# Configuration of an SSL Inspection Policy on the Cisco FireSIGHT System

## Contents

[Introduction](#)
[Prerequisites](#)
[Components Used](#)
[Configurations](#)
[1. Decrypt and Resign](#)
[Option 1: Use the FireSIGHT Center as a root Certificate Authority (CA)](#)
[Option 2: Have an internal CA sign your certificate](#)
[Option 3: Import a CA certificate and key](#)
[2. Decrypt with Known Key](#)
[Importing Known Certificate (Alternative to Decrypt and Resign)](#)
[Additional Configurations](#)
[Verification](#)
[Decrypt - Resign](#)
[Decrypt - Known Certificate](#)
[Troubleshooting](#)
[Issue 1: Some websites may not load on the Chrome browser](#)
[Issue 2: Getting an untrusted warning/error in some browsers](#)
[References](#)
[Related Cisco Support Community Discussions](#)

## Introduction

The SSL inspection feature allows you to either block encrypted traffic without inspecting it, or inspect encrypted or decrypted traffic with access control. This document describes the configuration steps to set up an SSL inspection policy on the Cisco FireSIGHT System.

## Prerequisites

### Components Used

- Cisco FireSIGHT Management Center
- Cisco Firepower 7000 or 8000 Appliances
- Software Version 5.4.1 or higher

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

> **Warning:** If you apply an SSL inspection policy on your managed device, it can impact network performance.

# Configurations

You can configure an SSL inspection policy to decrypt traffic the following ways:

1. Decrypt and Resign:

- Option 1: Use the FireSIGHT Center as a root Certificate Authority (CA), or
- Option 2: Have a internal CA sign your certificate, or
- Option 3: Import a CA certificate and key

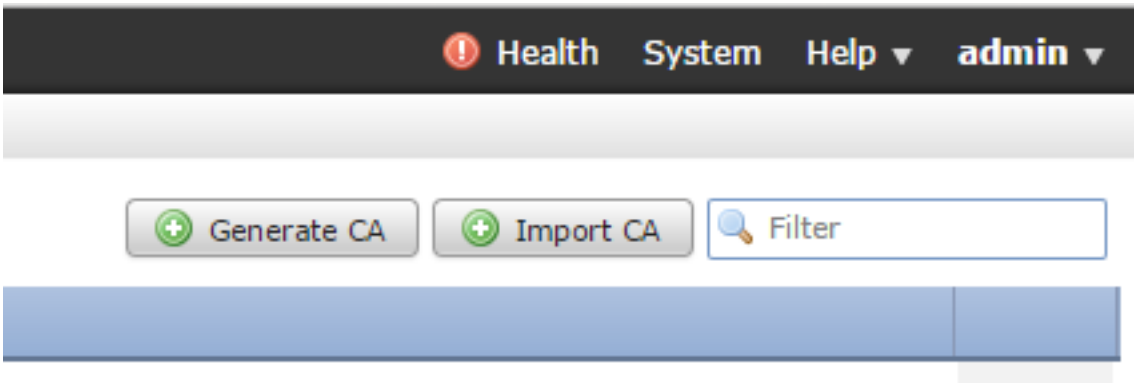
2. Decrypt with Known Cert:

- Log into the FireSIGHT Management Center, then navigate to **Objects**.
- On the **Objects** page, expand the **PKI** and select **Internal CAs**.

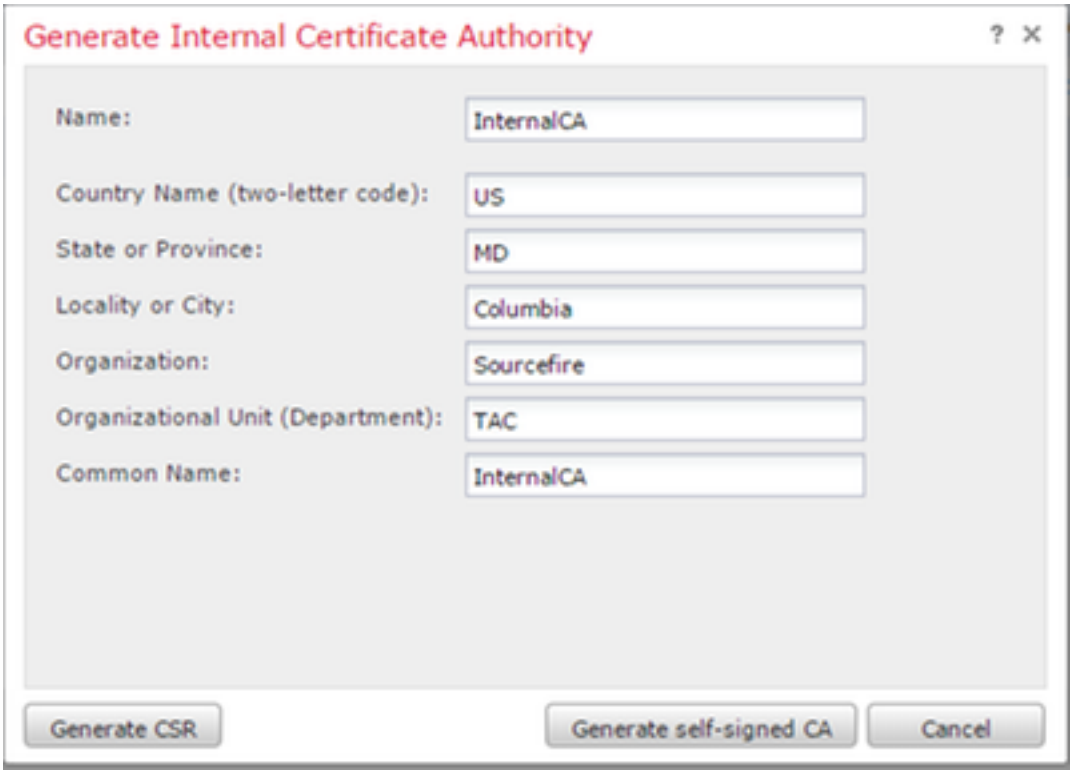
## 1. Decrypt and Resign

**Option 1: Use the FireSIGHT Center as a root Certificate Authority (CA)**

i. Click **Generate CA**.



