server	non-EST uses, such as authenticating devices in different contexts beyond EST operations.
EST server certificate	third-party CA	providing authentication for the EST server by a CA that is not part of the EST infrastructure but is trusted as a third party.
third-party client certificate	third-party CA	authenticating the EST client to the EST server for initial interactions before enrollment.

Trust anchor databases and their uses in the EST protocol

Describes the trust anchor databases used to validate EST server and client certificates during enrollment and re-enrollment.

Use this topic to identify the trust anchor databases used during EST authentication.

A trust anchor (TA) database is a repository of trusted anchor certificates used to verify the authenticity of a Certification Authority (CA) during certificate enrollment. Use this table to identify each TA database and its purpose.

Table 119: Trust anchor databases and their uses

If the TA database type is...	Then it is used by ...	For the purpose of...
EST server	EST server	authenticating certificates issued by the EST CA during enroll and re-enroll operations.
EST server (implicit)	EST server	authenticating certificates issued by the EST CA; can be disabled if not needed.
EST client	EST client	authenticating EST server certificates issued by the EST CA.
EST client (implicit)	EST client	authenticating an EST server using an externally issued certificate; can be disabled if unnecessary.

EST message types

Describes the predefined EST messages used to obtain CA certificates, enroll certificates, and renew certificates.

Use this topic to identify the predefined messages used for EST certificate provisioning.

The EST protocol includes these predefined message types:

- CA Certs: Requests CA certificates so that the client can validate the CA authenticity before certificate enrollment.
- SimpleEnroll: Performs simple certificate enrollment. The client sends a CSR to the CA, and the CA issues a certificate when the request is valid.
- Reenroll: Requests certificate renewal or reissuance so that the client can obtain a certificate with updated validity or information before an existing certificate expires.

Configure the EST protocol

Configures EST authentication, PKI trustpoints, certificate enrollment, and certificate verification for secure automated provisioning.

Configure the EST protocol on the router to enable secure and automated certificate provisioning.

Before you begin

Before configuring EST, ensure that:

- Access to the EST server is established and operational.
- The certificates and keys required for configuration are available.
- For HTTP-based authentication, create an RSA key with the **crypto key generate rsa key** command and use the generated key to configure the EST trustpoint.
- Enable Type 6 encryption with the **password6 encryption aes** command for secure password handling.
- Create a master key and store it in the Trust Anchor Module (TAM) with the **key config-key password-encryption** command.

A master key for Type 6 encryption is stored in the device internal memory in a protected area known as the TAM.

Procedure

1. Select the authentication method.

   Configure TLS certificate-based authentication, HTTP-based authentication, or both. Use this table to select the appropriate substep.

   Table 120: Bootstrap certificate and authentication method

If...	Then follow ...
A bootstrap certificate is not available	HTTP-based authentication substep.
A bootstrap certificate is available	TLS-based authentication substep.

   a) For HTTP-based authentication, create client authentication profiles for enrollment and re-enrollment.

   Example

```
Router(config)# client-authentication profile sample-P1_enroll
Router(config-client-authentication-profile)# username sample-estuser
password Encrypt6 sample-estpwd_enrol
Router(config-client-authentication-profile)# exit
Router(config)# client-authentication profile sample-P2_re-enroll
Router(config-client-authentication-profile)# username sample-estuser
```

```
password Encrypt6 sample-estpwd_re_enrol
Router(config-client-authentication-profile)# commit
Router(config-client-authentication-profile)# exit
Router(config)# crypto ca trustpoint sample-EST_TRUSTPOINT
Router(config-trustp)# method est
Router(config-trustpoint)# enrollment url https://192.168.30.1:port
Router(config-trustpoint)# authentication-profile sample-P1_enroll
Router(config-trustpoint)# re-enrollment authentication-profile
sample-P2_re-enroll
Router(config-trustpoint)# rsakeypair sample-est-client
Router(config-trustpoint)# commit
```

You can use IPv4, IPv6, or hostname addresses as the enrollment URL.

Table 121: HTTP-based re-enrollment

If...	Then...
Separate profiles are configured for enrollment and re-enrollment	Use the enrollment profile for enrollment and the re-enrollment profile for re-enrollment.
Only an enrollment profile is configured	Use the enrollment profile for both enrollment and re-enrollment.

If you use a multi-tier CA certificate system, import the CA root certificate into another trustpoint using the terminal, SCEP, or local file method.

b) For TLS-based authentication, configure an SSL profile with the bootstrap certificate.

Example

```
Router# configure terminal
Router(config)# ssl profile sample-EST_SSL_profile
Router(config-SSL-profile)# certificate sample-EST_BOOTSTRAP_TP
Router(config-SSL-profile)# commit
```

For initial enrollment, add a bootstrap certificate using an existing method such as terminal or SCEP.

