FIPS

- Federal Information Processing Standard (FIPS), on page 1
- TLS signature algorithm enhancements for NDcPP v3.0e , on page 6
- Configure syslog TLS profile ciphersuites (CLI), on page 10
- Configure HTTP secure ciphersuites, on page 11
- Verify Syslog TLS profile, on page 11
- Verify HTTP client and server applications, on page 11

Federal Information Processing Standard (FIPS)

A Federal Information Processing Standard (FIPS) is a security standard that

- defines requirements for cryptographic modules intended to secure sensitive but unclassified (SBU) information
- is mandated for use by U.S. government agencies and adopted by many regulated industries such as finance and healthcare, and
- establishes assurance levels to validate the strength and reliability of cryptographic implementations.

Sensitive but unclassified (SBU) information: Data that, while not classified for national security, still requires protection due to its sensitive nature and potential impact if disclosed.

Additional reference information

Federal Information Processing Standard (FIPS) 140-2 is a well-known FIPS standard specifying security requirements for cryptographic modules protecting SBU information. These cryptographic modules are produced by the private sector for use in government and regulated environments.

With FIPS in the enabled state, some passwords and pre-shared keys must have the following minimum lengths:

- For Software-Defined Access Wireless, between the controller and map server, a pre-shared key (such as the LISP authentication key) used for authenticating TCP messages must be at least 14 characters long.
- The ISAKMP key (for example, the Crypto ISAKMP key) must also be at least 14 characters long.

Guidelines and restrictions for FIPS

- Controller switches use a legacy key to support legacy APs. In FIPS mode, the crypto engine identifies the key as weak and displays this error message: **% Error in generating keys: could not generate test signature.** Ignore error messages during controller startup in FIPS mode.
- When you enable FIPS, SSH clients that use SHA1 cannot access the controller.
- When you configure a WLAN, set the PSK to at least 15 characters. Otherwise, APs cannot join the controller after you change tags.
- You need to use FIPS-compliant SSH clients to access the controller.
- You cannot use TrustSec in FIPS mode.
- You cannot configure PAC keys in FIPS mode.
- You cannot use level-6 encrypted passwords in FIPS mode. Also, 802.1X authentications fail if the RADIUS shared secret uses a type-6 encryption key.
- The console of APs get disabled when the controller is operating in FIPS mode.
- The weak or legacy cipher like SHA1 is not supported in FIPS mode.
- APs would not reload immediately, if you change the FIPS status.
- We recommend a minimum RSA key size of 2048 bits under RADSEC when operating in FIPS mode. Otherwise, the RADSEC fails.

FIPS self-tests

A FIPS self-test is a cryptographic module validation mechanism that

- verifies functionality and integrity during device power-up
- performs conditional checks whenever security functions are invoked, and
- ensures FIPS 140-2 compliance by enforcing automated self-testing procedures.

Power-up self-tests

Power-up self-tests run automatically after the device powers up. A device enters FIPS mode only after all self-tests are successfully completed. If any self-test fails, the device logs a system message and transitions into an error state. If the power-up self-test fails, the device also fails to boot.

The module uses a known-answer test (KAT), in which a cryptographic algorithm runs on data for which the correct output is already known, and then the calculated output is compared to this expected value. If the calculated output does not equal the known answer, the known-answer test fails.

Power-up self-tests include:

- Software integrity: Validates the integrity of module software as it loads.
- Algorithm tests: Verifies correct operation of cryptographic algorithms using KATs.

Conditional self-tests

Conditional self-tests are performed each time an applicable security function or operation is accessed, unlike power-up self-tests that run only at startup.

The device uses a cryptographic algorithm known-answer test (KAT) on each FIPS 140-2-approved function, which includes encryption, decryption, authentication, and random number generation. The algorithm is applied to a set of data with a known correct output, and the calculated output is compared to this known value. If the calculated output is different, the KAT fails.

Conditional self-tests run automatically whenever their associated security function or operation is invoked. They include the following:

- Pair-wise consistency test: Runs when a public or private key-pair is generated to confirm the correct relationship.
- Continuous random number generator test: Runs each time a random number is generated to check the quality of randomness.
- Bypass: Evaluates bypass operations if performed.
- Software load: Validates integrity when new software is loaded.

Examples

- When a device boots, it performs power-up self-tests to confirm its cryptographic algorithms work as intended before entering FIPS mode.
- Whenever the device generates a new cryptographic key pair, a pair-wise consistency test is executed.
- If a random number is generated, the system runs a continuous random number generator test, verifying the output for compliance.

