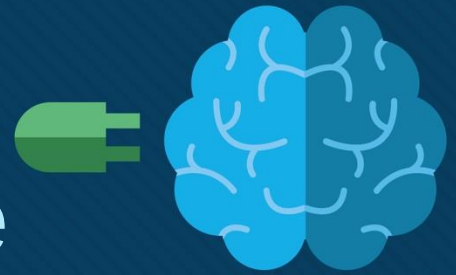

Chapter 26: Network Device Access Control and Infrastructure Security

Instructor Materials

CCNP Enterprise: Core Networking

Chapter 26 Content

This chapter covers the following content:

Access Control Lists (ACLs) - This section explains how to configure and verify ACLs to secure the network infrastructure.

Terminal Lines and Password Protection - This section explains how to configure and verify local network device access control through local usernames and passwords for authentication and how to configure and verify role-based access control (RBAC) through privilege levels.

Authentication, Authorization, and Accounting (AAA) - This section explains how to configure and verify network device access control on IOS through an AAA TACACS+ server.

Zone-Based Firewall (ZBFW) - This section explains how to configure and verify stateful firewall functionality on IOS routers.

Control Plane Policing (CoPP) - This section explains how to configure and verify CoPP, which is used to protect the route processor (RP) or CPU of a router. Device Hardening: This section

Access Control Lists (ACLs)

- ACLs are sequential lists of access control entries (ACEs) that perform permit or deny packet classification, based on predefined conditional matching statements.
- Packet classification starts at the top (lowest sequence) and proceeds down (higher sequence) until a matching pattern is identified.
- When a match is found, the appropriate action (permit or deny) is taken, and processing stops.
- At the end of every ACL is an implicit deny ACE, which denies all packets that did not match earlier in the ACL.

ACLs

While different kinds of ACLs can be used for packet filtering, only the following types are covered in this chapter:

- **Numbered standard ACLs -** These ACLs define packets based solely on the source network, and they use the numbered entries 1–99 and 1300–1999.
- **Numbered extended ACLs -** These ACLs define packets based on source, destination, protocol, port, or a combination of other packet attributes, and they use the numbered entries 100–199 and 2000–2699.
- **Named ACLs -**These ACLs allow standard and extended ACLs to be given names instead of numbers.
- **Port ACLs (PACLs) -** These ACLs can use standard, extended, named, and named extended MAC ACLs to filter traffic on Layer 2 switchports.
- **VLAN ACLs (VACLs) -** These ACLs can use standard, extended, named, and named extended MAC ACLs to filter traffic on VLANs.

Wildcard Masks

ACLs use wildcard masks instead of subnet masks to classify packets that are being evaluated.

All that is required to convert a subnet mask into a wildcard mask is to subtract the subnet mask from 255.255.255.255.

```
  255 . 255 . 255 . 255
− 255 . 255 . 128 .   0   Subnet Mask
    0 .   0 . 127 . 255 Wildcard Mask
```

Applying ACLs

ACLs have no effect until they are applied to an interface. The next step after creating an ACL is to apply it to an interface.

In addition to the interface, the direction (in or out) in which the ACL needs to be applied must be specified. Cisco routers allow only one inbound ACL and one outbound ACL per interface.

ACLs can also be used for various other services in addition to applying to interfaces, such as route maps, class maps, NAT, SNMP, virtual terminal (vty) lines, or traffic-classification techniques.

Numbered ACLs

The process for defining a numbered standard ACL for IOS nodes is as follows:

Step 1. Define the ACL by using the command **access-list-number** { **deny** | **permit** } *source* [*source-wildcard*] [**log**]. The ACL number can be 1–99 or 1300–1999.

Step 2. Apply the ACL to an interface by using the command **ip access-group** {*acl-number*} {**in**|**out**} under interface configuration mode.

The keywords **any** and **host** can be used as abbreviations for *source* [*source-wildcard*]. Using the keyword any is the equivalent to specifying 0.0.0.0 255.255.255.255, which matches all packets.

The keyword **host** is used to match a specific host. It is the equivalent to having specified a host IP address followed by a wildcard mask of 0.0.0.0. The source and source-wildcard reflect a matching pattern for the network prefix that is being matched.

Numbered ACLs (Cont.)

Example 26-1 demonstrates how a numbered standard ACL is created and applied to an interface to deny traffic from the 172.16.0.0/24 subnet and from host 192.168.1.1/32 while allowing all other traffic coming into interface Gi0/1.

Example 26-1 *Creating and Applying a Numbered Standard ACL*

```
R1(config)# access-list 1 deny 172.16.0.0 0.0.255.255
R1(config)# access-list 1 deny host 192.168.1.1
R1(config)# access-list 1 permit any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 1 in
```

Notice that the last ACE in the ACL explicitly permits all traffic (permit any). If this ACE is not included, all traffic will be dropped because of the implicit deny (deny any) at the end of every ACL.

ACE Entry	Networks
Permit any	Permits all networks
permit 172.16.0.0 0.0.255.255	Permits all networks in the 172.16.0.0/16 range
permit host 192.168.1.1	Permits only the 192.168.1.1/32 network

Numbered Extended ACLs

The process for defining a numbered extended ACL is as follows:

Step 1. Define the ACL by using the command **access-list** *acl-number* {**deny**|**permit**} *protocol source source-wildcard destination destination-wildcard* [*protocol-options*] [**log** | **log-input**]. The ACL number can be 100–199 or 2000–2699.

Step 2. Apply the ACL to an interface by using the command **ip access-group** {*acl-number*} {**in**|**out**} under interface configuration mode.

As with standard ACLs, source source-wildcard and destination destination-wildcard can be defined to match a single host with the host keyword or match any subnet with the any keyword.

The *protocol-options* keyword differs based on the protocol specified in the ACE.

Numbered Extended ACLs (Cont.)

Example 26-2 demonstrates how a numbered extended ACL is created and applied to an interface to block all Telnet and ICMP traffic as well as deny all IP traffic from host 10.1.2.2 to host 10.1.2.1. Notice how Telnet's TCP port 23 is being matched with the eq keyword.

Example 26-2 *Creating and Applying Numbered Extended ACLs*

```
R1(config)# access-list 100 deny tcp any any eq 23
R1(config)# access-list 100 deny icmp any any
R1(config)# access-list 100 deny ip host 10.1.2.2 host 10.1.2.1
R1(config)# access-list 100 permit ip any any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 100 in
```

Named ACLs

Named ACLs allow for ACLs to be named, which makes administering ACLs much easier as long as proper ACL naming conventions are followed. To create and apply a named ACL, follow these steps:

Step 1. Define the ACL by using the command **ip access-list standard|extended** {acl-number | acl-name}. Entering this command places the CLI in ACL configuration mode.

Step 2. Configure the specific ACE in ACL configuration mode by using the command **[sequence] {permit | deny}** *source source-wildcard*.

Step 3. Apply the ACL to an interface by using the command **ip access-group** { acl-number | acl-name } {in|out} under interface configuration mode.

Standard and Extended Names ACLs

Example 26-3 shows how named standard and extended ACLs are created and applied to an interface.

