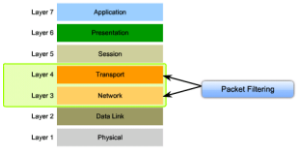
Implementing Firewall Technologies

Firewalls

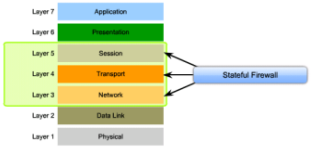
- Network firewalls separate protected from non-protected areas preventing unauthorized users from accessing protected network resources.
- Technologies used:
  - ACLs
    - Standard, extended, numbered and named ACLs
  - Advanced ACLs
    - Stateful firewall - ACLs with the **`established`** keyword
    - Reflexive (dynamic) ACLs, timed-based ACLs
  - Zone-Based Firewall Feature

Common Types of Firewalls

- Packet-filtering firewall



- Stateful firewall



ACLs

ACL Types

- Virtually any type of traffic can be defined explicitly by using an appropriately Numbered ACL.

Protocol	Range
IP	1-99 , 1300-1999
Extended IP	100-199 , 2000-2699
Ethernet type code	200-299
Ethernet address	700-799
IPX SAP	1000-1099
Extended transparent bridging	1100-1199

- In the past, the Ethernet type field of an Ethernet frame header was used to define certain types of traffic.
  - For example, Ethernet type 0x0806 indicated an ARP frame, 0x8035 indicated a RARP frame, ...
- It was also common to create ACLs based on MAC addresses.

Standard ACLs

Types of Cisco ACLs

Standard ACLs filter IP packets based on the

```
access-list 10 permit 192.168.30.0 0.0.0.255
```

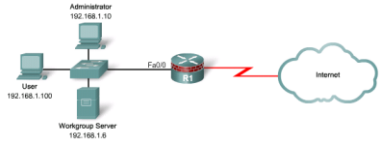
```
access-list {1-99} {permit | deny} source-addr [source-wildcard]
```

- **Note:**
  - Can be applied in an incoming or outgoing direction on an interface using the **`ip access-group`** command.
  - It can also be applied on a VTY port using the **`access-class`** command.

Standard ACLs

- Create a standard named ACL on R1 called **RESTRICT-VTY** that permits Telnet access to only the administrative host.

```
R1(config)# ip access-list standard RESTRICT-VTY
R1(config-std-nacl)# remark Permit only Admin host
R1(config-std-nacl)# permit host 192.168.1.10
R1(config-std-nacl)# exit
R1(config)# line vty 0 4
R1(config-line)# access-class RESTRICT-VTY
R1(config-line)# exit
```



Extended ACLs

Extended ACLs filter IP packets based on several attributes, including the following:

-
-
-

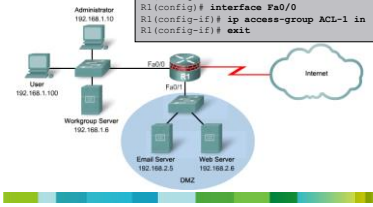
```
access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80
```

```
access-list {100-199} {permit | deny} protocol source-addr
[source-wildcard][operator operand] destination-addr [destination-
wildcard] [operator operand][established]
```


Extended ACLs - 1

- Create an extended named ACL called **ACL-1**, applied incoming on the Fa0/0 interface, that denies the workgroup server outside access but permits the remainder of the LAN users outside access using the **established** keyword.

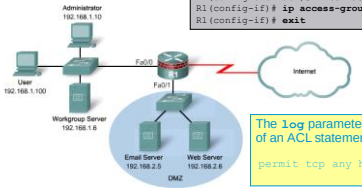
```
R1(config)# ip access-list extended ACL-1
R1(config-ext-nacl)# remark LAN ACL
R1(config-ext-nacl)# deny ip host 192.168.1.6 any
R1(config-ext-nacl)# permit tcp 192.168.1.0 0.0.0.255 any
established
R1(config-ext-nacl)# deny ip any any
R1(config-ext-nacl)# exit
R1(config)# interface Fa0/0
R1(config-if)# ip access-group ACL-1 in
R1(config-if)# exit
```



Extended ACLs - 2

- Create an extended named ACL called **ACL-2**, applied outgoing on the Fa0/1 DMZ interface, permitting access to the specified Web and Email servers.

```
R1(config)# ip access-list extended ACL-2
R1(config-ext-nacl)# remark DMZ ACL
R1(config-ext-nacl)# permit tcp any host 192.168.2.5 eq 25
R1(config-ext-nacl)# permit tcp any host 192.168.2.6 eq 80
R1(config-ext-nacl)# deny ip any any
R1(config-ext-nacl)# interface Fa0/1
R1(config-if)# ip access-group ACL-2 out
R1(config-if)# exit
```

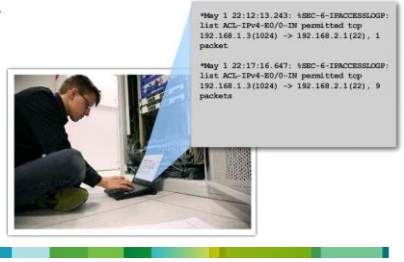


The `log` parameter can be appended to the end of an ACL statement.

```
permit tcp any host 192.168.2.6 eq 80 log
```


Logging

- When configured, the IOS software compares packets and finds a match to the statement.
- The router then logs it to any enabled logging facility, such as:
  - the console
  - the internal buffer
  - syslog server

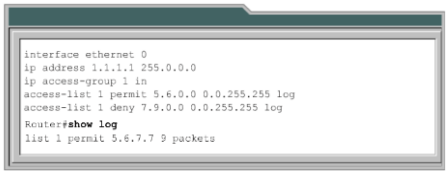


Logging

- Several pieces of information are logged:
  - Action - permit or deny
  - Protocol - TCP, UDP, or ICMP
  - Source and destination addresses
  - For TCP and UDP - source and destination port numbers
  - For ICMP - message types
- Log messages are ***processed switched*** on the first packet match and then at five minute intervals after that first packet match.

View ACL operation

- A useful command for viewing access list operation is the **show log** command.
- To reset counters, use the **clear ip access-list counter** [*number* | *name*] command.



ACL Caveats

- Implicit deny all:
  - All Cisco ACLs end with an implicit "deny all" statement.
- Standard ACL packet filtering:
  - Standard ACLs are limited to packet filtering based on source addresses only.
  - Extended ACLs might need to be created to fully implement a security policy.
- Order of statements:
  - ACLs have a policy of first match; when a statement is matched, the list is no longer examined.
  - Ensure that statements at the top of the ACL do not negate any statements found lower.
  - Place specific ACL statements higher in the ACL and more general statements near the end.

ACL Caveats

- Directional filtering:
  - ACLs can be applied to inbound packets (toward the interface) or outbound packets (away from the interface).
  - Double-check the direction of data that an ACL is filtering.
- Special packets:
  - Router-generated packets, such as routing table updates, are not subject to outbound ACL statements on the source router.
  - If the security policy requires filtering these types of packets, inbound ACLs on adjacent routers or other router filter mechanism must be used.
- Modifying ACLs:
  - New entries are added to an ACL, are always added to the bottom.
  - Starting with Cisco IOS 12.3, sequence numbers can be used to edit an ACL.
  - The ACL is processed top-down based on the sequence numbers of the statements (lowest to highest).

ACL Sequence Numbers

- The default behavior when adding a statement to an ACL is that the statement is added to the end. Without sequence numbers the only way to add a statement between existing entries was to delete the ACL and recreate it.
- Likewise, the only way to delete an entry was to delete the entire ACL and recreate it.
- IP access list sequence numbers allow you to selectively remove a statement from an existing ACL or to add a new statement at any position within the ACL.
- This feature is not available on old-style numbered access lists, which existed before named access lists. Keep in mind that you can name an access list with a number, so numbers are allowed when they are entered in the standard or extended named access list configuration mode using **the ip access-list {standard | extended}** *access-list-name* command.