Configure FIPS (CLI)

Enable or disable FIPS mode on controllers for compliance and security.

Ensure that both the active and standby controllers have the same FIPS authorization key.

Procedure

Step 1 Enter global configuration mode.

Example:

```
Device
# configure terminal
```

Step 2 Enable FIPS mode.

Example:

```
Device
(config)#
fips authorization-key key
```

The key length must be 32 hexadecimal characters.

Note
When FIPS is enabled, you may need to trigger more than one factory reset using the reset button.

To disable FIPS mode on the device, use the **no** form of this command.

Step 3 Return to privileged EXEC mode.

Example:

```
Device
(config)#
end
```

Enable or disable FIPS mode. Reboot the controller to apply changes. After the controller reboots, access points also reboot once they rejoin.

What to do next

Reboot the controller each time you enable or disable FIPS mode.

How FIPS configuration works in high availability setups

Set up a high availability (HA) pair in FIPS mode by configuring both controllers with the same FIPS authorization key before forming the HA pair.

If you configure the key after forming the HA pair, it will not sync to the standby. This may cause reload loops when you reboot.

- Break the HA pair if you configured the FIPS key after pairing.
- Configure the same FIPS authorization key on both members independently.
- Form the HA pair again.

Summary

In this process, you configure FIPS in an HA setup by setting the same FIPS authorization key on both members before forming the HA pair.

- You must perform the configuration steps on both controllers.
- Configure and pair both controllers (member1 and member2) in HA mode.

Matching FIPS authorization keys on both controllers in the HA pair prevent synchronization issues and reload loops.

Workflow

These stages describe the steps to configure FIPS in an HA setup.

1. Turn off both members of the stack.
2. Power on only member1 and wait for the controller to start. Log in from the console when prompted.
3. Log in to member1 with valid credentials and use the following commands to verify FIPS status and configuration:
   - Use the **show fips status** command
   - Use the **show fips authorization-key** command
   - Use the **show romvar** command
   - Use the **show chassis** command

Note Keep the configured FIPS authorization key available.

4. Configure the FIPS key on member1 if you have not already configured it.
   - conf t
   - fips authorization-key <thirty-two hexadecimal characters>
5. Save your changes and power off member1.
6. Power on only member2 and wait for the controller to start. Log in from the console when prompted.
7. Log in to member2 with valid credentials and use the following commands to verify FIPS status and configuration:
   - Use the **show fips status** command
   - Use the **show fips authorization-key** command
   - Use the **show romvar** command
   - Use the **show chassis** command

Note Keep the configured FIPS authorization key available.

8. Configure the FIPS key on member2 if you have not already configured it.

Note The key value must be the same on both members of the stack.

   - conf t
   - fips authorization-key <thirty-two hexadecimal characters>

9. Save your changes and power off member2.
10. Power on both members together and wait for the stack to form.
11. Monitor the system for any crash or unexpected reload events.

Note The members are not expected to reload because of FIPS issues.

Result

Both controllers in the HA pair operate in synchronized FIPS mode. Correct configuration prevents synchronization issues and reload loops caused by mismatched or improperly applied FIPS authorization keys.

Verify FIPS configuration

You can verify FIPS configuration using these commands:

Use this **show** command to display the installed authorization key:

```
Device# show fips authorization-key
FIPS: Stored key (16) : 12345678901234567890123456789012
```

Use this **show** command to display the status of FIPS on the device:

```
Device# show fips status
Chassis is running in fips mode
```

TLS signature algorithm enhancements for NDcPP v3.0e

The TLS signature algorithm enhancements in FIPS Common Criteria (CC) mode is a security posture that

- ensures compliance with the NDcPP v3.0e standard
- removes outdated and less secure signature algorithms and ciphersuites, and
- protects against known vulnerabilities.

Feature history

Table 1: Feature history for TLS signature algorithm enhancements for NDcPP v3.0e

Feature Name	Release Information	Feature Description
TLS signature algorithm enhancements for NDcPP v3.0e	Cisco IOS XE 17.18.2	This feature ensures that Cisco products operating in FIPS Common Criteria (CC) mode are fully compliant with the NDcPP v3.0e security standard. It achieves this by: • Removing outdated and unapproved algorithms • Eliminating weaker algorithms thereby ensuring robust cryptographic protection for sensitive communications. • Enabling both TLS clients (listing supported signature algorithms in the Client Hello message) and TLS/DTLS servers (listing signature algorithms in the Certificate Request message) to operate within the strict requirements of NDcPP v3.0e.

