



## Traffic Decryption Overview

The following topics provide an overview of Transport Layer Security/Secure Sockets Layer (TLS/SSL) inspection, discuss the prerequisites for TLS/SSL inspection configuration, and detail deployment scenarios.



### Note

Because TLS and SSL are often used interchangeably, we use the expression *TLS/SSL* to indicate that either protocol is being discussed. The SSL protocol has been deprecated by the IETF in favor of the more secure TLS protocol, so you can usually interpret *TLS/SSL* as referring to TLS only.

The exception is SSL policies. Because the FMC configuration option is **Policies > Access Control heading > SSL**, we use the term *SSL policies* although these policies are used to define rules for TLS and SSL traffic.

For more information about SSL and TLS protocols, see a resource such as [SSL vs. TLS - What's the Difference?](#)

- [Traffic Decryption Explained, on page 1](#)
- [TLS/SSL Handshake Processing, on page 3](#)
- [TLS/SSL Rule and Policy Basics, on page 8](#)
- [TLS Crypto Acceleration, on page 15](#)
- [How to Configure SSL Policies and Rules, on page 18](#)
- [History for SSL Policy, on page 19](#)

## Traffic Decryption Explained

Most traffic on the internet is encrypted and in most cases, you don't want to decrypt it; even if you don't, you can still glean some information about it and block it from your network if necessary.

Your choices are:

- Decrypt the traffic and subject it to the full array of deep inspection:
  - Advanced Malware Protection
  - Security intelligence
  - Threat Intelligence Director
  - Application detectors
  - URL and category filtering

- Leave the traffic encrypted and set up your access control and SSL policy to look for and potentially block:
  - Old protocol versions (such as Secure Sockets Layer)
  - Unsecure cipher suites
  - Applications with high risk and low business relevance
  - Untrusted issuer Distinguished Names

### Access control policy

An access control policy is the main configuration that invokes subpolicies and other configurations, including an SSL policy. If you associate an SSL policy with access control, the system uses that SSL policy to handle encrypted sessions before it evaluates the sessions with access control rules. If you do not configure TLS/SSL inspection, or your devices do not support it, access control rules handle all encrypted traffic.

Access control rules also handle encrypted traffic when your TLS/SSL inspection configuration allows the traffic to pass. However, some access control rule conditions require unencrypted traffic, so encrypted traffic might match fewer rules. Also, by default, the system disables intrusion and file inspection of encrypted payloads. This helps reduce false positives and improves performance when an encrypted connection matches an access control rule that has intrusion and file inspection configured.

### Notes

TLS/SSL rules require processing overhead that can impact performance. Before you decide to decrypt traffic, see [When to Decrypt Traffic, When Not to Decrypt, on page 9](#).

The Firepower System does not support mutual authentication; that is, you cannot upload a [client certificate](#) to the Firepower Management Center and use it for either **Decrypt - Resign** or **Decrypt - Known Key** TLS/SSL rule actions. For more information, see [Decrypt and Resign \(Outgoing Traffic\), on page 11](#). and [Known Key Decryption \(Incoming Traffic\), on page 11](#).

The Firepower System does not currently support TLS version 1.3 encryption or decryption. When users visit a web site that negotiates TLS 1.3 encryption, users might see errors similar to the following in their web browser:

- **ERR\_SSL\_PROTOCOL\_ERROR**
- **SEC\_ERROR\_BAD\_SIGNATURE**
- **ERR\_SSL\_VERSION\_INTERFERENCE**

For more information about how to control this behavior, contact [Cisco TAC](#).

If you set the value of TCP maximum segment size (MSS) using FlexConfig, the observed MSS could be less than your setting. For more information, see [About the TCP MSS](#).

### Related Topics

[TLS/SSL Handshake Processing, on page 3](#)

[TLS/SSL Rule and Policy Basics, on page 8](#)

# TLS/SSL Handshake Processing

In this documentation, the term *TLS/SSL handshake* represents the two-way handshake that initiates encrypted sessions in both the SSL protocol and its successor protocol, TLS.

In an inline deployment, the system processes the TLS/SSL handshake, potentially modifying the ClientHello message and acting as a TCP proxy server for the session.

The following figure shows an inline deployment.



After the client establishes a TCP connection with the server (after it successfully completes the TCP [three-way handshake](#)), the managed device monitors the TCP session for any attempt to initiate an encrypted session. The TLS/SSL handshake establishes an encrypted session using the exchange of specialized packets between client and server. In the SSL and TLS protocols, these specialized packets are called *handshake messages*. The handshake messages communicate which encryption attributes both the client and server support:

- ClientHello—The client specifies multiple supported values for each encryption attribute.
- ServerHello—The server specifies a single supported value for each encryption attribute, and the ServerHello response determines which encryption method the system uses during the secure session.

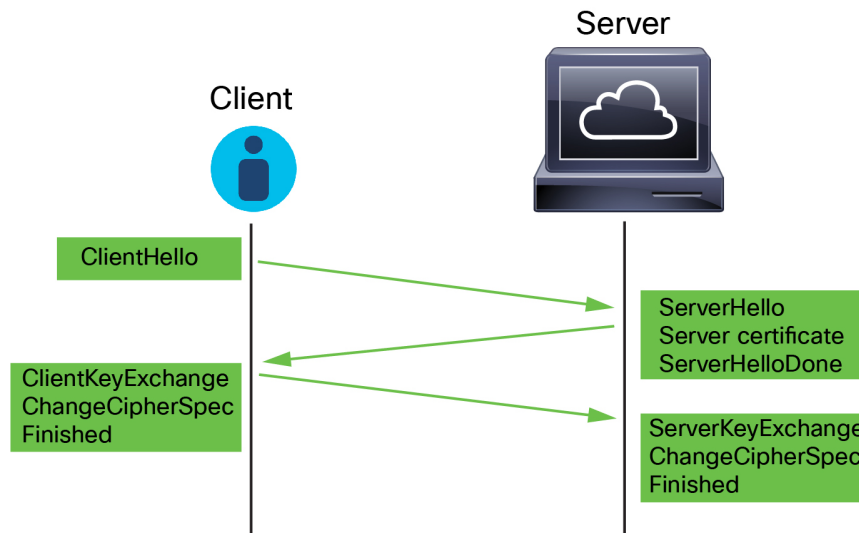
After a TLS/SSL handshake completes, the managed device caches encrypted session data, which allows session resumption without requiring the full handshake. The managed device also caches server certificate data, which allows faster handshake processing in subsequent sessions that use the same certificate.