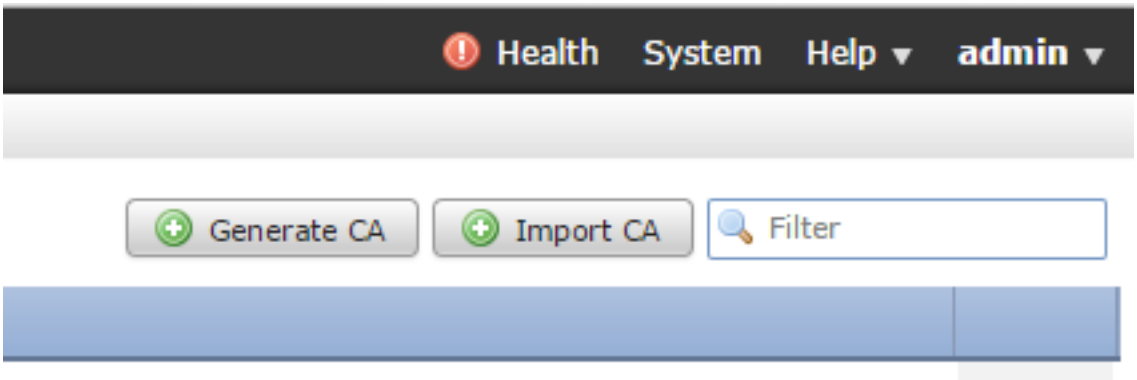

ii. Fill in the relevant information



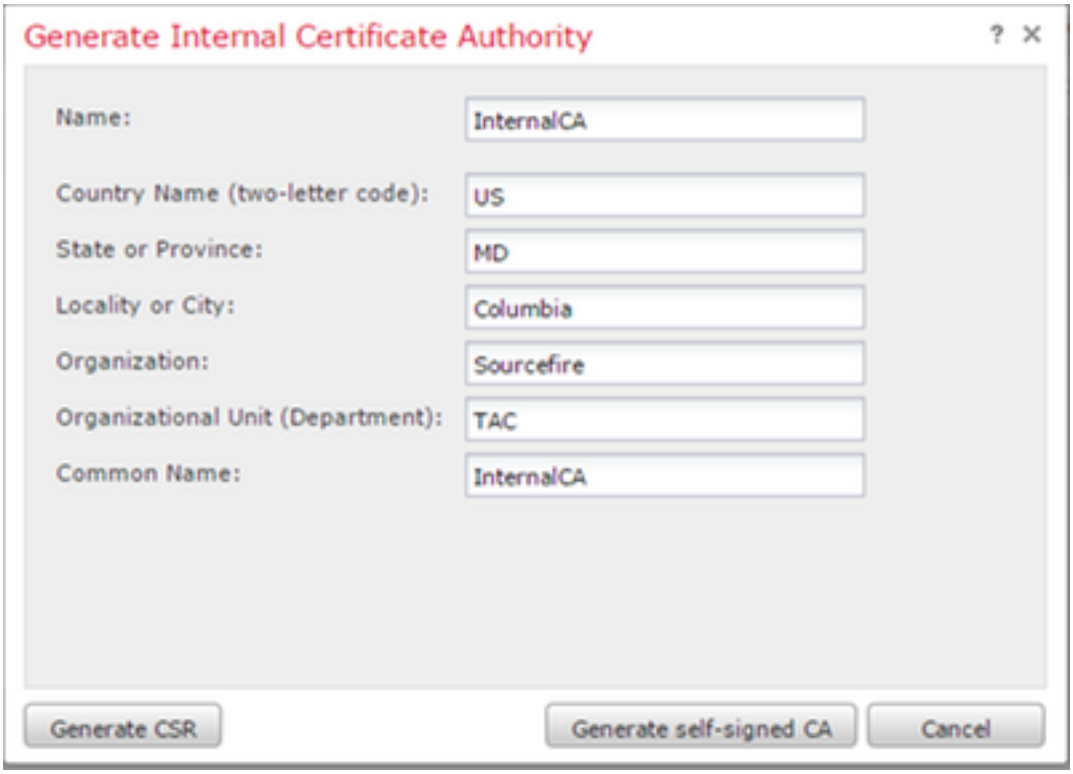

iii. Click **Generate self-signed CA**.

**Option 2: Have an internal CA sign your certificate**

i. Click **Generate CA**.

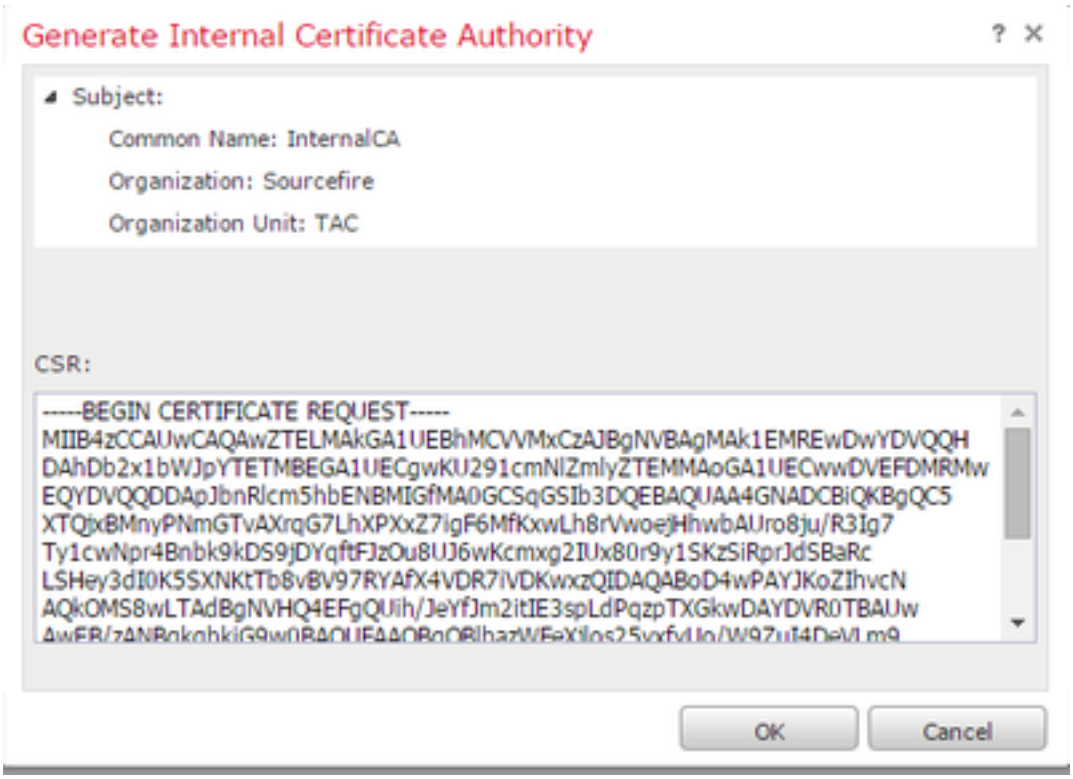



ii. Fill in the relevant information.



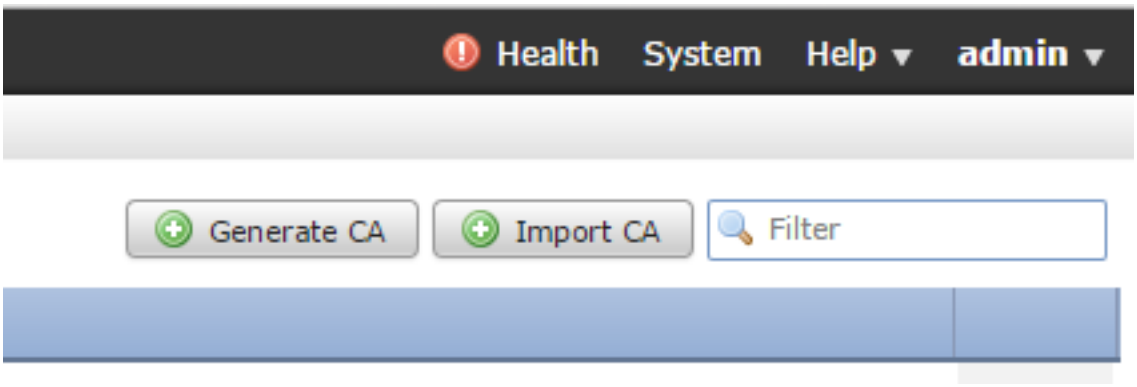

**Note:**You may need to contact your CA administrator to determine if they have a template for the signing request.

iii. Copy the entire certificate including the `---BEGIN CERTIFICATE REQUEST---` and `---END CERTIFICATE REQUEST---` and then save it to a text file with the `.req` extension.