2. Configure a PKI trustpoint with EST and attach the authentication profiles.

Example

```
Router# configure terminal
Router(config)# crypto ca trustpoint sample-EST-TP
Router(config)# ssl-profile sample-EST_enroll
Router(config-trustp)# method est
Router(config-method-est)# commit
Router(config-trustp)# enrollment authentication-profile sample-P1_enroll
Router(config-trustp)# re-enrollment authentication-profile sample-P2_re-enroll
Router(config-trustp)# commit
```

The **method est** command initializes EST for the specified trustpoint. The example attaches both HTTP-based and TLS-based authentication profiles.

3. Authenticate and enroll the certificates.

Example

```
Router# crypto ca authenticate sample-EST_TP
Router# crypto ca enroll sample-EST_TP
```

4. Verify that certificate enrollment through EST is successful.

Example

```
Router# show run crypto ca trustpoint sample-EST_TP
crypto ca trustpoint sample-EST-TP
 ssl-profile sample-EST_enroll
 method est
 !
 enrollment authentication-profile sample-P1_enroll
 re-enrollment authentication-profile sample-P2_re-enroll
!
```

The output displays the trustpoint, SSL profile, EST method, and enrollment and re-enrollment authentication profiles.

5. Verify that the certificates are enrolled on the trustpoint.

Example

```
Router# show crypto ca certificates sample-EST_TP
Trustpoint : EST_TP
CA certificate
Serial Number : 10:01
Subject:
CN=SUB_CA_CERT,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
Issued By:
CN=TWO-LEVEL-CA,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
Validity Start : 12:31:40 UTC Sun Jun 14 2020
Validity End : 12:31:40 UTC Wed Jun 12 2030
CRL Distribution Point
http://10.105.236.78/crl.der
SHA1 Fingerprint:
D8E0C11ECED96F67FDBC800DB6A126676A76BD62
Trusted Certificate Chain
Serial Number : 0F:A0:06:7A:C9:5E:A9:E7:61:A2:B9:2B:27:D1:D6:8F:3D:51:43:3B
Router certificate
Key usage : General Purpose
Status : Available
Serial Number : 28:E5
Subject:
CN=test
Associated Trustpoint: EST_TP
```

Verify the CA certificate, trusted certificate chain, router certificate, validity period, certificate fingerprints, and associated trustpoint.

The router uses EST to authenticate the server, enroll certificates, and verify the resulting certificate chain on the configured trustpoint.

27 Upgrade CURL libraries for HTTP client

Topics:

- Key benefits for libcurl upgrades
- Support and restrictions for libcurl upgrade
- How libcurl upgrade works

Explains how libcurl upgrades enhance security, compatibility, and protocol support by updating the HTTP client infrastructure with the latest libcurl features and fixes.

A libcurl upgrade is a feature that

- updates the embedded libcurl library from version 7.30 to 8.17.0 to align with upstream security and protocol improvements,
- preserves specific customizations such as VRF awareness, XR SSL integrations, and TCP performance tuning, and
- ensures backward compatibility and automatic benefit for all dependent applications without user intervention.

Table 122: Feature History Table

Feature Name	Release Information	Feature Description
Upgrade CURL libraries for HTTP client	Release 26.3.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q100, Q200, P100], 8700[ASIC: P100, K100], 8010 [ASIC: A100])(select variants only*); Centralized Systems(8600 [ASIC:Q200]) ; Modular Systems (8800 [LC ASIC: Q100, Q200, P100]). You gain enhanced security and improved reliability through an upgraded embedded libcurl library. This transparent update preserves existing functionality and continues to support HTTP/1.0 and HTTP/1.1.

Key benefits for libcurl upgrades

Describes the key attributes, compatibility considerations, preserved patches, supported features, and operational behavior of the libcurl upgrade.

The following are significant attributes and considerations for the libcurl upgrade:

- Updates the libcurl version to 8.17.0 for the underlying HTTP stack.
- Maintains compatibility for all existing APIs, so no changes are required for dependent applications.
- Retains and independently manages platform-specific patches, such as VRF, SSL, and TCP optimizations, by rebasing and validating them against the upgraded libcurl version, simplifying future maintenance.
- Supports major features such as Smart Licensing, Shell utility (Copy), Install Manager, CEPKI, SKS, and Zero Touch Provisioning (ZTP).
- Is not user-configurable and cannot be disabled; no new CLI or configuration is introduced.