## ClientHello Message Handling

The client sends the ClientHello message to the server that acts as the packet destination if a secure connection can be established. The client sends the message to initiate the TLS/SSL handshake or in response to a ServerHello message from the destination server.

### Overview

The following figure shows an example. Also see [RFC 8446, sec. 4](#) [RFC 5246, sec. 7.3](#). You can also consult a resource such as [What Happens in a TLS Handshake?](#) at [cloudflare.com](#).



The process can be summarized as follows:

1. ClientHello initiates the process.

The ClientHello message contains the [Server Name Indication \(SNI\)](#), which has the server's fully qualified domain name.

2. After a managed device processes a ClientHello message and transmits it to the destination server, the server determines whether it supports the decryption attributes the client specified in the message. If it does not support those attributes, the server sends a handshake failure alert to the client. If it supports those attributes, the server sends the ServerHello message. If the agreed-upon key exchange method uses certificates for authentication, the server certificate message immediately follows the ServerHello message.

The server certificate contains the [Subject Alternative Name \(SAN\)](#), which can have fully qualified domain names and IP addresses. For more information about the SAN, see [Distinguished Name](#).

3. When the managed device receives these messages, it attempts to match them with TLS/SSL rules configured on the system. These messages contain information that was absent from either the ClientHello message or the session data cache. Specifically, the system can potentially match these messages on TLS/SSL rules' Distinguished Names, Certificate Status, Cipher Suites, and Versions conditions.

Note that although the data transmitted in the session is encrypted, the handshake messages are not encrypted.

### Data exchange

If you configure TLS/SSL decryption, when a managed device receives a ClientHello message, the system attempts to match the message to TLS/SSL rules that have the **Decrypt - Resign** or **Decrypt - Known Key** action. The match relies on data from the ClientHello message and from cached server certificate data. Possible data includes:

**Table 1: Data Availability for TLS/SSL Rule Conditions**

| TLS/SSL Rule Condition | Data Present In |
|------------------------|-----------------|
| Zones                  | ClientHello     |
| Networks               | ClientHello     |

| TLS/SSL Rule Condition | Data Present In                               |
|------------------------|---|
| VLAN Tags              | ClientHello                                   |
| Ports                  | ClientHello                                   |
| Users                  | ClientHello                                   |
| Applications           | ClientHello (Server Name Indicator extension) |
| Categories             | ClientHello (Server Name Indicator extension) |
| Certificate            | Server certificate (potentially cached)       |
| Distinguished Names    | Server certificate (potentially cached)       |
| Certificate Status     | Server certificate (potentially cached)       |
| Cipher Suites          | ServerHello                                   |
| Versions               | ServerHello                                   |



**Important** Use the **Cipher Suite** and **Version** rule conditions *only* in rules with either the **Block** or **Block with reset** rule actions. Do not use **Cipher Suite** and **Version** with either Decrypt - Resign or Decrypt - Known Key rule actions. These conditions in rules with other rule actions can interfere with the system's ClientHello processing, resulting in unpredictable performance.

### ClientHello modifications

If the ClientHello message matches a **Decrypt - Resign** or **Decrypt - Known Key** rule, the system modifies the ClientHello message as follows:

- Compression methods—Strips the `compression_methods` element, which specifies the compression methods the client supports. The system cannot decrypt compressed sessions.
- Cipher suites—Strips cipher suites from the `cipher_suites` element if the system does not support them. If the system does not support any of the specified cipher suites, the system transmits the original, unmodified element. This modification reduces the Unknown Cipher Suite and Unsupported Cipher Suite types of undecryptable traffic.
- Session identifiers—Strips any value from the `Session Identifier` element and the [SessionTicket extension](#) (RFC 5077, sec 3.2) that does not match cached session data. If a ClientHello value matches cached data, an interrupted session can resume without the client and server performing the full TLS/SSL handshake. This modification increases the chances of session resumption and reduces the Session Not Cached type of undecryptable traffic.
- Elliptic curves—Strips elliptic curves from the Supported Elliptic Curves extension if the system does not support them. If the system does not support any of the specified elliptic curves, the managed device removes the extension and strips any related cipher suites from the `cipher_suites` element.
- ALPN extensions—Strips any value from the Application-Layer Protocol Negotiation (ALPN) extension that is unsupported in the system (for example, the HTTP/2 protocol).

- Other Extensions—Strips the Next Protocol Negotiation (NPN) and TLS Channel IDs extensions.

TLS/SSL rules with a **Decrypt - Resign** or **Decrypt - Known Key** action now natively support the Extended Master Secret (EMS) extension during ClientHello negotiation, enabling more secure communications. The EMS extension is defined by [RFC 7627](#).

After the system modifies the ClientHello message, it determines whether the message passes access control evaluation (which can include deep inspection). If the message passes, the system transmits it to the destination server.

If the ClientHello message does *not* match a **Decrypt - Resign** or **Decrypt - Known Key** rule, the system does not modify the message. It then determines whether the message passes access control evaluation (which can include deep inspection). If the message passes inspection, the system transmits it to the destination server.

ClientHello is *not* modified if traffic matches a **Monitor** rule condition.

