Configuring Security

About Security Configuration

Access control is the way you control who is allowed access to the network server and what services they are allowed to use once they have access. Authentication, authorization, and accounting (AAA) network security services provide the primary framework through which you set up access control on APIC.

Overview of the AAA Configuration

To configure security on APIC using AAA, follow this process:

1. To use a separate security server, configure security protocol parameters using the **radius-server** , **ldap-server** , or **tacacs-server** configuration commands.
2. Define the method lists for authentication by using an **aaa authentication** command.
3. Apply the method lists to a particular interface or line, if required.
4. (Optional) Configure authorization using the **aaa authentication** command.

Login Authentication Using a Local Password

Use the **aaa authentication login** command with the *method* argument to specify that APIC will use the local username database for authentication. For example, to specify the local username database as the method of user authentication at login when no other method list has been defined, enter the following commands:

```
apic1# configure
apic1(config)# aaa authentication login default
apic1(config-default)# realm local
```

For information about adding users into the local username database, refer to the section “Configuring a Locally Authenticated User.”

Login Authentication Using a Remote Server

Use the aaa authentication login command with the server radius/tacacs/ldap method to specify RADIUS/TACACS+/LDAP as the login authentication method. For example, to specify RADIUS as the method of user authentication at login when no other method list has been defined, enter the following commands:

```
apic1# configure
apic1(config)# aaa authentication login default
apic1(config-default)# realm radius
```

Before you can use RADIUS as the login authentication method, you need to enable communication with the RADIUS security server, same is true for TACACS+ or LDAP. For more information about establishing communication with a remote security server, see the appropriate chapter:

- "Configuring a RADIUS Server"
- "Configuring a TACACS+ Server"
- "Configuring an LDAP Server"

Configuring AAA

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**aaa authentication login console** **Example:** `apic1(config)# aaa authentication login console`	Enters console configuration mode for users accessing APIC through the console.

	Command or Action	Purpose

Step 3	[**no**] **realm** {**ldap** | **local** | **radius** | **tacacs**}
	Example:
	`apic1(config-console)# realm radius`
		Specifies the authentication method.

Step 4	[**no**] **group** *group-name* **Example:** `apic1(config-console)# group radiusGroup5`	Specifies an authentication server group.
Step 5	**exit** **Example:** `apic1(config-console)# exit`	Returns to global configuration mode.
Step 6	**aaa authentication login default** **Example:** `apic1(config)# aaa authentication login default`	Enters the configuration mode for default login authentication.

Step 7	[**no**] **realm** {**ldap** | **local** | **radius** | **tacacs**}
	Example:
	`apic1(config-default)# realm radius`
		Specifies the authentication method.

Step 8	[**no**] **group** *group-name* **Example:** `apic1(config-default)# group radiusGroup`	Specifies an authentication server group.
Step 9	**exit** **Example:** `apic1(config-default)# exit`	Returns to global configuration mode.
Step 10	**aaa authentication login domain** {*domain-name* | **fallback**} **Example:** `apic1(config)# aaa authentication login domain cisco`	Enters the configuration mode for default login authentication. A login domain specifies the authentication domain for a user.

Step 11	[**no**] **realm** {**ldap** | **local** | **none** | **radius** | **tacacs**}
	Example:
	`apic1(config-domain)# realm radius`
		Specifies the authentication method.

Step 12	[**no**] **group** *group-name* **Example:** `apic1(config-domain)# group radiusGroup`	Specifies an authentication server group.

	Command or Action	Purpose
Step 13	**exit** **Example:** `apic1(config-domain)# exit`	Returns to global configuration mode.
Step 14	**aaa banner** *text* **Example:** `apic1(config)# aaa banner 'Welcome to APIC'`	Specifies the informational banner to be displayed before the user login. The banner must be contained in single quotes.
Step 15	**aaa group** {**ldap** | **radius** | **tacacs**} *group-name* **Example:** `apic1(config)# aaa group radius radiusGroup`	Creates or configures an authentication server group.
Step 16	[**no**] **server** {*ip-address* | *hostname*} **priority** *priority-number* **Example:** `apic1(config-radius)# server 192.0.20.71 priority 2`	Adds a server to the authentication server group and specifies its priority within the server group. The priority can be between 0 and 17.
Step 17	**exit** **Example:** `apic1(config-radius)# exit`	Returns to global configuration mode.
Step 18	**aaa scvmm-certificate** *certificate-name* **Example:** `apic1(config)# aaa scvmm-certificate myScvmmCert`	Specifies an SCVMM certificate. See the *Cisco ACI Virtualization Guide*.

Step 19	**aaa user default-role** {**assign-default-role** | **no-login**}
	Example:
	`apic1(config)# aaa user default-role assign-default-role`
		Specifies how to respond when remote users who do not have a user role attempt to log in to APIC. The action can be either of these options:
		• **assign-default-role** —Remote users who do not have a user role are assigned a default role.
		• **no-login** —Remote users who do not have a user role cannot log in.

Step 20	**show aaa authentication** **Example:** `apic1(config)# show aaa authentication`	Displays configured AAA methods.

	Command or Action	Purpose
Step 21	**show aaa groups** **Example:** `apic1(config)# show aaa groups`	Displays configured AAA server groups.

Examples

This example shows how to configure AAA.

```
apic1# configure terminal
apic1(config)# aaa authentication login console
apic1(config-console)# realm local
apic1(config-console)# exit
apic1(config)# aaa authentication login default
apic1(config-default)# realm radius
apic1(config-default)# group radiusGroup5
apic1(config-default)# exit
apic1(config)# aaa authentication login domain cisco
apic1(config-domain)# realm none
apic1(config-domain)# exit
apic1(config)# aaa banner 'Welcome to APIC'
apic1(config)# aaa group radius radiusGroup
apic1(config-radius)# server 192.0.20.71 priority 2
apic1(config-radius)# exit
apic1(config)# aaa user default-role assign-default-role
apic1(config)# show aaa authentication
Default : radius
Console : local

apic1(config)# show aaa groups
Total number of Groups : 1

RadiusGroups : radiusGroup5
TacacsGroups :
LdapGroups   :
```

Configuring Security Servers

Configuring a RADIUS Server

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[**no**] **radius-server retries** *count* **Example:** `apic1(config)# radius-server retries 1`	Specifies how many times APIC transmits each RADIUS request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all RADIUS servers unless overridden in the specific RADIUS host configuration.
Step 3	[**no**] **radius-server timeout** *seconds* **Example:** `apic1(config)# radius-server timeout 5`	Specifies the number of seconds APIC waits for a reply to a RADIUS request before retransmitting the request. In the global configuration mode, this command applies to all RADIUS servers unless overridden in the specific RADIUS host configuration.

Step 4	[**no**] **radius-server host** {*ip-address* | *hostname*}
	Example:
	`apic1(config)# radius-server host 192.0.20.71`
		Specifies the IP address or hostname of the RADIUS server.

Step 5	(Optional) [**no**] **retries** *count* **Example:** `apic1(config-host)# retries 2`	For this RADIUS server, specifies how many times APIC transmits each RADIUS request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.
Step 6	(Optional) [**no**] **timeout** *seconds* **Example:** `apic1(config-host)# timeout 3`	For this RADIUS server, specifies the number of seconds APIC waits for a reply to a RADIUS request before retransmitting the request. If no timeout is set, the global value is used.
Step 7	(Optional) [**no**] **descr** *text* **Example:** `apic1(config-host)# descr "My primary RADIUS server"`	Provides descriptive information about this RADIUS server. The text can be up to 128 alphanumeric characters. If the text contains spaces, it must be enclosed by single or double quotes.
Step 8	[**no**] **key** *key-value* **Example:** `apic1(config-host)# key myRaDiUSpassWoRd`	Specifies the shared secret text string used between APIC and this RADIUS server for authentication. The key can be up to 32 characters.
Step 9	[**no**] **port** *port-number* **Example:**	Specifies a UDP port on this RADIUS server to be used solely for authentication.

	Command or Action	Purpose
	`apic1(config-host)# port 1812`	

Step 10	[**no**] **protocol** {**chap** | **mschap** | **pap**}
	Example:
	`apic1(config-host)# protocol pap`
		Specifies the RADIUS server protocol for authentication.

Step 11	**exit** **Example:** `apic1(config-host)#`	Returns to global configuration mode.
Step 12	**show radius-server** **Example:** `apic1(config)# show radius-server`	(Optional) Displays the RADIUS server information.

Examples

This example shows how to configure RADIUS settings globally and on one RADIUS server.

```
apic1# configure
apic1(config)# radius-server retries 1
apic1(config)# radius-server timeout 5
apic1(config)# radius-server host 192.0.20.71
apic1(config-host)# retries 2
apic1(config-host)# timeout 3
apic1(config-host)# descr "My primary RADIUS server"
apic1(config-host)# key myRaDiUSpassWoRd
apic1(config-host)# port 1812
apic1(config-host)# protocol pap
apic1(config-host)# exit
apic1(config)# show radius-server
timeout : 5
retries : 1

Total number of servers : 1

Hostname : 192.0.20.71
Port     : 1812
Protocol : pap
Timeout  : 3
Retries  : 2
User     : test
Descr    : My primary RADIUS server
```

Configuring a TACACS+ Server

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	[**no**] **tacacs-server retries** *count* **Example:** `apic1(config)# tacacs-server retries 1`	Specifies how many times APIC transmits each TACACS+ request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all TACACS+ servers unless overridden in the specific TACACS+ host configuration.
Step 3	[**no**] **tacacs-server timeout** *seconds* **Example:** `apic1(config)# tacacs-server timeout 5`	Specifies the number of seconds APIC waits for a reply to a TACACS+ request before retransmitting the request. In the global configuration mode, this command applies to all TACACS+ servers unless overridden in the specific TACACS+ host configuration.

Step 4	[**no**] **tacacs-server host** {*ip-address* | *hostname*}
	Example:
	`apic1(config)# tacacs-server host 192.0.20.71`
		Specifies the IP address or hostname of the TACACS+ server.

Step 5	(Optional) [**no**] **retries** *count* **Example:** `apic1(config-host)# retries 2`	For this TACACS+ server, specifies how many times APIC transmits each TACACS+ request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.