I don't see my sequence numbers!

- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. Therefore, sequence numbers are not displayed in the **show running-config** or **show startup-config** output.
- To view the sequence numbers, use the **show ip access-lists** *access-list-name* command or the **show access-list** command.
- By default sequence numbers start at 10 and are incremented by 10 if a sequence number is not specified when adding statements.

Modify an ACL using Sequence Numbers

- First use the **show** command to view the existing sequence numbers.

```
R1# show access-list 150
Extended IP acess list 150
    10 permit tcp any any eq www
    20 permit tcp any any eq telnet
    30 permit tcp any any eq smtp
    40 permit tcp any any eq pop3
    50 permit tcp any any eq 21
    60 permit tcp any any eq 20
```

- Resequence if necessary.
- Use the **no** *sequence-number* command to delete a statement.

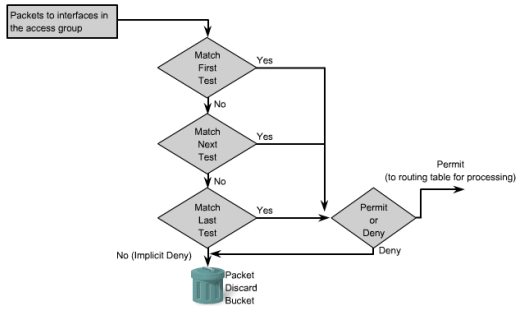
```
R1(config)# ip access-list extended 150
R1(config-ext-nacl)# no 20
```

- Use the *sequence-number* **{permit | deny}** command to add a statement within the ACL.

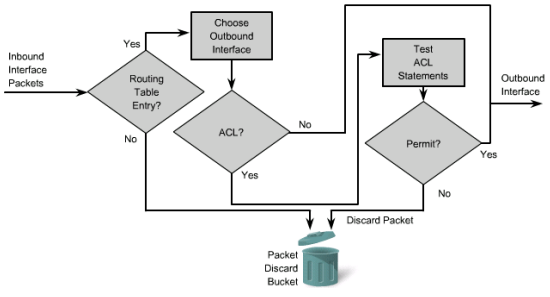
```
R1(config)# ip access-list extended 150
R1(config-ext-nacl)# 20 permit tcp host 192.168.1.100 any eq telnet
```

ACL Placement

Inbound ACL Operation Flow



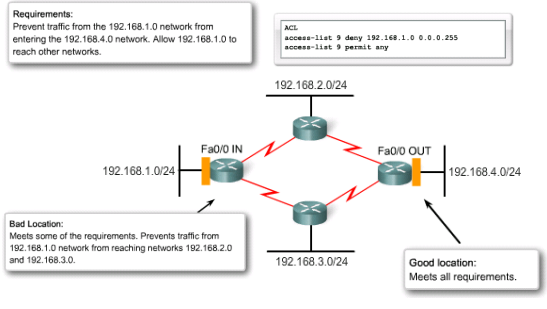
Outbound ACL Operation Flow



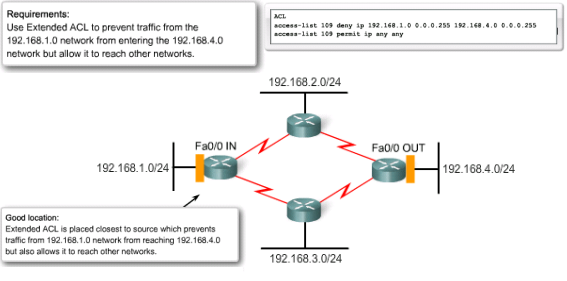
ACL Placement

- Standard ACL placement:
  - Standard ACLs are placed **as close to the destination as possible**.
  - Standard ACLs filter packets based on the source address only so placing these ACLs too close to the source can adversely affect packets by denying all traffic, including valid traffic.
- Extended ACL placement:
  - Extended ACLs are placed on routers **as close to the source as possible** that is being filtered.
  - Placing Extended ACLs too far from the source is inefficient use of network resources because packets can be sent a long way only to be dropped or denied.

Where to place a Standard ACL?

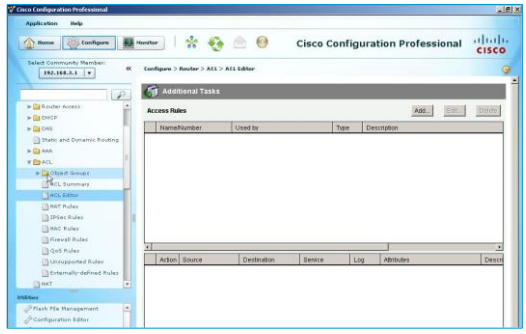


Where to place a Extended ACL?

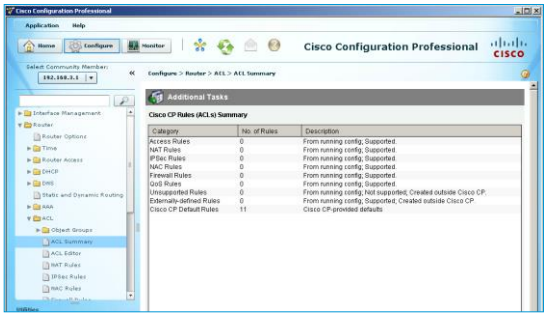


Configuring ACLs using CCP

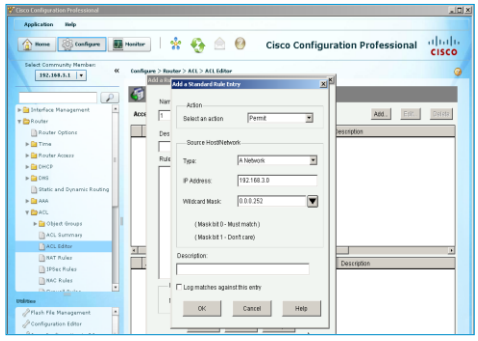
Configuring ACLs using CCP



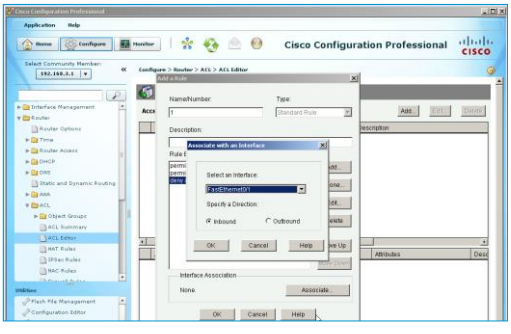
Configuring ACLs using CCP



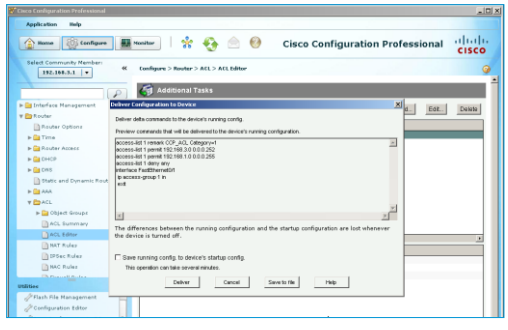
Configuring ACLs using CCP



Configuring ACLs using CCP



Configuring ACLs using CCP



Complex ACLs

TCP Sessions

- In a modern network all traffic from the outside should be blocked from entering the inside unless:
  - It is explicitly permitted by an ACL.
  - It is returning traffic initiated from the inside of the network.
- Many common applications rely on TCP, which builds a virtual circuit between two endpoints.
- Traffic filtering solutions based on the two way connectivity of TCP were introduced:
  - TCP Established
  - Reflexive ACLs

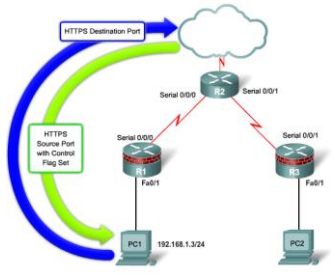
TCP Established ACLs

- In 1995, the first generation IOS traffic filtering solution based on the TCP **`established`** keyword for extended IP ACLs.
  - The TCP **`established`** keyword blocks all traffic coming from the Internet except for the TCP reply traffic associated with established TCP traffic initiated from the inside of the network.
- The **`established`** keyword forces the router to check whether the TCP ACK or RST control flag is set.
  - If the ACK flag is set, the TCP traffic is allowed in.
  - If not, it is assumed that the traffic is associated with a new connection initiated from the outside.

