HTTPS Access

This chapter contains the following sections:

Overview

This article provides an example of how to configure a custom certificate for HTTPS access when using Cisco ACI.

Configuring Custom Certificate Guidelines

- Exporting a private key that is used to generate a Certificate Signing Request (CSR) on the Cisco Application Policy Infrastructure Controller (APIC) is not supported. If you want to use the same certificate on multiple servers through a wildcard in the Subject Alternative Name (SAN) field, such as "*cisco.com," by sharing the private key that was used to generate the CSR for the certificate, generate the private key outside of Cisco Application Centric Infrastructure (ACI) fabric and import it to the Cisco ACI fabric.

- You must download and install the public intermediate and root CA certificates before generating a Certificate Signing Request (CSR). Although a root CA Certificate is not technically required to generate a CSR, Cisco requires the root CA certificate before generating the CSR to prevent mismatches between the intended CA authority and the actual one used to sign the CSR. The Cisco APIC verifies that the certificate submitted is signed by the configured CA.

- To use the same public and private keys for a renewed certificate generation, you must satisfy the following guidelines:

    - You must preserve the originating CSR as it contains the public key that pairs with the private key in the key ring.

  - The same CSR used for the originating certificate must be resubmitted for the renewed certificate if you want to re-use the public and private keys on the Cisco APIC.
  - Do not delete the original key ring when using the same public and private keys for the renewed certificate. Deleting the key ring will automatically delete the associated private key used with CSRs.

- Cisco ACI Multi-Site, VCPlugin, VRA, and SCVMM are not supported for certificate-based authentication.
- Only one SSL certificate is allowed per Cisco APIC cluster.
- You must disable certificate-based authentication before downgrading to release 4.0(1) from any later release.
- To terminate the certificate-based authentication session, you must log out and then remove the CAC card.
- The custom certificate configured for the Cisco APIC will be deployed to the leaf and spine switches. If the URL or DN that is used to connect to the fabric node is within the **Subject** or **Subject Alternative Name** field, the fabric node will be covered under the certificate.
- The Cisco APIC GUI can accept a certificate with a maximum size of 4k bytes.
- When a self-signed SSL certificate that you are using for HTTPS access expires, the certificate gets renewed automatically.

Modifying the SSL Cipher Configuration

SSL ciphers can be enabled, disabled, or removed entirely. Depending on the desired cipher settings, you should understand which exact combination is required. Disabling and enabling ciphers in a manner that results in no ciphers remaining is a misconfiguration and will result in NGINX failing validation.

NGINX uses the OpenSSL cipher list format. For information about the format, go to the OpenSSL website.

Mapping the Cisco APIC SSL Configuration Options to the Cipher List Formatting

Enabling a cipher results in the cipher being written to the NGINX configuration file. Disabling a cipher results in the cipher being written in the NGINX configuration file with a preceding exclamation mark (!). For example, disabling "EEDCH" will cause it to be written as "!EEDCH". Removing a cipher will result in the cipher not being written the NGINX configuration file at all.

Note As stated in the OpenSSL cipher list format document, "If ! is used then the ciphers are permanently deleted from the list. The ciphers deleted can never reappear in the list even if they are explicitly stated." This can result in the removal of combination ciphers referencing the one that was set to "Disabled," regardless of the ciphers' "Enabled" state.

Example: Disabling "EEDCH," but enabling "EECDH+aRSA+SHA384." This will cause the following to be written to the NGINX configuration file: "!EEDCH:EECDH+aRSA+SHA384". The "!EEDCH" will prevent "EECDH+aRSA+SHA384" from ever being added. This will result in no ciphers being used, which will fail NGINX validation and prevent NGINX updates from succeeding, such as applying custom HTTPS certificates.

Testing the Cipher List Format Before Modifying the Cisco APIC SSL Configuration

Before making any cipher modifications to the Cisco Application Policy Infrastructure Controller (APIC), validate the results of the planned cipher combination using the `openssl ciphers -V 'cipher_list'` command and ensure that the cipher output matches your desired result.