### Man-in-the-middle

Direct communication between the client and server is no longer possible during the TLS/SSL handshake, because after message modification the Message Authentication Codes (MACs) computed by the client and server no longer match. For all subsequent handshake messages (and for the encrypted session once established), the managed device acts as a man-in-the-middle. It creates two TLS/SSL sessions, one between client and managed device, one between managed device and server. As a result, each session contains different cryptographic session details.




---

**Note** The cipher suites that the system can decrypt are frequently updated and do not correspond directly to the cipher suites you can use in TLS/SSL rule conditions. For the current list of decryptable cipher suites, contact Cisco TAC.

---

### Related Topics

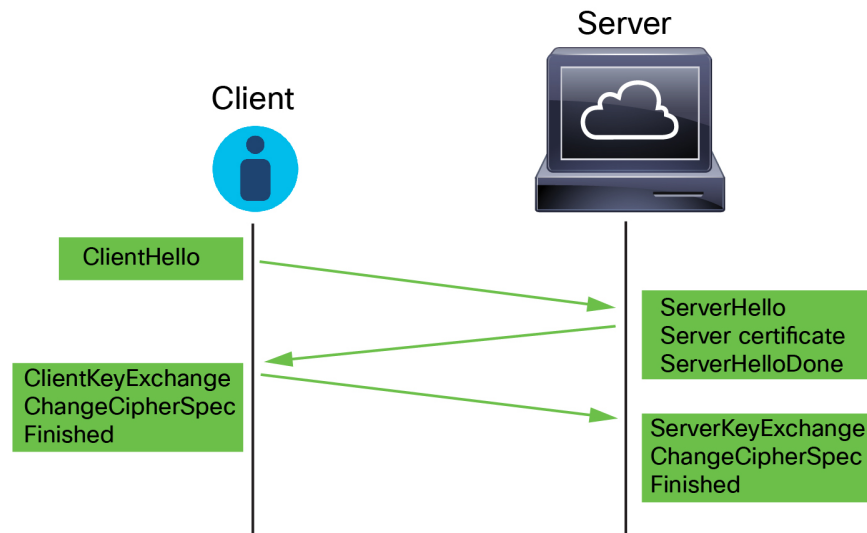
[Default Handling Options for Undecryptable Traffic](#)

[ServerHello and Server Certificate Message Handling](#), on page 6

## ServerHello and Server Certificate Message Handling

### Overview

The following figure shows an example. Also see [RFC 8446, sec. 4](#) [RFC 5246, sec. 7.3](#). You can also consult a resource such as [What Happens in a TLS Handshake?](#) at [cloudflare.com](#).



The process can be summarized as follows:

1. ClientHello initiates the process.

The ClientHello message contains the [Server Name Indication \(SNI\)](#), which has the server's fully qualified domain name.

2. After a managed device processes a ClientHello message and transmits it to the destination server, the server determines whether it supports the decryption attributes the client specified in the message. If it does not support those attributes, the server sends a handshake failure alert to the client. If it supports those attributes, the server sends the ServerHello message. If the agreed-upon key exchange method uses certificates for authentication, the server certificate message immediately follows the ServerHello message.

The server certificate contains the [Subject Alternative Name \(SAN\)](#), which can have fully qualified domain names and IP addresses. For more information about the SAN, see [Distinguished Name](#).

3. When the managed device receives these messages, it attempts to match them with TLS/SSL rules configured on the system. These messages contain information that was absent from either the ClientHello message or the session data cache. Specifically, the system can potentially match these messages on TLS/SSL rules' Distinguished Names, Certificate Status, Cipher Suites, and Versions conditions.

Note that although the data transmitted in the session is encrypted, the handshake messages are not encrypted.

### TLS/SSL Rule actions

If the messages do not match any TLS/SSL rules, the managed device performs the [SSL Policy Default Actions](#).

If the messages match a rule that belongs to an SSL policy associated with an access control policy, the managed device continues as appropriate:

#### Action: Monitor

The TLS/SSL handshake continues to completion. The managed device tracks and logs traffic but does not decrypt encrypted it.

#### Action: Block or Block with Reset

The managed device blocks the TLS/SSL session and, if configured, resets the TCP connection.

### Action: Do Not Decrypt

The TLS/SSL handshake continues to completion. The managed device does not decrypt the application data exchanged during the TLS/SSL session.

### Action: Decrypt - Known Key

The managed device attempts to match the server certificate data to an Internal Certificate object previously imported into the Firepower Management Center . Because you cannot generate an Internal Certificate object, and you must possess its private key, we assume you own the server on which you're using known key decryption.

If the certificate matches a known certificate, the TLS/SSL handshake continues to completion. The managed device uses the uploaded private key to decrypt and re encrypt the application data exchanged during the TLS/SSL session.

If the server changes its certificate between the initial connection with the client and subsequent connections, you must import the new server certificate in the Firepower Management Center for future connections to be decrypted.

### Action: Decrypt - Resign

The managed device processes the server certificate message and re-signs the server certificate with the previously imported or generated certificate authority (CA). The TLS/SSL handshake continues to completion. The managed device then uses the uploaded private key to decrypt and re encrypt the application data exchanged during the TLS/SSL session.



**Note** The Firepower System does not support mutual authentication; that is, you cannot upload a [client certificate](#) to the Firepower Management Center and use it for either **Decrypt - Resign** or **Decrypt - Known Key** TLS/SSL rule actions. For more information, see [Decrypt and Resign \(Outgoing Traffic\)](#), on page 11. and [Known Key Decryption \(Incoming Traffic\)](#), on page 11.

### Related Topics

[ClientHello Message Handling](#), on page 3

## TLS/SSL Rule and Policy Basics

This section discusses information you should keep in mind when creating your SSL policies and rules.



**Note** Because TLS and SSL are often used interchangeably, we use the expression *TLS/SSL* to indicate that either protocol is being discussed. The SSL protocol has been deprecated by the IETF in favor of the more secure TLS protocol, so you can usually interpret *TLS/SSL* as referring to TLS only.

