

Implementing Secure Logging

This chapter describes the implementation of secure logging over Transport Layer Security (TLS). TLS, the successor of Secure Socket Layer (SSL), is an encryption protocol designed for data security over networks.

Table 1: Feature History Table

Release	Modification
Release 7.0.1	This feature was introduced.

- System Logging over Transport Layer Security (TLS), on page 1
- Restrictions for Syslogs over TLS, on page 3
- Configuring Syslogs over TLS, on page 3
- Security template framework for TLS applications, on page 5
- TLS RFC 5289 compliance for security template, on page 13

System Logging over Transport Layer Security (TLS)

System Log (syslog) messages indicate the health of the device and provide valuable information about any problems encountered. By default, the syslog process sends messages to the console terminal.

Due to limited size of the logging buffer in a router, these syslog messages get overwritten in a short time. Moreover, the logging buffer doesn't retain syslogs across router reboots. To avoid these issues, you can configure the router to send syslog messages to an external syslog server for storage.

Note For more information on configuring system logging, see *Implementing System Logging* chapter in the *System Monitoring Configuration Guide for Cisco NCS 540 Series Routers*

Traditionally, routers transfer syslogs to an external syslog server using User Datagram Protocol (UDP), which is an insecure way of transferring logs. To guarantee secure transport of syslogs, Cisco NCS 540 Series Router supports Secure Logging based on RFC 5425 (Transport Layer Security Transport Mapping for Syslog). With this feature, the router sends syslogs to a remote server, over a trusted channel which implements the secure Transport Layer Security (TLS) encryption protocol.

TLS ensures secure transport of syslogs by:

