Extended ACLs Introduction

- **Topic:** ACLs are topic 5.6 of the CCNA exam topics list ("configure and verify access control lists").
- **Comparison to Standard ACLs:** The purpose, operation, and application of extended ACLs are the same as standard ACLs.
- **Key Difference:** Extended ACLs can perform more specific matching than standard ACLs.
  - Standard ACLs can only match based on the **source IP address**.
- **Lecture Agenda:**
  - Configuring numbered ACLs (alternative method).
  - Editing ACLs (adding/removing specific entries).
  - Introducing extended numbered and named ACLs.

Alternative Configuration Method for Numbered ACLs

- **Traditional Numbered ACL Configuration:**

```
R1(config)# access-list 1 deny 192.168.1.1
R1(config)# access-list 1 permit any
```

  - Configured in **global config mode**.
  - Uses the `ACCESS-LIST` command directly.
  - Example: A simple ACL denying 192.168.1.1 but permitting all other traffic.

- **Traditional Named ACL Configuration:**

```
R1(config)# ip access-list standard BLOCK_PC1
R1(config-std-nacl)# deny 192.168.1.1
R1(config-std-nacl)# permit any
```

  - Configured with subcommands in a different config mode.
  - Uses the `IP ACCESS-LIST STANDARD` command to enter standard named ACL config mode.
  - Example: The same simple ACL (deny 192.168.1.1, permit all) configured as a named ACL.
- **Modern Alternative Method:**

```
R1(config)# ip access-list standard 1
R1(config-std-nacl)# deny 192.168.1.1
R1(config-std-nacl)# permit any
```

  - In modern Cisco IOS, numbered ACLs can be configured using the **exact same method as named ACLs**.
  - Example: A numbered ACL configured in the same way as the named ACL above.
- **Important Note:**
  - This is just a different way of configuring numbered ACLs.
  - If you check the ACL in the **running config**, it will be displayed as if it was configured using the traditional method (directly from global config mode).

Demonstration: Why Use Named ACL Config Mode for Numbered ACLs?

```
R1(config)#ip access-list standard ?
  <1-99>        Standard IP access-list number
  <1300-1999>   Standard IP access-list number (expanded range)
  WORD          Access-list name

R1(config)#ip access-list standard 1
R1(config-std-nacl)#deny 192.168.1.1
R1(config-std-nacl)#permit any
R1(config-std-nacl)#
R1(config-std-nacl)#do show running-config | section access-list
access-list 1 deny   192.168.1.1
access-list 1 permit any
R1(config-std-nacl)#
```

- **Demonstration Setup:**
  - From global config mode, the professor entered `IP ACCESS-LIST STANDARD` and checked the options.
  - Both numbered and named ACLs are valid options.
- **Configuration:**
  - Configured ACL 1 using the named ACL configuration method.
  - Command: `IP ACCESS-LIST STANDARD 1` followed by the two separate entries.
- **Verification:**
  - After checking the running config, it displays as if configured using the traditional numbered ACL method (entries from global config mode).
- **Why Use Named ACL Config Mode?**
  - If it ends up being just like a regular numbered ACL, why configure it this way?
  - There are a few advantages to using named ACL config mode, even for numbered ACLs.

Advantage 1: Deleting Individual ACL Entries

```
R1(config-std-nacl)#do show access-lists
Standard IP access list 1
    10 deny   192.168.1.1
    20 deny   192.168.1.2
    30 deny   192.168.3.0, wildcard bits 0.0.0.255
    40 permit any
R1(config-std-nacl)#
R1(config-std-nacl)#no 30
R1(config-std-nacl)#
R1(config-std-nacl)#do show access-lists
Standard IP access list 1
    10 deny   192.168.1.1
    20 deny   192.168.1.2
    40 permit any
R1(config-std-nacl)#
```

- **Deleting in Named ACL Config Mode:**
  - You can easily delete individual ACL entries with the command `NO`, followed by the entry number.
  - Default sequence numbers start at 10 and increase by 10 (e.g., 10, 20, 30, 40).
  - In named ACL config mode, you can also specify the sequence number manually.
- **Example:**
  - Verification shows an ACL with four entries (sequence numbers 10, 20, 30, 40).
  - Command `NO 30` is used from named ACL config mode for ACL 1.
  - Verification shows that entry 30 has been deleted.
  - This is very convenient for editing ACLs.

Limitation of Deleting Entries in Global Config Mode

- **Attempting to Delete an Entry (Global Config Mode):**
  - Starting with the same ACL with four entries.
  - Tried to delete entry 30 using the `NO` command with the full entry syntax.

```
R1(config)# no access-list 1 deny 192.168.3.0 0.0.0.255
```

- **Verification:**
  - Checked with `show access-lists` and `running-config`; the command removed the **entire ACL**, not just entry 30.

- **Restriction:**
  - When editing numbered ACLs from global config mode, you **cannot delete individual entries**.
  - Using `NO` deletes the entire ACL, requiring you to rebuild it from zero.
- **Recommendation:**
  - Use named ACL config mode to edit ACLs to avoid deleting the whole list.
  - You can configure a numbered ACL in global config mode initially, then switch to named ACL config mode to perform edits.

Summary of Advantage 1

- **Key Takeaway:**
  - The first advantage of using named ACL config mode, even for numbered ACLs, is the ability to delete individual entries.
- **Command:**
  - Use `NO` followed by the entry number to delete a specific entry.

