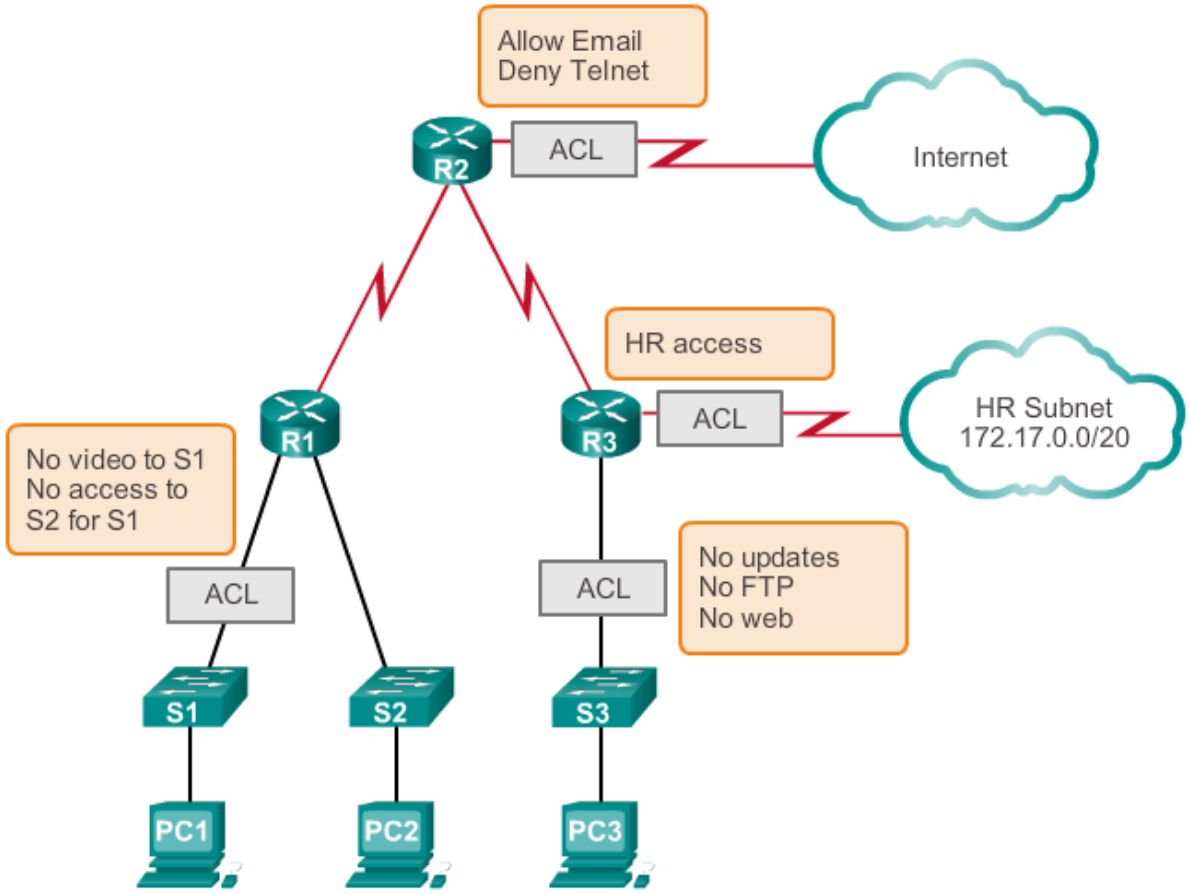
Chapter 9: Access Control Lists

Routing Protocols

Cisco | Networking Academy®
Mind Wide Open™

Purpose of ACLs

What is an ACL?



Purpose of ACLs

Packet Filtering

- Packet filtering, sometimes called static packet filtering, controls access to a network by analyzing the incoming and outgoing packets and passing or dropping them based on given criteria, such as the source IP address, destination IP addresses, and the protocol carried within the packet.
- A router acts as a packet filter when it forwards or denies packets according to filtering rules.
- An ACL is a sequential list of permit or deny statements, known as access control entries (ACEs).

ACL Operation

An inbound ACL filters packets coming into a specific interface and before they are routed to the outbound interface.

An outbound ACL filters packets after being routed, regardless of the inbound interface.

The last statement of an ACL is always an implicit deny. This statement is automatically inserted at the end of each ACL even though it is not physically present. The implicit deny blocks all traffic. Because of this implicit deny, an ACL **that does not have at least one permit statement will block all traffic.**

Types of Cisco IPv4 ACLs

Standard ACLs

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

Standard ACLs filter IP packets based on the source address only.

