Chapter 21: Troubleshooting ACLs and Prefix Lists

Instructor Materials

CCNP Enterprise: Advanced Routing

Chapter 21 Content

This chapter covers the following content:

- **Troubleshooting IPv4 ACLs -** This section examines how to read IPv4 ACLs so that you are more efficient at troubleshooting IPv4 ACL-related issues. It also shows the commands and processes you use for troubleshooting IPv4 packet filtering with standard, extended, and time-based IPv4 ACLs.
- **Troubleshooting IPv6 ACLs -** This section examines how to read IPv6 ACLs so that you are more efficient at troubleshooting IPv6 ACL-related issues. It also shows the commands and processes that you can use for troubleshooting IPv6 packet filtering.
- **Troubleshooting Prefix Lists -** This section reviews how to efficiently examine a prefix list for troubleshooting purposes so that when you are dealing with an issue that has a prefix list associated with it, you can determine whether the prefix list is or is not the problem.
- **Trouble Tickets -** This section provides trouble tickets that demonstrate how to use a structured troubleshooting process to solve a reported problem.

Troubleshooting IPv4 ACLs

- The purpose of an access control list is to identify traffic based on different criteria, such as source or destination IP address, source or destination port number, transport layer protocol, and quality of service (QoS) markings.
- IPv4 ACLS can be used to identify the private IP addresses that will be translated to a public address with Network Address Translation (NAT) and Port Address Translation (PAT). They can also be used to control which routes will be redistributed, which packets will be subject to policy-based routing, and which packets will be permitted or denied through the router.
- This section explains how to troubleshoot an IPv4 ACL to make sure it is correctly created for the purpose it is intended. The section also provides examples related to packet filtering.

Reading an IPv4 ACL

Being able to read an ACL and understand what it's created for is important for troubleshooting. Following this list of steps that IPv4 ACLs use to help you identify why an IPv4 ACL is behaving the way it is:

Step 1. **Top-down processing -** An ACL is made up of various entries. These entries are processed from the top of the ACL to the bottom of the ACL, in order.

Step 2. **Immediate execution upon a match -** The very first entry that matches the values in the packet that are being compared is the entry that is used. This may be a permit entry or a deny entry, and it dictates how the packet is treated based on the ACL implementation. It doesn't matter if there is another entry later in the ACL that matches; only the first entry that matches matters.

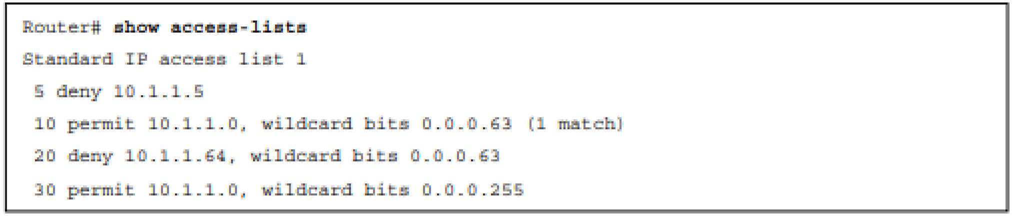
Step 3. **Implicit deny any -** If there is no matching entry for the packet, the packet is automatically denied based on the invisible implicit deny any entry at the end of an ACL.

Troubleshooting IPv4 ACLs

Standard Numbered and Extended Numbered ACLs

Example 21-1 shows a sample ACL with standard numbering that uses only source IPv4 addresses. In this example, the ACL is numbered 1 and has four entries. The entries are listed from most specific to least specific.

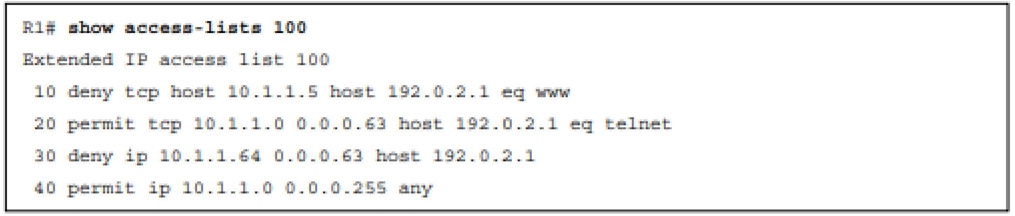
Example 21-1 *Sample Standard Numbered ACL*



```
Router# show access-lists
Standard IP access list 1
 5 deny 10.1.1.5
 10 permit 10.1.1.0, wildcard bits 0.0.0.63 (1 match)
 20 deny 10.1.1.64, wildcard bits 0.0.0.63
 30 permit 10.1.1.0, wildcard bits 0.0.0.255
```