Step 6	[**no**] **key**
	Example:
	`apic1(config-host)# key`
	`Enter key: myTacAcSpassWoRd`
	`Enter key again: myTacAcSpassWoRd`
		Specifies the shared secret text string used between APIC and this TACACS+ server for authentication. The key can be up to 32 characters. For increased security, entering the key value is interactive.

Step 7	[**no**] **port** *port-number* **Example:** `apic1(config-host)# port 49`	Specifies a UDP port on this TACACS+ server to be used for TACACS+ accounting messages.

	Command or Action	Purpose

Step 8	[**no**] **protocol** {**chap** | **mschap** | **pap**}
	Example:
	`apic1(config-host)# protocol pap`
		Specifies the TACACS+ server protocol for authentication.

Step 9	**exit** **Example:** `apic1(config-host)#`	Returns to global configuration mode.
Step 10	**show tacacs-server** **Example:** `apic1(config)# show tacacs-server`	(Optional) Displays the TACACS+ server information.

Examples

This example shows how to configure TACACS+ settings globally and on one TACACS+ server.

```
apic1# configure
apic1(config)# tacacs-server retries 1
apic1(config)# tacacs-server timeout 5
apic1(config)# tacacs-server host 192.0.20.72
apic1(config-host)# retries 2
apic1(config-host)# timeout 3
apic1(config-host)# key myTaCaCspassWoRd
apic1(config-host)# port 49
apic1(config-host)# protocol pap
apic1(config-host)# exit
apic1(config)# show tacacs-server
timeout : 5
retries : 1

Total number of servers : 1

Hostname : 192.0.20.72
Port     : 1812
Protocol : pap
Timeout  : 3
Retries  : 2
User     : test
```

Configuring an LDAP Server

Some **ldap-server** commands can be entered in either the global configuration mode or in the configuration mode for a specific LDAP host. In the global configuration mode, the command applies to all LDAP servers unless overridden in the specific LDAP host configuration.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.

Step 2	[**no**] **ldap-server host** {*ip-address* | *hostname*}
	Example:
	`apic1(config)# ldap-server host 192.0.20.73`
		Specifies the IP address or hostname of the LDAP server and enters the configuration mode of that server.

Step 3	[**no**] **ldap-server attribute** *attribute-name* **Example:** `apic1(config-host)# ldap-server attribute memberOf`	Specifies an LDAP endpoint attribute to be used as the CiscoAVPair. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 4	[**no**] **ldap-server basedn** **Example:** `apic1(config-host)# ldap-server basedn DC=sampledesign,DC=com`	Specifies the location in the LDAP hierarchy where the server should begin searching when it receives an authorization request. This can be a string of up to 127 characters. Spaces are not permitted in the string, but other special characters are allowed. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.

Step 5	[**no**] **ldap-server binddn**
	Example:
	`apic1(config-host)# ldap-server binddn`
	`CN=ucsbind,OU=CiscoUsers,DC=sampledesign,DC=com`
		Specifies the distinguished name (DN) for an LDAP database account that has read and search permissions for all objects under the base DN. This can be a string of up to 127 characters. Spaces are not permitted in the string, but other special characters are allowed.

Step 6	[**no**] **ldap-server retries** *count* **Example:** `apic1(config-host)# ldap-server retries 1`	Specifies how many times APIC transmits each LDAP request to the server before giving up. The range is 0 to 5. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 7	[**no**] **ldap-server timeout** *seconds* **Example:** `apic1(config-host)# ldap-server timeout 30`	Specifies the number of seconds APIC waits for a reply to a LDAP request before retransmitting the request.

	Command or Action	Purpose
		In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.
Step 8	[**no**] **ldap-server filter** *filter-expression* **Example:** `apic1(config-host)# ldap-server filter sAMAccountName=$userid`	Specifies a filter to filter the results of LDAP searches. The filter can contain a maximum of 63 characters. In the global configuration mode, this command applies to all LDAP servers unless overridden in the specific LDAP host configuration.

Step 9	[**no**] **key** *key-value*
	Example:
	`apic1(config-host)# key`
	`Enter key: myLdAppassWoRd`
	`Enter key again: myLdAppassWoRd`
		Specifies the shared secret text string used between APIC and this LDAP server for authentication. The key can be up to 32 characters.

Step 10	[**no**] **port** *port-number* **Example:** `apic1(config-host)# port 389`	Specifies the LDAP server port for authentication.
Step 11	(Optional) [**no**] **retries** *count* **Example:** `apic1(config-host)# retries 2`	For this LDAP server, specifies how many times APIC transmits each LDAP request to the server before giving up. The range is 0 to 5. If no retry count is set, the global value is used.
Step 12	[**no**] **enable-ssl** **Example:** `apic1(config-host)# enable-ssl`	Enables an SSL connection with the LDAP provider.
Step 13	[**no**] **ssl-validation-level** [**permissive** | **strict**] **Example:** `apic1(config-host)# ssl-validation-level permissive`	Sets the LDAP Server SSL Certificate validation level.
Step 14	(Optional) [**no**] **timeout** *seconds* **Example:** `apic1(config-host)# timeout 3`	For this LDAP server, specifies the number of seconds APIC waits for a reply to a LDAP request before retransmitting the request. If no timeout is set, the global value is used.
Step 15	**exit** **Example:** `apic1(config-host)# exit`	Returns to global configuration mode.

	Command or Action	Purpose
Step 16	**show ldap-server** **Example:** `apic1(config)# show ldap-server`	

Examples

This example shows how to configure LDAP server settings globally and on one LDAP server.

```
apic1# configure
apic1(config)# ldap-server retries 1
apic1(config)# ldap-server timeout 30
apic1(config)# ldap-server host 192.0.20.73
apic1(config-host)# retries 2
apic1(config-host)# timeout 3
apic1(config-host)# filter sAMAccountName=$userid
apic1(config-host)# key myLdAppassWoRd
apic1(config-host)# ssl-validation-level permissive
apic1(config-host)# enable-ssl
apic1(config-host)# port 389
apic1(config-host)# exit
apic1(config)# show ldap-server
timeout : 30
retries : 1
filter  : sAMAccountName=$userid

Total number of servers : 1

Hostname  : 192.0.20.73
Port      : 389
Timeout   : 3
Retries   : 2
SSL       : yes
SSL Level : permissive
User      : test
```

Configuring the Password Policy

The password policy configuration in this topic set the password history and password change interval properties for all locally authenticated APIC users. You cannot specify different password policies for each locally authenticated user.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	[**no**] **password change-count** *count* **Example:** `apic1(config)# password change-count 5`	Sets the number of password changes allowed within the change interval. The range is 0 to 10 changes.

Step 3	[**no**] **password change-during-interval** {**enable** | **disable**}
	Example:
	`apic1(config)# password change-during-interval enable`
		Enables or disables restricting the number of password changes a locally authenticated user can make within the change interval.

Step 4	[**no**] **password change-interval** *hours* **Example:** `apic1(config)# password change-interval 300`	When the **change-during-interval** is enabled, restricts the number of password changes a locally authenticated user can make within a given number of hours. The range is 1 to 745 hours.
Step 5	[**no**] **password no-change-interval** *hours* **Example:** `apic1(config)# password no-change-interval 60`	Sets a minimum period before which a user cannot change the password again. The range is 1 to 745 hours.
Step 6	**password expiration-warn-time** **Example:** `apic1(config)# password expiration-warn-time 5`	Sets a warning period before password expiration to display warning. The range is 0 to 30 days.
Step 7	[**no**] **password history-count** *count* **Example:** `apic1(config)# password history-count 10`	The password history count allows you to prevent locally authenticated users from reusing the same password over and over again. When this property is configured, APIC stores passwords that were previously used by locally authenticated users up to a maximum of 15 passwords. The passwords are stored in reverse chronological order with the most recent password first to ensure that the only the oldest password can be reused when the history count threshold is reached. A user must create and use the number of passwords configured in the password history count before being able to reuse one. For example, if you set the password history count to 8, a locally authenticated user cannot reuse the first password until after the ninth password has expired. By default, the password history is set to 0. This value disables the history count and allows users to reuse previous passwords at any time. If

	Command or Action	Purpose
		necessary, you can clear a user's password history using the **clear-pwd-history** command in the username configuration mode for that user.
Step 8	[**no**] **password pwd-strength-check** **Example:** `apic1(config)# password pwd-strength-check`	Enforces strong passwords for all users.

Examples

This example shows how to configure global password settings for locally authenticated users.

```
apic1# configure
apic1(config)# password change-count 5
apic1(config)# password change-during-interval enable
apic1(config)# password change-interval 300
apic1(config)# password no-change-interval 60
apic1(config)# password expiration-warn-time 5
apic1(config)# password history-count 10
apic1(config)# password pwd-strength-check
```

This example shows how to prevent the password from being changed within 48 hours after a locally authenticated user changes his or her password.

```
apic1# configure
apic1(config)# password change-during-interval disable
apic1(config)# password no-change-interval 48
```

This example shows how to allow the password to be changed a maximum of once within 24 hours after a locally authenticated user changes his or her password

```
apic1# configure
apic1(config)# password change-count 1
apic1(config)# password change-during-interval enable
apic1(config)# password change-interval 24
```

Configuring Users

Configuring a Locally Authenticated User

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	**username** {*name* | **admin**}
	Example:
	`apic1(config)# username user5`
		Creates a locally-authenticated user account or configures an existing user. The name can be a maximum of 28 characters.

Step 3	[**no**] **first-name** *first* **Example:** `apic1(config-username)# first-name George`	Sets the first name of this user.
Step 4	[**no**] **last-name** *last* **Example:** `apic1(config-username)# last-name Washington`	Sets the last name of this user.
Step 5	[**no**] **email** *email-address* **Example:** `apic1(config-username)# email gwashington@exampleCorp.com`	Sets the email address of this user.
Step 6	[**no**] **phone** *phone-number* **Example:** `apic1(config-username)# phone 14085551212`	Sets the phone number of this user.