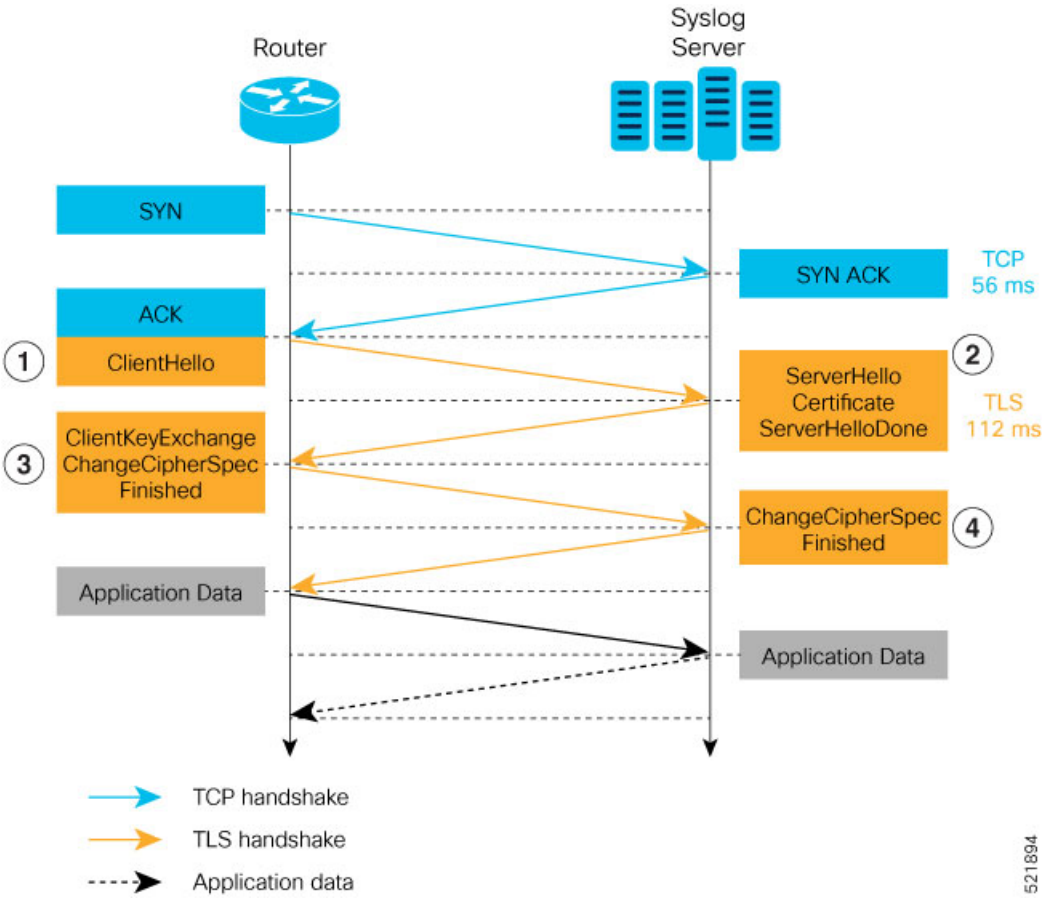
- Authenticating the server and client

- Encrypting the syslog data transferred
- Verifying the integrity of data

The Cisco NCS 540 Series Router is the TLS client and remote syslog server is the TLS server. TLS runs over Transmission Control Protocol (TCP). So, the client must complete the TCP handshake with the server before starting TLS handshake.

Sequence of TLS Handshake

Figure 1: TLS Handshake



To establish the TLS session, the following interactions take place between the router and the syslog server after TCP handshake is complete:

1. The router sends Client Hello message to the server to begin TLS handshake.
2. The server shares its TLS certificate, which contains its public key and a unique session key, with the router to establish a secure connection. Each TLS certificate consists of a key pair made of a public key and private key.
3. The router confirms the server certificate with the Certification Authority and checks the validity of the TLS certificate. Then, the router sends a Change Cipher Spec message to the server to indicate that messages sent are encrypted using the negotiated key and algorithm.

4. The server decrypts the message using its private key. And then, sends back a Change Cipher Spec message encrypted with the session key to complete the TLS handshake and establish the session.

For more information on configuring Certification Authority interoperability, refer *Implementing Certification Authority Interoperability* chapter in this guide.

Restrictions for Syslogs over TLS

The following restrictions apply for sending syslogs to a remote syslog server over TLS:

- When you configure the remote syslog server on the router, specify only one identifier for a remote syslog server: its hostname or its IPv4 or IPv6 address. A single server identifier gives the TLS configuration one unambiguous endpoint identity.

Configuring Syslogs over TLS

The following steps show how to configure syslog over TLS:

1. Configure the trust-point for establishing the TLS channel as shown:

```
Router#conf t
Router(config)#crypto ca trustpoint tp
Router(config-trustp)#subject-name CN=new
Router(config-trustp)#enrollment terminal
Router(config-trustp)#rsakeypair k1
Router(config-trustp)#commit
```

Note You can either use the command **enrollment url** *SCEP-url* or the command **enrollment terminal** for configuring trustpoint certification authority (CA) enrollment. For more information, see *Implementing Certification Authority Interoperability* chapter in this guide.

2. Configure the settings to access the remote syslog server. You can use either the IPv4/v6 address of the server or the server hostname for this configuration. Based on the configured **severity**, the router sends syslogs to the server. Logging severity options include **alerts, critical, debugging, emergencies, errors, informational, notifications and warnings**. For more information about logging severity levels, see the topic *Syslog Message Severity Levels* in *Implementing System Logging* chapter in *System Monitoring Configuration Guide for Cisco NCS 540 Series Routers*.

   This example shows you how to configure syslog server settings with the IPv4 address.

```
Router(config)#logging tls-server TEST
Router(config-logging-tls-peer)#severity debugging
Router(config-logging-tls-peer)#trustpoint tp
Router(config-logging-tls-peer)#address ipv4 10.105.230.83
Router(config-logging-tls-peer)#commit
```

   Alternately, you can configure the syslog server settings with server hostname instead of the IPv4/v6 address.

```
Router(config)#logging tls-server TEST
Router(config-logging-tls-peer)#severity debugging
Router(config-logging-tls-peer)#trustpoint tp
```

```
Router(config-logging-tls-peer)#tls-hostname xyz.cisco.com
Router(config-logging-tls-peer)#commit
```