TCP Established ACLs

- Using the **`established`** keyword does not implement a stateful firewall on a router.
  - The **`established`** parameter allows any TCP segments with the appropriate control flag.
  - No stateful information is maintained to keep track of traffic initiated from the inside of the network since the router does not keep track of conversations to determine whether the traffic is return traffic associated with a connection initiated from inside the network.

TCP Established ACLs




```
R1(config)# access-list 100 permit tcp any eq 443 192.168.1.0 0.0.0.255 established
R1(config)# access-list 100 deny ip any any
R1(config)# interface s0/0/0
R1(config-if)# ip access-group 100 in
```


Reflexive ACLs

- In 1996, the second generation IOS solution for session filtering was Reflexive ACLs.
- Unlike the TCP Established feature which just used ACK and RST bits, reflexive ACLS filter traffic based on source, destination addresses, and port numbers.
- Also, session filtering uses temporary filters that are removed when a session is over adding a time limit on a hacker's attack opportunity.

Reflexive ACLs

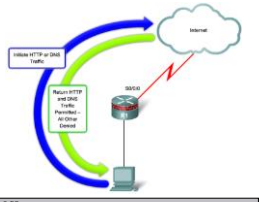
- Network administrators use reflexive ACLs to allow IP traffic for sessions originating from their network while denying IP traffic for sessions originating outside the network.
- The router examines the outbound traffic and when it sees a new connection, it adds an entry to a temporary ACL to allow replies back in.
  - These entries are automatically created when a new IP session begins, for example, with an outbound packet, and the entries are automatically removed when the session ends.

Configuring a Reflexive ACL

- Step 1.
  - Create an internal ACL that looks for new outbound sessions and creates temporary reflexive ACEs.
- Step 2.
  - Create an external ACL that uses the reflexive ACLs to examine return traffic.
- Step 3.
  - Activate the Named ACLs on the appropriate interfaces.

Reflexive ACL Example

- Create a reflexive ACL that matches internal users surfing the Internet with a web browser and relying on DNS with a 10 second timeout period.




```
R1(config)# ip access-list extended INTERNAL_ACL
R1(config-ext-nacl)# permit tcp any any eq 80 reflect WEB-ONLY-REFLEXIVE-ACL
R1(config-ext-nacl)# permit udp any any eq 53 reflect DNS-ONLY-REFLEXIVE-ACL timeout 10
R1(config-ext-nacl)# exit
R1(config)# ip access-list extended EXTERNAL_ACL
R1(config-ext-nacl)# evaluate WEB-ONLY-REFLEXIVE-ACL
R1(config-ext-nacl)# evaluate DNS-ONLY-REFLEXIVE-ACL
R1(config-ext-nacl)# deny ip any any
R1(config-ext-nacl)# exit
R1(config)# interface s0/0/0
R1(config-if)# ip access-group INTERNAL_ACL out
R1(config-if)# ip access-group EXTERNAL_ACL in
```

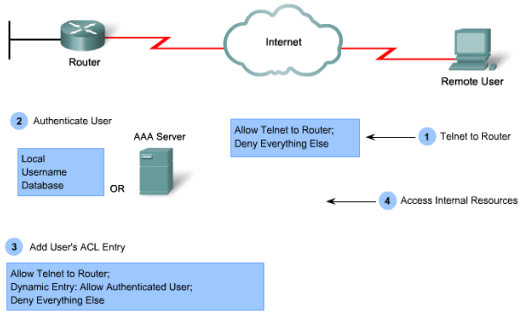

Dynamic ACLs

- Dynamic ACLs are also called lock-and-key ACLs.
- Dynamic ACLs authenticate the user and then permits limited access through your firewall router for a host or subnet for a finite period.
- Dynamic ACLs are dependent on:
  - Telnet connectivity
  - Authentication (local or remote)
  - Extended ACLs

Implementing Dynamic ACLs

- An extended ACL is applied to block all traffic through the router except Telnet.
  - Users who want to traverse the router are blocked by the ACL until they use Telnet to connect to the router and are authenticated.
- Users authenticate using Telnet, and then dropped.
  - However, a single-entry dynamic ACL is added to the extended ACL that exists.
  - This permits traffic for a particular period; idle and absolute timeouts are possible.

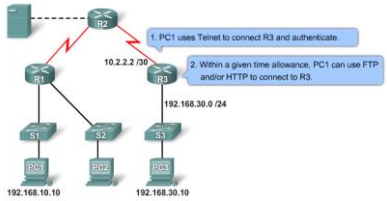
Configuring Dynamic ACLs



When to Use Dynamic ACLs

- When you want a specific remote user or group of remote users to access a host within your network, connecting from their remote hosts via the Internet.
- When you want a subset of hosts on a local network to access a host on a remote network that is protected by a firewall.

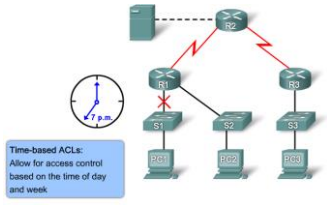
Dynamic ACL Example




```
R3(config)# username Student password cisco
R3(config)# access-list 101 permit tcp any host 10.2.2.2 eq telnet
R3(config)# access-list 101 dynamic TESTLIST timeout 15 permit ip 192.168.10.0 0.0.0.255
192.168.3.0 0.0.0.255
R3(config)# interface s0/0/1
R3(config-if)# ip access-group 101 in
R3(config-if)# exit
R3(config)# line vty 0 4
R3(config-line)# login local
R3(config-line)# autocommand access-enable host timeout 15
```


Time-based ACLs

- Time-based ACLs allow for access control based on time.



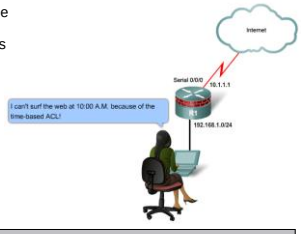
Time-based ACLs

- To implement time-based ACLs:
  - Create a time range that defines specific times of the day and week.
  - Identify the time range with a name and then refer to it by a function.
  - The time restrictions are imposed on the function itself.

Step 1	R1(config)#**time-range EVERYOTHERDAY** R1(config-time-range)#**periodic Monday Wednesday Friday 8:00 to 17:00**
Step 2	R1(config)#**access-list 101 permit tcp 192.168.10.0 0.0.0.255 any eq telnet time-range EVERYOTHERDAY**
Step 3	R1(config)#**interface s0/0/0** R1(config-if)#**ip access-group 101 out**

Time-based ACL Example

- Users are not allowed to access the Internet during business hours, except during lunch and after hours between 5 p.m. and 7 p.m.