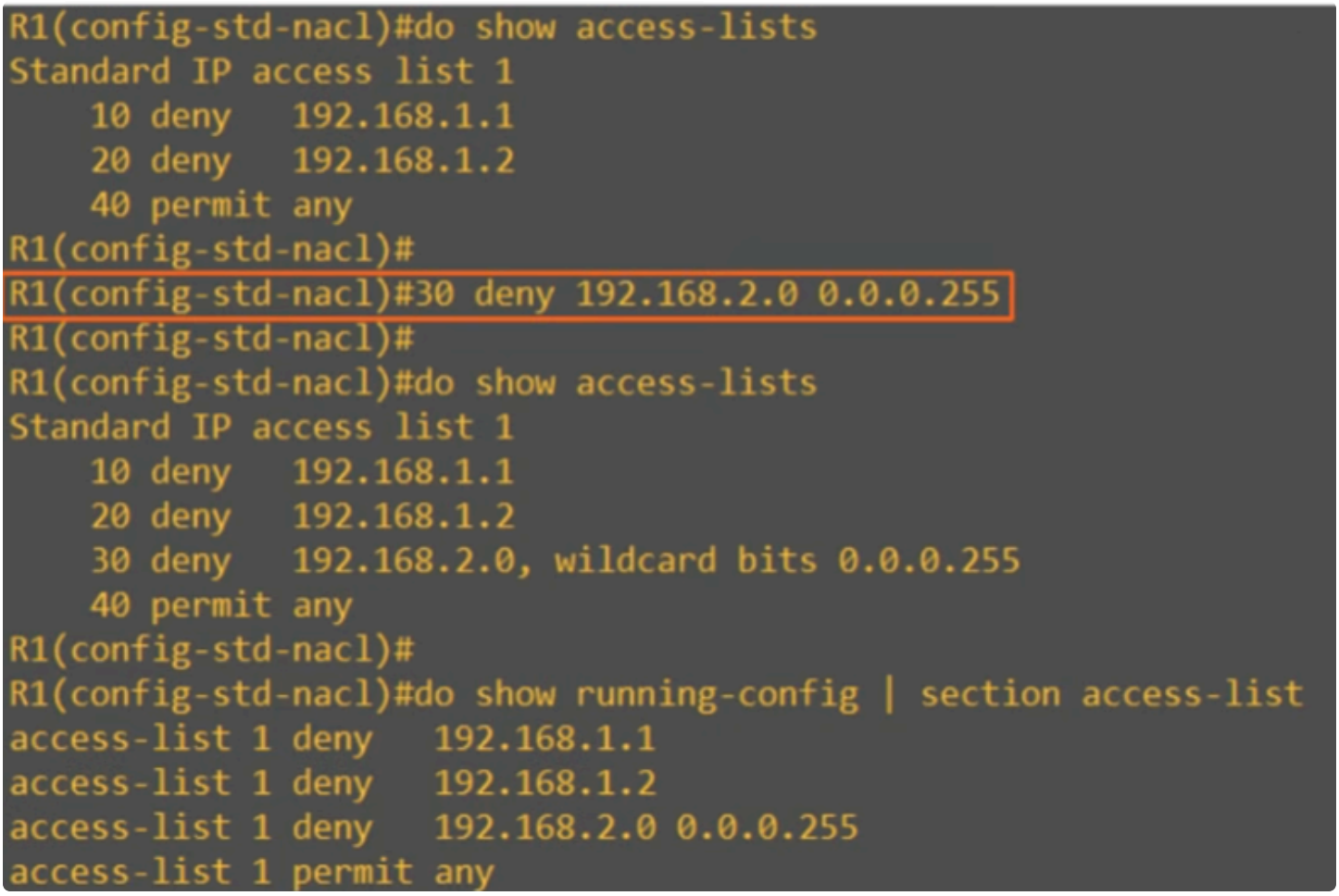
```
R1(config-std-nacl)# no 30
```

Advantage 2: Inserting Entries with Specific Sequence Numbers

- **Global Config Mode Limitation:**
  - When configuring an ACL from global config mode, you **cannot specify the sequence number**.
  - The entry is simply added to the **end** of the ACL.
  - The sequence number is automatically set to **10 higher** than the current highest sequence number.

- **Named ACL Config Mode Capability:**
  - From named ACL config mode, you **can set the sequence number**.
  - This allows you to **insert new entries in the middle** of an ACL.

Example of Inserting an Entry:



- Starting with the ACL after entry 30 was deleted.
- A new entry is configured with a specific sequence number of 30.

```
R1(config-std-nacl)# 30 deny 192.168.2.0 0.0.0.255
```

- **Verification Command**
  - **Command:**

```
R1# do show running-config | section access-list
```

  - **Result:**
    - The running config displays the entry as if it were configured in global config mode.
    - The new entry is inserted between the 2nd and 4th entries.
  - The new entry is inserted **between entries 20 and 40**.
  - The running config displays the entry as if it were configured in global config mode.

Method 3: Resequencing ACL Entries

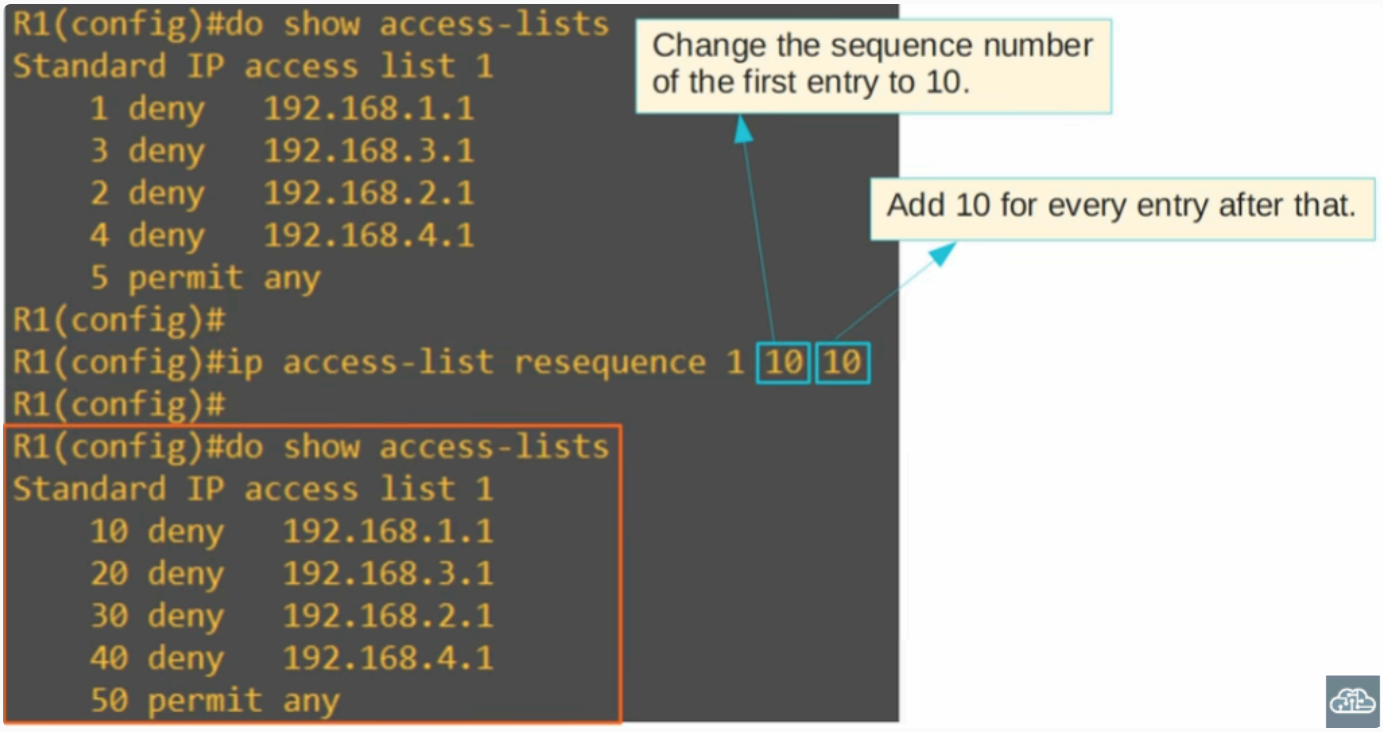
- **Purpose:**
  - Rearranges the sequence numbers of existing ACL entries to create gaps, making it possible to insert new entries in the middle.
- **Command:**
  - `IP ACCESS-LIST RESEQUENCE <ACL_ID> <STARTING_SEQ_NUM> <INCREMENT>`
- **Explanation of Command:**
  - **ACL_ID:** The number or name of the ACL.
  - **Starting Sequence Number:** The sequence number assigned to the first entry.
  - **Increment:** The amount to add to the sequence number for each subsequent entry.

- **Example Scenario:**

```
R1(config)#do show access-lists
Standard IP access list 1
    1 deny   192.168.1.1
    3 deny   192.168.3.1
    2 deny   192.168.2.1
    4 deny   192.168.4.1
    5 permit any
R1(config)#
R1(config)#ip access-list resequence 1 10 10
R1(config)#
R1(config)#do show access-lists
Standard IP access list 1
    10 deny   192.168.1.1
    20 deny   192.168.3.1
    30 deny   192.168.2.1
    40 deny   192.168.4.1
    50 permit any
```

  - An ACL is configured with sequence numbers 1, 2, 3, 4, and 5.
  - What is bad about these entries?
    - It is impossible to insert an entry in between the existing ones because there are no free sequence numbers.

- **Resequencing Process:**



  - Command: `IP ACCESS-LIST RESEQUENCE 1 10 10`

```
R1(config)# ip access-list resequence 1 10 10
```

  - This sets the first entry to sequence number 10.
  - It adds an increment of 10 to each subsequent entry.
- **Result:**
  - The order of the entries remains the same.
  - The sequence numbers are changed to 10, 20, 30, 40, and 50.
  - Now it is simple to insert new entries between the existing ones.
- **Note:**
  - This command is performed from global config mode.
  - It works for all ACLs (numbered, named, standard, and extended).

