

Traffic Decryption Overview

These 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.

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

- Traffic decryption, on page 1
- How TLS/SSL handshakes work, on page 3
- Decryption rule and policy basics, on page 8
- TLS crypto acceleration, on page 14
- DTLS crypto acceleration, on page 17
- History for decryption policy , on page 19

Traffic decryption

Traffic decryption enables deep inspection of traffic for analysis.

Traffic handling options

Most traffic on the internet is encrypted and in most cases, you might choose not to decrypt it. You can glean some information from encrypted traffic and you can also block it from your network if desired.

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 decryption 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 integration

An access control policy is the main configuration that invokes subpolicies and other configurations, including a decryption policy. If you associate a decryption policy with access control, the system uses that decryption 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.

Supported protocols for decryption:

- TLS versions up to 1.3 (if Snort 3 is enabled)
- DNS over HTTPS
- DNS over TLS (RFC 7858)
- FTPS, SMTPS, IMAPS, POP3S



Note As long as your managed devices have Snort 3 enabled, the system supports decrypting TLS 1.3 traffic. You can enable TLS 1.3 decryption in an decryption policy's advanced options; for more information, see Decryption policy advanced options.

Notes

Decryption rules require processing overhead that can impact performance. Before you decide to decrypt traffic, see Best practice for deciding when to decrypt traffic, on page 9.

The system does not support mutual authentication; that is, you cannot upload a client certificate to the Secure Firewall Management Center and use it for **Decrypt - Resign** or **Decrypt - Known Key** decryption rule actions.

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 TCP MSS.

How TLS/SSL handshakes work

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 figure shows an inline deployment.



Summary

The key components involved in the TLS/SSL handshake process are:

- Client: Initiates the encrypted session by sending a ClientHello message specifying multiple supported values for each encryption attribute
- Server: Responds with a ServerHello message specifying a single supported value for each encryption attribute, determining the encryption method for the secure session
- Managed device: Monitors TCP sessions for encrypted session initiation attempts and can act as a TCP proxy server
- Handshake messages: Specialized packets that communicate encryption attributes between client and server

Workflow

These stages describe how TLS/SSL handshakes establish encrypted sessions:

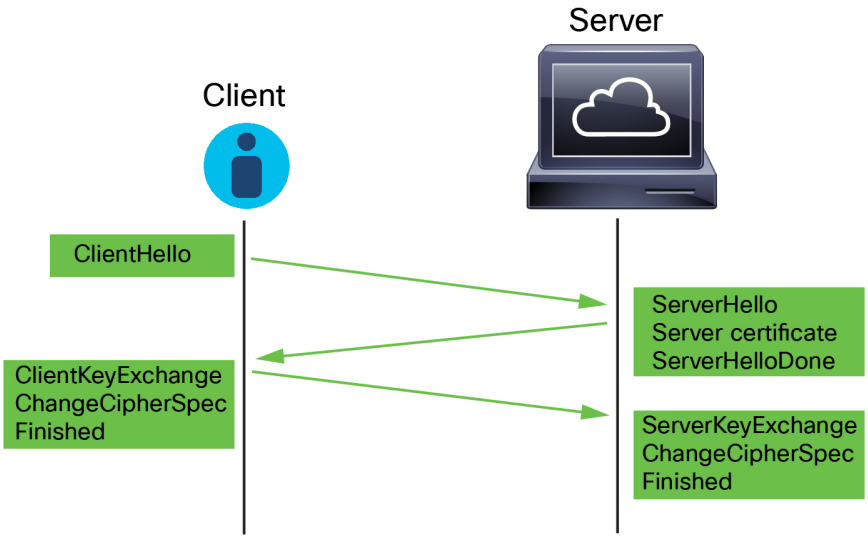
1. The client establishes a TCP connection with the server after successfully completing the TCP three-way handshake.
2. The managed device monitors the TCP session for any attempt to initiate an encrypted session.
3. The client and server exchange specialized handshake messages to establish an encrypted session.
   - 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.
4. After the TLS/SSL handshake completes, the managed device caches encrypted session data and server certificate data.

   This caching allows session resumption without requiring the full handshake and enables faster handshake processing in subsequent sessions that use the same certificate.