Example 21-2 provides a sample extended numbered ACL. In this example, it is numbered 100. It has four entries, listed from most specific to least specific.

Example 21-2 *Sample Extended Numbered ACL*

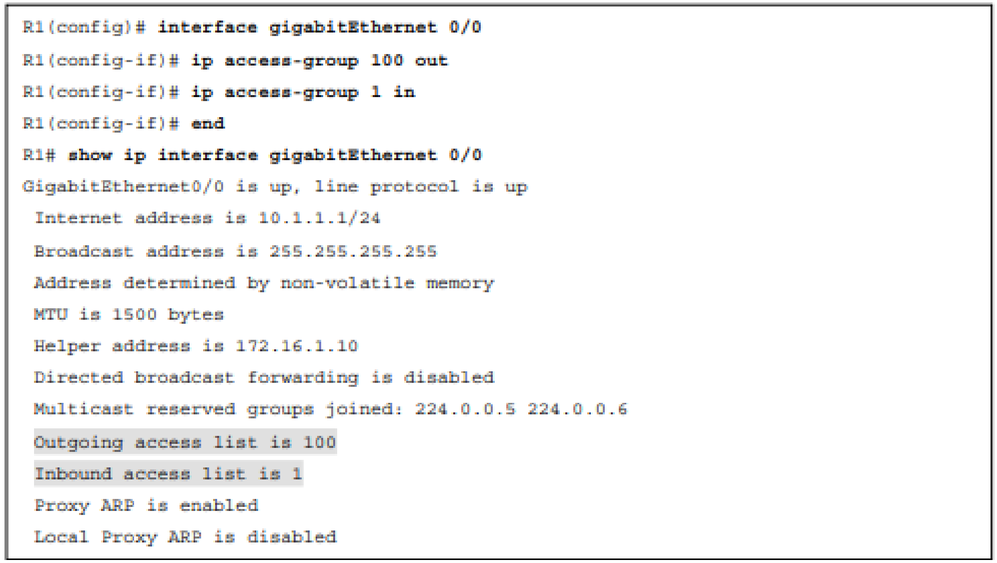


```
R1# show access-lists 100
Extended IP access list 100
 10 deny tcp host 10.1.1.5 host 192.0.2.1 eq www
 20 permit tcp 10.1.1.0 0.0.0.63 host 192.0.2.1 eq telnet
 30 deny ip 10.1.1.64 0.0.0.63 host 192.0.2.1
 40 permit ip 10.1.1.0 0.0.0.255 any
```

Troubleshooting IPv4 ACLs
Using an IPv4 ACL for Filtering

Using an ACL for packet filtering requires you to apply the ACL to an interface. Use the **ip access-group** {*acl_number* | *acl_name*} {**in** | **out**} command in interface configuration mode, as shown in Example 21-3. The direction in which you apply the ACL on an interface is significant. Verify the ACLs that are applied to an interface by using the **show ip interface** *interface_type interface_number* command. Example 21-3 shows how access list 1is applied inbound on GigabitEthernet0/0 and access list 100 is applied outbound on Gig0/0.

