LAB 1

* **MAC flooding**
* use macof command
* when switch address-table is full, all frames are broadcasted
* only unicast addresses will work (8th bit of MAC is 0)
* need to flood for at least 300s for old entries to be removed (aging-time)
* to block, enable port security (maximum) to limit MAC per end device

**Switch spoofing / VLAN hopping**

* use yersinia command -> DTP -> enable trunking
* form 802.1Q trunk to receive all frames from all configured VLANs
* should be used with MAC flooding to receive all frames
* vulnerable switches using DTP (dynamic auto / dynamic desirable)
* to block, shutdown all unused ports or move them to unused & removed VLAN (removing prevents trunk from carrying frames from the VLAN)

LAB 2

**Double frame tagging**

* use yersinia command -> 802.1Q -> double enc. packet
* 802.1Q in 802.1Q (Q-in-Q) is used in LAN (inner) + metropolitan-area Ethernet (outer)
* access switch MUST be configured to accept VLAN tags (by enabling VoIP on data line on a different VLAN)
* native VLAN tag is stripped at access switch before going trunk for backward compatibility
* will only work if attacking port = native trunk VLAN & switchport voice is enabled
* attack is only one-way, mainly for DOS
* to block, ensure native VLAN is not used for any access ports or force all traffic to be tagged (newer switches)

**Spanning Tree Protocol (STP) attack**

* use yersinia command -> STP -> claiming root role
* use Switch Port Analyzer (SPAN) to forward from rouge switch to attacker
* attacker connect to 2 STP-blocking ports to inject BPDU with lower bridge ID to take over root switch and capture frames
* to block new endpoint <-> switch, enable portfast (enables BPDU guard) or manually enable BPDU guard
* to block existing switch <-> switch, enable root guard on root switch

**Network information leakage (via Cisco Discovery Protocol (CDP))**

* use wireshark to capture CDP packets
* CDP is used by cisco devices to share device information (VLAN ID, IP, etc)
* to block, disable CDP except on VoIP ports which require it

LAB 3

**DHCP Starvation Attack**

* use yersinia command -> DHCP -> sending DISCOVER packet
* usable IP addresses get exhausted
* port security alone is insufficient as Client Host Address (chaddr) in DCHP Application layer can be spoofed while maintaining 1 mac in Ethernet layer

**Rouge DHCP Server (DHCP Spoofing)**

* use Metasploit -> use auxiliary/server/dhcp -> set <options in show options> <ips>
* to be used after DHCP starvation to ensure attacker is the only DHCP server

**Rouge DNS Server (DNS Spoofing)**

* use dnschef --interface <your ip> --fakeip <fake webserver>
* redirect all dns traffic of hosts connecting to you to a fake webserver (usually for stealing credentials)
* only works if you are the default gateway

**Blocking above 3 attacks**

* enable DHCP snooping -> ip dhcp snooping
* perform deep packet inspection and verify Ethernet layer MAC = Application layer chaddr -> ip dhcp snooping verify mac-address
* set DHCP snooping rate limit to prevent multiple DHCP DISCOVER packets from 1 port
* when DHCP snooping is enabled, all untrusted ports are disallowed to send DHCP OFFER & DHCP ACK packets, preventing rouge server. To trust actual DHCP server, trust port -> interface <if>; ip dhcp snooping trust
* when DHCP snooping is enabled, option-82 agent field is inserted into DHCP DISCOVER packet. Since relay-agent IP is 0.0.0.0 it will be discarded by next hop switch or router. To fix -> [DHCP Router] ip dhcp relay information trust-all OR [Relay Switch] no ip dhcp snooping information option

