

Rule-Based Decryption Rules

The following topics discuss how to manage rule-based decryption rules.

- Rule-based decryption, on page 1
- Requirements and prerequisites for Rule-based decryption rules, on page 1
- Rule-based decryption rules guidelines and limitations, on page 2
- Best practice for Decryption policy rule order, on page 10
- Rule-based decryption rule traffic handling, on page 11
- Rule-based decryption rule conditions, on page 15
- Rule-based decryption rule actions, on page 32
- Monitor TLS/SSL hardware acceleration, on page 38
- Troubleshoot Rule-based decryption rules, on page 40
- Troubleshoot using crypto archives, on page 48

Rule-based decryption

Decryption rules are a granular method that

- handle encrypted traffic across multiple managed devices
- provide options to block traffic without inspection, inspect without decrypting, or decrypt for access control analysis, and
- support access control rule evaluation.

Requirements and prerequisites for Rule-based decryption rules

This reference provides the domain support and user role requirements necessary for Rule-based decryption rules.

Supported domains

Any

User roles

- Administrator
- Access Admin
- Network Admin

Rule-based decryption rules guidelines and limitations

Keep these points in mind when setting up your rule-based decryption rule. Properly configuring rule-based decryption rule is a complex task, but one that is essential to building an effective deployment that handles encrypted traffic. Many factors influence how you configure rules, including certain application behavior that you cannot control.

In addition, rules can preempt each other, require additional licenses, or contain invalid configurations. Thoughtfully configured rules can also reduce the resources required to process network traffic. Creating overly complex rules and ordering rules the wrong way can adversely affect performance.

For detailed information, see Best practices for access control rules.

Related Topics

Best practices for access control rules
Guidelines for using TLS/SSL decryption, on page 2
Rule-based decryption rules unsupported features, on page 4
Guidelines for TLS/SSL Do Not Decrypt configuration, on page 4
Best practices for TLS/SSL incoming traffic decryption, on page 5
TLS/SSL Decrypt - Resign guidelines, on page 6
TLS/SSL block guidelines, on page 8
TLS/SSL certificate pinning guidelines, on page 8
TLS/SSL heartbeat guidelines, on page 9
TLS/SSL anonymous cipher suite limitation, on page 9
TLS/SSL normalizer guidelines, on page 9
Other Rule-based decryption rule guidelines, on page 9
Rule and other policy warnings
Best practice for Decryption policy rule order, on page 10

Guidelines for using TLS/SSL decryption

General guideline

Rule-based decryption rules require processing overhead that can impact performance. Determine which traffic must be decrypted and subjected to deep inspection before you set up any policies or rules.

- You cannot decrypt traffic on a device that uses passive interface.
- You cannot decrypt traffic on a device that uses inline tap interface.
- You cannot decrypt traffic on a device that uses TCP state bypass (only if the interfaces are in routed mode).

Guidelines for undecryptable traffic

We can determine that certain traffic is not decryptable either because the website itself is not decryptable or because the website uses TLS/SSL pinning, which effectively prevents users from accessing a decrypted site without errors in their browser.

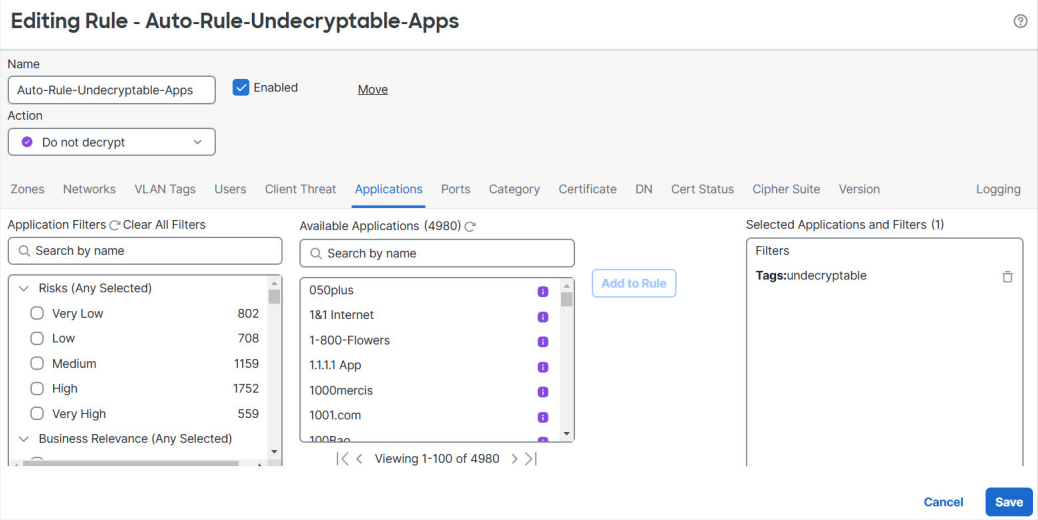
For more information about certificate pinning, see TLS/SSL pinning, on page 43.

We maintain the list of these sites as follows:

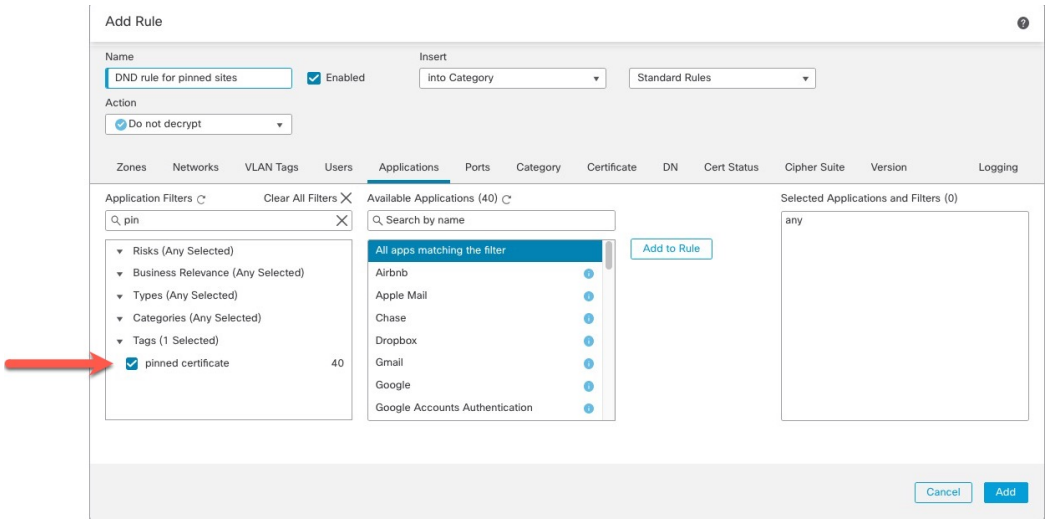
- A Distinguished Name (DN) group named **Cisco-Undecryptable-Sites**
- The **pinned certificate** or **undecryptable** application filter

If you are decrypting traffic and you do not want users to see errors in their browsers when going to these sites, we recommend you set up a **Do Not Decrypt** rule toward the bottom of your decryption rules.

If you use the decryption policy wizard to create a policy for outbound traffic protection, a **Do Not Decrypt** rule for pinned certificates is created for you as the following example shows.



An example of setting up a **pinned certificate** application filter follows.



Rule-based decryption rules unsupported features

This reference provides guidance for handling RC4, DES, and TLS 1.3 limitations in the Cisco Firepower System, including configuration requirements for decryption policies when dealing with unsupported cipher suites and interface types.

RC4 cipher unsupported

The Rivest Cipher 4 (also referred to as *RC4* or *ARC4* cipher) is known to have vulnerabilities and is considered unsecure.

Decryption policies identify the RC4 cipher suite as unsupported; you should configure the **Unsupported Cipher Suite** action in the policy's **Undecryptable Actions** page to match your organization's requirements. For more information, see Default handling options for undecryptable traffic.

Passive, inline tap mode, and SPAN interfaces not supported

TLS/SSL traffic cannot be decrypted on passive, inline tap mode, or SPAN interfaces.

Guidelines for TLS/SSL Do Not Decrypt configuration

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

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)

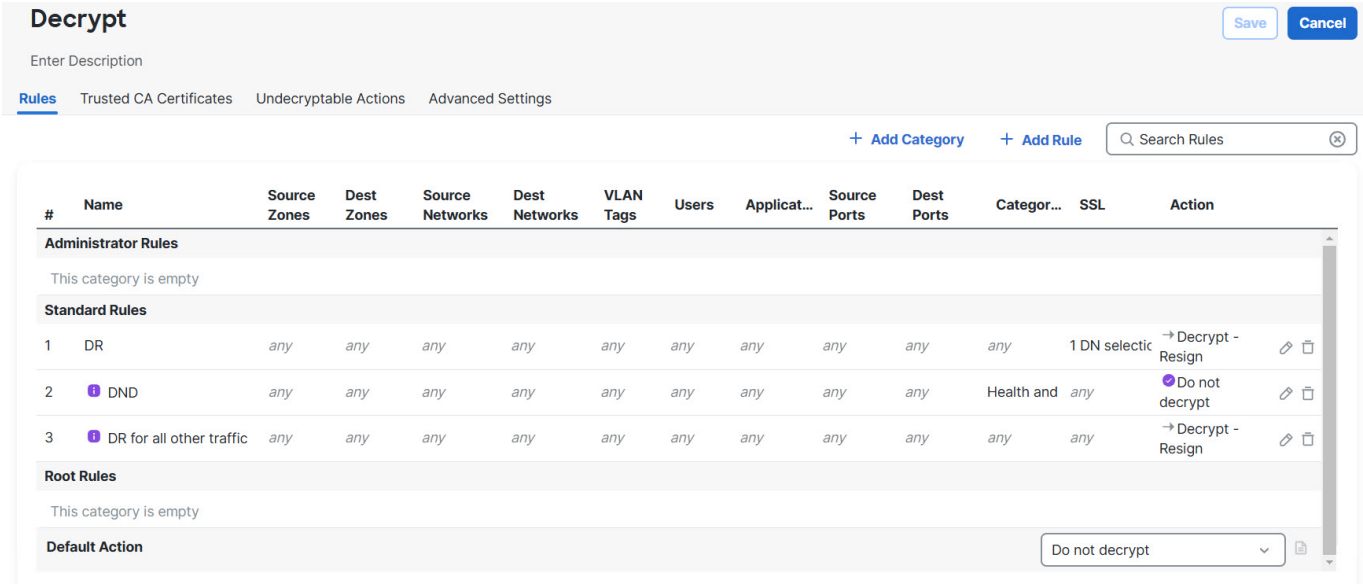
- URL category and reputation
- Port
- User group

Limitations of categories in Do Not Decrypt rules

You can optionally choose to include categories in your decryption policies. These categories, also referred to as *URL filtering*, are updated by the Cisco Talos intelligence group. Updates are based on machine learning and human analysis according to content that is retrievable from the website destination and sometimes from its hosting and registration information. Categorization is not based on the declared company vertical, intent, or security. While we strive to continuously update and improve URL filtering categories, it is not an exact science. Some websites are not categorized at all and it's possible some websites might be improperly categorized.

Avoid overusing categories in do not decrypt rules to avoid decrypting traffic without a reason; for example, the Health and Medicine category includes the WebMD website, which does not threaten patient privacy.

Following is a sample decryption policy that can prevent decryption for websites in the Health and Medicine categories but allows decryption for WebMD and everything else. General information about decryption rules can be found in Guidelines for using TLS/SSL decryption, on page 2.



Note Don't confuse URL filtering with application detection, which relies on reading some of the packet from a website to determine more specifically what it is (for example, Facebook Message or Salesforce). For more information, see Recommendations for application control.

Best practices for TLS/SSL incoming traffic decryption

Certificate and key association

Known key rule action

You can associate one or more server certificates and paired private keys with a **Decrypt - Known Key** rule action.

If traffic matches the rule, and the certificate used to encrypt the traffic matches the certificate associated with the action, the system uses the appropriate private key to obtain the session encryption and decryption keys. Because you must have access to the private key, this action is best suited to decrypt traffic incoming to servers your organization controls.

Restrictions and considerations

Note these restrictions when configuring decryption rules:

- Anonymous cipher suite unsupported:

  By nature, anonymous cipher suites are not used for authentication and do not use key exchanges. There are limited uses for anonymous cipher suites; for more information, see RFC 5246, appendix F.1.1.1. (Replaced for TLS 1.3 by RFC 8446 appendix C.5.)

  You cannot use the **Decrypt - Resign**, **Decrypt - Replace Cert** or **Decrypt - Known Key** action in the rule because anonymous cipher suites are not used for authentication.

- Cannot match on Distinguished Name or Certificate:

  You cannot match on **Distinguished Name** or **Certificate** rule conditions when creating a decryption rule with a **Decrypt - Known Key** action. The assumption is that if this rule matches traffic, the certificate, subject DN, and issuer DN already match the certificate associated with the rule.

- Do not use Version or Cipher Suite rule conditions.



Important Never use either **Cipher Suite** or **Version** rule conditions in a rule with a **Decrypt - Resign**, **Decrypt - Replace Cert**, or **Decrypt - Known Key** rule action. The use of these conditions in rules with other rule actions can interfere with the system's ClientHello processing, resulting in unpredictable performance.