Common criteria certifications using NDcPP v3.0e mandate that TLS Signature Algorithms that are outdated or unapproved are removed from being offered in TLS 1.2 and TLS 1.3, DTLS 1.2. For TLS clients, the signature algorithms are listed in the **Client Hello** message whereas for TLS/DTLS server, the signature algorithms are listed in the **Certificate Request** message.

As security standards evolve and threats become more sophisticated, it is crucial to update and remove outdated and less secure signature algorithms and ciphersuites. This enhancement prevents potential non-compliance and security risks, ensuring robust and compliant network operations. This requirement is vital for maintaining secure communications in environments adhering to the NDcPP v3.0e standard.

Signature algorithms and ciphersuites in TLS and DTLS applications

Allowed signature algorithms

In FIPS Common Criteria mode, you can use the following signature algorithms with TLS and DTLS:

- ecdsa_secp256r1_sha256 (0x0403)
- ecdsa_secp384r1_sha384 (0x0503)
- ecdsa_secp521r1_sha512 (0x0603)
- rsa_pss_pss_sha256 (0x0809)
- rsa_pss_pss_sha384 (0x080a)
- rsa_pss_pss_sha512 (0x080b)
- rsa_pss_rsae_sha256 (0x0804)
- rsa_pss_rsae_sha384 (0x0805)
- rsa_pss_rsae_sha512 (0x0806)

- rsa_pkcs1_sha256 (0x0401)
- rsa_pkcs1_sha384 (0x0501)
- rsa_pkcs1_sha512 (0x0601)

Note No signature algorithms other than those listed are allowed in CC mode.

These sections provide a brief description of the allowed and deprecated signature algorithms and ciphersuites for EST TLS client, RadSec, Syslog, and HTTP Secure (TLS) Server in IOS-XE under FIPS Common Criteria mode:

EST TLS client

Allowed signature algorithms: The EST TLS client offers only allowed signature Algorithms in CC mode, where the IOS-XE device has a TLS client connection to an EST capable CA-server.

Ciphersuites: This is the list of plain RSA ciphers supported from the HTTP side. As part of FIPS Common Criteria mode these ciphersuites are not offered from default client 'Hello' message.

- AES128-SHA
- AES256-SHA
- AES128-SHA256
- AES256-SHA256
- AES128-GCM-SHA256
- AES256-GCM-SHA384

Note
- Because the changes are in HTTP client infrastructure, all HTTP client applications in FIPS Common Criteria mode are affected.
- If the peer supports a wide range of ciphersuites, applications are not affected.
- If the peer supports only the listed ciphers, the TLS handshake fails in FIPS+CC mode.

RadSec

For RadSec TLS and DTLS, these are the ciphers that are supported and deprecated with FIPS common criteria mode.

Supported cipher list when FIPS + CC mode is enabled:

- **DTLS (IOS and BINOS)**
  - ECDHE-ECDSA-AES128-GCM-SHA256 and ECDHE-ECDSA-AES256-GCM-SHA384
  - ECDHE-ECDSA-AES128-SHA256 and ECDHE-ECDSA-AES256-SHA384
  - ECDHE-RSA-AES128-GCM-SHA256 and ECDHE-RSA-AES256-GCM-SHA384

- ECDHE-RSA-AES128-SHA256 and ECDHE-RSA-AES256-SHA384

- **TLS (BINOS)**
  - ECDHE-ECDSA-AES128-GCM-SHA256 and ECDHE-ECDSA-AES256-GCM-SHA384
  - ECDHE-RSA-AES128-GCM-SHA256 and ECDHE-RSA-AES256-GCM-SHA384

- **TLS (IOS)**
  - ECDHE-ECDSA-AES128-GCM-SHA256
  - ECDHE-RSA-AES128-GCM-SHA256

Deprecated Cipher list when FIPS + CC mode is enabled:

- **DTLS (IOS and BINOS)**:
  - AES128-SHA256
  - AES256-SHA256

- **TLS (BINOS)**:
  - RC4-MD5
  - NULL-SHA
  - AES128-SHA

- **TLS (IOS)**:
  - AES128-SHA

Note

- When FIPS mode is not enabled, the current existing set of ciphers is supported, which is a combination of both strong and weak ciphers.
- RadSec TLS connections allow only TLS signature algorithms listed in the 'Allowed signature algorithms' section.

Syslog

These are the list of ciphersuites from the Syslog component that are deprecated for FIPS Common Criteria mode.

- AES128-SHA
- AES256-SHA
- AES128-SHA256
- AES256-SHA256
- AES128-GCM-SHA256

- AES256-GCM-SHA384

Note These ciphersuites are excluded in the Syslog default ciphersuite and can be configured through commands.