Example 26-3 *Standard and Extended Named ACLs*

```
Named Standard ACL

R1(config)# ip access-list standard STANDARD_ACL
R1(config-std-nacl)# deny 172.16.0.0 0.0.255.255
R1(config-std-nacl)# deny host 192.168.1.1
R1(config-ext-nacl)# permit any
R1(config-ext-nacl)# exit
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group STANDARD_ACL in

Numbered Standard ACL

R1(config)# access-list 1 deny 172.16.0.0 0.0.255.255
R1(config)# access-list 1 deny host 192.168.1.1
R1(config)# access-list 1 permit any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 1 in
```

```
Named Extended ACL

R1(config)# ip access-list extended EXTENDED_ACL
R1(config-ext-nacl)# deny tcp any any eq 23
R1(config-ext-nacl)# deny icmp any any
R1(config-ext-nacl)# deny ip host 10.1.2.2 host 10.1.2.1
R1(config-ext-nacl)# permit ip any any
R1(config-ext-nacl)# exit
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group EXTENDED_ACL in

Numbered Extended ACL

R1(config)# access-list 100 deny tcp any any eq 23
R1(config)# access-list 100 deny icmp any any
R1(config)# access-list 100 deny ip host 10.1.2.2 host 10.1.2.1
R1(config)# access-list 100 permit ip any any
R1(config)# interface GigabitEthernet0/1
R1(config-if)# ip access-group 100 in
```

Port ACLs (PACLs)

Access lists applied on Layer 2 ports are called port access control lists (PACLs). PACLs can be standard, extended, or named IPv4 ACLs for Layer 3, and they can be named MAC address ACLs for Layer 2. PACLs have a few restrictions that vary from platform to platform. The following are some of the most common restrictions:

- PACLs only support filtering incoming traffic on an interface (no outbound filtering support).
- PACLs cannot filter Layer 2 control packets, such as CDP, VTP, DTP, PAgP, UDLD, and STP.
- PACLs are supported only in hardware.
- PACLs do not support ACLs to filter IPv6, ARP, or Multiprotocol Label Switching (MPLS) traffic.

Applying a PACL

An IPv4 PACL is applied to an interface with the **ip access-group** *access-list* **in** command.

Example 26-4 shows a PACL applied to a Layer 2 interface Gi0/1 to block ICMP, Telnet traffic, and host 10.1.2.2 access to host 10.1.2.1.

Example 26-4 *Applying a PACL*

```
R1(config)# ip access-list extended PACL
R1(config-ext-nacl)# deny tcp any any eq 23
R1(config-ext-nacl)# deny icmp any any
R1(config-ext-nacl)# deny ip host 10.1.2.2 host 10.1.2.1
R1(config-ext-nacl)# permit ip any any
R1(config-ext-nacl)# exit
R1(config)# interface GigabitEthernet0/1
R1(config-if)# switchport
R1(config-if)# ip access-group PACL in
```

VLAN ACLs (VACLs)

Access lists applied to VLANs are called VLAN access control lists (VACLs). VACLs can filter traffic that is bridged within a VLAN or that is routed into or out of a VLAN. To create and apply a VACL, follow these steps:

Step 1. Define a VLAN access map by using the command **vlan access-map** *name sequence*.

Step 2. Configure the match statement by using the command **match** { **ip address** { *acl-number* | *acl-name* } | **mac address** *acl-name* }.

Step 3. Configure the action statement by using the command **action forward**|**drop** [**log**]. The action statement specifies the action to be taken when a match occurs.

Step 4. Apply the VACL by using the command **vlan filter** *vlan-access-map-name vlan-list*. *vlan-list* can be a single VLAN, a range of VLANs (such as 5–30), or a comma-separated list of multiple VLANs (such as 1,2–4,6)

Creating and Applying a VACL

Example 26-5 shows a VLAN access map applied to VLAN 20 to drop ICMP and Telnet traffic and allow other traffic.

Notice that the named ACLs, ICMP and TELNET, only include ACEs with a permit statement.

This is because the ACLs are only used as matching criteria by the VLAN access maps, while the VLAN access maps are configured with the action to drop the matched traffic.

Example 26-5 *Creating and Applying a VACL*

```
SW1(config)# ip access-list extended ICMP
SW1(config-ext-nacl)# permit icmp any any
SW1(config-ext-nacl)# exit

SW1(config)# ip access-list extended TELNET
SW1(config-ext-nacl)# permit tcp any any eq 23
SW1(config-ext-nacl)# exit

SW1(config)# ip access-list extended OTHER
SW1(config-ext-nacl)# permit ip any any
SW1(config-ext-nacl)# exit

SW1(config)# vlan access-map VACL_20 10
SW1(config-access-map)# match ip address ICMP
SW1(config-access-map)# action drop
SW1(config-access-map)# exit

SW1(config)# vlan access-map VACL_20 20
SW1(config-access-map)# match ip address TELNET
SW1(config-access-map)# action drop log
SW1(config-access-map)# exit

SW1(config)# vlan access-map VACL_20 30
SW1(config-access-map)# match ip address OTHER
SW1(config-access-map)# action forward

SW1(config)# vlan filter VACL_20 vlan-list 20
```

PACL, VACL, and PACL Interaction

When a PACL, a VACL, and a RACL are all configured in the same VLAN, the ACLs are applied in a specific order, depending on whether the incoming traffic needs to be bridged or routed.

Bridged traffic processing order (within the same VLAN):

1. Inbound PACL on the switchport (for example, VLAN 10)
2. Inbound VACL on the VLAN (for example, VLAN 10)
3. Outbound VACL on the VLAN (for example, VLAN 10)

Routed traffic processing order (across VLANs):

1. Inbound PACL on the switchport (for example, VLAN 10)
2. Inbound VACL on the VLAN (for example, VLAN 10)
3. Inbound ACL on the SVI (for example, SVI 10)
4. Outbound ACL on the SVI (for example, SVI 20)
5. Outbound VACL on the VLAN (for example, VLAN 20)

Terminal Lines and Password Protection

- Password protection to control or restrict access to the CLI to protect the router from unauthorized remote access and unauthorized local access is the most common type of security that needs to be implemented.

Terminal Lines

There are three basic methods to gain access to the CLI of an IOS device:

- **Console port (cty) line -** On any IOS device, this appears in configuration as line con 0 and in the output of the command show line as cty. The console port is mainly used for local system access using a console terminal.

- **Auxiliary port (aux) line -** This appears in the configuration as line aux 0. The aux port is mainly used for remote access into the device through a modem.

- **Virtual terminal (vty) lines -** These lines are displayed by default in the configuration as line vty 0 4. They are used solely for remote Telnet and SSH connections. They are virtual because they are logical lines with no physical interface associated to them.

Password Protection

Each of these types of terminal lines should be password protected. There are three ways to add password protection to the lines:

- Using a password configured directly on the line (not recommended)
- Using username-based authentication (recommended as a fallback)
- Using an AAA server: Highly recommended and covered later in this chapter, in the section “Authentication, Authorization, and Accounting (AAA)”

Password Types

There are five available password types in Cisco IOS:

- **Type 0 passwords -** These passwords are the most insecure because they are not encrypted and are visible in the device configuration in plaintext. The command **enable password** is an example of a command that uses a type 0 password.
- **Type 5 passwords -** These passwords use an improved Cisco proprietary encryption algorithm that makes use of the MD5 hashing algorithm. The command **enable secret** specifies an additional layer of security over the command enable password.
- **Type 7 passwords -** These passwords use a Cisco proprietary Vigenere cypher encryption algorithm and are known to be weak. Type 7 encryption is enabled by the command **service password-encryption.**
- **Type 8 passwords -** Type 8 passwords specify a Password-Based Key Derivation Function 2 (PBKDF2) with a SHA-256 hashed secret and are considered to be uncrackable.
- **Type 9 passwords -** These use the SCRYPT hashing algorithm. Just like type 8 passwords, they are considered to be uncrackable.

Password Encryption

The **service password-encryption** command in global configuration mode is used to encrypt type 0 passwords in the configuration (for example, BGP passwords) or over a plaintext session such as Telnet in an effort to prevent unauthorized users from viewing the password.

Passwords configured prior to configuring the command **service password-encryption** are not encrypted and must be reentered into the configuration. Password encryption is applied to all type 0 passwords, including authentication key passwords; cty, aux, and vty line passwords; and BGP neighbor passwords.

Unfortunately, the command **service password-encryption** encrypts passwords with type 7 encryption, which is easily reversible.

Username and Password Authentication

Username accounts can be used for several applications, such as console, aux, and vty lines. To establish a username and password login authentication system, you can create usernames on a device for all device users or groups. There are three different ways to configure a username on IOS:

- Using the command **username** {*username*} **password** {*password*} configures a plaintext password (type 0).
- Using the command **username** {*username*} **secret** {*password*} provides type 5 encryption.
- Using the command **username** {*username*} **algorithm-type** {**md5** | **sha256** | **scrypt**} **secret** {*password*} provides type 5, type 8, or type 9 encryption, respectively.