**Note:** Your CA administrator request another file extension besides `.req`.

**Option 3: Import a CA certificate and key**

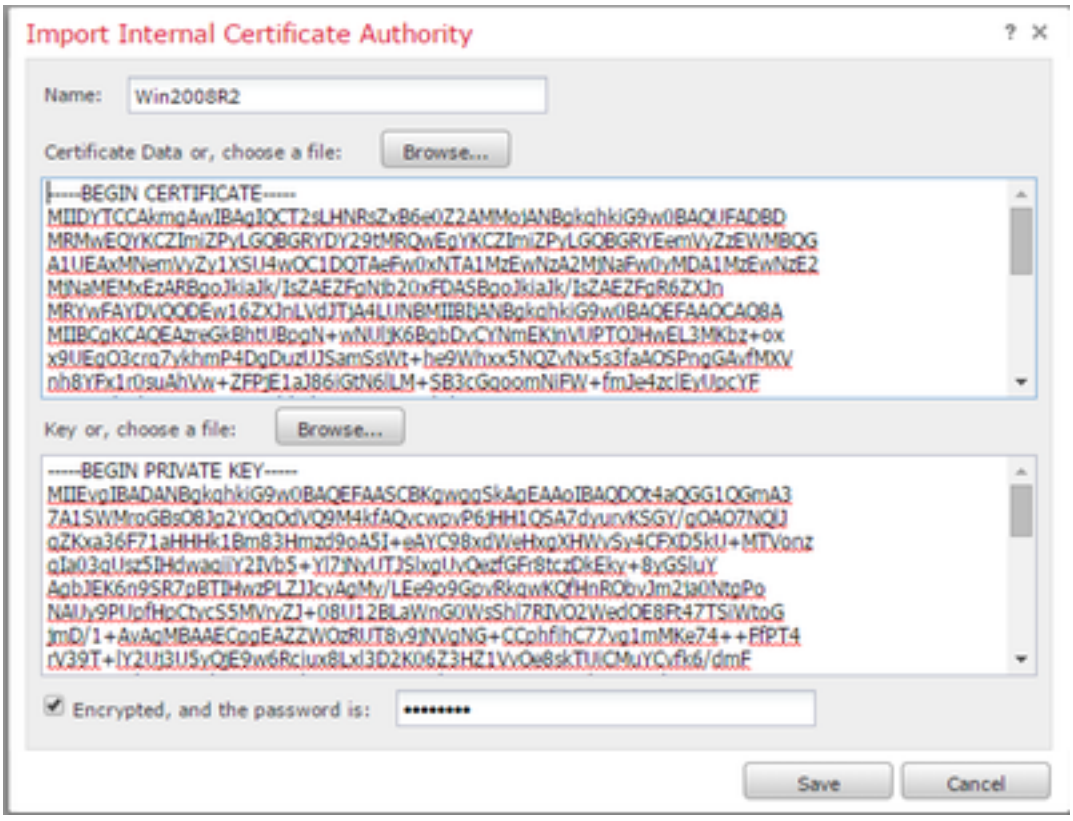



i. Click **Import CA**.

ii. Browse to or paste in the certificate.

iii. Browse to or paste in the private key.

iv. Check the encrypted box and type in a password.




**Note:** If there is no password, check the encrypted box and leave it blank.

## 2. Decrypt with Known Key

**Importing Known Certificate (Alternative to Decrypt and Resign)**

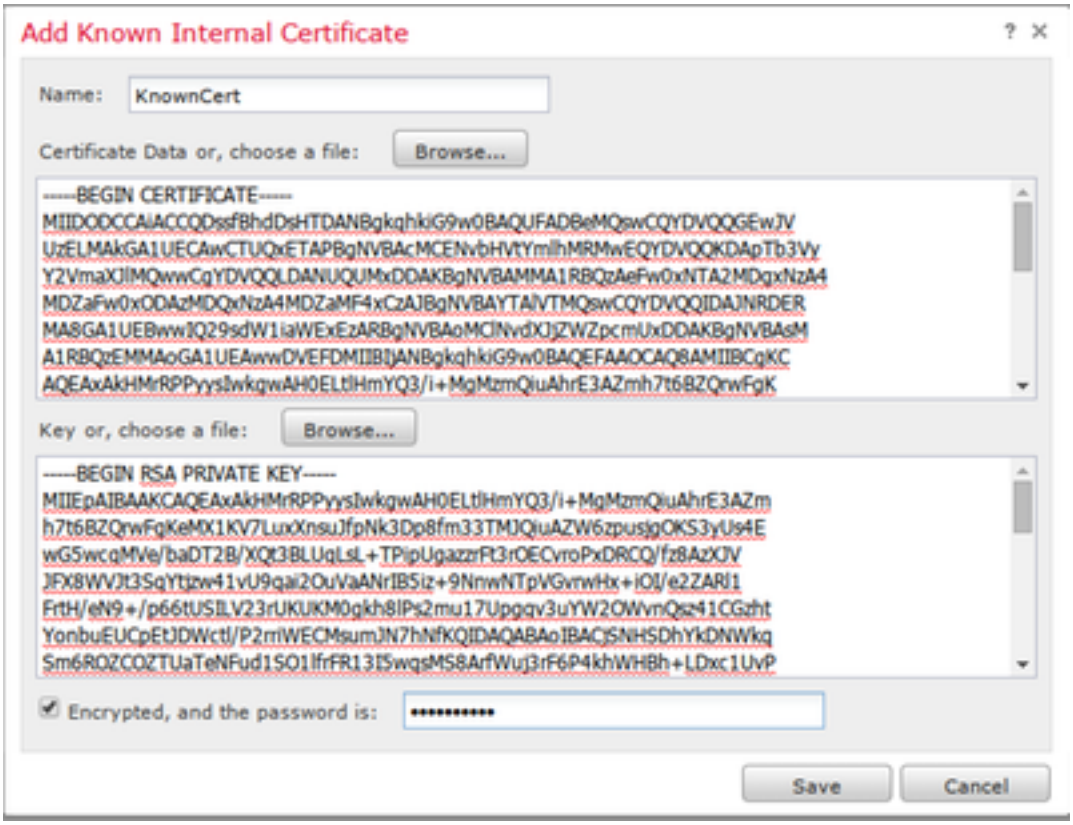
i. From the Objects page on the left expand out PKI and select Internal Certs.

ii. Click **Add Internal Cert**.

iii. Browse to or paste in the certificate.

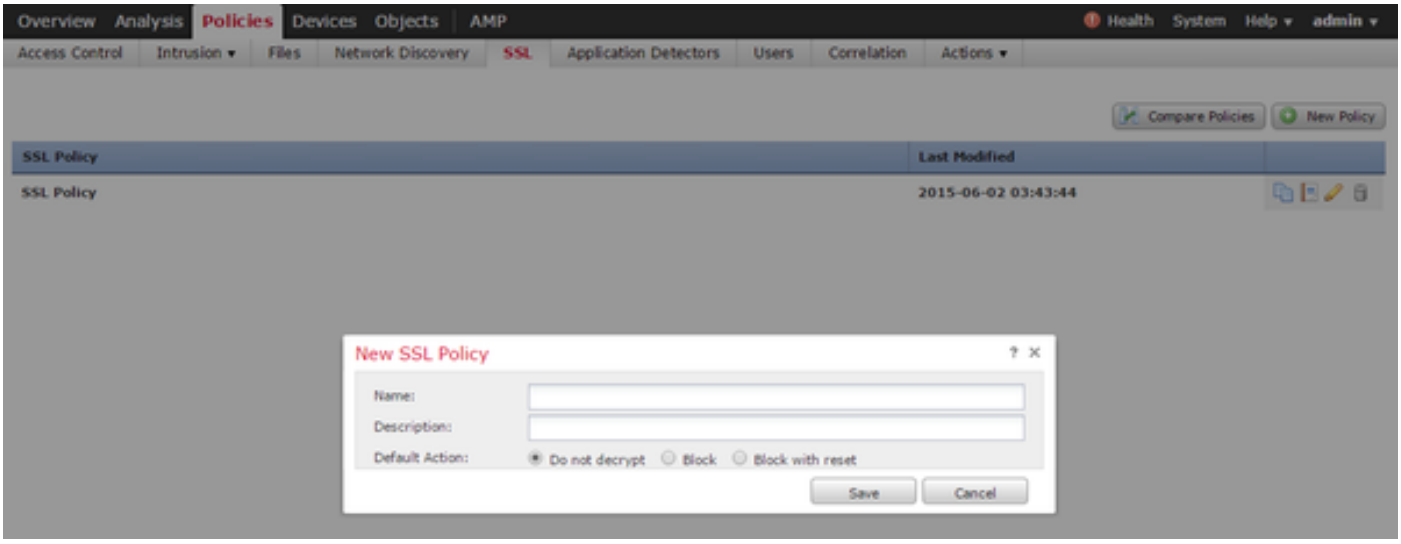
iv. Browse to or paste in the private key.

v. Check the **Encrypted** box and type in a password.




