Post 20

Thursday, January 24, 2019 11:16 PM

ACCESS CONTROL LISTS P2

Editing #'d ACL: 2 ways to edit

Txt Editor	Notepad: Create/edit copy/paste • Existing ACL? show run to display then copy/paste • no access-list # [*global config*] show run to verify
Sequence #	show access-lists # | Sequence # at beginning of statement • Auto assigned when access-list entered ip access-lists standard to config • ACL # used as name • Del misconfig statement w/no sequence # • Add new sequence # ○ Example: 11 deny host 192.168.10.10 Statements can't be overwritten using same sequence # as existing: • Current must del/then new one added show access-lists Verify • Can also be used to edit standard named ACL's

Editing Named ACL

ip access-list standard Config named ACLs: Insert/remove statements
no sequence-number Delete individual statements

- Adding a statement to deny another requires inserting a numbered line

Verify ACLs

show ip int Verify ACL on int	show access-lists *number/name* View individual access list/statistics

clear access-lists Clear counters on ACLs

VTY Access

access-class [*line config*] Restricts incoming/outgoing connections between VTY/addresses in access list

- Outbound Telnet extended doesn't prevent r-initiated Telnet sessions
- Filtering Telnet/SSH considered extended IP ACL function (higher lvl protocol): Standard can be used though

R1(config-line)# access-class access-list-number { in [vrf-also] | out }
in Restricts connections bet addresses in access-list/device
out Restricts outgoing connections between particular device/addresses in access-list
Note: Both named/numbered access lists | Identical restrictions: Set on all VTYs

Extended ACLs

Extended	100-199 | 2000-2699: 799 possible numbered: Can also be named • Used more often: Provide greater control They check: Source/destination addresses of packets | Protocols | Port numbers/services • Greater range of criteria ○ Example: Allow email to specific destination WHILE denying file transfers/browsing **Logical operators:** • eq Equal | neq Not equal | gt Greater than | lt Less than

R1(config)# access-list 101 permit tcp any any eq ?

0-65535	Port #	**bgp**	179	**chargen**	Char generator 19
cmd	Remote cmds; rcmd 514	**daytime**	13	**discard**	9
domain	DNS 53	**drip**	Dynamic RIP 3949	**echo**	7
exec	rsh, 512	**finger**	79	**ftp**	21

ftp-data	Connections 20	**gopher**	70	**hostname**	NIC hostname server 101
ident	Ident protocol 113	**irc**	IRC 194	**klogin**	Kerberos 543
kshell	Kerberos 544	**login**	rlogin 513	**lpd**	Printer service 515
nntp	Network News Transport 119	**pim-auto-rp**	PIM 496	**pop2**	109
pop3	110	**smtp**	25	**sunrpc**	111
tacacs	TAC Access Control Sys 49	**talk**	517	**telnet**	23
uucp	Unix-to-Unix Copy Program 540	**whois**	Nicname 43	**www**	HTTP; 80

Config Extended ACLs: Steps for config'ing extended ACLs same for standard

- First config: Then activated on an int
- Syntax/params more complex
- Internal logic applied to ordering of standard statements doesn't apply to extended
- Order statements are entered during config = Order displayed/processed

? Help

R1(config)# access-list access-list-# { deny | permit | remark } protocol {source *source-wildcard*} [operator port [*port #/name*]] {destination *destination-wildcard*} [operator port [*port#/name*]]

Parameter	Description
access-list #	ID's list using # 100-199 (extended IP) and 2000-2699 (expanded IP)
deny	Denies access if conditions match
permit	Permits access if conditions match
remark	Comment
protocol	Name/# of a protocol: icmp/ip/tcp/udp: To match any protocol use ip
source	Network/host packet is being sent from
source-wildcard	Wildcard applied to source
destination	Network/host packet is being sent to
destination-wildcard	Wildcard applied to destination
operator	(Optional) Compares source/destination ports [eq/neq/lt/gt] range: inclusive range
port	(Optional) #/name of tcp/udp port
established	(Optional) TCP only: Indicates established connection

Example:
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

Note: ACL's won't filter until applied to int
Creating Named Extended ACLs

1. ip access-list extended *name* [*global config*] Define name for extended ACL
2. Specify permit | deny conditions [*ACL config mode*]
3. show access-lists [*priv EXEC*] Verify settings
4. copy running-config startup-config Save config

Remove named extended ACL: no ip access-list extended name [*global config*]

Unlike standard ACLs, extended ACLs don't implement the same internal logic/hashing function
Editing Extended ACLs: Accomplished using same process as standard

Inbound/Outbound ACL Logic
Inbound

1. If header/statement match: Rest of statements skipped/packet is permit/deny specified
2. If header doesn't match ACL statement: Packet tested against next statement
3. Continues until end of list