Step 7	[**no**] **account-status** {**active** | **inactive** | *status*}
	Example:
	`apic1(config-username)# account-status active`
		Activates or deactivates this user account.

Step 8	**clear-pwd-history** **Example:** `apic1(config-username)# clear-pwd-history`	Clears the user's password history list and allows this user to reuse previous passwords.

	Command or Action	Purpose
Step 9	[**no**] **expires** **Example:** `apic1(config-username)# expires`	Enables expiration of this user account at the date and time configured by the **expiration** command.
Step 10	**expiration** *date-time* **Example:** `apic1(config-username)# expiration 2017-12-31T23:59+08:00`	Sets an expiration date and time for this user account. The format is UTC Date format (YYYY-MM-DDThh:mmTZD). You must also enable expiration by configuring the **expires** command.
Step 11	**password** *password* **Example:** `apic1(config-username)# password c1\$c0123`	Sets the user password. **Note** Special characters such as '$' or '!' should be escaped with a backslash ('\$') in this command to avoid misinterpretation by Bash. The escape backslash is necessary only when setting the password in this command; the user does not enter the backslash when logging in.
Step 12	[**no**] **pwd-lifetime** *days* **Example:** `apic1(config-username)# pwd-lifetime 90`	Sets the lifetime of the user password. The range is 0 to 3650 days.

Step 13	[**no**] **domain** {**all** | **common** | **mgmt** | *domain-name*}
	Example:
	`apic1(config-username)# domain mySecDomain`
		Specifies or creates the AAA domain to which this user belongs.

Step 14	[**no**] **role** *role* **Example:** `apic1(config-domain)# role tenant-admin`	Creates the AAA domain role to set privilege bitmask of a user domain.

Step 15	[**no**] **priv-type** {**readPriv** | **writePriv**}
	Example:
	`apic1(config-role)# priv-type writePriv`
		Creates the AAA domain role to set privilege bitmask of a user domain.

Step 16	**exit** **Example:** `apic1(config-role)# exit`	Returns to domain configuration mode.

	Command or Action	Purpose
Step 17	**exit** **Example:** `apic1(config-domain)# exit`	Returns to username configuration mode.
Step 18	**show username** *name* **Example:** `apic1(config-username)# show username user5`	Displays configuration details about this user.

Examples

This example shows how to configure a local user.

```
apic1# configure terminal
apic1(config)# username user5
apic1(config-username)# first-name George
apic1(config-username)# last-name Washington
apic1(config-username)# email gwashington@exampleCorp.com
apic1(config-username)# phone 14085551212
apic1(config-username)# account-status active
apic1(config-username)# domain mySecDomain
apic1(config-username)# clear-pwd-history
apic1(config-username)# expires
apic1(config-username)# expiration 2017-12-31T23:59+08:00
apic1(config-username)# password c1$c0123
apic1(config-username)# pwd-lifetime 90
apic1(config-username)# domain mySecDomain
apic1(config-domain)# role tenant-admin
apic1(config-role)# priv-type writePriv
apic1(config-role)# exit
apic1(config-domain)# exit
apic1(config-username)# show username user5
UserName                : user5
First-Name              : George
Last-Name               : Washington
Email                   : gwashington@exampleCorp.com
Acount Status           : active
Password strength check : yes
```

What to do next

To configure an SSH key or certificate for the local user, see "Configuring Certificates and SSH-Keys."

Configuring a Certificate and SSH-Key for a Local User

This topic describes how to configure a certificate or an SSH key so that a local user can log in without being prompted for a password.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	**username** {*name* | **admin**}
	Example:
	`apic1(config)# username user5`
		Creates a locally-authenticated user account or configures an existing user. The name can be a maximum of 28 characters.

Step 3	[**no**] **certificate** *certificate-name* **Example:** `apic1(config-username)# certificate myCertificate`	Enters certificate configuration mode.
Step 4	**data** *certificate-data* **Example:** `apic1(config-certificate)# data -----BEGIN CERTIFICATE-----MIIC4j.....`	Sets PEM-encoded certificate.
Step 5	**exit** **Example:** `apic1(config-certificate)# exit`	Returns to username configuration mode.
Step 6	[**no**] **ssh-key** *ssh-key-name* **Example:** `apic1(config-username)# ssh-key mySSHkey`	Sets an SSH key to log in using the SSH client without being prompted for a password.
Step 7	**data** *key-data* **Example:** `apic1(config-ssh-key)# data AAAAB3NzaC1yc2EAA......`	Sets the SSH key. The key can be up to 64 characters.
Step 8	**exit** **Example:** `apic1(config-ssh-key)# exit`	Returns to username configuration mode.

Examples

This example shows how to configure an SSH key and a certificate for a local user.

```
apic1# configure terminal
apic1(config)# username user5
apic1(config-username)# certificate myCertificate
apic1(config-certificate)# data -----BEGIN CERTIFICATE-----MIIC4j.....
```

```
apic1(config-certificate)# exit
apic1(config-username)# ssh-key mySSHkey
apic1(config-ssh-key)# data AAAAB3NzaC1yc2EAA...
apic1(config-ssh-key)# exit
```

Configuring Public Key Infrastructure

Configuring a Certificate Authority and Chain of Trust

Certificate authorities (CAs) manage certificate requests and issue certificates to participating entities such as hosts, network devices, or users. APIC locally stores the self-signed root certificate of the trusted CA (or certificate chain for a subordinate CA). The stored information about a trusted CA is called the trustpoint and the CA itself is called a trustpoint CA.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	[**no**] **crypto ca** *trustpoint-name* **Example:** `apic1(config)# crypto ca myCA`	Enters configuration mode for the specified trustpoint certificate authority (CA).
Step 3	[**no**] **cert-chain** *pem-data* **Example:** `apic1(config-ca)# cert-chain -----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....`	Stores the certificate chain in PEM format. Enter the entire chain of trust from the trustpoint to a trusted root authority.

Examples

This example shows how to configure a CA.

```
apic1# configure

apic1(config)# crypto ca myCA
apic1(config-ca)# cert-chain -----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
```

Configuring Keys and a Keyring

You can obtain an identity certificate for APIC by generating an RSA key pair and associating the key pair with a trustpoint CA where APIC intends to enroll. The RSA keys are stored by APIC in a crypto keyring.

The APIC software allows you to generate an RSA key pair with a configurable key size (or modulus). The default key size is 512. You can also configure an RSA key-pair label. The default key label is the device fully qualified domain name (FQDN).

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **crypto keyring** {**default** | *keyring-name*}
	Example:
	`apic1(config)# crypto keyring myKeyring`
		Creates or configures a keyring to hold an SSL certificate.

Step 3	**regen** **Example:** `apic1(config-keyring)# regen`	Forces regeneration of the RSA key pair.
Step 4	[**no**] **cert** *certificate-data* **Example:** `apic1(config-keyring)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....`	Imports a certificate containing a public key and signed information. The certificate data must be enclosed in quotes.
Step 5	[**no**] **tp** *certificate-name* **Example:** `apic1(config-keyring)# tp myCertificate`	Sets a third-party certificate from a trusted source for device identity.
Step 6	[**no**] **key** *key-data* **Example:** `apic1(config-keyring)# key xxxxxxxxxxxxxxxxxxxxxxx`	Creates the private key of the certificate.

Step 7	[**no**] **modulus** {**mod512** | **mod1024** | **mod1536** | **mod2048**}
	Example:
	`apic1(config-keyring)# modulus mod1024`
		Sets the length of the encryption keys.

Step 8	**exit** **Example:** `apic1(config-keyring)# exit`	Returns to global configuration mode.

Examples

This example shows how to configure a keyring.

```
apic1# configure
apic1(config)# crypto keyring myKeyring
apic1(config-keyring)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
apic1(config-keyring)# tp myCertificate
apic1(config-keyring)# key XXXXXXXXXXXXXXXXXXXXXXX
apic1(config-keyring)# modulus mod1024
apic1(config-keyring)# exit
```

Generating a Certificate Signing Request

A certificate signing request (CSR) is a message that an applicant sends to a CA in order to apply for a digital identity certificate. Before a CSR is created, the applicant first generates a key pair, which keeps the private key secret. The CSR contains information that identifies the applicant, such as the public key generated by the applicant. The corresponding private key is not included in the CSR, but is used to digitally sign the entire request.

Before you begin

Before generating a certificate signing request (CSR), you must configure a trustpoint certificate authority (CA) and generate a key pair.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **crypto keyring** {**default** | *keyring-name*}
	Example:
	`apic1(config)# crypto keyring default`
		Creates or configures a keyring to hold an SSL certificate.

Step 3	**csr** **Example:** `apic1(config-keyring)# csr`	Creates a certificate signing request for this keyring.
Step 4	**subj-name** *name* **Example:** `apic1(config-csr)# subj-name www.exampleCorp.com`	Sets the fully qualified domain name or distinguished name of the requesting device. The name can be up to 64 characters.
Step 5	[**no**] **cert** *certificate-data* **Example:** `apic1(config-csr)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....`	Imports a certificate containing a public key and signed information. The certificate data must be enclosed in quotes.

	Command or Action	Purpose

Step 6	**password**
	Example:
	`apic1(config-csr)# password`
	`Enter password: c1$c0123`
	`Enter password again: c1$c0123`
		Sets the new password.

Step 7	**org-name** **Example:** `apic1(config-csr)# org-name ExampleCorp`	Sets the full legal name of the organization.
Step 8	**org-unit-name** **Example:** `apic1(config-csr)# org-unit-name Sales`	Sets the department or unit name within the organization.
Step 9	**email** **Example:** `apic1(config-csr)# email admin@exampleCorp.com`	Sets the email address of the organization contact person.
Step 10	**locality** *city-name* **Example:** `apic1(config-csr)# locality SanJose`	Sets the city or town of the organization.
Step 11	**state** *state* **Example:** `apic1(config-csr)# state CA`	Sets the state or province in which the organization is located.
Step 12	**country** *country-code* **Example:** `apic1(config-csr)# country US`	Sets the two-letter ISO code for the country where the organization is located.
Step 13	**exit** **Example:** `apic1(config-csr)# exit`	Returns to keyring configuration mode.