ClientHello message handling

The client sends the ClientHello message to the server, which acts as the packet destination, when a secure connection can be established. The client uses this message to initiate the TLS/SSL handshake to respond to a ServerHello message from the destination server.

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



ClientHello message process

The process consists of these steps:

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 names.

3. When the managed device receives these messages, it attempts to match them with decryption 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 decryption rules' Distinguished Names, Certificate Status, Cipher Suites, and Versions conditions.

The entire process is 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 decryption 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. The table lists the possible types of data available for matching conditions.

Table 1: Data Availability for Decryption rule Conditions

Decryption rule Condition	Data Present In
Zones	ClientHello
Networks	ClientHello
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 **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 message modifications

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

- (TLS 1.2 only; TLS 1.3 does not support compression.) 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 improves the likelihood of session resumption and reduces the Session Not Cached incidents 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.

  Decryption 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 not possible during the TLS/SSL handshake. After the message is modified, 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. Each session contains distinct 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 decryption rule conditions. For the current list of decryptable cipher suites, contact Cisco TAC .

How ServerHello and server certificate messages work

Summary

The key components involved in ServerHello and server certificate message handling are:

- Managed device: Processes ServerHello and server certificate messages and performs rule matching
- decryption rules: Define actions to take when messages match specific criteria
- Server certificate data: Certificate information used for matching against internal certificate objects

- TLS/SSL handshake: The secure connection establishment process
- Certificate authority (CA): Used for re-signing certificates in decrypt-resign actions

Workflow

These stages describe how ServerHello and server certificate message handling works:

1. The managed device receives ServerHello and server certificate messages and checks for rule matches.

   If the messages do not match any decryption rules, the managed device performs the Decryption policy default actions.

2. When messages match a rule that belongs to a decryption policy associated with an access control policy, the managed device performs actions based on the rule configuration.
   - **Monitor**: The TLS/SSL handshake continues to completion. The managed device tracks and logs traffic but does not decrypt it.
   - **Block or Block with Reset**: The managed device blocks the TLS/SSL session and, if configured, resets the TCP connection.
   - **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.

3. For decrypt actions with known key or resign methods, the managed device performs certificate matching and processing.
   - **Decrypt - Known Key**: The managed device attempts to match the server certificate data to an Internal Certificate object previously imported into the Secure Firewall 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.
   - **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.

   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 Secure Firewall Management Center for future connections to be decrypted.



Note The system does not support mutual authentication; that is, you cannot upload a client certificate to the Secure Firewall Management Center and use it for **Decrypt - Resign** or **Decrypt - Known Key** decryption rule actions.

Decryption rule and policy basics

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

Key information for decryption policy creation

Consider this information when creating your decryption 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.

For more information about SSL and TLS protocols, refer to a resource such as SSL vs. TLS - What's the Difference?

Guidelines for traffic decryption

Traffic that is encrypted when it passes through the system can be allowed or blocked, 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 decryption policy to determine if they should be decrypted or blocked.

You can also configure decryption 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, including:

- 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.

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.

Detects encrypted application traffic using StartTLS (SMTPS, POPS, FTPS, TelnetS, IMAPS). Also identifies certain encrypted applications based on Server Name Indication in the TLS ClientHello message or subject distinguished name from the server certificate.

Best practice for deciding when to decrypt traffic

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.) Decryption 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 decryption 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 decryption 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)
- URL category and reputation
- Port
- User group

Decryption rules provide a **Do Not Decrypt** action for traffic you do not want to decrypt; for more information, see Decryption 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 Decryption 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. If your managed device has sufficient memory and processing power to decrypt traffic, 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.

Decrypt and re-sign (outgoing traffic)

Decrypt and re-sign of traffic 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 traffic. It is used with outgoing traffic where the destination server is outside your protected network and requires the client to trust the firewall's certificate to complete the connection.

How decrypt and re-sign works

The Firewall Threat Defense 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 Firewall Threat Defense device. At the same time, the device connects to the destination web site and creates a TLS/SSL tunnel between the server and the Firewall Threat Defense device.

Thus, the client sees the CA certificate configured for the decryption rule instead of the certificate from the destination server. The client must trust the firewall's certificate to complete the connection. The Firewall Threat Defense 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.