- **At end of every ACL: Implicit deny any statement: Not shown in output**

    1. Final implied statement applied to all packets for conditions didn't test true
    2. Matches all other packets/results in deny
    3. Instead of proceeding in/out int: Router drops all remaining packets

AKA Implicit deny any | or | **deny all traffic** statement

1. ACL should have at least 1 permit or blocks all traffic

Outbound ACL Logic

1. Before packet fwded to outbound int: Router checks table to see if packet routable
2. If packet isn't: Dropped/not tested against ACEs
3. Router checks to see whether outbound int grouped to ACL
4. If not grouped: Packet sent to output buffer

No ACL applied to int	If outbound int not grouped to outbound ACL: Packet sent to outbound int
ACL applied to int	If outbound int grouped to outbound ACL: • Packet not sent on outbound int until tested by combo of ACEs associated w/int • Based on ACL tests: Packet permit/deny

Routing

ACL/Routing: Processes	**1. Packet arrives at r-int:** Process same [ACLs/not] **2. Frame enters int:** R-checks if destination L2 address matches int L2 address/or if frame is broadcast □ If accepted: Frame info stripped off/r-checks for ACL on inbound int □ If exist: Packet tested against statements in list □ If packet matches statement: Permit/deny □ If packet accepted: Checked against table to determine destination int □ If entry exists for destination: Packet switched to outgoing int: Otherwise dropped • **R-checks whether outgoing int has ACL** ○ If ACL exists: Packet tested against list ○ If packet matches statement: Permit/deny ○ If no ACL/packet permitted: Packet encapsulated in new L2 protocol ○ Fwded out int to next device

Standard Decision Process: Standard only examines source IPv4: Destination of packet/ports not considered

IPv6 ACLs: similar to IPv4 operation/config

- IPv4: 2 types: standard/extended: Both can be numbered/named
- no numbered ACLs in IPv6

IPv6 ACLs are:

- Named only
- Same as IPv4 Extended
- IPv4/IPv6 ACL can't share same name

3 differences

Applying IPv6 ACL	**Cmd to apply IPv6 ACL to int:** • **IPv4:** ip access-group Apply ACL to int • **IPv6:** ipv6 traffic-filter Apply ACL to int
No Wildcard Masks	**Doesn't use wildcard masks** • Uses prefix-length (/64) to indicate how much source/destination address matched

Additional Default Statements	**2 implicit permit statements at end of each IPv6 list** • IPv4 end [sta/ext]: Implicit deny any | deny ip any any • IPv6 end: deny ipv6 any any **IPv6 also includes 2 other implicit statements by default:** • **permit icmp any any nd-na** • **permit icmp any any nd-ns** ○ Allow router to participate in IPv6 equivalent of ARP IPv4 ○ ARP: Used to resolve L3 addresses to L2 MAC's ○ IPv6 uses ICMP ND msgs to accomplish same **ND: Neighbor Discovery** Msg **NS: Neighbor Solicitation** ND uses NS & NA msgs **NA: Neighbor Advertisement** • ND msgs: Encapsulated in v6 packets: Require network layer while ARP [v4] doesn't use L3 • B/C IPv6 uses L3 for ND: ACLs need to implicitly permit ND packets sent/received on int • Both ND-NA (nd-na) | ND-NS (nd-ns) msgs permitted

show ipv6 int br Verify address/state of int

Config IPv6 ACLs: 3 steps to config IPv6 ACL:

1. ipv6 access-list name [*global config*] Create IPv6 ACL
    1. Alphanumeric | Case sensitive | Unique

- **No need for standard/extended in ipv6**

1. permit/deny statements [*named ACL config mode*] Specify conditions if packet is fwded/dropped
2. end [*priv EXEC*] Return

R1(config)# ipv6 access-list *access-list-name*
R1(config-ipv6-acl)# deny | permit protocol {source-ipv6-prefix/*prefix-length* | any | host source-ipv6-address} [operator[*port#*]] {destination-ipv6-prefix/*prefix-length* | any | host destination-ipv6-address} [operator[*port#*]]

Parameter	Description
any	Abbreviation for IPv6 prefix ::/0 | Matches all addresses
host	Enter source/destination ipv6 host address

Example:
R1(config)# ipv6 access-list NO-MOO-COW
R1(config-ipv6-acl)# deny ipv6 2001:db8:cafe:30::/64 any
R1(config-ipv6-acl)# permit ipv6 any any
R1(config-ipv6-acl)# end

Applying IPv6 ACL to int: After config: Link to int:
Router(config-if)# ipv6 traffic-filter access-list-name { in | out }
Remove ACL from int: no ipv6 traffic-filter [int] | global no ipv6 access-list (Remove list)
access-class IPv4/IPv6: Apply access-list to VTY

show ipv6 int
show access-lists Displays all lists on router: IPv4/IPv6