**Note:** If there is no password, leave the **Encrypted** box blank.

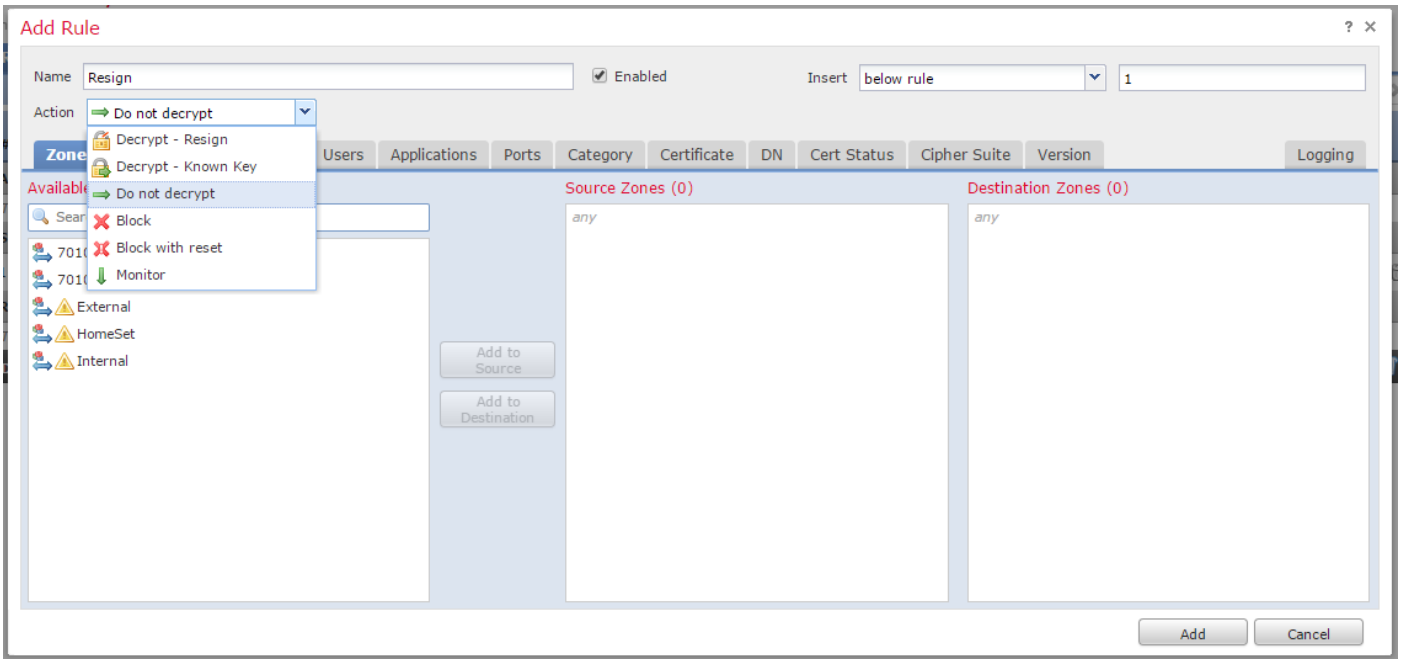
4. Navigate to **Policies > SSL** then click **New Policy**.




5. Provide a name and select a **Default Action**. The SSL policy editor page appears. The SSL policy editor page works the same as the Access Control Policy editor page.

> **Note:** If unsure about the **Default Action**, **Do not decrypt** is the recommended starting point.

6. On the SSL policy editor page, click **Add Rule**. In the Add Rule window, provide a name for the rule, and fill in all other relevant information.




The following section describes various options on the **Add Rule** window:

**Action**

## Decrypt - Resign

- The sensor acts as a Man in the Middle (MitM) and accepts the connection with the user, then establishes a new connection to the server. For example: User types in https://www.facebook.com in a browser. The traffic reaches the sensor, the sensor then negotiates with the user using the selected CA certificate and SSL tunnel A is built. At the same time the sensor connects to https://www.facebook.com and creates SSL tunnel B.
- End result: User see the certificate in the rule, not facebook's.

- This action requires an Internal CA. Select Replace Key if you wish the key to be replaced. The user will receive the certificate you select.

  **Note:** This cannot be used in passive mode.

### Decrypt - Known Key

- The sensor has the key that will be used to decrypt the traffic. For example: User types in https://www.facebook.com in a browser. The traffic reaches the sensor, the sensor decrypts the traffic, then inspects the traffic.
- End result: User see facebook's certificate
- This action requires an Internal Certificate. This is added in **Objects** > **PKI** > **Internal Certs**.

  **Note:** Your organization must be the owner of the domain and certificate. For the example of facebook.com the only possible way to have the end user see facebook's certificate would be if you actually own the domain facebook.com (i.e. your company is Facebook, Inc) and have ownership of the facebook.com certificate signed by a public CA. You can only decrypt with known keys for sites that your organization owns.

The main purpose of decrypt known key is to decrypt traffic heading to your https server to protect your servers from external attacks. For inspecting client side traffic to external https sites you will be using decrypt resign as you do not own the server and you are interested in inspecting the client traffic in your network connecting to external encrypted sites.

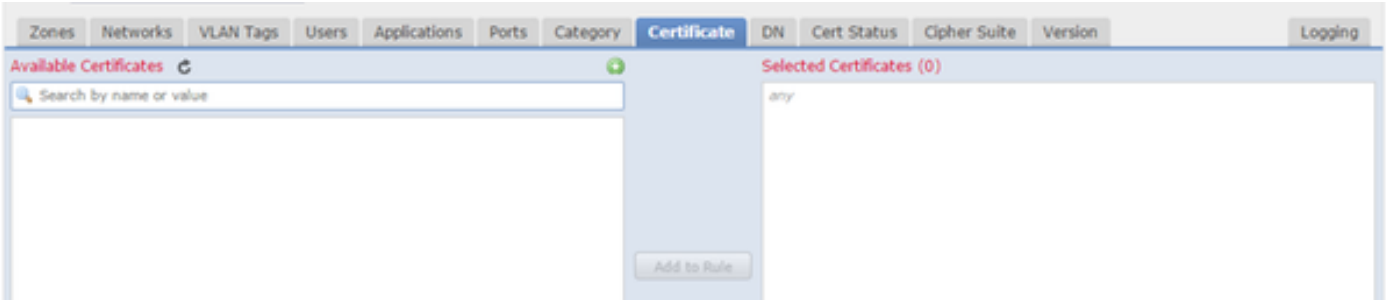
**Note:** For DHE and ECDHE to decrypt we must be in-line.

### Do Not Decrypt

Traffic bypasses the SSL policy and continues to the Access Control Policy.

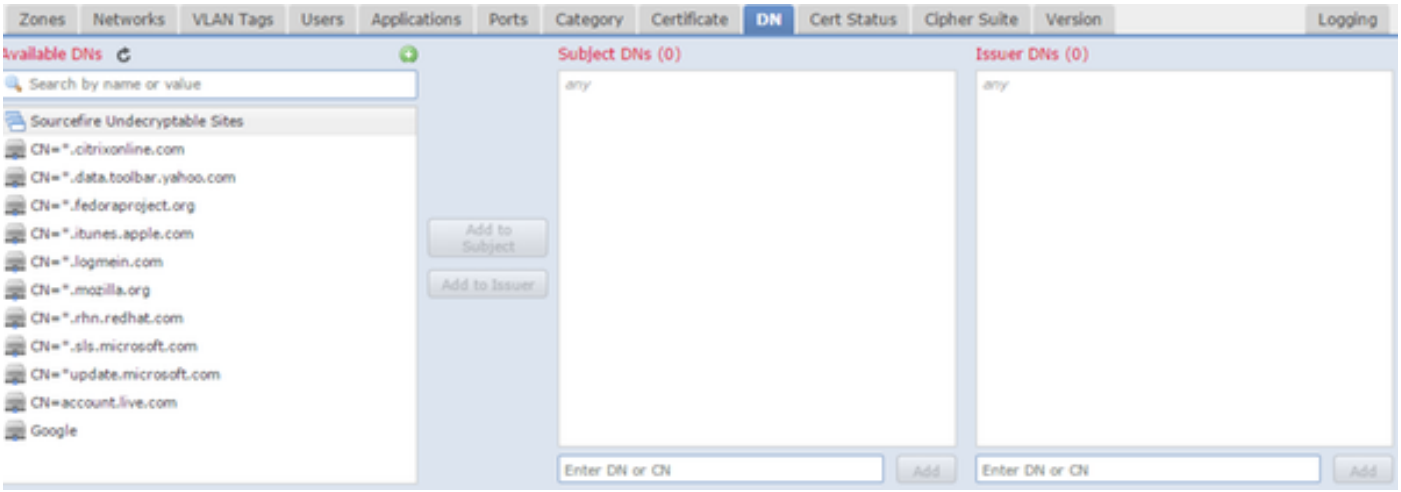
#### Certificate

Rule matches SSL traffic using this particular certificate.