We simplify the process of configuring the **Decrypt - Resign** rule by automatically adding **Do Not Decrypt** rules to the decryption policy based on your choices for types of traffic to exempt from encryption, such as undecryptable applications. (Typically, these applications use certificate pinning so that decrypting the connection breaks the connection.) For more information, see Create a decryption policy with outbound connection protection.

Note The system does not support mutual authentication; that is, you cannot upload a client certificate to the Secure Firewall Management Center and use it for **Decrypt - Resign** or **Decrypt - Known Key** decryption rule actions.

Known key decryption for incoming traffic

The **Decrypt - Known Key** decryption rule action uses a server's private key to decrypt traffic. The **Decrypt - Known Key** rule action is used with incoming traffic, where 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.

Prerequisites and requirements

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 system does not support mutual authentication; that is, you cannot upload a client certificate to the Secure Firewall Management Center and use it for **Decrypt - Resign** or **Decrypt - Known Key** decryption rule actions.

Other decryption rule actions

These topics discuss other decryption rule actions.

Decryption rule components

Each decryption rule has the following components.

State

A rule can have either of two states: enabled or disabled. 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 a decryption policy are numbered, starting at 1. The system matches traffic to rules in top-down order by ascending rule number.

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 category and reputation, 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 traffic is subject to further inspection. Note that the system does *not* inspect 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 decryption 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 Secure Firewall 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 *Cisco Secure Firewall Management Center Administration Guide*.

Tip Properly creating and ordering decryption rules 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 decryption policy interface has a robust warning and error feedback system for rules.

Categories

For information about using decryption rule categories (such as Applications, Category, and Cert Status), see Decryption Rule conditions.

Decryption Rule order evaluation

Decryption Rule order evaluation determines the sequence in which the system evaluates decryption rules in a policy.

Rule evaluation process

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

In most cases, the system handles network traffic according to the *first* decryption 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 category and reputation, 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.



Note Proper decryption 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.