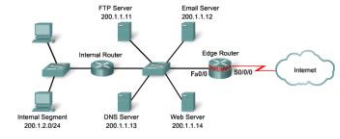



```
R1(config)# time-range EMPLOYEE-TIME
R1(config-time-range)# periodic weekdays 12:00 to 13:00
R1(config-time-range)# periodic weekdays 17:00 to 19:00
R1(config-time-range)# exit
R1(config)# access-list 100 permit ip 192.168.1.0 0.0.0.255 any time-range EMPLOYEE-TIME
R1(config)# access-list 100 deny ip any any
R1(config)# interface FastEthernet 0/1
R1(config-if)# ip access-group 100 in
R1(config-if)# exit
```


Troubleshooting ACLs

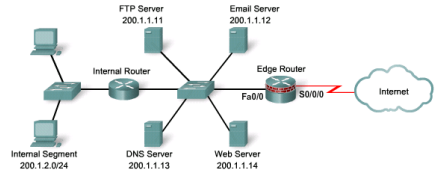
ACL Troubleshooting Commands

- Two commands are very useful for troubleshooting ACLs:
  - **show access-lists**
  - **debug ip packet (detail)**



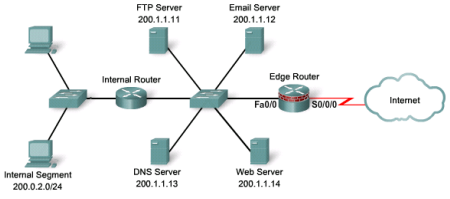

```
Perimeter# show access-list 100
Extended IP access list 100
    permit tcp any host 200.1.1.14 eq www  (189 matches)
    permit udp any host 200.1.1.13 eq domain (32 matches)
    permit tcp any host 200.1.1.12 eq smtp
    permit tcp any eq smtp host 200.1.1.12 established
    permit tcp any host 200.1.1.11 eq ftp
    permit tcp any host 200.1.1.11 eq ftp-data
    permit tcp any eq www 200.1.2.0 0.0.0.255 established
    permit udp any eq domain 200.1.2.0 0.0.0.255
    deny ip any any (1237 matches)
```


show access-lists




```
Perimeter# show access-list 100
Extended IP access list 100
    permit tcp any host 200.1.1.14 eq www  (189 matches)
    permit udp any host 200.1.1.13 eq domain (32 matches)
    permit tcp any host 200.1.1.12 eq smtp
    permit tcp any eq smtp host 200.1.1.12 established
    permit tcp any host 200.1.1.11 eq ftp
    permit tcp any host 200.1.1.11 eq ftp-data
    permit tcp any eq www 200.1.2.0 0.0.0.255 established
    permit udp any eq domain 200.1.2.0 0.0.0.255
    deny ip any any (1237 matches)
```


debug ip packet




```
Edge# debug ip packet
IP packet debugging is on

IP: s=200.0.2.2 (FastEthernet0/0), d=172.69.2.42 (Serial0/0/0), g=172.69.16.2, forward
IP: s=200.0.2.2 (FastEthernet0/0), d=172.16.2.42 (Serial0/0/0), g=172.69.16.2, forward
IP: s=200.0.2.2 (FastEthernet0/0), d=172.69.43.126 (Serial0/0/0), g=172.69.16.2, forward
IP: s=200.0.2.2 (FastEthernet0/0), d=172.16.2.42 (Serial0/0/0), g=172.69.16.2, access denied
```


cisco

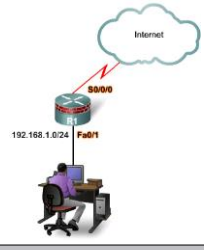
Mitigating Attacks with ACLs

Mitigating Attacks with ACLs

- ACLs can be used to mitigate many network threats:
  - IP address spoofing, inbound and outbound
  - DoS TCP SYN attacks
  - DoS smurf attacks
- ACLs can also filter the following traffic:
  - ICMP messages, inbound and outbound
  - traceroute

Do Not Allow Addresses to be Spoofed

- Deny all IP packets containing the following IP addresses in their source field:
  - Any local host addresses (127.0.0.0/8)
  - Any reserved private addresses (RFC 1918)
  - Any addresses in the IP multicast address range (224.0.0.0/4)

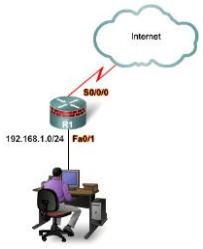


Inbound on S0/0/0

```
R1(config)# access-list 150 deny ip 0.0.0.0 0.255.255.255 any
R1(config)# access-list 150 deny ip 10.0.0.0 0.255.255.255 any
R1(config)# access-list 150 deny ip 127.0.0.0 0.255.255.255 any
R1(config)# access-list 150 deny ip 172.16.0.0 0.15.255.255 any
R1(config)# access-list 150 deny ip 192.168.0.0 0.0.255.255 any
R1(config)# access-list 150 deny ip 224.0.0.0 15.255.255.255 any
R1(config)# access-list 150 deny ip host 255.255.255.255 any
```

Do Not Allow Addresses to be Spoofed

- Do not allow any outbound IP packets with a source address other than a valid IP address of the internal network.
  - Create an ACL that permits only those packets that contain source addresses from inside the network and denies all others.

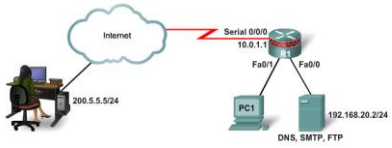


Inbound on Fa0/1

```
R1(config)# access-list 105 permit ip 192.168.1.0 0.0.0.255 any
```


Protect DNS, SMTP, and FTP

- DNS, SMTP, and FTP are common services that often must be allowed through a firewall.



Outbound on Fa0/0

```
R1(config)# access-list 180 permit udp any host 192.168.20.2 eq domain
R1(config)# access-list 180 permit tcp any host 192.168.20.2 eq smtp
R1(config)# access-list 180 permit tcp any host 192.168.20.2 eq ftp
R1(config)# access-list 180 permit tcp host 200.5.5.5 host 192.168.20.2 eq telnet
R1(config)# access-list 180 permit tcp host 200.5.5.5 host 192.168.20.2 eq 22
R1(config)# access-list 180 permit udp host 200.5.5.5 host 192.168.20.2 eq syslog
R1(config)# access-list 180 permit udp host 200.5.5.5 host 192.168.20.2 eq snmptrap
```


Filter ICMP Messages

- Hackers use ICMP packets for pings sweeps and DoS flood attacks, and use ICMP redirect messages to alter host routing tables.
  - Both ICMP echo and redirect messages should be blocked inbound by the router.

Filter ICMP Messages

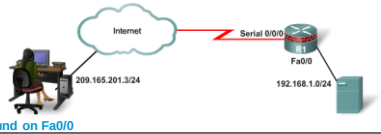
- Several inbound ICMP messages are required for proper network operation:
  - **Echo reply** - Allows internal users to ping external hosts.
  - **Source quench** - Requests the sender to decrease the traffic rate.
  - **Unreachable** - Unreachable messages are generated for packets that are administratively denied by an ACL.

Inbound on S0/0/0

```
R1(config)# access-list 150 permit icmp any any echo-reply
R1(config)# access-list 150 permit icmp any any source-quench
R1(config)# access-list 150 permit icmp any any unreachable
R1(config)# access-list 150 deny icmp any any
R1(config)# access-list 150 permit ip any any
```


Filter ICMP Messages

- Several outbound ICMP messages are required for proper network operation:
  - **Echo** - Allows users to ping external hosts.
  - **Parameter problem** - Informs the host of packet header problems.
  - **Packet too big** - Required for packet MTU discovery.
  - **Source quench** - Throttles down traffic when necessary.