Example 21-3 *Verifying Access Lists Applied to Interfaces*



```
R1(config)# interface gigabitEthernet 0/0
R1(config-if)# ip access-group 100 out
R1(config-if)# ip access-group 1 in
R1(config-if)# end
R1# show ip interface gigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
 Internet address is 10.1.1.1/24
 Broadcast address is 255.255.255.255
 Address determined by non-volatile memory
 MTU is 1500 bytes
 Helper address is 172.16.1.10
 Directed broadcast forwarding is disabled
 Multicast reserved groups joined: 224.0.0.5 224.0.0.6
 Outgoing access list is 100
 Inbound access list is 1
 Proxy ARP is enabled
 Local Proxy ARP is disabled
```

Troubleshooting IPv4 ACLs

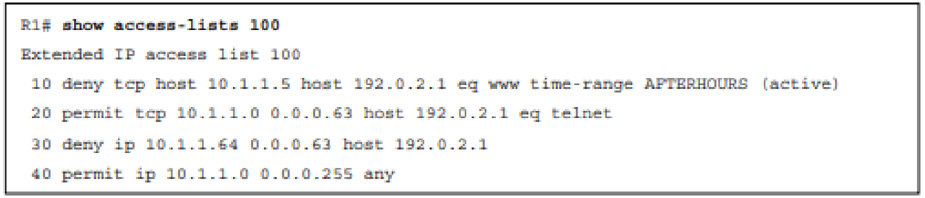
Using a Time-Based IPv4 ACL

Perhaps you want to allow traffic to the internet during open hours, but deny it after hours. To accomplish this, use time-based ACLs. Example 21-4 provides a sample time-based ACL. Notice that the ACL entry with sequence number 10 has the time-range option. The time range is based on values configured in the AFTERHOURS time range.

Example 21-5 shows the AFTERHOURS time range with the **show time-range AFTERHOURS** command. It has two weekdays entries, one from 5 p.m. to midnight and the other from midnight to 9 a.m. It also has a weekend entry that covers all day and all night. It also states that it is active and used in an ACL.

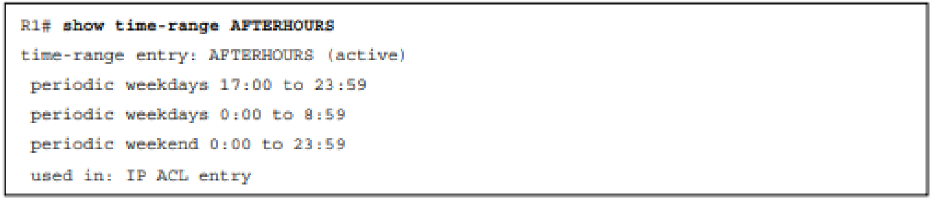
A time-based ACL is based on the device's clock. If the clock is not correct, the time-based ACL may be active or inactive at the wrong time. Example 21-6 shows how you can verify the current time on a router by using the show clock command.

Example 21-4 *Sample Time-Based ACL*



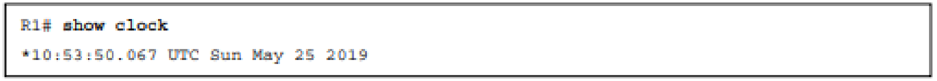
```
R1# show access-lists 100
Extended IP access list 100
 10 deny tcp host 10.1.1.5 host 192.0.2.1 eq www time-range AFTERHOURS (active)
 20 permit tcp 10.1.1.0 0.0.0.63 host 192.0.2.1 eq telnet
 30 deny ip 10.1.1.64 0.0.0.63 host 192.0.2.1
 40 permit ip 10.1.1.0 0.0.0.255 any
```

Example 21-5 *Sample Time Range Configured on R1*



```
R1# show time-range AFTERHOURS
time-range entry: AFTERHOURS (active)
 periodic weekdays 17:00 to 23:59
 periodic weekdays 0:00 to 8:59
 periodic weekend 0:00 to 23:59
 used in: IP ACL entry
```

Example 21-6 *Viewing the Time on a Cisco Router*



```
R1# show clock
*10:53:50.067 UTC Sun May 25 2019
```

Troubleshooting IPv6 ACLs

- IPv6 ACLs allow you to classify traffic for different reasons. For example, you might need to classify traffic that will be subject to policy-based routing, or you might need to classify the traffic that will be filtered as it passes through the router.
- IPv6 traffic filtering can be done on an interface-by-interface basis with IPv6 access lists.
- This section explains how to read IPv6 access lists so that you can troubleshoot them efficiently and identify whether they have been correctly applied to an interface for filtering purposes.