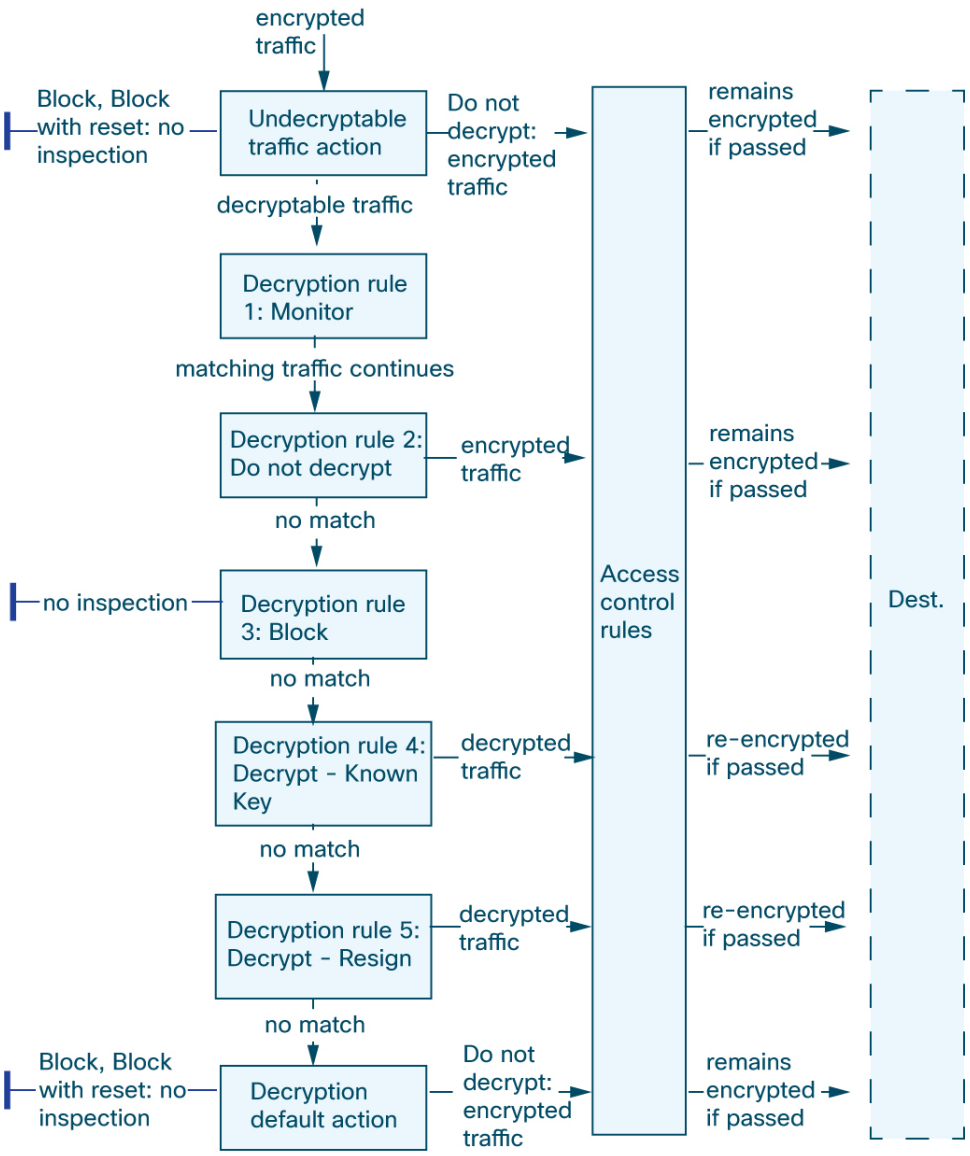
Related Topics

Best practices for access control rules

Multi-rule example

This reference example demonstrates how the firewall handles traffic using a decryption policy that contains multiple rules.

The following scenario summarizes the ways that decryption 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.
- **Decryption 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.
- **Decryption 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.

- **Decryption rule 3: Block** evaluates encrypted traffic fourth. Matching traffic is blocked without further inspection. Traffic that does not match continues to the next rule.
- **Decryption 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 decryption rule continues to the next rule.
- **Decryption rule 5: Decrypt - Resign** is the final rule. If traffic matches, the system re-signs the server certificate with an uploaded CA certificate and decrypts traffic. The decrypted traffic is then evaluated against access control rules, which treat decrypted and unencrypted traffic the same. This additional inspection allows the system to block traffic. All remaining traffic is re-encrypted before being sent to its destination. Traffic that does not match the decryption rule rule continues to the next rule.
- **Decryption policy Default Action** handles all traffic that does not match any of the decryption 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 and acceleration details

The following hardware models support TLS crypto acceleration:

- Secure Firewall 3100
- Secure Firewall 4200
- Firepower 4100/9300

  For information about TLS crypto acceleration support on Firepower 4100/9300 Firewall Threat Defense 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 include:

- Managed devices where Firewall Threat Defense 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

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–Damgard 5 (MD5)
- SSL v3

FIPS is enabled when you configure the Firewall Management Center 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: Configure 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 decryption 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, refer to *The SSL Preprocessor* section in the "Custom Snort 2 Intrusion Policies for Access Control" document.

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 decryption policy's **Undecryptable Actions**:

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

If the setting for **Handshake Errors** in the decryption 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 these options:

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

Keep these guidelines in mind if your managed device has TLS crypto acceleration enabled.

View the status of TLS crypto acceleration

This task allows you to determine if TLS crypto acceleration is enabled on your managed devices.

Perform this task in the Firewall Management Center to check the TLS crypto acceleration status for your managed devices.

Procedure

Step 1 Log in to the Firewall Management Center.

Step 2 Click **Devices** > **Device Management**.

Step 3 Click **Edit** (✎) to edit a managed device.

Step 4 Click **Device** page.

TLS crypto acceleration status is displayed in the General section.

DTLS crypto acceleration

Firewall Threat Defense supports the Datagram Transport Layer Security (*DTLS*) cryptographic acceleration for specific firewall models. This feature improves the throughput of the DTLS-encrypted and DTLS-decrypted traffic for both IPv4 and IPv6 traffic. The Firewall Threat Defense also performs optimization of the egress-encrypted packets to improve latency. The data path is optimized to enhance performance for single tunnel flows. Both features are enabled by default and work only for DTLS 1.2.

Supported models

- Secure Firewall 3100
- Secure Firewall 4200

Crypto accelerator details

In Secure Firewall 3100 and 4200, FPGA and the Nitrox V crypto accelerator support DTLS cryptographic acceleration

Limitations

These features are not supported with:

- Flows that use DTLS 1.0 or packet compression
- Rekeyed DTLS keys
- Clustering or multi-instance mode

You can disable DTLS crypto acceleration using FlexConfig and the **no flow-offload-DTLS** and the **no flow-offload-DTLS egress-optimization** commands..