The third command is recommended because it allows for the highest level of password encryption (type 8 and type 9).

Configuring Line Local Password Authentication

To enable password authentication on a line, the following two commands are required under line configuration mode:

- **password** *password* to configure the password
- **login** to enable password checking at login

In Example 26-7, a password is configured for all users attempting to connect to the cty, vty, and aux lines.

Example 26-7 *vty, cty, and aux Lines with Password-Based Authentication*

```
R1# show running-config | section line
Building configuration...

line con 0
password My.C0n5ole.P@s5
login
line aux 0
password My.AuX.P@s5
login
line vty 0 4
password My.vTy.P@s5
login
!
end
```

Verifying Line Local Password Authentication

Example 26-8 shows an example in which the console line password is being tested.

All that is required to test the password is to log off the console and log back in again using the configured console password.

Example 26-8 *Console Password Test*

```
R1# exit

Router con0 is now available

Press RETURN to get started.

User Access Verification
Password:

! Password entered here is not displayed by the router


Router>
```

Configuring Line Local Username & Password Authentication

To enable username and password authentication, the following two commands are required:

- The command **username** in global configuration mode (using one of the options shown in the “Username and Password Authentication” section, earlier in this chapter).
- The command **login local** under line configuration mode to enable username-based authentication at login.

Example 26-9 shows three usernames (type0, type5, and type9) with different password encryptions each that are allowed to log in to the device

Example 26-9 *Local Username-Based Authentication for a vty Line*

```
R1# show running-config
Building configuration...
!
! Output Omitted for Brevity

username type0 password 0 weak

username type5 secret 5 $1$b1Ju$kZbBS1Pyh4QzwXyZ1kSZ2/

username type9 secret 9 $9$vFpMf8elb4RVV8$seZ/bDAx1uV4yH75Z/nwUuegLJDVCc4UXOAE83JgsOc
!
! Output Omitted for Brevity

line con 0
 login local

line aux 0
 login local

line vty 0 4
 login local
!
end
```

Verifying Line Local Username & Password Authentication

Example 26-10 shows user type5 establishing a Telnet session from R2 into R1 using username-based authentication.

Example 26-10 *Verifying Local Username-Based Authentication for vty Lines*

```
! Telnet session initiated from R2 into R1

R2# telnet 10.1.12.1
Trying 10.1.12.1 ... Open


User Access Verification


Username: type5
Password:

! Password entered is not displayed by the router


R1>
```

Privilege Levels & Role-Based Access Control (RBAC)

The Cisco IOS CLI by default includes three privilege levels, each of which defines what commands are available to a user:

- **Privilege level 0 -** Includes the **disable**, **enable**, **exit**, **help**, and **logout** commands.

- **Privilege level 1: -**Also known as User EXEC mode. The command prompt in this mode includes a greater-than sign (R1>). From this mode it is not possible to make configuration changes; in other words, the command **configure terminal** is not available.

- **Privilege level 15 -** Also known as Privileged EXEC mode. This is the highest privilege level, where all CLI commands are available. The command prompt in this mode includes a hash sign (R1#).

The global configuration command **privilege** {*mode*} **level** {*level*} {*command string*} is used to change or set a privilege level for a command to any of these levels.

Configuring a Username with Privilege Level

Example 26-11 shows a configuration where the user noc is created along with the type 9 (scrypt) secret password cisco123.

Notice that the privilege level is also configured to level 5 as part of the username command.

Level 5 allows the user to go into any interface on the router and shut it down, unshut it, and configure an IP address on it.

Example 26-11 *Configuring a Username with Privilege Level*

```
R1(config)# username noc privilege 5 algorithm-type scrypt secret cisco123
R1(config)# privilege exec level 5 configure terminal
R1(config)# privilege configure level 5 interface
R1(config)# privilege interface level 5 shutdown
R1(config)# privilege interface level 5 no shutdown
R1(config)# privilege interface level 5 ip address
```

Verifying Privilege Levels

The example shows a quick test to verify that the only commands allowed for privilege level 5 users are those specified by the privilege level command.

```
R1# telnet 1.2.3.4
Trying 1.2.3.4 ... Open

User Access Verification

Username: noc
Password: cisco123

R1# show privilege
Current privilege level is 5
R1#
R1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)# interface gigabitEthernet 0/1
R1(config-if)# ?
Interface configuration commands:
  default   Set a command to its defaults
  exit      Exit from interface configuration mode
  help      Description of the interactive help system
  ip        Interface Internet Protocol config commands
  no        Negate a command or set its defaults
  shutdown  Shutdown the selected interface

R1 (config-if)# ip ?
Interface IP configuration subcommands:
  address  Set the IP address of an interface
```

```
R1(config-if)# ip address 10.1.1.1 255.255.255.0
R1(config-if)# no ?
  ip        Interface Internet Protocol config commands
  shutdown  Shutdown the selected interface

R1(config-if)# no shutdown
R1(config-if)#
*Apr 27 18:14:23.749: %LINK-3-UPDOWN: Interface GigabitEthernet0/1, changed state to up
*Apr 27 18:14:24.750: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1,
  changed state to up
R1(config-if)#
R1(config-if)# shutdown
R1(config-if)# end
*Apr 27 18:14:38.336: %LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state
  to administratively down
*Apr 27 18:14:39.336: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1,
  changed state to down
R1#
*Apr 27 18:14:40.043: %SYS-5-CONFIG_I: Configured from console by noc on vty0
  (1.2.3.4)
R1#
```

Controlling Access to vty Lines with ACLs

Access to the vty lines of an IOS device can be further secured by applying inbound ACLs on them, allowing access only from a restricted set of IP addresses. Outbound vty connections from an IOS device can also be controlled by applying outbound ACLs to vtys.

To apply a standard or an extended access list to a vty line, use the command **access-class** {*access-list-number*|*access-list-name*} {**in**|**out**} under line configuration mode. The **in** keyword applies an inbound ACL, and the **out** keyword applies an outbound ACL.

Verifying Access to vty Lines with ACLs

Example 26-13 demonstrates R1 using Telnet to get into R2 before and after applying an ACL to R2's vty line. R1 is configured with IP address 10.12.1.1 and R2 with 10.12.1.2. The ACL being applied to R2's vty line is meant to block vty access into it from R1.

Example 26-13 *Verifying Access to vty Lines with ACLs*

```
! Prior to applying an ACL to R2's vty line, R1 is allowed to telnet into R2


R1# telnet 10.12.1.2
Trying 10.12.1.2... Open

User Access Verification

Username: noc
Password:

R2#
R2# exit
[Connection to 10.12.1.2 closed by foreign host]

! Access list to deny R1's IP address is created and applied to the vty lines 0 to 4

R2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)# access-list 1 deny 10.12.1.1
R2(config)# access-list 1 permit any
R2 (config)# line vty 0 4
R2(config-line)# access-class 1 in
R2(config-line)# end
R2#
R2# show running-config | section line vty
line vty 0 4
 access-class 1 in
 login local
R2#
*Apr 27 19:49:45.599: %SYS-5-CONFIG_I: Configured from console by console

! After applying an ACL to R2's vty line, R1 is not allowed to telnet into R2

R1# telnet 10.12.1.2
Trying 10.12.1.2 ...
% Connection refused by remote host

R1#
```

Controlling Access to vty Lines Using Transport Input

Another way to further control what type of protocols are allowed to access the vty lines is to use the command **transport input {all | none | telnet | ssh}** under line configuration mode.

Table 26-3 includes a description for each of the transport input command keywords.

Keyword	Description
All	Allows Telnet and SSH
None	Blocks Telnet and SSH
telnet	Allows Telnet only
ssh	Allows SSH only
telnet ssh	Allows Telnet and SSH

Table 26-3 Transport Input Command Keyword Description

Controlling Access to vty Lines Using Transport Input

Example 26-14 shows the vty lines from 0 to 4 configured with different **transport input** command keywords.

Keep in mind that vty lines are evaluated from the top (vty 0) onward, and each vty line accepts only one user.

