1. Wildcard mask match 1 host
   Vd: Tính wildcard mask match host 192.168.1.1
   Theo nguyên tắc: bit 0 kiểm tra – bit 1 bỏ qua
   -> Địa chỉ IP: 192.168.1.1 0.0.0.0 hoặc từ khóa "host"

2. Wildcard mask match tất cả địa chỉ IP
Vd: Tính wildcard mask match tất cả địa chỉ IP
Theo nguyên tắc: bit 0 kiểm tra – bit 1 bỏ qua
-> Địa chỉ IP: 192.168.1.1 255.255.255.255 hoặc từ khóa "any"

3. Wildcard mask match 1 subnet
Vd: Tính wildcard mask match subnet 192.168.1.0/24
Cách tính: Lấy 255.255.255.255 trừ đi subnet mask của subnet
-> Địa chỉ IP: 192.168.1.1 0.0.0.255

4. Tính Wildcard mask match range địa chỉ IP liên tục
Vd: Tính wildcard mask match range từ 192.168.2.0 đến 192.168.4.255
Cách tính: Lấy địa chỉ cuối trừ địa chỉ đầu
-> Địa chỉ IP: 192.168.2.0 0.0.2.255

5. Tính widcard mask match 1 số IP add đầu tiên
Vd: Cho địa chỉ IP 192.168.1.0, tính wildcard mask match X host đầu tiên
-> Dải địa chỉ cần match: 192.168.1.0 - > 192.168.1.X
-> wildcard mask: 0.0.0.X (lấy địa chỉ cuối trừ địa chỉ đầu)
-> Địa chỉ IP: 192.168.1.0 0.0.0.X

6. Tính wildcard mask của nửa trên (upper half) hoặc nửa dưới (lower half) 1 dải mạng:
Vd: Cho địa chỉ IP 192.168.1.0, tính wildcard mask match nửa dải IP phía trên và dưới:
-> Dải địa chỉ nửa trên: 192.168.1.0 - > 192.168.1.127
-> wildcard mask: 0.0.0.127 (lấy địa chỉ cuối trừ địa chỉ đầu)
-> Địa chỉ IP: 192.168.1.0 0.0.0.127

-> Dải địa chỉ nửa dưới: 192.168.1.128 - > 192.168.1.255
-> wildcard mask: 0.0.0.127 (lấy địa chỉ cuối trừ địa chỉ đầu)
-> Địa chỉ IP: 192.168.1.128 0.0.0.127

7. Tính wildcard mask match IP lẻ, hoặc IP chẵn
1 địa chỉ Ip lẻ / chẵn là địa chỉ có octet cuối cùng dạng thập phân là số lẻ / chẵn
Vd: IP lẻ - 192.168.1.1
IP chẵn – 192.168.1.2
Nhận xét: bit cuối cùng của IP lẻ luôn là bit 1, bit cuối cùng của IP chẵn luôn là bit 0. Vậy wildcard mask thỏa mãn phải tạo ra một dải địa chỉ IP có bit cuối của octet cuối không đổi bằng 0 hoặc 1.
Giải pháp: để router luôn match bit cuối của octet cuối của địa chỉ IP, bit tương ứng trên wildcard mask phải là bit 0
-> Vd1: Cho địa chỉ IP: 192.168.1.0, tính wildcard mask match tất cả IP chẵn:
-> wildcard mask: 0.0.0.254 (dạng nhị phân: 00000000.00000000.00000000.11111110)
-> Địa chỉ IP: 192.168.1.0 0.0.0.254 (IP chẵn có bit cuối luôn bằng 0)

-> Vd2: Cho địa chỉ IP: 192.168.1.0, tính wildcard mask match tất cả IP lẻ
-> wildcard mask: 0.0.0.254 (dạng nhị phân: 00000000.00000000.00000000.11111110)
-> Địa chỉ IP: 192.168.1.1 0.0.0.254 (IP lẻ có bit cuối luôn bằng 1)

8. Tính wildcard mask match 1 range IP address không liên tục
Đây là dạng toán tính wildcard mask phức tạp nhất vì không có cách nào sử dụng 1 wildcard mask để tạo thành địa chỉ IP match tất cả dải IP ban đầu:
Vd: Tính wildcard mask match dải: 192.168.1.15 - > 192.168.1.75
Nhận xét: Đây là một dải IP không liên tục , không có 1 wildcard mask nào có thể thỏa mãn dải không liên tục. Tuy nhiên đối với những dải IP liên tục thì luôn có wildcard mask thỏa mãn.
Giải pháp: Chia dải IP ban đầu thành những dải nhỏ mà trong đó luôn tìm được 1 wildcard mask thỏa mãn mỗi dải. Vậy cách chia như thế nào? Nhắc lại: mỗi bit trong octet phần host đại diện cho một nhóm các host gọi là một block size. Bit cuối cùng là block size 1 vì nó thể hiện 1 host, tương tự bit đầu tiên là block size 128. Và, mỗi block size luôn tìm được 1 wildcard mask thỏa mãn.
Chia dải thành các block size:
- 192.168.1.15 (1)
- 192.168.1.16 - > 192.168.1.31 (2)
- 192.168.1.32 - > 192.168.1.63 (3)
- 192.168.1.64 -> 192.168.75 (4)
Tính wildcard mask cho mỗi block size:
- (1): 192.168.1.15 0.0.0.0 - > IP host
- (2): 192.168.1.16 0.0.0.15
- (3): 192.168.1.32 0.0.0.31
- (4): Chưa có wildcard mask phù hợp, ta phân tích dạng nhị phân octet cuối để tách tiếp wildcard mask:
.64: 01000000
.75: 01001011
-> Ta tách thành: 01000000 -> 01000111 (5)
01001000 -> 01001011 (6)
-> (5): 192.168.1.64 0.0.0.7
(6): 192.168.1.72 0.0.0.3
Tổng kết: Như vậy, từ dải IP ban đầu, ta tách thành 6 dải nhỏ (1)(2)(3)(4)(5)(6).