3. Configure the domain to map the IP address of the remote syslog server and its hostname.

```
Router(config)#domain ipv4 host xyz.cisco.com 10.105.230.83
Router(config)#domain name cisco.com
Router(config)#commit
```

Verification Steps

TCP port 6514 is the default port for syslog over TLS. Verify the TLS configuration by checking if port 6514 is associated with the IP address of the syslog server in the output of the command **show lpts bindings brief**.

```
Router#show lpts bindings brief

@ - Indirect binding; Sc - Scope

 Location   Clnt Sc L3   L4    VRF-ID    Interface    Local-Address,Port Remote-Address,Port

 ---------- ---- -- ---- ------ --------- ------------ ------------------------------------

  0/RP0/CPU0 TCP LR IPV4 TCP    default   any          5.10.18.5,35926 10.105.230.83,6514
```

The output of **show logging** command displays the IP address of the TLS server and the number of messages sent to the remote syslog server.

```
Router#show logging

Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 185 messages logged
    Monitor logging: level debugging, 94 messages logged
    Trap logging: level informational, 0 messages logged
    Logging to TLS server 10.105.230.83, 66 message lines logged
    Buffer logging: level debugging, 183 messages logged

Log Buffer (2097152 bytes):
...................
```

The output of **show crypto ca certificates** command displays the Certification Authority (CA) certificate details.

```
Router#show crypto ca certificates

Trustpoint       : tp
================================================
CA certificate
  Serial Number  : B5:68:C8:96:A4:7C:1A:BA
  Subject:
      CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Issued By      :
      CN=cacert,OU=SPBU,O=CSCO,L=BGL,ST=KA,C=IN
  Validity Start : 05:39:51 UTC Tue Aug 13 2019
  Validity End   : 05:39:51 UTC Mon Aug 08 2039

  CRL Distribution Point
      http://10.105.236.78/crl_xxx/crl.der
  SHA1 Fingerprint:
      03BD57E04A2AA4648A84F515A46EF99CCF488387
```

When the TLS channel between the router and syslog server comes up, the router displays the following syslog messages on the console:

```
RP/0/RP0/CPU0: syslogd[148]: %SECURITY-XR_SSL-6-CERT_VERIFY_INFO : SSL Certificate
verification: Peer certificate verified successfully
RP/0/RP0/CPU0: syslogd[148]: %OS-SYSLOG-5-LOG_NOTICE : Secure Logging: Successfully
established TLS session , server :10.105.230.83
```

Security template framework for TLS applications

A security template is a configuration bundle that:

- centralizes and standardizes security policy configuration for TLS-enabled applications,
- encapsulates certificate authentication policies, TLS protocol controls, and compliance mode settings, and
- acts as a reusable, named source of truth that multiple applications can reference instead of embedding security settings locally.

Security templates simplify management by letting applications reference a named template, avoiding duplicated or inconsistent security settings.

Table 2: Feature History Table

Feature Name	Release Information	Feature Description
Security template framework for TLS enabled applications	Release 25.4.1	Security templates reduce misconfiguration risks and operational overhead by centralizing and standardizing security policy configuration for TLS-enabled applications. A security template bundles certificate authentication policy, TLS controls, and compliance mode settings. It acts as a single source of truth that applications reference, avoiding local embedding of security settings. This template defines how certificates are handled and controls various aspects of the TLS handshake. The feature introduces these changes: CLI: • **security-template**

The security template infrastructure centralizes and standardizes the configuration of certificate-based authentication and TLS settings for applications. It provides a reusable, named configuration bundle—called a security template—which can be referenced by multiple applications to enforce consistent security policies and reduce operational overhead.

Security templates reduce operational overhead and misconfiguration risk by enabling administrators to manage security as reusable templates. They support features such as:

- Version and cipher suite control for TLS,
- Selection of elliptic curve groups and digital signature algorithms,
- Integration with certificate onboarding methods such as SCEP, file-based, and Certz profiles,
- Change notification and automatic configuration updates for registered applications.