Inbound on Fa0/0

```
R1(config)# access-list 105 permit icmp 192.168.1.0 0.0.0.255 any echo
R1(config)# access-list 105 permit icmp 192.168.1.0 0.0.0.255 any parameter-problem
R1(config)# access-list 105 permit icmp 192.168.1.0 0.0.0.255 any packet-too-big
R1(config)# access-list 105 permit icmp 192.168.1.0 0.0.0.255 any source-quench
R1(config)# access-list 105 deny icmp any any
R1(config)# access-list 105 permit ip any any
```


IPv6 ACLs

IPv6 ACL Configuration

- IPv6 ACLs are similar to IPv4 ACLs.
  - They allow filtering on source and destination addresses, source and destination ports, and protocol type.
- IPv6 ACLs are created using the **ipv6 access-list** command.

```
Router(config)# ipv6 access-list access-list-name
Router(config-ipv6-acl)# {permit | deny} protocol [source-ipv6-prefix/prefix-length] [operator operand] [destination-ipv6-prefix/prefix-length] [operator operand]
```

- IPv6 ACLs are applied to an interface using the **ipv6 traffic-filter** *access-list-name* {**in** | **out**} command.

IPv6 ACL Implicit Entries

- All IPv6 ACLs contain 2 implicit permit statements to allow IPv6 neighbor discovery packets to be sent and received.
  - **permit icmp any any nd-na**
  - **permit icmp any any nd-ns**
- Like IPv4 ACLs, all IPv6 ACLs include an implicit deny as the last statement.
  - **deny ipv6 any any**
- These statements will not display in the configuration output. A best practice is to manually enter all 3 implicit commands.
  - Manually entering the implicit deny statement will also allow you to log denied packets without affecting neighbor discovery.

cisco

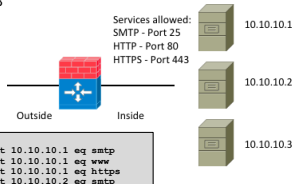
Object Groups

Using Object Groups in ACLs

- Object groups are used to classify users, devices, or protocols into groups.
- These groups can then be used to create access control policies for groups of objects in easy to read statements.
- Both IPv4 and IPv6 ACLs can use object groups.

Why Use Object Groups?

- In this example topology, there are 3 servers, each requiring outside to inside access for 3 protocols.
- Without object groups, we have to configure a permit statement for each server, for each protocol:




```
R1(config)# ip access-list extended In
R1(config-ext-nacl)# permit tcp any host 10.10.10.1 eq smtp
R1(config-ext-nacl)# permit tcp any host 10.10.10.1 eq www
R1(config-ext-nacl)# permit tcp any host 10.10.10.1 eq https
R1(config-ext-nacl)# permit tcp any host 10.10.10.2 eq smtp
R1(config-ext-nacl)# permit tcp any host 10.10.10.2 eq www
R1(config-ext-nacl)# permit tcp any host 10.10.10.2 eq https
R1(config-ext-nacl)# permit tcp any host 10.10.10.3 eq smtp
R1(config-ext-nacl)# permit tcp any host 10.10.10.3 eq www
R1(config-ext-nacl)# permit tcp any host 10.10.10.3 eq https
```

- But, what if other servers or protocols are added later? You will have to edit the ACL!

Object Groups Example

- For the same topology, using object group configuration, first create the service object for the services:

```
R1(config)# object-group service Web-svcs tcp
R1(config-service-group)# tcp smtp
R1(config-service-group)# tcp www
R1(config-service-group)# tcp https
```

- Next, create the network object for the servers:
  - This example uses the **range** keyword, you can also use the **host** keyword or define a subnet.

```
R1(config)# object-group network Webservers
R1(config-network-group)# range 10.10.10.1 10.10.10.3
```

- Finally, create the access list:

```
R1(config)# ip access-list extended In
R1(config-ext-nacl)# permit tcp any object-group Webservers object-group Web-svcs
```

- When a new server or service is added, simply edit the object group…you don't have to touch the ACL!

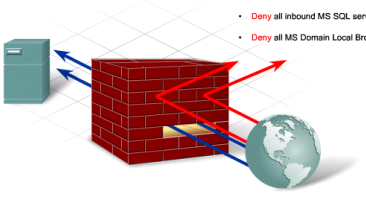
Securing Networks with Firewalls

Firewall

- A firewall prevents undesirable traffic from entering prescribed areas within a network.
- A firewall is a system or group of systems that enforces an access control policy between networks.
  - For example:
    - A packet filtering router
    - A switch with two VLANs
    - Multiple hosts with firewall software
- In 1989, AT&T Bell Laboratories developed the first stateful firewall.
  - A stateful firewall is able to determine if a packet belongs to an existing flow of data.

Firewall

- Allow web traffic from any external address to the web server
- Allow traffic to FTP server
- Allow traffic to SMTP server
- Allow traffic to internal IMAP server
- Deny all inbound traffic with network addresses matching internal-registered IP addresses
- Deny all inbound traffic to server from external addresses
- Deny all inbound ICMP echo request traffic
- Deny all inbound MS Active Directory
- Deny all inbound MS SQL server ports
- Deny all MS Domain Local Broadcasts



Stateless versus Stateful Packet Filtering

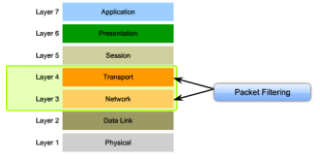
- Stateless packet filtering:
  - ACLs filter traffic based on source and destination IP addresses, TCP and UDP port numbers, TCP flags, and ICMP types and codes.
- Stateful packet filtering:
  - Inspection remembers certain details, or the state of that request.
  - Device maintains records of all connections passing through the firewall, and is able to determine whether a packet is the start of a new connection, or part of an existing connection.
  - A stateful firewall monitors the state of connections, whether the connection is in an initiation, data transfer, or termination state.
- Note:
  - A packet-filtering firewall typically can filter up to the transport layer, while a stateful firewall can filter up to the session layer.

Packet Filtering Firewalls

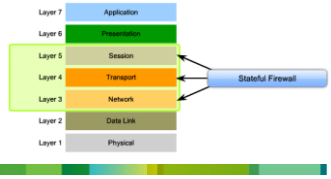
- Packet-filtering firewalls are usually part of a router firewall and primarily uses ACLs.
  - It examines a packet based on the information in a packet header.
- Packet-filtering firewalls use a simple policy table lookup that permits or denies traffic based on specific criteria:
  - Source IP address
  - Destination IP address
  - Protocol
  - Source port number
  - Destination port number
  - Synchronize/start (SYN) packet receipt

Common Types of Firewalls

- Packet-filtering firewall



- Stateful firewall



Stateful Firewalls

- Stateful firewalls are the most versatile and common firewall technology in use.
- Stateful filtering tracks each connection traversing all interfaces of the firewall and confirms that they are valid.
  - The firewall examines information in the headers of Layer 3 packets and Layer 4 segments.

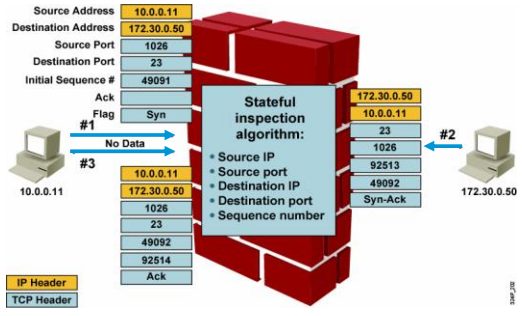
Stateful Firewalls

- Also called “stateful packet filters” and “application-aware packet filters.”
- Stateful firewalls have two main improvements over packet filters:
  - They maintain a session table (state table) where they track all connections.
  - They recognize dynamic applications and know which additional connections will be initiated between the endpoints.