Introduction

This document describes how IP access control lists (ACLs) can filter network traffic. It also contains brief descriptions of the IP ACL types, feature availability, and an example of use in a network.

Access the Software Advisor (registered customers only) tool in order to determine the support of some of the more advanced Cisco IOS® IP ACL features.

RFC 1700 contains assigned numbers of well-known ports. RFC 1918 contains address allocation for private Internets, IP addresses which should not normally be seen on the Internet.

Note: ACLs might also be used for purposes other than to filter IP traffic, for example, defining traffic to Network Address Translate (NAT) or encrypt, or filtering non-IP protocols such as AppleTalk or IPX. A discussion of these functions is outside the scope of this document.

Prerequisites

Requirements

There are no specific prerequisites for this document. The concepts discussed are present in Cisco IOS® Software Releases 8.3 or later. This is noted under each access list feature.

Components Used

This document discusses various types of ACLs. Some of these are present since Cisco IOS Software Releases 8.3 and others were introduced in later software releases. This is noted in the discussion of each type.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

ACL Concepts

This section describes ACL concepts.

Masks

Masks are used with IP addresses in IP ACLs to specify what should be permitted and denied. Masks in order to configure IP addresses on interfaces start with 255 and have the large values on the left side, for example, IP address 209.165.202.129 with a 255.255.255.224 mask. Masks for

IP ACLs are the reverse, for example, mask 0.0.0.255. This is sometimes called an inverse mask or a wildcard mask. When the value of the mask is broken down into binary (0s and 1s), the results determine which address bits are to be considered in processing the traffic. A 0 indicates that the address bits must be considered (exact match); a 1 in the mask is a "don't care". This table further explains the concept.

Mask Example	
network address (traffic that is to be processed)	*10.1.1.0*
mask	*0.0.0.255*
network address (binary)	*00001010.00000001.00000001.00000000*
mask (binary)	*00000000.00000000.00000000.11111111*

Based on the binary mask, you can see that the first three sets (octets) must match the given binary network address exactly (00001010.00000001.00000001). The last set of numbers are "don't cares" (.11111111). Therefore, all traffic that begins with 10.1.1. matches since the last octet is "don't care". Therefore, with this mask, network addresses 10.1.1.1 through 10.1.1.255 (10.1.1.x) are processed.

Subtract the normal mask from 255.255.255.255 in order to determine the ACL inverse mask. In this example, the inverse mask is determined for network address 172.16.1.0 with a normal mask of 255.255.255.0.

- 255.255.255.255 - 255.255.255.0 (normal mask) = 0.0.0.255 (inverse mask)

Note these ACL equivalents.

- The source/source-wildcard of 0.0.0.0/255.255.255.255 means "any".
- The source/wildcard of 10.1.1.2/0.0.0.0 is the same as "host 10.1.1.2".

ACL Summarization

Note: Subnet masks can also be represented as a fixed length notation. For example, 192.168.10.0/24 represents 192.168.10.0 255.255.255.0.

This list describes how to summarize a range of networks into a single network for ACL optimization. Consider these networks.

```
192.168.32.0/24
192.168.33.0/24
```

```
192.168.34.0/24
192.168.35.0/24
192.168.36.0/24
192.168.37.0/24
192.168.38.0/24
192.168.39.0/24
```

The first two octets and the last octet are the same for each network. This table is an explanation of how to summarize these into a single network.

The third octet for the previous networks can be written as seen in this table, according to the octet bit position and address value for each bit.

Decimal	128	64	32	16	8	4	2	1
32	0	0	1	0	0	0	0	0
33	0	0	1	0	0	0	0	1
34	0	0	1	0	0	0	1	0
35	0	0	1	0	0	0	1	1
36	0	0	1	0	0	1	0	0
37	0	0	1	0	0	1	0	1
38	0	0	1	0	0	1	1	0
39	0	0	1	0	0	1	1	1
	M	M	M	M	M	D	D	D

Since the first five bits match, the previous eight networks can be summarized into one network (192.168.32.0/21 or 192.168.32.0 255.255.248.0). All eight possible combinations of the three low-order bits are relevant for the network ranges in question. This command defines an ACL that permits this network. If you subtract 255.255.248.0 (normal mask) from 255.255.255.255, it yields 0.0.7.255.

```
access-list acl_permit permit ip 192.168.32.0 0.0.7.255
```

Consider this set of networks for further explanation.

```
192.168.146.0/24
192.168.147.0/24
192.168.148.0/24
192.168.149.0/24
```

The first two octets and the last octet are the same for each network. This table is an explanation of how to summarize these.