Prerequisites for DTLS crypto acceleration

Before you implement DTLS crypto acceleration, make sure you understand this information.

Supported license

- Management center Essentials (formerly Base) license must allow export-controlled functionality.

  Choose **System** (⚙) > **Licenses** > **Smart Licenses** to verify this functionality in the management center.

Supported versions

- Firewall Management Center must be Version 7.6.0 and later.
- Cisco Secure Firewall 4200 and 3100 series devices must be Version 7.6.0 and later.

Supported devices

- Secure Firewall 3100

- Secure Firewall 4200

Monitor DTLS crypto acceleration

Use CLI commands on the threat defense device to verify and monitor DTLS crypto acceleration and optimization of egress-encrypted packets.

- To verify the status of DTLS crypto acceleration and optimization of egress-encrypted packets, use this command:

```
> show flow-offload-dtls info
  DTLS offload : Enabled
  Egress Optimization: Enabled
```

- To view the DTLS crypto acceleration statistics, use this command:

```
> show flow-offload-dtls statistics
  Packet stats of Pipe 0
  ----------------------
  Rx Packet count : 975638666
  Tx Packet count : 975638666
  Error Packet count : 0
  Drop Packet count : 0

  CAM stats of Pipe 0
  -----------------------
  Option ID Table CAM Hit Count : 1145314723
  Option ID Table CAM Miss Count : 0
  Tunnel Table CAM Hit Count : 0
  Tunnel Table CAM Miss Count : 0
  6-Tuple CAM Hit Count : 975638666
  6-Tuple CAM Miss Count : 169676057
   NOTE: The counters displayed are cumulative counters
   for all offload applications and indicates the total packets
  offloaded
```

- To view the device's Nitrox V crypto accelerator statistics for Secure Firewall 3100 and 4200, use this command:

```
> show crypto accelerator statistics

Crypto Accelerator Status
-------------------------
<snip>
[Offloaded SSL Input statistics, Pipe 0]
      Input packets: 290593023
      Input bytes: 147049729714
      Decrypted packets: 290593023
      Decrypted bytes: 147049729714
   [Offloaded SSL Output statistics, Pipe 0]
      Output packets: 254271808
      Output bytes: 136352952720
      Encrypted packets: 254271808
      Encrypted bytes: 136352952720
.
.
.
```

History for decryption policy

Provides a chronological overview of decryption policy features, including version support and configuration changes.

Feature	Minimum Firewall Management Center	Minimum Firewall Threat Defense	Details
Hardware DTLS 1.2 crypto acceleration for the Secure Firewall 3100/4200.	7.6.0	7.6.0	The Secure Firewall 3100/4200 now supports DTLS 1.2 cryptographic acceleration and egress optimization, which improves throughput of DTLS-encrypted and decrypted traffic. This is automatically enabled on new and upgraded devices. To disable, use FlexConfig. New/modified FlexConfig commands: **flow-offload-dtls**, **flow-offload-dtls egress-optimization**, **show flow-offload-dtls**
QUIC decryption.	7.6.0 (experimental)	7.6.0 with Snort 3	You can configure the decryption policy to apply to sessions running on the QUIC protocol. QUIC decryption is disabled by default. You can selectively enable QUIC decryption per decryption policy and write decryption rules to apply to QUIC traffic. By decrypting QUIC connections, the system can then inspect the connections for intrusion, malware, or other issues. You can also apply granular control and filtering of decrypted QUIC connections based on specific criteria in the access control policy. We modified the decryption policy Advanced Settings to include the option to enable QUIC decryption.
Easily bypass decryption for sensitive and undecryptable traffic.	7.6.0	7.6.0	It is now easier to bypass decryption for sensitive and undecryptable traffic, which protects users and improves performance. New decryption policies now include predefined rules that, if enabled, can automatically bypass decryption for sensitive URL categories (such as finance or medical), undecryptable distinguished names, and undecryptable applications. Distinguished names and applications are undecryptable typically because they use TLS/SSL certificate pinning, which is itself not decryptable. For outbound decryption, you enable/disable these rules as part of creating the policy. For inbound decryption, the rules are disabled by default. After the policy is created, you can edit, reorder, or delete the rules entirely. New/modified screens: **Policies** > **Access Control** > **Decryption** > **Create Decryption Policy**