Key Terminologies

- **Certificate authentication policy:** A set of rules specifying how certificates are used for authenticating servers or clients, including trust anchor selection and certificate validation settings.
- **Common Criteria (CC) mode:** An enhanced security mode that enforces stricter compliance-focused behavior.
- **Certz profile:** A profile providing identity certificates, private keys, and CA bundles from a centralized certificate management service. For more information, see github.

Benefits of Security template framework

Security template framework offers these benefits:

- Centralized Security Management: It allows you to create and manage security templates that apply to multiple applications.
- Flexibility: The template supports configurations like Common Criteria (CC) mode and certificate authentication policies. It also allows for future expansion with additional security policies.
- Simplified Configuration: Centralized security templates reduce the complexity of managing application-specific settings.
- Consistency and Compliance: The uniform application of policies ensures adherence to organizational and regulatory standards.
- Extensibility: The system easily adapts to future security needs. It accommodates additional fields and configurations.
- Operational Efficiency: Automated notifications minimize errors and enable faster updates.

Use cases

Examples of use cases for the security template framework include:

- Managing TLS settings for Syslog.
- Supporting web-scale environments with numerous root CAs and custom certificate validation needs.

- Enforcing advanced crypto policies per application. For example, specific elliptic curves and signature algorithms.

Configuration guidelines for security template framework

Consider these best practices and guidelines to ensure scalable, consistent, and secure management of TLS and certificate configurations for all supported applications.

- Ensure that you centralize security policy configuration for TLS-enabled applications using security templates. This standardizes certificate authentication, TLS controls, and compliance settings across services, reducing operational overhead and misconfiguration risk.
- Use a unique name for each security template and apply it as the single source of truth for applications, instead of embedding security settings locally.
- Enable Common Criteria (CC) mode for enhanced compliance where required.
- Configure certificate authentication policies to specify identity certificates and peer trust anchors as required. Prefer sourcing certificates from a Certz profile when available, as it takes precedence over individual trustpoints.
- When both a standalone trustpoint and a security template are configured under syslog server, the trustpoint specified within the security template overrides the standalone trustpoint.
- Register applications to receive and handle security template update notifications so that changes are applied dynamically and consistently.

Restriction for security template

Currently, only syslog messages sent over TLS can utilize security templates.

How security template framework works

As organizations scale their network infrastructure and deploy more TLS-enabled applications, managing security settings like certificate authorities, protocol versions, cipher suites, and compliance requirements becomes complex. A centralized security template framework addresses operational inefficiencies, compliance challenges, and risk of misconfiguration by providing reusable, consistent policy bundles.

Summary

Security templates in Cisco IOS XR centralize and standardize the configuration and enforcement of TLS and certificate-based security settings across multiple applications. This approach streamlines management, reduces misconfiguration risk, and ensures consistent policy application.

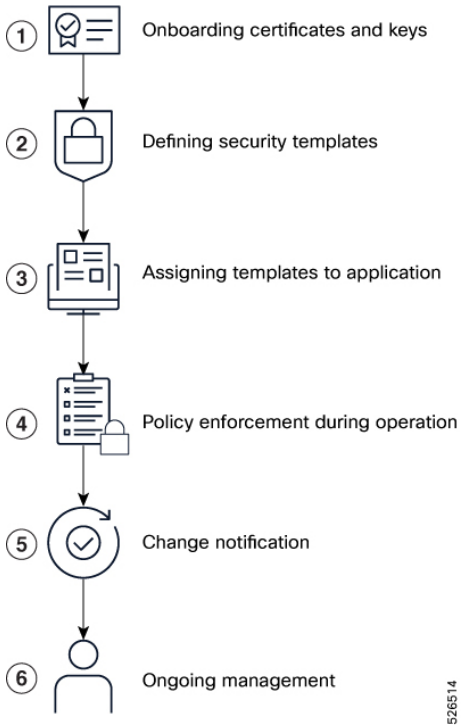
The key components involved in the process are:

- **Security template infrastructure:** Central repository and logic that manages security templates and their application to different XR services.
- **Applications like Syslog:** Consume and enforce security settings by referencing named security templates.
- **Trustpoints and Certz profiles:** Provide certificate and key material for authentication and trust validation.