Extended ACLs

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

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

- Source and destination IP addresses
- Source and destination TCP and UDP ports
- Protocol type/ Protocol number (example: IP, ICP, UDP, TCP, etc.)

Standard versus Extended IPv4 ACLs

Numbering and Naming ACLs

Numbered ACL:

You assign a number based on which protocol you want filtered:

- (1 to 99) and (1300 and 1999): Standard IP ACL
- (100 to 199) and (2000 to 2699): Extended IP ACL

Named ACL:

You assign a name by providing the name of the ACL:

- Names can contain alphanumeric characters.
- It is suggested that the name be written in CAPITAL LETTERS.
- Names cannot contain spaces or punctuation.
- You can add or delete entries within the ACL.

Introducing ACL Wildcard Masking

Wildcard masks and subnet masks differ in the way they match binary 1s and 0s. Wildcard masks use the following rules to match binary 1s and 0s:

- Wildcard mask bit **0 - Match** the corresponding bit value in the address.
- Wildcard mask bit **1 - Ignore** the corresponding bit value in the address.

Wildcard masks are often referred to as an inverse mask. The reason is that, unlike a subnet mask in which binary 1 is equal to a match and binary 0 is not a match, in a wildcard mask the reverse is true.

Wildcard Mask Examples: Hosts / Subnets

Example 1

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.0.	00000000.00000000.00000000.00000000
Result	192.168.1.1	11000000.10101000.00000001.00000001

Example 2

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	255.255.255.255	11111111.11111111.11111111.11111111
Result	0.0.0.0	00000000.00000000.00000000.00000000

Example 3

	Decimal	Binary
IP Address	192.168.1.1	11000000.10101000.00000001.00000001
Wildcard Mask	0.0.0.255	00000000.00000000.00000000.11111111
Result	192.168.1.0	11000000.10101000.00000001.00000000

Calculating the Wildcard Mask

Calculating wildcard masks can be challenging. One shortcut method is to subtract the subnet mask from 255.255.255.255.

Example 1

```
  255.255.255.255
- 255.255.255.000
  000.000.000.255
```

Example 2

```
  255.255.255.255
- 255.255.255.240
  000.000.000.015
```

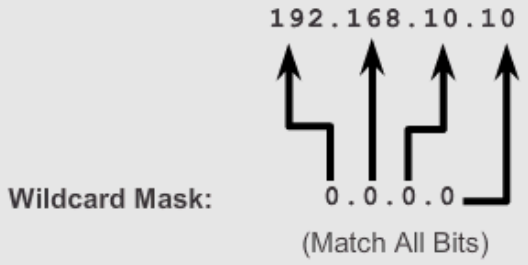
Example 3

```
  255.255.255.255
- 255.255.252.000
  000.000.003.255
```

Wildcard Mask Keywords

Example 1

- 192.168.10.10 0.0.0.0 matches all of the address bits
- Abbreviate this wildcard mask using the IP address preceded by the keyword **host** (**host 192.168.10.10**)



Example 2

- 0.0.0.0 255.255.255.255 ignores all address bits
- Abbreviate expression with the keyword **any**

Examples Wildcard Mask Keywords

Example 1:

```
R1(config)#access-list 1 permit 0.0.0.0 255.255.255.255
R1(config)#access-list 1 permit any
```

Example 2:

```
R1(config)#access-list 1 permit 192.168.10.10 0.0.0.0
R1(config)#access-list 1 permit host 192.168.10.10
```

Guidelines for ACL creation

General Guidelines for Creating ACLs

- Use ACLs in firewall routers positioned between your internal network and an external network such as the Internet.
- Use ACLs on a router positioned between two parts of your network to control traffic entering or exiting a specific part of your internal network.
- Configure ACLs on border routers, that is routers situated at the edges of your networks.
- Configure ACLs for each network protocol configured on the border router interfaces.

Guidelines for ACL creation

General Guidelines for Creating ACLs (cont.)

The Three Ps

- One ACL **per protocol** - To control traffic flow on an interface, an ACL must be defined for each protocol enabled on the interface.
- One ACL **per direction** - ACLs control traffic in one direction at a time on an interface. Two separate ACLs must be created to control inbound and outbound traffic.
- One ACL **per interface** - ACLs control traffic for an interface, for example, GigabitEthernet 0/0.