The third octet for the previous networks can be written as seen in this table, according to the octet bit position and address value for each bit.

Decimal	128	64	32	16	8	4	2	1
146	1	0	0	1	0	0	1	0
147	1	0	0	1	0	0	1	1
148	1	0	0	1	0	1	0	0
149	1	0	0	1	0	1	0	1
	M	M	M	M	M	?	?	?

Unlike the previous example, you cannot summarize these networks into a single network. If they are summarized to a single network, they become 192.168.144.0/21 because there are five bits similar in the third octet. This summarized network 192.168.144.0/21 covers a range of networks from 192.168.144.0 to 192.168.151.0. Among these, 192.168.144.0, 192.168.145.0, 192.168.150.0, and 192.168.151.0 networks are not in the given list of four networks. In order to cover the specific networks in question, you need a minimum of two summarized networks. The given four networks can be summarized into these two networks:

- For networks 192.168.146.x and 192.168.147.x, all bits match except for the last one, which is a "don't care." This can be written as 192.168.146.0/23 (or 192.168.146.0 255.255.254.0).
- For networks 192.168.148.x and 192.168.149.x, all bits match except for the last one, which is a "don't care." This can be written as 192.168.148.0/23 (or 192.168.148.0 255.255.254.0).

This output defines a summarized ACL for the above networks.

```
!--- This command is used to allow access access for devices with IP
!--- addresses in the range from 192.168.146.0 to 192.168.147.254.

access-list 10 permit 192.168.146.0 0.0.1.255


!--- This command is used to allow access access for devices with IP
!--- addresses in the range from 192.168.148.0 to 192.168.149.254

access-list 10 permit 192.168.148.0 0.0.1.255
```

Process ACLs

Traffic that comes into the router is compared to ACL entries based on the order that the entries occur in the router. New statements are added to the end of the list. The router continues to look until it has a match. If no matches are found when the router reaches the end of the list, the traffic is denied. For this reason, you should have the frequently hit entries at the top of the list. There is an implied deny for traffic that is not permitted. A single-entry ACL with only one deny entry has the effect of denying all traffic. You must have at least one permit statement in an ACL or all traffic is blocked. These two ACLs (101 and 102) have the same effect.

```
!--- This command is used to permit IP traffic from 10.1.1.0
```

```
!--- network to 172.16.1.0 network. All packets with a source
!--- address not in this range will be rejected.

access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255

!--- This command is used to permit IP traffic from 10.1.1.0
!--- network to 172.16.1.0 network. All packets with a source
!--- address not in this range will be rejected.

access-list 102 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 102 deny ip any any
```

In this example, the last entry is sufficient. You do not need the first three entries because TCP includes Telnet, and IP includes TCP, User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

```
!--- This command is used to permit Telnet traffic
!--- from machine 10.1.1.2 to machine 172.16.1.1.

access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet

!--- This command is used to permit tcp traffic from
!--- 10.1.1.2 host machine to 172.16.1.1 host machine.

access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1

!--- This command is used to permit udp traffic from
!--- 10.1.1.2 host machine to 172.16.1.1 host machine.

access-list 101 permit udp host 10.1.1.2 host 172.16.1.1

!--- This command is used to permit ip traffic from
!--- 10.1.1.0 network to 172.16.1.10 network.

access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
```

Define Ports and Message Types

In addition to defining ACL source and destination, it is possible to define ports, ICMP message types, and other parameters. A good source of information for well-known ports is RFC 1700. ICMP message types are explained in RFC 792.

The router can display descriptive text on some of the well-known ports. Use a **?** for help.

```
access-list 102 permit tcp host 10.1.1.1 host 172.16.1.1 eq ?
  bgp           Border Gateway Protocol (179)
  chargen       Character generator (19)
  cmd           Remote commands (rcmd, 514)
```

During configuration, the router also converts numeric values to more user-friendly values. This is an example where you type the ICMP message type number and it causes the router to convert the number to a name.

```
access-list 102 permit icmp host 10.1.1.1 host 172.16.1.1 14
```

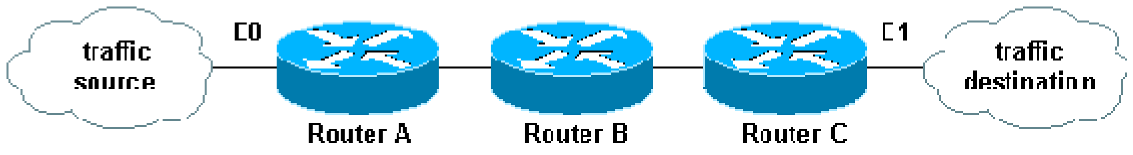
becomes

```
access-list 102 permit icmp host 10.1.1.1 host 172.16.1.1 timestamp-reply
```

Apply ACLs

You can define ACLs without applying them. But, the ACLs have no effect until they are applied to the interface of the router. It is a good practice to apply the ACL on the interface closest to the source of the traffic. As shown in this example, when you try to block traffic from source to destination, you can apply an inbound ACL to E0 on router A instead of an outbound list to E1 on router C. An access-list has a **deny ip any any** implicitly at the end of any access-list. If traffic is related to a DHCP request and if it is not explicity permitted, the traffic is dropped because when you look at DHCP request in IP, the source address is s=0.0.0.0 (Ethernet1/0), d=255.255.255.255, len 604, rcvd 2 UDP src=68, dst=67. Note that the source IP address is 0.0.0.0 and destination address is 255.255.255.255. Source port is 68 and destination 67. Hence, you should permit this kind of traffic in your access-list else the traffic is dropped due to implicit deny at the end of the statement.