Reading an IPv6 ACL

Being able to read an IPv6 ACL and understand what it is created for is important for troubleshooting. IPv6 ACLS have four steps and steps 1, 2 and 4 are the same as steps 1, 2, and 3 for IPv4 ACLS.

- For IPv6 ACLS, **Step 3** is **Implicit permit icmp nd**: If the packet is an NA or NS message, permit it.
- IPv6 relies on the Neighbor Discovery Protocol (NDP) NA (neighbor advertisement) and NS (neighbor solicitation) messages to determine the MAC address associated with an IPv6 address. Therefore, the *implicit permit icmp nd* entries for NA and NS messages have been added before the implicit deny any, so they are not denied:

  permit icmp any any nd-na

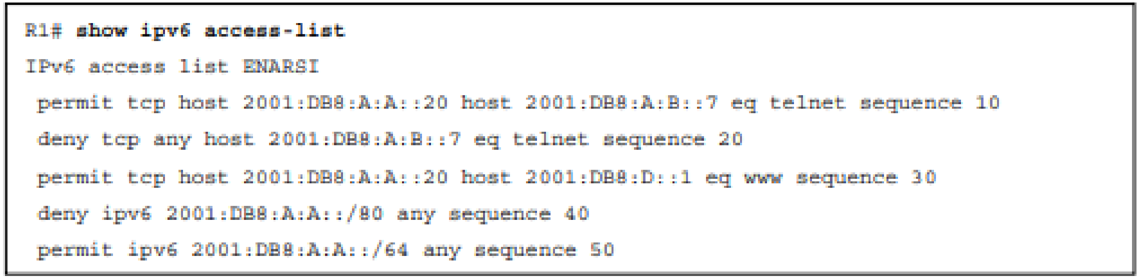
  permit icmp any any nd-ns
- Because these are implicit permit statements, all statically entered commands come before them. If you issue the **deny ipv6 any any log** command at the end of an IPv6 ACL, you will break the NDP process because NA and NS messages will be denied.

Troubleshooting IPv6 ACLs

Sample IPv6 ACL

- With IPv6, you have just one type of ACL which is similar to an IPv4 extended ACL. Within an IPv6 ACL entry, you provide as little or as much information as you need to accomplish your goal.
- Example 21-7 provides a sample IPv6 ACL that was created on R1. The IPv6 access list is named ENARSI, and you read it exactly as you read an IPv4 ACL.

Example 21-7 *Sample IPv6 ACL*



```
R1# show ipv6 access-list
IPv6 access list ENARSI
 permit tcp host 2001:DB8:A:A::20 host 2001:DB8:A:B::7 eq telnet sequence 10
 deny tcp any host 2001:DB8:A:B::7 eq telnet sequence 20
 permit tcp host 2001:DB8:A:A::20 host 2001:DB8:D::1 eq www sequence 30
 deny ipv6 2001:DB8:A:A::/80 any sequence 40
 permit ipv6 2001:DB8:A:A::/64 any sequence 50
```

Notice that there are no wildcard masks with IPv6. Instead, you specify a prefix, as shown in sequences 40 and 50 in Example 21-7, which accomplishes the same goal as the wildcard mask (defining a range of addresses).