Introduction to Extended ACLs

- **Functionality:**
  - Extended ACLs function mostly the same as standard ACLs.
  - They can be numbered or named.
  - Processed from top to bottom.
- **Numbered Ranges:**
  - **Extended ACLs:** 100 to 199, and 2000 to 2699.
  - **Standard ACLs (Reminder):** 1 to 99, and 1300 to 1999.
- **Key Difference from Standard ACLs:**
  - Extended ACLs can match traffic based on more parameters, making them more precise and complex.
  - You can specify specific kinds of traffic, from specific hosts, to specific destinations.
- **Matching Parameters (Focus of this Lectures):**
  - Layer 4 protocol and port number.
  - Source IP address.
  - Destination IP address.

Configuring Extended ACLs

- **Extended Numbered ACL (Global Config Mode):**
  - Command syntax: `ACCESS-LIST <number> PERMIT|DENY <protocol> <source IP> <destination IP>`
  - The number must be in the ranges 100–199 or 2000–2699.
- **Extended Named ACL:**
  - Command syntax: `IP ACCESS-LIST EXTENDED <name|number>`
  - This enters extended named ACL config mode.

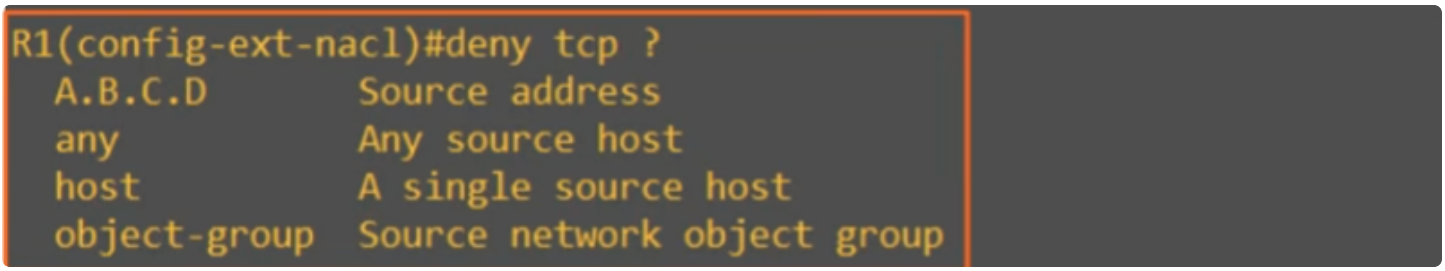
- **Configuration Focus:**
  - The lecture focuses on using the **named method** for configuration.
  - Extended numbered ACLs can also be configured in named ACL config mode.
- **Note on Complexity:**
  - Extended ACLs have many variations and options.
  - The Lecture covers basic options required for the CCNA.
  - Use the `?` in labs to explore all available options.

Matching Traffic Based on Protocol

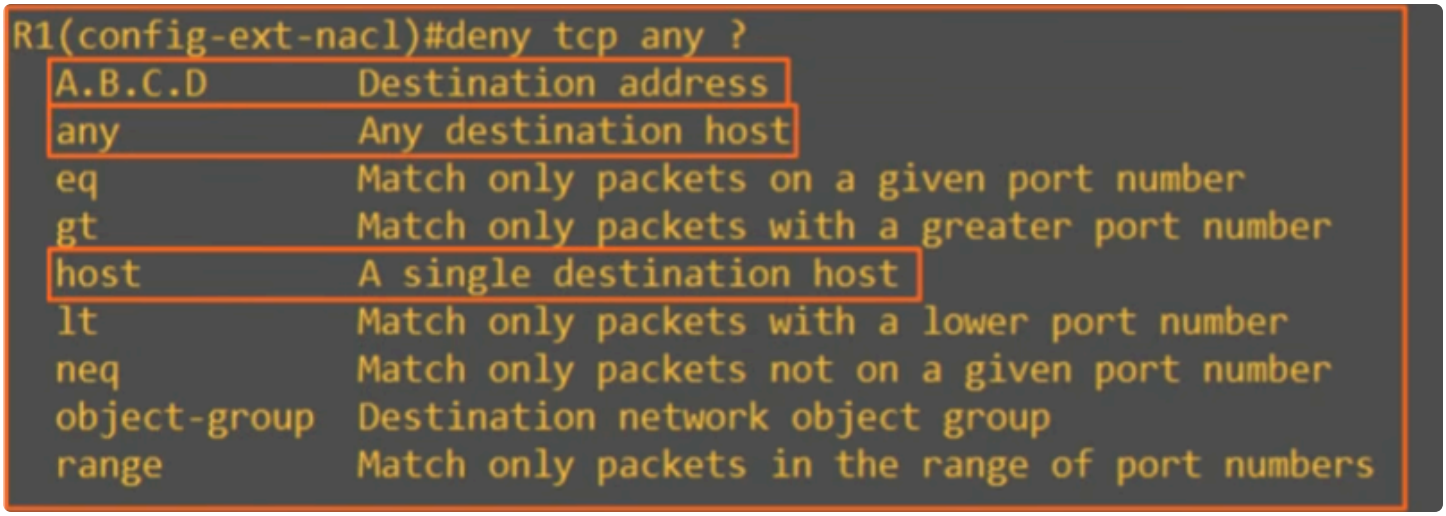
```
R1(config)#ip access-list extended EXAMPLE
R1(config-ext-nacl)#deny ?
  <0-255>        An IP protocol number
  ahp            Authentication Header Protocol
  eigrp          Cisco's EIGRP routing protocol
  esp            Encapsulation Security Payload
  gre            Cisco's GRE tunneling
  icmp           Internet Control Message Protocol
  igmp           Internet Gateway Message Protocol
  ip             Any Internet Protocol
  ipinip         IP in IP tunneling
  nos            KA9Q NOS compatible IP over IP tunneling
  object-group   Service object group
  ospf           OSPF routing protocol
  pcp            Payload Compression Protocol
  pim            Protocol Independent Multicast
  sctp           Stream Control Transmission Protocol
  tcp            Transmission Control Protocol
  udp            User Datagram Protocol
```

- **Protocol Identification Methods:**
  - **IP Protocol Number:** Identified by the number in the IPv4 header's protocol field.
    - This field identifies the protocol encapsulated inside the IP header, such as TCP or UDP.
  - **Protocol Name:** Preferred method as it is easier to remember.
- **Common Protocol Numbers:**
  - **1:** ICMP
  - **6:** TCP
  - **17:** UDP
  - **88:** EIGRP
  - **89:** OSPF
- **Usage Examples:**
  - You can use an extended ACL to block OSPF messages on an interface.
  - You can deny ICMP packets to deny pings.
- **Focus of Lesson:**
  - **TCP and UDP:** Specific transport layer protocols.
  - **IP:** Matches **all IP packets**.
    - Used when the specific protocol doesn't matter.
    - Example: Used for a `permit any` statement at the end of an ACL to allow all traffic.

Adding Source and Destination IP Addresses



- **Protocol Selection:**
  - Example: Selecting `TCP` as the protocol matches any IP packets with a TCP segment inside.
- **Specifying /32 Addresses:**
  - **Important Difference:** In extended ACLs, to specify a /32 source or destination, you **must** use the `HOST` option or specify the full wildcard mask (0.0.0.0).
  - You cannot just write the IP address alone (unlike in standard ACLs).



- **Source Address Options:**
  - **ANY:** Matches all source IP addresses.
- **Destination Address Options:**
  - **Specific IP:** The destination IP address followed by a wildcard mask.
  - **ANY:** Matches all destination IP addresses.
  - **HOST:** Specifies a single destination host.

- **Example Entry Analysis:**

```
R1(config-ext-nacl)#deny tcp any 10.0.0.0 0.0.0.255
R1(config-ext-nacl)#
```

  - Command logic: Deny TCP, source ANY, destination 10.0.0.0 with wildcard 0.0.0.255.
  - **Function:** Denies all packets encapsulating a TCP segment from **any** source IP address to the destination network **10.0.0.0/24**.