Note: For UDP traffic to pass through, UDP traffic must also be permited explicitly by the ACL.



Define In, Out, Inbound, Outbound, Source, and Destination

The router uses the terms in, out, source, and destination as references. Traffic on the router can be compared to traffic on the highway. If you were a law enforcement officer in Pennsylvania and wanted to stop a truck going from Maryland to New York, the source of the truck is Maryland and the destination of the truck is New York. The roadblock could be applied at the Pennsylvania–New York border (out) or the Maryland–Pennsylvania border (in).

When you refer to a router, these terms have these meanings.

- **Out**—Traffic that has already been through the router and leaves the interface. The source is where it has been, on the other side of the router, and the destination is where it goes.
- **In**—Traffic that arrives on the interface and then goes through the router. The source is where it has been and the destination is where it goes, on the other side of the router.
- **Inbound** —If the access list is inbound, when the router receives a packet, the Cisco IOS software checks the criteria statements of the access list for a match. If the packet is permitted, the software continues to process the packet. If the packet is denied, the software discards the packet.
- **Outbound**—If the access list is outbound, after the software receives and routes a packet to the outbound interface, the software checks the criteria statements of the access list for a match. If the packet is permitted, the software transmits the packet. If the packet is denied, the software discards the packet.

The in ACL has a source on a segment of the interface to which it is applied and a destination off of any other interface. The out ACL has a source on a segment of any interface other than the interface to which it is applied and a destination off of the interface to which it is applied.

Edit ACLs

When you edit an ACL, it requires special attention. For example, if you intend to delete a specific line from a numbered ACL that exists as shown here, the entire ACL is deleted.

```
!--- The access-list 101 denies icmp from any to any network
!--- but permits IP traffic from any to any network.

  router#configure terminal
  Enter configuration commands, one per line.  End with CNTL/Z.
  router(config)#access-list 101 deny icmp any any
  router(config)#access-list 101 permit ip any any
  router(config)#^Z

  router#show access-list
  Extended IP access list 101
      deny icmp any any
      permit ip any any
  router#
  *Mar  9 00:43:12.784: %SYS-5-CONFIG_I: Configured from console by console

  router#configure terminal
  Enter configuration commands, one per line.  End with CNTL/Z.
  router(config)#no access-list 101 deny icmp any any
  router(config)#^Z

  router#show access-list
  router#
  *Mar  9 00:43:29.832: %SYS-5-CONFIG_I: Configured from console by console
```

Copy the configuration of the router to a TFTP server or a text editor such as Notepad in order to edit numbered ACLs. Then make any changes and copy the configuration back to the router.

You can also do this.

```
router#configure terminal
  Enter configuration commands, one per line.
  router(config)#ip access-list extended test

!--- Permits IP traffic from 2.2.2.2 host machine to 3.3.3.3 host machine.

  router(config-ext-nacl)#permit ip host 2.2.2.2 host 3.3.3.3

!--- Permits www traffic from 1.1.1.1 host machine to 5.5.5.5 host machine.

  router(config-ext-nacl)#permit tcp host 1.1.1.1 host 5.5.5.5 eq www

!--- Permits icmp traffic from any to any network.

  router(config-ext-nacl)#permit icmp any any
```

```
!--- Permits dns traffic from 6.6.6.6 host machine to 10.10.10.0 network.

  router(config-ext-nacl)#permit udp host 6.6.6.6 10.10.10.0 0.0.0.255 eq
domain
  router(config-ext-nacl)#^Z
  1d00h: %SYS-5-CONFIG_I: Configured from console by consoles-l

  router#show access-list
  Extended IP access list test
      permit ip host 2.2.2.2 host 3.3.3.3
      permit tcp host 1.1.1.1 host 5.5.5.5 eq www
      permit icmp any any
      permit udp host 6.6.6.6 10.10.10.0 0.0.0.255 eq domain
```

Any deletions are removed from the ACL and any additions are made to the end of the ACL.

```
router#configure terminal
   Enter configuration commands, one per line.  End with CNTL/Z.
   router(config)#ip access-list extended test

!--- ACL entry deleted.

   router(config-ext-nacl)#no permit icmp any any

!--- ACL entry added.

   router(config-ext-nacl)#permit gre host 4.4.4.4 host 8.8.8.8
   router(config-ext-nacl)#^Z
   1d00h: %SYS-5-CONFIG_I: Configured from console by consoles-l

   router#show access-list
   Extended IP access list test
       permit ip host 2.2.2.2 host 3.3.3.3
       permit tcp host 1.1.1.1 host 5.5.5.5 eq www
       permit udp host 6.6.6.6 10.10.10.0 0.0.0.255 eq domain
       permit gre host 4.4.4.4 host 8.8.8.8
```

You can also add ACL lines to numbered standard or numbered extended ACLs by sequence number in Cisco IOS. This is a sample of the configuration:

Configure the extended ACL in this way:

```
Router(config)#access-list 101 permit tcp any any
Router(config)#access-list 101 permit udp any any
Router(config)#access-list 101 permit icmp any any
Router(config)#exit
Router#
```

Issue the **show access-list** command in order to view the ACL entries. The sequence numbers such as 10, 20, and 30 also appear here.

```
Router#show access-list
Extended IP access list 101
    10 permit tcp any any
    20 permit udp any any
    30 permit icmp any any
```

Add the entry for the access list 101 with the sequence number 5.