Troubleshooting IPv6 ACLs

Using an IPv6 ACL for Filtering

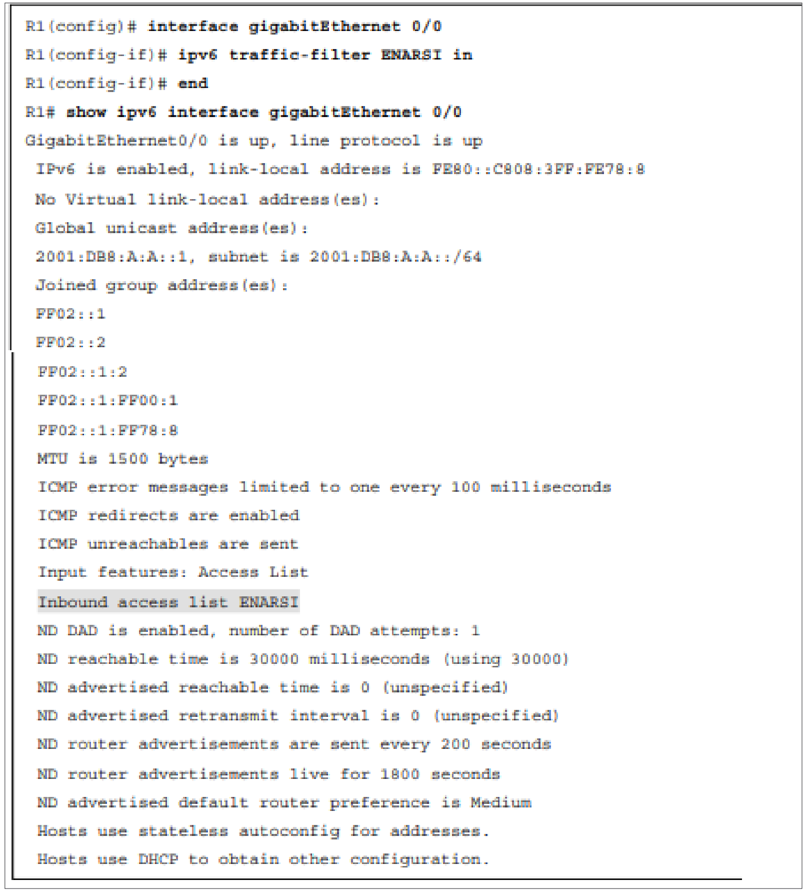
To use an IPv6 ACL for packet filtering, you need to apply the IPv6 ACL to an interface.

Use the **ipv6 traffic-filter** *acl_name* {**in** | **out**} command in interface configuration mode. The direction in which you apply the IPv6 ACL on an interface is significant.

Verify the IPv6 ACLs that are applied to an interface by using the **show ipv6 interface** *interface_type interface_number* command.

Example 21-8 shows the IPv6 ACL named ENARSI is applied inbound on interface Gig0/0.

Example 21-8 *Verifying IPv6 Access Lists Applied to Interfaces*



```
R1(config)# interface gigabitEthernet 0/0
R1(config-if)# ipv6 traffic-filter ENARSI in
R1(config-if)# end
R1# show ipv6 interface gigabitEthernet 0/0
GigabitEthernet0/0 is up, line protocol is up
 IPv6 is enabled, link-local address is FE80::C808:3FF:FE78:8
 No Virtual link-local address(es):
 Global unicast address(es):
 2001:DB8:A:A::1, subnet is 2001:DB8:A:A::/64
 Joined group address(es):
 FF02::1
 FF02::2
 FF02::1:2
 FF02::1:FF00:1
 FF02::1:FF78:8
 MTU is 1500 bytes
 ICMP error messages limited to one every 100 milliseconds
 ICMP redirects are enabled
 ICMP unreachables are sent
 Input features: Access List
 Inbound access list ENARSI
 ND DAD is enabled, number of DAD attempts: 1
 ND reachable time is 30000 milliseconds (using 30000)
 ND advertised reachable time is 0 (unspecified)
 ND advertised retransmit interval is 0 (unspecified)
 ND router advertisements are sent every 200 seconds
 ND router advertisements live for 1800 seconds
 ND advertised default router preference is Medium
 Hosts use stateless autoconfig for addresses.
 Hosts use DHCP to obtain other configuration.
```

Troubleshooting Prefix Lists

- ACLs do not give you granular control when matching routes for route filtering. Prefix lists allow you to define the route and prefix that you want to match. This section explains how to read a prefix list so that when you are troubleshooting features that call upon a prefix list, you will have the ability to eliminate the prefix list as the cause of the issue or prove that the prefix list is the cause of the issue.
- All the examples in this section are based on IPv4, but the content applies to both IPv4 prefix lists and IPv6 prefix lists.