Guidelines for ACL Placement

Where to Place ACLs

Every ACL should be placed where it has the greatest impact on efficiency. The basic rules are:

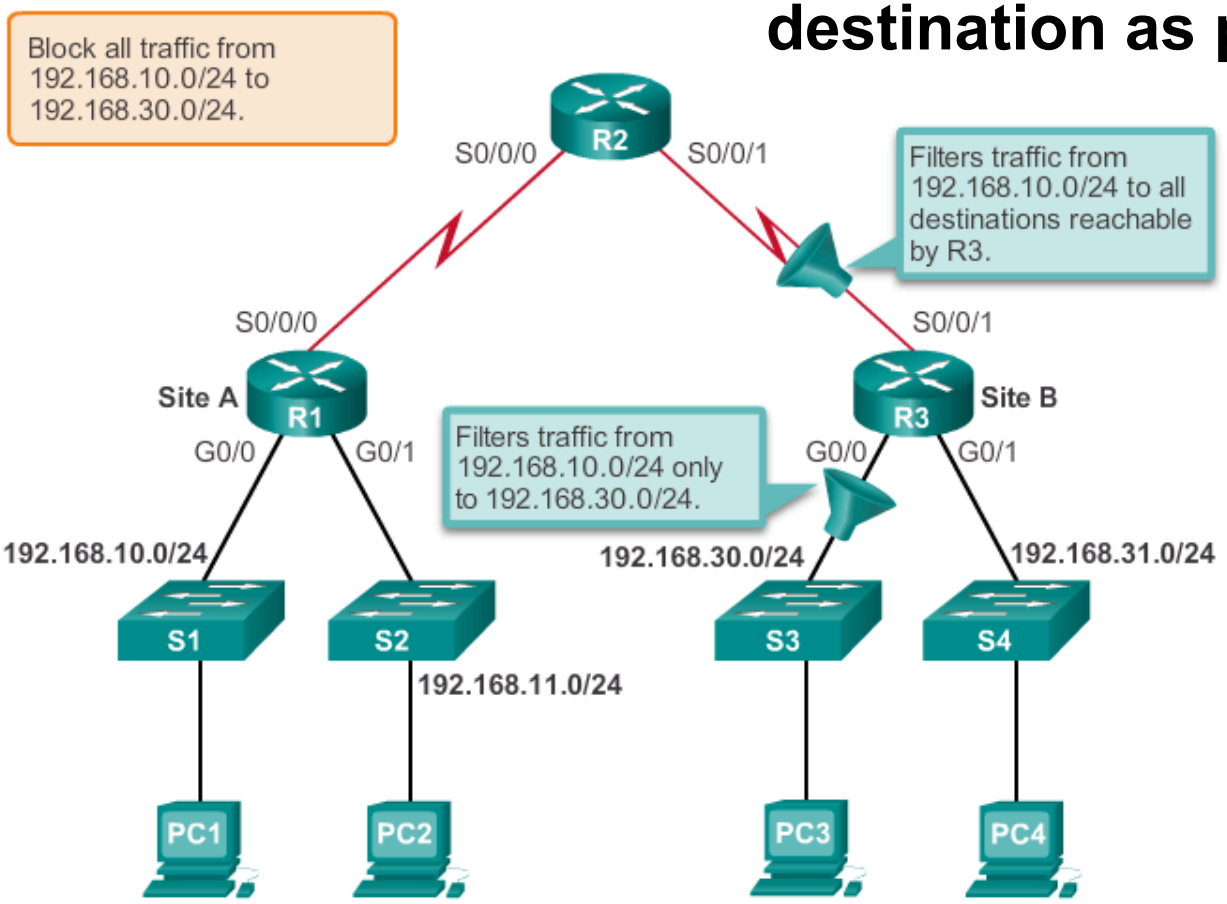
- Extended ACLs - **Locate extended ACLs as close as possible to the source of** the traffic to be filtered.
- Standard ACLs - **Because standard ACLs do not specify destination addresses, place them as close to the destination as possible**.

Placement of the ACL and therefore the type of ACL used may also depend on: the extent of the network administrator's control, bandwidth of the networks involved, and ease of configuration.