Example 1:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#ip access-list extended 101
Router(config-ext-nacl)#5 deny tcp any any eq telnet
Router(config-ext-nacl)#exit
Router(config)#exit
Router#
```

In the **show access-list** command output, the sequence number 5 ACL is added as the first entry to the access-list 101.

```
Router#show access-list
Extended IP access list 101
    5 deny tcp any any eq telnet
    10 permit tcp any any
    20 permit udp any any
    30 permit icmp any any
Router#
```

Example 2:

```
internetrouter#show access-lists
Extended IP access list 101
    10 permit tcp any any
    15 permit tcp any host 172.162.2.9
    20 permit udp host 172.16.1.21 any
    30 permit udp host 172.16.1.22 any

internetrouter#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
internetrouter(config)#ip access-list extended 101
internetrouter(config-ext-nacl)#18 per tcp any host 172.162.2.11
internetrouter(config-ext-nacl)#^Z

internetrouter#show access-lists
Extended IP access list 101
    10 permit tcp any any
    15 permit tcp any host 172.162.2.9
    18 permit tcp any host 172.162.2.11
    20 permit udp host 172.16.1.21 any
    30 permit udp host 172.16.1.22 any
internetrouter#
```

Similarly, you can configure the standard access list in this way:

```
internetrouter(config)#access-list 2 permit 172.16.1.2
internetrouter(config)#access-list 2 permit 172.16.1.10
internetrouter(config)#access-list 2 permit 172.16.1.11

internetrouter#show access-lists
Standard IP access list 2
    30 permit 172.16.1.11
    20 permit 172.16.1.10
    10 permit 172.16.1.2

internetrouter(config)#ip access-list standard 2
internetrouter(config-std-nacl)#25 per 172.16.1.7
internetrouter(config-std-nacl)#15 per 172.16.1.16

internetrouter#show access-lists
Standard IP access list 2
    15 permit 172.16.1.16
    30 permit 172.16.1.11
    20 permit 172.16.1.10
    25 permit 172.16.1.7
    10 permit 172.16.1.2
```

The major difference in a standard access list is that the Cisco IOS adds an entry by descending order of the IP address, not on a sequence number.

This example shows the different entries, for example, how to permit an IP address (192.168.100.0) or the networks (10.10.10.0).

```
internetrouter#show access-lists
Standard IP access list 19
    10 permit 192.168.100.0
    15 permit 10.10.10.0, wildcard bits 0.0.0.255
    19 permit 201.101.110.0, wildcard bits 0.0.0.255
    25 deny any
```

Add the entry in access list 2 in order to permit the IP Address 172.22.1.1:

```
internetrouter(config)#ip access-list standard 2
internetrouter(config-std-nacl)#18 permit 172.22.1.1
```

This entry is added in the top of the list in order to give priority to the specific IP address rather than network.

```
internetrouter#show access-lists
Standard IP access list 19
    10 permit 192.168.100.0
    18 permit 172.22.1.1
    15 permit 10.10.10.0, wildcard bits 0.0.0.255
    19 permit 201.101.110.0, wildcard bits 0.0.0.255
    25 deny   any
```

Note: The previous ACLs are not supported in Security Appliance such as the ASA/PIX Firewall.

Guidelines to change access-lists when they are applied to crypto maps

- If you add to an existing access-list configuration, there is no need to remove the crypto map. If you add to them directly without the removal of the crypto map, then that is supported and acceptable.
- If you need to modify or delete access-list entry from an existing access-lists, then you must remove the crypto map from the interface. After you remove crypto map, make all

changes to the access-list and re-add the crypto map. If you make changes such as the deletion of the access-list without the removal of the crypto map, this is not supported and can result in unpredictable behavior.

Troubleshoot

How do I remove an ACL from an interface?

Go into configuration mode and enter **no** in front of the **access-group** command, as shown in this example, in order to remove an ACL from an interface.