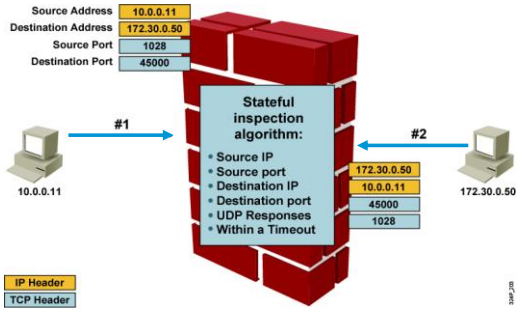
Stateful Firewalls

- Stateful firewalls inspect every packet, compare the packet against the state table, and may examine the packet for any special protocol negotiations.
- Stateful firewalls operate mainly at the Transport (TCP and UDP) layer.

Cisco IOS Firewall TCP Handling

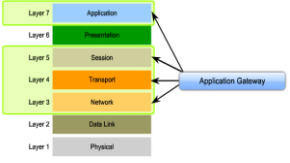


Cisco IOS Firewall UDP Handling

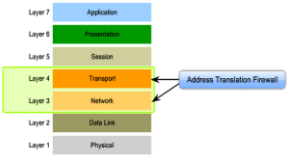


Other Types of Firewalls

- **Application gateway firewall (proxy firewall):**
  - Filters information at OSI Layers 3, 4, 5, and 7.
  - Firewall control and filtering is done in software.



- **Address-translation firewall:**
  - A firewall that expands the number of IP addresses available and hides network addressing design.



Other Types of Firewalls

- **Host-based (server and personal) firewall:**
  - A PC or server with firewall software running on it.
- **Transparent firewall:**
  - A firewall that filters IP traffic between a pair of bridged interfaces.
- **Hybrid firewall:**
  - A firewall that is a combination of the various firewalls types.
  - For example, an application inspection firewall combines a stateful firewall with an application gateway firewall.

Firewall Design

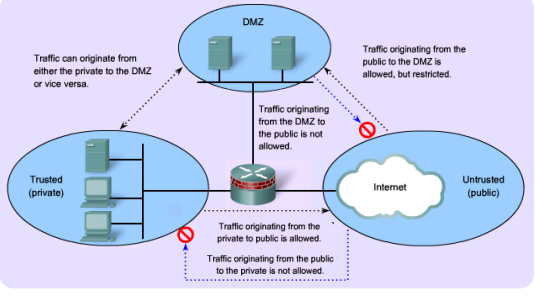
Simple Firewall Design

- Firewall designs can be as simple as having an inside network and outside network using two interfaces.
  - The inside network (or private network) is trusted.
    - The traffic from the inside is usually permitted to traverse the firewall to the outside with little or no restrictions.
    - Traffic returning from the outside that is associated with traffic originating from the inside is permitted to traverse from the untrusted interface to the trusted interface.
  - The outside network (or public network) is untrusted.
    - Traffic originating from the outside is generally blocked entirely or very selectively permitted.

Modern Firewall Design

- Designs involve three or more interfaces on a firewall:
  - One inside network
    - Traffic to the outside is freely permitted.
    - Traffic to the DMZ is freely permitted.
  - One outside network
    - Traffic from the outside is generally blocked entirely unless it is associated with traffic originating from the inside or the DMZ.
  - One DMZ network
    - Traffic from the outside should be very specific such as email, DNS, HTTP, or HTTPS traffic.
    - Traffic to the outside is freely permitted.

Modern Firewall Design



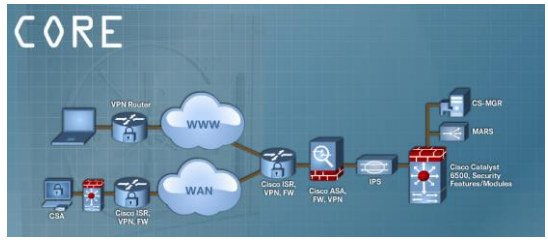
The Cisco IOS Firewall Feature Set

- NAT
- Standard and extended ACLs
- Cisco IOS Firewall
- Cisco IOS IPS
- IPsec network security
- TCP intercept
- Authentication proxy
- User authentication and authorization
- Event logging

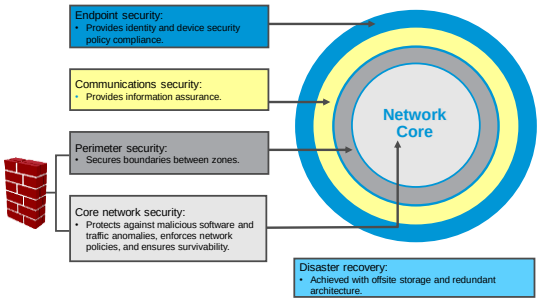
Misconceptions

- *"A firewall is all that is needed to ensure a safe internal network!"*
- It helps but it's not "all that is needed"!
  - A significant number of intrusions, such as viruses, come from hosts within the network.
  - Firewalls do not protect against rogue modem installations.
  - Firewalls do not replace backup and disaster recovery mechanisms resulting from attack or hardware failure.
  - A firewall is no substitute for informed administrators and users.

Defense In-Depth

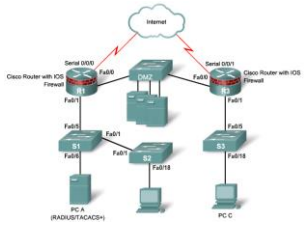


Defense In-Depth



ISR Routers

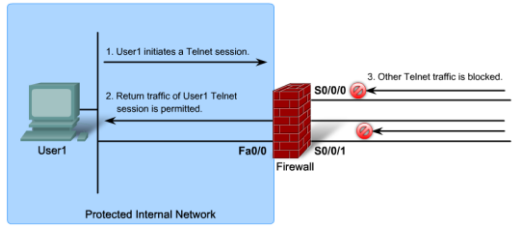
- A Cisco router running Cisco IOS Firewall is both a router and a firewall.
- If there are two firewalls, one design option is to join them with a LAN functioning as a DMZ.
- It also provides hosts in the untrusted public network redundant access to DMZ resources.



CBAC

CBAC

- Context-Based Access Control



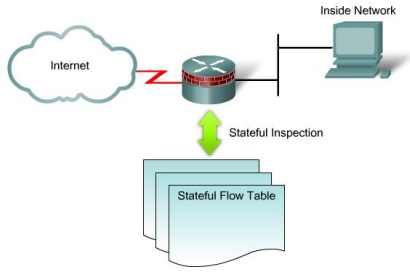
CBAC features

- Traffic Filtering
- Traffic Inspection
- Intrusion Detection
- Alert and Audit Generation

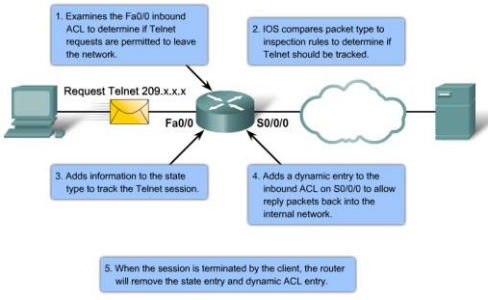
CBAC Capabilities

Tracks TCP Connection Setup Examines TCP Sequence Numbers
Monitors UDP Session Information
Inspects DNS Queries and Replies
Inspects Common ICMP Message Types
Supports Applications with Multiple Channels, such as FTP and Multimedia
Inspects Embedded Addresses
Inspects Application Layer Information

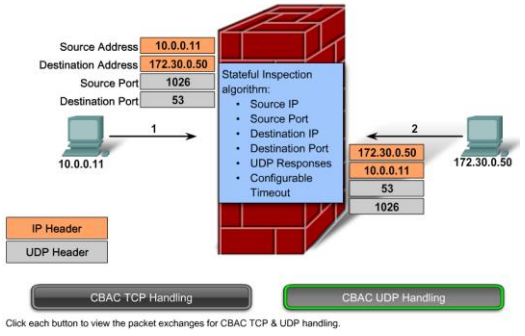
CBAC is statefull



How does it work?