Guidelines for ACL Placement

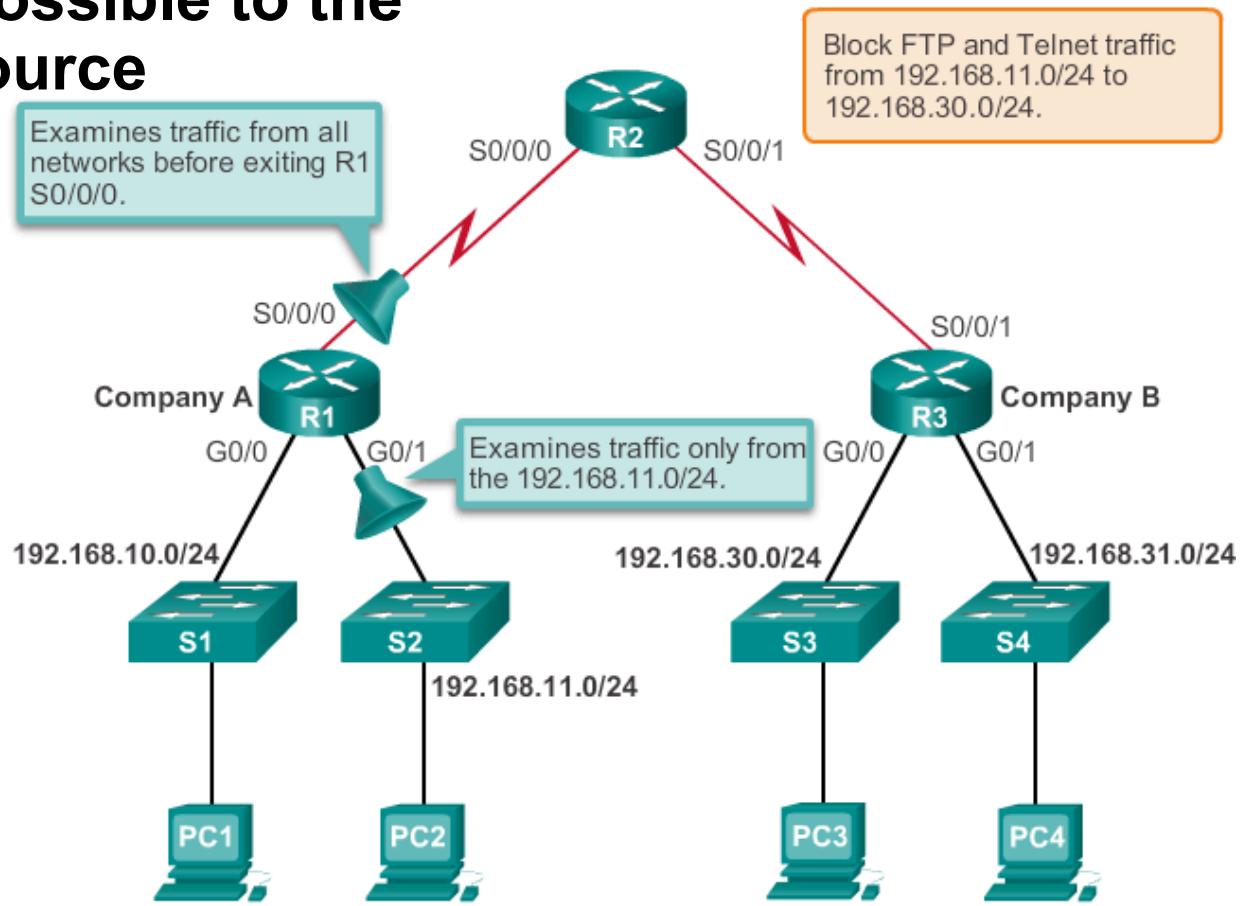
Standard ACL Placement

place them as close to the destination as possible.