Examples

This example shows how to generate a certificate signing request (CSR).

```
apic1# configure
apic1(config)# crypto keyring default
apic1(config-keyring)# csr
apic1(config-csr)# subj-name www.exampleCorp.com
apic1(config-csr)# cert "-----BEGIN CERTIFICATE----- MIIC4jCCAoygAw.....
apic1(config-csr)# pwd c1$c0123
```

```
apic1(config-csr)# org-name ExampleCorp
apic1(config-csr)# org-unit-name Sales
apic1(config-csr)# email admin@exampleCorp.com
apic1(config-csr)# locality SanJose
apic1(config-csr)# state CA
apic1(config-csr)# country US
apic1(config-csr)# exit
```

What to do next

Submit the CSR to a CA.

Configuring Webtokens

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	[**no**] **crypto webtoken** **Example:** `apic1(config)# crypto webtoken`	
Step 3	[**no**] **max-validity-period** *hours* **Example:** `apic1(config-webtoken)# max-validity-period 10`	Sets the maximum validity period for a webtoken. The range is 4 to 24 hours.
Step 4	[**no**] **session-record-flags** *csv-list* **Example:** `apic1(config-webtoken)# session-record-flags login,refresh`	Enables or disables refresh in the session records. The session record flags are specified as a comma-separated value list of one or more of the following flags: **login** , **logout** , and **refresh** .
Step 5	[**no**] **ui-idle-timeout-seconds** *seconds* **Example:** `apic1(config-webtoken)# ui-idle-timeout-seconds 120`	Sets the maximum GUI idle duration before requiring login refresh. The range is 60 to 65525 seconds.
Step 6	[**no**] **webtoken-timeout-seconds** *seconds* **Example:** `apic1(config-webtoken)# webtoken-timeout-seconds 1200`	Sets the webtoken timeout interval. The range is 600 to 9600 seconds.
Step 7	**exit** **Example:**	

	Command or Action	Purpose
	`apic1(config-webtoken)# exit`	

Examples

This example shows how to configure a webtoken.

```
apic1# configure
apic1(config)# crypto webtoken
apic1(config-webtoken)# max-validity-period 10
apic1(config-webtoken)# session-record-flags login,refresh
apic1(config-webtoken)# ui-idle-timeout-seconds 120
apic1(config-webtoken)# webtoken-timeout-seconds 1200
```

Configuring Communication Policies

Configuring the HTTP Policy

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **comm-policy** {**default** | *policy-name*}
	Example:
	`apic1(config)# comm-policy myCommPolicy`
		Enters communication policy configuration mode.

Step 3	**http** **Example:** `apic1(config-comm-policy)# http`	Enters HTTP policy configuration mode.
Step 4	[**no**] **admin-state-enable** **Example:** `apic1(config-http)# admin-state-enable`	Enables HTTP communication service.
Step 5	[**no**] **allow-origin** *url* **Example:** `apic1(config-http)# allow-origin www.example.com`	Specifies the URL to return in the Access-Control-Allow-Origin HTTP header.

	Command or Action	Purpose
Step 6	[**no**] **port** *port-number* **Example:** `apic1(config-http)# port 8080`	Sets the port used for HTTP communication service.
Step 7	[**no**] **redirect** **Example:** `apic1(config-http)# no redirect`	Enables HTTP redirection.
Step 8	[**no**] **request-status-count** *count* **Example:** `apic1(config-http)# request-status-count 512`	Sets the maximum count of HTTP requests to track. The range is 0 to 10240.
Step 9	**exit** **Example:** `apic1(config-http)# exit`	Returns to communications policy configuration mode.

Examples

This example shows how to configure HTTP service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# http
apic1(config-http)# admin-state-enable
apic1(config-http)# allow-origin www.example.com
apic1(config-http)# port 8080
apic1(config-http)# no redirect
apic1(config-http)# request-status-count 512
apic1(config-http)# exit
```

Configuring the HTTPS Policy

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **comm-policy** {**default** | *policy-name*}
	Example:
	`apic1(config)# comm-policy myCommPolicy`
		Enters communication policy configuration mode.

	Command or Action	Purpose
Step 3	**https** **Example:** `apic1(config-comm-policy)# https`	Enters HTTPS policy configuration mode.
Step 4	[**no**] **admin-state-enable** **Example:** `apic1(config-https)# admin-state-enable`	Enables HTTPS communication service.
Step 5	[**no**] **port** *port-number* **Example:** `apic1(config-https)# port 443`	Sets the port used for HTTPS communication service.
Step 6	[**no**] **request-status-count** *count* **Example:** `apic1(config-https)# request-status-count 512`	Sets the maximum count of HTTPS requests to track. The range is 0 to 10240.

Step 7	[**no**] **ssl-protocols** {**TLSv1** | **TLSv1.1** | **TLSv1.2**}
	Example:
	`apic1(config-https)# ssl-protocols TLSv1.1,TLSv1.2`
		Specifies in a comma-separated list the SSL protocols that are supported. The options are **TLSv1** , **TLSv1.1** , and **TLSv1.2** .

Step 8	[**no**] **use-keyring** *keyring-name* **Example:** `apic1(config-https)# use-keyring myKeyRing`	Specifies a keyring to use for the HTTPS server SSL certificate.
Step 9	**exit** **Example:** `apic1(config-https)# exit`	Returns to communications policy configuration mode.

Examples

This example shows how to configure HTTPS service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# https
apic1(config-https)# admin-state-enable
apic1(config-https)# port 443
apic1(config-https)# request-status-count 512
apic1(config-https)# ssl-protocols TLSv1.1,TLSv1.2
apic1(config-https)# use-keyring myKeyRing
apic1(config-https)# exit
```

Configuring the SSH Policy

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **comm-policy** {**default** | *policy-name*}
	Example:
	`apic1(config)# comm-policy myCommPolicy`
		Enters communication policy configuration mode.

Step 3	**ssh-service** **Example:** `apic1(comm-policy)# ssh-service`	Enters SSH policy configuration mode.
Step 4	[**no**] **admin-state-enable** **Example:** `apic1(config-ssh-service)# admin-state-enable`	Enables HTTP communication service.
Step 5	[**no**] **port** *port-number* **Example:** `apic1(config-ssh-service)# port 22`	Sets the port used for SSH communication service.
Step 6	**exit** **Example:** `apic1(config-ssh-service)# exit`	Returns to communications policy configuration mode.

Examples

This example shows how to configure SSH service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# ssh-service
apic1(config-ssh-service)# admin-state-enable
apic1(config-ssh-service)# port 22
apic1(config-ssh-service)# exit
```

Configuring the Telnet Policy

Before you begin

To allow telnet communications, you must configure an out-of-band contract allowing telnet traffic, which is normally on TCP and UDP ports 23.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.

Step 2	[**no**] **comm-policy** {**default** | *policy-name*}
	Example:
	`apic1(config)# comm-policy myCommPolicy`
		Enters communication policy configuration mode.

Step 3	**telnet** **Example:** `apic1(config-comm-policy)# telnet`	Enters Telnet policy configuration mode.
Step 4	[**no**] **admin-state-enable** **Example:** `apic1(config-telnet)# admin-state-enable`	Enables Telnet communication service.
Step 5	[**no**] **port** *port-number* **Example:** `apic1(config-telnet)# port 23`	Sets the port used for Telnet communication service.
Step 6	**exit** **Example:** `apic1(config-telnet)# exit`	Returns to communications policy configuration mode.

Examples

This example shows how to configure Telnet service.

```
apic1# configure
apic1(config)# comm-policy myCommPolicy
apic1(config-comm-policy)# telnet
apic1(config-telnet)# admin-state-enable
apic1(config-telnet)# port 23
apic1(config-telnet)# exit
```

Configuring AES Encryption

Beginning with Cisco APIC Release 1.1(2), the secure properties of APIC configuration files can be encrypted by enabling AES-256 encryption. AES encryption is a global configuration option; all secure properties conform to the AES configuration setting. It is not possible to export a subset of the ACI fabric configuration such as a tenant configuration with AES encryption while not encrypting the remainder of the fabric configuration. For a list of secure properties, see "Appendix K: Secure Properties" in *Cisco Application Centric Infrastructure Fundamentals*.

The APIC uses a 16 to 32 character passphrase to generate the AES-256 keys. The APIC GUI displays a hash of the AES passphrase. This hash can be used to see whether the same passphrase is used on two ACI fabrics. This hash can be copied to a client computer where it can be compared to the passphrase hash of another ACI fabric to see if they were generated with the same passphrase. The hash cannot be used to reconstruct the original passphrase or the AES-256 keys.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**crypto aes** **Example:** `apic1(config)# crypto aes`	Enters AES configuration mode.
Step 3	(Optional) **clear-encryption-key** **Example:** `apic1(config-aes)# clear-encryption-key`	Deletes any existing AES encryption key.

Step 4	**passphrase**
	Example:
	`apic1(config-aes)# passphrase`
	`Enter passphrase: "This is my passphrase"`
	`Enter passphrase again: "This is my passphrase"`
		Specifies the AES encryption passphrase. The passphrase can be 16 to 32 characters and must be enclosed in quotes. For increased security, entering the passphrase is interactive.

Step 5	[**no**] **encryption** **Example:** `apic1(config-aes)# encryption`	Enables (or disables) AES encryption.

Examples

This example shows how to enable AES encryption and configure a passphrase.