The exception is SSL policies. Because the FMC configuration option is **Policies > Access Control heading > SSL**, we use the term *SSL policies* although these policies are used to define rules for TLS and SSL traffic.

For more information about SSL and TLS protocols, see a resource such as [SSL vs. TLS - What's the Difference?](#)



**Related Topics**

- [The Case for Decryption](#), on page 9
- [When to Decrypt Traffic, When Not to Decrypt](#), on page 9
- [Other TLS/SSL Rule Actions](#), on page 12
- [TLS/SSL Rule Components](#), on page 12
- [TLS/SSL Rule Order Evaluation](#), on page 13
- [TLS 1.3 Decryption Best Practices](#)

## The Case for Decryption

Traffic that is encrypted when it passes through the system can be allowed or blocked only but it *cannot* be subjected to deep inspection or the full range of policy enforcement (such as intrusion prevention).

All encrypted connections:

- Are sent through the SSL policy to determine if they should be decrypted or blocked.

You can also configure TLS/SSL rules to block encrypted traffic of types you know you do not want on your network, such as traffic that uses the nonsecure SSL protocol or traffic with an expired or invalid certificate.

- If unblocked, whether or not decrypted, traffic goes through the access control policy for a final allow or block decision.

Only *decrypted* traffic takes advantage of the system's threat defense and policy enforcement features, such as:

- Advanced Malware Protection
- Security intelligence
- Threat Intelligence Director
- Application detectors
- URL and category filtering

Keep in mind that decrypting and then re-encrypting traffic adds a processing load on the device, which can reduce overall system performance.

In summary:

- Encrypted traffic can be allowed or blocked by policy; encrypted traffic *cannot* be inspected
- Decrypted traffic is subject to threat defense and policy enforcement; decrypted traffic can be allowed or blocked by policy

**Related Topics**

- [Deep Inspection Using File and Intrusion Policies](#)

## When to Decrypt Traffic, When Not to Decrypt

This section provides guidelines on when you should decrypt traffic and when you should allow it to pass through the firewall encrypted.

### When not to decrypt traffic

You should not decrypt traffic if doing so is forbidden by:

- Law; for example, some jurisdictions forbid decrypting financial information
- Company policy; for example, your company might forbid decrypting privileged communications
- Privacy regulations
- Traffic that uses certificate pinning (also referred to as *TLS/SSL pinning*) must remain encrypted to prevent breaking the connection

(Snort 3.) SSL policy is *not* bypassed for any connections that match access control rules with actions of Trust, Block, or Block with Reset, unless the traffic is prefiltered. The encrypted traffic is first evaluated by SSL policy and then proceeds to the access control policy, where a final allow or block decision is made.

Encrypted traffic can be allowed or blocked on any TLS/SSL rule condition, including, but not limited to:

- Certificate status (for example, expired or invalid certificate)
- Protocol (for example, the nonsecure SSL protocol)
- Network (security zone, IP address, VLAN tag, and so on)
- Exact URL or URL category
- Port
- User group

TLS/SSL rules provide a **Do Not Decrypt** action for this traffic; for more information, see [TLS/SSL Rule Do Not Decrypt Action](#).



**Note** The related information links at the end of this topic explain how some aspects of rule evaluation work. Conditions such as URL and application filtering have limitations with respect to encrypted traffic. Make sure you understand those limitations.

For more information about using URL filtering in **Do Not Decrypt** rules, see [TLS/SSL Rule Do Not Decrypt Action](#).

### When to decrypt traffic

All encrypted traffic must be decrypted to take advantage of the system's threat protection and policy enforcement features. To the extent your managed device allows traffic to be decrypted (subject to its memory and processing power), you should decrypt traffic that is not prohibited by law or regulation. If you must decide what traffic to decrypt, base your decision on the risk of allowing the traffic on your network. The system provides a flexible framework for classifying traffic using rule conditions, which include URL reputation, cipher suite, protocol, and many other factors.

### Related Topics

[Decrypt and Resign \(Outgoing Traffic\)](#), on page 11  
[Known Key Decryption \(Incoming Traffic\)](#), on page 11  
[TLS/SSL Rule Guidelines and Limitations](#)  
[SSL Rule Order](#)

[URL Conditions \(URL Filtering\)](#)  
[Application Rule Order](#)  
[TLS 1.3 Decryption Best Practices](#)

## Decrypt and Resign (Outgoing Traffic)

The **Decrypt - Resign** TLS/SSL rule action enables the system to act as a man in the middle, intercepting, decrypting, and (if the traffic is allowed to pass) inspecting and re-encrypting it. The **Decrypt - Resign** rule action is used with outgoing traffic; that is, the destination server is outside your protected network.

The FTD device negotiates with the client using an internal Certificate Authority (CA) object specified in the rule and builds a TLS/SSL tunnel between the client and the FTD device. At the same time, the device connects to the destination web site and creates an SSL tunnel between the server and the FTD device.

Thus, the client sees the CA certificate configured for the TLS/SSL rule instead of the certificate from the destination server. The client must trust the firewall's certificate to complete the connection. The FTD device then performs decryption/re-encryption in both directions for traffic between the client and the destination server.

### Prerequisite

To use the **Decrypt - Resign** rule action, you must create an internal CA object using a CA file and paired private key file. You can generate a CA and private key in the system if you don't already have them.



**Note** The Firepower System does not support mutual authentication; that is, you cannot upload a [client certificate](#) to the Firepower Management Center and use it for either **Decrypt - Resign** or **Decrypt - Known Key** TLS/SSL rule actions. For more information, see [Decrypt and Resign \(Outgoing Traffic\)](#), on page 11. and [Known Key Decryption \(Incoming Traffic\)](#), on page 11.

### Related Topics

[TLS/SSL Rule Decrypt Actions](#)  
[External Certificate Objects](#)

## Known Key Decryption (Incoming Traffic)

The **Decrypt - Known Key** TLS/SSL rule action uses a server's private key to decrypt traffic. The **Decrypt - Known Key** rule action is used with incoming traffic; that is, the destination server is inside your protected network.