Example 26-14 *vty Lines with Different* transport input *Keywords*

```
line vty 0
 login local
 transport input all
line vty 1
 login local
 transport input none
line vty 2
 login local
 transport input telnet
line vty 3
 login local
 transport input ssh
line vty 4
 login local
 transport input telnet ssh
```

Verifying Access to vty Lines Using Transport Input

Example 26-15 demonstrates how Telnet sessions are assigned to different vty lines on R1.

Example 26-15 *Verifying Access to vty Lines*

```
! An asterisk to the left of the row indicates the line is in use
! The output below shows a user is connected into the console (cty)

R1# show line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX    9600/9600  -    -      -    -    -      0       0     0/0       -
   578 VTY              -    -      -    -    -      1       0     0/0       -
   579 VTY              -    -      -    -    -      0       0     0/0       -
   580 VTY              -    -      -    -    -      0       0     0/0       -
   581 VTY              -    -      -    -    -      0       0     0/0       -
   582 VTY              -    -      -    -    -      0       0     0/0       -
R1#

! Telnet connection from R2 into R1 is established

R2# telnet 10.1.12.1
Trying 10.1.12.1 ... Open
```

```
User Access Verification
Username: noc
Password:
R1>

! The asterisk in the output of show line on R1 indicates the first vty 0 is now in
use.
! vty 0 is mapped to vty 578 automatically.

R1# show line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX    9600/9600  -    -      -    -    -      0       0     0/0       -
*  578 VTY              -    -      -    -    -      2       0     0/0       -
   579 VTY              -    -      -    -    -      0       0     0/0       -
   580 VTY              -    -      -    -    -      0       0     0/0       -
   581 VTY              -    -      -    -    -      0       0     0/0       -
   582 VTY              -    -      -    -    -      0       0     0/0       -
R1#
```

Verifying Access to vty Lines Using Transport Input (Cont.)

```
! Telnet connection from R3 into R1 is established

R3# telnet 10.1.13.1
Trying 10.1.13.1 ... Open

User Access Verification
Username: noc
Password:
R1>

! The output of show line on R1 indicates the vty 0 and vty 2 are now in use
! vty 2 is mapped to vty 580

R1# show line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX    9600/9600  -    -      -    -    -      0       0     0/0       -
*  578 VTY              -    -      -    -    -      2       0     0/0       -
   579 VTY              -    -      -    -    -      0       0     0/0       -
*  580 VTY              -    -      -    -    -      1       0     0/0       -
   581 VTY              -    -      -    -    -      0       0     0/0       -
   582 VTY              -    -      -    -    -      0       0     0/0       -
```

```
R1#

! Telnet connection from R4 into R1 is established

R4# telnet 10.1.14.1
Trying 10.1.14.1 ... Open

User Access Verification
Username: noc
Password:
R1>

! The output of show line on R1 indicates the vty 0, vty 2 and vty 4 are now in use
! vty 4 is mapped to vty 582. This leaves no more vty lines available for telnet

R1# show line
   Tty Typ     Tx/Rx    A Modem  Roty AccO AccI   Uses   Noise  Overruns   Int
*    0 CTY              -    -      -    -    -      0       0     0/0       -
     1 AUX    9600/9600  -    -      -    -    -      0       0     0/0       -
*  578 VTY              -    -      -    -    -      2       0     0/0       -
   579 VTY              -    -      -    -    -      0       0     0/0       -
*  580 VTY              -    -      -    -    -      1       0     0/0       -
   581 VTY              -    -      -    -    -      0       0     0/0       -
*  582 VTY              -    -      -    -    -      1       0     0/0       -


! Trying to telnet into R1 from R5 will fail since there are no more vtys available
for telnet

R5# telnet 10.1.15.1
Trying 10.1.15.1 ...
% Connection refused by remote host

R5#
```

Enabling SSH vty Access

Telnet session packets are sent in plaintext, and this makes it very easy to sniff and capture session information. A more reliable and secure method for device administration is to use the Secure Shell (SSH) protocol. SSH, which provides secure encryption and strong authentication, is available in two versions:

- **SSH Version 1 (SSHv1) -** This is an improvement over using plaintext Telnet, but some fundamental flaws exist in its implementation, so it should be avoided in favor of SSHv2.

- **SSH Version 2 (SSHv2) -** This is a complete rework and stronger version of SSH that is not compatible with SSHv1. SSHv2 has many benefits and closes a security hole that is found in SSH version 1. SSH version 2 is certified under the National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS) 140-1 and 140-2 U.S. cryptographic standards and should be used where feasible.

Configuring vty Access Using SSH

The steps needed to configure SSH on an IOS device are as follows:

Step 1. Configure a hostname other than Router by using the command hostname {hostname name}.

Step 2. Configure a domain name by using the command **ip domain-name** {*domain-name*}.

Step 3. Generate crypto keys by using the command **crypto key generate rsa**. When entering this command, you are prompted to enter a modulus length. The longer the modulus, the stronger the security. However, a longer modulus takes longer to generate. The modulus length needs to be at least 768 bits for SSHv2.

Example 26-16 *Configuring vty Access Using SSH*

```
R1(config)# hostname R1
R1(config)# username cisco secret cisco
R1(config)# ip domain-name cisco.com
R1(config)# crypto key generate rsa
The name for the keys will be: R1.cisco.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 768
% Generating 768 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 1 seconds)

R1(config)#
*May  8 20:44:48.319: %SSH-5-ENABLED: SSH 1.99 has been enabled
R1(config)#
R1(config)# line vty 0 4
R1(config-line)# login local
R1(config-line)# end
R1#
```

To force the IOS SSH server to disable SSHv1 and accept only SSHv2 connections, enter the command **ip ssh version 2** under global configuration mode.

Aux Port and Exec Timeout

Some devices have an auxiliary (aux) port available for remote administration through a dialup modem connection. In most cases, the aux port should be disabled by using the command no exec under line aux 0.

By default, an idle EXEC session is not terminated, which poses an enormous security risk. The command **exec-timeout** {*minutes*}{*seconds*} under line configuration mode can be used to disconnect idle user sessions. The default setting is 10 minutes.

Example 26-17 shows a configuration in which the exec-timeout for the console line is configured to time out after 5 minutes of inactivity and 2 minutes and 30 seconds for the vty lines.

Example 26-17 *Configuring EXEC Timeout*

```
line con 0
 exec-timeout 5 0
line vty 0 4
 exec-timeout 2 30
```

Absolute Timeout

The command **absolute-timeout** {*minutes*} under line configuration mode terminates an EXEC session after the specified timeout period has expired, even if the connection is being used at the time of termination.

It is recommended to use it in combination with the command **logout-warning** {seconds} under line configuration mode to display a “line termination” warning to users about an impending forced timeout.

Example 26-18 shows the commands **absolute-timeout** and l**ogout-warning** configured on the vty lines.

Example 26-18 *Configuring Absolute Timeout*

```
line vty 4
 exec-timeout 2 0
 absolute-timeout 10
 logout-warning 20
```

Authentication, Authorization, and Accounting (AAA)

AAA is an architectural framework for enabling a set of three independent security functions:

- Authentication
- Authorization
- Accounting

AAA is commonly used in the networking industry for the following two use cases:

- Network device access control
- Secure network access control

AAA Framework

AAA is an architectural framework for enabling a set of three independent security functions:

- **Authentication -** Enables a user to be identified and verified prior to being granted access to a network device and/or network services.

- **Authorization -** Defines the access privileges and restrictions to be enforced for an authenticated user.

- **Accounting -** Provides the ability to track and log user access, including user identities, start and stop times, executed commands (that is, CLI commands), and so on. In other words, it maintains a security log of events.

There are many AAA protocols available, but the two most popular ones are Remote Authentication Dial-In User Service (RADIUS) and Terminal Access Controller Access-Control System Plus (TACACS+).