- **Notification and change management mechanism:** Ensures applications are updated when security templates change.

Workflow

Figure 2: Visual representation of security template process



The process involves the following stages:

1. **Onboarding certificates and keys:** Trustpoints are configured using various onboarding methods (e.g., SCEP, file import) to manage CA certificates, identity certificates, and keys. Certz profiles may also be set up for advanced lifecycle management.
2. **Defining security templates:** An administrator creates a named security template using CLI, specifying settings such as TLS versions, cipher suites, certificate authentication policy, trust anchors, identity sources, and compliance modes.
3. **Assigning templates to applications:** Applications are configured to reference a specific security template rather than embedding their own security settings.
4. **Application registration and initialization:** Each application registers with the security template infrastructure, providing a callback for configuration change notifications and initializing its SSL/TLS context using library APIs.
5. **Policy enforcement during operation:** Applications use the security template's parameters for certificate validation, TLS negotiation, and compliance enforcement during secure communications.
6. **Change notification and dynamic updates:** When a security template or associated certificate material changes, the infrastructure notifies all registered applications, which can then re-initialize their sessions or contexts to apply the new configuration.

7. **Ongoing management:** Administrators can update templates, rotate certificates, and audit compliance centrally, with changes automatically propagated to all consuming applications.

Result

This process enables centralized, consistent, and scalable management of security policies across Cisco IOS XR applications, reducing operational overhead, minimizing misconfiguration risk, supporting rapid adaptation to regulatory requirements, and facilitating secure, compliant communications in large-scale deployments.

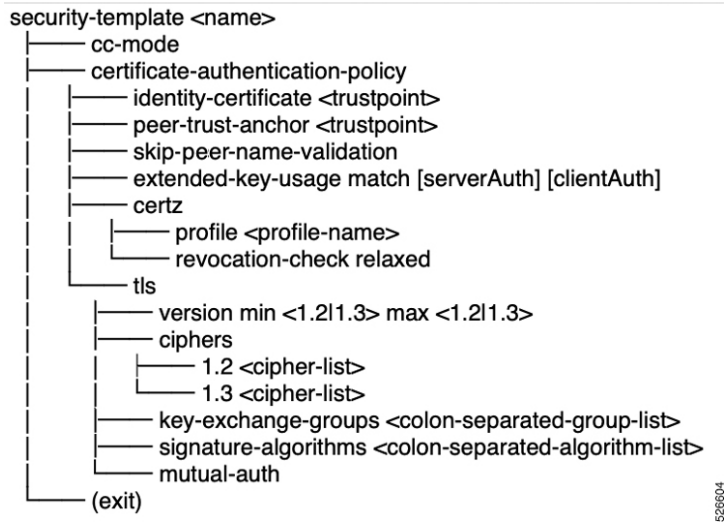
Configure security template

Centralize and standardize the configuration and management of security policies, certificate authentication, and TLS controls for applications running on Cisco routers.

Use a security template to enforce consistent certificate policies, TLS settings, and compliance controls across multiple services and applications, reducing operational overhead and misconfiguration risk.

Configuration concepts and definitions

Figure 3: CLI Structure



Before creating and configuring a security template, you must understand the core concepts and the supported features.

- **Security Template Mode:** You define a template with a unique name.
- **Common Criteria (CC) Mode:** You enable stricter, compliance-focused behavior.
- **Certificate Authentication Policy:** This policy configures how the system handles certificates:
  - **Identity Certificate:** This specifies the trustpoint for the leaf certificate and its chain. The system uses this when a server or client (for mutual TLS or mTLS) must present a certificate chain.
  - **Peer Trust Anchor:** This trustpoint validates server or client certificates. If you configure both a peer identity and a peer trust anchor, the peer trust anchor takes precedence for validating the incoming certificate.