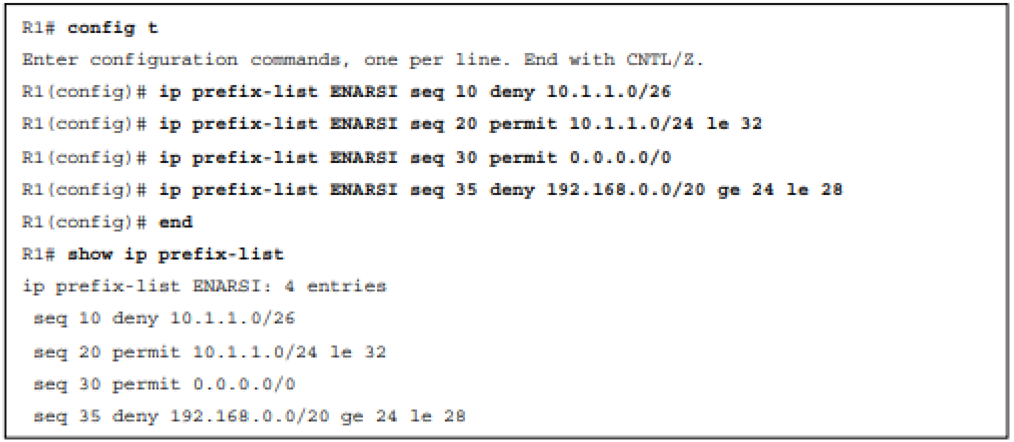
Troubleshooting Prefix Lists

Reading a Prefix List

Example 21-9 shows the commands used to create a sample prefix list called ENARSI and the output of show ip prefix-list, which you can use to verify the IPv4 prefix lists configured on a router.

To verify IPv6 prefix lists, you use the command **show ipv6 prefix-list**.

Example 21-9 *Sample IPv4 Prefix List*



```
R1# config t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# ip prefix-list ENARSI seq 10 deny 10.1.1.0/26
R1(config)# ip prefix-list ENARSI seq 20 permit 10.1.1.0/24 le 32
R1(config)# ip prefix-list ENARSI seq 30 permit 0.0.0.0/0
R1(config)# ip prefix-list ENARSI seq 35 deny 192.168.0.0/20 ge 24 le 28
R1(config)# end
R1# show ip prefix-list
ip prefix-list ENARSI: 4 entries
 seq 10 deny 10.1.1.0/26
 seq 20 permit 10.1.1.0/24 le 32
 seq 30 permit 0.0.0.0/0
 seq 35 deny 192.168.0.0/20 ge 24 le 28
```

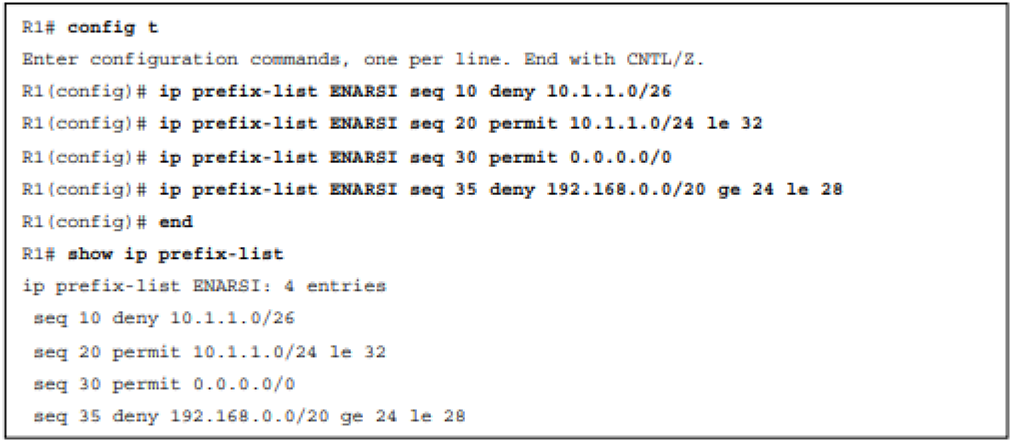
Troubleshooting Prefix Lists

Reading a Prefix List (Cont.)

There are two different ways to read a prefix list entry, depending on whether there is an **le** (less than or equal to) or **ge** (greater than or equal to) at the end of the prefix list entry:

- No **ge** or **le -** If the entry does not contain a **ge** or **le**, the prefix is treated as an address and a subnet mask. Refer to the entry with sequence number 10 in Example 21-9. There is no ge or le; therefore, the network 10.1.1.0/26 is matched exactly.
- There is a **ge** or **le -** If the entry does contain a **ge** or **le**, the prefix is treated as an address and a wildcard mask. Refer to the entry with sequence number 20 in Example 21-9. Because there is a **ge** or **le**, the entry is defining a range of values.