- Elliptic Curve Digital Signature Algorithm (ECDSA) certificate results in blocked traffic:

  (TLS 1.3 decryption enabled only.) If you use an ECDSA certificate with a **Decrypt - Known Key** rule action, matching traffic will be blocked. To avoid this, use a certificate with another type of certificate.

TLS/SSL Decrypt - Resign guidelines

You can associate one internal Certificate Authority (CA) certificate and private key with the **Decrypt - Resign** action. If traffic matches this rule, the system re-signs the server certificate with the CA certificate, then 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 different cryptographic session details, and allows the system to decrypt and reencrypt traffic.

- Use the **Decrypt - Resign** rule action for decrypting *outgoing* traffic, as opposed to incoming traffic for which we recommend the **Decrypt - Known Key**or **Decrypt - Replace Cert** rule action.

  For more information about **Decrypt - Known Key**or **Decrypt - Replace Cert**, see Best practices for TLS/SSL incoming traffic decryption, on page 5.

- Always check the **Replace Key Only** check box when you set up a **Decrypt - Resign** rule action.

  When a user browses to a web site that uses a *self-signed* certificate, the user sees a security warning in the web browser and is aware that they are communicating with an unsecure site.

  When a user browses to a web site that uses a trusted certificate, the user does not see a security warning.

Guidelines and limitations

Anonymous cipher suite unsupported:

By nature, anonymous cipher suites are not used for authentication and do not use key exchanges. There are limited uses for anonymous cipher suites; for more information, see RFC 5246, appendix F.1.1.1. (Replaced for TLS 1.3 by RFC 8446 appendix C.5.)

You cannot use the **Decrypt - Resign** or **Decrypt - Known Key** action in the rule because anonymous cipher suites are not used for authentication.

- **Do not use Version or Cipher Suite rule conditions**:



Important Never use either **Cipher Suite** or **Version** rule conditions in a rule with a **Decrypt - Resign**, **Decrypt - Replace Cert**, or **Decrypt - Known Key** rule action. The use of these conditions in rules with other rule actions can interfere with the system's ClientHello processing, resulting in unpredictable performance.

- Decrypt - Resign rule action and a Certificate Signing Request: To use a **Decrypt - Resign** rule action, you should create a Certificate Signing Request (CSR) and have it signed by a trusted certificate authority. (You can use the FMC to create a CSR: **Objects** > **PKI** > **Internal Certs**.)

  To be used in a **Decrypt - Resign** rule, your certificate authority (CA) must have at least one of these extensions

  - **CA: TRUE**

    For more information, see the discussion of Basic Constraints in RFC3280, section 4.2.1.10.

  - **KeyUsage=CertSign**

    For more information see RFC 5280, section 4.2.1.3.

  To verify that your CSR or CA has at least one of the preceding extensions, use the **openssl** command as discussed in the openssl documentation.

  This is necessary because, for **Decrypt - Resign** inspection to work, the CA used in the decryption policy generates certificates on-the-fly and signs them so as to act as man-in-the middle and proxy all TLS/SSL connections.

- **Certificate pinning**: If the customer's browser uses certificate pinning to verify a server certificate, you cannot decrypt this traffic by re-signing the server certificate. To allow this traffic, configure a decryption rule with the **Do Not Decrypt** action to match the server certificate common name or distinguished name.

- **Non-matching cipher suite**: This error is displayed if you attempt to save a decryption rule with a cipher suite that does not match the certificate.

```
Traffic cannot match this rule; none of your selected cipher suites contain a
signature algorithm that matches the resigning CA's signature algorithm
```

- **Untrusted Certificate Authority**: If the client does not trust the Certificate Authority (CA) used to re-sign the server certificate, it warns the user that the certificate should not be trusted. To prevent this, import the CA certificate into the client trusted CA store. Alternatively, if your organization has a private PKI, you can issue an intermediate CA certificate signed by the root CA which is automatically trusted by all clients in the organization, then upload that CA certificate to the device.
- **HTTP proxy limitation**: The system cannot decrypt traffic if an HTTP proxy is positioned between a client and your managed device, and the client and server establish a tunneled TLS/SSL connection using the CONNECT HTTP method. The **Handshake Errors** undecryptable action determines how the system handles this traffic.
- **Upload signed CA**: If you create an internal CA object and choose to generate a certificate signing request (CSR), you cannot use this CA for a **Decrypt - Resign** action until you upload the signed certificate to the object.
- **Mismatched signature algorithm**: If you configure a rule with the **Decrypt - Resign** action, and mismatch signature algorithm type for one or more external certificate objects or cipher suites, the policy editor displays an **Information** () next to the rule. If you mismatch signature algorithm type for all external certificate objects, or all cipher suites, the policy displays a warning icon **Warning** () next to the rule, and you cannot deploy the access control policy associated with the decryption policy.

TLS/SSL block guidelines

If your decryption rule is associated with an access control policy that has a rule with an action of either **Interactive Block** or **Interactive Block with reset**, the system displays a customizable response page.

If you enabled logging in your rule, two connection events are displayed (in **Events & Logs** > **Analysis** > **Unified Events**): One event for the interactive block and another event to indicate whether or not the user chose to continue to the site or not.

Related Topics

HTTP response pages

TLS/SSL certificate pinning guidelines

Some applications use a technique referred to as *TLS/SSL pinning* or *certificate pinning*, which embeds the fingerprint of the original server certificate in the application itself. As a result, if you configured a decryption rule with a **Decrypt - Resign** action, when the application receives a resigned certificate from a managed device, validation fails and the connection is aborted.

Because TLS/SSL pinning is used to avoid man-in-the-middle attacks, there is no way to prevent or work around it. We recommend adding a **Do Not Decrypt** rule before the **Decrypt - Resign** rule so pinning traffic is excluded from being decrypted.

- Create a **Do Not Decrypt** for those applications rule ordered before **Decrypt - Resign** rules.
- Instruct users to access the applications using a web browser.

For more information about rule ordering, see Best practice for Decryption policy rule order, on page 10.

To determine whether applications are using TLS/SSL pinning, see Troubleshoot TLS/SSL pinning, on page 43.

TLS/SSL heartbeat guidelines

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.

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.

For more information, refer to TLS heartbeat, on page 41.

TLS/SSL anonymous cipher suite limitation

Anonymous cipher suites impact decryption rules.

By nature, anonymous cipher suites are not used for authentication and do not use key exchanges. There are limited uses for anonymous cipher suites; for more information, see RFC 5246, appendix F.1.1.1. (Replaced for TLS 1.3 by RFC 8446 appendix C.5.)

You cannot use the **Decrypt - Resign** or **Decrypt - Known Key** action in the rule because anonymous cipher suites are not used for authentication.

You can add an anonymous cipher suite to the **Cipher Suite** condition in a decryption rule, but the system automatically strips anonymous cipher suites during ClientHello processing. For the system to use the rule, you must also configure your decryption rules in an order that prevents ClientHello processing. For more information, refer to Best practice for Decryption policy rule order, on page 10

TLS/SSL normalizer guidelines

If you enable the **Normalize Excess Payload** option in the inline normalization preprocessor, when the preprocessor normalizes decrypted traffic, it might drop a packet and replace it with a trimmed packet. This does not end the TLS/SSL session. If the traffic is allowed, the trimmed packet is encrypted as part of the TLS/SSL session.

Other Rule-based decryption rule guidelines

Consider users and groups configuration when creating decryption rules. If you add a group or user to a rule, then change your realm settings to exclude that group or user, the rule has no effect. (The same applies to disabling the realm.) For more information about realms, see Create an LDAP realm or an Active Directory realm and realm directory.

Consider all relevant categories when configuring decryption rules. If your decryption policy has a **Decrypt - Resign** action but web sites are not being decrypted, check **Category** page on rules associated with that policy.

In some cases, a web site redirects to another site for authentication or other purposes and the redirected site might have a different URL categorization than the site you're trying to decrypt. For example, `gmail.com` (**Web based email** category) redirects to `accounts.gmail.com` (**Internet Portals** category) for authentication. Be sure to include all relevant categories in the SSL rule.

Note In order to fully process traffic based on URL category, you must also configure URL filtering. See the URL Filtering Rules chapter.

Consider configuring cloud query settings for URLs not in the local database. If you create a **Decrypt - Resign** rule and users browse to a web site whose category and reputation are not in the local database, data might not be decrypted. Some web sites are not categorized in the local database and, if not, data from those web sites is not decrypted by default.

You can control this behavior with the **Query Cisco cloud for unknown URLs** setting in **Integrations** > **Cloud Services**

Best practice for Decryption policy rule order

General rule ordering

Order your rules with specific conditions (such as IP addresses and networks) *before* rules with general conditions (such as applications).

Allow traffic from certificate pinned sites

Some applications use a technique referred to as *TLS/SSL pinning* or *certificate pinning*, which embeds the fingerprint of the original server certificate in the application itself. As a result, if you configured a decryption rule with a **Decrypt - Resign** action, when the application receives a resigned certificate from a managed device, validation fails and the connection is aborted.

- To confirm that TLS/SSL pinning is occurring, attempt to log in to a mobile application like Facebook. If a network connection error is displayed, log in using a web browser. (For example, you *cannot* log in to a Facebook mobile application but *can* log in to Facebook using Safari or Chrome.) You can use Firepower Management Center connection events as further proof of TLS/SSL pinning
- Configure an SSL rule with the **Do Not Decrypt** action to match the server certificate common name or distinguished name. In the SSL policy, order this rule before all **Decrypt - Resign** rules that also match the traffic.

You can retrieve the pinned certificate from the client's browser after a successful connection to the website. You can also view the certificate from the logged connection event, regardless of whether the connection succeeded or failed.

Note TLS/SSL pinning is not limited to mobile applications.

Prioritize ClientHello modifications

Place rules that match on conditions that are available in the ClientHello message before rules that match on ServerHello or server Certificate conditions.

When a managed device processes an SSL handshake, it can modify the ClientHello message to increase the likelihood of decryption. For example, it may remove compression methods because the system cannot decrypt compressed sessions.

The system modifies ClientHello messages only if it can conclusively match them to an SSL rule with either a **Decrypt - Resign** or **Decrypt - Known Key** action. The first time the system detects an encrypted session to a new server, server Certificate data is not available for ClientHello processing, which can result in an undecrypted first session. For subsequent connections from the same client, the system can match the ClientHello message conclusively to rules with server Certificate conditions and process the message to maximize decryption potential.

If you place rules that match on ServerHello or server Certificate conditions (certificate, distinguished names, certificate status, cipher suites, version) before rules that match on ClientHello conditions (zones, networks, VLAN tags, ports, users, applications, URL categories), you can preempt ClientHello modification and increase the number of undecrypted sessions.

Decryption policy bypass situations

The Decryption policy is bypassed for any connections that match access control rules with actions of **Trust**, **Block**, or **Block with reset** if those rules:

- Use security zone, network, geolocation, and port only as the traffic matching criteria.
- Precede other rules that require inspection, such as rules that match connections based on application or URL, or allow rules that apply intrusion or file inspection.

Rule-based decryption rule traffic handling

Rule-based decryption rule traffic handling is a security mechanism that matches traffic to rule-based decryption rules in a specified order to determine appropriate actions.

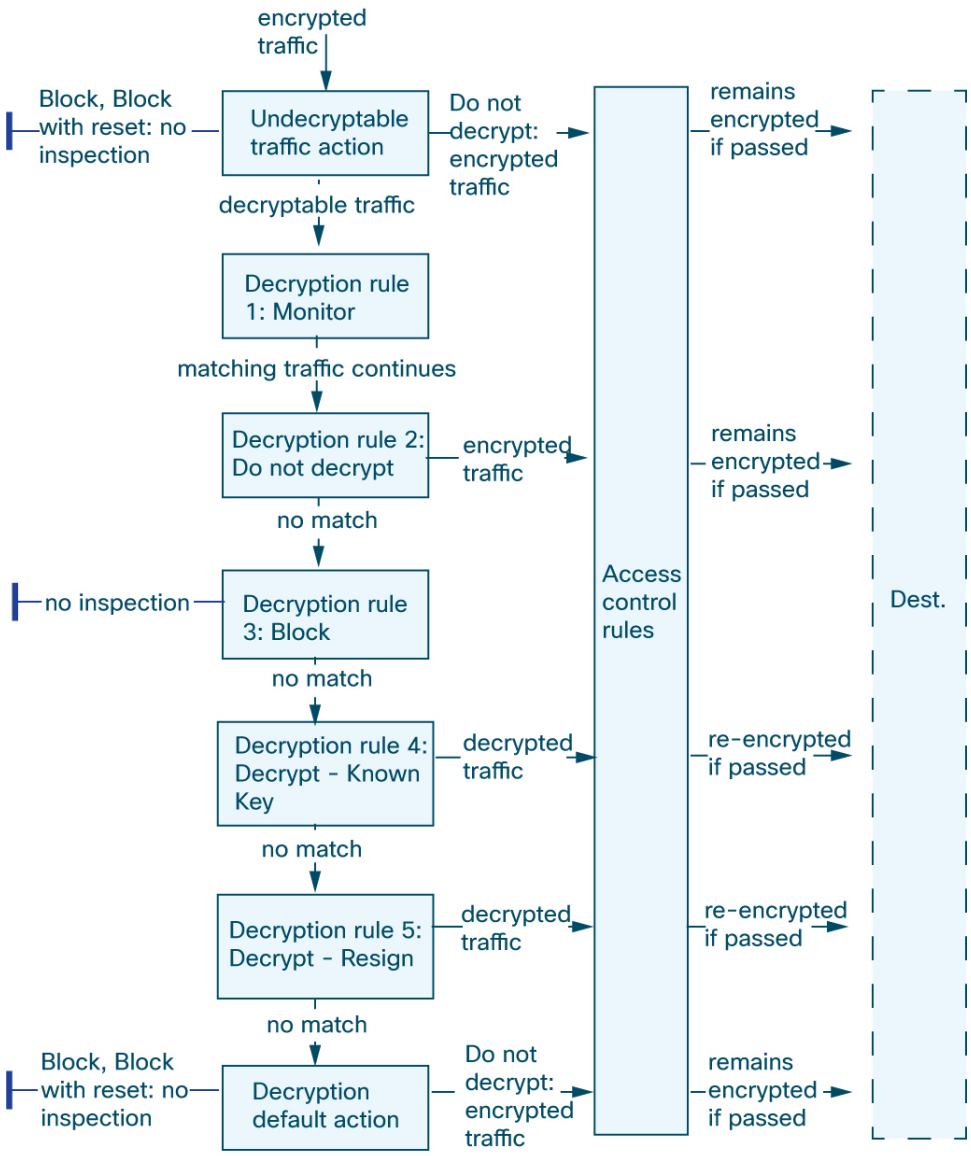
Traffic matching and processing behavior

The system matches traffic to rule-based decryption rules in the order you specify. In most cases, the system handles encrypted traffic according to the *first* decryption rule where *all* the rule's conditions match the traffic. 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 inspect encrypted and undecryptable traffic with access control. However, some access control rule conditions require unencrypted traffic, so encrypted traffic may match fewer rules. Also, by default, the system disables intrusion and file inspection of encrypted payloads.