Example:

```
apic# openssl ciphers -V 'EECDH+aRSA+SHA256:EECDH+aRSA+SHA384'
          0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)
Mac=SHA256
          0xC0,0x28 - ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)
Mac=SHA384
```

If your tested cipher list results in an error or "no cipher match," do not apply this configuration to the Cisco APIC. Doing so can result in NGINX issues with symptoms including making the Cisco APIC GUI inaccessible and breaking custom certificate application.

Example:

```
apic# openssl ciphers -V '!EECDH:EECDH+aRSA+SHA256:EECDH+aRSA+SHA384'
Error in cipher list
132809172158128:error:1410D0B9:SSL routines:SSL_CTX_set_cipher_list:no cipher
match:ssl_lib.c:1383:
```

Configuring a Custom Certificate for Cisco ACI HTTPS Access Using the GUI

Caution PERFORM THIS TASK ONLY DURING A MAINTENANCE WINDOW AS THERE IS A POTENTIAL FOR DOWNTIME.

The downtime affects access to the Cisco Application Policy Infrastructure Controller (APIC) cluster and switches from external users or systems and not the Cisco APIC to switch connectivity. There will be an impact to external connectivity due to the NGINX processes running on the switches, but not the fabric data plane. Access to the Cisco APIC, configuration, management, troubleshooting, and such are impacted. The NGINX web server running on the Cisco APIC and switches restart during this operation.

Before you begin

Determine from which authority that you obtain the trusted certification so that you can create the appropriate Certificate Authority.

Step 1 On the menu bar, click the **Admin** > **AAA**.

Step 2 In the **Navigation** pane, select **Security**.

Step 3 In the **Work** pane, choose **Certificate Authorities** > **Actions** > **Create Certificate Authority**.

Step 4 In the **Create Certificate Authority** screen, in the **Name** field, enter a name for the certificate authority.

Step 5 (Optional) Enter a **Description** for the certificate authority.

Step 6 In the **Certificate Chain** field, copy the intermediate and root certificates for the certificate authority that will sign the Certificate Signing Request (CSR) for the Cisco APIC.

The certificate has to be in Base64 encoded X.509 CER (Cisco Emergency Responder) format. The intermediate certificate is placed before the root CA certificate. It should look similar to the following example:

```
-----BEGIN CERTIFICATE-----
<Intermediate Certificate>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Root CA Certificate>
-----END CERTIFICATE-----
```

Step 7 Click **Save**.

Step 8 In the **Work** pane, choose **Key Rings** > **Actions** > **Create Key Ring**.

The **Key Rings** enables you to manage:

a. Private keys (imported from an external device or internally generated on the Cisco APIC).

b. CSR generated by the private key.

c. Certificate signed through the CSR.

Step 9 In the **Create Key Ring** dialog box, in the **Name** field, enter a name.

Step 10 (Optional) Enter a **Description** for the key ring.

Step 11 In the **Certificate Authority** field, click **Select Certificate Authority** to choose the certificate authority that you created earlier, or click **Create Certificate Authority**.

Step 12 Choose the required radio button for the **Private Key** field.

The options are:

a. Generate New Key.

b. Import Existing Key.

Step 13 Enter a Private Key. This option is displayed only if you chose the **Import Existing Key** option for **Private Key**.

Step 14 Choose the required radio button for **Key Type** if you chose the **Generate New Key** option for the **Private Key** field.

The choices are:

a. **RSA** (Rivest, Shamir, and Adleman).

b. **ECC** (Elliptic-curve cryptography) also known as ECDSA (Elliptic Curve Digital Signature Algorithm).

Step 15 In the **Certificate** field, do not add any content if you want to generate a CSR using the Cisco APIC through the key ring. If you already have the signed certificate content that was signed by the CA from the previous steps by generating a private key and CSR outside of the Cisco APIC, you can add it to the **Certificate** field.

Step 16 Select the required key strength for the cipher. This option is displayed only if you have selected the Generate New Key option in the **Private Key** field. **Modulus** drop-down list for RSA or **ECC Curve** checking the radio buttons for ECC **Key Type**.

a) If you chose **RSA** for the **Key Type**, from the **Modulus** drop-down list, choose a modulus value.
b) If you chose **ECC** for the **Key Type**, from the list of **ECC Curve** radio buttons, choose an appropriate curve.