HTTP secure (TLS) server

Signature algorithm:

When client authentication is enabled on the IOS-XE device, the HTTP secure (TLS) server includes the list of signature algorithms in the Certificate Request Message.

Ciphersuites:

The HTTP side supports these plain RSA ciphers. In FIPS Common Criteria mode, these ciphersuites are not included in the default client 'Hello'.

- AES128-SHA
- AES256-SHA
- AES128-SHA256
- AES256-SHA256
- AES128-GCM-SHA256
- AES256-GCM-SHA384

Configure syslog TLS profile ciphersuites (CLI)

To specify the allowed ciphersuites for secure Syslog communication in FIPS Common Criteria mode.

In FIPS Common Criteria mode, certain weaker ciphersuites are deprecated and removed from the default list for Syslog. You can view the available ciphersuites and configure them.

Procedure

Step 1 Enter global configuration mode.

Example:

```
Device# configure terminal
```

Step 2 Access the TLS profile for syslog. Replace `tls-profile-name` with your profile name.

Example:

```
Device(config)# logging tls-profile tls-profile-name
```

Step 3 Select and configure a ciphersuite.

Example:

```
Device(config-tls-profile)# ciphersuite ecdhe-rsa-aes-cbc-sha2
```

You can view the ciphersuites by running the `Device(config-tls-profile)# ciphersuite ?` command.

The syslog TLS profile is configured with the specified ciphersuite.

Configure HTTP secure ciphersuites

To explicitly configure secure ciphersuites for HTTP client and server communications.

This feature ensures that HTTP client and server communications adhere to enhanced security standards by allowing explicit configuration of secure ciphersuites. There is no change in default ciphersuite behavior for HTTP client and server.

Procedure

Step 1 Enter global configuration mode.

Example:

```
Device# configure terminal
```

Step 2 Configure secure ciphersuites for the HTTP server. You can specify one or more ciphersuites.

Example:

```
Device(config)# ip http secure-ciphersuite ciphersuites
```

Step 3 Configure secure ciphersuites for the HTTP client.

Example:

```
Device(config)# ip http client secure-ciphersuite ciphersuites
```

HTTP client and server communications use the explicitly configured secure ciphersuites.

Verify Syslog TLS profile

To verify the configured Syslog TLS profile, use this command.

```
Device# show running-config | section logging tls-profile syslog_1
logging tls-profile syslog_1
ciphersuite ecdhe-ecdsa-aes-gcm-sha2
```

Verify HTTP client and server applications

To verify the configured HTTP secure ciphersuites for the server, use this command.

```
Device# show running-config | include ip http secure-ciphersuite
ip http secure-ciphersuite ecdhe-ecdsa-aes-gcm-sha2 tls13-aes256-gcm-sha384
```

To verify the configured HTTP secure ciphersuites for the client, use this command.

```
Device# show running-config | include ip http client secure-ciphersuite
ip http client secure-ciphersuite ecdhe-rsa-aes-gcm-sha2 tls13-aes128-gcm-sha256
```

To verify the configured HTTP secure server status information, use this command.

```
Device# show ip http server secure status
HTTP secure server status: Enabled
HTTP secure server port: 443
HTTP secure server ciphersuite:  dhe-aes-cbc-sha2 dhe-aes-gcm-sha2
        ecdhe-rsa-aes-cbc-sha2 ecdhe-rsa-aes-gcm-sha2 ecdhe-ecdsa-aes-gcm-sha2
        tls13-aes128-gcm-sha256 tls13-aes256-gcm-sha384
HTTP secure server TLS version:  TLSv1.2
HTTP secure server client authentication: Disabled
HTTP secure server PIV authentication: Disabled
HTTP secure server PIV authorization only: Disabled
HTTP secure server trustpoint: TP-self-signed-745339156
HTTP secure server peer validation trustpoint:
HTTP secure server ECDHE curve: secp256r1
HTTP secure server active session modules: ALL
```

To verify the configured HTTP secure client status information, use this command.

```
Device# show ip http client secure status
HTTP secure client ciphersuite:  dhe-aes-cbc-sha2 dhe-aes-gcm-sha2
        ecdhe-rsa-aes-cbc-sha2 ecdhe-rsa-aes-gcm-sha2 ecdhe-ecdsa-aes-gcm-sha2
        tls13-aes128-gcm-sha256 tls13-aes256-gcm-sha384
HTTP secure client TLS version:  TLSv1.3 TLSv1.2
HTTP secure client trustpoint:
```