Traffic handling process

This scenario summarizes the ways that Rule-based 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.
- 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.
- 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.

- Rule 3: Block evaluates encrypted traffic fourth. Matching traffic is blocked without further inspection. Traffic that does not match continues to the next rule.
- 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.
- 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 continues to the next rule.
- Default Action handles all traffic that does not match any decryption rule. The default action either blocks encrypted traffic without further inspection or does not decrypt it, passing it for access control inspection.

Encrypted traffic inspection configuration

You must create reusable public key infrastructure (PKI) objects to control encrypted traffic based on encrypted session characteristics and to decrypt encrypted traffic. Add this information dynamically when you upload trusted certificate authority (CA) certificates to a decryption policy and create a decryption rule. An associated object is created during this process. However, configuring these objects in advance reduces the chance of improper object creation.

Decrypting encrypted traffic with certificates and paired keys

The system can decrypt incoming encrypted traffic if you configure an internal certificate object by uploading the server certificate and private key used to encrypt the session. If you reference that object in a decryption policy rule with an action of **Decrypt - Known Key** and traffic matches that rule, the system uses the uploaded private key to decrypt the session.

The system can also decrypt outgoing traffic if you configure an internal CA object by uploading a CA certificate and private key. If you reference that object in a decryption rule with an action of **Decrypt - Resign** and traffic matches that rule, the system re-signs the server certificate passed to the client browser, then acts as a man-in-the-middle to decrypt the session. You can optionally replace the self-signed certificate key only and not the entire certificate, in which case users see a self-signed certificate key notice in the browser.

Controlling traffic based on encrypted session characteristics

The system can control encrypted traffic based on the cipher suite or server certificate used to negotiate the session. You can configure one of several different reusable objects and reference the object in a decryption rule condition to match traffic. This table describes the different types of reusable objects you can configure:

If you configure...	You can control encrypted traffic based on whether...
A cipher suite list containing one or more cipher suites	The cipher suite used to negotiate the encrypted session matches a cipher suite in the cipher suite list

If you configure...	You can control encrypted traffic based on whether...
A trusted CA object by uploading a CA certificate your organization trusts	The trusted CA trusts the server certificate used to encrypt the session, whether: • The CA issued the certificate directly • The CA issued a certificate to an intermediate CA that issued the server certificate
An external certificate object by uploading a server certificate	The server certificate used to encrypt the session matches the uploaded server certificate
A distinguished name object containing a certificate subject or issuer distinguished name	The subject or issuer common name, country, organization, or organizational unit on the certificate used to encrypt the session matches the configured distinguished name

Related Topics

Cipher suite lists
Distinguished names
PKI objects

Rule-based decryption rule order evaluation

Rule-based decryption rule order evaluation determines the sequence in which the system evaluates decryption rules in a policy.

Rule evaluation process

This information applies only to rule-based decryption policies and rules.

When you create a rule-based decryption rule in a rule-based decryption policy , you specify its position using the **Insert** list in the rule editor. Decryption rules in 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

Rule-based decryption rule conditions

A rule-based decryption rule condition identifies the type of encrypted traffic the rule handles. Conditions can be simple or complex, and you can specify more than one condition type per rule. Only if traffic meets all the conditions in a rule does the rule apply to the traffic.

Condition evaluation and rule actions

If you do not configure a particular condition for a rule, the system does not match traffic based on that criterion. For example, a rule with a certificate condition but no version condition evaluates traffic based on the server certificate used to negotiate the session, regardless of the session SSL or TLS version.

Every rule-based decryption policy has an associated action that determines handling and logging for matching encrypted traffic:

- Handling: Most importantly, the rule action governs whether the system will monitor, trust, block, or decrypt encrypted traffic that matches the rule's conditions.
- Logging: The rule action determines when and how you can log details about matching encrypted traffic.

Your TLS/SSL inspection configuration handles, inspects, and logs decrypted traffic:

- The decryption policy's undecryptable actions handle traffic that the system cannot decrypt.
- The policy's default action handles traffic that does not meet the condition of any non-Monitor decryption rule.

You can log a connection event when the system blocks or trusts an encrypted session. 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. Connection logs for encrypted sessions contain details, such as the certificate used to encrypt that session. You can log only end-of-connection events.

- For blocked connections (Block, Block with reset), the system immediately ends the sessions and generates an event.
- For Do Not Decrypt connections, the system generates an event when the session ends.

Minimize the number of matching criteria whenever possible, especially those for security zones, network objects, and port objects. When you specify multiple criteria, the system must match against *every* combination of the contents of the criteria you specify.



Caution Adding the first or removing the last active authentication rule when TLS/SSL decryption is disabled (that is, when the access control policy does not include a decryption policy) restarts the Snort process when you deploy configuration changes, temporarily interrupting traffic inspection. Whether traffic drops during this interruption or passes without further inspection depends on how the assigned device handles traffic. Refer to Snort restart traffic behavior.

Note that an active authentication rule has either an **Active Authentication** rule action, or a **Passive Authentication** rule action with **Use active authentication if passive or VPN identity cannot be established** selected.

Related Topics

Security zone rule conditions, on page 16
Network rule conditions
VLAN tag rule conditions
User rule conditions
Application rule conditions
Best practice: use application-based rules over port-based rules
Category rule conditions, on page 20
Server certificate-based rule conditions, on page 21

Security zone rule conditions

A security zone rule condition is a network traffic control mechanism that

- segments your network to help you manage, classify, and decrypt traffic flow by grouping interfaces across multiple devices,
- controls or decrypts traffic by its source and destination security zones, and
- requires all interfaces in a zone to be of the same type (all inline, passive, switched, or routed).

Zone condition traffic matching

If you add both source and destination zones to a zone condition, matching traffic must originate from an interface in one of the source zones and leave through an interface in one of the destination zones.

All zones used in a zone condition must be of the same type. Because devices deployed passively do not transmit traffic, you cannot use a zone with passive interfaces as a destination zone.

Minimize the number of matching criteria whenever possible, especially those for security zones, network objects, and port objects. When you specify multiple criteria, the system must match against *every* combination of the contents of the criteria you specify.

Tip Constraining rules by zone is one of the best ways to improve system performance. If a rule does not apply to traffic through any of device's interfaces, that rule does not affect that device's performance.

Network rule conditions

A network condition is a traffic matching criterion that

- matches traffic for rate limiting based on source and destination IP addresses using inner packet headers,
- enables administrators to specify exact IP addresses or blocks for granular policy enforcement, and
- can be built using predefined network objects or by manually entering IP addresses or address blocks.

Network rule condition restrictions

Minimize the number of matching criteria whenever possible, especially those for security zones, network objects, and port objects. When you specify multiple criteria, the system must match against *every* combination of the contents of the criteria you specify.

Note You *cannot* use FQDN network objects in identity rules.

VLAN tag rule conditions

VLAN tags rule conditions are filtering mechanisms that

- control VLAN-tagged traffic, including Q-in-Q (stacked VLAN) traffic
- use the innermost VLAN tag to filter VLAN traffic, with the exception of prefilter policy which uses the outermost VLAN tag, and
- apply only to inline sets and do not match traffic on firewall interfaces.

VLAN tag rule conditions configuration details

Note VLAN tags in access rules only apply to inline sets. Access rules with VLAN tags do not match traffic on firewall interfaces.

The system uses the innermost VLAN tag to filter VLAN traffic, with the exception of the prefilter policy, which uses the outermost VLAN tag in its rules.

Note this Q-in-Q support:

- Firewall Threat Defense on Firepower 4100/9300—Does not support Q-in-Q (supports only one VLAN tag).

- Firewall Threat Defense on all other models:
  - Inline sets and passive interfaces—Supports Q-in-Q, up to 2 VLAN tags.
  - Firewall interfaces—Does not support Q-in-Q (supports only one VLAN tag).

You can use predefined objects to build VLAN conditions, or manually enter any VLAN tag from 1 to 4094. Use a hyphen to specify a range of VLAN tags.

In a cluster, if you encounter problems with VLAN matching, edit the access control policy advanced options, Transport/Network Preprocessor Settings, and select the **Ignore the VLAN header when tracking connections** option.

User rule conditions

A user rule condition is a traffic matching mechanism that

- matches traffic based on the user initiating the connection or the group to which they belong
- can be configured for users in Microsoft Active Directory realms only, and
- enables policy enforcement based on user identity and group membership.

Special identities users

In addition to configuring users and groups for configured realms, you can set policies for the special identities users:

- Failed Authentication: User who failed authentication with the captive portal.
- Guest: Users configured as guest users in the captive portal.
- No Authentication Required: Users that match an identity **No Authentication Required** rule action.
- Unknown: Users that cannot be identified such as users who are not downloaded by a configured realm.

For access control rules only, you must first associate an identity policy with the access control policy as discussed in Associate prefilter, decryption, and identity policies with an access control.

Configure a QoS rule to rate-limit traffic initiated by users in the Finance group.

Application rule conditions

Application rule conditions are application awareness and control mechanisms that

- identify and classify applications through IP traffic analysis
- enable control with organized filters
- require one enabled detector for each condition.

Application filters and control benefits

System-provided *application filters* help perform control by organizing applications according to basic characteristics, such as type, risk, business relevance, category, and tags. You can also create reuseable, user-defined filters using combinations of system-provided filters, or custom combinations of applications.

At least one detector must be enabled for each application rule condition in the policy. If no detector is enabled, the system automatically enables all available system-provided detectors, or the most recently modified user-defined detector. For more information about application detectors, see Application detectors.

You can utilize both application filters and individually specified applications to ensure complete coverage. Understand the application's requirements to effectively order access control rules.

Application filters help you to quickly configure application control effectively. For example, use a system-provided filter to create a QoS rule that identifies high-risk, low-business-relevance applications and rate-limits matching traffic.

Using application filters simplifies policy creation and administration. It assures you that the system controls application traffic as expected. Because Cisco frequently updates and adds application detectors via system and vulnerability database (VDB) updates, you can be sure that the system uses up-to-date detectors to monitor application traffic. You can also create your own detectors and assign characteristics to the applications they detect, automatically adding them to existing filters.

The system characterizes detected applications using several criteria which form effective filters.

Table 1: Application characteristics

Characteristic	Description	Example
Type	Application protocols represent communications between hosts. Clients represent software running on a host. Web applications represent the content or requested URL for HTTP traffic.	HTTP and SSH are application protocols. Web browsers and email applications are clients. MPEG video and Facebook are web applications.
Risk	The likelihood that the application is being used for purposes that might be against your organization's security policy.	Peer-to-peer applications tend to have a very high risk.
Business Relevance	The likelihood that the application is being used within the context of your organization's business operations, as opposed to recreationally.	Gaming applications tend to have a very low business relevance.
Category	A general classification for the application that describes its most essential function. Each application belongs to at least one category.	Facebook is in the social networking category.
Tag	Additional information about the application. Applications can have any number of tags, including none.	Video streaming web applications often are tagged `high bandwidth` and `displays ads`.

Best practice: use application-based rules over port-based rules

Use application filtering criteria instead of port filtering to target traffic. Applications can be configured to use unique ports to bypass access control blocks.

Port conditions allow you to control traffic by its source and destination ports.

Minimize the number of matching criteria whenever possible, especially those for security zones, network objects, and port objects. When you specify multiple criteria, the system must match against *every* combination of the contents of the criteria you specify.

The traditional method targets applications by using port specifications, but applications can manipulate unique ports to bypass these controls

Application filtering is recommended for applications to avoid blocking legitimate applications, like Firewall Threat Defense, that dynamically open separate channels for control and data flow. Port-based access control rules could block such applications, impacting desirable operations.

Using source and destination port constraints

If you add both source and destination port constraints, you can only add ports that share a single transport protocol (TCP or UDP).

