Post 19

Thursday, January 24, 2019 11:16 PM

ACCESS CONTROL LISTS P1

Firewall HW/SW that enforce netsec policies
ACL: Access Control List: List of permit/deny statements applied to addresses/protocols
What is an ACL? A series of cmds that control whether router fwds/drops packets based on info in packet header
ACLs:

Limit traffic	Increased performance • Example: No video traffic? ACLs can block
Traffic flow control	Can restrict r-updates if not needed: BW saved
Basic Security	Can allow 1 host to access parts of network | stop others • Example: Access to HR restricted to certain usrs
Filter traffic by type	Can permit email, but block Telnet
Screen hosts	Permit/deny access to services • Example: FTP | HTTP

Default: No ACL config: Routers don't filter specific traffic: Traffic is routed from info in table
When applied to int: Router evaluates ALL packets passing int to determine if it can be fwded

- ACLs used to select types of traffic for analyzing/fwding/processing
  - Example: Classify traffic to enable priorities

TCP: Enables admins to control traffic in/out network

TCP Communication	▪ Client requests data from web server: IP manages comm bet PC [source]|server [destination] □ Manages communication bet application | server ▪ Breaks data down into segments for IP before sent | Assembled from segments when arrived ▪ Connection-oriented/reliable/byte stream service: □ **Connection-oriented:** 2 apps must establish TCP connection prior to data exchange □ Full-duplex protocol: Each connection supports pair of byte streams: Streams 1 direction Flow-control mechanism for each stream: ○ Allows receiver to limit how much data sender can transmit | Congestion-control

SYN Starts session | **ACK** Acknowledges segment received | **FIN** Finishes session | **SYN/ACK** Acknowledges transfer sync'd

- Data segments include higher lvl protocols needed to direct data to correct app
- ID's port that matches requested service | HTTP = 80 | SMTP = 25 | FTP = 20/21

TCP PORTS: Range 0-1023: **Well-known** | 1024-49151: **Registered** | 49152-65535: **Private/Dynamic**

Well-known TCP	○ 21 FTP ○ 23 Telnet ○ 25 SMTP ○ 80 HTTP ○ 143 IMAP ○ 194 IRC ○ 443 HTTPS	**Registered TCP**	○ 1863 MSN Messenger ○ 2000 Cisco SCCP (VoIP) ○ 8008 Alternate HTTP ○ 8080 Alternate HTTP

UDP Ports: 0-1023: Well-known | 1024-49151: Registered | 49152-65535

Well-Known UDP	○ 69 TFTP ○ 520 RIP	**Registered UDP**	○ 1812 RADIUS Authentication Protocol ○ 5004 RTP (Voice/Video Transport Protocol) ○ 5060 SIP (VoIP)

TCP/UDP Common Ports

Well-Known	○ 53 DNS ○ 161 SNMP ○ 531 AOL Instant Messenger, IRC	**Registered**	○ 1433 MS SQL ○ 2948 WAP (MMS)

How ACL uses info passed during TCP/IP conversation to filter traffic
Packet filtering: AKA Static packet filtering: Controls access to network: Analyzes in/out packets/passing/dropping on criteria

Source IP	Destination IP	Protocol w/in packet

Router	**Acts as packet filter when fwds/denies packets according to filter rules** 1. Packet arrives at filtered router: Extracts info from packet header 2. Router makes decisions: Based on config rules: Can packet pass/be discarded? 1. Works at diff layers of OSI | Internet layer of TCP/IP 2. Packet-filtering router: Uses rules to determine permit/deny traffic 3. Can also perform filtering at L4 [Transport] 4. Can filter packets based on source/destination port of TCP | UDP segment 5. Rules defined using ACLs

ACL: AKA ACE: Access Control Entries: Sequential list of permit/deny statements

- ACE: Created to filter traffic based on certain criteria like:

Source/destination address	**Protocol**	**Port #s**

When traffic passes through int config'd w/ACL: Router compares info w/in packet against each ACE

- In order: Determines if packet matches 1 of statements | If match: Packet processed: Network/subnet

ACL extracts info from L3 packet header:

Source IP	**Destination IP**	**ICMP msg type**

Can extract L4 layer info from header: TCP/UDP source | TCP/UDP destination ports
IF packet **TCP SYN** from **Network A** on **Port 80**: Allowed *All other access denied to those usrs*
IF packet **TCP SYN** from **Network B** on **Port 80**: Blocked *All other access permitted*
ACL Operation

- ACLs define sets of rules that give control for packets that enter inbound ints/relay/exit outbound ints.
- They don't act on packets that originate from the router itself
- Can apply to inbound/outbound traffic

Inbound	Incoming packets processed before they are routed to outbound int • Efficient: Saves overhead of lookups if packet discarded • If permitted: It's processed for routing • Filters packets when network attached to inbound int is only source of packets examined
Outbound	Incoming packets routed to outbound int: Then processed through outbound ACL • Same filter applied to packets coming from multiple inbound ints before exiting same outbound int

Last statement of an ACL: ALWAYS implicit deny

- Statement auto inserted at end of each ACL even though not physically present
- **Implicit deny blocks all traffic**
- If an ACL that doesn't have at least 1 permit statement: It will block all traffic

Types of Cisco IPv4 ACLs: Two types

- **Standard ACL**
- **Extended ACL**

Standard	**Can be used to permit/deny traffic only from source IPv4** • Destination packet/ports involved NOT evaluated • Created in global config Example: access-list 10 permit 192.168.30.0 0.0.0.255 **• Filter based on source address only**
Extended ACLs	**Filter packets based on 7 attributes:** • Protocol type | Source IPv4 | Destination IPv4 | Source TCP/UDP ports | Destination

	TCP/UDP ports • Optional protocol type info • Created in global config Example: access-list 103 permit tcp 192.168.30.0 0.0.0.255 any eq 80

Numbering/Naming ACLs: Can be created using #/name to ID ACL/list of statements

Numbered ACL	**Assign # on protocol to be filtered:** • 1-99 | 1300 | 1999: Standard IP ACL • 100-199 | 2000-2699: Extended IP ACL • 200-1299: Skipped: Other protocols/obsolete/legacy
Named ACL	**Assign name to ID:** • Alphanumeric • CAPITAL LETTERS • No spaces/punctuation • Added/del w/in ACL

Wildcard Masking: IPv4 ACEs include wildcard masks
Wildcard mask: A string of 32 binary digits used by router to determine which bits of address to examine for match

- **IPv6: No wildcard masks:** Prefix-length used to indicate how much IPv6 source/destination address matched
- 1/0 in wildcard mask ID how to treat corresponding IP bits: These bits are used for different purposes/rules

Subnet masks: Use 1/0 to ID network/subnet/host portion of IP
Wildcard masks: Use 1/0 to filter individual IP's/groups of IP's to permit/deny access to resources

- Differ in way match 1/0

Wildcards use following	**0:** Match corresponding bit value in address **1:** Ignore corresponding bit value in address • **AKA inverse mask** • Subnet mask: 1 = match | 0 = 0 not match Used when config IPv4 r-protocols [OSPF] to enable it on specific ints

Wildcard Mask Keywords

- Host/any Help ID most common uses of wildcard masking
- Eliminate entering wildcard masks when ID specific host/entire network
- Can also be used w/IPv6 ACL

host

- **Subs 0.0.0.0 mask:** States all IPv4 bits must match/only 1 host matched

Router(config)# access-list 1 permit 192.168.10.10 0.0.0.0
Router(config)# access-list 1 permit host 192.168.10.10
any

- **Subs IP/255.255.255.255 mask:** Says to ignore entire IPv4/accept any addresses

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

Guidelines: For every int: May be multiple policies needed to manage traffic allowed to enter/exit

- Use in firewall routers: Positioned bet internal/external network [Internet]
- Use on router positioned between 2 parts of network to control traffic entering/exiting specific part
- Config on border routers [routers at edge of network]: Provides buffer from outside
- Configure for each protocol on border r-ints

3-Ps: Can config 1 ACL per:

- **Protocol**
- **Per direction**
- **Per int**

Per protocol	Control traffic flow on int: ACL must be defined for each protocol enabled on it
Per direction	Control traffic in 1 direction at a time on an int: 2 separate ACLs must be created to control in/out traffic
Per int	Control traffic for an int: Example: GigabitEthernet 0/0 (g0/0)

Best Practices:

Guideline	Benefit
Base on sec policy of org	Ensures organizational sec guidelines
Prepare description of what you want them to do	Helps avoid access problems
Text editor to create/edit/save	Creates a reusable lib (common sense)
Test before implementing	Avoids errors

Where to Place: Basic rules:

Extended ACLs	○ As close to source of traffic filtered as possible ○ Undesirable traffic? Denied w/out crossing infrastructure
Standard ACLs	○ As close to destination as possible ○ Placing at source of traffic will prevent it from reaching other networks through the int applied

Placement of ACL/type depends on:

1. **Admin control:** Whether/not there is control of both source/destination networks
2. **BW:** Filtering unwanted traffic [source] prevents BW loss on path to destination
3. **Config ease:** Deny traffic coming from networks? Standard on router closest to destination
    1. Disadvantage: Traffic from those networks will use BW: No reason
    2. Extended could be used on each router where traffic originated
    3. Save BW by filtering traffic at source: Requires extended ACLs on multiple routers

Standard Placement	Only filter based on source • As close as possible to destination • Allows traffic to reach other networks except where packets filter
Extended Placement	Can filter based on Source | Destination address | Protocol | Port # • Flexibility in type of traffic filtered/placement • As close to source as possible • Prevents unwanted traffic from being denied when it reaches destination

Criteria Statements

- Traffic enters router: It compares to all ACEs in order entries occur in ACL
- Processes ACEs until finds match: Will process packets based on 1st match found/no other ACEs examined
- If no matches: Traffic denied
- Default: Implied deny at end of all ACLs for traffic not matched to config entry
- A single-entry ACL w/1 deny has effect of denying all traffic: At least 1 permit in ACL or all traffic blocked

Config Standard ACL: 1st create standard ACL/then activate on an int

access-list [*global config*] Defines standard ACL w/a # 1-99

- IOS 12.0.1: Extended by 1300-1999 | Max of 798 possible | Add #'s AKA **expanded IP ACLs**

Router(config)# access-list access-list-number { deny | permit | remark } source [source-wildcard][log]

access-list-#	# of ACL: 1-99 | 1300-1999 (Standard ACL)
deny	Denies access if conditions matched
permit	Permits access if conditions matched
remark	Remark about entries in access list to make easier to understand: Limited to 100 chars
source	# of network host/host packet is being sent **2 ways to specify:** • 32bit quantity in 4-part decimal fmt • any as abbreviation for source/wildcard 0.0.0.0 | 255.255.255.255
source-wildcard	**Optional:** 32-bit mask to apply to source: Places 1's in bit positions to ignore
log	**Optional:** Causes info logging msg about packets that match entries to be sent to console

	• Msgs logged controlled by logging console • Msg includes: ACL # | Whether packet permit/deny | source address | # of packets • Msg generated for 1st packet match and at 5-min intervals • Includes number of packets permit/deny in the previous 5-min interval

R1(config)# access-list 10 permit host 192.168.10.10
R1(config)# access-list 10 permit 192.168.10.0 0.0.0.255

no access-list [*global config*] Remove ACL
show access-list Confirms access list removed

Standard ACL Config
R1(config-if)# ip access-group { access-list-number | access-list-name } { in | out }
Removal: no ip access-group on int | no access-list

R1(config)# access-list 1 permit 192.168.10.0 0.0.0.255
R1(config)# int s0/0/0
R1(config-if)# ip access-group 1 out

1. access-list to create entry in IPv4 ACL (remark to add description)
2. Select int to apply ACL
3. Activate existing ACL on int

Named Standard ACLs

1. ip access-list [*global config*] Create named ACL [Alphanumeric/case sensitive/unique]
    1. ip access-list standard *name* Create standard | or | ip access-list extended *name*Extended
2. permit/deny statements to specify conditions [named ACL config mode]
3. ip access-group Apply ACL to int: Specify if applied as packets enter int in or out

Commenting access-list # remark comment [*global config*] | To remove use no