Practice Questions: Writing Individual Entries

Instructions: Practice writing individual entries for the following scenarios. These are not entries in the same ACL.

Scenario 1: Permit all traffic.

1. Allow all traffic

```
permit ip any any
```

- `permit` = allow
- `ip` = all IPv4 protocols, including TCP, UDP, and ICMP
- First `any` = any source
- Second `any` = any destination

Scenario 2: Prevent 10.0.0.0/16 from sending UDP traffic to 192.168.1.1/32.

2. Prevent `10.0.0.0/16` from sending UDP traffic to `192.168.1.1/32`

Convert `/16` to a wildcard mask:

```
/16 subnet mask:   255.255.0.0
Wildcard mask:       0.0.255.255
```

`192.168.1.1/32` is one specific host, so use `host`.

```
deny udp 10.0.0.0 0.0.255.255 host 192.168.1.1
```

Breakdown:

```
deny udp
|    └── Block UDP
└─────── Deny traffic

10.0.0.0 0.0.255.255
└── Source network: 10.0.0.0/16

host 192.168.1.1
└── One destination device
```

Scenario 3: Prevent 172.16.1.1/32 from pinging hosts in 192.168.0.0/24.

3. Prevent `172.16.1.1/32` from pinging hosts in `192.168.0.0/24`

Ping uses **ICMP echo**.

Convert `/24` to a wildcard mask:

```
/24 subnet mask:   255.255.255.0
Wildcard mask:       0.0.0.255
```

The source is one host:

```
deny icmp host 172.16.1.1 192.168.0.0 0.0.0.255 echo
```

Breakdown:

```
deny icmp
└── Block ICMP traffic

host 172.16.1.1
└── Source device

192.168.0.0 0.0.0.255
└── Destination network: 192.168.0.0/24

echo
└── ICMP echo request, which is a ping request
```

1. permit ip any any

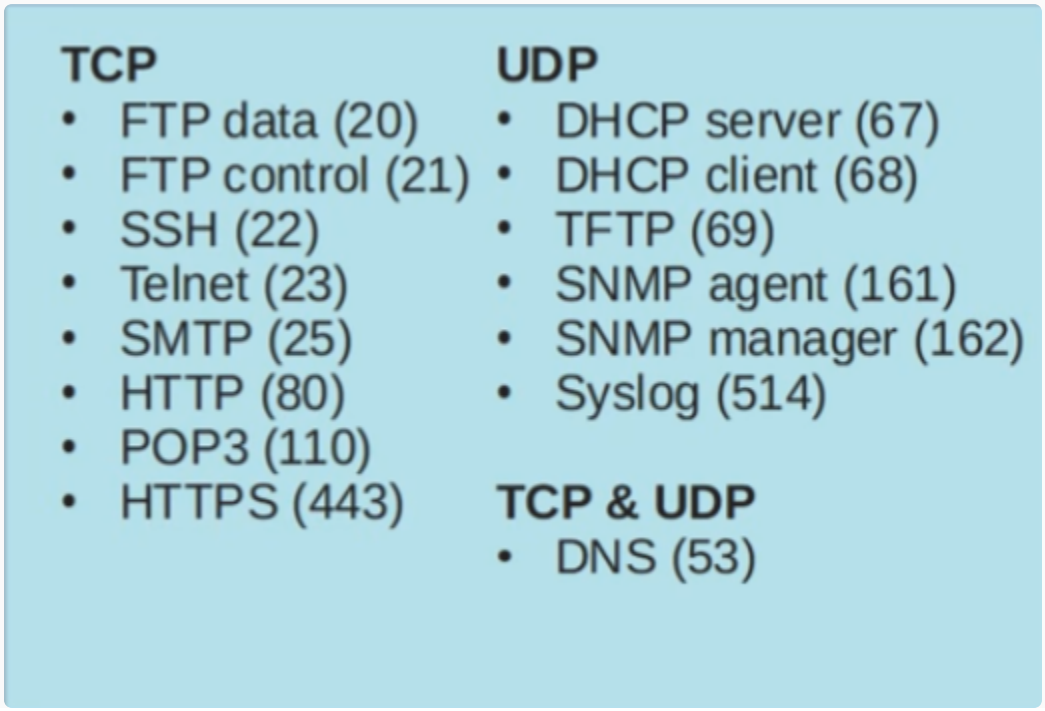
2. deny udp 10.0.0.0 0.0.255.255 host 192.168.1.1

3. deny icmp host 172.16.1.1 192.168.0.0 0.0.0.255 echo

Matching TCP and UDP Port Numbers

- **Optionality:** When specifying TCP or UDP as the protocol, you can **optionally** specify source and/or destination port numbers.
  - If ports are not specified, **all port numbers** are matched.
- **Command Format:**
  - Syntax: `<PERMIT|DENY> <TCP|UDP> <src-ip> <src-port> <dest-ip> <dest-port>`

- **Source Port Specification:** Specified after the source IP address and wildcard mask.
  - **Syntax Options:**
    - `EQ <port>`: Equal to a specific port (e.g., `EQ 80` matches TCP source port 80).
    - `GT <port>`: Greater than a specific port (e.g., `GT 80` matches all ports greater than 80, so 81 and up).
    - `LT <port>`: Less than a specific port (e.g., `LT 80` matches all port numbers less than 80, so 79 and below).
    - `NEQ <port>`: Not equal to a specific port (e.g., `NEQ 80` matches all ports except 80).
    - `RANGE <start> <end>`: Range of ports (e.g., `RANGE 80 100` matches all port numbers from 80 to 100).
- **Destination Port Specification:** Specified after the destination IP address.
  - Uses the same syntax options as source ports (`EQ`, `GT`, `LT`, `NEQ`, `RANGE`).
- **Best Practice:** The most common choice is `EQ` to match traffic for a specific port number.
- Hopefully you remember these port numbers from day 30 of the course, here is an image below:



- **SSH (22)** → Secure remote management (preferred over Telnet).

- **Telnet (23)** → Insecure remote management.
- **HTTP (80)** → Unencrypted web traffic.
- **HTTPS (443)** → Encrypted web traffic.
- **DNS (53)** → Converts names (e.g., `google.com`) to IP addresses.
- **DHCP (67/68)** → Automatically assigns IP configuration.
- **FTP (20/21)** → File transfers.
- **SMTP (25)** → Sends email.
- **POP3 (110)** → Downloads email.
- **SNMP (161/162)** → Network monitoring.
- **Syslog (514)** → Collects device logs.
- **TFTP (69)** → Simple file transfers for network devices.

Example: Matching Destination Port 80

- **Command Example:**

```
R1(config-ext-nacl)#deny tcp any host 1.1.1.1 eq ?
  <0-65535>     Port number
  bgp           Border Gateway Protocol (179)
  chargen       Character generator (19)
  cmd           Remote commands (rcmd, 514)
  daytime       Daytime (13)
  discard       Discard (9)
  domain        Domain Name Service (53)
  drip          Dynamic Routing Information Protocol (3949)
  echo          Echo (7)
  exec          Exec (rsh, 512)
  finger        Finger (79)
  ftp           File Transfer Protocol (21)
  ftp-data      FTP data connections (20)
  gopher        Gopher (70)
  hostname      NIC hostname server (101)
  ident         Ident Protocol (113)
  irc           Internet Relay Chat (194)
  klogin        Kerberos login (543)
  kshell        Kerberos shell (544)
  login         Login (rlogin, 513)
  lpd           Printer service (515)
  nntp          Network News Transport Protocol (119)
  onep-plain    ONEP Cleartext (15001)
  onep-tls      ONEP TLS (15002)
  pim-auto-rp   PIM Auto-RP (496)
  pop2          Post Office Protocol v2 (109)
  pop3          Post Office Protocol v3 (110)
  smtp          Simple Mail Transport Protocol (25)
  sunrpc        Sun Remote Procedure Call (111)
  tacacs        TAC Access Control System (49)
  talk          Talk (517)
  telnet        Telnet (23)
  time          Time (37)
  uucp          Unix-to-Unix Copy Program (540)
  whois         Nicname (43)
  www           World Wide Web (HTTP, 80)
```