- For example, if you add DNS over TCP as a source port, you can add Yahoo Messenger Voice Chat (TCP) as a destination port but not Yahoo Messenger Voice Chat (UDP).
- If you add only source ports or only destination ports, you can add ports that use different transport protocols. For example, you can add both DNS over TCP and DNS over UDP as source port conditions in a single access control rule.

Category rule conditions

Category rule conditions are policy elements that

- enable control of web access based on Cisco Talos intelligence
- support policy creation through URL filtering without relying on company vertical, intent, or declared security, and
- can be included in decryption policies to manage encrypted traffic.

Additional information

You can optionally choose to include categories in your decryption policies. These categories, also referred to as *URL filtering*, are updated by the Cisco Talos intelligence group. Updates are based on machine learning and human analysis according to content that is retrievable from the website destination and sometimes from its hosting and registration information. Categorization is *not* based on the declared company vertical, intent, or security.

For more information, see URL filtering.

If you are using category rule conditions in decryption policies in a rule with the **Do Not Decrypt** rule action, see Decryption rule Do Not Decrypt action, on page 33.

Server certificate-based rule conditions

Server certificate-based rule conditions are decryption rules configurations that

- handle and decrypt encrypted traffic based on server certificate characteristics
- inspect traffic using certificate attributes, status, cipher suites, and session versions, and
- enhance secure traffic management and visibility through certificate validation criteria.

Server certificate attribute conditions

You can configure decryption rules based on these server certificate attributes:

- **Distinguished name conditions**: Allow you to handle and inspect encrypted traffic based on the CA that issued a server certificate, or the certificate holder. Based on the issuer distinguished name, you can handle traffic based on the CA that issued a site's server certificate.
- **Certificate conditions**: In decryption rules allow you to handle and inspect encrypted traffic based on the server certificate used to encrypt that traffic. You can configure a condition with one or more certificates; traffic matches the rule if the certificate matches any of the condition's certificates.
- **Certificate status conditions**: In decryption rules allow you to handle and inspect encrypted traffic based on the status of the server certificate used to encrypt the traffic, including whether a certificate is valid, revoked, expired, not yet valid, self-signed, signed by a trusted CA, whether the Certificate Revocation List (CRL) is valid; whether the Server Name Indication (SNI) in the certificate matches the server in the request.
- **Cipher suite conditions**: In decryption rules allow you to handle and inspect encrypted traffic based on the cipher suite used to negotiate the encrypted session.
- **Session conditions**: In decryption rules allow you to inspect encrypted traffic based on the SSL or TLS version used to encrypt the traffic.

To detect multiple cipher suites in a rule, the certificate issuer, or the certificate holder, you can create reusable cipher suite list and distinguished name objects and add them to your rule. To detect the server certificate and certain certificate statuses, you must create external certificate and external CA objects for the rule.

Related Topics

Certificate rule conditions, on page 21
Certificate Status Decryption rule conditions, on page 28
Trust external certificate authorities, on page 27
Match traffic on certificate status
Cipher suite rule conditions, on page 31
Encryption protocol version rule conditions, on page 32

Certificate rule conditions

When you build a certificate-based decryption rule condition, you can upload a server certificate and save it as an external certificate *object*. The certificate object is reusable and associates a name with a server certificate. Alternately, you can configure certificate conditions with existing external certificate objects and object groups.

You can search the **Available Certificates** field in the rule condition based for external certificate objects and object groups based on these certificate distinguished name characteristics:

- Subject or issuer common name (CN), or if the URL is contained in the certificate's Subject Alternative Name (SAN)

  The URL the user enters in the browser matches the Common Name (CN)

- Subject or issuer organization (O)
- Subject or issuer organizational unit (OU)

You can choose to match against multiple certificates in a single certificate rule condition. If the certificate used to encrypt the traffic matches any of the uploaded certificates, the encrypted traffic matches the rule.

You can add a maximum of 50 external certificate objects and external certificate object groups to the **Selected Certificates** in a single certificate condition.

Note these items:

- You cannot configure a certificate condition if you also select the **Decrypt - Known Key** action. That action requires you to select a server certificate to decrypt traffic. Therefore, the certificate already matches the traffic.
- When the system first detects an encrypted session to a new server, it does not have certificate data for ClientHello processing. As a result, the first session may not be decrypted. After the initial session, the managed device caches data from the server Certificate message. For subsequent connections from the same client, the system can match the ClientHello message conclusively to rules with certificate conditions and process the message to maximize decryption potential.

Conditions for distinguished name rules in decryption policies

Learn to use distinguished name conditions in a decryption rule. If you're not sure, you can find a certificate's Subject Alternative Name (SAN) and Common Name using a web browser. You can then add those values to a decryption rule as distinguished name conditions.

For detailed information about SANs, see RFC 5280, section 4.2.1.6.

Overview

Distinguished name (DN) conditions allow you to specify certificate field values in decryption rules. This helps control which encrypted traffic is decrypted or excluded from decryption based on the certificate's Common Name (CN) and Subject Alternative Name (SAN).

Identifying Certificate Values

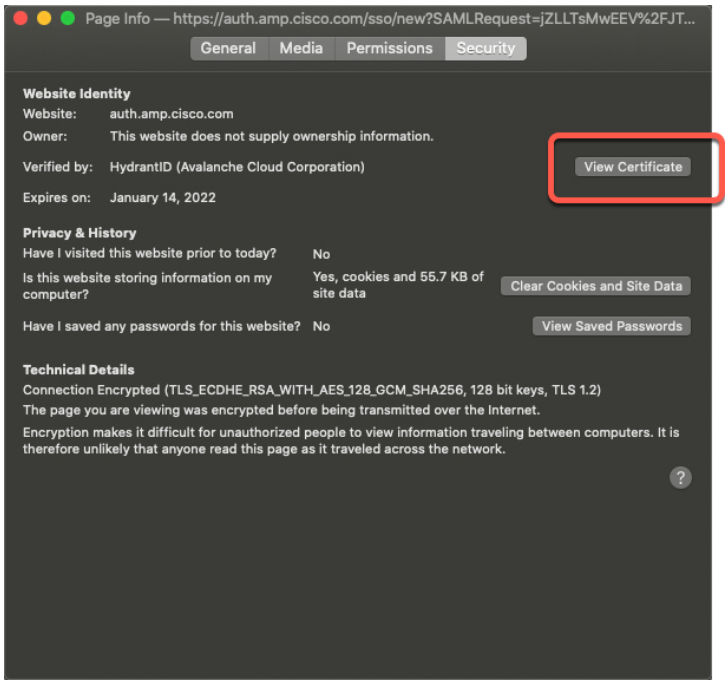
Use these steps to find a certificate's Common Name. These steps also work to find the Common Name and SANs for a self-signed certificate.

These instructions use Firefox as an example, but other browsers work similarly. The following procedure uses `amp.cisco.com` as an example.

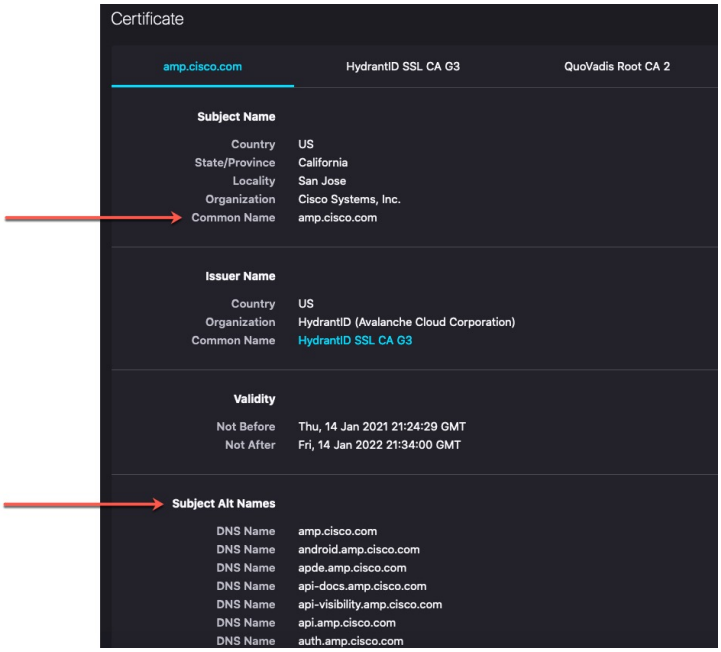
1. Browse to `amp.cisco.com` in Firefox.
2. In the browser's location bar, to the left of the URL, click .
3. Click **Connection secure** > **More Information**.

(For a non-secure or self-signed certificate, click **Connection not secure** > **More Information**.)

4. On the Page Info dialog box, click **View Certificate**.



5. The next page shows certificate details.



Note the following:

- `CN=auth.amp.cisco.com`, if used as a DN rule condition, would match ONLY that host name (that is, SNI). The SNI `amp.cisco.com` would NOT match.

- Wildcards can be used in DNs to match multiple domains.

  For example, to match `auth.amp.cisco.com`, use `CN=*.amp.cisco.com`. To match `auth.us.amp.cisco.com`, use `CN=*.*.amp.cisco.com`.

  A DN like `CN=*.example.com` matches `www.example.com` but NOT `example.com`. To match both SNIs, use two DNs in the rule condition.

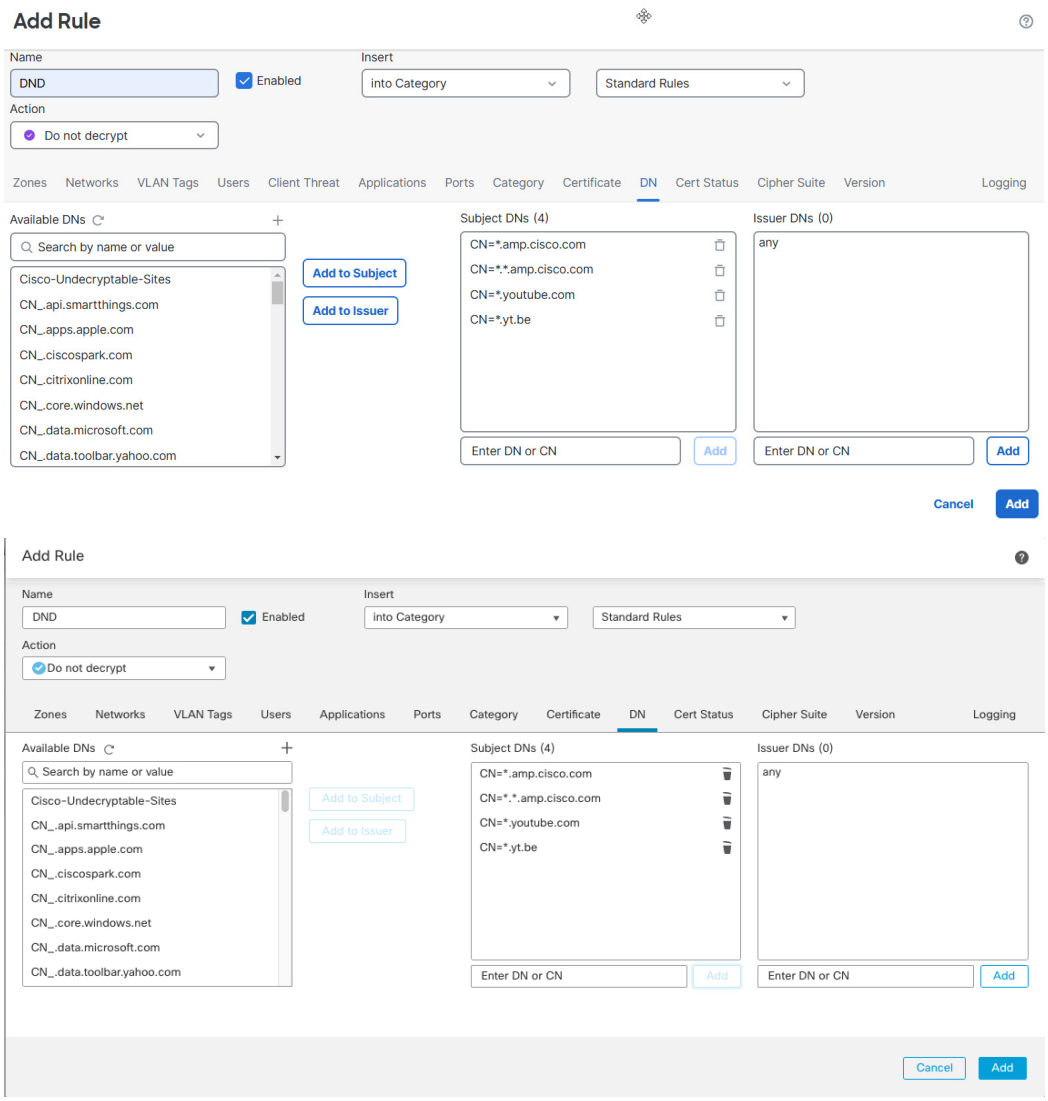
- Avoid overly broad wildcards like `CN=*.google.com` matches a very large number of SANs. Instead of `CN=*.google.com`, use a DN object like `CN=*.youtube.com` as the DN object so it matches names like `www.youtube.com`.

  You can also use variations of the SNI that match SANs like `CN=*.youtube.com`, `CN=youtu.be`, `CN=*.yt.be`, and so on.