#### DN

Rule matches SSL traffic using certain Domain Names in the certificates.



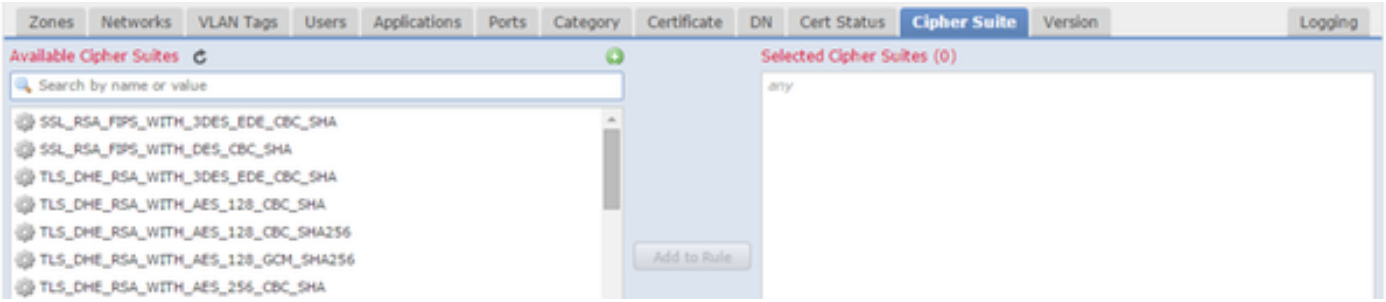

**Cert Status**

Rule matches SSL traffic with these certificate statuses.




**Cipher Suite**

Rule matches SSL traffic using these Cipher Suites.




**Version**

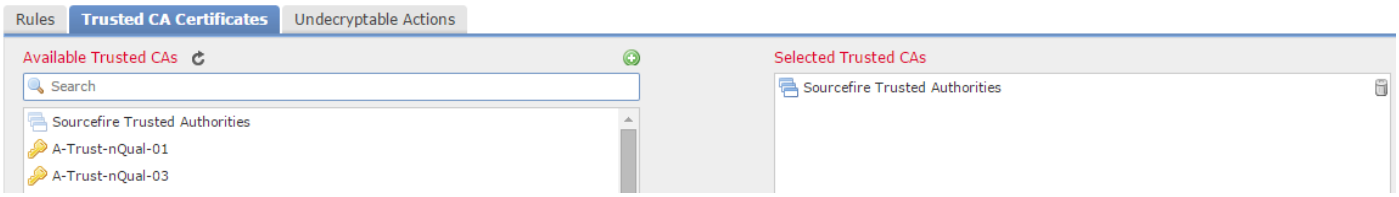
Rules applies only to SSL traffic with the selected versions of SSL.



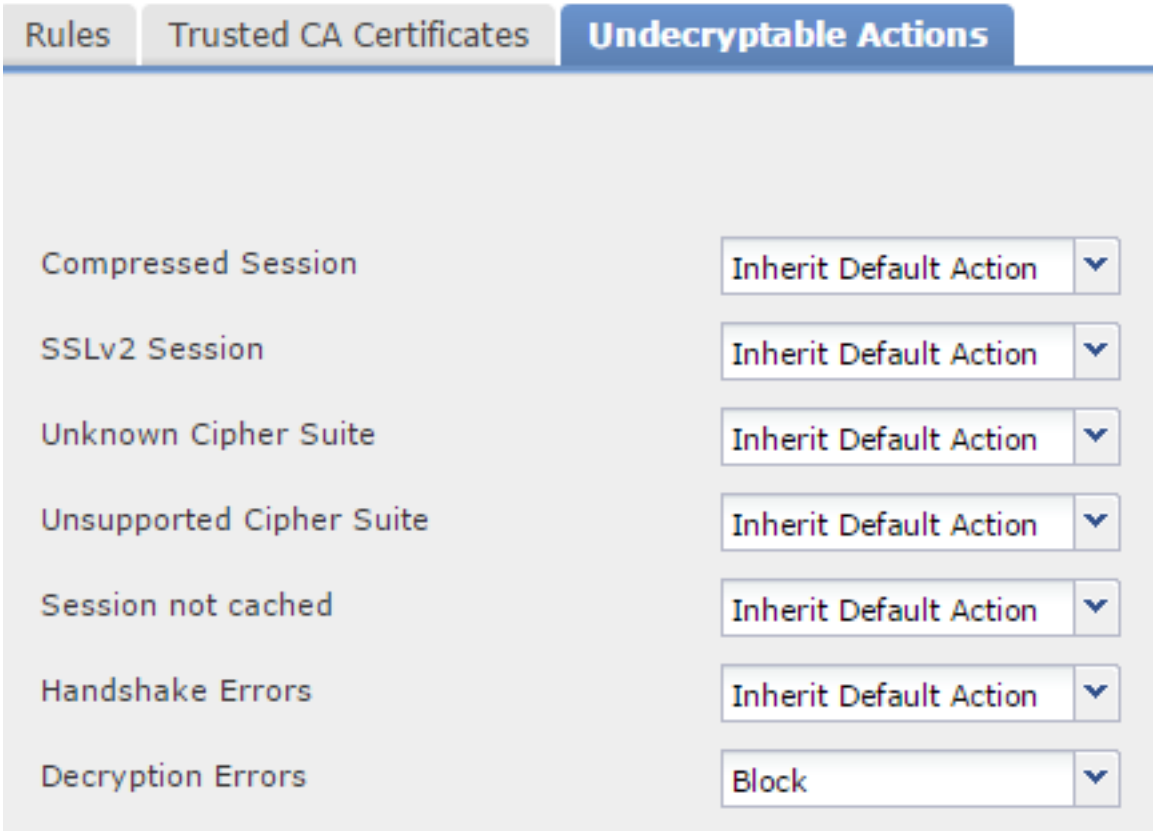

**Logging**

Enable logging to see connection events for the SSL traffic.

7. Click **Trusted CA Certificate**. This is where Trusted CA are added to the policy.




8. Click **Undecryptable Actions**. Here are the actions for which the sensor cannot decrypt the traffic. You can find the definitions are from the online help (**Help > Online**) of the FireSIGHT Management Center.




- **Compressed Session**: The SSL session applies a data compression method.
- **SSLv2 Session**: The session is encrypted with SSL version 2. Note that traffic is decryptable if the client hello message is SSL 2.0, and the remainder of the transmitted traffic is SSL 3.0.
- **Unknown Cipher Suite**: The system does not recognize the cipher suite.
- **Unsupported Cipher Suite**: The system does not support decryption based on the detected

cipher suite.