Table 123: Libcurl upgrade attribute comparison

Attribute	Previous implementation	Upgraded implementation
Libcurl version	7.30	8.17.0
Supported HTTP protocols	HTTP, HTTPS	HTTP, HTTPS
XR-specific customizations	VRF support, SSL integration (XR libcrypto), TCP tuning	Maintained
API compatibility	Legacy APIs supported	All existing APIs maintained
Configuration impact	None	None (not configurable)

Note

The upgrade is completely transparent to users and feature owners. No CLI changes or configuration are required, and the previous libcurl version cannot be restored by configuration.

Support and restrictions for libcurl upgrade

Lists the supported Cisco platforms, associated restrictions, and key characteristics of the libcurl upgrade.

Provides an overview of the Cisco platforms that support the libcurl upgrade, details operational or configuration restrictions, and explains how the enhancement integrates with system infrastructure in supported environments.

The following points summarize platform support and key restrictions for the libcurl upgrade:

- The upgrade applies only to platforms that run Cisco IOS XR software, including supported hardware and virtual platforms that use the upgraded HTTP infrastructure.
- Users cannot enable, configure, or roll back the upgrade. The system image manages the upgrade as a transparent infrastructure enhancement.

- The upgrade preserves VRF awareness and compatibility with XR-specific socket, SSL, and networking features across supported platform variants.
- Existing HTTP client APIs and applications continue to operate without change on supported platforms.

How libcurl upgrade works

Describes how the HTTP client infrastructure uses the upgraded libcurl library, including integration, transaction flow, and improved security and maintainability.

The libcurl upgrade is an internal infrastructure improvement that replaces the embedded libcurl library version 7.30 with version 8.17.0. This change aligns the HTTP client stack with modern standards for protocols, security, and maintainability. The upgrade is transparent to users; no changes to commands or configuration are required. Application components continue to function as before, preserving backward compatibility. Security updates, platform-specific features, and operational observability are maintained through integration of upstream fixes and Cisco patches.

- The upgrade is internal and does not alter configuration workflows.
- All existing APIs and application interactions continue to be supported.
- Security and feature enhancements are introduced through targeted patches to the upgraded library.

Summary

The HTTP client infrastructure processes HTTP and HTTPS transactions through coordinated components that use a stable API and asynchronous communication.

- Application component: Initiates HTTP or HTTPS operations by calling the HTTP client library API.
- HTTP client library: Serves as the API interface and manages communication with the HTTP client process.
- HTTP client process: Handles request and response states, session management, concurrency, and calls libcurl.
- libcurl library: Executes protocol operations and network communication, with enhancements for XR integration.

Workflow

These stages describe the transaction flow for HTTP requests by using the upgraded libcurl library:

1. An application component, such as Smart Licensing, Install Manager, Zero Touch Provisioning, or the Copy utility, starts an HTTP or HTTPS request by calling the HTTP client library API.
   - The API remains unchanged, so existing feature components continue to work without modification.
   - The HTTP client library receives and processes the request.
2. The HTTP client library packages the request, adds required parameters, and sends the request asynchronously to the HTTP client process.
   - Asynchronous interprocess communication prevents operations from blocking.
   - Request details, such as VRF information and metadata, are included for execution.
   - The HTTP client process receives the prepared request.
3. The HTTP client process manages the request by queuing transactions and maintaining session state for multiple requests.
   - The process uses the libcurl multi interface to scale concurrent requests efficiently.
   - The process performs error handling and allocates required system resources.

- Requests are sequenced and validated before dispatch to the communication layer. The process then prepares to call libcurl for protocol execution.

4. The HTTP client process calls the `curl_multi_perform` API for efficient asynchronous network operations.
   - Handles XR-custom socket management, VRF context, and SSL using Cisco extensions.
   - Manages concurrency by using `select` and the libcurl multi interface.

   This stage provides reliable, efficient transaction processing. Network protocol negotiation and TLS setup occur in the updated, security-patched libcurl library.
5. The libcurl library establishes the network connection, negotiates SSL or TLS as needed, and completes the HTTP transaction by using HTTP/1.0 or HTTP/1.1.
   - Maintains socket management, SSL via `libssl_xr`/`libcrypto_xr`, and TCP settings.
   - Security patches and protocol improvements are applied in this stage.
   - The library sends response and transaction state information to the HTTP client process.
6. The HTTP client process parses the libcurl response, updates transaction status, logs actions or errors, and tracks operational state.
   - Uses standardized error codes and logging to simplify troubleshooting and monitoring.
   - Makes status information available through show commands.

   Transaction results are passed to the client library. The HTTP client library supplies the response and status to the application.
7. The application receives the response from the HTTP client library and continues normal operation.

   The HTTP transaction process completes, and resources are released. No user or application-level awareness or adaptation is needed for the upgrade.

Result

The system benefits from a secure, up-to-date HTTP client infrastructure with improved protocol support, stronger security, and better operational efficiency, with no change to applications or user experience.