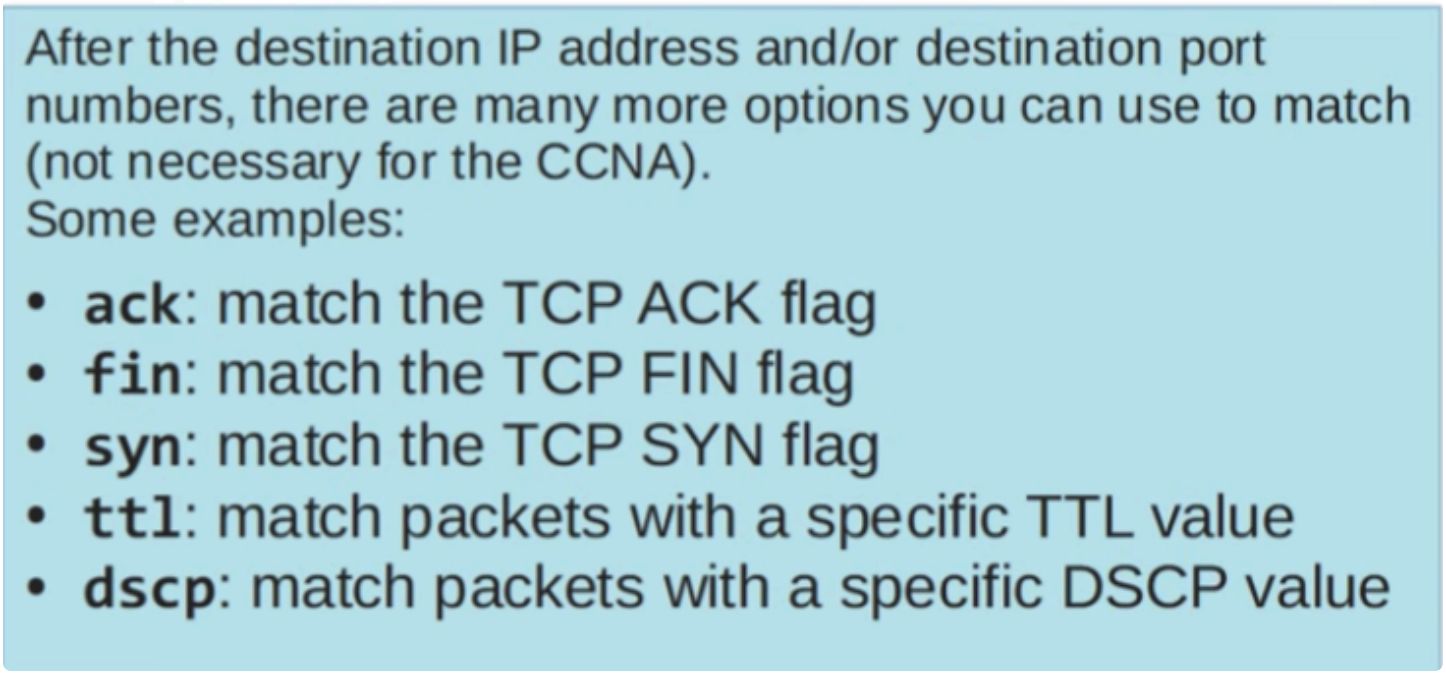
```
R1(config-std-nacl)#deny tcp any host 1.1.1.1 eq 80
```

→ Deny all packets destined for IP address 1.1.1.1/32, TCP port 80.

- Command: `DENY TCP ANY HOST 1.1.1.1 EQ 80`

    - Logic: Deny TCP, source ANY, destination HOST 1.1.1.1, destination port equal to 80.
- **Keyword Options:**
    - You can enter the specific port number or use a keyword (e.g., `WWW` for HTTP, port 80).
    - Many common port numbers do not have a keyword, so it is important to learn the actual numbers.
- **Effect of Entry:**
    - Denies all packets destined for IP address **1.1.1.1/32** on **TCP port 80**.

Additional Matching Options (Not Required for CCNA)



- **Options:**
    - `ACK`: Match the TCP ACK flag.
    - `FIN`: Match the TCP FIN flag.
    - `SYN`: Match the TCP SYN flag.
    - `TTL`: Match packets with a specific Time to Live value.
    - `DSCP`: Match packets with a specific Differentiated Services Code Point value.

- **Matching Logic:**
  - If multiple parameters are specified (protocol, source IP, source port, destination IP, destination port), a packet must match **ALL** specified values to match the ACL entry.
  - If a packet matches all parameters except one, it will not match that entry.

Additional Practice Questions

Instructions: Write an extended ACL entry for the following scenarios.

Scenario 1: Allow traffic from 10.0.0.0/16 to access the server at 2.2.2.2/32 using HTTPS.

1. Allow `10.0.0.0/16` to access `2.2.2.2` using HTTPS

HTTPS uses:

```
TCP port 443
```

`/16` wildcard mask:

```
Subnet mask:   255.255.0.0
Wildcard mask:   0.0.255.255
```

Command:

```
permit tcp 10.0.0.0 0.0.255.255 host 2.2.2.2 eq 443
```

You can also use the protocol name:

```
permit tcp 10.0.0.0 0.0.255.255 host 2.2.2.2 eq https
```

Scenario 2: Prevent all hosts from using source UDP port numbers from 20000 to 30000 from accessing the server at 3.3.3.3/32.

2. Block hosts using UDP source ports `20000–30000` from accessing `3.3.3.3`

Because the question says **source ports**, place the port range immediately after the source address:

```
deny udp any range 20000 30000 host 3.3.3.3
```

Breakdown:

```
deny udp any range 20000 30000 host 3.3.3.3
```

No destination port is entered, so this blocks access to **all UDP destination ports** on `3.3.3.3`.

Scenario 3: Allow hosts in 172.16.1.0/24 using a TCP source port greater than 9999 to access all TCP ports on server 4.4.4.4/32 except port 23.

3. Allow `172.16.1.0/24` with TCP source ports greater than `9999` to access every TCP port on `4.4.4.4` except port `23`

Port `23` is Telnet.

`/24` wildcard mask:

```
 Subnet mask:   255.255.255.0
Wildcard mask:   0.0.0.255
```

Command:

```
permit tcp 172.16.1.0 0.0.0.255 gt 9999 host 4.4.4.4 neq 23
```

Port placement:

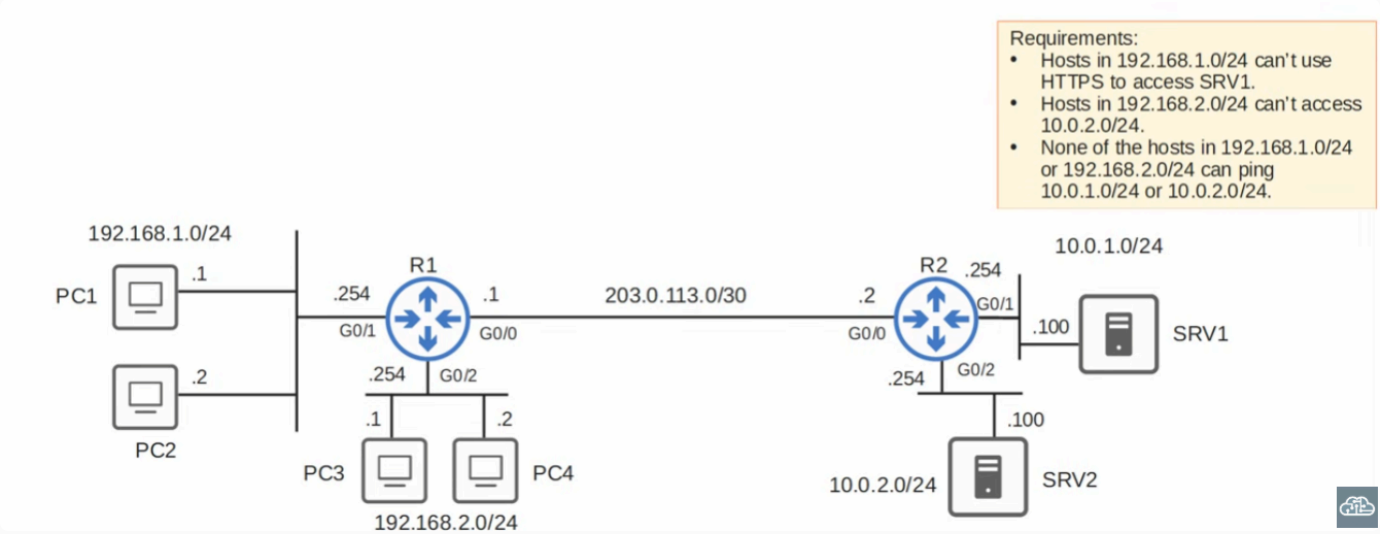
```
172.16.1.0 0.0.0.255 gt 9999
                        └── Source port greater than 9999

host 4.4.4.4 neq 23
             └── Destination port cannot equal 23
```

1. permit tcp 10.0.0.0 0.0.255.255 host 2.2.2.2 eq 443
2. deny udp any range 20000 30000 host 3.3.3.3
3. permit tcp 172.16.1.0 0.0.0.255 gt 9999 host 4.4.4.4 neq 23

Applying Extended ACLs: Network Example

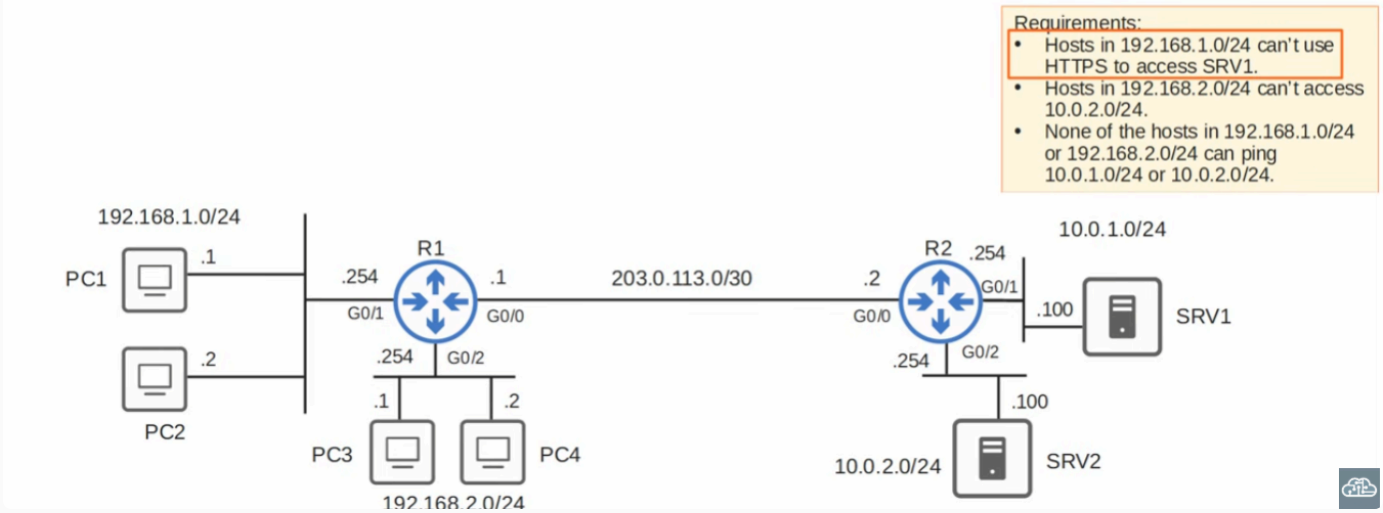
- **Requirements:**



1. Hosts in 192.168.1.0/24 can't use HTTPS to access SRV1.
2. Hosts in 192.168.2.0/24 can't access 10.0.2.0/24.
3. Hosts in 192.168.1.0/24 and 2.0/24 cannot ping 10.0.1.0/24 or 2.0/24.

- **Requirement 1 Configuration (HTTPS Access):**

```
R1(config)#ip access-list extended HTTP_SRV1
R1(config-ext-nacl)#deny tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 443
R1(config-ext-nacl)#permit ip any any
```



  - Deny TCP traffic from 192.168.1.0/24 to destination 10.0.1.100 (SRV1) on port 443.
  - Permit all other traffic (`permit ip any any`).
- **Placement Rule:**
  - **Standard ACLs:** Apply as close to the **destination** as possible (to avoid blocking unintended traffic).
  - **Extended ACLs:** Apply as close to the **source** as possible.
- **Reasoning for Extended ACL Placement:**
  - Extended ACLs are specific; applying them near the source prevents unwanted packets from crossing the network.
  - This saves router resources and limits the "distance" denied packets travel.

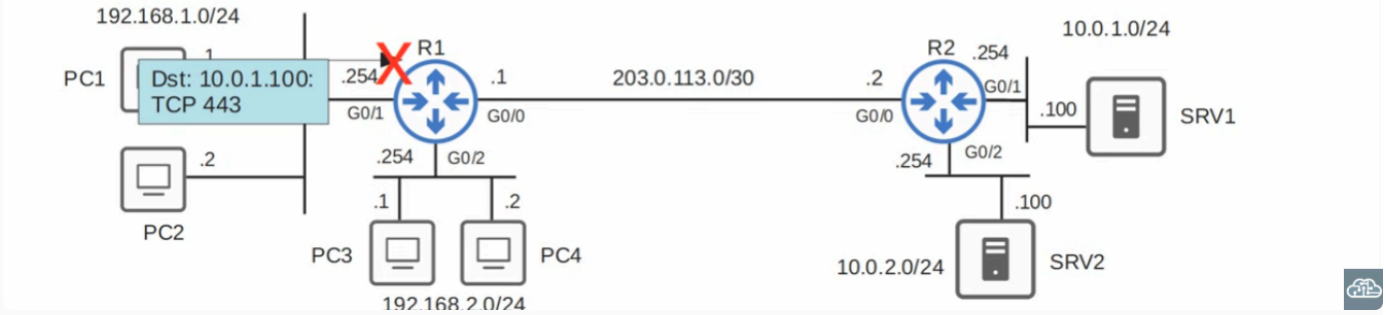
- **Applying Requirement 1:**

```
R1(config)#ip access-list extended HTTP_SRV1
R1(config-ext-nacl)#deny tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 443
R1(config-ext-nacl)#permit ip any any
R1(config-ext-nacl)#interface g0/1
R1(config-if)#ip access-group HTTP_SRV1 in
```

Extended ACLs should be applied as close to the source as possible, to limit how far the packets travel in the network before being denied.