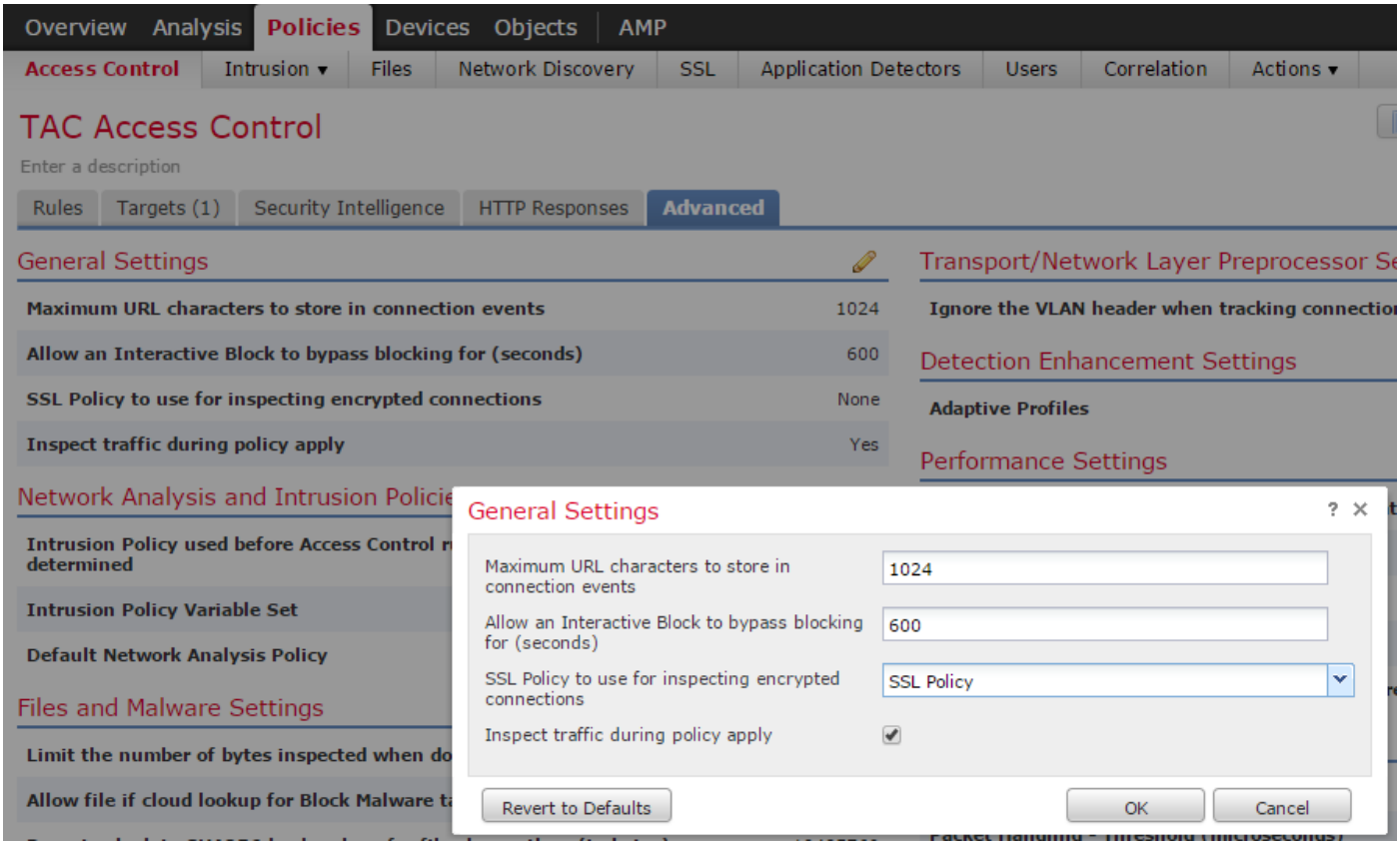
- **Session not cached**: The SSL session has session reuse enabled, the client and server reestablished the session with the session identifier, and the system did not cache that session identifier.
- **Handshake Errors**: An error occurred during SSL handshake negotiation.
- **Decryption Errors**: An error occurred during traffic decryption.

**Note:** By default these inherit the Default Action. If your default action is Block, you may experience unexpected issues

9. Save the policy.

10. Navigate to **Policies** > **Access Control**. Edit your policy or create a new Access Control Policy.

11. Click **Advanced** and edit the **General Settings**.



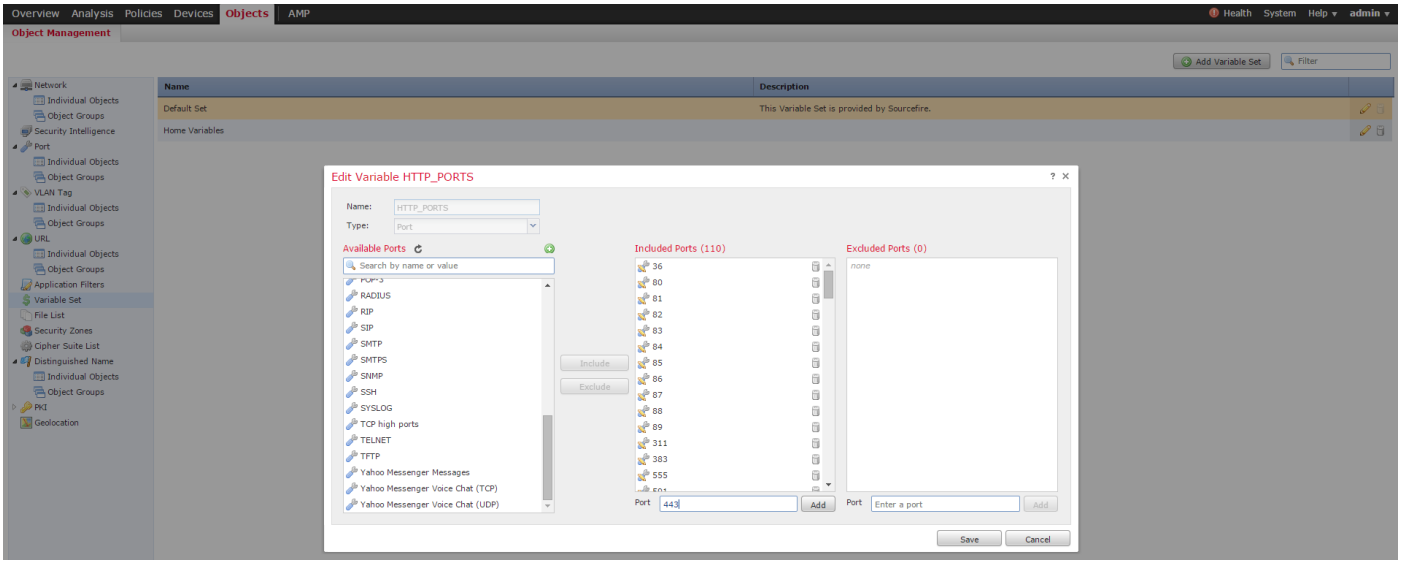

12. From the drop down menu select your **SSL Policy**.

13. Click **OK** to save.

**Additional Configurations**

The following changes should be made on the intrusion policies for proper identification:

i. Your `$HTTP_PORTS` variable should include port 443 and any other ports with https traffic that will be decrypted by your policy (**Objects** > **Object Management** > **Variable Set** > **Edit** the variable set).




ii. The Network Analysis policy that is inspecting the encrypted traffic must have port 443 (and any other ports with https traffic that will be decrypted by your policy) included in the ports field of the HTTP preprocessor settings otherwise none of the http rules with http content modifiers (i.e. `http_uri, http_header, etc.) will trigger because this is reliant on the http ports defined and the http buffers in snort will not be populated for traffic that is not going over the ports specified.`

iii. (Optional but recommended for better inspection) Add your https ports to the **TCP Stream Configuration** settings in the **Perform Stream Reassembly on Both Ports** field.

iv. Reapply the revised Access Control policy during a scheduled maintenance window.

> **Warning:** This modified policy can cause signifigant performance issues. This should be tested outside of production hours to reduce risk for network outage or preformance.