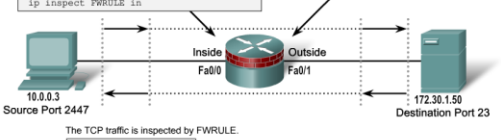
UDP and CBAC



Configuration

```
ip inspect name FWRULE telnet
access-list 101 permit tcp any any eq 23
interface FastEthernet0/0
 ip access-group 101 in
 ip inspect FWRULE in
```

```
access-list 102 deny ip any any
interface FastEthernet0/1
 ip access-group 102 in
```



The TCP traffic is inspected by FWRULE.

1. `ip inpsect FWRULE in`

The firewall creates a dynamic ACL allowing return traffic back through the firewall.

2. `access-list 102 permit tcp host 172.30.1.50 eq 23 host 10.0.0.3 eq 2447`

3. The firewall continues to inspect control traffic and dynamically creates and removes ACLs as required by the application. It also monitors and protects against application-specific attacks.

4. The firewall detects when an application terminates or times out and removes all dynamic ACLs for that session.

ip inspect ….

```
Router(config)#
ip inspect name inspection_name protocol [alert {on | off}] [audit-trail {on | off}]
[timeout seconds]
```

Parameter	Description
inspection_name	Names the set of inspection rules. If you want to add a protocol to an existing set of rules, use the same inspection name for the rules.
protocol	The protocol to inspect.
alert {on | off}	(Optional) For each inspected protocol, the generation of alert messages can be set to on or off. If no option is selected, alerts are generated based on the setting of the **ip inspect alert-off** command.
audit-trail {on | off}	(Optional) For each inspected protocol, the **audit-trail** option can be set to on or off. If no option is selected, **audit trail** messages are generated based on the setting of the **ip inspect audit-trail** command.
timeout *seconds*	(Optional) Specify the number of seconds for a different idle timeout to override the global TCP or UDP idle timeouts for the specified protocol. This timeout overrides the global TCP and UDP timeouts but does not override the global Domain Name Service (DNS) timeout.

Alerts and logs

```
%FW-4-SMTP_INVALID_COMMAND: Invalid SMTP command from
initiator(209.165.201.5:49387)
```

```
Router(config)#
no ip inspect alert-off
```

- Enables real time alerts.

```
%FW-6-SESS_AUDIT_TRAIL: tcp session initiator
(192.168.1.2:32782) sent 22 bytes responder (209.165.201.1:23)
sent 200 bytes
```

```
Router(config)#
ip inspect audit-trail
```

- Enables the delivery of audit trail messages using syslog.

show commands for CBAC

```
Router# show ip inspect name inspect_outbound
Inspection name inspect_outbound
cuseeme alert is on audit-trail is on timeout 3600
ftp alert is on audit-trail is on timeout 3600
http alert is on audit-trail is on timeout 3600
rcmd alert is on audit-trail is on timeout 3600
realaudio alert is on audit-trail is on timeout 3600
smtp max-data 20000000 alert is on audit-trail is on timeout
3600
tftp alert is on audit-trail is on timeout 30
udp alert is on audit-trail is on timeout 15
tcp alert is on audit-trail is on timeout 3600
```

```
Router# show ip inspect sessions
Established Sessions
Session 25A3378 (209.165.201.1:20)=>(192.168.1.2:32704) ftp-
data SIS_OPEN
Session 25A5AC2 (192.168.1.2:32703)=>(209.165.201.1:21) ftp
SIS_OPEN
Router# show ip access-list
Extended IP access list 100
permit tcp host 209.165.201.1 eq 21 host 192.168.1.2 eq 32703
(24 matches)
permit tcp host 209.165.201.1 eq 20 host 192.168.1.2 eq 32704
(88 matches)
<output omitted>
```


Zone-Based Policy Firewall

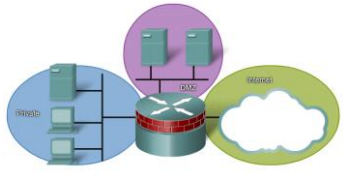
Benefits of ZPF

- Not dependent on ACLs.
- The router security posture is to block unless explicitly allowed.
- Policies are easy to read and troubleshoot with C3PL.
- One policy affects any given traffic, instead of needing multiple ACLs and inspection actions.

Zone-based policy firewall

- Zone-based policy firewall configuration model (ZPF or ZBF or ZFW) was introduced in 2006 with Cisco IOS Release 12.4(6)T.
- With ZPF the interfaces are assigned to zones and then an inspection policy is applied to traffic moving between the zones.
  - The default policy is to block all traffic unless explicitly allowed (CBACs default was allow all).
  - It supports previous firewall features, including SPI, application inspection, URL filtering, and DoS mitigation.

Basic ZPF Zone Topology



- If a new interface is added to the Private zone, the hosts on the new interface can pass traffic to all hosts in the Private zone.
- The new interface also inherits all existing Private zone policies when passing traffic to other zones.

CBAC or ZPF?

- Both CBAC and zones can be enabled concurrently on a router, just not on the same interface.
- For example, an interface cannot be configured as a security zone member and configured for IP inspection simultaneously.

3 Actions of ZPF

- Inspect
  - Configures Cisco IOS SPI (equivalent to ip inspect command).
  - It automatically allows for return traffic and potential ICMP messages.
  - For protocols requiring multiple parallel signaling and data sessions (for example, FTP or H.323), the inspect action also handles the proper establishment of data sessions.
- Pass
  - Analogous to a permit statement in an ACL.
  - It does not track the state of connections or sessions within the traffic.
  - Pass allows the traffic only in one direction.
  - A corresponding policy must be applied to allow return traffic to pass in the opposite direction.
- Drop
  - Analogous to a deny statement in an ACL.
  - A log option is available to log the rejected packets.

ZPF Rules

- A zone must be configured before it can be assigned to a zone.
- We can assign an interface to only one security zone.
- If traffic is to flow between all interfaces in a router, each interface must be a member of a zone.
- Traffic is implicitly allowed to flow by default among interfaces that are members of the same zone.
- To permit traffic to and from a zone member interface, a policy allowing or inspecting traffic must be configured between that zone and any other zone.
- Traffic cannot flow between a zone member interface and any interface that is not a zone member.
- We can apply pass, inspect, and drop actions only between two zones.
- Interfaces that have not been assigned to a zone function can still use a CBAC stateful packet inspection configuration.
- If we do not want an interface to be part of the zone-based firewall policy, it might still be necessary to put that interface in a zone and configure a pass-all policy (also known as a dummy policy) between that zone and any other zone to which traffic flow is desired.

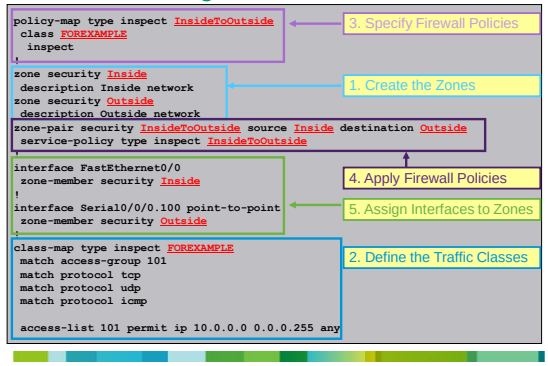
The Self Zone

- The ZPF rules for a zone-based policy firewall are different when the router is the source or the destination of the traffic.
  - When an interface is configured to be a zone member, the hosts that are connected to the interface are included in the zone.
  - However, traffic to the router is not subject to the zone policies.
  - By default, all router IP interfaces are part of the self zone.
- A zone-pair that includes the self zone and associated policy, applies to router generated or traffic destined to the router.
  - It does not apply to traffic traversing the router.
- A policy can be defined using the self zone as either the source or the destination zone.
  - The self zone is a system-defined zone.
  - It does not require any interfaces to be configured as members.