The main purpose of decrypting with a known key is to protect your servers from external attacks.

### Prerequisite

To use the **Decrypt - Known Key** rule action, you must create an internal certificate object using the server's certificate file and paired private key file.



**Note** The Firepower System does not support mutual authentication; that is, you cannot upload a [client certificate](#) to the Firepower Management Center and use it for either **Decrypt - Resign** or **Decrypt - Known Key** TLS/SSL rule actions. For more information, see [Decrypt and Resign \(Outgoing Traffic\)](#), on page 11. and [Known Key Decryption \(Incoming Traffic\)](#), on page 11.

**Related Topics**

[Known Key Decryption \(Incoming Traffic\)](#), on page 11  
[TLS/SSL Rule Decrypt Actions](#)  
[Internal Certificate Objects](#)

## Other TLS/SSL Rule Actions

The following sections discuss other TLS/SSL rule actions.

**Related Topics**

[TLS/SSL Rule Blocking Actions](#)  
[TLS/SSL Rule Monitor Action](#)

## TLS/SSL Rule Components

Each TLS/SSL rule has the following components.

**State**

By default, rules are enabled. If you disable a rule, the system does not use it to evaluate network traffic, and stops generating warnings and errors for that rule.

**Position**

Rules in an SSL policy are numbered, starting at 1. The system matches traffic to rules in top-down order by ascending rule number. With the exception of Monitor rules, the first rule that traffic matches is the rule that handles that traffic.

**Conditions**

Conditions specify the specific traffic the rule handles. Conditions can match traffic by security zone, network or geographical location, VLAN, port, application, requested URL, user, certificate, certificate subject or issuer, certificate status, cipher suite, or encryption protocol version. The use of conditions can depend on target device licenses.

**Action**

A rule's action determines how the system handles matching traffic. You can monitor, allow, block, or decrypt encrypted matching traffic. Decrypted and allowed encrypted traffic is subject to further inspection. Note that the system does *not* perform inspection on blocked encrypted traffic.

**Logging**

A rule's logging settings govern the records the system keeps of the traffic it handles. You can keep a record of traffic that matches a rule. You can log a connection when the system blocks an encrypted session or allows it to pass without decryption, according to the settings in an SSL policy. You can also force the system to log connections that it decrypts for further evaluation by access control rules, regardless of how the system later handles or inspects the traffic. You can log connections to the Firepower Management Center database, as well as to the system log (syslog) or to an SNMP trap server.

For more information about logging, see Best Practices for Connection Logging in the [Firepower Management Center Administration Guide](#).



**Tip** Properly creating and ordering TLS/SSL rule is a complex task. If you do not plan your policy carefully, rules can preempt other rules, require additional licenses, or contain invalid configurations. To help ensure that the system handles traffic as you expect, the SSL policy interface has a robust warning and error feedback system for rules.

### Categories

For information about using TLS/SSL rule categories (such as Applications, Category, and Cert Status), see [TLS/SSL Rule Conditions](#).

## TLS/SSL Rule Order Evaluation

When you create the TLS/SSL rule in an SSL policy, you specify its position using the **Insert** list in the rule editor. TLS/SSL rules in an SSL policy are numbered, starting at 1. The system matches traffic to TLS/SSL rules in top-down order by ascending rule number.

In most cases, the system handles network traffic according to the *first* TLS/SSL rule where *all* the rule's conditions match the traffic. Except in the case of Monitor rules (which log traffic but do not affect traffic flow), the system does *not* continue to evaluate traffic against additional, lower-priority rules after that traffic matches a rule. Conditions can be simple or complex; you can control traffic by security zone, network or geographical location, VLAN, port, application, requested URL, user, certificate, certificate distinguished name, certificate status, cipher suite, or encryption protocol version.

Each rule also has an *action*, which determines whether you monitor, block, or inspect matching encrypted or decrypted traffic with access control. Note that the system does *not* further inspect encrypted traffic it blocks. It does subject encrypted and undecryptable traffic to access control. However, access control rule conditions require unencrypted traffic, so encrypted traffic matches fewer rules.

Rules that use *specific* conditions (such as network and IP addresses) should be ordered *before* rules that use general conditions (such as applications). If you're familiar with the Open Systems Interconnect (OSI) model, use similar numbering in concept. Rules with conditions for layers 1, 2, and 3 (physical, data link, and network) should be ordered first in your rules. Conditions for layers 5, 6, and 7 (session, presentation, and application) should be ordered later in your rules. For more information about the OSI model, see this [Wikipedia article](#).



**Tip** Proper TLS/SSL rule order reduces the resources required to process network traffic, and prevents rule preemption. Although the rules you create are unique to every organization and deployment, there are a few general guidelines to follow when ordering rules that can optimize performance while still addressing your needs.

In addition to ordering rules by number, you can group rules by category. By default the system provides three categories: Administrator, Standard, and Root. You can add custom categories, but you cannot delete the system-provided categories or change their order.