Example 21-9 *Sample IPv4 Prefix List*



```
R1# config t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# ip prefix-list ENARSI seq 10 deny 10.1.1.0/26
R1(config)# ip prefix-list ENARSI seq 20 permit 10.1.1.0/24 le 32
R1(config)# ip prefix-list ENARSI seq 30 permit 0.0.0.0/0
R1(config)# ip prefix-list ENARSI seq 35 deny 192.168.0.0/20 ge 24 le 28
R1(config)# end
R1# show ip prefix-list
ip prefix-list ENARSI: 4 entries
 seq 10 deny 10.1.1.0/26
 seq 20 permit 10.1.1.0/24 le 32
 seq 30 permit 0.0.0.0/0
 seq 35 deny 192.168.0.0/20 ge 24 le 28
```

Troubleshooting Prefix Lists

Prefix List Processing

The following is a list of steps that prefix lists use to help you identify why a prefix list is behaving the way it is:

Step 1. Top-down processing - A prefix list is made up of various sequences; these sequences are processed from the top of the prefix list to the bottom of the prefix list, in order of sequence number.

Step 2. **Immediate execution upon a match -** The very first sequence that matches is the sequence that is used. This may be a permit sequence or a deny sequence, and it dictates how the information is treated. It doesn't matter if there is another sequence later in the prefix list that matches. Only the first sequence that matches matters.

Step 3. **Implicit deny any -** If there is no matching sequence, the information is automatically denied based on the invisible implicit deny any entry at the end of a prefix list.

Because there is an implicit **deny any** at the end of a prefix list, you need at least one permit sequence in a prefix list, or everything will be denied. To permit everything that does not match a prefix list sequence, you need to include an entry that does so.

IPv4: **ip prefix-list NAME seq 30 permit 0.0.0.0/0 le 32**

IPv6: **ipv6 prefix-list NAME seq 30 permit ::/0 le 128**

Trouble Tickets

- This section presents various trouble tickets relating to the topics discussed in this chapter. The purpose of these trouble tickets is to show a process that you can follow when troubleshooting in the real world or in an exam environment.

Trouble Tickets

Trouble Ticket 21-1: IPv4 ACL

Problem: A user at PC1 (see Figure 21-1) has indicated that he cannot Telnet to 192.0.2.1, and he needs to. However, he can ping 192.0.2.1 and access web-enabled resources.

Refer to your text for next steps and examples to troubleshoot and resolve this trouble ticket.

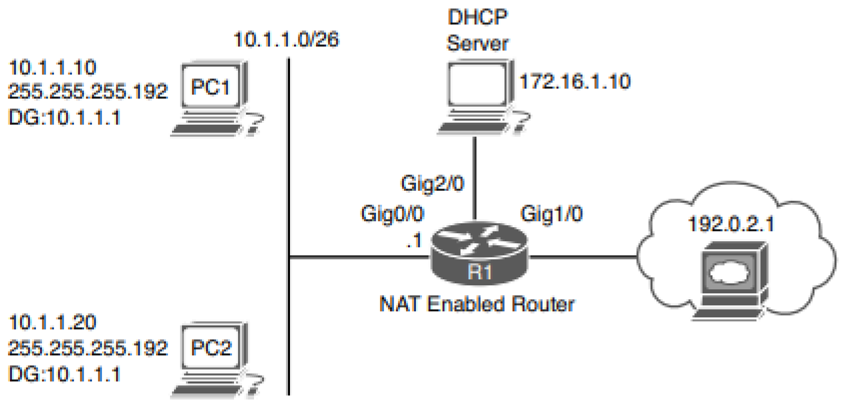


Figure 21-1 *IPv4 ACL Trouble Ticket Topology*

Trouble Tickets

Trouble Ticket 21-2: IPv6 ACL

Problem: A user at PC2 (see Figure 21-2) has indicated that she is not able to Telnet to 2001:db8:a:b::7, and she requires Telnet access. However, she can ping 2001:db8:a:b::7 and receive DHCP-related information from the DHCP server.