Configuring ZPF

1. Create the Zones for the firewall.
   - `zone security`
2. Define Traffic Classes.
   - `class-map type inspect`
3. Specify Firewall Policies.
   - `policy-map type inspect`
4. Apply Firewall Policies to pairs of source and destination zones.
   - `zone-pair`
5. Assign Router Interfaces to zones.
   - `zone-member security`

Final ZPF Configuration

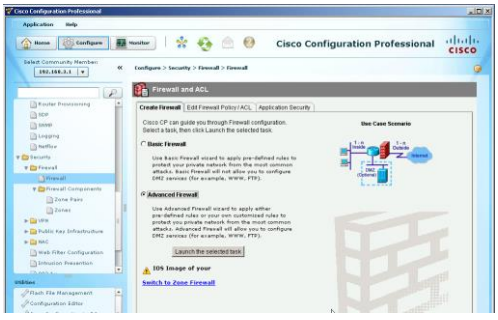


CCP Firewall Wizard

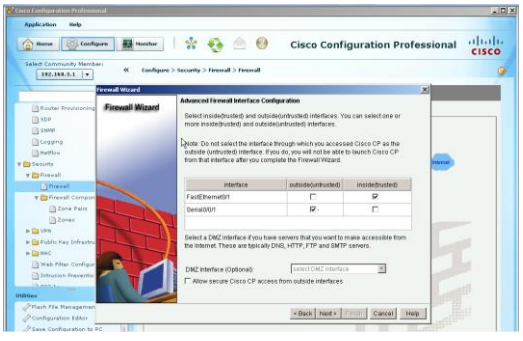
Configuring using the Basic Firewall Wizard

- ZPF can also be configured using the Basic Firewall Wizard.
  - Step 1. From Cisco CCP, choose **Configure > Security > Firewall**.
  - Step 2. In the Create Firewall tab, click the **Advanced Firewall** option and click **Launch the Selected Task** button.
  - Step 3. The Advanced Firewall Configuration Wizard window appears. Click **Next** to begin the configuration.

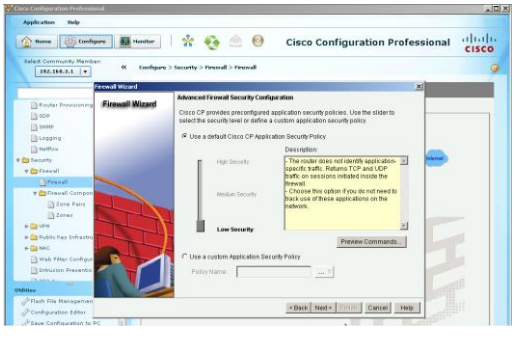
Basic Firewall Wizard



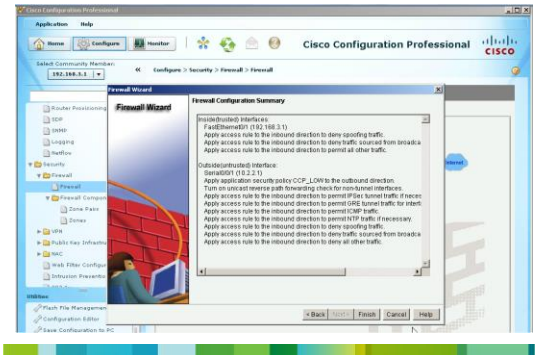
Define Interfaces



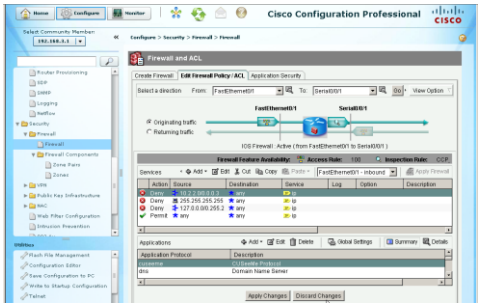
Select the Security Level



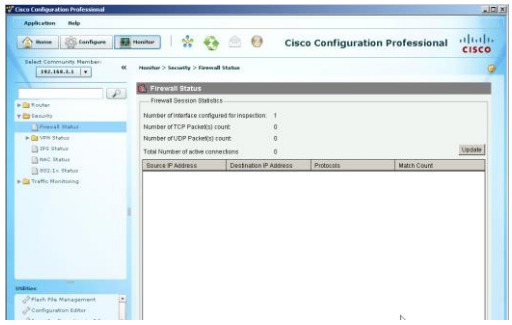
Review the Configuration



Verify the Configuration



Monitor the Firewall

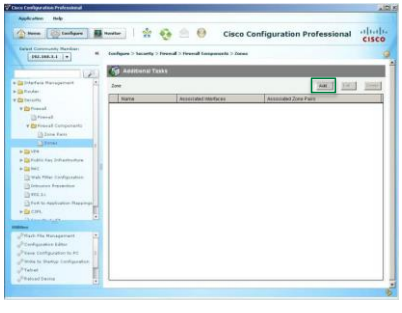


Manual ZPF using CCP

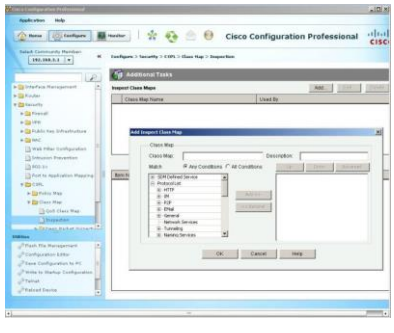
Configuring ZPF using CCP

- There are four steps to configure ZPF with CCP:
  - Step 1. Define zones.
  - Step 2. Configure class maps to describe traffic between zones.
  - Step 3. Create policy maps to apply actions to the traffic of the class maps.
  - Step 4. Define zone pairs and assign policy maps to the zone pairs.
- Unlike the CCP Basic Firewall Wizard, with manual CCP ZPF configuration, zones, zone pairs, traffic classification, policy maps, and application of the various elements are performed independently.

Define Zones

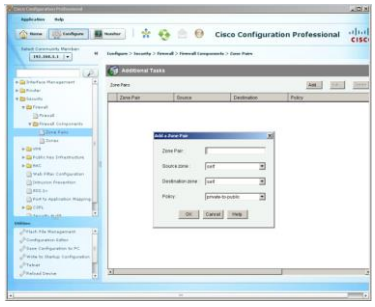


Configure class maps

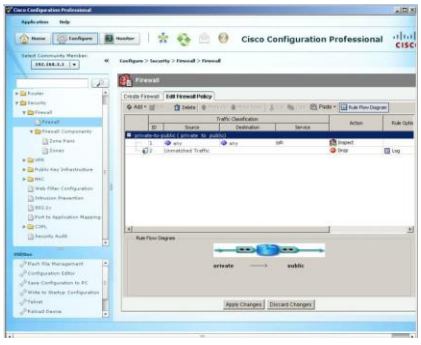


Create policy maps

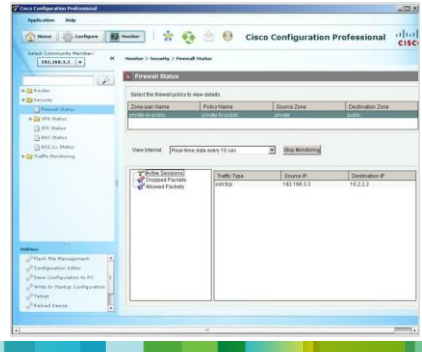
Define zone pairs



View Firewall Policy



Monitor Active Sessions



View Dropped Packets