

35. EXTENDED ACCESS CONTROL LISTS (EACL)

ANOTHER WAY TO CONFIGURE NUMBERED ACLs

- In DAY 34, you learned that numbered ACLs are configured in Global Config mode:

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

Figure 1: image

- You learned that named ACLs are configured with subcommands in a separate config mode:

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

Figure 2: image

- However, in modern IOS you can also configure numbered ACLs in the exact same way as named ACLs:

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

Figure 3: image

ADVANTAGES OF NAMED ACL CONFIG MODE

- You can easily DELETE individual entries in the ACL with NO *entry-number*
- You can easily DELETE individual entries in the ACL with NO *sequence-number*

This doesn't work with NUMBERED access lists

- You can insert NEW entries in-between other entries by specifying the SEQUENCE NUMBER

RESEQUENCING ACLs

- There is a *resequencing* function that helps edit ACLs

Configuring numbered ACLs with subcommands

```
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)#

```

Figure 4: image

```
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)#

```

Figure 5: image

```
R1(config)#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)#do show running-config | section access-list
access-list 1 deny  192.168.1.1
access-list 1 deny  192.168.1.2
access-list 1 deny  192.168.3.0 0.0.0.255
access-list 1 permit any
R1(config)#no access-list 1 deny 192.168.3.0 0.0.0.255
R1(config)#do show access-lists
R1(config)#do show running-config | section access-list
R1(config)#

```

Figure 6: image

```
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)#
R1(config-std-nacl)#30 deny 192.168.2.0 0.0.0.255
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
 30 deny  192.168.2.0, wildcard bits 0.0.0.255
 40 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 deny  192.168.1.2
access-list 1 deny  192.168.2.0 0.0.0.255
access-list 1 permit any

```

Figure 7: image

- The command is R1(config)#ip access-list resequence *acl-id starting-seq-num increment*

```

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

```

Figure 8: image

EXTENDED NUMBERS AND NAMED ACLS

- EXTENDED ACLs function mostly the same as STANDARD ACLs
- They can be NUMBERED or NAMED, just like STANDARD ACLs
 - NUMBERED ACLs use the following ranges: **100 - 199, 2000 - 2699**
- Processed from TOP to BOTTOM, just like STANDARD ACLs
- However, they can match traffic based on MORE PARAMETERS, so they are more PRECISE (and more complex) than STANDARD ACLs
- We will focus on matching based on these main parameters:
 - LAYER 4 protocol / port**
 - Source Address**
 - Destination Address**

EXTENDED NUMBERED ACL

```
R1(config)# access-list *number* [permit | deny] *protocol
src-ip dest-ip*
```

EXTENDED NAMED ACL

```
R1(config)# ip access-list extended {name | number}
```

```
R1(config-ext-nacl)# {seq-num} {permit | deny} *protocol src-ip  
dest-ip*
```

MATCHING THE 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

1: ICMP
6: TCP
17: UDP
88: EIGRP
89: OSPF

Figure 9: image

IP Protocol Number is the number used in the IPv4 Header Protocol field

Examples: (1) ICMP, (6) TCP, (17) UDP, (88) EIGRP, (89) OSPF

MATCHING THE SOURCE / DESTINATION IP ADDRESS

This command:

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

Deny ALL PACKETS that encapsulate a TCP segment from ANY source to
DESTINATION 10.0.0.0/24

PRACTICE QUESTIONS:

- 1) ALLOW ALL TRAFFIC

```
R1(config-ext-nacl)# permit ip any any (ip is used for "all protocols")
```

- 2) PREVENT 10.0.0.0/16 from SENDING UDP traffic to 192.168.1.1/32

```
R1(config-ext-nacl)# deny udp 10.0.0.0 0.0.255.255 host 192.168.1.1
```

- 3) PREVENT 172.16.1.1/32 from pinging hosts in 192.168.0.0/24

```

R1(config-ext-nacl)#deny tcp ?
  A.B.C.D      Source address
  any          Any source host
  host         A single source host
  object-group Source network object group

R1(config-ext-nacl)#deny tcp any ?
  A.B.C.D      Destination address
  any          Any destination host
  eq           Match only packets on a given port number
  gt           Match only packets with a greater port number
  host         A single destination host
  lt           Match only packets with a lower port number
  neq          Match only packets not on a given port number
  object-group Destination network object group
  range        Match only packets in the range of port numbers

R1(config-ext-nacl)#deny tcp any 10.0.0.0 ?
  A.B.C.D Destination wildcard bits

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

```

In extended ACLs, to specify a /32 source or destination you have to use the **host** option or specify the wildcard mask.
You can't just write the address without either of those.