Step 17 Click **Save** (**Create Key Ring** screen).

Step 18 In the **Work** pane, choose **Key Rings** > ***key_ring_name*** (or you could also double click the required key ring row).

If you have not entered the signed certificate and the private key, in the **Work** pane, in the **Key Rings** area, the **Admin State** for the key ring that is created displays **Started**, waiting for you to generate a CSR. Proceed to step 19.

If you entered both the signed certificate and the private key, in the **Key Rings** area, the **Admin State** for the key ring that is created displays **Completed**. Proceed to step 22.

Note Do not delete the key ring. Deleting the key ring will automatically delete the associated private key that is used with CSRs.

Click the expand button, a new screen with the selected key ring is displayed.

Step 19 In the **Certificate Request** pane, click **Create Certificate Request**.

The **Request Certificate** window is displayed.

a) In the **Subject** field, enter the Common Name (CN) of the CSR.

You can enter the fully qualified domain name (FQDN) of the Cisco APICs using a wildcard, but in a modern certificate, we recommend that you enter an identifiable name of the certificate and enter the FQDN of all Cisco APICs in the **Alternate Subject Name** field (also known as the *SAN* – Subject Alternative Name) because many modern browsers expect the FQDN in the SAN field.

b) In the **Alternate Subject Name** field, enter the FQDN of all Cisco APICs, such as "DNS:apic1.example.com,DNS:apic2.example.com,DNS:apic3.example.com" or "DNS:*example.com".

Alternatively, if you want SAN to match an IP address, enter the Cisco APICs' IP addresses with the following format:

```
IP:192.168.2.1
```

You can use DNS names, IPv4 addresses, or a mixture of both in this field. IPv6 addresses are not supported.

c) In the **Locality** field, enter the city or town of the organization.
d) In the **State** field, enter the state in which the organization is located.
e) In the **Country** field, enter the two-letter ISO code for the country in which the organization is located.
f) Enter the **Organization Name** and a unit in the organization for the **Organization Unit Name**.
g) Enter the **Email** address of the organization's contact person.
h) Enter a **Password** and enter the password again in the **Confirm Password** field.
i) Click **OK**.

Step 20 The Certificate Request Settings pane now displays the information that you entered above (step 19).

Step 21 In the **Work** pane, choose **Key Rings** > ***key_ring_name*** (or you could also double click the required key ring row).

A new screen with the selected **Key Rings** is displayed with the Certificate details.

Note CSR which is not signed by a certificate authority that is indicated in the key ring or has MS-DOS line endings is not accepted. An error message is displayed, remove the MS-DOS line endings to resolve it.

After the key is verified sucessfully, in the **Work** pane, the **Admin State** changes to **Completed** and is now ready for use in the HTTP policy.

Step 22 On the menu bar, select **Fabric** > **Fabric Policies**.

Step 23 In the Navigation pane, click **Policies** > **Pod** > **Management Access** > **default**.

Step 24 In the **Work** pane, in the **Admin Key Ring** drop-down list, choose the desired key ring.

Step 25 (Optional) For Certificate based authentication, in the **Client Certificate TP** drop-down list, choose the previously created Local User policy and click **Enabled** for **Client Certificate Authentication state**.

Step 26 Click **Submit**.

All web servers restarts, activating the certificate, and the nondefault key ring is associated with the HTTPS access.

What to do next

Be wary of the expiration date of the certificate and take the required action before it expires. To retain the same key pair for the renewed certificate, preserve the CSR. CSR contains the public key that pairs with the private key in the key ring. Resubmit the same CSR, before the certificate expires. Do not delete or create a new key ring. Deleting the key ring deletes the private key that is stored in the Cisco APIC.

Configuring the Default SSL Protocols and Diffie-Hellman Key Exchange Using the GUI

This procedure configures the default SSL protocols and Diffie-Hellman key exchange. You must configure these parameters based on the security policy of your organization and the needs of any applications that you use.