```
interface <interface>
no ip access-group <acl-number> in|out
```

What do I do when too much traffic is denied?

If too much traffic is denied, study the logic of your list or try to define and apply an additional broader list. The **show ip access-lists** command provides a packet count that shows which ACL entry is hit.

The **log** keyword at the end of the individual ACL entries shows the ACL number and whether the packet was permitted or denied, in addition to port-specific information.

Note: The **log-input** keyword exists in Cisco IOS Software Release 11.2 and later, and in certain Cisco IOS Software Release 11.1 based software created specifically for the service provider market. Older software does not support this keyword. Use of this keyword includes the input interface and source MAC address where applicable.

How do I debug at the packet level that uses a Cisco router?

This procedure explains the debug process. Before you begin, be certain that there are no currently applied ACLs, that there is an ACL, and that fast switching is not disabled.

Note: Use extreme caution when you debug a system with heavy traffic. Use an ACL in order to debug specific traffic. But, be sure of the process and the traffic flow.

1. Use the **access-list** command in order to capture the desired data.

   In this example, the data capture is set for the destination address of 10.2.6.6 or the source address of 10.2.6.6.

   ```
   access-list 101 permit ip any host 10.2.6.6
   access-list 101 permit ip host 10.2.6.6 any
   ```

2. Disable fast switching on the interfaces involved. You only see the first packet if fast switching is not disabled.
   ```
   3. config interface
   4. no ip route-cache
   ```

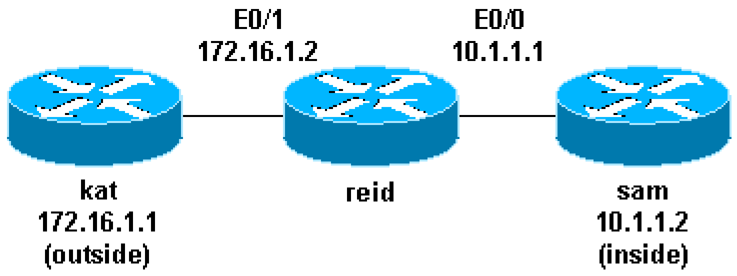
5. Use the **terminal monitor** command in enable mode in order to display **debug** command output and system error messages for the current terminal and session.
6. Use the **debug ip packet 101** or **debug ip packet 101 detail** command in order to begin the debug process.
7. Execute the **no debug all** command in enable mode and the **interface configuration** command in order to stop the debug process.
8. Restart caching.

```
9. config interface
10.   ip route-cache
```

Types of IP ACLs

This section of the document describes ACL types.

Network Diagram



Standard ACLs

Standard ACLs are the oldest type of ACL. They date back to as early as Cisco IOS Software Release 8.3. Standard ACLs control traffic by the comparison of the source address of the IP packets to the addresses configured in the ACL.

This is the command syntax format of a standard ACL.

```
access-list access-list-number {permit|deny}
{host|source source-wildcard|any}
```

In all software releases, the *access-list-number* can be anything from 1 to 99. In Cisco IOS Software Release 12.0.1, standard ACLs begin to use additional numbers (1300 to 1999). These additional numbers are referred to as expanded IP ACLs. Cisco IOS Software Release 11.2 added the ability to use list *name* in standard ACLs.

A *source/source-wildcard* setting of 0.0.0.0/255.255.255.255 can be specified as **any**. The wildcard can be omitted if it is all zeros. Therefore, host 10.1.1.2 0.0.0.0 is the same as host 10.1.1.2.

After the ACL is defined, it must be applied to the interface (inbound or outbound). In early software releases, out was the default when a keyword out or in was not specified. The direction must be specified in later software releases.

```
interface <interface>
ip access-group number {in|out}
```

This is an example of the use of a standard ACL in order to block all traffic except that from source 10.1.1.x.

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 1 in
access-list 1 permit 10.1.1.0 0.0.0.255
```

Extended ACLs

Extended ACLs were introduced in Cisco IOS Software Release 8.3. Extended ACLs control traffic by the comparison of the source and destination addresses of the IP packets to the addresses configured in the ACL.

This is the command syntax format of extended ACLs. Lines are wrapped here for spacing considerations.

IP

```
access-list access-list-number
     [dynamic dynamic-name [timeout minutes]]
     {deny|permit} protocol source source-wildcard
     destination destination-wildcard [precedence precedence]
     [tos tos] [log|log-input] [time-range time-range-name]
```

ICMP

```
access-list access-list-number
     [dynamic dynamic-name [timeout minutes]]
     {deny|permit} icmp source source-wildcard
     destination destination-wildcard
     [icmp-type [icmp-code] |icmp-message]
     [precedence precedence] [tos tos] [log|log-input]
     [time-range time-range-name]
```

TCP

```
access-list access-list-number
     [dynamic dynamic-name [timeout minutes]]
     {deny|permit} tcp source source-wildcard [operator [port]]
     destination destination-wildcard [operator [port]]
     [established] [precedence precedence] [tos tos]
     [log|log-input] [time-range time-range-name]
```

UDP

```
access-list access-list-number
     [dynamic dynamic-name [timeout minutes]]
     {deny|permit} udp source source-wildcard [operator [port]]
     destination destination-wildcard [operator [port]]
     [precedence precedence] [tos tos] [log|log-input]
```

```
    [time-range time-range-name]
```

In all software releases, the *access-list-number* can be 101 to 199. In Cisco IOS Software Release 12.0.1, extended ACLs begin to use additional numbers (2000 to 2699). These additional numbers are referred to as expanded IP ACLs. Cisco IOS Software Release 11.2 added the ability to use list *name* in extended ACLs.