(Standard ACLs are less specific, so if they are applied close to the source there is a risk of blocking more traffic than intended)

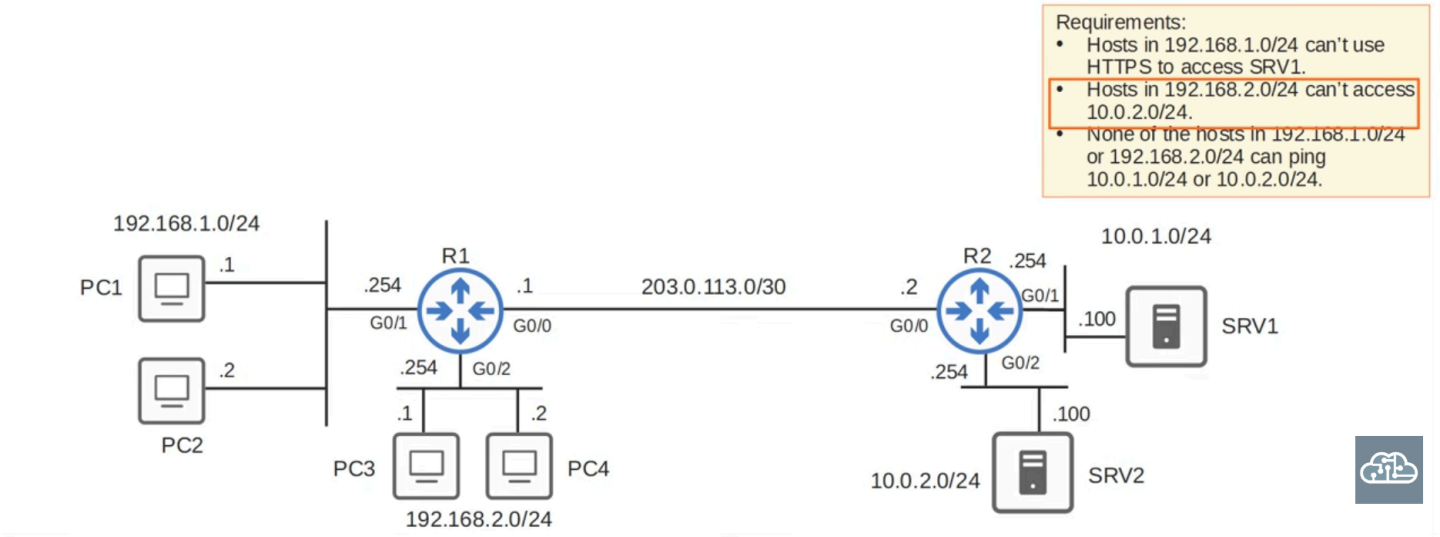
Requirements:
- Hosts in 192.168.1.0/24 can't use HTTPS to access SRV1.
- Hosts in 192.168.2.0/24 can't access 10.0.2.0/24.
- None of the hosts in 192.168.1.0/24 or 192.168.2.0/24 can ping 10.0.1.0/24 or 10.0.2.0/24.



- **Interface:** R1's G0/1 interface (the source of 192.168.1.0/24).
- **Direction:** Inbound.

Requirement 2 Configuration (Block Access to 10.0.2.0/24)

```
R1(config)#ip access-list extended BLOCK_10.0.2.0/24
R1(config-ext-nacl)#deny ip 192.168.2.0 0.0.0.255 10.0.2.0 0.0.0.255
R1(config-ext-nacl)#permit ip any any
R1(config-ext-nacl)#interface g0/2
R1(config-if)#ip access-group BLOCK_10.0.2.0/24 in
```

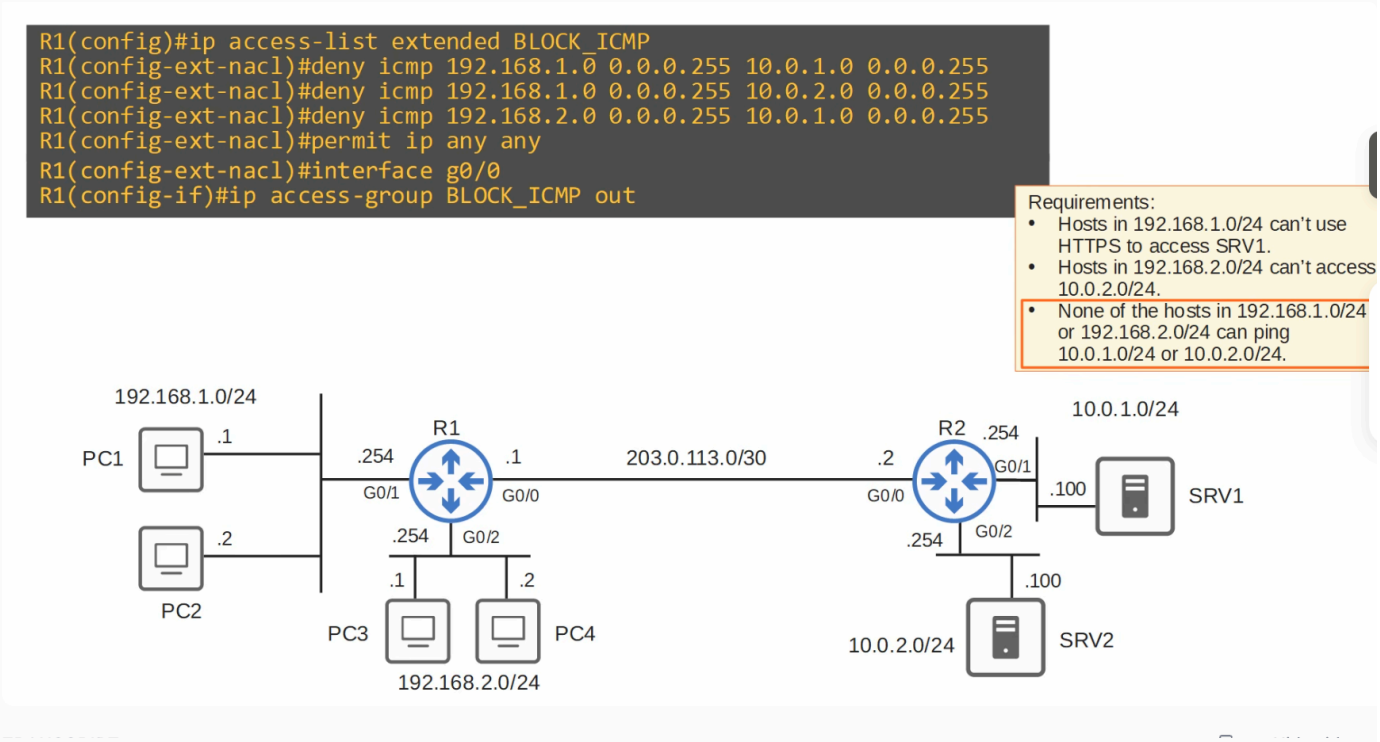


- **Context:**
  - Requirement: Hosts in 192.168.2.0/24 cannot access 10.0.2.0/24.
  - Action: Create a new ACL on R1.
- **ACL Configuration:**
  - Specify **IP** as the protocol to match all packets (anything with an IP header).
  - Deny traffic from source **192.168.2.0/24** to destination **10.0.2.0/24**.
  - Add a `PERMIT IP ANY ANY` statement to allow all other traffic.
- **Interface Application:**
  - **Placement Rule:** Extended ACLs should be applied as close to the source as possible.
  - **Source Identification:** The source is 192.168.2.0/24.
  - **Result:** Apply the ACL **inbound** on R1's **G0/2** interface.