Step 1 On the menu bar, choose **Fabric** > **Fabric Policies**.

Step 2 In the **Navigation** pane, choose **Policies** > **Pod** > **Management Access** > **default**.

Step 3 In the **Work** pane, find the **HTTPS** section.

a) For **SSL Protocols**, put a check in the boxes for the transport layer security (TLS) versions that your network allows. Leave the box empty for any TLS version that your network does not allow.

b) In the 6.0(1) release, for **DH Param**, choose the desired key size (in bits).

   Choosing one of the key sizes enables the standard Diffie-Hellman (DH) key exchange in addition to the elliptic-curve Diffie-Hellman (ECDH) key exchange and uses the chosen number of bits for the DH key exchange. Choosing **None** instead uses only the elliptic-curve ECDH key exchange. In any case, ECDH always uses 256 bits.

   Beginning with the 6.0(2) release, the DH parameters are dynamically determined during the communication handshake with the client. You no longer manually choose the key size.

c) Click **Submit**.

Enabling Certificate Based Authentication Using the NX-OS CLI

To enable Certificate Based authentication:

Example:

```
To enable CAC for https access:
  configure terminal
   comm-policy default
    https
     client-cert-ca <ca name>
     client-cert-state-enable
 To disable:
```

```
configure terminal
 comm-policy default
  https
   no client-cert-state-enable
   no client-cert-ca
```

About SSL Ciphers

The Cisco Application Centric Infrastructure (ACI) Representational State Transfer (REST) Application Programming Interface (API) has gone through an evolution from the day the solution debuted to recent versions where the HTTPS/SSL/TLS support has gotten increasingly more stringent. This document is intended to cover the evolution of HTTPS, SSL, and TLS support on the Cisco ACI REST API and provide customers with a guide of what is required for a client to utilize the REST API securely.

HTTPS is a protocol that utilizes either Secure Socket Layers (SSL) or Transport Layer Security (TLS) to form a secure connection for a HTTP session. SSL or TLS is used to encrypt the traffic between a client and a HTTP server. In addition, servers that support HTTPS have a certificate that can usually be used by the client to verify the server's authenticity. This is the opposite of the client authenticating with the server. In this case, the server is saying, "I am server_xyz and here is the certificate that proves it." The client can then utilize that certificate to verify the server is "server_xyz."

There are other important aspects to SSL/TLS that involve the supported encryption ciphers available in each protocol as well as the inherent security of the SSL or TLS protocols. SSL has gone through three iterations - SSLv1, SSLv2 and SSLv3 - all of which are now considered insecure. TLS has gone through three iterations - TLSv1, TLSv1.1 and TLSv1.2 - of which only TLSv1.1 and TLSv1.2 are considered "secure." Ideally, a client should utilize the highest available TLS version it can and the server should support only TLSv1.1 and TLSv1.2. However, most servers must keep TLSv1 for outdated clients.

Almost all modern browsers support both TLSv1.1 and TLSv1.2. However, a client that utilizes HTTPS may not be a browser. The client may be a java application or a python script that communicates with a web server and must negotiate HTTPS/TLS. In this type of a situation, the questions of what is supported and where becomes much more important.

Determining the Supported SSL Ciphers Using the CLI

Before you begin

This section describes how to use the CLI to determine which SSL ciphers are supported.

Step 1 Get the supported ciphers in your OpenSSL environment, which is shown as follows:

Example:

```
openssl ciphers 'ALL:eNULL'
```

Step 2 Separate the ciphers using sed or some other tool, which is shown as follows:

Example:

```
openssl ciphers 'ALL:eNULL' |  sed -e 's/:/\n/g'
```

Step 3 Loop over the ciphers and poll the APIC to see which ones are supported, shown as follows:

Example:

```
openssl s_client -cipher '<some cipher to test>' -connect <apic ipaddress>:<ssl port, usually 443>
```

See the following example cipher:

Example:

```
openssl s_client -cipher 'ECDHE-ECDSA-AES128-GCM-SHA256' -connect 10.1.1.14:443
```

Note If the response contains `CONNECTED`, then the cipher is supported.