- **Skip peer name Validation:** This disables hostname verification against the certificate's DNS Subject Alternative Name (SAN) or subject Common Name (CN). Applications typically use this only in lab, testing, or controlled environments.
- **Extended Key Usage Match:** This enforces validation of `serverAuth` and `clientAuth` extensions in certificates. It prevents certificates from misuse outside their authorized purpose.
- **Certz Profile:** This specifies the Certz profile to source the identity certificate, private key, and Certificate Authority (CA) bundle from a gNSI Certz profile. This option takes precedence over trustpoints if configured.
- **Certz Revocation Method Relaxed:** Enabling this option prevents session authentication failure if the Certificate Revocation List (CRL) bundle misses a CRL for any certificate in the verified chain.

- **TLS Settings:** These settings control TLS protocol behavior:
  - **Version Range Control:** You configure the minimum and maximum TLS versions that the system can negotiate (e.g., minimum TLS 1.2, maximum TLS 1.3). This ensures compliance with security policies and prevents downgrade attacks.
  - **Cipher Suites per TLS Version:** You define the allowed set of ciphers for each TLS version (e.g., AES-GCM). This provides fine-grained control to disable weak ciphers.
  - **Key Exchange Groups for Negotiation:** You define the elliptic curves or finite field groups (e.g., x25519, secp256r1, ffdhe3072) used during the TLS key exchange process. This ensures the use of modern and secure Diffie–Hellman parameters.
  - **Signature Algorithms Allow-list:** You specify the permitted digital signature algorithms (e.g., rsa_pss_rsae_sha256, ecdsa_secp256r1_sha256) for TLS handshakes. This prevents the use of deprecated options.
  - **Mutual-auth:** This enables mutual TLS (mTLS).

Before you begin

- Ensure required libraries (`libcrypto_xr`, `libssl_xr`) and header files are available.
- Have necessary trustpoints and/or Certz profiles configured for certificate management.
- Confirm the application supports integration with security templates.

Follow these steps to configure a security template for TLS-enabled applications:

Procedure

Step 1 Enter security template mode by defining a unique template name.

Example:

```
Router(config)# security-template template1
```

Step 2 (Optional) Enable Common Criteria (CC) mode if enhanced compliance is required.

Example:

```
Router(config-security-template)# cc-mode
```

Step 3 Enter certificate authentication policy mode to configure certificate-based settings.

- Specify the identity certificate trustpoint for leaf certificate and chain.
- Specify the peer trust anchor trustpoint for validating incoming certificates.
- (Optional) Skip peer name validation if using in lab or controlled environments.
- (Optional) Enforce extended key usage matching for serverAuth and/or clientAuth.
- (Optional) Specify a Certz profile to source certificates and keys.

Example:

```
Router(config-security-template)# certificate-authentication-policy
Router(config-certificate-authentication-polic)# peer-trust-anchor trustpoint1
Router(config-certificate-authentication-polic)# identity-certificate CA2
Router(config-certificate-authentication-polic)# extended-key-usage match serverAuth
Router(config-certificate-authentication-polic)# skip-peer-name-validation
Router(config-certificate-authentication-polic)# certz-profile certz1
```

Step 4 Enter TLS configuration mode to set protocol and cryptographic parameters.

- Define the minimum and maximum TLS versions allowed (e.g., min 1.2, max 1.3)
- Specify allowed cipher suites for each TLS version as needed.
- Set permitted key exchange groups (elliptic curves or Diffie-Hellman groups).
- Specify allowed signature algorithms for TLS handshakes.
- (Optional) Enable mutual authentication (mTLS) if required.