### Related Topics

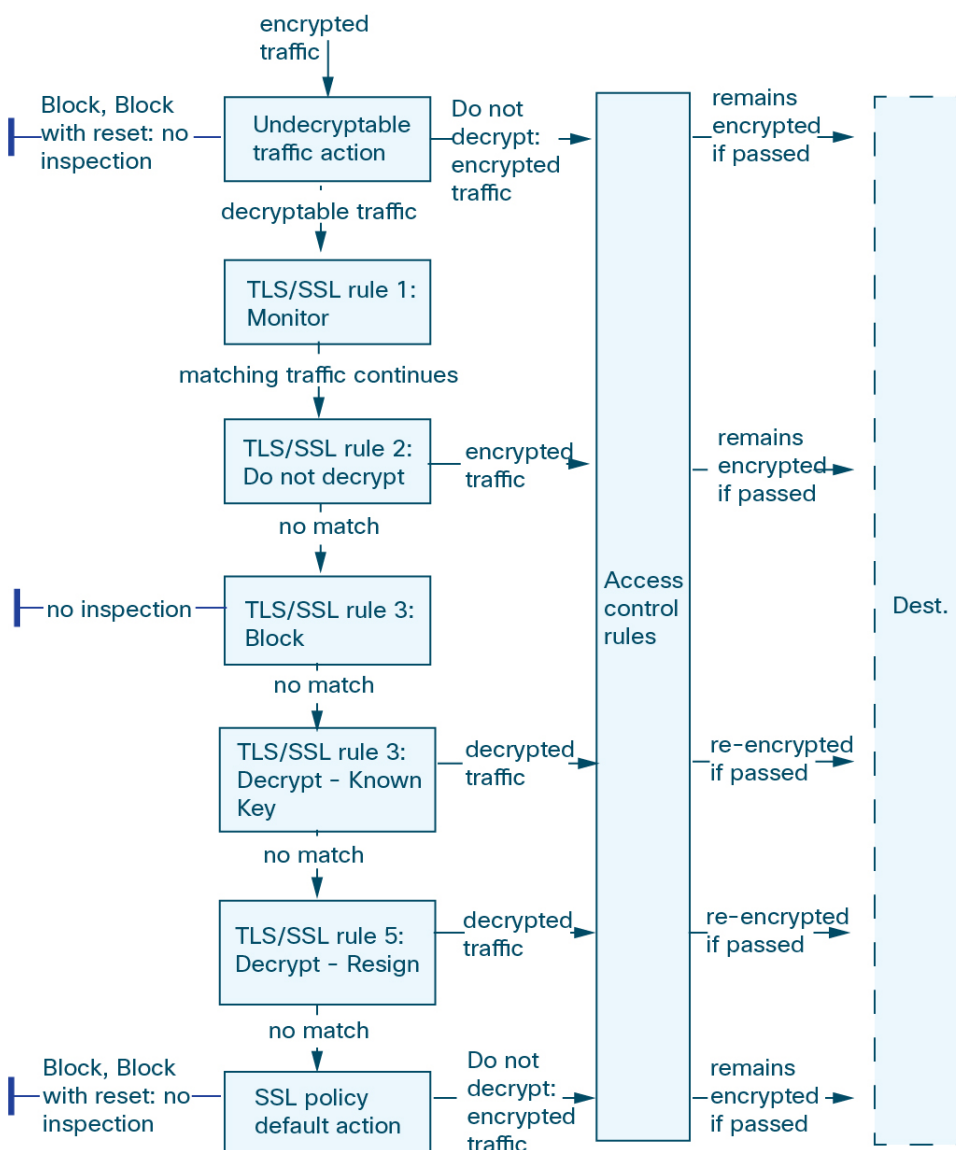
[Best Practices for Access Control Rules](#)

[Default Handling Options for Undecryptable Traffic](#)

[SSL Rule Order](#)

## Multi-Rule Example

The following scenario summarizes the ways that TLS/SSL rules handle traffic in an inline deployment.



In this scenario, traffic is evaluated as follows:

- **Undecryptable Traffic Action** evaluates encrypted traffic first. For traffic the system cannot decrypt, the system either blocks it without further inspection or passes it for access control inspection. Encrypted traffic that does not match continues to the next rule.
- **TLS/SSL Rule 1: Monitor** evaluates encrypted traffic next. Monitor rules track and log encrypted traffic but do not affect traffic flow. The system continues to match traffic against additional rules to determine whether to permit or deny it.

- **TLS/SSL Rule 2: Do Not Decrypt** evaluates encrypted traffic third. Matching traffic is not decrypted; the system inspects this traffic with access control, but not file or intrusion inspection. Traffic that does not match continues to the next rule.
- **TLS/SSL Rule 3: Block** evaluates encrypted traffic fourth. Matching traffic is blocked without further inspection. Traffic that does not match continues to the next rule.
- **TLS/SSL Rule 4: Decrypt - Known Key** evaluates encrypted traffic fifth. Matching traffic incoming to your network is decrypted using a private key you upload. The decrypted traffic is then evaluated against access control rules. Access control rules handle decrypted and unencrypted traffic identically. The system can block traffic as a result of this additional inspection. All remaining traffic is reencrypted before being allowed to the destination. Traffic that does not match the TLS/SSL rule continues to the next rule.
- **TLS/SSL Rule 5: Decrypt - Resign** is the final rule. If traffic matches this rule, the system re-signs the server certificate with an uploaded CA certificate, then acts as a man-in-the-middle to decrypt traffic. The decrypted traffic is then evaluated against access control rules. Access control rules treat decrypted and unencrypted traffic identically. The system can block traffic as a result of this additional inspection. All remaining traffic is reencrypted before being allowed to the destination. Traffic that does not match the SSL rule continues to the next rule.
- **SSL policy Default Action** handles all traffic that does not match any of the TLS/SSL rules. The default action either blocks encrypted traffic without further inspection or does not decrypt it, passing it for access control inspection.

## TLS Crypto Acceleration

TLS crypto acceleration accelerates the following:

- TLS/SSL encryption and decryption
- VPN, including TLS/SSL and IPsec

### Supported Hardware

The following hardware models support TLS crypto acceleration:

- Secure Firewall 3100
- Firepower 2100
- Firepower 4100/9300

For information about TLS crypto acceleration support on Firepower 4100/9300 FTD container instances, see the *FXOS Configuration Guide*.

TLS crypto acceleration is *not* supported on any virtual appliances or on any hardware except for the preceding.



#### Note

For more information about TLS crypto acceleration and the 4100/9300, see the *FXOS Configuration Guide*.

### Features Not Supported by TLS crypto acceleration

Features *not* supported by TLS crypto acceleration include the following:

- Managed devices where FTD container instance is enabled.
- If the inspection engine is configured to preserve connections and the inspection engine fails unexpectedly, TLS/SSL traffic is dropped until the engine restarts.