```
apic1# configure
apic1(config)# crypto aes
```

```
apic1(config-aes)# clear-encryption-key
apic1(config-aes)# passphrase "This is my passphrase"
apic1(config-aes)# encryption
```

Configuring Fabric Secure Mode

Fabric secure mode prevents parties with physical access to the fabric equipment from adding a switch or APIC controller to the fabric without manual authorization by an administrator. Starting with Cisco APIC Release 1.2(1x), the firmware checks that switches and controllers in the fabric have valid serial numbers associated with a valid Cisco digitally signed certificate. This validation is performed upon upgrade to this release or during an initial installation of the fabric. The default setting for this feature is permissive mode; an existing fabric continues to run as it has after an upgrade to Release 1.2(1). An administrator with fabric-wide access rights must enable strict mode.

Permissive Mode (default) operates as follows:

- Allows an existing fabric to operate normally even though one or more switches have an invalid certificate.
- Does not enforce serial number based authorization.
- Allows auto-discovered controllers and switches to join the fabric without enforcing serial number authorization.

Strict Mode operates as follows:

- Only switches with a valid Cisco serial number and SSL certificate are allowed.
- Enforces serial number based authorization.
- Requires an administrator to manually authorize controllers and switches to join the fabric.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.

Step 2	**system fabric-security-mode** {**permissive** | **strict**}
	Example:
	`apic1(config)# system fabric-security-mode strict`
		Specifies the fabric security mode.

Step 3	**system controller-id** *controller-id* {**approve** | **reject**}
	Example:
	`apic1(config)# system controller-id FCH1750V025 approve`
		In strict mode, approves or rejects a controller to join the fabric.

Examples

This example shows how to change the fabric security mode to strict.

```
apic1# configure
apic1(config)# system fabric-security-mode strict
```

This example shows how to approve a controller to join the fabric when strict mode is configured.

```
apic1# configure
apic1(config)# system controller-id FCH1750V025 approve
```

Configuring COOP Authentication

About COOP Authentication

Council of Oracles Protocol (COOP) is used to communicate the mapping information (location and identity) to the spine proxy. A leaf switch will forward endpoint address information to a spine using ZeroMQ (Zero Message Queue or ZMQ). COOP running on the spine nodes ensures that all spine nodes maintain a consistent copy of end point address and location information and additionally maintains the distributed hash table (DHT) repository of endpoint identity to location mapping database.

Without COOP authentication, it is possible for users to send arbitrary COOP messages, which would be acted on by the fabric nodes. Cisco APIC Release 2.0 adds an MD5 TCP option to provide authentication and integrity protection to the ZMQ TCP transportation. Two authentication modes are supported:

- **Compatible** - COOP accepts both MD5 authenticated and non-authenticated ZMQ connections for message transportation. COOP data path communication gives high priority to transport via secured connections.
- **Strict** - COOP allows MD5 authenticated ZMQ connections only.

Changing the configuration of the COOP authentication type has the following effects:

- When the configuration changes from compatible to strict mode, all non-authenticated ZMQ connections are disconnected.
- When the configuration changes from strict to compatible mode, COOP immediately accepts both authenticated and non-authenticated ZMQ connections.

Configuring COOP Authentication

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters global configuration mode.
Step 2	**coop-fabric** **Example:** `apic1(config)# coop-fabric`	Enters COOP fabric configuration mode.

Step 3	**authentication type** {**compatible** | **strict**}
	Example:
	`apic1(config-coop-fabric)# authentication type compatible`
		Configures the COOP authentication type as one of the following:
		• `compatible` - COOP allows MD5 authenticated and non-authenticated ZMQ connections.
		• `strict` - allows MD5 authenticated ZMQ connections only.

Example

This example shows how to configure COOP authentication in compatible mode:

```
apic1# configure
apic1(config)# coop-fabric
apic1(config-coop-fabric# authentication type compatible
```

Configuring FIPS

About Federal Information Processing Standards (FIPS)

The Federal Information Processing Standards (FIPS) Publication 140-2, Security Requirements for Cryptographic Modules, details the U.S. government requirements for cryptographic modules. FIPS 140-2 specifies that a cryptographic module should be a set of hardware, software, firmware, or some combination that implements cryptographic functions or processes, including cryptographic algorithms and, optionally, key generation, and is contained within a defined cryptographic boundary.

FIPS specifies certain cryptographic algorithms as secure, and it also identifies which algorithms should be used if a cryptographic module is to be called FIPS compliant.

Guidelines and Limitations

Follow these guidelines and limitations:

- When FIPS is enabled, it is applied across Cisco APIC.
- When performing a Cisco APIC software downgrade, you must disable FIPS first.
- Make your passwords a minimum of eight characters in length.
- Disable Telnet. Users should log in using SSH only.
- Delete all SSH Server RSA1 keypairs.
- Disable remote authentication through RADIUS/TACACS+. Only local and LDAP users can be authenticated.
- Secure Shell (SSH) and SNMP are supported.
- Disable SNMP v1 and v2. Any existing user accounts on the switch that have been configured for SNMPv3 should be configured only with SHA for authentication and AES for privacy.
- Starting with release 2.3(1x), FIPS can be configured at the switch level.
- Starting with release 3.1(1x), when FIPs is enabled, NTP will operate in FIPS mode, Under FIPS mode NTP supports authentication with HMAC-SHA1 and no authentication.

Configuring FIPS for Cisco APIC Using NX-OS Style CLI

When FIPS is enabled, it is applied across Cisco APIC.

Procedure

	Command or Action	Purpose
Step 1	**configure** **Example:** `apic1# configure`	Enters configuration mode.
Step 2	**fips mode enable** **Example:** `apic1(config)# fips mode enable`	Enables FIP. The **no fips mode enable** command disables FIPS. You must reboot to complete the configuration. Anytime you change the mode, you must reboot to complete the configuration.

Configuring Control Plane Policing

Information About CoPP

Control Plane Policing (CoPP) protects the control plane, which ensures network stability, reachability, and packet delivery.

This feature allows specification of parameters, for each protocol that can reach the control processor to be rate-limited using a policer. The policing is applied to all traffic destined to any of the IP addresses of the router or Layer 3 switch. A common attack vector for network devices is the denial-of-service (DoS) attack, where excessive traffic is directed at the device interfaces.

The Cisco ACI Leaf/Spine NX-OS provides CoPP to prevent DoS attacks from impacting performance. Such attacks, which can be perpetrated either inadvertently or maliciously, typically involve high rates of traffic destined to the supervisor module of an ACI Leaf/Spine CPU or CPU itself.

The supervisor module of ACI Leaf/Spine switches divides the traffic that it manages into two functional components or planes:

- **Data plane**—Handles all the data traffic. The basic functionality of a Cisco NX-OS device is to forward packets from one interface to another. The packets that are not meant for the switch itself are called the transit packets. These packets are handled by the data plane.
- **Control plane**—Handles all routing protocol control traffic. These protocols, such as the Border Gateway Protocol (BGP) and the Open Shortest Path First (OSPF) Protocol, send control packets between devices. These packets are destined to router addresses and are called control plane packets.

The ACI Leaf/Spine supervisor module has a control plane and is critical to the operation of the network. Any disruption or attacks to the supervisor module will result in serious network outages. For example, excessive traffic to the supervisor module could overload and slow down the performance of the entire Cisco ACI fabric. Another example is a DoS attack on the ACI Leaf/Spine supervisor module that could generate IP traffic streams to the control plane at a very high rate, forcing the control plane to spend a large amount of time in handling these packets and preventing the control plane from processing genuine traffic.

Examples of DoS attacks are as follows:

- Internet Control Message Protocol (ICMP) echo requests
- IP fragments
- TCP SYN flooding

These attacks can impact the device performance and have the following negative effects:

- Reduced service quality (such as poor voice, video, or critical applications traffic)
- High route processor or switch processor CPU utilization
- Route flaps due to loss of routing protocol updates or keepalives
- Processor resource exhaustion, such as the memory and buffers
- Indiscriminate drops of incoming packets

Note ACI Leaf/Spines are by default protected by CoPP with default settings. This feature allows for tuning the parameters on a group of nodes based on customer needs.

Control Plane Protection

To protect the control plane, the Cisco NX-OS running on ACI Leaf/Spines segregates different packets destined for the control plane into different classes. Once these classes are identified, the Cisco NX-OS device polices the packets, which ensures that the supervisor module is not overwhelmed.

Control Plane Packet Types:

Different types of packets can reach the control plane:

- **Receive Packets**—Packets that have the destination address of a router. The destination address can be a Layer 2 address (such as a router MAC address) or a Layer 3 address (such as the IP address of a router interface). These packets include router updates and keepalive messages. Multicast packets can also be in this category where packets are sent to multicast addresses that are used by a router.
- **Exception Packets**—Packets that need special handling by the supervisor module. For example, if a destination address is not present in the Forwarding Information Base (FIB) and results in a miss, the supervisor module sends an ICMP unreachable packet back to the sender. Another example is a packet with IP options set.
- **Redirect Packets**—Packets that are redirected to the supervisor module. Features such as Dynamic Host Configuration Protocol (DHCP) snooping or dynamic Address Resolution Protocol (ARP) inspection redirect some packets to the supervisor module.
- **Glean Packets**—If a Layer 2 MAC address for a destination IP address is not present in the FIB, the supervisor module receives the packet and sends an ARP request to the host.

All of these different packets could be maliciously used to attack the control plane and overwhelm the Cisco ACI Fabric. CoPP classifies these packets to different classes and provides a mechanism to individually control the rate at which the ACI Leaf/Spine supervisor module receives these packets.

Classification for CoPP:

For effective protection, the ACI Leaf/Spine NX-OS classifies the packets that reach the supervisor modules to allow you to apply different rate controlling policies based on the type of the packet. For example, you might want to be less strict with a protocol packet such as Hello messages but more strict with a packet that is sent to the supervisor module because the IP option is set.

Rate Controlling Mechanisms:

Once the packets are classified, the ACI Leaf/Spine NX-OS has different mechanisms to control the rate at which packets arrive at the supervisor module.

You can configure the following parameters for policing:

- **Committed information rate (CIR)**—Desired bandwidth, specified as a bit rate or a percentage of the link rate.
- **Committed burst (BC)**—Size of a traffic burst that can exceed the CIR within a given unit of time and not impact scheduling.

Default Policing Policies:

When the Cisco ACI Leaf/Spine is bootup, the platform setup pre-defined CoPP parameters for different protocols are based on the tests done by Cisco.

Guidelines and Limitations for CoPP

CoPP has the following configuration guidelines and limitations:

- We recommend that you use the default CoPP policy initially and then later modify the CoPP policies based on the data center and application requirements.
- Customizing CoPP is an ongoing process. CoPP must be configured according to the protocols and features used in your specific environment as well as the supervisor features that are required by the server environment. As these protocols and features change, CoPP must be modified.
- We recommend that you continuously monitor CoPP. If drops occur, determine if CoPP dropped traffic unintentionally or in response to a malfunction or attack. In either event, analyze the situation and evaluate the need to modify the CoPP policies.
- You must ensure that the CoPP policy does not filter critical traffic such as routing protocols or interactive access to the device. Filtering this traffic could prevent remote access to the Cisco ACI Leaf/Spine and require a console connection.
- Do not mis-configure CoPP pre-filter entries. CoPP pre-filter entries might impact connectivity to multi-pod configurations, remote leaf switches, and Cisco ACI Multi-Site deployments.
- You can use the APIC UI to be able to tune the CoPP parameters.
- Per interface per protocol is only supported on Leaf switches.
- FEX ports are not supported on per interface per protocol.
- For per interface per protocol the supported protocols are; ARP, ICMP, CDP, LLDP, LACP, BGP, STP, BFD, and OSPF.
- The TCAM entry maximum for per interface per protocol is 256. Once the threshold is exceeded a fault will be raised.

Configuring CoPP Using the Cisco NX-OS CLI

Procedure

Step 1 Configure a CoPP leaf profile:

Example:

```
# configure copp Leaf Profile
apic1(config)# policy-map type control-plane-leaf leafProfile
apic1(config-pmap-copp-leaf)# profile-type custom
apic1(config-pmap-copp-leaf)# set arpRate 786
# create a policy group to be applied on leaves
apic1(config)# template leaf-policy-group coppForLeaves
apic1(config-leaf-policy-group)# copp-aggr leafProfile
apic1(config-leaf-policy-group)# exit
# apply the leaves policy group on leaves
apic1(config)# leaf-profile applyCopp
```

```
apic1(config-leaf-profile)# leaf-group applyCopp
apic1(config-leaf-group)# leaf 101-102
apic1(config-leaf-group)# leaf-policy-group coppForLeaves
```

Step 2 Configure a CoPP Spine profile:

Example:

```
# configure copp Spine Profile
apic1(config)# policy-map type control-plane-spine spineProfile
apic1(config-pmap-copp-spine)# profile-type custom
apic1(config-pmap-copp-spine)# set arpRate 786
# create a policy group to be applied on spines
apic1(config)# template leaf-policy-group coppForSpines
apic1(config-spine-policy-group)# copp-aggr spineProfile
apic1(config-spine-policy-group)# exit
# apply the spine policy group on spines
apic1(config)# spine-profile applyCopp
apic1(config-spine-profile)# spine-group applyCopp
apic1(config-spine-group)# spine 201-202
apic1(config-spine-group)# spine-policy-group coppForSpines
```

Configuring Per Interface Per Protocol CoPP Policy Using the NX-OS Style CLI

Procedure

Step 1 Define the CoPP class map and policy map:

Example:

```
(config)# policy-map type control-plane-if <name>
        (config-pmap-copp)# protocol bgp bps <value>
        (config-pmap-copp)# protocol ospf bps <value>
```

Step 2 Applying the configuration to an interface on the leaf:

Example:

```
(config)# leaf 101
        (config-leaf)# int eth 1/10
        (config-leaf-if)# service-policy type control-plane-if output<name>
```

Configuring First Hop Security

About First Hop Security

First-Hop Security (FHS) features enable a better IPv4 and IPv6 link security and management over the layer 2 links. In a service provider environment, these features closely control address assignment and derived operations, such as Duplicate Address Detection (DAD) and Address Resolution (AR).

The following supported FHS features secure the protocols and help build a secure endpoint database on the fabric leaf switches, that are used to mitigate security threats such as MIM attacks and IP thefts:

- ARP Inspection—allows a network administrator to intercept, log, and discard ARP packets with invalid MAC address to IP address bindings.
- ND Inspection—learns and secures bindings for stateless autoconfiguration addresses in Layer 2 neighbor tables.
- DHCP Inspection—validates DHCP messages received from untrusted sources and filters out invalid messages.
- RA Guard—allows the network administrator to block or reject unwanted or rogue router advertisement (RA) guard messages.
- IPv4 and IPv6 Source Guard—blocks any data traffic from an unknown source.
- Trust Control—a trusted source is a device that is under your administrative control. These devices include the switches, routers, and servers in the Fabric. Any device beyond the firewall or outside the network is an untrusted source. Generally, host ports are treated as untrusted sources.

FHS features provide the following security measures:

- **Role Enforcement**—Prevents untrusted hosts from sending messages that are out the scope of their role.
- **Binding Enforcement**—Prevents address theft.
- **DoS Attack Mitigations**—Prevents malicious end-points to grow the end-point database to the point where the database could stop providing operation services.
- **Proxy Services**—Provides some proxy-services to increase the efficiency of address resolution.

FHS features are enabled on a per tenant bridge domain (BD) basis. As the bridge domain, may be deployed on a single or across multiple leaf switches, the FHS threat control and mitigation mechanisms cater to a single switch and multiple switch scenarios.

ACI FHS Deployment

Most FHS features are configured in a two-step fashion: firstly you define a policy which describes the behavior of the feature, secondly you apply this policy to a "domain" (being the Tenant Bridge Domain or the Tenant Endpoint Group). Different policies that define different behaviors can be applied to different intersecting domains. The decision to use a specific policy is taken by the most specific domain to which the policy is applied.

The policy options can be defined from the Cisco APIC GUI found under the Tenant_*name*>Networking>Protocol Policies>First Hop Security tab.

Guidelines and Limitations

Follow these guidelines and limitations:

- Starting with release 3.1(1), FHS is supported with virtual Endpoints (AVS only).
- FHS is supported with both VLAN and VXLAN encapsulation.

- Any secured endpoint entry in the FHS Binding Table Database in **DOWN** state will get cleared after **18 Hours** of timeout. The entry moves to **DOWN** state when the front panel port where the entry is learned is link down. During this window of **18 Hours**, if the endpoint is moved to a different location and is seen on a different port, the entry will be gracefully moved out of **DOWN** state to **REACHABLE/STALE** as long as the endpoint is reachable from the other port it is moved from.
- When IP Source Guard is enabled, the IPv6 traffic that is sourced using IPv6 Link Local address as IP source address is not subject to the IP Source Guard enforcement (i.e. Enforcement of Source Mac <=> Source IP Bindings secured by IP Inspect Feature). This traffic is permitted by default irrespective of binding check failures.
- FHS is not supported on L3Out interfaces.
- FHS is not supported N9K-M12PQ based TORs.
- FHS in ACI Multi-Site is a site local capability therefore it can only be enabled in a site from the APIC cluster. Also, FHS in ACI Multi-Site only works when the BD and EPG is site local and not stretched across sites. FHS security cannot be enabled for stretched BD or EPGs.
- FHS is not supported on a Layer 2 only bridge domain.
- Enabling FHS feature can disrupt traffic for 50 seconds because the EP in the BD are flushed and EP Learning in the BD is disabled for 50 seconds.

Configuring FHS Using the NX-OS CLI

Before you begin

- The tenant and Bridge Domain configured.

Procedure

Step 1 **configure**

Enters configuration mode.

Example:

```
apic1# configure
```

Step 2 Configure FHS policy.

Example:

```
apic1(config)# tenant coke
apic1(config-tenant)# first-hop-security
apic1(config-tenant-fhs)# security-policy pol1
apic1(config-tenant-fhs-secpol)#
apic1(config-tenant-fhs-secpol)# ip-inspection-admin-status enabled-both
apic1(config-tenant-fhs-secpol)# source-guard-admin-status enabled-both
apic1(config-tenant-fhs-secpol)# router-advertisement-guard-admin-status enabled
apic1(config-tenant-fhs-secpol)# router-advertisement-guard
apic1(config-tenant-fhs-raguard)#
apic1(config-tenant-fhs-raguard)# managed-config-check
apic1(config-tenant-fhs-raguard)# managed-config-flag
apic1(config-tenant-fhs-raguard)# other-config-check
```

```
apic1(config-tenant-fhs-raguard)# other-config-flag
apic1(config-tenant-fhs-raguard)# maximum-router-preference low
apic1(config-tenant-fhs-raguard)# minimum-hop-limit 10
apic1(config-tenant-fhs-raguard)# maximum-hop-limit 100
apic1(config-tenant-fhs-raguard)# exit
apic1(config-tenant-fhs-secpol)# exit
apic1(config-tenant-fhs)# trust-control tcpol1
pic1(config-tenant-fhs-trustctrl)# arp
apic1(config-tenant-fhs-trustctrl)# dhcpv4-server
apic1(config-tenant-fhs-trustctrl)# dhcpv6-server
apic1(config-tenant-fhs-trustctrl)# ipv6-router
apic1(config-tenant-fhs-trustctrl)#  router-advertisement
apic1(config-tenant-fhs-trustctrl)# neighbor-discovery
apic1(config-tenant-fhs-trustctrl)# exit
apic1(config-tenant-fhs)# exit
apic1(config-tenant)# bridge-domain bd1
apic1(config-tenant-bd)# first-hop-security security-policy pol1
apic1(config-tenant-bd)# exit
apic1(config-tenant)# application ap1
apic1(config-tenant-app)# epg epg1
apic1(config-tenant-app-epg)# first-hop-security trust-control tcpol1
```

Step 3 Show FHS configuration example:

Example:

```
leaf4# show fhs bt all

Legend:
    TR       : trusted-access                       UNRES : unresolved              Age
: Age since creation
    UNTR     : untrusted-access                     UNDTR : undetermined-trust      CRTNG
: creating
    UNKNW    : unknown                              TENTV : tentative               INV
: invalid
    NDP      : Neighbor Discovery Protocol          STA   : static-authenticated    REACH
: reachable
    INCMP    : incomplete                           VERFY : verify                  INTF
: Interface
    TimeLeft : Remaining time since last refresh    LM    : lla-mac-match           DHCP
: dhcp-assigned

EPG-Mode:
    U : unknown    M : mac    V : vlan    I : ip

BD-VNID             BD-Vlan             BD-Name
15630220            3                   t0:bd200