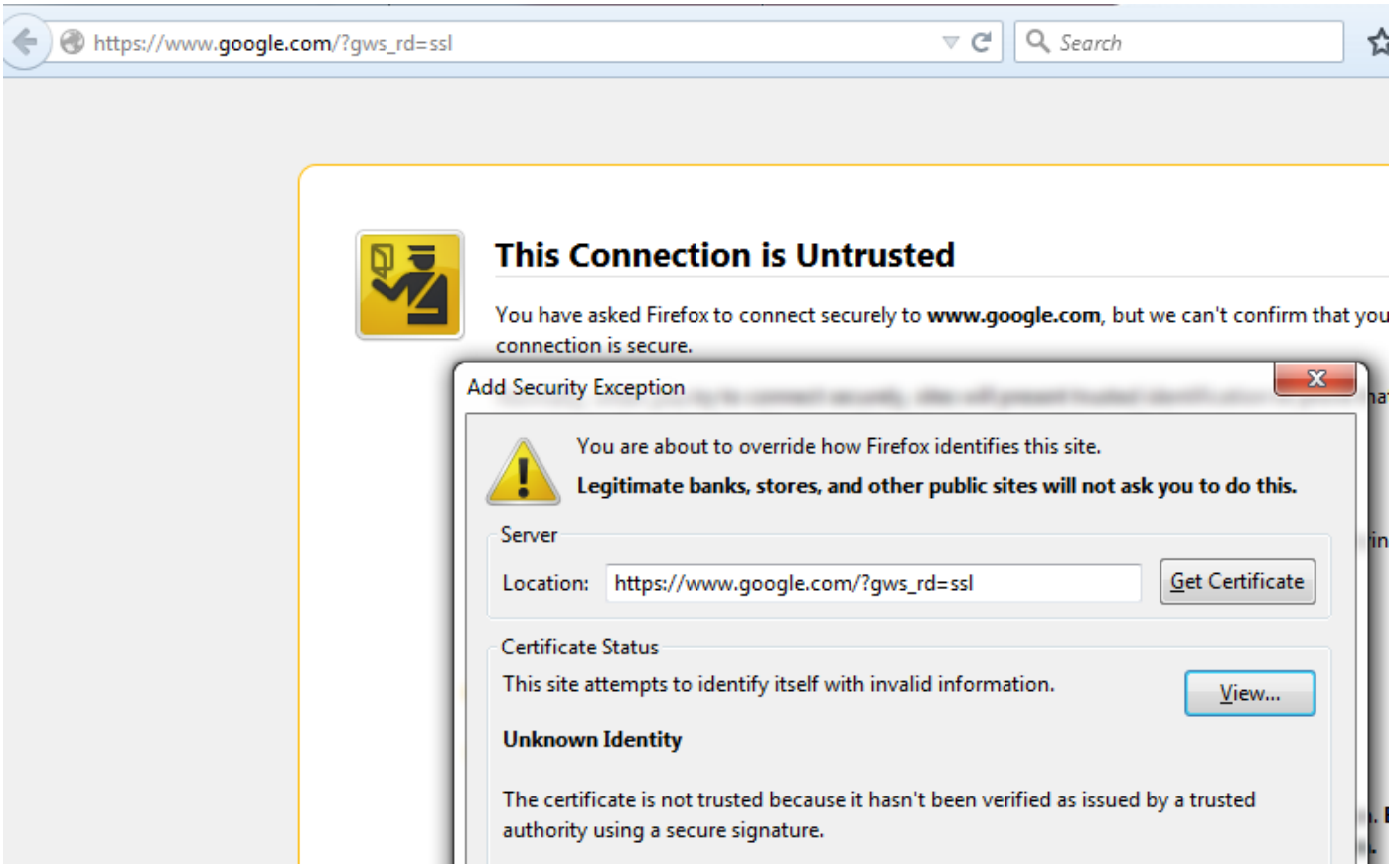
# Verification

**Decrypt - Resign**

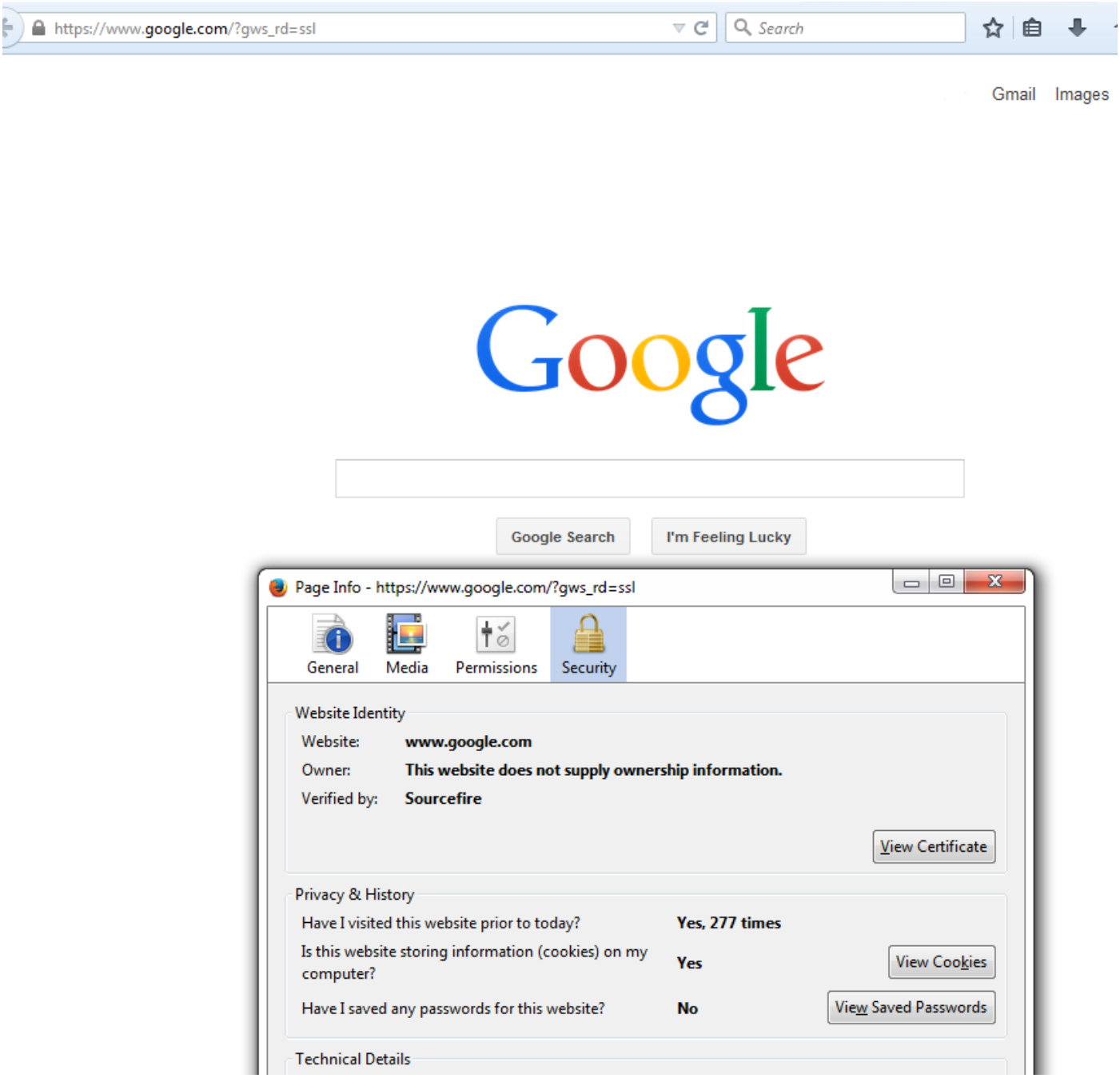
1. Open a web browser.

> **Note:** The Firefox browser is used in the example below. This example may not work in Chrome. See the Troubleshooting section for detail.

2. Navigate to an SSL website. In the example below https://www.google.com is used, The websites of financial institution will work as well. You will see one of the following pages:




**Note:**You will see the above page if the certificate itself is not trusted and the signing CA certificate is not trusted by your browser. To figure out how the browser determines trusted CA certificates see the Trusted Certificate Authorities section below.




**Note:** If this page is seen, you have succesfully re-signed the traffic. Note the section **Verified by: Sourcefire**.