AAA Use Cases

AAA is commonly used in the networking industry for the following two use cases:

- **Network device access control -** As described earlier in this chapter, Cisco IOS provides local features for simple device access control, such as local username-based authentication and line password authentication. However, these features do not provide the same degree of access control and scalability that is possible with AAA. For this reason, AAA is the recommended method for access control. TACACS+ is the protocol of choice for network device access control.

- **Secure network access control -** AAA can be used to obtain the identity of a device or user before that device or user is allowed to access to the network. RADIUS is the preferred protocol for secure network access.

TACACS+

Cisco developed TACACS+ and released it as an open standard in the early 1990s. Although TACACS+ is mainly used for AAA device access control, it is possible to use it for some types of AAA network access.

The TACACS+ protocol uses Transmission Control Protocol (TCP) port 49 for communication between the TACACS+ clients and the TACACS+ server.

Figure 26-1 shows an end user who can access a Cisco switch using Telnet, SSH, or the console. The Cisco switch is acting as a TACACS+ client that communicates with the TACACS+ server using the TACACS+ protocol.

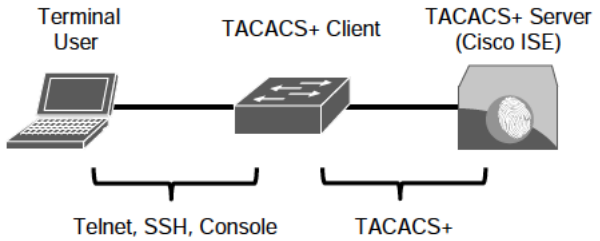

Figure 26-1 *TACACS+ Client/Server Communication*

RADIUS

One of the key differentiators of TACACS+ is its capability to separate authentication, authorization, and accounting into independent functions. This is why TACACS+ is so commonly used for device administration instead of RADIUS, even though RADIUS is capable of providing network device access control.

RADIUS is an IETF standard AAA protocol. RADIUS is the AAA protocol of choice for secure network access. The reason for this is that RADIUS is the AAA transport protocol for Extensible Authentication Protocol (EAP), while TACACS+ does not support this functionality.

Another major difference between TACACS+ and RADIUS is that RADIUS needs to return all authorization parameters in a single reply, while TACACS+ can request authorization parameters separately and multiple times throughout a session.

RADIUS & TACACS+ Comparison

Table 26-4 provides a summary comparison of RADIUS and TACACS+.

Component	RADIUS	TACACS+
Protocol and port(s) used	Cisco's implementation: • UDP: port 1645 (authentication and authorization) • UDP: port 1646 (accounting) Industry standard: • UDP: port 1812 (authentication and authorization) • UDP: port 1813 (accounting)	TCP: port 49
Encryption	• Encrypts only the password field • Supports EAP for 802.1x authentication	• Encrypts the entire payload • Does not support EAP
Authentication and authorization	• Combines authentication and authorization • Cannot be used to authorize which CLI commands can be executed individually	• Separates authentication and authorization • Can be used for CLI command authorization
Accounting	Does not support network device CLI command accounting	Supports network device CLI command accounting
Primary Use	Secure network access	Network device access control

Configuring AAA for Network Device Access Control

There are two parts to configuring TACACS+:

- The configuration of the device itself
- The configuration of the TACACS+ AAA server (for example, Cisco ISE)

The following steps are for configuring an IOS device with TACACS+ for device access control. Configuration for the TACACS+ server is not included here because it is beyond the scope of this book:

Step 1. Create a local user with full privilege for fallback or to avoid being locked out after enabling AAA by using the command
username {*username*} **privilege 15 algorithm-type** {**md5** | **sha256** | **scrypt**} **secret** {*password*}

Configuring AAA for Network Device Access Control (Cont.)

Step 2. Enable AAA functions on by using with the command **aaa new-model**.

Step 3. Add a TACACS+ server using one of these methods, depending on the IOS version:

- To add a TACACS+ server on IOS versions prior to 15.x, use the command
  tacacs-server host { *hostname* | *host-ip-address* } **key** *key-string*
- To add a TACACS+ server on IOS versions 15.x and later, use the following commands:
  tacacs server *name*
  address ipv4 { *hostname* | *host-ip-address* }
  key *key-string*

Step 4. Create an AAA group by using the following commands**:**

aaa group server tacacs+ *group-name*
server name *server-name.*

This creates an AAA group that includes the TACACS+ servers that are added to the group with the **server name** command.

Configuring AAA for Network Device Access Control (Cont.)

Step 5. Enable AAA login authentication by using the command **aaa authentication login** { **default** | *custom-list-name* } *method1* [*method2* . . .] Method lists enable login authentication. The default keyword applies the method lists that follow (method1 [method2 . . .) to all lines (cty, tty, aux, and so on). The *custom list-name* CLI assigns a custom name for the method lists that follow it. To apply a custom list to a line, use the command **login authentication** *custom-list-name* under the line configuration mode. Method lists are applied sequentially from left to right.

Step 6. Enable AAA authorization for EXEC by using the command **aaa authorization exec** { **default** | *custom-list-name* } *method1* [*method2* . . .] This command enables EXEC shell authorization for all lines except the console line.

Step 7. Enable AAA authorization for the console by using the command **aaa authorization console**. Authorization for the console is disabled by default to prevent inexperienced users from locking themselves out.

Configuring AAA for Network Device Access Control (Cont.)

Step 8. Enable AAA command authorization by using the command **aaa authorization commands** {*privilege level*} { **default** | *custom-list-name* } *method1* [*method2* . . .]

Step 9. Enable command authorization in global configuration mode (and all global configuration submodes) by using the command **aaa authorization config-commands**

Step 10. Enable login accounting by using the command **aaa accounting exec** { **default** | *custom-list-name* } *method1* [*method2* . . .]

Step 11. Enable command accounting by using the command **aaa accounting commands** {*privilege level*} { **default** | *custom-list-name* } *method1* [*method2* . . .]

AAA Configuration For Device Control Example

Example 26-19 shows a common AAA IOS configuration for device access control.

Example 26-19 *Common AAA Configuration for Device Access Control*

```
aaa new-model

tacacs server ISE-PRIMARY
  address 10.10.10.1
  key my.S3cR3t.k3y

tacacs server ISE-SECONDARY
  address 20.20.20.1
  key my.S3cR3t.k3y

aaa group server tacacs+ ISE-TACACS+
  server name ise-primary
  server name ise-secondary

aaa authentication login default group ISE-TACACS+ local
aaa authentication login CONSOLE-CUSTOM-AUTHENTICATION-LIST local line enable
aaa authentication enable default group ISE-TACACS+ enable
aaa authorization exec default group ISE-TACACS+ if-authenticated
aaa authorization exec CONSOLE-CUSTOM-EXEC-AUTHORIZATION-LIST none
aaa authorization commands 0 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST none
aaa authorization commands 1 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST none
aaa authorization commands 15 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST none
aaa authorization commands 0 default group ISE-TACACS+ if-authenticated
aaa authorization commands 1 default group ISE-TACACS+ if-authenticated
aaa authorization commands 15 default group ISE-TACACS+ if-authenticated
aaa authorization console
aaa authorization config-commands
aaa accounting exec default start-stop group ISE-TACACS+
aaa accounting commands 0 default start-stop group ISE-TACACS+
aaa accounting commands 1 default start-stop group ISE-TACACS+
aaa accounting commands 15 default start-stop group ISE-TACACS+

line con 0
authorization commands 0 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST
authorization commands 1 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST
authorization commands 15 CONSOLE-CUSTOM-COMMAND-AUTHORIZATION-LIST
authorization exec CONSOLE-CUSTOM-EXEC-AUTHORIZATION-LIST
privilege level 15
login authentication CONSOLE-CUSTOM-AUTHENTICATION-LIST

line vty 0 4
<uses default method-lists for AAA>
```

Apart from the IOS configuration, the AAA server also needs to be configured with the AAA client information (hostname, IP address, and key), the login credentials for the users, and the commands the users are authorized to execute on the device.

Verifying AAA Configuration

Example 26-20 demonstrates SSH sessions being initiated from R2 into R1, using the netadmin and netops accounts.