This behavior is controlled by the **configure snort preserve-connection {enable | disable}** command.

## TLS Crypto Acceleration Guidelines and Limitations

Keep the following in mind if your managed device has TLS crypto acceleration enabled.

### HTTP-only performance

Using TLS crypto acceleration on a managed device that is not decrypting traffic can affect performance.

### Federal Information Processing Standards (FIPS)

If TLS crypto acceleration and Federal Information Processing Standards (FIPS) are both enabled, connections with the following options fail:

- RSA keys less than 2048 bytes in size
- Rivest cipher 4 (RC4)
- Single Data Encryption Standard (single DES)
- Merkle–Damgård 5 (MD5)
- SSL v3

FIPS is enabled when you configure the FMC and managed devices to operate in a security certifications compliance mode. To allow connections when operating in those modes, you can configure web browsers to accept more secure options.

For more information:

- Ciphers supported by FIPS: [About SSL Settings](#).
- [Security Certifications Compliance Modes](#).
- [Common Criteria](#).

### TLS heartbeat

Some applications use the *TLS heartbeat* extension to the Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) protocols defined by [RFC6520](#). TLS heartbeat provides a way to confirm the connection is still alive—either the client or server sends a specified number of bytes of data and requests the other party echo the response. If this is successful, encrypted data is sent.

When a managed device with TLS crypto acceleration enabled encounters a packet that uses the TLS heartbeat extension, the managed device takes the action specified by the setting for **Decryption Errors** in the SSL policy's **Undecryptable Actions**:



- Block
- Block with reset

For more information, see [Default Handling Options for Undecryptable Traffic](#).

To determine whether applications are using TLS heartbeat, see [Troubleshoot TLS Heartbeat](#).

You can configure a **Max Heartbeat Length** in a Network Analysis Policy (NAP) to determine how to handle TLS heartbeats. For more information, see [The SSL Preprocessor](#).

### TLS/SSL oversubscription

*TLS/SSL oversubscription* is a state where a managed device is overloaded with TLS/SSL traffic. Any managed device can experience TLS/SSL oversubscription but only managed devices that support TLS crypto acceleration provide a configurable way to handle it.

When a managed device with TLS crypto acceleration enabled is oversubscribed, any packet received by the managed device is acted on according to the setting for **Handshake Errors** in the SSL policy's **Undecryptable Actions**:

- Inherit default action
- Do not decrypt
- Block
- Block with reset

If the setting for **Handshake Errors** in the SSL policy's **Undecryptable Actions** is **Do Not decrypt** and the associated access control policy is configured to inspect the traffic, inspection occurs; decryption does *not* occur.

If a significant amount of oversubscription is occurring, you have the following options:

- Upgrade your managed devices to increase TLS/SSL processing capacity.
- Change your SSL policies to add **Do Not Decrypt** rules for traffic that is not a high priority to decrypt.

## View the Status of TLS Crypto Acceleration

This topic discusses how you can determine if TLS crypto acceleration is enabled.

Perform the following task in the FMC.

### Procedure

- 
- |               |   |
|---------------|---|
| <b>Step 1</b> | Log in to the FMC.  |
| <b>Step 2</b> | Click <b>Devices &gt; Device Management</b> .   |
| <b>Step 3</b> | Click <b>Edit</b> (✎) to edit a managed device.   |
| <b>Step 4</b> | Click <b>Device</b> page. TLS crypto acceleration status is displayed in the General section. |
-

# How to Configure SSL Policies and Rules

This topic provides a high-level overview of tasks you must complete to configure SSL policies and TLS/SSL rules in those policies to block, monitor, or allow TLS/SSL traffic on your network.

You must be an Admin, Access Admin, or Network Admin to perform this task.

## Procedure

|               | Command or Action  | Purpose   |
|---------------|--|---|
| <b>Step 1</b> | Create an SSL policy.  | An SSL policy is a container for one or more rules. To use an SSL policy and its rules for access control, you must later associate the SSL policy with an access control policy. For more information, see <a href="#">Create Basic SSL Policies</a> .                             |
| <b>Step 2</b> | Set a default action for your SSL policy.  | The default action is taken when traffic matches no rules defined by the SSL policy. See <a href="#">SSL Policy Default Actions</a> .   |
| <b>Step 3</b> | Specify how undecryptable traffic should be handled.   | Traffic can be undecryptable for a number of reasons, including unsecure protocols, uses and unknown cipher suite, or in the event of errors with the handshake or decryption. See <a href="#">Default Handling Options for Undecryptable Traffic</a> .                             |
| <b>Step 4</b> | For <b>Decrypt - Known Key</b> (to decrypt inbound traffic to a server in your network) TLS/SSL rules, create an internal certificate object.                      | The internal certificate object uses your server's certificate and private key. See <a href="#">Internal Certificate Objects</a> .  |
| <b>Step 5</b> | For <b>Decrypt - Resign</b> (to decrypt outbound traffic to a server outside of your network) TLS/SSL rules, create an internal certificate authority (CA) object. | The internal CA object uses a CA and private key. See <a href="#">Internal Certificate Authority Objects</a> .  |
| <b>Step 6</b> | Create your TLS/SSL rules.   |   |
| <b>Step 7</b> | Associate the SSL policy with an access control policy.  | Unless you associate your SSL policy with an access control policy, it has no effect. After you do this, you can choose to allow or block traffic that matches the access control rule and take other actions. See <a href="#">Associating Other Policies with Access Control</a> . |
| <b>Step 8</b> | Configure your access control rules to allow or block decrypted traffic.   | See <a href="#">Access Control Policy Components</a> .  |
| <b>Step 9</b> | Deploy the access control policy to managed devices.   | Before your policy can take effect, it must be deployed to managed devices. See <a href="#">Deploy Configuration Changes</a> .  |

## Related Topics

[TLS/SSL Rules](#)