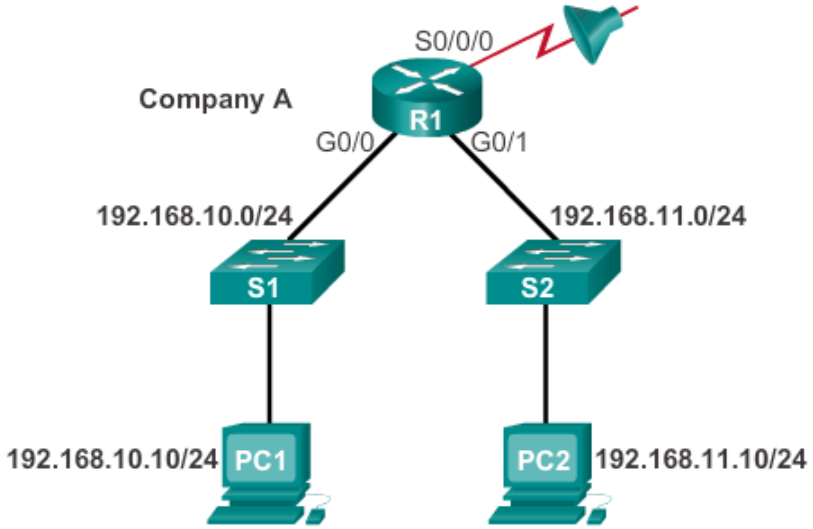


Extended ACL Placement

close as possible to the source



Entering Criteria Statements



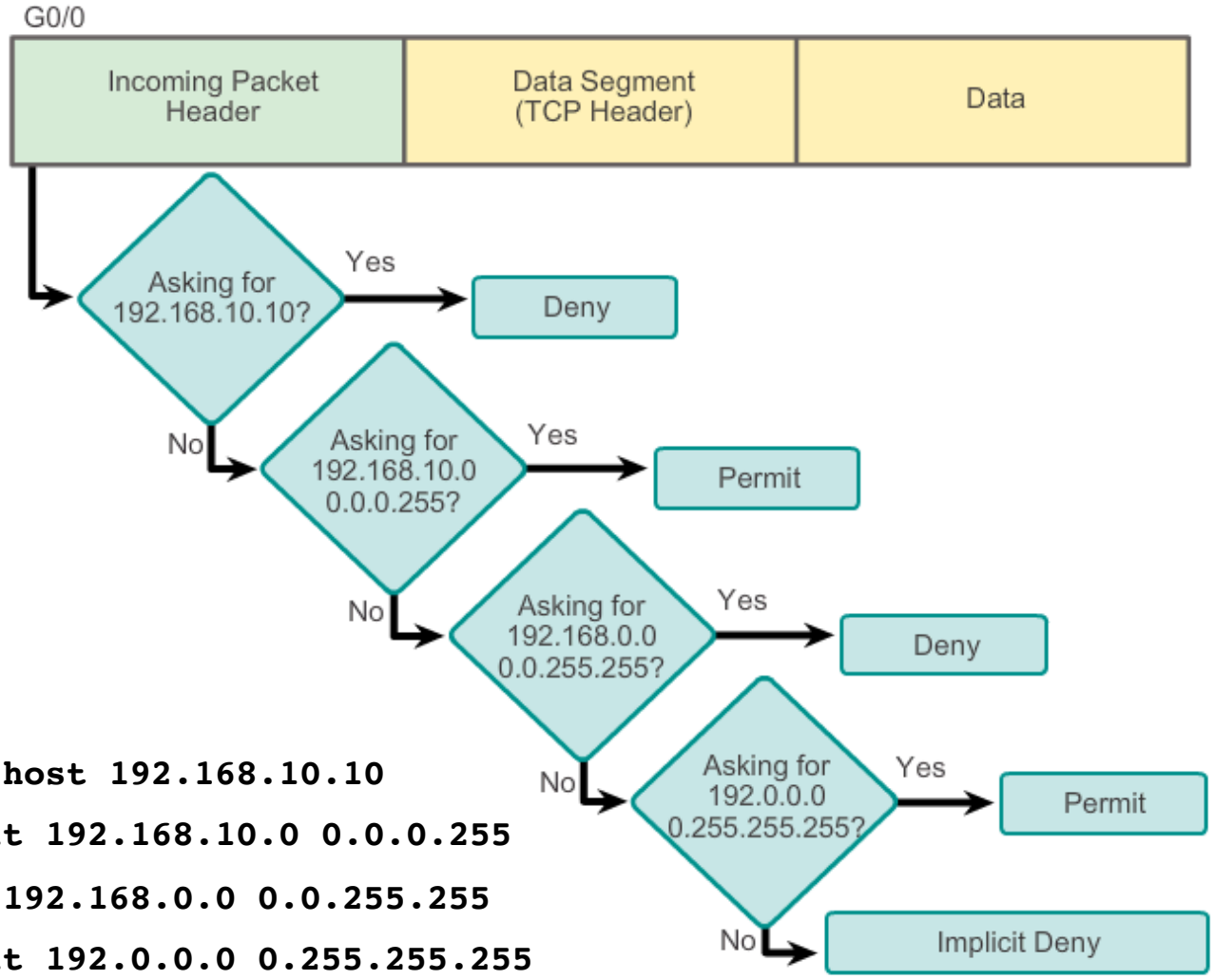
ACL 1

```
R1(config)#access-list 1 permit ip 192.168.10.0 0.0.0.255
```

ACL 2

```
R1(config)#access-list 2 permit ip 192.168.10.0 0.0.0.255
R1(config)#access-list 2 deny any
```

Configuring a Standard ACL



Example ACL

- `access-list 2 deny host 192.168.10.10`
- `access-list 2 permit 192.168.10.0 0.0.0.255`
- `access-list 2 deny 192.168.0.0 0.0.255.255`
- `access-list 2 permit 192.0.0.0 0.255.255.255`

Configuring a Standard ACL (cont.)