The netadmin account was configured in the AAA server with privilege 15, and netops was configured with privilege 1. The netadmin account has access to the full set of commands, while netops is very limited.

Example 26-20 *Verifying AAA Configuration*

```
! Establish SSH session from R2 into R1 using netadmin account
R2# ssh netadmin@10.12.1.1
Password:
R1# show privilege
Current privilege level is 15
R1#
R1# configure terminal
R1(config)#

! Establish SSH session from R2 into R1 using netops account

R2# ssh netops@192.168.1.26
Password:
R1> show privilege
Current privilege level is 1
R1> show version
Cisco IOS Software, IOSv Software (VIOS-ADVENTERPRISEK9-M), Version 15.6(3)M2,
RELEASE SOFTWARE (fc2)
! Output Omitted for Brevity

R1> show running-config
Command authorization failed.

R1> enable
Command authorization failed.
```

Zone-Based Firewall (ZBFW)

- Cisco Zone-Based Firewall (ZBFW) is the latest integrated stateful firewall technology included in IOS.
- ZBFW uses a flexible and straightforward approach to providing security by establishing security zones.
- A zone establishes a security border on the network and defines acceptable traffic that is allowed to pass between zones.
- By default, interfaces in the same security zone can communicate freely with each other, but interfaces in different zones cannot communicate with each other.

The Self & Default Zones

The self zone is a system-level zone and includes all the routers' IP addresses. By default, traffic to and from this zone is permitted to support management (for example, SSH protocol, SNMP) and control plane (for example, EIGRP, BGP) functions. After a policy is applied to the self zone and another security zone, interzone communication must be explicitly defined.

The default zone is a system-level zone, and any interface that is not a member of another security zone is placed in this zone automatically. Upon initialization of this zone, any interface not associated to a security zone is placed in this zone. When the unassigned interfaces are in the default zone, a policy map can be created between the two security zones.

ZMFW Configuration

ZBFW is configured in five steps:

Step 1. Configure the security zones by using the command **zone security** *zone-name*. A zone needs to be created for the outside zone (the Internet). The self zone is defined automatically.

Example 26-21 demonstrates the configuration of a security zone.

Example 26-21 *Defining the Outside Security Zone*

```
Zone security OUTSIDE
 description OUTSIDE Zone used for Internet Interface
```

ZMFW Configuration (Cont.)

Step 2. Define the inspection class map. The class map for inspection defines a method for classification of traffic. The class map is configured using the command **classmap type inspect** [**match-all** | **match-any**] *class-name*.

Example 26-22 shows a sample configuration of inspection class maps and their associated ACLs.

Example 26-22 *Inspecting the Class Map Configuration*

```
ip access-list extended ACL-IPSEC
 permit udp any any eq non500-isakmp
 permit udp any any eq isakmp
ip access-list extended ACL-PING-AND-TRACEROUTE
 permit icmp any any echo
 permit icmp any any echo-reply
 permit icmp any any ttl-exceeded
 permit icmp any any port-unreachable
 permit udp any any range 33434 33463 ttl eq 1
ip access-list extended ACL-ESP
 permit esp any any
ip access-list extended ACL-DHCP-IN
 permit udp any eq bootps any eq bootpc
ip access-list extended ACL-GRE
 permit gre any any
!
class-map type inspect match-any CLASS-OUTSIDE-TO-SELF-INSPECT
 match access-group name ACL-IPSEC
 match access-group name ACL-PING-AND-TRACEROUTE
class-map type inspect match-any CLASS-OUTSIDE-TO-SELF-PASS
 match access-group name ACL-ESP
 match access-group name ACL-DHCP-IN
 match access-group name ACL-GRE
```

ZMFW Configuration (Cont.)

The configuration of inspect class maps can be verified with the command **show class-map type inspect** [*class-name*], as shown in Example 26-23.

Example 26-23 *Verifying the Inspect Class Map Configuration*

```
R1# show class-map type inspect
 Class Map type inspect match-any CLASS-OUTSIDE-TO-SELF-PASS (id 2)
   Match access-group name ACL-ESP
   Match access-group name ACL-DHCP-IN
   Match access-group name ACL-GRE

 Class Map type inspect match-any CLASS-OUTSIDE-TO-SELF-INSPECT (id 1)
   Match access-group name ACL-IPSEC
   Match access-group name ACL-PING-AND-TRACEROUTE
```

ZMFW Configuration (Cont.)

Step 3. Define the inspection policy map, which applies firewall policy actions to the class maps defined in the policy map.

The policy map is then associated to a zone pair.

The inspection policy map is defined with the command **policy-map type inspect** *policy-name*.

After the policy map is defined, the various class maps are defined with the command **class type inspect** *class-name*.

ZMFW Configuration (Cont.)

Under the class map, the firewall action is defined with these commands:

- **drop** [log]: This default action silently discards packets that match the class map. The log keyword adds syslog information that includes source and destination information (IP address, port, and protocol).

- **pass** [log]: This action makes the router forward packets from the source zone to the destination zone. Packets are forwarded in only one direction. A policy must be applied for traffic to be forwarded in the opposite direction. The pass action is useful for protocols like IPsec, Encapsulating Security Payload (ESP), and other inherently secure protocols with predictable behavior. The optional log keyword adds syslog information that includes the source and destination information.

- **inspect**: The inspect action offers state-based traffic control. The router maintains connection/session information and permits return traffic from the destination zone without

ZMFW Configuration (Cont.)

Example 26-24 demonstrates the configuration of the inspect policy map. Notice that in the class default class, the **drop** command does not include the log keyword because of the potential to fill up the syslog.

The inspection policy map can be verified with the command **show policy-map type inspect** [*policy-name*], as shown in Example 26-25.

Example 26-24 *Configuring the Inspection Policy Map*

```
policy-map type inspect POLICY-OUTSIDE-TO-SELF
 class type inspect CLASS-OUTSIDE-TO-SELF-INSPECT
   inspect
 class type inspect CLASS-OUTSIDE-TO-SELF-PASS
   pass
 class class-default
   drop
```

Example 26-25 *Verifying the Inspection Policy Map*

```
R1# show policy-map type inspect
  Policy Map type inspect POLICY-OUTSIDE-TO-SELF
    Class CLASS-OUTSIDE-TO-SELF-INSPECT
      Inspect
    Class CLASS-OUTSIDE-TO-SELF-PASS
      Pass
    Class class-default
      Drop
```

ZMFW Configuration (Cont.)

Step 4. Apply a policy map to a traffic flow source to a destination by using the **command zone-pair security** *zone-pair-name* **source** *source-zone-name* **destination** *destination-zone-name*. The inspection policy map is then applied to the zone pair with the command **service-policy type inspect** *policy-name*. Traffic is statefully inspected between the source and destination, and return traffic is allowed.

Example 26-26 defines the zone pairs and associates the policy map to the zone pair.

Example 26-26 *Configuring the ZBFW Zone Pair*

```
zone-pair security OUTSIDE-TO-SELF source OUTSIDE destination self
  service-policy type inspect POLICY-OUTSIDE-TO-SELF
```

ZMFW Configuration (Cont.)

Step 5. Apply the security zones to the appropriate interfaces. An interface is assigned to the appropriate zone by entering the interface configuration submode with the command interface interface-id and associating the interface to the correct zone with the command zone-member security zone-name, as defined in step 1.

Example 26-27 demonstrates the outside security zone being associated to the Internet-facing interface GigabitEthernet 0/2.

Example 26-27 *Applying the Security Zone to the Interface*

```
interface GigabitEthernet 0/2
  zone-member security OUTSIDE
```

Verifying the Outside-to-Self Policy

Now that the outside-to-self policy has been fully defined, traffic statistics can be viewed with the command **show policy-map type inspect zone-pair** [*zone-pair-name*].

Example 26-28 demonstrates the verification of the configured ZBFW policy.