# History for SSL Policy

| Feature  | Minimum FMC | Minimum FTD | Details   |
|--|-------------|-------------|---|
| TLS 1.3 decryption.  | 7.2.0       | 7.2.0       | <p>You can now enable TLS 1.3 decryption in an SSL policy's advanced actions. TLS 1.3 decryption requires the managed device run Snort 3.</p> <p>Other options are available as well; for more information, see <a href="#">SSL Policy Advanced Options</a>.</p> <p>New/modified screens: <b>SSL Policy &gt; Advanced Settings</b></p>  |
| SSL policy advanced settings.  | 7.2.0       | 7.1.0       | <p>SSL policy advanced settings</p> <p>New/modified screens: <b>SSL Policy &gt; Advanced Settings</b></p>   |
| Ability to specify handling of URLs having unknown reputation.   | 6.7.0       | 6.7.0       | For details, see <a href="#">About URL Filtering with Category and Reputation</a> .   |
| ClientHello modification for <b>Decrypt - Known</b> key rules.   | 6.7.0       | 6.7.0       | For details, see <a href="#">ClientHello Message Handling, on page 3</a> .  |
| Ability to extract the certificate in TLS 1.3 traffic to enable traffic to match URL and application criteria in access control rules. | 6.7.0       | 6.7.0       | <p>New/modified screens: <b>Policies &gt; Access Control &gt; (edit an access control policy) &gt; Advanced</b> link.</p> <p>For details, see <a href="#">SSL Policy Advanced Options</a>.</p>  |
| Changes to category and reputation-based URL Filtering.  | 6.7.0       | 6.5.0       | For details, see <a href="#">About URL Filtering with Category and Reputation</a> .   |
| TLS crypto acceleration cannot be disabled.  | 6.4.0       | 6.4.0       | <p>TLS crypto acceleration is enabled on all supported devices.</p> <p>On a managed device with native interfaces, TLS crypto acceleration cannot be disabled.</p> <p>Support for TLS crypto acceleration on FTD container instances is limited as discussed in the next row of this table.</p> <p>Removed commands:</p> <ul style="list-style-type: none"> <li>• <b>system support ssl-hw-accel enable</b></li> <li>• <b>system support ssl-hw-accel disable</b></li> <li>• <b>system support ssl-hw-status</b></li> </ul> |

| Feature  | Minimum FMC | Minimum FTD | Details   |
|--|-------------|-------------|---|
| Support for TLS crypto acceleration on one FTD container instance on a Firepower 4100/9300 module/security engine. | 6.4.0       | 6.4.0       | <p>You can now enable TLS crypto acceleration for one FTD container instance on a module/security engine. TLS crypto acceleration is disabled for other container instances, but enabled for native instances.</p> <p>New/modified commands:</p> <ul style="list-style-type: none"> <li>• <b>config hwCrypto enable</b></li> <li>• <b>show crypto accelerator status</b> replaces <b>system support ssl-hw-status</b>)</li> </ul> |
| TLS/SSL hardware acceleration is now referred to as <i>TLS crypto acceleration</i> .                               | 6.4.0       | 6.4.0       | <p>The name change reflects that TLS/SSL encryption and decryption acceleration is supported on more devices. Depending on the device, acceleration might be performed in software or in hardware.</p> <p>New/modified screens: <b>Devices &gt; Device Management &gt; Edit &gt; Device &gt; General &gt; TLS Crypto Acceleration</b></p>   |
| Extended Master Secret extension supported (see <a href="#">RFC 7627</a> ).  | 6.3.0.1     | 6.3.0.1     | The TLS Extended Master Secret extension is supported for SSL policies; specifically, policies with a rule action of <b>Decrypt - Resign</b> or <b>Decrypt - Known Key</b> .  |
| Extended Master Secret extension not supported.  | 6.3.0       | 6.3.0       | The extension is stripped during ClientHello modification for <b>Decrypt - Resign</b> rules.  |
| TLS/SSL hardware acceleration enabled by default.  | 6.3.0       | 6.3.0       | TLS/SSL hardware acceleration is enabled by default on all supported devices but can be disabled if desired.  |
| Extended Master Secret extension supported (see <a href="#">RFC 7627</a> ).  | 6.2.3.9     | 6.2.3.9     | The TLS Extended Master Secret extension is supported for SSL policies; specifically, policies with a rule action of <b>Decrypt - Resign</b> or <b>Decrypt - Known Key</b> .  |
| Aggressive TLS 1.3 downgrade.  | 6.2.3.7     | 6.2.3.7     | Using the <b>system support ssl-client-hello-enabled aggressive_tls13_downgrade {true/false}</b> CLI command, you can determine the behavior for downgrading TLS 1.3 traffic to TLS 1.2. For details, see the <a href="#">Cisco Secure Firewall Threat Defense Command Reference</a> .  |
| TLS/SSL hardware acceleration introduced.  | 6.2.3       | 6.2.3       | <p>Certain managed device models perform TLS/SSL encryption and decryption in hardware, improving performance. By default, the feature is enabled.</p> <p>Affected screen: To view the status of TLS/SSL hardware acceleration, <b>Devices &gt; Device Management &gt; Device</b>, General page.</p>  |
| Category and reputation conditions supported.  | 6.2.2       | 6.2.2       | Access control rules or SSL rules with category/reputation conditions.  |

| Feature               | Minimum FMC | Minimum FTD | Details  |
|-----------------------|-------------|-------------|--|
| SafeSearch supported. | 6.1.0       | 6.1.0       | <p>The system displays an HTTP response page for connections decrypted by the SSL policy, then blocked (or interactively blocked) either by access control rules or by the access control policy default action. In these cases, the system encrypts the response page and sends it at the end of the reencrypted SSL stream.</p> <p>SafeSearch filters objectionable content and stops people from searching adult sites.</p> |
| TLS/SSL policy.       | 6.0.0       | 6.0.0       | Feature introduced.  |