The value of 0.0.0.0/255.255.255.255 can be specified as **any**. After the ACL is defined, it must be applied to the interface (inbound or outbound). In early software releases, out was the default when a keyword out or in was not specified. The direction must be specified in later software releases.

```
interface <interface>
ip access-group {number|name} {in|out}
```

This extended ACL is used to permit traffic on the 10.1.1.x network (inside) and to receive ping responses from the outside while it prevents unsolicited pings from people outside, permitting all other traffic.

```
interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0
ip access-group 101 in
access-list 101 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 101 permit ip any 10.1.1.0 0.0.0.255
```

Note: Some applications such as network management require pings for a keepalive function. If this is the case, you might wish to limit blocking inbound pings or be more granular in permitted/denied IPs.

Lock and Key (Dynamic ACLs)

Lock and key, also known as dynamic ACLs, was introduced in Cisco IOS Software Release 11.1. This feature is dependent on Telnet, authentication (local or remote), and extended ACLs.

Lock and key configuration starts with the application of an extended ACL to block traffic through the router. Users that want to traverse the router are blocked by the extended ACL until they Telnet to the router and are authenticated. The Telnet connection then drops and a single-entry dynamic ACL is added to the extended ACL that exists. This permits traffic for a particular time period; idle and absolute timeouts are possible.

This is the command syntax format for lock and key configuration with local authentication.

```
username user-name password password
interface <interface>
ip access-group {number|name} {in|out}
```

The single-entry ACL in this command is dynamically added to the ACL that exists after authentication.

```
access-list access-list-number dynamic name {permit|deny} [protocol]
{source source-wildcard|any} {destination destination-wildcard|any}
```

```
[precedence precedence][tos tos][established] [log|log-input]
[operator destination-port|destination port]

line vty line_range


login local
```

This is a basic example of lock and key.

```
username test password 0 test

!--- Ten (minutes) is the idle timeout.

username test autocommand access-enable host timeout 10


interface Ethernet0/0
  ip address 10.1.1.1 255.255.255.0
  ip access-group 101 in

access-list 101 permit tcp any host 10.1.1.1 eq telnet

!--- 15 (minutes) is the absolute timeout.

access-list 101 dynamic testlist timeout 15 permit ip 10.1.1.0 0.0.0.255
172.16.1.0 0.0.0.255


line vty 0 4
login local
```

After the user at 10.1.1.2 makes a Telnet connection to 10.1.1.1, the dynamic ACL is applied. The connection is then dropped, and the user can go to the 172.16.1.x network.

IP Named ACLs

IP named ACLs were introduced in Cisco IOS Software Release 11.2. This allows standard and extended ACLs to be given names instead of numbers.

This is the command syntax format for IP named ACLs.

```
ip access-list {extended|standard} name
```

This is a TCP example:

```
{permit|deny} tcp source source-wildcard [operator [port]]
destination destination-wildcard [operator [port]] [established]
[precedence precedence] [tos tos] [log] [time-range time-range-name]
```

This is an example of the use of a named ACL in order to block all traffic except the Telnet connection from host 10.1.1.2 to host 172.16.1.1.

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group in_to_out in

ip access-list extended in_to_out
permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
```

Reflexive ACLs

Reflexive ACLs were introduced in Cisco IOS Software Release 11.3. Reflexive ACLs allow IP packets to be filtered based on upper-layer session information. They are generally used to allow outbound traffic and to limit inbound traffic in response to sessions that originate inside the router.

Reflexive ACLs can be defined only with extended named IP ACLs. They cannot be defined with numbered or standard named IP ACLs, or with other protocol ACLs. Reflexive ACLs can be used in conjunction with other standard and static extended ACLs.

This is the syntax for various reflexive ACL commands.

```
interface
ip access-group {number|name} {in|out}

ip access-list extended name
permit protocol any any reflect name [timeoutseconds]
ip access-list extended name

evaluate name
```

This is an example of the permit of ICMP outbound and inbound traffic, while only permitting TCP traffic that has initiated from inside, other traffic is denied.

```
ip reflexive-list timeout 120

interface Ethernet0/1
 ip address 172.16.1.2 255.255.255.0
 ip access-group inboundfilters in
 ip access-group outboundfilters out

ip access-list extended inboundfilters
permit icmp 172.16.1.0 0.0.0.255 10.1.1.0 0.0.0.255
evaluate tcptraffic


!--- This ties the reflexive ACL part of the outboundfilters ACL,
!--- called tcptraffic, to the inboundfilters ACL.

ip access-list extended outboundfilters
permit icmp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
permit tcp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255 reflect tcptraffic
```

Time-Based ACLs Using Time Ranges

Time-based ACLs were introduced in Cisco IOS Software Release 12.0.1.T. While similar to extended ACLs in function, they allow for access control based on time. A time range is created that defines specific times of the day and week in order to implement time-based ACLs. The time range is identified by a name and then referenced by a function. Therefore, the time restrictions are imposed on the function itself. The time range relies on the router system clock. The router clock can be used, but the feature works best with Network Time Protocol (NTP) synchronization.

These are time-based ACL commands.