Requirement 3 Configuration (Blocking Ping Traffic)

- **Context:**
  - Requirement: None of the hosts in 192.168.1.0/24 or 192.168.2.0/24 can ping 10.0.1.0/24 or 10.0.2.0/24.
  - Protocol: Ping uses **ICMP**.
- **ACL Configuration:**

```
R1(config)#ip access-list extended BLOCK_ICMP
R1(config-ext-nacl)#deny icmp 192.168.1.0 0.0.0.255 10.0.1.0 0.0.0.255
R1(config-ext-nacl)#deny icmp 192.168.1.0 0.0.0.255 10.0.2.0 0.0.0.255
R1(config-ext-nacl)#deny icmp 192.168.2.0 0.0.0.255 10.0.1.0 0.0.0.255
R1(config-ext-nacl)#permit ip any any
R1(config-ext-nacl)#interface g0/0
R1(config-if)#ip access-group BLOCK_ICMP out
```



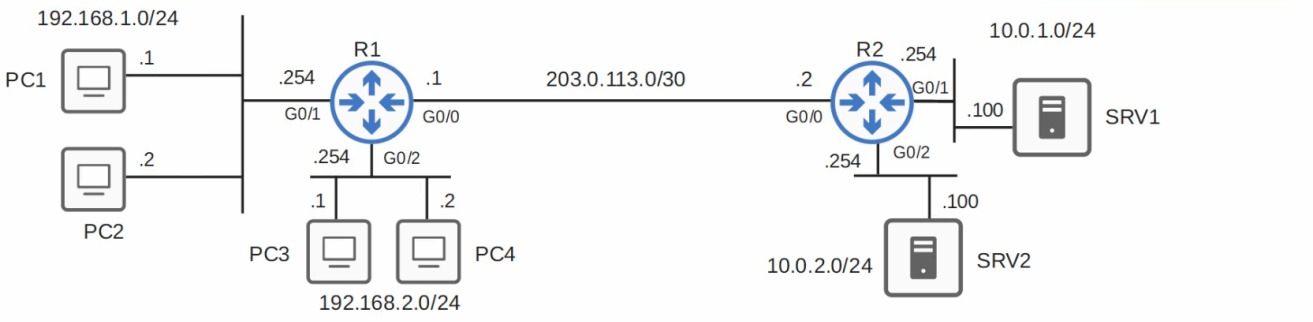
  - **Entries created:**
    - Two deny entries for source IP **192.168.1.0/24** (one for destination 10.0.1.0/24, one for 10.0.2.0/24).
    - One deny entry for source IP **192.168.2.0/24** (for destination 10.0.1.0/24).
  - **Logic/Explanation:**
    - Only one entry is needed for source 192.168.2.0/24 because the previous ACL already blocks all traffic from 192.168.2.0/24 to 10.0.2.0/24.
  - **Permit Statement:**
    - Added `PERMIT IP ANY ANY` at the end to allow all other traffic.

- **Interface Application:**
  - **Placement:** Outbound on R1's **G0/0** interface.
  - **Reasoning:** Applying it here ensures the ACL filters traffic from both the 192.168.1.0/24 and 192.168.2.0/24 networks before they reach the server networks.

Summary of Configured ACLs

- **Overview:**
  - The three ACLs previously discussed have been configured on R1 to fulfill the network requirements.
- **Configuration Flexibility:**

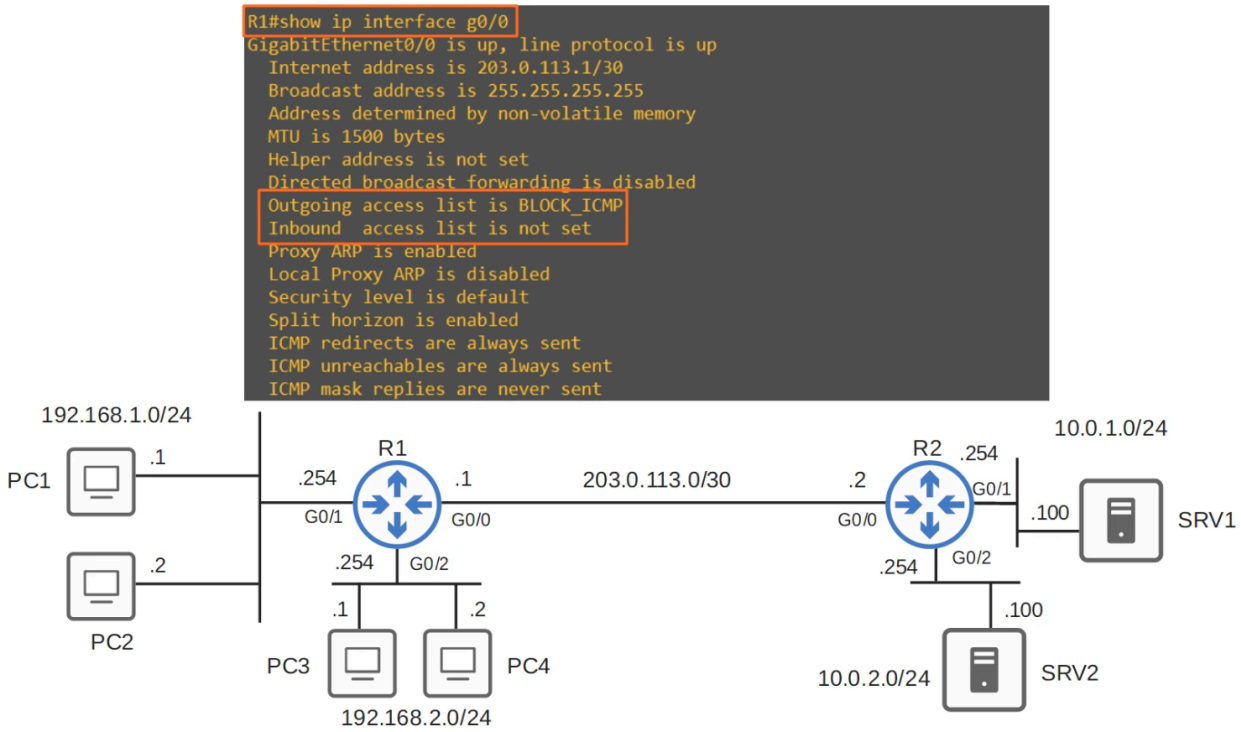
```
R1#show access-lists
Extended IP access list BLOCK_10.0.2.0/24
    10 deny ip 192.168.2.0 0.0.0.255 10.0.2.0 0.0.0.255
    20 permit ip any any
Extended IP access list BLOCK_ICMP
    10 deny icmp 192.168.1.0 0.0.0.255 10.0.1.0 0.0.0.255
    20 deny icmp 192.168.1.0 0.0.0.255 10.0.2.0 0.0.0.255
    30 deny icmp 192.168.2.0 0.0.0.255 10.0.1.0 0.0.0.255
    40 permit ip any any
Extended IP access list HTTP_SRV1
    10 deny tcp 192.168.1.0 0.0.0.255 host 10.0.1.100 eq 443
    20 permit ip any any
```



- ACL configuration can be quite flexible.
- This specific solution is not the only one that works.

- **Efficiency Challenge:**
  - The presented solution is **not the most efficient**.
  - **Challenge:** Try to create a more efficient solution that uses fewer ACLs or fewer entries while still fulfilling the requirements.

Verifying Applied ACLs



- **Command:**
  - `SHOW IP INTERFACE` followed by the interface name.
  - Example: `R1# show ip interface g0/1`
- **Distinction from Brief Command:**
  - The `SHOW IP INTERFACE BRIEF` command is used often, but the regular version (without BRIEF) provides more detailed information, including ACL assignments.

- **Output Information:**
  - The output will explicitly state which ACL is applied **outbound** and which is applied **inbound**.
  - If no ACL is configured, the output will say **'not set'**.
- **Alternative Verification:**
  - You can also verify applied ACLs by checking the **running config**.
- **Importance:**
  - It is important to know this command for both the exam and real-world purposes.

Lecture Review

- **Alternative Configuration Method:**
  - Showed another way to configure numbered ACLs.
  - You can configure numbered ACLs in named ACL config mode.
  - **Big Advantage:** Editing ACLs.
  - Named ACL config mode lets you delete individual ACL entries.
  - You can specify the sequence number of new entries to insert them in the middle of an ACL.
- **Extended ACLs:**
  - Covered extended numbered and named ACLs.
  - Extended ACLs are much more powerful than standard ACLs.
  - They can match traffic based on:
    - Protocol.
    - Source and destination IP addresses.
    - Source and destination port numbers.
  - This makes them more complex to configure, but practice makes it comfortable.