- A self-signed certificate should work the same way. You can confirm it's a self-signed certificate by the fact the issuer DN is the same as the subject DN.

Example: DN rule matching

Suppose you want to prevent decryption for traffic going to `amp.cisco.com` or YouTube. Configure DN rule conditions such as:



These DN conditions would match URLs like:

- www.amp.cisco.com
- auth.amp.cisco.com
- auth.us.amp.cisco.com
- www.youtube.com
- kids.youtube.com
- www.yt.be

These DN conditions would not match any of the URLs listed, so the traffic would not match the Do Not Decrypt rule. However, it might match another decryption rules in the same decryption policy .

- amp.cisco.com
- youtube.com

- yt.be

To match any of the preceding host names, add more CNs to the rule (for example, adding `CN=yt.be` would match that URL.)

System behavior: SNI and SANs usage

The host name in the client request is called the Server Name Indication (SNI). During the TLS handshake, the client specifies which hostname they want to connect to (for example, `auth.amp.cisco.com`) using the SNI extension. The server then selects the corresponding private key and certificate chain that are required to establish the connection while hosting all certificates on a single IP address.

If the SNI matches the CN or a SAN in the certificate, the SNI is used for DN rule matching.

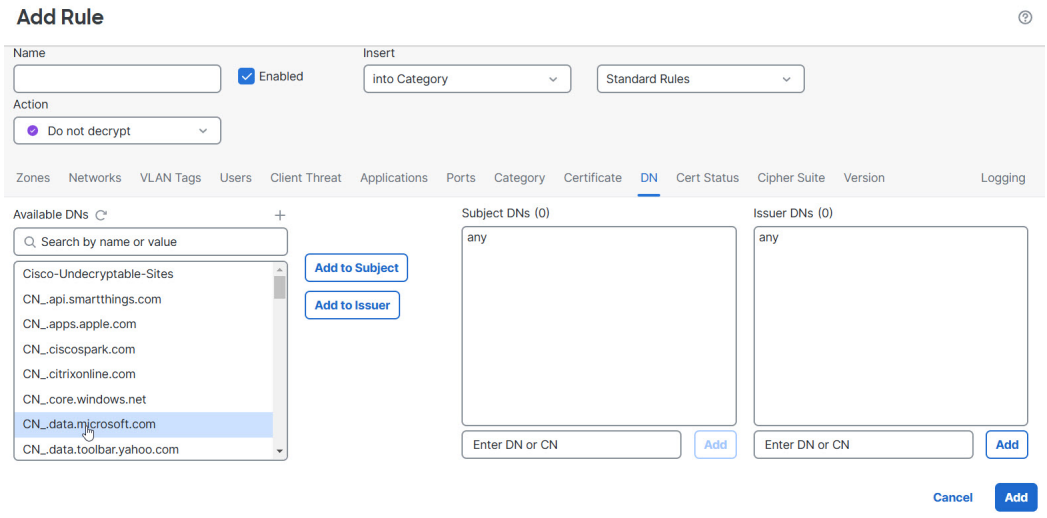
If there is no SNI, or the SNI does not match the certificate, the CN is used for DN rule matching.

Add a DN rule condition

After you identify the CN you want to match, edit the decryption rule by using an existing DN, creating a new DN object, or adding the DN directly.

- Use an existing DN.

  Click the name of a DN and then click either **Add to Subject** or **Add to Issuer**. (**Add to Subject** is much more common.) To view the value of a DN object, hover the mouse pointer over it.)

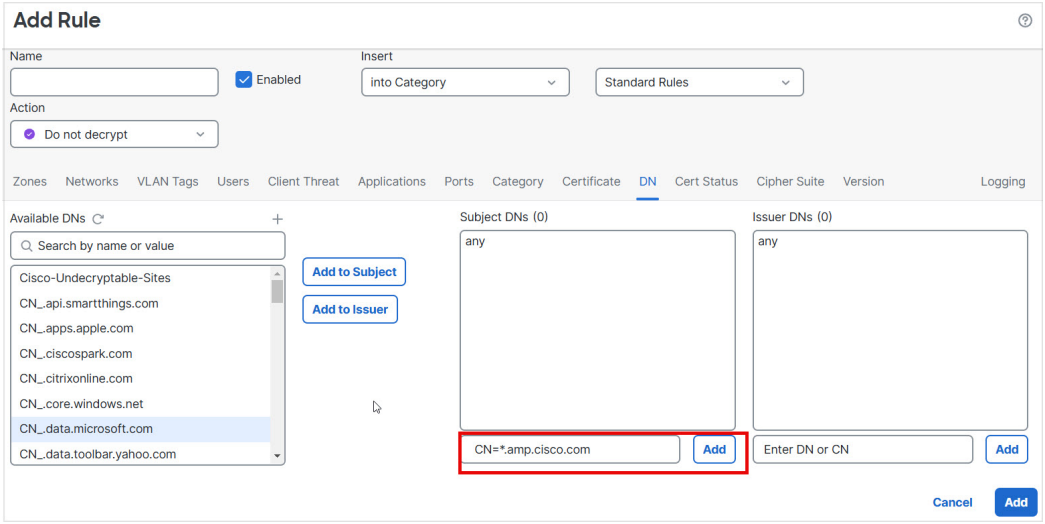


- Create a new DN object.

  Click **Add** (+) to the right of Available DNs. The DN object must consist of a name and a value.

- Add the DN directly.

  Enter the DN in the field at the bottom of the **Subject DNs** field or the **Issuer DNs** field. (**Subject DNs** is more common.) After you enter the DN, click **Add**.



Related Topics

Distinguished names

Trust external certificate authorities

Trust CAs by adding root and intermediate CA certificates to your decryption policy, then use these trusted CAs to verify server certificates used to encrypt traffic.

If a trusted CA certificate contains an uploaded certificate revocation list (CRL), you can also verify whether a trusted CA revoked the encryption certificate.

Tip Upload all certificates in a root CA's chain of trust to the list of trusted CA certificates, including the root CA certificate and all intermediate CA certificates. Otherwise, it is more difficult to detect trusted certificates issued by intermediate CAs. Also, if you configure certificate status conditions to trust traffic based on the root issuer CA, all traffic within a trusted CA's chain of trust can be allowed without decryption, rather than unnecessarily decrypting it.

For more information, see Trusted CA objects.

Note When you create a decryption policy, the policy's **Trusted CA Certificate** tab page is populated with several trusted CA certificates, including the **Cisco-Trusted-Authorities** group, which is added to the **Select Trusted CAs** list.

Procedure

Step 1 Click **Policies** > **Security policies** > **Decryption**.

Step 2 Click **Edit** (✎) next to the decryption policy to edit.

Step 3 Click **Add Rule** to add a new decryption rule or click **Edit** (✎) to edit an existing rule.

Step 4 Click the **Certificates** tab.

Step 5 Find the trusted CAs you want to add from the **Available Certificates**, as follows:

- To add a trusted CA object on the fly, which you can then add to the condition, click **Add** (+) above the **Available Certificates** list.
- To search for trusted CA objects and groups to add, click the **Search by name or value** prompt above the **Available Certificates** list, then enter either the name of the object, or a value in the object. The list updates as you type to display matching objects.

Step 6 To select an object, click it. To select all objects, right-click and then **Select All**.

Step 7 Click **Add to Rule**.

Tip You can also drag and drop selected objects.

Step 8 Add or continue editing the rule.

The trusted CA certificates are configured in your decryption policy and can be used to verify server certificates for secure traffic inspection.

What to do next

- Add a certificate status decryption rule condition to your SSL rule.
- Deploy configuration changes.

Certificate Status Decryption rule conditions

For each certificate status decryption rule you configure, you can match traffic against the presence or absence of a given status. You can select several statuses in one rule condition. If the certificate matches any of the selected statuses, the rule matches the traffic.

You can choose to match against the presence or absence of multiple certificate statuses in a single certificate status rule condition. The certificate needs to match only one of the criteria to match the rule.

You should consider, when setting this parameter, whether you're configuring a decrypt rule or a block rule. Typically, you should click **Yes** for a block rule and **No** for a decrypt rule. Examples:

- If you're configuring a **Decrypt - Resign** rule, the default behavior is to decrypt traffic with an expired certificate. To change that behavior, click **No** for **Expired** so traffic with an expired certificate is not decrypted and resigned.
- If you're configuring a **Block** rule, the default behavior is to allow traffic with an expired certificate. To change that behavior click **Yes** for **Expired** so traffic with an expired certificate is blocked.

This table describes how the system evaluates encrypted traffic based on the status of the encrypting server's certificate.

Table 2: Certificate status rule condition criteria

Status Check	Status Set to Yes	Status Set to No
Revoked	The policy trusts the CA that issued the server certificate, and the CA certificate uploaded to the policy contains a CRL that revokes the server certificate.	The policy trusts the CA that issued the certificate, and the CA certificate uploa policy does not contain a CRL that rev certificate.
Self-signed	The detected server certificate contains the same subject and issuer distinguished name.	The detected server certificate contains subject and issuer distinguished names
Valid	All of these are true: • The policy trusts the CA that issued the certificate. • The signature is valid. • The issuer is valid. • None of the policy's trusted CAs revoked the certificate. • The current date is between the certificate Valid From and Valid To date.	At least one of these is true: • The policy does not trust the CA t certificate. • The signature is invalid. • The issuer is invalid. • A trusted CA in the policy revoke certificate. • The current date is before the cert From date. • The current date is after the certifi date.
Invalid signature	The certificate's signature cannot be properly validated against the certificate's content.	The certificate's signature is properly val the certificate's content.
Invalid issuer	The issuer CA certificate is not stored in the policy's list of trusted CA certificates.	The issuer CA certificate is stored in th of trusted CA certificates.
Expired	The current date is after the certificate Valid To date.	The current date is before or on the cer To date.
Not yet valid	The current date is before the certificate Valid From date.	The current date is after or on the certi From date.

Status Check	Status Set to Yes	Status Set to No
Invalid certificate	The certificate is not valid. At least one of these is true: • Invalid or inconsistent certificate extension; that is, a certificate extension had an invalid value (for example, an incorrect encoding) or some value inconsistent with other extensions. • The certificate cannot be used for the specified purpose. • The Basic Constraints path length parameter has been exceeded. For more information, see RFC 5280, section 4.2.1.9. • The certificate's value for Not Before or Not After is invalid. These dates can be encoded as UTCTime or GeneralizedTime For more information, see RFC 5280 section 4.1.2.5. • The format of the name constraint is not recognized; for example, an email address format of a form not mentioned in RFC 5280, section 4.2.1.10. This could be caused by an improper extension or some new feature not currently supported. An unsupported name constraint type was encountered. OpenSSL currently supports only directory name, DNS name, email, and URI types. • The root certificate authority is not trusted for the specified purpose. • The root certificate authority rejects the specified purpose.	The certificate is valid. All of these are tru • Valid certificate extension. • The certificate can be used for the spe purpose. • Valid Basic Constraints path length. • Valid values for Not Before and Not A • Valid name constraint. • The root certificate is trusted for the s purpose. • The root certificate accepts the specifie

Status Check	Status Set to Yes	Status Set to No
Invalid CRL	The Certificate Revocation List (CRL) digital signature is not valid. At least one of these is true: • The value of the CRL's Next Update or Last Update field is invalid. • The CRL is not yet valid. • The CRL has expired. • An error occurred when attempting to verify the CRL path. This error occurs only if extended CRL checking is enabled. • CRL could not be found. • The only CRLs that could be found did not match the scope of the certificate.	The CRL is valid. All of these are true: • Next Update and Last Update fiel • The CRL's date is valid. • The path is valid. • The CRL was found. • The CRL matches the certificate's
Server mismatch	The server name does not match the server's Server Name Indication (SNI) name, which could indicate an attempt to spoof the server name.	The server name matches the SNI name to which the client is requesting access

Note that even though a certificate might match more than one status, the rule causes an action to be taken on the traffic only once.

Checking whether a CA issued or revoked a certificate requires uploading root and intermediate CA certificates and associated CRLs as objects. You then add these trusted CA objects to an decryption policy's list of trusted CA certificates.

Cipher suite rule conditions

The system provides predefined cipher suites you can add to a cipher suite rule condition for Block or Block WITH Reset rule actions. You can also add cipher suite list objects containing multiple cipher suites.

Important Cipher suite rule conditions should be used only to *block* traffic, they should not be used to *decrypt* traffic.

Note You cannot add new cipher suites. You can neither modify nor delete predefined cipher suites.

You can add a maximum of 50 cipher suites and cipher suite lists to the **Selected Cipher Suites** in a single cipher suite condition.

Note these considerations:

- If you add cipher suites not supported for your deployment, you cannot deploy your configuration. For example, passive deployments do not support decrypting traffic with the any of the ephemeral Diffie-Hellman (DHE) or ephemeral elliptic curve Diffie-Hellman (ECDHE) cipher suites. Creating a rule with these cipher suites prevents you from deploying your access control policy.

- You can add an anonymous cipher suite to the **Cipher Suite** condition in decryption policy. To use the rule, you must also configure your settings in an order that prevents ClientHello processing. For more information, refer to Best practice for Decryption policy rule order, on page 10.
- When you specify a cipher suite as a rule condition, the rule matches the negotiated cipher suite in the ServerHello message, not the entire list from the ClientHello message. During ClientHello processing, the managed device removes unsupported cipher suites from the ClientHello message. If all specified cipher suites are removed, the system retains the original list. If the system keeps unsupported cipher suites, it cannot decrypt the session during subsequent evaluation.