Example:

```
Router(config-certificate-authentication-polic)# tls
Router(config-tls)# ciphers 1.2 AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA25 1.3
TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256
Router(config-tls)# version min 1.2 max 1.3
Router(config-tls)# key-exchange-groups P-256:P-384:X25519
Router(config-tls)# signature-algorithms
rsa_pss_rsae_sha256:rsa_pss_rsae_sha384:rsa_pss_rsae_sha512:rsa_pkcs1_sha256:rsa_pkcs1_sha384:rsa_pkcs1_sha512
```

Example:

Configuration Example:

```
Router(config)# security-template template1
Router(config-security-template)# cc-mode
Router(config-security-template)# certificate-authentication-policy
Router(config-certificate-authentication-polic)# tls
Router(config-tls)# ciphers 1.2 ECDHE-RSA-AES128-GCM-SHA256 1.3 TLS_AES_256_GCM_SHA384
Router(config-tls)# version min 1.2 max 1.3
Router(config-tls)# key-exchange-groups P-256:P-384:X25519
Router(config-tls)# signature-algorithms rsa_pss_rsae_sha256:ecdsa_secp256r1_sha256
Router(config-tls)# !
Router(config-certificate-authentication-polic)# peer-trust-anchor trustpoint1
Router(config-certificate-authentication-polic)# identity-certificate trustpoint2
Router(config-certificate-authentication-polic)#  extended-key-usage match serverAuth
Router(config-certificate-authentication-polic)# skip-peer-name-validation
Router(config-certificate-authentication-polic)# !
```

Step 5 View running configuration.

Example:

```
Router(config-security-template)# sh run security-template template1

security-template template1
 cc-mode
 certificate-authentication-policy
   tls
    ciphers 1.2 ECDHE-RSA-AES128-GCM-SHA256 1.3 TLS_AES_256_GCM_SHA384
    version min 1.2 max 1.3
    key-exchange-groups P-256:P-384:X25519
    signature-algorithms rsa_pss_rsae_sha256:ecdsa_secp256r1_sha256
  !
  peer-trust-anchor trustpoint1
  identity-certificate trustpoint2
  extended-key-usage match serverAuth clientAuth
  skip-peer-name-validation
 !
!
```

Step 6 Apply the configured security template to desired applications (e.g., syslog, HTTP client) by referencing the template name in their configuration.

Example:

```
Router(config)# logging tls-server TEST
Router(config-logging-tls-peer)# severity debugging
Router(config-logging-tls-peer)# trustpoint tp
Router(config-logging-tls-peer)# address ipv4 10.105.230.83
Router(config-logging-tls-peer)# security-template template1
```

Note When both a standalone trustpoint and a security template are configured under syslog server, the trustpoint specified within the security template overrides the standalone trustpoint.

Step 7 View running configuration.

Example:

```
Router(config-security-template)# sh run logging tls-server TEST

logging tls-server TEST
 severity debugging
 address ipv4 10.105.230.83
 security-template template1
!
```

Step 8 Register the application with the security template infrastructure to receive change notifications and apply updates dynamically.

The security template enforces unified security policies and TLS parameters across all referenced applications. Applications receive and respond to configuration changes automatically, maintaining compliance and reducing risk.

What to do next

- Monitor application logs and debug output for correct application of the security template.

- For certificate rotation or policy updates, update the security template as needed; registered applications will receive notifications and can adapt dynamically.

TLS RFC 5289 compliance for security template

TLS RFC 5289 compliance is a security feature that:

- supports Common Criteria (CC) mode,
- specifies new cipher suites, and
- provides stronger Elliptic Curve Cryptography (ECC) algorithms.

Table 3: Feature History Table

Feature Name	Release Information	Feature Description
TLS RFC 5289 compliance for security template framework	Release 25.4.1	The security template framework is based on RFC 5289, which specifies new cipher suites for the Transport Layer Security (TLS) protocol. This feature supports Common Criteria (CC) mode which is an enhanced security mode that enforces stricter compliance-focused behavior. It enhances TLS security by introducing stronger Elliptic Curve Cryptography (ECC) algorithms.

The security template framework for TLS enabled applications uses RFC 5289 compliance. The RFC moves away from HMAC-SHA1, utilizing SHA-256 and SHA-384 for message authentication codes. For more information on security template framework, see Security template framework for TLS applications.