**Could not verify this certificate because the issuer is unknown.**

**Issued To**

| | |
|---|---|
| Common Name (CN) | www.google.com |
| Organization (O) | Google Inc |
| Organizational Unit (OU) | <Not Part Of Certificate> |
| Serial Number | 13:E3:D5:7D:4E:5F:8F:E7 |

**Issued By**

| | |
|---|---|
| Common Name (CN) | Sourcefire TAC |
| Organization (O) | Sourcefire |
| Organizational Unit (OU) | Tac |

**Period of Validity**

| | |
|---|---|
| Begins On | 5/6/2015 |
| Expires On | 8/3/2015 |

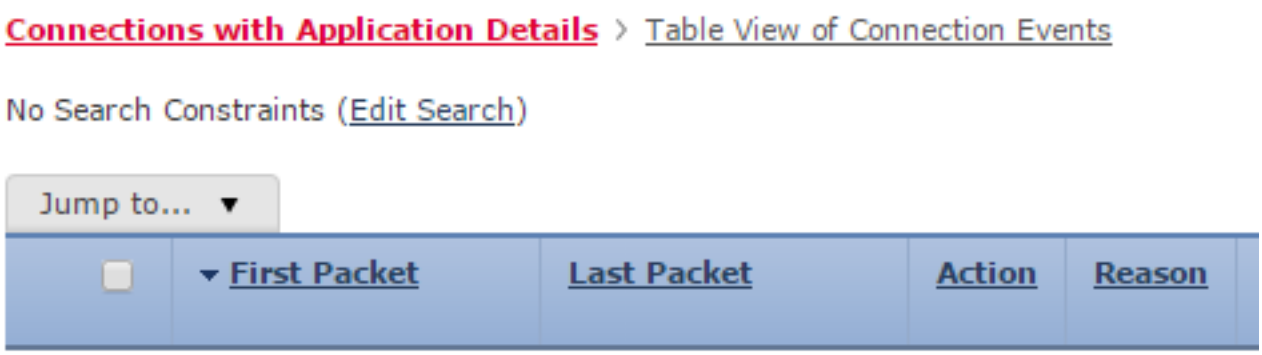
**Fingerprints**

| | |
|---|---|
| SHA-256 Fingerprint | 20:00:CB:25:13:8B:1F:89:4D:4A:CF:C5:E2:21:38:92:<br>06:66:00:2E:B7:83:27:72:98:EA:B1:6A:10:B3:67:A1 |
| SHA1 Fingerprint | 1B:C2:30:D9:66:84:DB:97:CF:A9:5E:5F:29:DA:4C:3F:13:E9:DE:5D |

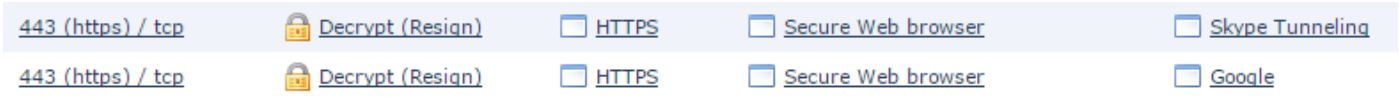
**Note:** This is a close up look at the same certificate.

3. In the Management Center go to **Analysis > Connections > Events**.

4. Depending upon your workflow you may or may not see SSL decrypt option. Click **Table View of Connection Events**.



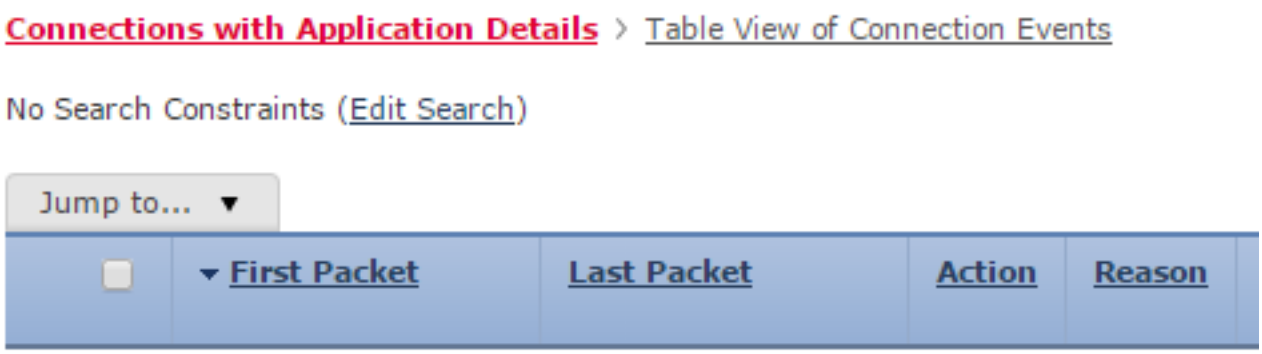

5. Scroll to the right and look for the SSL Status. You should see options similar to below:



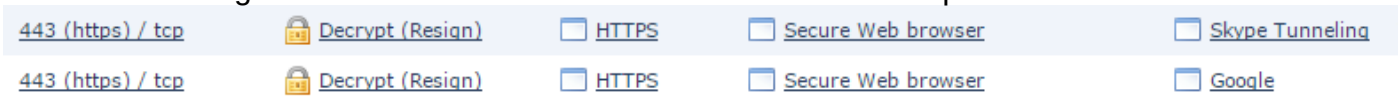

**Decrypt - Known Certificate**

1. In the FireSIGHT Management Center, navigate to **Analysis > Connections > Events**.

2. Depending upon your workflow, you may or may not see the SSL decrypt option. Click **Table View of Connection Events**.




3. Scroll to the right and look for the SSL Status. You should see options similar to below:




# Troubleshooting

## Issue 1: Some websites may not load on the Chrome browser

### Example

www.google.com may not load with a Decrypt - Resign using Chrome**.**

### Reason

The Google Chrome browser is capable of detecting fraudulent certificates for google properties in order to prevent man-in-the-middle attacks. If the Chrome browser (client) tries to connect to a google.com domain (server) and a certificate is returned that is not a valid google certificate, the browser will deny the connection.

### Solution

If you experience this, add a **Do Not Decrypt** rule for `DN=*.google.com, *.gmail.com, *.youtube.com. Then clear the browser cache and history.`

## Issue 2: Getting an untrusted warning/error in some browsers

### Example

When you connect to a site using Internet Explorer and Chrome, you do not receive a security warning, however when you use Firefox browser, you have to trust the connection every time you close and reopen the browser.

### Reason

The list of trusted CAs is dependant on the browser. When you trust a certificate this does not propegate across browsers and the trusted entry typically only persists while the browser is open, so once it is closed all certificates that were trusted will be pruned and the next time you open the browser and visit the site you must add it to the list of trusted certificates again.

### Solution

In this scenario both IE and Chrome use the list of trusted CAs in the operating system but Firefox maintains it's own list. So the CA cert was imported to the OS store but was not imported into the Firefox browser. In order to avoid getting the security warning in Firefox you have to import the CA cert into the browser as a trusted CA.

### Trusted Certificate Authorities

When an SSL connection is made the browser first checks to see if this certificate is trusted (i.e. you have been to this site before and manually told the browser to trust this certificate). If the certificate is not trusted the browser then checks the Certificate Authority (CA) certificate that verified the certificate for this site. If the CA certificate is trusted by the browser, it considers it a trusted certificate and allow the connection. If the CA certificate is not trusted, the browser displays a security warning and force you to manually add the certificate as a trusted certificate.

The list of trusted CAs in a browser is completely dependant on the brower's implementation and each browser can populate it's trusted list differently than other browsers. In general there are 2 ways that current browsers populate a list of trusted CAs:

1. They use the list of trusted CAs that the operating system trusts
2. They ship a list of trusted CAs with the software and it is built into the browser.

For the most common browsers the trusted CAs are populated as follows:

- **Google Chrome**: Operating system's trusted CA list
- **Firefox**: Maintains it's own trusted CA list
- **Internet Explorer**: Operating system's trusted CA list
- **Safari**: Operating system's trusted CA list

It is important to know the difference because the behavior seen on the client will vary depending on this. For example, in order to add a trusted CA for Chrome and IE you have to import the CA certificate to the OS's trusted CA store. If you import the CA certificate to the OS's trusted CA store you will no longer get a warning when connecting to sites with a certificate signed by this CA. On the Firefox browser, you must manually import the CA certificate into the trusted CA store in the browser itself. After doing this, you will no longer get a security warning when connecting to sites verified by that CA.

## References

- Getting started with SSL rules