Encryption protocol version rule conditions

You can choose to match against traffic encrypted with SSL version 3.0, or TLS version 1.0, 1.1, or 1.2. By default, all protocol versions are selected when you create a rule; if you select multiple versions, encrypted traffic that matches any of the selected versions matches the rule. You must select at least one protocol version when saving the rule condition.

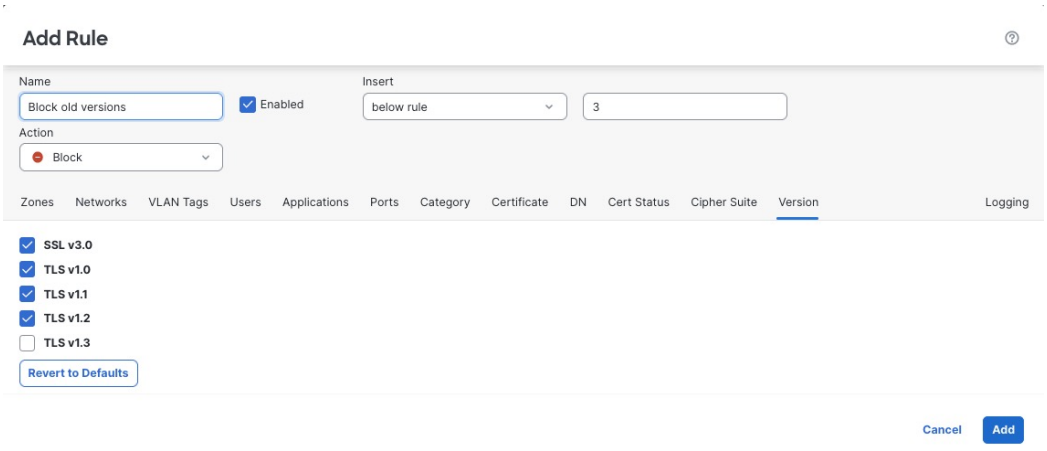
You can use SSL 3.0 in a Do Not Decrypt, Block, or Block with Reset rule action.

You *cannot* select SSL v2.0 in a version rule condition; the system does not support decrypting traffic encrypted with SSL version 2.0. You can configure an undecryptable action to allow or block this traffic without further inspection. For more information, see Set default handling for undecryptable traffic.

Important Protocol version rule conditions should be used *only* to *block* traffic, they should not be used to *decrypt* traffic.

For example, to block all SSL v3.0, TLS v1.0, TLS v1.1, and TLS v1.2 traffic, set the options as follows:

Figure 1: Block all SSL v3.0, TLS v1.0, TLS v1.1, and TLS 1.2 traffic



Rule-based decryption rule actions

Rule-based decryption rule actions are operational functions that define how the system processes SSL/TLS traffic based on configured decryption rules.

Available actions

The system provides various actions for rule-based decryption rule.

Decryption rule monitor action

The **Monitor** action is not designed to permit or deny traffic. Rather, its primary purpose is to force connection logging, regardless of how matching traffic is eventually handled. The ClientHello message is not modified if traffic matches a **Monitor** rule condition.

Monitor action traffic handling

Traffic that matches a **Monitor** rule is processed in this sequence:

1. Traffic matches the Monitor rule condition and connection logging is forced.
2. Traffic is then matched against additional rules, if present, to determine whether to trust, block, or decrypt it.
3. The first non-Monitor rule matched determines traffic flow and any further inspection.
4. If there are no additional matching rules, the system uses the default action.

The system automatically logs end-of connection events for monitored traffic to the Security Cloud Control database, regardless of the logging configuration of the rule or default action that later handles the connection.

Decryption rule Do Not Decrypt action

The Do Not Decrypt action passes encrypted traffic for evaluation by the access control policy's rules and default action. Because some access control rule conditions require unencrypted traffic, this traffic might match fewer rules. The system cannot perform deep inspection on encrypted traffic, such as intrusion or file inspection.

Typical use cases for do not decrypt rules

Typical reasons for a **Do Not Decrypt** rule action include:

- When decrypting TLS/SSL traffic is prohibited by law.
- Sites you know you can trust.
- Sites you can disrupt by inspecting traffic (such as Windows Update).
- To view the values of TLS/SSL fields using connection events. (You do not need to decrypt traffic to view connection event fields.)

For more information, see Default handling options for undecryptable traffic

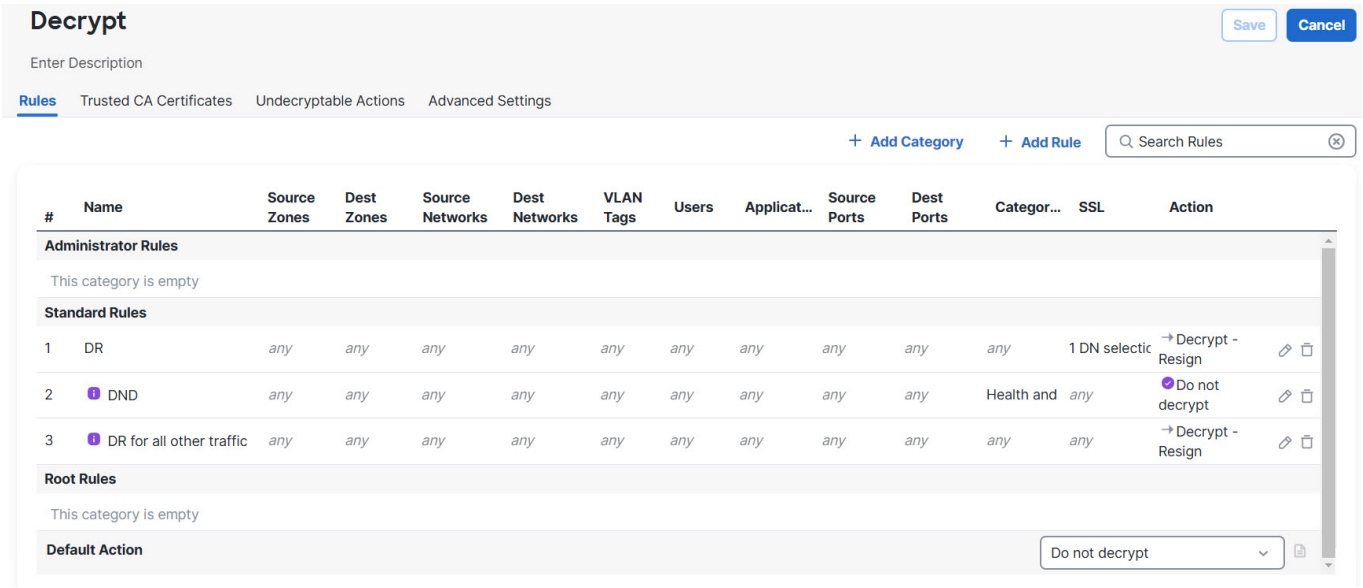
Limitations of categories in Do Not Decrypt rules

You can optionally choose to include categories in your decryption policies. These categories, also referred to as *URL filtering*, are updated by the Cisco Talos intelligence group. Updates are based on machine learning and human analysis according to content that is retrievable from the website destination and sometimes from its hosting and registration information. Categorization is *not* based on the declared company vertical, intent, or security. While we strive to continuously update and improve URL filtering categories, it is not an exact

science. Some websites are not categorized at all and it's possible some websites might be improperly categorized.

Avoid overusing categories in do not decrypt rules to avoid decrypting traffic without a reason; for example, the Health and Medicine category includes the WebMD website, which does not threaten patient privacy.

Following is a sample decryption policy that can prevent decryption for websites in the Health and Medicine categories but allows decryption for WebMD and everything else. General information about decryption rules can be found in Guidelines for using TLS/SSL decryption, on page 2.



Note Don't confuse URL filtering with application detection, which relies on reading some of the packet from a website to determine more specifically what it is (for example, Facebook Message or Salesforce). For more information, see Recommendations for application control.

Decryption rule blocking actions

Decryption rule blocking actions provide different termination methods to control connection handling.

Available blocking actions

The system provides these decryption rule actions for traffic you do not want to pass through the system:

- **Block**: Terminates the connection, resulting in an error in the client browser. The error message does not indicate the site was blocked due to policy. Instead, errors might indicate that there are no common encryption algorithms. It is not obvious from this message that you blocked the connection on purpose.
- **Block with reset**: Terminates and resets the connection, resulting in an error in the client browser. The error indicates the connection was reset but does not indicate why.

Tip You cannot use the **Block** or **Block with reset** action in a passive or inline (tap mode) deployment because the device does not directly inspect the traffic. If you create a rule with the **Block** or **Block with reset** action that contains passive or inline (tap mode) interfaces within a security zone condition, the policy editor displays a warning ( ) next to the rule.

Decryption rule decrypt actions

Decrypt - Known Key, and **Decrypt - Resign** are rule actions that decrypt encrypted traffic. The system inspects decrypted traffic with access control. Access control rules handle decrypted and unencrypted traffic identically — you can inspect it for discovery data as well as detect and block intrusions, prohibited files, and malware. The system reencrypts allowed traffic before passing it to its destination.

Certificate authority recommendations

We recommend you use a certificate from a trusted Certificate Authority (CA) to decrypt traffic. This prevents **Invalid Issuer** from being displayed in the SSL Certificate Status column in connection events.

For more information about adding trusted objects, see Trusted certificate authority objects.

Related topic: TLS 1.3 decryption best practices.

Related Topics

TLS 1.3 decryption best practices

Incoming traffic decryption actions

Both the **Decrypt - Replace Cert** and **Decrypt - Known Key** decryption actions enable you to decrypt incoming traffic to a server that you want to protect from malicious attacks.

Decryption action comparison

The **Decrypt - Replace Cert** decryption action is provided as an alternative to **Decrypt - Known Key** for these reasons:

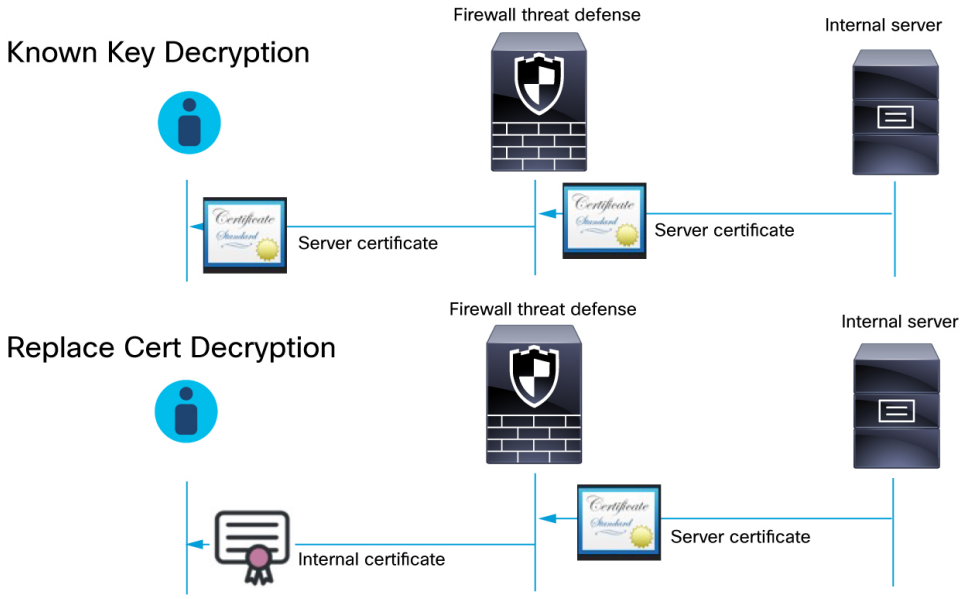
- Certificates can expire or need to be replaced anytime.

  If you choose the **Decrypt - Known Key** action, decryption can stop and the rule can stop functioning when a certificate is replaced for any reason.

- With **Decrypt - Replace Cert**, the connection matches and traffic is decrypted. Deploying the certificate on the device hours or days later is not an issue.

- **Decrypt - Replace Cert** gives you the flexibility to have separate internal and external facing certificates. For example, you could use `payroll.example.local` as an internal server hostname, but use `payroll.example.com` as an externally-facing hostname. You can also use an Enterprise PKI certificate for an internal server but use DigiCert or some other public provider for my external certificate.

The figure shows the difference between **Decrypt - Replace Cert** and **Decrypt - Known Key** decryption actions.



This table provides more information.

	Known Key	Replace Cert
Must match certificate installed on internal server or resource	Yes	No
The device must trust the internal certificate (that is, it must be chained to Trusted CAs list in the decryption policy)	No	No
Certificate hostnames can be different between device-installed certificate and that on the internal server	No, certificates must be identical. (Although multiple SANs can be on the certificates to accommodate different internal and external host names.)	Yes, certificates and certificate host names can be completely different, as long as the internal certificate is trusted by the user's device.
Works if the internal server certificate is expired	Yes – As long as it matches the certificate installed on the device.	No – an expired certificate causes broken trust, and therefore a mismatch.

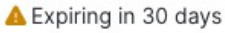
Replace an internal certificate (decrypt - replace cert only)

Replace expiring internal certificates in decryption policies to maintain secure encrypted traffic inspection capabilities.