**ARP Poisoning (leading to Man in the Middle Attack)**

* use bettercap -> net probe on -> net probe off -> net show -> set arp.spoof.targets <ip> -> set arp.spoof.fullduplex true -> arp.spoof on -> net.sniff.local true -> net.sniff on
* attacker sends gratuitous ARP reply to victim advertising that router's IP is bound to attacker's MAC
* attacker sends gratuitous ARP reply to router advertising that victim's IP is bound to attacker's MAC
* all communications now pass through attacker who can monitor and modify traffic
* use driftnet to see images
* use SSL stripping to bypass HTTPS/HSTS
* to block, enable Dynamic ARP Inspection (DAI) with DHCP Snooping (to allow comparison with DHCP Snooping Binding Table) -> ip arp inspection -> ip arp inspection trust (on DHCP server)
* to block without DHCP Snooping, configure DAI with ARP ACL -> arp access-list <acl\_name> -> permit ip host <sender-ip> mac host <sender-mac> -> ip arp inspection vlan <vlan> -> ip arp inspection filter <acl\_name> vlan <vlan> -> (trust ports between network devices)
* to block probing for devices on network, set ARP rate limit (default 15) -> ip arp inspection limit <n>
* DAI can also validate contents of ARP reply (between L2 & L3) to match Ethernet (L2) and IP (L3) layer to prevent bypassing port security -> ip arp inspection validate <src-mac/dst-mac/ip>

**IP/MAC Spoofing (bypass ACL)**

* change based on OS
* bypass ACL rules
* to block, enable IP source guard with DHCP snooping -> ip verify source (port-security)

LAB 4

* Standard numbered ACL is used to filter network traffic based on source IP addresses, at Layer 3 only. (any number from 1-99 or 1300-1999)
* Extended numbered ACL is more powerful as it can filter network traffic based on protocol type, source/destination IP addresses, and source/destination port numbers, at Layer 3 and 4 based on more attributes. (any number from 100-199 or 2000-2699). At a minimum, the test-conditions must specify protocol type, source IP address and destination IP address. For tcp and udp protocol type, the test-conditions can further specify to filter based on layer 4 source port numbers and/or destination port numbers if need to.
* Named ACL enables ACL to be configured with meaningful name to make it easier to remember. (any alphanumeric characters)
* Time-based ACL enhances ACL with time range to automatically turn on/off ACL which can be useful in practice. Time-based ACLs are basically ACLs enhanced with time-range which will only become operational during certain time of the day or week
* Object group-based ACL is a new way of writing ACL to make it more readable and manageable. First, configure related objects into object groups:
  + Network
    - -host IP
    - -IP ranges
    - -subnet
    - -etc
  + Service
    - -protocol
    - -port number
    - -etc
    - -e.g. tcp www

Action taken: Permit or deny

Conditions: Standard -> matches source IP address

Extended -> matches source IP address, destination IP address, protocol type, port number

\*\*ACLs are processed from top to bottom, i.e. packet will get checked against the rule in the first line, then the second etc.

\*\*Whitelist ACLs should usually end with a default (deny ip any any), to deny all other traffic.

\*\*Blacklist ACLs should end with a default (permit ip any any), to allow all other traffic passing through

\*\* Typically, ACL is written with more specific ACEs (in terms of IP addresses) at the top and more general ACEs at the bottom.

**General guidelines on ACL:**

* One ACL per direction: ACL can only filter packets in one direction as configured. To filter both ingress and egress traffic, two separate ACLs must be written.
* One ACL per interface: Only one in ACL and one out ACL can be applied on each interface.
* One ACL per protocol: Separate ACLs must be written to filter IPv4 and IPv6 packets (not covered in this module)
* Commonly applied on the switch SVIs or router interfaces to filter ingress or egress IP packets, usually to and from the Internet.
* Standard ACLs can only filter based on source IP. To avoid unintentionally filtering more packets than necessary, it is recommended to be implemented near the destination
* numbered ACL is written by (1) specifying the related ACEs using the same access-list-number, and (2) applying the complete ACL on the required interface
* More powerful extended numbered ACL for controlling access to networks or end hosts/servers

**Ingess-Filtering ACL**

* Router filters all IP packets received on that interface
* Only permitted packets are then processed and forwarded

**Egress-Filtering ACL**

* Only after forwarding decision is made
* IP packets are then filtered before permitted to be sent out from that interface

------------------------------------------------------LAB 5 Routers---------------------------------------------------------

* Usually first point of contact and entry
* To prevent external attackers, ACLs (access control lists) are commonly implemented at the perimeter/edge router and firewall which connect the enterprise network to the Internet

**Recon**

* Network probing
* Port scanning attack
* Technically, network probing and scanning attacks are based on sending ARP, ICMP, TCP or UDP requests and receiving/not receiving the replies
* Unfortunately, it can be difficult to fully prevent network probing and scanning attacks cus of normal traffic; but some may be blocked by suitable ACLs; e.g. ingress filtering of ICMP scan.