Figure 10: image

R1(config-ext-nacl)# deny icmp host 172.16.1.1 192.168.0.0 0.0.0.255
MATCHING THE TCP / UDP PORT NUMBERS

- When matching TCP / UDP, you can optionally specify the SOURCE and/or DESTINATION PORT NUMBERS to match

```

R1(config-ext-nacl)#deny tcp src-ip   eq    src-port-num   dest-ip   eq    dst-port-num
                  gt
                  lt
                  neq
                  range

```

- **eq 80** = equal to port 80
- **gt 80** = greater than 80 (81 and greater)
- **lt 80** = less than 80 (79 and less)
- **neq 80** = NOT 80
- **range 80 100** = from port 80 to port 100

TCP	UDP
• FTP data (20)	• DHCP server (67)
• FTP control (21)	• DHCP client (68)
• SSH (22)	• TFTP (69)
• Telnet (23)	• SNMP agent (161)
• SMTP (25)	• SNMP manager (162)
• HTTP (80)	• Syslog (514)
• POP3 (110)	
• HTTPS (443)	TCP & UDP
	• DNS (53)

Figure 11: image

eq = equal than

gt = greater than

lt = less than

neq = not equal to

range = range of ports

You can use either the PORT NUMBER or the specific TYPE (that has a KNOWN PORT NUMBER)

PRACTICE QUESTIONS 2:

- 1) ALLOW TRAFFIC from 10.0.0.0/16 to access the server at 2.2.2.2/32 using HTTPS

```
R1(config-ext-nacl)# permit tcp 10.0.0.0 0.0.255.255 host 2.2.2.2  
eq 443
```

- 2) PREVENT ALL HOSTS using SOURCE UDP Port Numbers from 20000 to 30000 from accessing the server at 3.3.3.3/32

```
R1(config-ext-nacl)# deny udp any range 20000 30000 host 3.3.3.3
```

- 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

```
R1(config-ext-nacl)# permit tcp 172.16.1.0 0.0.0.255 gt 9999 host  
4.4.4.4 neq 23
```

EXAMPLE NETWORK

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

EXTENDED ACL #1 (Applied at R1 G0/1 INBOUND interface)

```
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)# int g0/1
```

```
R1(config-if)# ip access-group HTTP_SRV1 in
```

EXTENDED ACL #2 (APPLIED at R1 G0/2 INBOUND interface)

```
R1(config)# ip access-list extended BLOCK_10.0.2.0
```

```
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)# int g0/2
```

```
R1(config-if)# ip access-group BLOCK_10.0.2.0 in
```

```
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)
```

Figure 12: image

```
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.

After the destination IP address and/or destination port numbers, there are many more options you can use to match (not necessary for the CCNA).

Some examples:

- **ack**: match the TCP ACK flag
- **fin**: match the TCP FIN flag
- **syn**: match the TCP SYN flag
- **ttl**: match packets with a specific TTL value
- **dscp**: match packets with a specific DSCP value

If you specify the protocol, source IP, source port, destination IP, destination port, etc, a packet must match ALL of those values to match the ACL entry. Even if it matches all except one of the parameters, the packet won't match that entry of the ACL.

Figure 13: image

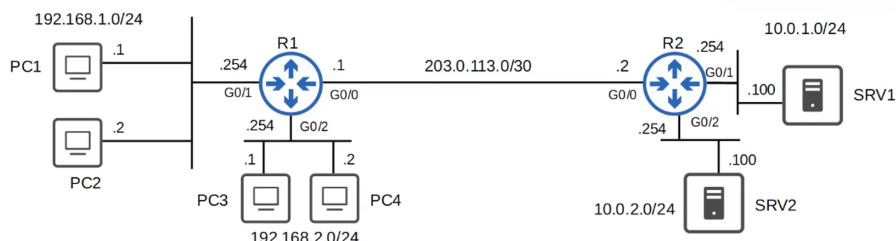


Figure 14: image

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)

Figure 15: image

EXTENDED ACL #3 (APPLIED at R1 g0/0 OUTBOUND interface)

```
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)# int g0/0
R1(config-if)# ip access-group BLOCK_ICMP out
```

What the EXTENDED ACLs look like

```
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
```

Figure 16: image

HOW TO SEE WHICH EXTENDED ACL's ARE APPLIED TO AN INTERFACE

```
R1#show ip interface g0/0
GigabitEthernet0/0 is up, line protocol is up
  Internet address is 203.0.113.1/30
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is BLOCK_ICMP
  Inbound access list is not set
  Proxy ARP is enabled
  Local Proxy ARP is disabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
```

Figure 17: image