This task discusses how to replace the internal certificate used in an incoming decryption rule using the Cloud-Delivered Firewall Management Center.

You can also replace the certificate using the API as discussed in *Cloud-Delivered Firewall Management Center REST API Quick Start Guide*.

The system indicates that an internal certificate is *expiring* when today's date is within 30 days of its expiration date. The following figure shows an example.



Before you begin

You must choose the **Decrypt - Replace Cert** decryption rule action in one of these ways:

- In a rule-based decryption policy by clicking the action from the rule's **Action** list.
- Create a standard decryption policy, which always uses the **Decrypt - Replace Cert** rule action.

Follow these steps to replace an internal certificate using the Decrypt - Replace Cert rule action:

Procedure

Step 1 Log in to Security Cloud Control if you haven't already done so.

Step 2 Click **Firewall**.

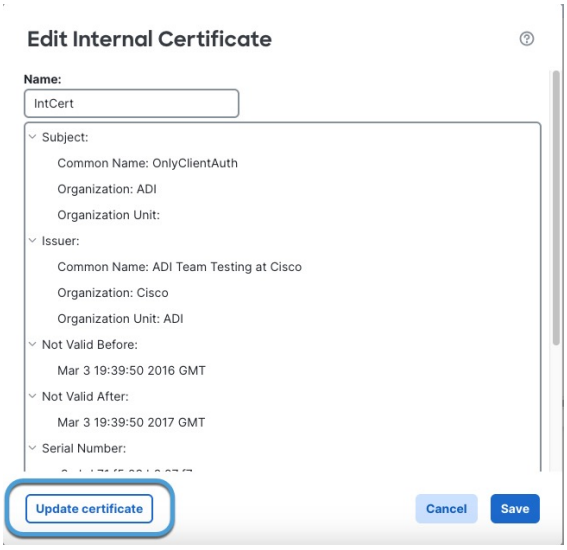
Step 3 Click **Administration** > **(name of** Cloud-Delivered Firewall Management Center**)** > **Objects** > **PKI** > **Internal Certs**.

Step 4 Locate the internal certificate that is expired or expiring (for example, an expired certificate is indicated by 

).

Step 5 Click **Edit** (✎).

Step 6 Click **Update Certificate** as this figure shows.



Step 7 Enter the requested information; see Upload an internal certificate for inbound protection for more information.

Step 8 Follow the prompts on your screen to complete the action.

Decrypt - Resign rule action

The **Decrypt - Resign** rule action is a decryption rule action that enables you to decrypt *outgoing* traffic so the traffic can be subjected to deep inspection.

Select internal certificate objects

This task enables you to create a decryption rule with a **Decrypt - Known Key** action by selecting and assigning internal certificates to decrypt server traffic. You can add, manage, and save certificate selections for decryption rules in your configuration.

Use this procedure when configuring decryption rules that require internal certificates to decrypt traffic from specific servers.

Before you begin

To create a decryption rule with a **Decrypt - Known Key** action, you must add internal certificates for the servers to decrypt traffic. For more information, see Internal certificate objects.

Procedure

Step 1 Click **Edit** ( ) in the field next to the decryption rule **Action**.

Step 2 In the Select Internal Certificate Objects dialog box, do the following:
Available Certificates list: Click the name of a certificate and click **Add to Rule** to use it in this rule. You can add one or more certificates to the rule.

Selected Certificates list: Displays internal certificates used in the rule. To remove a certificate, click **Delete** ( ).

Step 3 To save these changes, click **OK**.

Monitor TLS/SSL hardware acceleration

TLS/SSL hardware acceleration monitoring is a set of topics that discuss how to monitor the status of TLS/SSL hardware acceleration.

Informational counters

Monitor TLS/SSL hardware acceleration performance and interpret KEY counter values, such as PRIV_KEY_RECV and SECU_PARAM_RECV, to assess system load and connection activity using the **show counters** command.

If a system under load is working well, you should see large counts for the following counters. Because there are two sides to the tracker process per connection, you can see these counters increase by two per connection. The PRIV_KEY_RECV and SECU_PARAM_RECV counters are the most important, and are highlighted. The CONTEXT_CREATED and CONTEXT_DESTROYED counters relate to the allocation of cryptographic chip memory.

```
> show counters
Protocol      Counter                          Value   Context
SSLENC        CONTEXT_CREATED                 258225   Summary
SSLENC        CONTEXT_DESTROYED               258225   Summary
TLS_TRK       OPEN_SERVER_SESSION             258225   Summary
TLS_TRK       OPEN_CLIENT_SESSION             258225   Summary
TLS_TRK       UPSTREAM_CLOSE                  516450   Summary
TLS_TRK       DOWNSTREAM_CLOSE                516450   Summary
TLS_TRK       FREE_SESSION                    516450   Summary
TLS_TRK       CACHE_FREE                      516450   Summary
TLS_TRK       PRIV_KEY_RECV                   258225   Summary
TLS_TRK       NO_KEY_ENABLE                   258225   Summary
TLS_TRK       SECU_PARAM_RECV                 516446   Summary
TLS_TRK       DECRYPTED_ALERT                 258222   Summary
TLS_TRK       DECRYPTED_APPLICATION         33568976   Summary
TLS_TRK       ALERT_RX_CNT                    258222   Summary
TLS_TRK       ALERT_RX_WARNING_ALERT          258222   Summary
TLS_TRK       ALERT_RX_CLOSE_NOTIFY           258222   Summary
TCP_PRX       OPEN_SESSION                    516450   Summary
TCP_PRX       FREE_SESSION                    516450   Summary
TCP_PRX       UPSTREAM_CLOSE                  516450   Summary
TCP_PRX       DOWNSTREAM_CLOSE                516450   Summary
TCP_PRX       FREE_CONN                       258222   Summary
TCP_PRX       SERVER_CLEAN_UP                 258222   Summary
TCP_PRX       CLIENT_CLEAN_UP                 258222   Summary
```

Alert counters

Alert counters identify potential issues based on standardized alert types.

These counters are based on the TLS 1.2 specification. FATAL or BAD alerts could indicate issues. However, ALERT_RX_CLOSE_NOTIFY is normal.

For details, see RFC 5246 section 7.2.

```
TLS_TRK       ALERT_RX_CNT                      311   Summary
TLS_TRK       ALERT_TX_CNT                        2   Summary
TLS_TRK       ALERT_TX_IN_HANDSHAKE_CNT           2   Summary
TLS_TRK       ALERT_RX_IN_HANDSHAKE_CNT           2   Summary
TLS_TRK       ALERT_RX_WARNING_ALERT            308   Summary
TLS_TRK       ALERT_RX_FATAL_ALERT                3   Summary
TLS_TRK       ALERT_TX_FATAL_ALERT                2   Summary
TLS_TRK       ALERT_RX_CLOSE_NOTIFY             308   Summary
TLS_TRK       ALERT_RX_BAD_RECORD_MAC             2   Summary
TLS_TRK       ALERT_TX_BAD_RECORD_MAC             2   Summary
TLS_TRK       ALERT_RX_BAD_CERTIFICATE            1   Summary
```

Error counters

Error counters help you interpret system performance data to identify potential issues and optimize performance using the show counters command.

Fatal counters

Monitor and interpret TLS/SSL hardware acceleration fatal counters to identify serious errors, assess system health, and troubleshoot persistent crypto acceleration issues on Cisco devices. Reset or review counters for proactive system maintenance and support.

Fatal counters indicate serious errors. These counters should be at or near 0 on a healthy system. This is an example list of the fatal counters.

```
> show counters
Protocol     Counter                          Value   Context
CRYPTO       RING_FULL                            1   Summary
CRYPTO       ACCELERATOR_CORE_TIMEOUT             1   Summary
CRYPTO       ACCELERATOR_RESET                    1   Summary
CRYPTO       RSA_PRIVATE_DECRYPT_FAILED           1   Summary
```

The RING_FULL counter is not a fatal counter, but indicates how often the system overloaded the cryptographic chip. The ACCELERATOR_RESET counter is the number of times the TLS crypto acceleration process failed unexpectedly, which also causes the failure of pending operations, which are the numbers you see in ACCELERATOR_CORE_TIMEOUT and RSA_PRIVATE_DECRYPT_FAILED.

If you have persistent problems, disable TLS crypto acceleration (or **config hwCrypto disable**) and work with Cisco TAC to resolve the issues.

Note You can do additional troubleshooting using the **show snort tls-offload** and **debug snort tls-offload** commands. Use the **clear snort tls-offload** command to reset the counters displayed in the **show snort tls-offload** command to zero.

Troubleshoot Rule-based decryption rules

This topic discusses how to troubleshoot rule-based decryption rules.

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.

TLS/SSL oversubscription details

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.

Troubleshoot TLS/SSL oversubscription

Monitor and troubleshoot TLS/SSL oversubscription by configuring decryption policies, enabling logging, and adding SSL Flow Flags to connection event views in Cloud-Delivered Firewall Management Center.

If your managed device has TLS crypto acceleration enabled, you can view connection events to determine whether or not the devices are experiencing SSL oversubscription. You must add at least the **SSL Flow Flags** event to the table view of connection events.

Before you begin

- Configure a decryption policy with a setting for **Handshake Errors** on **Undecryptable Actions** page.

  For more information, see Set default handling for undecryptable traffic.

- Enable logging for your SSL rules as discussed in the section on logging decryptable connections in decryption rules in the Secure Firewall Management Center and Threat Defense Management Network Administration guide.

Follow these steps to troubleshoot TLS/SSL oversubscription:

Procedure

Step 1 Click **Events & Logs** > **Analysis** > **Unified Events**

Step 2 Click the colum picker ( ) icon and add additional columns **TLS Flow Flags** and **TLS Flow Messages**.

Step 3 Click **Apply**.
TLS/SSL oversubscription is indicated by the values of `ERROR_EVENT_TRIGGERED` and `OVER_SUBSCRIBED` in the **SSL Flow Flags** column.

Step 4 If TLS/SSL oversubscription is occurring, log in to the managed device and enter any of the following commands:

Command	Result
show counters	If the value of **TCP_PRX BYPASS_NOT_ENOUGH_MEM** is large, consider upgrading your device to one with a larger capacity for SSL traffic or use **Do NOT Decrypt** rules for lower priority encrypted traffic.
show snort TLS-offload	If the value of **BYPASS_NOT_ENOUGH_MEM** is large, consider upgrading your device to one with a larger capacity for SSL traffic or use **Do NOT Decrypt** rules for lower priority encrypted traffic.

TLS heartbeat

A TLS heartbeat extension enables applications to confirm active TLS or DTLS connections.

TLS heartbeat handling

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

Related Topics

Troubleshoot TLS heartbeat, on page 42

Troubleshoot TLS heartbeat

This task helps you identify TLS heartbeat activity in your network to determine if applications are using the TLS heartbeat extension.

If your managed device has TLS crypto acceleration enabled, you can view connection events to determine whether or not the devices are seeing traffic with the TLS heartbeat extension. You must add at least the **SSL Flow Messages** event to the table view of connection events.

TLS heartbeat is indicated by the value of `HEARTBEAT` in the **SSL Flow Messages** column in the table view of connection events.

Before you begin

To determine if applications in your network use TLS heartbeat, first perform these tasks:

- Configure an decryption policy with a setting for **Decryption Errors** on **Undecryptable Actions** page.

  For more information, refer to Set default handling for undecryptable traffic.

- Enable logging for your SSL rules as discussed in Secure Firewall Management Center and Threat Defense Management Network Administration.

Follow these steps to troubleshoot TLS heartbeat:

Procedure

Step 1 Click **Events & Logs** > **Analysis** > **Unified Events**

Step 2 Click the colum picker ( ) icon and add additional columns **TLS Flow Flags** and **TLS Flow Messages**.

Step 3 Click **Apply**.
TLS heartbeat is indicated by the value of `HEARTBEAT` in the **SSL Flow Messages** column.

Step 4 If applications in your network use TLS heartbeat, refer to Rule-based decryption rules guidelines and limitations, on page 2.

TLS/SSL pinning

TLS/SSL pinning is a security mechanism that

- validates server certificates against a predetermined set of trusted certificates or keys
- prevents man-in-the-middle attacks by rejecting connections to servers with unexpected certificates, and
- requires applications to maintain a local store of trusted certificate information for validation.

TLS/SSL pinning implementation details

Some applications use a technique referred to as *TLS/SSL pinning* or *certificate pinning*, which embeds the fingerprint of the original server certificate in the application itself. As a result, if you configured a decryption rule with a **Decrypt - Resign** action, when the application receives a resigned certificate from a managed device, validation fails and the connection is aborted.

To confirm that TLS/SSL pinning is occurring, attempt to log in to a mobile application like Facebook. If a network connection error is displayed, log in using a web browser. (For example, you *cannot* log in to a Facebook mobile application but *can* log in to Facebook using Safari or Chrome.) You can use Firepower Management Center connection events as further proof of TLS/SSL pinning

Note TLS/SSL pinning is not limited to mobile applications.

If applications in your network use SSL pinning, see TLS/SSL certificate pinning guidelines, on page 8.

Related Topics

Troubleshoot TLS/SSL pinning, on page 43

Troubleshoot TLS/SSL pinning

This task helps you identify and confirm whether applications in your network are using TLS/SSL pinning by analyzing SSL connection events and flow DATA.

You can view connection events to determine whether or not the devices are experiencing TLS/SSL pinning. You must add at least the **TLS Flow Flags** and **TLS Flow Messages** columns to the table view of connection events.