The full syntax of the standard ACL command is as follows:

```
Router config)# access-list access-list-number
deny permit remark source [ source-wildcard ] [
log ]
```

To remove the ACL, the global configuration **no access-list** command is used.

The **remark** keyword is used for documentation and makes access lists a great deal easier to understand.

Internal Logic

- Cisco IOS applies an internal logic when accepting and processing standard access list statements. As discussed previously, access list statements are **processed sequentially**. Therefore, the order in which statements are entered is important.

```
R1(config)#access-list 3 deny 192.168.10.0 0.0.0.255
R1(config)#access-list 3 permit host 192.168.10.10
% Access rule can't be configured at higher sequence num as
it is part of the existing rule at sequence num 10
R1(config)#
```

ACL 3: Host statement conflicts with previous range statement.

Applying Standard ACLs to Interfaces

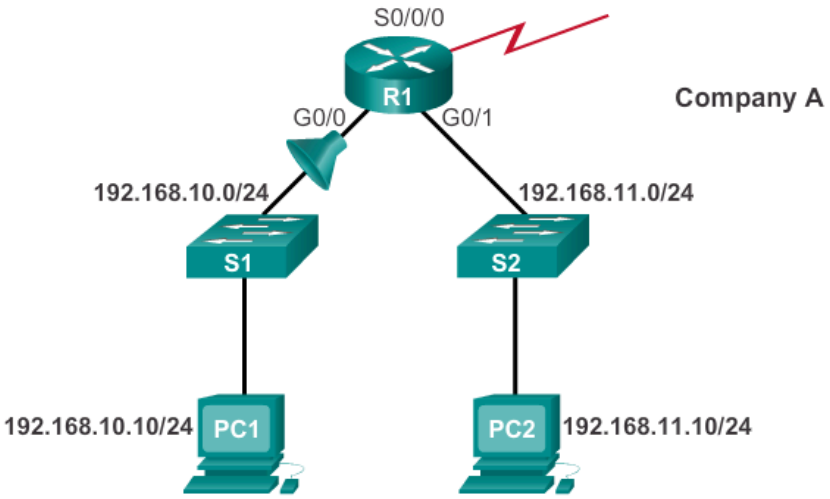
After a standard ACL is configured, it is linked to an interface using the **ip access-group** command in interface configuration mode:

```
Router(config-if)# ip access-group { access-list-number | access-list-name } { in | out }
```

To remove an ACL from an interface, first enter the **no ip access-group** command on the interface, and then enter the global **no access-list** command to remove the entire ACL.

Applying Standard ACLs to Interfaces (Cont.)

Deny a Specific Host




```
R1(config)#access-list 1 deny host 192.168.10.10
R1(config)#access-list 1 permit any
R1(config)#interface g0/0
R1(config-if)#ip access-group 1 in
```

Modify IPv4 ACLs

Verifying ACLs

```
R1# show ip interface s0/0/0
Serial0/0/0 is up, line protocol is up
  Internet address is 10.1.1.1/30
<output omitted>
  Outgoing access list is 1
  Inbound  access list is not set
<output omitted>

R1# show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 192.168.10.1/24
<output omitted>
  Outgoing access list is NO_ACCESS
  Inbound  access list is not set
<output omitted>
```

```
R1# show access-lists
Standard IP access list 1
    10 deny   192.168.10.10
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO_ACCESS
    15 deny   192.168.11.11
    10 deny   192.168.11.10
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Modify IPv4 ACLs

ACL Statistics

```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (4 match(es))
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO_ACCESS
    15 deny   192.168.11.11
    10 deny   192.168.11.10 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Output after pinging PC3 from PC1.

```
R1#show access-lists
Standard IP access list 1
    10 deny   192.168.10.10 (8 match(es))
    20 permit 192.168.0.0, wildcard bits 0.0.255.255
Standard IP access list NO_ACCESS
    15 deny   192.168.11.11
    10 deny   192.168.11.10 (4 match(es))
    20 permit 192.168.11.0, wildcard bits 0.0.0.255
R1#
```

Matches have been incremented.