-------------------------------------------------------------------------------------------
| Origin | IP               | MAC               | INTF   | EPG(sclass)(mode) | Trust-lvl |
State | Age      | TimeLeft |
-------------------------------------------------------------------------------------------
| ARP    | 192.0.200.12     | D0:72:DC:A0:3D:4F | eth1/1 | epg300(49154)(V)  | LM,TR     |
STALE | 00:04:49 | 18:08:13 |
| ARP    | 172.29.205.232   | D0:72:DC:A0:3D:4F | eth1/1 | epg300(49154)(V)  | LM,TR     |
STALE | 00:03:55 | 18:08:21 |
| ARP    | 192.0.200.21     | D0:72:DC:A0:3D:4F | eth1/1 | epg300(49154)(V)  | LM,TR     |
REACH | 00:03:36 | 00:00:02 |
| LOCAL  | 192.0.200.1      | 00:22:BD:F8:19:FF | vlan3  | LOCAL(16387)(I)   | STA       |
REACH | 04:49:41 | N/A      |
| LOCAL  | fe80::200        | 00:22:BD:F8:19:FF | vlan3  | LOCAL(16387)(I)   | STA       |
REACH | 04:49:40 | N/A      |
| LOCAL  | 2001:0:0:200::1  | 00:22:BD:F8:19:FF | vlan3  | LOCAL(16387)(I)   | STA       |
REACH | 04:49:39 | N/A      |
------------------------------------------------------------------------------------------
```

The trust levels are:

- **TR**— Trusted. Displayed when the endpoint is learned from an EPG where the trust configuration is enabled.
- **UNTR**— Untrusted. Displayed when the endpoint is learned from an EPG where the trust configuration is not enabled.
- **UNDTR**— Undetermined. Displayed in the case of a DHCP relay topology where the DHCP server bridge domain (BD) is on a remote leaf and the DHCP clients are on a local leaf. In this situation, the local leaf will not know whether the DHCP server BD has trust DHCP enabled.

Step 4 Show violations with the different types and reasons example:

Example:

```
leaf4# show fhs violations all

Violation-Type:
    POL  : policy      THR : address-theft-remote
    ROLE : role        TH  : address-theft
    INT  : internal

Violation-Reason:
    IP-MAC-TH   : ip-mac-theft                    OCFG_CHK  : ra-other-cfg-check-fail
ANC-COL     : anchor-collision
    PRF-LVL-CHK : ra-rtr-pref-level-check-fail    INT-ERR   : internal-error
TRUST-CHK   : trust-check-fail
    SRV-ROL-CHK : srv-role-check-fail             ST-EP-COL : static-ep-collision
LCL-EP-COL  : local-ep-collision
    MAC-TH      : mac-theft                       EP-LIM    : ep-limit-reached
MCFG-CHK    : ra-managed-cfg-check-fail
    HOP-LMT-CHK : ra-hoplimit-check-fail          MOV-COL   : competing-move-collision
RTR-ROL-CHK : rtr-role-check-fail
    IP-TH       : ip-theft

EPG-Mode:
    U : unknown    M : mac    V : vlan    I : ip

BD-VNID             BD-Vlan             BD-Name
15630220            3                   t0:bd200
------------------------------------------------------------------------------------------
| Type | Last-Reason | Proto | IP           | MAC               | Port    | EPG(sclass)(mode)
 | Count |
------------------------------------------------------------------------------------------
| THR  | IP-TH       | ARP   | 192.0.200.21 | D0:72:DC:A0:3D:4F | tunnel5 | epg300(49154)(V)
  | 21    |
------------------------------------------------------------------------------------------
Table Count: 1
```

Step 5 Show FHS configuration:

Example:

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security binding-table

Pod/Node  Type    Family  IP Address           MAC Address        Interface     Level
        State
--------  ------  ------  --------------------  -----------------  ------------
---------------  -----
1/102     local   ipv4    192.0.200.1          00:22:BD:F8:19:FF  vlan3         static-
```

```
        reach

authenticated    able
1/102     local   ipv6    fe80::200             00:22:BD:F8:19:FF  vlan3         static-
        reach

authenticated    able
1/102     local   ipv6    2001:0:0:200::1       00:22:BD:F8:19:FF  vlan3         static-
        reach

authenticated    able
1/101    arp     ipv4    192.0.200.23         D0:72:DC:A0:02:61  eth1/2       lla-mac-match
    stale
                                                                               ,untrusted-

                                                                                  access
1/101     local   ipv4    192.0.200.1           00:22:BD:F8:19:FF  vlan3         static-
        reach

authenticated    able
1/101    nd      ipv6   fe80::d272:dcff:fea0  D0:72:DC:A0:02:61  eth1/2       lla-mac-match
    reach
                          :261                                                 ,untrusted-
      able
                                                                                  access
1/101    nd      ipv6    2001:0:0:200::20     D0:72:DC:A0:02:61  eth1/2       lla-mac-match
    stale
                                                                               ,untrusted-

                                                                                  access
1/101    nd      ipv6   2001::200:d272:dcff:  D0:72:DC:A0:02:61  eth1/2       lla-mac-match
    stale
                          fea0:261                                             ,untrusted-

                                                                                  access
1/101     local   ipv6    fe80::200             00:22:BD:F8:19:FF  vlan3         static-
        reach

authenticated    able
1/101     local   ipv6    2001:0:0:200::1       00:22:BD:F8:19:FF  vlan3         static-
        reach

authenticated    able
1/103     local   ipv4    192.0.200.1           00:22:BD:F8:19:FF  vlan4         static-
        reach

authenticated    able
1/103     local   ipv6    fe80::200             00:22:BD:F8:19:FF  vlan4         static-
        reach

authenticated    able
1/103     local   ipv6    2001:0:0:200::1       00:22:BD:F8:19:FF  vlan4         static-
        reach

authenticated    able
1/104    arp     ipv4    192.0.200.10         F8:72:EA:AD:C4:7C  eth1/1       lla-mac-match
    stale

,trusted-access
1/104    arp     ipv4    172.29.207.222       D0:72:DC:A0:3D:4C  eth1/1       lla-mac-match
    stale

,trusted-access
1/104     local   ipv4    192.0.200.1           00:22:BD:F8:19:FF  vlan4         static-
```

```
        reach

authenticated    able
1/104    nd     ipv6    fe80::fa72:eaff:fead  F8:72:EA:AD:C4:7C  eth1/1       lla-mac-match
    stale
                           :c47c
,trusted-access
1/104    nd     ipv6    2001:0:0:200::10      F8:72:EA:AD:C4:7C  eth1/1       lla-mac-match
    stale

,trusted-access
1/104     local   ipv6    fe80::200             00:22:BD:F8:19:FF  vlan4         static-
        reach

authenticated    able
1/104     local   ipv6    2001:0:0:200::1       00:22:BD:F8:19:FF  vlan4         static-
        reach

authenticated    able


Pod/Node  Type    IP Address            Creation TS                    Last Refresh TS
            Lease Period
--------  ------  --------------------  -----------------------------
-----------------------------  ------------
1/102     local   192.0.200.1           2017-07-20T04:22:38.000+00:00
2017-07-20T04:22:38.000+00:00
1/102     local   fe80::200             2017-07-20T04:22:56.000+00:00
2017-07-20T04:22:56.000+00:00
1/102     local   2001:0:0:200::1       2017-07-20T04:22:57.000+00:00
2017-07-20T04:22:57.000+00:00
1/101     arp     192.0.200.23          2017-07-27T10:55:20.000+00:00
2017-07-27T16:07:24.000+00:00
1/101     local   192.0.200.1           2017-07-27T10:48:09.000+00:00
2017-07-27T10:48:09.000+00:00
1/101     nd      fe80::d272:dcff:fea0  2017-07-27T10:52:16.000+00:00
2017-07-27T16:04:29.000+00:00
                   :261
1/101     nd      2001:0:0:200::20      2017-07-27T10:57:32.000+00:00
2017-07-27T16:07:24.000+00:00
1/101     nd      2001::200:d272:dcff:  2017-07-27T11:21:45.000+00:00
2017-07-27T16:07:24.000+00:00
                   fea0:261
1/101     local   fe80::200             2017-07-27T10:48:10.000+00:00
2017-07-27T10:48:10.000+00:00
1/101     local   2001:0:0:200::1       2017-07-27T10:48:11.000+00:00
2017-07-27T10:48:11.000+00:00
1/103     local   192.0.200.1           2017-07-26T22:03:56.000+00:00
2017-07-26T22:03:56.000+00:00
1/103     local   fe80::200             2017-07-26T22:03:57.000+00:00
2017-07-26T22:03:57.000+00:00
1/103     local   2001:0:0:200::1       2017-07-26T22:03:58.000+00:00
2017-07-26T22:03:58.000+00:00
1/104     arp     192.0.200.10          2017-07-27T11:21:13.000+00:00
2017-07-27T16:05:48.000+00:00
1/104     arp     172.29.207.222        2017-07-27T11:54:48.000+00:00
2017-07-27T16:06:38.000+00:00
1/104     local   192.0.200.1           2017-07-27T10:49:13.000+00:00
2017-07-27T10:49:13.000+00:00
1/104     nd      fe80::fa72:eaff:fead  2017-07-27T11:21:13.000+00:00
2017-07-27T16:06:43.000+00:00
                   :c47c
1/104     nd      2001:0:0:200::10      2017-07-27T11:21:13.000+00:00
2017-07-27T16:06:19.000+00:00
```

```
1/104     local   fe80::200             2017-07-27T10:49:14.000+00:00
2017-07-27T10:49:14.000+00:00
1/104     local   2001:0:0:200::1       2017-07-27T10:49:15.000+00:00
2017-07-27T10:49:15.000+00:00

swtb23-ifc1#

swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics arp
Pod/Node         : 1/101
Request Received : 4
Request Switched : 2
Request Dropped  : 2
Reply Received   : 257
Reply Switched   : 257
Reply Dropped    : 0

Pod/Node         : 1/104
Request Received : 6
Request Switched : 6
Request Dropped  : 0
Reply Received   : 954
Reply Switched   : 954
Reply Dropped    : 0


swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics dhcpv4
Pod/Node                    : 1/102
Discovery Received          : 5
Discovery Switched          : 5
Discovery Dropped           : 0
Offer Received              : 0
Offer Switched              : 0
Offer Dropped               : 0
Request Received            : 0
Request Switched            : 0
Request Dropped             : 0
Ack Received                : 0
Ack Switched                : 0
Ack Dropped                 : 0
Nack Received               : 0
Nack Switched               : 0
Nack Dropped                : 0
Decline Received            : 0
Decline Switched            : 0
Decline Dropped             : 0
Release Received            : 0
Release Switched            : 0
Release Dropped             : 0
Information Received        : 0
Information Switched        : 0
Information Dropped         : 0
Lease Query Received        : 0
Lease Query Switched        : 0
Lease Query Dropped         : 0
Lease Active Received       : 0
Lease Active Switched       : 0
Lease Active Dropped        : 0
Lease Unassignment Received : 0
Lease Unassignment Switched : 0
Lease Unassignment Dropped  : 0
Lease Unknown Received      : 0
Lease Unknown Switched      : 0
Lease Unknown Dropped       : 0
```

```
swtb23-ifc1# show tenant t0 bridge-domain bd200 first-hop-security statistics
neighbor-discovery
Pod/Node                        : 1/101
Neighbor Solicitation Received  : 125
Neighbor Solicitation Switched  : 121
Neighbor Solicitation Dropped   : 4
Neighbor Advertisement Received : 519
Neighbor Advertisement Switched : 519
Neighbor Advertisement Drop     : 0
Router Solicitation Received    : 4
Router Solicitation Switched    : 4
Router Solicitation Dropped     : 0
Router Adv Received             : 0
Router Adv Switched             : 0
Router Adv Dropped              : 0
Redirect Received               : 0
Redirect Switched               : 0
Redirect Dropped                : 0

Pod/Node                        : 1/104
Neighbor Solicitation Received  : 123
Neighbor Solicitation Switched  : 47
Neighbor Solicitation Dropped   : 76
Neighbor Advertisement Received : 252
Neighbor Advertisement Switched : 228
Neighbor Advertisement Drop     : 24
Router Solicitation Received    : 0
Router Solicitation Switched    : 0
Router Solicitation Dropped     : 0
Router Adv Received             : 53
Router Adv Switched             : 6
Router Adv Dropped              : 47
Redirect Received               : 0
Redirect Switched               : 0
Redirect Dropped                : 0
```