Example 26-28 *Verifying the Outside-to-Self Policy*

```
R1# show policy-map type inspect zone-pair

policy exists on zp OUTSIDE-TO-SELF
  Zone-pair: OUTSIDE-TO-SELF

  Service-policy inspect : POLICY-OUTSIDE-TO-SELF

    Class-map: CLASS-OUTSIDE-TO-SELF-INSPECT (match-any)
      Match: access-group name ACL-IPSEC
        2 packets, 208 bytes
        30 second rate 0 bps
      Match: access-group name ACL-PING-AND-TRACEROUTE
        0 packets, 0 bytes
        30 second rate 0 bps
      Inspect
        Packet inspection statistics [process switch:fast switch]
        udp packets: [4:8]

        Session creations since subsystem startup or last reset 2
        Current session counts (estab/half-open/terminating) [0:0:0]
        Maxever session counts (estab/half-open/terminating) [2:1:0]
        Last session created 00:03:39
        Last statistic reset never
        Last session creation rate 0
        Maxever session creation rate 2
        Last half-open session total 0
        TCP reassembly statistics
        received 0 packets out-of-order; dropped 0
        peak memory usage 0 KB; current usage: 0 KB
        peak queue length 0

    Class-map: CLASS-OUTSIDE-TO-SELF-PASS (match-any)
      Match: access-group name ACL-ESP
        186 packets, 22552 bytes
        30 second rate 0 bps
      Match: access-group name ACL-DHCP-IN
        1 packets, 308 bytes
        30 second rate 0 bps
      Match: access-group name ACL-GRE
        0 packets, 0 bytes
        30 second rate 0 bps
      Pass
        187 packets, 22860 bytes

    Class-map: class-default (match-any)
      Match: any
      Drop
        30 packets, 720 bytes
```

ACL Counters from the Inspect Class Maps

Even though the ACLs are not used for blocking traffic, the counters do increase as packets match the ACL entries for the inspect class maps, as demonstrated in Example 26-29.

Example 26-29 *ACL Counters from the Inspect Class Maps*

```
R1# show ip access
Extended IP access list ACL-DHCP-IN
    10 permit udp any eq bootps any eq bootpc (1 match)
Extended IP access list ACL-ESP
    10 permit esp any any (170 matches)
Extended IP access list ACL-GRE
    10 permit gre any any
Extended IP access list ACL-IPSEC
    10 permit udp any any eq non500-isakmp
    20 permit udp any any eq isakmp (2 matches)
Extended IP access list ACL-PING-AND-TRACEROUTE
    10 permit icmp any any echo
    20 permit icmp any any echo-reply
    30 permit icmp any any ttl-exceeded
    40 permit icmp any any port-unreachable
    50 permit udp any any range 33434 33463 ttl eq 1
```

Verifying ZBFW

After the outside-to-self policy has been defined, it is time to verify connectivity to the internet, as shown in Example 26-30. Notice here that a simple ping from R1 to one of Google's Public DNS IP addresses 8.8.8.8 is failing.

Example 26-30 *Verifying Outside Connectivity*

```
R1# ping 8.8.8.8
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 8.8.8.8, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

Configuring the Self-to-Outside Policy

The reason for the packet failure is that the router needs to allow locally originated packets with a self-to-outside policy.

Example 26-31 demonstrates the configuration for the self-to-outside policy. ACL-IPSEC and ACL-ESP are reused from the outside-to-self policy.

Now that the second policy has been configured, R1 can successfully ping 8.8.8.8, as shown in Example 26-32.

Example 26-32 *Successful ping Test Between R1 and Google's Public DNS 8.8.8.8*

```
R31-Spoke# ping 8.8.8.8
Sending 5, 100-byte ICMP Echos to 8.8.8.8, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
```

Example 26-31 *Configuring the Self-to-Outside Policy*

```
ip access-list extended ACL-DHCP-OUT
permit udp any eq bootpc any eq bootps
!
ip access-list extended ACL-ICMP
permit icmp any any
!
class-map type inspect match-any CLASS-SELF-TO-OUTSIDE-INSPECT
 match access-group name ACL-IPSEC
 match access-group name ACL-ICMP

class-map type inspect match-any CLASS-SELF-TO-OUTSIDE-PASS
 match access-group name ACL-ESP
 match access-group name ACL-DHCP-OUT
!
policy-map type inspect POLICY-SELF-TO-OUTSIDE
  class type inspect CLASS-SELF-TO-OUTSIDE-INSPECT
   inspect
class type inspect CLASS-SELF-TO-OUTSIDE-PASS
   pass
class class-default
  drop log
!
zone-pair security SELF-TO-OUTSIDE source self destination OUTSIDE
 service-policy type inspect POLICY-SELF-TO-OUTSIDE
```

Control Plane Policing (CoPP)

- A control plane policing (CoPP) policy is a QoS policy that is applied to traffic to or sourced by the router's control plane CPU.
- CoPP policies are used to limit known traffic to a given rate while protecting the CPU from unexpected extreme rates of traffic that could impact the stability of the router.
- Typical CoPP implementations use only an input policy that allows traffic to the control plane to be policed to a desired rate.
- The CoPP policy is then implemented to limit traffic to the control plane CPU to a specific rate for each class.
- The QoS police command uses **conform**, **exceed**, and **violate** actions, which can be configured to transmit or drop traffic.

Configuring ACLs for CoPP

After the network traffic has been identified, ACLs can be built for matching in a class map.

Example 26-33 demonstrates a list of ACLs matching traffic identified by EPC and network documentation.

Example 26-33 *Configuring an Access List for CoPP*

```
ip access-list extended ACL-CoPP-ICMP
 permit icmp any any echo-reply
 permit icmp any any ttl-exceeded
 permit icmp any any unreachable
 permit icmp any any echo
 permit udp any any range 33434 33463 ttl eq 1
!
ip access-list extended ACL-CoPP-IPsec
 permit esp any any
 permit gre any any
 permit udp any eq isakmp any eq isakmp
 permit udp any any eq non500-isakmp
 permit udp any eq non500-isakmp any
!
ip access-list extended ACL-CoPP-Initialize
 permit udp any eq bootps any eq bootpc
!
ip access-list extended ACL-CoPP-Management
 permit udp any eq ntp any
 permit udp any any eq snmp
 permit tcp any any eq 22
 permit tcp any eq 22 any established
!
ip access-list extended ACL-CoPP-Routing
 permit tcp any eq bgp any established
 permit eigrp any host 224.0.0.10
 permit ospf any host 224.0.0.5
 permit ospf any host 224.0.0.6
 permit pim any host 224.0.0.13
 permit igmp any any
```

Class Configuration for CoPP

The class configuration for CoPP uses the ACLs to match known protocols being used and is demonstrated in Example 26-34.

Example 26-34 *Class Configuration for CoPP*

```
class-map match-all CLASS-CoPP-IPsec
 match access-group name ACL-CoPP-IPsec
class-map match-all CLASS-CoPP-Routing
 match access-group name ACL-CoPP-Routing
class-map match-all CLASS-CoPP-Initialize
 match access-group name ACL-CoPP-Initialize
class-map match-all CLASS-CoPP-Management
 match access-group name ACL-CoPP-Management
class-map match-all CLASS-CoPP-ICMP
 match access-group name ACL-CoPP-ICMP
```

Policy Configuration for CoPP

The policy map for how the classes operate shows how to police traffic to a given rate in order to minimize any ability to overload the router.

In order to guarantee that CoPP does not introduce issues, the **violate** action is set to **transmit** for all the vital classes until a baseline for normal traffic flows is established.

Example 26-35 shows the CoPP policy.

Example 26-35 *Policy Configuration for CoPP*

```
policy-map POLICY-CoPP
 class CLASS-CoPP-ICMP
  police 8000 conform-action transmit  exceed-action transmit
         violate-action drop
 class CLASS-CoPP-IPsec
  police 64000 conform-action transmit  exceed-action transmit
         violate-action transmit
 class CLASS-CoPP-Initialize
  police 8000 conform-action transmit  exceed-action transmit
         violate-action drop
 class CLASS-CoPP-Management
  police 32000 conform-action transmit  exceed-action transmit
         violate-action transmit
 class CLASS-CoPP-Routing
  police 64000 conform-action transmit  exceed-action transmit
         violate-action transmit
 class class-default
  police 8000 conform-action transmit  exceed-action transmit
         violate-action drop
```

Applying the Policy for CoPP

The CoPP policy map needs to be applied to the control plane with the command **service-policy** {**input**|**output**} *policy-name* under control plane configuration mode, as demonstrated in Example 26-36.

Example 26-36 *Applying the Policy for CoPP*