Graphical user interface, text, application

Description automatically generated

* No ip unreachables will prevent a router configured with ACL to drop packets from sending ICMP type 3 destination unreachable which will leak information to attackers
* implement egress filtering of ICMP to prevent someone inside your network from scanning outside destinations, be it intentionally or unknowingly. However, there is a need to still allow administrators to do pings as it is useful for testing network connectivity. Trade-off for allowing ICMP error-reporting messages vs attacks; 1. unreachable: inform destination unreachable but exploited by UDP scan, 2. time-exceeded: exploited by attacker to trace route to target

**Smurf Attack with spoofed IP address.**

* A DDos attack that is a reflection attack.
* By sending packets to intermediatory victims with a spoofed source IP address (the one of the target), the target is flooded by replies from the intermediatory victims. For example, spamming pings at the hosts of a network with the spoofed source IP of the server, causing the server to be flooded with ICMP replies from the hosts.
* In smurf attack, a single broadcast ping sent to a directed broadcast address by attacker using spoofed IP source address belonging to target would result in multiple ping replies (each from an intermediary in the directed broadcast subnet) reflected to the target.
* Prevention:
  + - * Disable routers from supporting directed broadcast;
      * Disable hosts from replying to broadcast ping.
      * In addition, organizations should implement anti-spoofing ACL at their edge routers to prevent being victim to attacks using spoofed IP addresses. Typically, anti-spoofing ACL is implemented as a part of overall infrastructure protection ACL at the edge router for ingress filtering of malicious traffic from entering the network.

**Tiny Fragments and overlapping fragments**

* However, blocking external attackers from reaching internal hosts will also lead to blocking internal hosts from accessing the Internet because all returning packets will be blocked
* So, ACL statement has been enhanced with an optional ‘established’ parameter for protecting internal hosts. The ‘established’ option applies only to TCP and is meant to:
  + - * Allow returning TCP segments of existing connections
      * Block TCP segments initiating a new connection from outside, or other malformed segments
* Or configure ACL to examine TCP Flags.
  + - * Allow returning TCP segments of existing connections, e.g. TCP flags with +syn +ack, +ack, +fin, +fin +ack, +rst
      * Block other TCP segments, e.g. +syn because it is the first segment of 3-way handshake initiating a new connection

**Defense**

* To defend against tiny fragment and overlapping fragment attacks, Cisco has implemented RFC 1858 to drop IP fragment carrying TCP with fragment offset = 1 (FO=1)
* Alternatively, filtering of non-initial fragments can also be done by configuring ACL statements with the optional ‘fragments’ parameter.
* Timeline

  Description automatically generatedHP Laptop drops fragment 0

Graphical user interface, application

Description automatically generatedText, timeline

Description automatically generated with medium confidence------------------------------------------------------------------Zen Notes-----------------------------------------------------

LAB 4

\* Only 1 ACL per interface

**Inbound (ingress) ACL**

* filter all IP packets received on interface
* only permitted packets are processed and forwarded

**Outbound (egress) ACL**

* only after forwarding decision has been made
* IP packets are then filtered before permitted to be sent out of that interface
* packets originating from router itself are NOT fitered

**Standard numbered ACL**

* 1 to 99 or 1300 to 1999
* filter at layer 3 based on source IP ONLY
* implemented near to destination to prevent certain IPs from connecting

**Extended numbered ACL**

* 100 to 199 or 2000 to 2699
* filter at layer 3 and 4 based on more attributes
* implemented near to source to save bandwidth as unnecesary packets are dropped