Before you begin

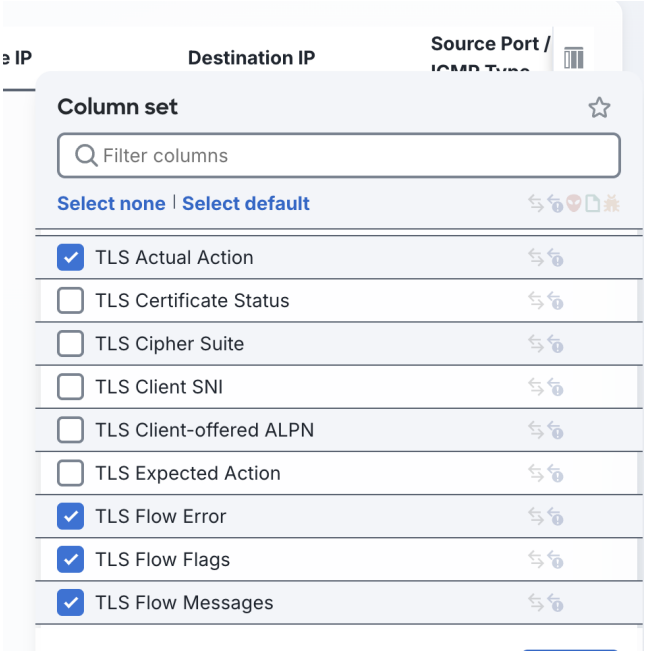
- Enable logging for your decryption rules as discussed in the section on logging decryptable connections in decryption rules in the Secure Firewall Management Center and Threat Defense Management Network Administration guide.
- Log in to a mobile application like Facebook. If a network connection error displays, log in to Facebook using Chrome or Safari. If you *can* log in using a web browser but not the native application, TLS/SSL pinning is likely occurring.

Follow these steps to troubleshoot TLS/SSL pinning:

Procedure

Step 1 Click **Events & Logs** > **+ Show more** > **Connection** > **Events**.

Step 2 In the table view, click the Column set icon and select the required columns.



Step 3 Click **Apply**.

Step 4 Analyze the SSL flow DATA to identify SSL pinning behavior.

Step 5 If you determine that applications in your network use SSL pinning, see Rule-based decryption rules guidelines and limitations, on page 2.

What to do next

You can use TLS/SSL connection events to confirm TLS/SSL pinning is occurring by looking for any of the following:

- Applications that send an SSL ALERT Message as soon as the CLIENT receives the SERVER_HELLO, SERVER_CERTIFICATE, SERVER_HELLO_DONE message from the SERVER, followed by a TCP Reset, exhibit the following symptoms. (The ALERT, `Unknown CA (48)`, can be viewed using a packet capture.)
  - The SSL Flow Flags column displays `ALERT_SEEN` but *not* `APP_DATA_C2S` or `APP_DATA_S2C`.
  - If your managed device has SSL hardware acceleration enabled, the SSL Flow Messages column typically displays: `CLIENT_ALERT`, `CLIENT_HELLO`, `SERVER_HELLO`, `SERVER_CERTIFICATE`, `SERVER_KEY_EXCHANGE`, `SERVER_HELLO_DONE`.

- If your managed device doesn't support SSL hardware acceleration or if the feature is disabled, the SSL Flow Messages column typically displays: `CLIENT_HELLO`, `SERVER_HELLO`, `SERVER_CERTIFICATE`, `SERVER_KEY_EXCHANGE`, `SERVER_HELLO_DONE`.
- `Success` is displayed in the SSL Flow Error column.

- Applications that send no alerts but instead send TCP Reset after the SSL handshake is FINISHED exhibit the following symptoms:
  - The SSL Flow Flags column does *not* display `ALERT_SEEN`, `APP_DATA_C2S`, or `APP_DATA_S2C`.
  - If your managed device has SSL hardware acceleration enabled, the SSL Flow Messages column typically displays: `CLIENT_HELLO`, `SERVER_HELLO`, `SERVER_CERTIFICATE`, `SERVER_KEY_EXCHANGE`, `SERVER_HELLO_DONE`, `CLIENT_KEY_EXCHANGE`, `CLIENT_CHANGE_CIPHER_SPEC`, `CLIENT_FINISHED`, `SERVER_CHANGE_CIPHER_SPEC`, `SERVER_FINISHED`.
  - If your managed device doesn't support SSL hardware acceleration or if the feature is disabled, the SSL Flow Messages column typically displays: `CLIENT_HELLO`, `SERVER_HELLO`, `SERVER_CERTIFICATE`, `SERVER_KEY_EXCHANGE`, `SERVER_HELLO_DONE`, `CLIENT_KEY_EXCHANGE`, `CLIENT_CHANGE_CIPHER_SPEC`, `CLIENT_FINISHED`,`SERVER_CHANGE_CIPHER_SPEC`, `SERVER_FINISHED`.
  - `Success` is displayed in the SSL Flow Error column.

Related Topics

Troubleshoot unknown or bad certificates or certificate authorities, on page 45

Troubleshoot unknown or bad certificates or certificate authorities

Enable logging for TLS/SSL decryption rules to monitor connection events and identify certificate-related issues. This helps you diagnose problems with certificate authorities, bad certificates, or unknown certificates in your TLS/SSL connections.

You can view connection events to determine whether or not the devices are experiencing unknown certificate authorities, bad certificates, or unknown certificates. This procedure can also be used if a TLS/SSL certificate has been pinned. You must add at least the and **TLS Flow Messages** columns to the table view of events.

Before you begin

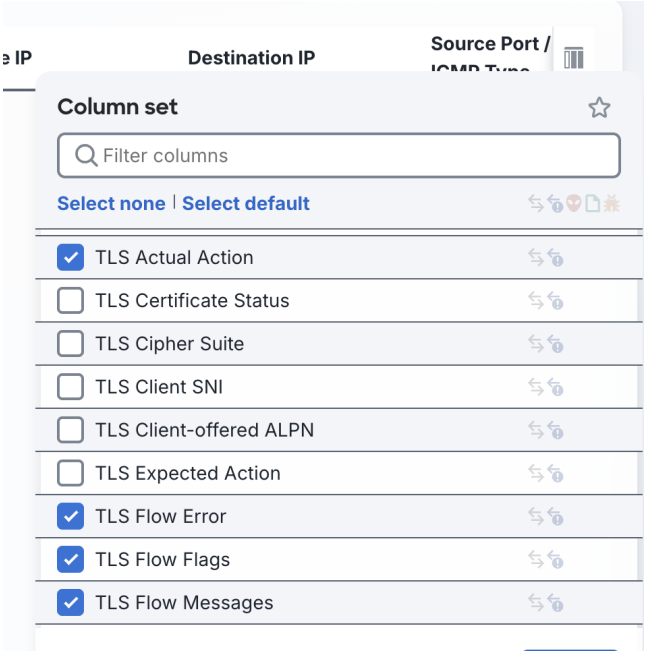
- Set up a decryption rule.
- Enable logging for your decryption rules as discussed in the section on logging decryptable connections in decryption rules in the Secure Firewall Management Center and Threat Defense Management Network Administration guide.

Follow these steps to troubleshoot unknown or bad certificates or certificate authorities:

Procedure

Step 1 Click **Events & Logs** > **+ Show more** > **Connection** > **Events**.

Step 2 In the table view, click the Column set icon and select the required columns.



Step 3 Click **Apply**.

Step 4 Review the SSL flow flags to determine if a certificate or certificate authority is bad or missing.

SSL flow flag	Meaning
CLIENT_ALERT_SEEN_UNKNOWN_CA	Indicates a valid certificate chain or partial chain was received by an SSL client application, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA. This message always indicates an unrecoverable error.
CLIENT_ALERT_SEEN_BAD_CERTIFICATE	A certificate was corrupt, contained signatures that did not verify correctly, or had other problems.
CLIENT_ALERT_SEEN_CERTIFICATE_UNKNOWN	Some other (unspecified) issue arose in processing the certificate, rendering it unacceptable.

Verify TLS/SSL cipher suites

This task helps you resolve cipher suite incompatibility errors that occur when saving a decryption rule that has cipher suite conditions. The error indicates that one or more of the cipher suites you chose for the decryption rule condition are incompatible with the certificate used in the decryption rule.

This topic discusses actions you must take if you see the following error when saving a decryption rule that has cipher suite conditions:

```
Traffic cannot match this rule; none of your selected cipher suites contain a signature
algorithm that matches the resigning CA's signature algorithm
```

Before you begin

To resolve the issue, you must have access to the certificate you're using.

Note The tasks in this topic assume knowledge of how TLS/SSL encryption works.

Follow these steps to verify TLS/SSL cipher suites:

Procedure

Step 1 When you attempt to save a decryption rule with either **Decrypt - Resign** or **Decrypt - Known Key** with specified cipher suites, the following error is displayed:

Example:

```
Traffic cannot match this rule; none of your selected cipher suites contain a
signature algorithm that the resigning CA's signature algorithm
```

Step 2 Locate the certificate you're using to decrypt traffic and, if necessary, copy the certificate to a system that can run openssl commands.

Step 3 Run the following command to display the signature algorithm used by the certificate:

Example:

openssl x509 -in *CertificateName* **-text -noout**

The first few lines of output are displayed similar to the following:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 4105 (0x1009)
    Signature Algorithm: ecdsa-with-SHA256
```

Step 4 The **Signature algorithm** tells you the following:

- The cryptographic function used (in the preceding example, **ECDSA** means Elliptic Curve Digital Signature Algorithm).
- The hash function used to create a digest of the encrypted message (in the preceding example, **SHA256**).

Step 5 Search a resource such as *OpenSSL at University of Utah* for cipher suites that match those values. The cipher suite must be in RFC format.
You can also search a variety of other sites, such as Server Side TLS at the Mozilla wiki or Appendix C of RFC 5246. Cipher Suites in TLS/SSL (Schannel SSP) in Microsoft documentation has a detailed explanation of cipher suites.

Step 6 If necessary, translate the OpenSSL name to an RFC name that the Firepower Management System uses. See the RFC mapping list on the `https://testssl.sh` site.

Step 7 The previous example, **ECDSA-with-SHA256**, can be found in the Modern Compatibility List on the Mozilla wiki.

Choose only cipher suites that have **ECDSA** and **SHA-256** in the name. These cipher suites follow:

```
ECDHE-ECDSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES128-SHA256
```

Find the corresponding RFC cipher suite on RFC mapping list. These cipher suites follow:

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
```

Step 8 Add the preceding cipher suites to your decryption rule.

Troubleshoot using crypto archives

Crypto issues are difficult to triage. Crypto archives help you to troubleshoot these issues. A crypto archive contains crypto session information about the crypto request, peer information, the component that sent the crypto request, and the timed-out crypto session information.

About crypto archives

Threat defense does not save keys and initialization vectors (IVs) for the session. For SSL and IPsec, you can also view these information:

- For SSL: Session SSL version, source, destination IP addresses, and ports.
- For IPsec: IPsec security association information.

A ring can hold 2000 crypto command entries. Threat defense pushes the crypto command in one of the rings and pulls out the result after completing the crypto request. Crypto archive files now have the timed-out crypto request's ring and entry index. The ring and its entry index help in troubleshooting problematic crypto commands.

The crypto archive has two formats: a text file and a binary file. You can use the **debug menu ctm 103** command to decode the binary file. The crypto archive files are available at disk0:/crypto_archive.

For example:

```
FTD# debug menu ctm 103 crypto_eng0_arch_4.bin
[Nitrox V Archive Header v1.0 Info]
ASA Image Version: PIX (9.20) #0: Tue Mar 29 16:20:30 GMT 2022
...
SE SSL microcode: CNN5x-MC-SE-SSL-0011
AE microcode: CNN5x-MC-AE-MAIN-0002
Crypto Engine 0
Crash type: SE Ring Timeout
...
Core Soft Resets: 11
...
Timeout Ring (SE): 12
Timeout Entry: 642
SE TIMEOUT:
Core SE 6 Touts: 2
Core SE 8 Touts: 2
Core SE 12 Touts: 4
Core SE 32 Touts: 2
Core SE 37 Touts: 1
.....
[Timeout Session Info]
```

```
Active: TRUE
Sync: FALSE
Callback: TRUE
Saved Callback: FALSE
Commands in progress: 1
Engine : hardware
Device : n5 (Nitrox V)
Session : ssl
Priority: normal
NP VPN context handle : 0x00000000
Flag : 0
vcid : 0
Block size : 2050
async cb ring index: 0
tls offload rsa: FALSE
Session context:
SSL Version : dtls1.2
SSL Context Type : handshake
Encryption Mode : gcm
Auth Algorithm : null
Hash Algorithm : none
Key Size : 32
SSL V : dtls1.2
Source IP : 82.1.2.2
Source Port : 51915
Dest IP : 82.29.155.32
Dest Port : 443
```

In this example, the highlighted information shows the timeout ring, the crash time (timeout entry), and SSL session information.

Supported devices for crypto archives

These devices with Nitrox V crypto accelerator support crypto archives:

- Cisco Firepower 3105, 3110, 3120, 3130, 3140
- Cisco Firepower 4112, 4115, 4125, 4145
- Cisco Firepower 9300 SM-40, SM-48 and SM-56
- Cisco Secure Firewall 4200