Feature	Minimum Firewall Management Center	Minimum Firewall Threat Defense	Details
Decryption policy.	7.3.0	7.3.0	Feature renamed to *decryption policy* to better reflect what it does. We now enable you to configure a decryption policy with one or more **Decrypt - Resign** or **Decrypt - Known Key** rules at the same time. New/modified screens: • **Policies** > **Access Control heading** > **Decryption** (create new decryption policy) • The Create Decryption Policy dialog box now has two tab pages: **Outbound Connections** and **Inbound Connections** . • **Policies** > **Access Control heading** > **Decryption** (edit a decryption rule) **Advanced Settings** has new options discussed in TLS 1.3 decryption best practices . • **Policies** > **Access Control Heading** > **Access Control** (edit an access control policy), click the word **Decryption** to associate a decryption policy with an access control policy.
TLS 1.3 decryption.	7.2.0	7.2.0	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. Other options are available as well; for more information, see TLS 1.3 decryption best practices . New/modified screens: **SSL Policy** > **Advanced Settings**
SSL policy advanced settings.	7.2.0	7.1.0	SSL policy advanced settings New/modified screens: **SSL Policy** > **Advanced Settings**
Ability to specify handling of URLs having unknown reputation.	6.7.0	6.7.0	For details, see URL filtering with category and reputation .
ClientHello modification for **Decrypt - Known** key rules.	6.7.0	6.7.0	For details, see ClientHello message handling, on page 4 .
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	New/modified screens: **Policies** > **Access Control** > **(edit an access control policy)** > **Advanced** link. For details, see Decryption policy advanced options .
Changes to category and reputation-based URL Filtering.	6.7.0	6.5.0	For details, see URL filtering with category and reputation .

Feature	Minimum Firewall Management Center	Minimum Firewall Threat Defense	Details
TLS crypto acceleration cannot be disabled.	6.4.0	6.4.0	TLS crypto acceleration is enabled on all supported devices. On a managed device with native interfaces, TLS crypto acceleration cannot be disabled. Removed commands: • **system support ssl-hw-accel enable** • **system support ssl-hw-accel disable** • **system support ssl-hw-status**
Support for TLS crypto acceleration on one container instance on a Firepower 4100/9300 module/security engine.	6.4.0	6.4.0	You can now enable TLS crypto acceleration for one container instance on a module/security engine. New/modified commands: • **config hwCrypto enable** • **show crypto accelerator status**
TLS/SSL hardware acceleration is now referred to as TLS crypto acceleration .	6.4.0	6.4.0	The name change reflects that TLS/SSL encryption and decryption acceleration is supported on more devices. New/modified screens: **Devices** > **Device Management** > **Edit** > **Device** > **General** > **TLS Crypto Acceleration**
Extended Master Secret extension supported (see RFC 7627).	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 **Decrypt - Resign** or **Decrypt - Known Key** .
Extended Master Secret extension not supported.	6.3.0	6.3.0	The extension is stripped during ClientHello modification for **Decrypt - Resign** 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 RFC 7627).	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 **Decrypt - Resign** or **Decrypt - Known Key** .
Aggressive TLS 1.3 downgrade.	6.2.3.7	6.2.3.7	Using the **system support ssl-client-hello-enabled aggressive_tls13_downgrade {true|false}** CLI command, you can determine the behavior for downgrading TLS 1.3 traffic to TLS 1.2.
TLS/SSL hardware acceleration introduced.	6.2.3	6.2.3	Certain managed device models perform TLS/SSL encryption and decryption in hardware, improving performance. By default, the feature is enabled. Affected screen: To view the status of TLS/SSL hardware acceleration , **Devices** > **Device Management** > **Device** , General page.

Feature	Minimum Firewall Management Center	Minimum Firewall Threat Defense	Details
Category and reputation conditions supported.	6.2.2	6.2.2	Access control rules or SSL rules with category/reputation conditions.
SafeSearch supported.	6.1.0	6.1.0	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. SafeSearch filters objectionable content and stops people from searching adult sites.
TLS/SSL policy.	6.0.0	6.0.0	Feature introduced.