Securing VTY ports with a Standard IPv4 ACL

Configuring a Standard ACL to Secure a VTY Port

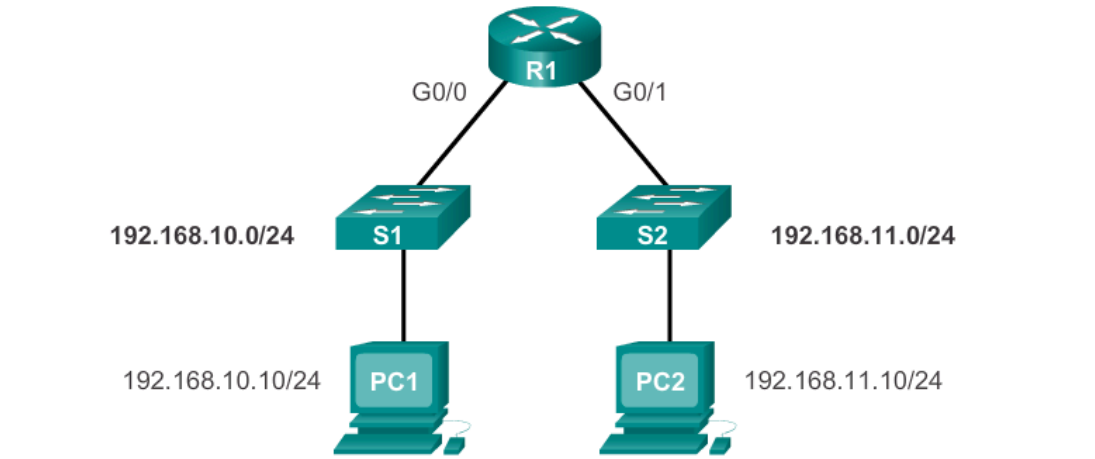
Filtering Telnet or SSH traffic is typically considered an extended IP ACL function because it filters a higher level protocol. However, because the **access-class** command is used to filter incoming or outgoing Telnet/SSH sessions by source address, a standard ACL can be used.

```
Router(config-line)# access-class access-list-number { in [ vrf-also ] | out }
```

Securing VTY ports with a Standard IPv4 ACL

Verifying a Standard ACL used to Secure a VTY Port

```
R1#show access-lists
Standard IP access list 21
    10 permit 192.168.10.0, wildcard bits 0.0.0.255 (2 matches)
    20 deny   any (1 match)
R1#
```




```
PC1>ssh 192.168.10.1

Login as: admin
Password: *****
R1>
```

```
PC2>ssh 192.168.11.1
ssh connect to host 192.168.11.1 port
22: Connection refused

PC2>
```

Structure of an Extended IPv4 ACL

Extended ACLs

Extended ACLs can filter on:

- Source address
- Destination address
- Protocol
- Port numbers

Structure of an Extended IPv4 ACL

Extended ACLs (Cont.)

Using Port Numbers

```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 23
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 21
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq 20
```

Using Keywords

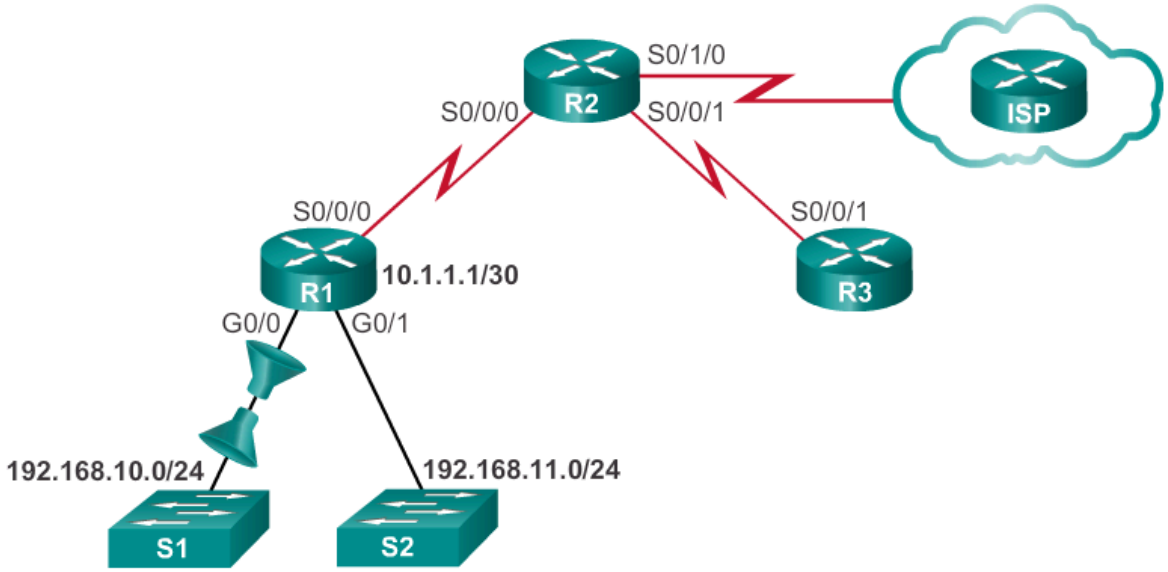
```
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq telnet
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp
access-list 114 permit tcp 192.168.20.0 0.0.0.255 any eq ftp-data
```