```
!--- Defines a named time range.

time-range time-range-name

!--- Defines the periodic times.

periodic days-of-the-week hh:mm to [days-of-the-week] hh:mm


!--- Or, defines the absolute times.

absolute [start time date] [end time date]

!--- The time range used in the actual ACL.

ip access-list name|number <extended_definition>time-rangename_of_time-range
```

In this example, a Telnet connection is permitted from the inside to outside network on Monday, Wednesday, and Friday during business hours:

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 101 in

access-list 101 permit tcp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
eq telnet time-range EVERYOTHERDAY

time-range EVERYOTHERDAY
periodic Monday Wednesday Friday 8:00 to 17:00
```

Commented IP ACL Entries

Commented IP ACL entries were introduced in Cisco IOS Software Release 12.0.2.T. Comments make ACLs easier to understand and can be used for standard or extended IP ACLs.

This is the commented name IP ACL command syntax.

```
ip access-list {standard|extended} access-list-name
remark remark
```

This is the commented numbered IP ACL command syntax.

```
access-list access-list-number remark remark
```

This is an example of commenting a numbered ACL.

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 101 in

access-list 101 remark permit_telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
```

Context-Based Access Control

Context-based access control (CBAC) was introduced in Cisco IOS Software Release 12.0.5.T and requires the Cisco IOS Firewall feature set. CBAC inspects traffic that travels through the firewall in order to discover and manage state information for TCP and UDP sessions. This state information is used in order to create temporary openings in the access lists of the firewall. Configure **ip inspect** lists in the direction of the flow of traffic initiation in order to allow return traffic and additional data connections for permissible session, sessions that originated from within the protected internal network, in order to do this.

This is the syntax for CBAC.

```
ip inspect name inspection-name protocol [timeoutseconds]
```

This is an example of the use of CBAC in order to inspect outbound traffic. Extended ACL 111 normally block the return traffic other than ICMP without CBAC opening holes for the return traffic.

```
ip inspect name myfw ftp timeout 3600
ip inspect name myfw http timeout 3600
ip inspect name myfw tcp timeout 3600
ip inspect name myfw udp timeout 3600
ip inspect name myfw tftp timeout 3600
interface Ethernet0/1
       ip address 172.16.1.2 255.255.255.0
       ip access-group 111 in
       ip inspect myfw out
access-list 111 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 111 permit icmp any 10.1.1.0 0.0.0.255
```

Authentication Proxy

Authentication proxy was introduced in Cisco IOS Software Release 12.0.5.T. This requires that you have the Cisco IOS Firewall feature set. Authentication proxy is used to authenticate inbound or outbound users, or both. Users who are normally blocked by an ACL can bring up a browser to go through the firewall and authenticate on a TACACS+ or RADIUS server. The server passes additional ACL entries down to the router in order to allow the users through after authentication.

Authentication proxy is similar to lock and key (dynamic ACLs). These are the differences:

- Lock and key is turned on by a Telnet connection to the router. Authentication proxy is turned on by HTTP through the router.
- Authentication proxy must use an external server.
- Authentication proxy can handle the addition of multiple dynamic lists. Lock and key can only add one.
- Authentication proxy has an absolute timeout but no idle timeout. Lock and key has both.

Refer to the Cisco Secure Integrated Software Configuration Cookbook for examples of authentication proxy.

Turbo ACLs

Turbo ACLs were introduced in Cisco IOS Software Release 12.1.5.T and are found only on the 7200, 7500, and other high-end platforms. The turbo ACL feature is designed in order to process ACLs more efficiently in order to improve router performance.

Use the **access-list compiled** command for turbo ACLs. This is an example of a compiled ACL.

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq ftp
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq syslog
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq tftp
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq ntp
```

After the standard or extended ACL is defined, use the **global configuration** command in order to compile.

```
!--- Tells the router to compile.

access-list compiled

Interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0

!--- Applies to the interface.

ip access-group 101 in
```

The **show access-list compiled** command shows statistics about the ACL.

Distributed Time-Based ACLs

Distributed time-based ACLs were introduced in Cisco IOS Software Release 12.2.2.T in order to implement time-based ACLs on VPN-enabled 7500 series routers. Before the introduction of the distributed time-based ACL feature, time-based ACLs were not supported on line cards for the Cisco 7500 series routers. If time-based ACLs were configured, they behaved as normal ACLs. If an interface on a line card was configured with time-based ACLs, the packets switched into the interface were not distributed switched through the line card but forwarded to the route processor in order to process.

The syntax for distributed time-based ACLs is the same as for time-based ACLs with the addition of the commands in regards to the status of the Inter Processor Communication (IPC) messages between the route processor and line card.

```
debug time-range ipc
show time-range ipc
clear time-range ipc
```

Receive ACLs

Receive ACLs are used in order to increase security on Cisco 12000 routers by the protection of the gigabit route processor (GRP) of the router from unnecessary and potentially nefarious traffic. Receive ACLs were added as a special waiver to the maintenance throttle for Cisco IOS Software Release 12.0.21S2 and integrated into 12.0(22)S. Refer to GSR: Receive Access Control Lists for further information.

Infrastructure Protection ACLs

Infrastructure ACLs are used in order to minimize the risk and effectiveness of direct infrastructure attack by the explicit permission of only authorized traffic to the infrastructure equipment while permitting all other transit traffic. Refer to Protecting Your Core: Infrastructure Protection Access Control Lists for further information.

Transit ACLs

Transit ACLs are used in order to increase network security since they explicitly permit only required traffic into your network or networks. Refer to Transit Access Control Lists: Filtering at Your Edge for further information.