Refer to your text for next steps and examples to troubleshoot and resolve this trouble ticket.

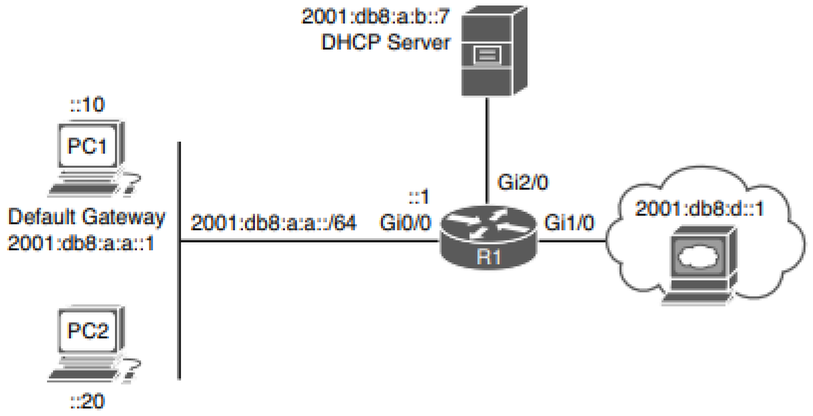


Figure 21-2 *IPv6 ACL Trouble Ticket Topology*

Trouble Tickets

Trouble Ticket 21-3: Prefix List

Problem: Your junior admin has contacted you indicating that R1 (see Figure 21-3) is not learning any routes from Enhanced Interior Gateway Routing Protocol (EIGRP). The admin has confirmed that neighbor relationships are being formed, interfaces are participating in the routing process, and other routers are learning about the routes. The admin has come to you for help. With your extensive knowledge, you ask your junior admin if he checked for any route filters. He says no.

Refer to your text for next steps and examples to troubleshoot and resolve this trouble ticket.

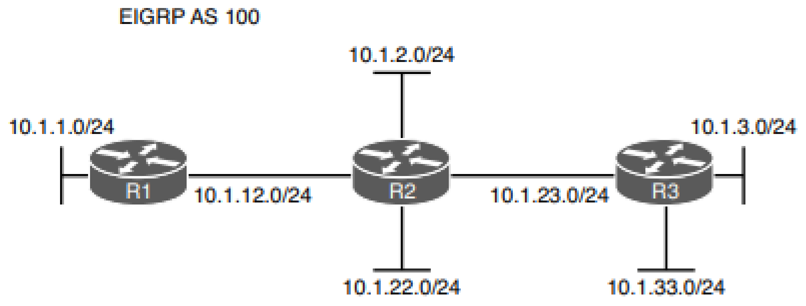


Figure 21-3 *IPv4 Prefix List Trouble Ticket Topology*

Prepare for the Exam

Prepare for the Exam

Key Topics for Chapter 21

Description	
The IPv4 ACL order of operations	Sample IPv6 ACL
Reading an IPv4 standard ACL	Verifying IPv6 access lists applied to interfaces
Reading an IPv4 extended ACL	Reading a prefix list
Verifying access lists applied to interfaces	The prefix list order of operations
Sample time-based ACL	Reading an IPv4 standard ACL
The IPv6 ACL order of operations	

Prepare for the Exam

Key Terms for Chapter 21

Term	
standard ACL	implicit deny
extended ACL	implicit permit
named ACL	prefix list
time-based ACL	ge
IPv6 ACL	le

Prepare for the Exam
Command Reference for Chapter 21

Task	Command Syntax
Display all the access lists configured on the device	**show access-lists**
Display all the IPv4 access lists configured on the devices	**show ip access-lists**
Display all the IPv6 access lists configured on the devices	**show ipv6 access-list**
Display the inbound and outbound IPv4 access lists applied to an interface	**show ip interface** *interface_type interface_number*
Display the inbound and outbound IPv6 access lists applied to an interface	**show ipv6 interface** *interface_type interface_number*
Display any time ranges that have been configured on the device	**show time-range**
Display the date and time on the device	**show clock**