Configuring Extended ACLs

The procedural steps for configuring extended ACLs are the same as for standard ACLs. The extended ACL is first configured, and then it is activated on an interface. However, the command syntax and parameters are more complex to support the additional features provided by extended ACLs.

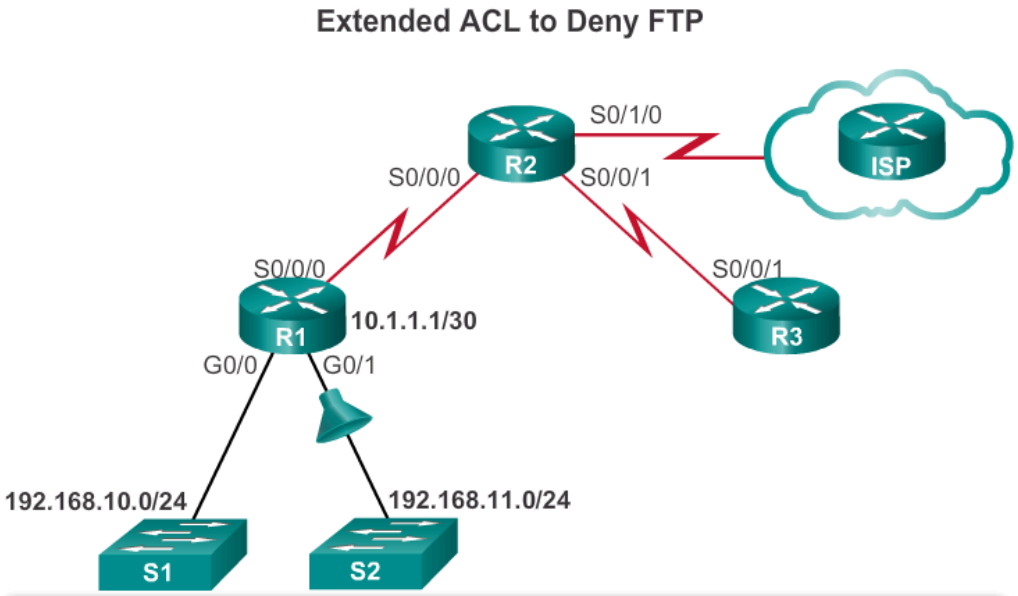
```
access-list access-list-number {deny | permit | remark}
protocol source [source-wildcard] [operator operand]
[port port-number or name] destination [destination-wildcard]
[operator operand] [port port-number or name][established]
```

Applying Extended ACLs to Interfaces




```
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config)#access-list 103 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config)#access-list 104 permit tcp any 192.168.10.0 0.0.0.255 established
R1(config)#interface g0/0
R1(config-if)#ip access-group 103 in
R1(config-if)#ip access-group 104 out
```

Filtering Traffic with Extended ACLs




```
R1(config)#access-list 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0
0.0.0.255 eq ftp
R1(config)#access-list 101 deny tcp 192.168.11.0 0.0.0.255 192.168.10.0
0.0.0.255 eq ftp-data
R1(config)#access-list 101 permit ip any any
R1(config)#interface g0/1
R1(config-if)#ip access-group 101 in
```

Verifying Extended ACLs

```
R1#show access-lists
Extended IP access list BROWSING
    10 permit tcp any 192.168.10.0 0.0.0.255 established
Extended IP access list SURFING
    10 permit tcp 192.168.10.0 0.0.0.255 any eq www
    20 permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1#
R1#show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 192.168.10.1/24
<output omitted for brevity>
  Outgoing access list is BROWSING
  Inbound  access list is SURFING
<output omitted for brevity>
```