Configuring 802.1x

802.1X Overview

802.1X defines a client-server based access control and authentication protocol that restricts unauthorized clients from connecting to a LAN through publicly accessible ports. The authentication server authenticates each client connected to a Cisco NX-OS device port.

Until the client is authenticated, 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) traffic through the port to which the client is connected. After authentication is successful, normal traffic can pass through the port.

The RADIUS distributed client/server system allows you to secure networks against unauthorized access. In the Cisco ACI implementation, RADIUS clients run on the ToRs and send authentication and accounting requests to a central RADIUS server that contains all user authentication and network service access information.

Host Support

The 802.1X feature can restrict traffic on a port with the following modes:

- **Single-host Mode**—Allows traffic from only one endpoint device on the 802.1X port. Once the endpoint device is authenticated, the APIC puts the port in the authorized state. When the endpoint device leaves the port, the APIC put the port back into the unauthorized state. A security violation in 802.1X is defined as a detection of frames sourced from any MAC address other than the single MAC address authorized as a result of successful authentication. In this case, the interface on which this security association violation is detected (EAPOL frame from the other MAC address) will be disabled. Single host mode is applicable only for host-to-switch topology and when a single host is connected to the Layer 2 (Ethernet access port) or Layer 3 port (routed port) of the APIC.
- **Multi-host Mode**—Allows multiple hosts per port but only the first one gets authenticated. The port is moved to the authorized state after the successful authorization of the first host. Subsequent hosts are not required to be authorized to gain network access once the port is in the authorized state. If the port becomes unauthorized when reauthentication fails or an EAPOL logoff message is received, all attached hosts are denied access to the network. The capability of the interface to shut down upon security association violation is disabled in multiple host mode. This mode is applicable for both switch-to-switch and host-to-switch topologies
- **Multi-Auth Mode**—Allows multiple hosts and all hosts are authenticated separately.

  Note Each host must have the same EPG/VLAN information.

- **Multi-Domain Mode**—For separate data and voice domain. For use with IP-Phones.

Authentication Modes

ACI 802.1X supports the following authentication modes:

- **EAP**—The authenticator then sends an EAP-request/identity frame to the supplicant to request its identity (typically, the authenticator sends an initial identity/request frame followed by one or more requests for authentication information). When the supplicant receives the frame, it responds with an EAP-response/identity frame.
- **MAB**—MAC Authentication Bypass (MAB) is supported as the fallback authentication mode. MAB enables port-based access control using the MAC address of the endpoint. A MAB-enabled port can be dynamically enabled or disabled based on the MAC address of the device that connects to it. Prior to MAB, the endpoint's identity is unknown and all traffic is blocked. The switch examines a single packet to learn and authenticate the source MAC address. After MAB succeeds, the endpoint's identity is known and all traffic from that endpoint is allowed. The switch performs source MAC address filtering to help ensure that only the MAB-authenticated endpoint is allowed to send traffic.

Guidelines and Limitations

802.1X port-based authentication has the following configuration guidelines and limitations:

- The Cisco ACI supports 802.1X authentication only on physical ports.

- The Cisco ACI does not support 802.1X authentication on port channels or subinterfaces.
- The Cisco ACI supports 802.1X authentication on member ports of a port channel but not on the port channel itself.
- Member ports with and without 802.1X configuration can coexist in a port channel. However, you must ensure the identical 802.1X configuration on all the member ports in order for channeling to operate with 802.1X
- When you enable 802.1X authentication, supplicants are authenticated before any other Layer 2 or Layer 3 features are enabled on an Ethernet interface.
- 802.1X is supported only on a leaf chassis that is EX or FX type.
- 802.1X is only supported Fabric Access Ports. 802.1X is not supported on Port-Channels, or Virtual-Port-Channels.
- IPv6 is not supported for dot1x clients in the 3.2(1) release.
- While downgrading to earlier releases especially in cases where certain interface config (host mode and auth type) is unsupported in that release, dot1x authentication type defaults to none. Host-mode would need to be manually re-configured to either single host/multi host depending on whatever is desired. This is to ensure that the user configures only the supported modes/auth-types in that release and doesn't run into unsupported scenarios.
- Multi-Auth supports 1 voice client and multiple data clients (all belonging to same data vlan/epg).
- Fail-epg/vlan under 802.1X node authentication policy is a mandatory configuration.
- Multi-domain more than 1 voice and 1 data client puts the port in security disabled state.
- The following platforms are not supported for 802.1X:
  - N9K-C9396PX
  - N9K-M12PQ
  - N9K-C93128TX
  - N9K-M12PQ

Configuration Overview

The 802.1X and RADIUS processes are started only when enabled by APIC. Internally, this means dot1x process is started when 802.1X Inst MO is created and radius process is created when radius entity is created. Dot1x based authentication must be enabled on each interface for authenticating users connected on that interface otherwise the behavior is unchanged.

RADIUS server configuration is done separately from dot1x configuration. RADIUS configuration defines a list of RADIUS servers and a way to reach them. Dot1x configuration contains a reference to RADIUS group (or default group) to use for authentication.

Both 802.1X and RADIUS configuration must be done for successful authentication. Order of configuration is not important but if there is no RADIUS configuration then 802.1X authentication cannot be successful.

Configuring 802.1X Node Authentication Using NX-OS Style CLI

Procedure

Step 1 Configure the radius authentication group:

Example:

```
apic1# configure
apic1(config)#
apic1(config)# aaa group server radius myradiusgrp
apic1(config-radius)#server 192.168.0.100 priority 1
apic1(config-radius)#exit
```

Step 2 Configure node level port authentication policy:

Example:

```
apic1(config)# policy-map type port-authentication mydot1x
apic1(config-pmap-port-authentication)#radius-provider-group myradiusgrp
apic1(config-pmap-port-authentication)#fail-auth-vlan 2001
apic1(config-pmap-port-authentication)#fail-auth-epg tenant tn1 application ap1 epg epg256
apic1(config)# exit
```

Step 3 Configure policy group and specify port authentication policy in the group:

Example:

```
apic1(config)#template leaf-policy-group lpg2
apic1(config-leaf-policy-group)# port-authentication mydot1x
apic1(config-leaf-policy-group)#exit
```

Step 4 Configure the leaf switch profile:

Example:

```
apic1(config)# leaf-profile mylp2
apic1(config-leaf-profile)#leaf-group mylg2
apic1(config-leaf-group)#  leaf-policy-group lpg2
apic1(config-leaf-group)#exit
```

Configuring 802.1X Port Authentication Using the NX-OS Style CLI

Procedure

Step 1 Configure a Policy Group:

Example:

```
apic1# configure
apic1(config)#
apic1(config)# template policy-group mypol
apic1(config-pol-grp-if)# switchport port-authentication mydot1x
apic1(config-port-authentication)# host-mode multi-host
apic1(config-port-authentication)# no shutdown
```

```
apic1(config-port-authentication)# exit
apic1(config-pol-grp-if)# exit
```

Step 2 Configure the leaf interface profile:

Example:

```
apic1(config)#
apic1(config)#leaf-interface-profile myprofile
apic1(config-leaf-if-profile)#leaf-interface-group mygroup
apic1(config-leaf-if-group)# interface ethernet 1/10-12
apic1(config-leaf-if-group)# policy-group mypol
apic1(config-leaf-if-group)# exit
apic1(config-leaf-if-profile)# exit
```

Step 3 Configure the leaf profile:

Example:

```
apic1(config)#
apic1(config)# leaf-profile myleafprofile
apic1(config-leaf-profile)# leaf-group myleafgrp
apic1(config-leaf-group)# leaf 101
apic1(config-leaf-group)# exit
```

Step 4 Apply an interface policy on the leaf switch profile:

Example:

```
apic1(config-leaf-profile)# leaf-interface-profile myprofile
apic1(config-leaf-group)# exit
```