```
control-plane
 service-policy input POLICY-CoPP
```

Verifying the Policy for CoPP

After the policy map has been applied to the control plane, it needs to be verified. In Example 26-37, traffic matching CLASS-CoPP-Routing has exceeded the configured rate.

Example 26-37 *Verifying the Policy for CoPP*

```
R1# show policy-map control-plane input
 Control Plane

  Service-policy input: POLICY-CoPP

    Class-map: CLASS-CoPP-ICMP (match-all)
      154 packets, 8912 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name ACL-CoPP-ICMP
      police:
          cir 8000 bps, bc 1500 bytes, be 1500 bytes
        conformed 154 packets, 8912 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        violated 0 packets, 0 bytes; actions:
          drop
        conformed 0000 bps, exceeded 0000 bps, violated 0000 bps

    Class-map: CLASS-CoPP-IPsec (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: access-group name ACL-CoPP-IPsec
      police:
          cir 64000 bps, bc 2000 bytes, be 2000 bytes
        conformed 0 packets, 0 bytes; actions:
          transmit
        exceeded 0 packets, 0 bytes; actions:
          transmit
        violated 0 packets, 0 bytes; actions:
          transmit
        conformed 0000 bps, exceeded 0000 bps, violated 0000 bps

    Class-map: CLASS-CoPP-Initialize (match-all)
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
```

Verifying the Policy for CoPP (Cont.)

```
    Match: access-group name ACL-CoPP-Initialize
    police:
        cir 8000 bps, bc 1500 bytes, be 1500 bytes
      conformed 0 packets, 0 bytes; actions:
        transmit
      exceeded 0 packets, 0 bytes; actions:
        transmit
      violated 0 packets, 0 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps, violated 0000 bps

  Class-map: CLASS-CoPP-Management (match-all)
    0 packets, 0 bytes
    5 minute offered rate 0000 bps, drop rate 0000 bps
    Match: access-group name ACL-CoPP-Management
    police:
        cir 32000 bps, bc 1500 bytes, be 1500 bytes
      conformed 0 packets, 0 bytes; actions:
        transmit
      exceeded 0 packets, 0 bytes; actions:
        transmit
      violated 0 packets, 0 bytes; actions:
        transmit
      conformed 0000 bps, exceeded 0000 bps, violated 0000 bps
```

```
  Class-map: CLASS-CoPP-Routing (match-all)
    92 packets, 123557 bytes
    5 minute offered rate 4000 bps, drop rate 0000 bps
    Match: access-group name ACL-CoPP-Routing
    police:
        cir 64000 bps, bc 2000 bytes, be 2000 bytes
      conformed 5 packets, 3236 bytes; actions:
        transmit
      exceeded 1 packets, 1383 bytes; actions:
        transmit
      violated 86 packets, 118938 bytes; actions:
        transmit
      conformed 1000 bps, exceeded 1000 bps, violated 4000 bps

  Class-map: class-default (match-any)
    56 packets, 20464 bytes
    5 minute offered rate 1000 bps, drop rate 0000 bps
    Match: any
    police:
        cir 8000 bps, bc 1500 bytes, be 1500 bytes

      conformed 5 packets, 2061 bytes; actions:
        transmit
      exceeded 0 packets, 0 bytes; actions:
        transmit
      violated 0 packets, 0 bytes; actions:
        drop
      conformed 0000 bps, exceeded 0000 bps, violated 0000 bps
```

Device Hardening

In addition to providing device access control and protection, disabling unused services and features, hardening a router reduces the amount of router CPU and memory utilization that would be required to process those packets.

Device Hardening

The following is a list of additional commands that can be used to harden a router. All interface-specific commands are applied only to the interface connected to the public network.

- **Disable topology discovery tools -** Tools such as Cisco Discovery Protocol (CDP) and Link Layer Discovery Protocol (LLDP) can provide unnecessary information to routers outside your control. The services can be disabled with the interface parameter commands **no cdp enable**, **no lldp transmit**, and **no lldp receive**.

- **Disable TCP and UDP small services -** The commands **service tcp-keepalive-in** and **service tcp-keepalive-out** ensure that devices send TCP keepalives for inbound/outbound TCP sessions. This ensures that the device on the remote end of the connection is still accessible and that half-open or orphaned connections are removed from the local device.

- **Disable IP redirect services -** An ICMP redirect is used to inform a device of a better path to the destination network. An IOS device sends an ICMP redirect if it detects network traffic hairpinning on it. This behavior is disabled with the interface parameter command **no ip redirects.**

- **Disable proxy Address Resolution Protocol (ARP) -** Proxy ARP is a technique that a router uses to answer ARP requests intended for a different router. The router fakes its identity and sends out an ARP response for the router that is responsible for that network.

Device Hardening (Cont.)

- **Disable service configuration -** Cisco devices support automatic configuration from remote devices through TFTP and other methods. This service should be disabled with the command **no service config**.

- **Disable the Maintenance Operation Protocol (MOP) service -** The MOP service is not needed and should be disabled globally with the command **no mop enabled** and with the interface parameter command **no mop enabled**.

- **Disable the packet assembler/disassembler (PAD) service -** The PAD service is used for X.25 and is not needed. It can be disabled with the **command no service pad**.

Prepare for the Exam

Prepare for the Exam

Key Topics for Chapter 26

Description	
Access Control List (ACL)	Password types
ACL categories	Local username configuration options
Applying ACL to an interface	Privilege levels
CLI access methods	SSH versions
Line password protection options	Authentication, authorization, and accounting (AAA)
AAA primary use cases	TACACS+ key differentiator
RADIUS key differentiators	Paragraph Zone-Based Firewall (ZBFW)
ZBFW default zones	Control plane policing (CoPP)

Prepare for the Exam

Key Terms for Chapter 26

Terms	
Access Control List (ACL)	Secure Shell (SSH)
Authentication, authorization, and accounting (AAA)	Telnet
Control plan policy (CoPP)	Terminal Access Controller Access-Control System Plus (TACACS+)
privilege level	Zone-Based Firewall (ZBFW)
Remote Authentication Dial-in User Service (RADIUS)	

Command Reference for Chapter 26

Task	Command Syntax
Apply an ACL to an interface	**ip access-group** {*access-list-number* | *name*} {**in**|**out**}
Apply an ACL to a vty line	**access-class** {*access-list-number*|*accesslist-name*} {**in**|**out**}
Encrypt type 0 passwords in the configuration	**service password-encryption**
Create a username with a type 8 and type 9 password option	**username** {*username*} **algorithm-type** {**md5** | **sha256** | **scrypt**} **secret** {*password*}
Enable username and password authentication on vty lines	**login local**
Change command privilege levels	**privilege** {*mode*} **level** {*level*}{*command string*
Allow only SSH for a vty line without using an ACL	**transport input ssh**

Command Reference for Chapter 26 (Cont.)

Task	Command Syntax
Enable SSHv2 on a router	**hostname** {*hostname name*}**ip domain-name** {*domain-name*}**crypto key generate rsa**
Disconnect terminal line users that are idle	**exec-timeout** {*minutes*} {*seconds*}
Enable AAA	**aaa new-model**
Enable AAA authorization for the console line	**aaa authorization console**
AAA fallback authorization method that authorizes commands if user is successfully authenticated	**if-authenticated**
Enable AAA authorization for config commands	**aaa authorization config-commands**
Apply a ZBFW security zone to an interface	**zone-member security** *zone-name*
Apply an inspection policy map to a zonepair	**service-policy type inspect** *policy-name*
Apply a CoPP policy map to the control plane (two commands)	**control planeservice-policy** {**input**|**output**} *policy-name*


CISCO