**Named ACL**

* uses alphanumeric characters

LAB 5

**Network probing and scanning attack**

* use nmap
* to block, use ACL -> deny icmp any any echo/timestamp-request/mask-request -> permit ip any any -> apply to ingress from internet facing int
* to block, go to int -> no ip unreachables (no reply denied message sent back to attacker)

**Reflection / amplification attack (DDOS)**

* use hping3 -1 --flood -a <target\_ip> <directed\_broadcast\_addr>
* spoof source IP as target, destination IP as each host in an intermediary network via directed broadcast address
* directed broadcast address is a broadcast address which can be forwarded to the next router (i.e 171.1.255.255 will target 171.1.x.x hosts in the network)
* to block, disable routers from supporting directed broadcast (default)
* to block, disable hosts from replying to broadcast ping (default)
* to block, ISPs should apply BCP 38 (allow only allocated IP out of interface)
* to block spoofing, organisations should filter their own allocated external IP and all internal IPs (spoofed) from entering edge router

**Tiny fragment / overlapping fragment attack**

* use fragrouter -i eth0 -F1
* when ACL filters by established connection (permit tcp any host <dst> established), it filters by dropping SYN flag in TCP header to prevent new TCP connection but allow existing
* when fragmenting to 1 byte, TCP flags are cut off into second fragment. Since ACL only scans first fragment, outside hosts are able to establish a connection with SYN in the second fragment (tiny fragment)
* alternatively, second fragment can have desired fragment offset (FO=1) to overlap part of first fragment's TCP header and be reassembled to form an attack payload
* once even a single fragment (first) is blocked, frame cannot be reassembled even if non-initial fragments pass through
* to block, cisco implemented blocking of FO=1 fragments (enabled by default when using ACL)
* to filter non-initial fragments, suffix all rules with "fragments" (i.e deny tcp host <ip> fragments)

-------------------------------------------------------------Martin Notes---------------------------------------------------------

**Topic 1**

* Switch MAC address table
  + Self-learning (automatically fills up)
  + Address known → Unicast to specific port
  + Address NOT known → Broadcast out all ports (except incoming)
  + Switch MAC address table is not infinite size
  + Multicast addresses are NOT learnt by switches
    - Last bit of first octet determines if it is a multicast (0 = unicast, 1 = multicast)
  + Entries in the table will not last forever and be removed automatically
    - Default aging time 300 seconds
    - Mac address-table aging-time [seconds] [vlan vlan-id]
    - No mac address-table aging-time (return to default)
    - Mac address-table aging-time 0 (disable)
* MAC flooding attack
  + Flood MAC address table with fake source MAC addresses until its full (macof)
  + MAC address is made up of two parts
    - Organizational Unique Identifier (OUI) – First 3 octets (24 bits)
    - Network Interface Controller (NIC) – Last 3 octets (24 bits)
* Preventing MAC flood
  + Enable port security
  + Limits MAC addresses allowed to access its switch interface
  + Switchport port-security [mac-address/sticky/maximum]
    - Dynamic (default) – MAC addresses learnt automatically from connected devices. Removed when state goes down or switch reboots.
    - Static – MAC addresses statically configured into running config. Can be made permanent into startup config
    - Sticky (hybrid) – MAC addresses learnt automatically from connected devices. Entries are added to running config like static, also removed unless saved into startup config.
    - Maximum – Specify max number of static secure MAC addresses tied to a single interface
* Verifying Port-Security & Violation
  + Show port-security / show port-security interface [x]
  + Switchport port-security violation [protect/restrict/shutdown] (incremental security level)

|  |  |  |  |
| --- | --- | --- | --- |
| Action / Port Security Level | Protect | Restrict | Shutdown |
| Discard offending traffic | Yes | Yes | Yes |
| Send logs & SNMP msgs | No | Yes | Yes |
| Disable interface & traffic | No | No | Yes |

* + In shutdown, if violated, ports are moved to err-disabled state
  + There are 2 ways to recover:
    - Manual (Go to the interface and shut / no shut it)
    - Auto (Reduce administrative overhead)
      * Errdisable recovery cause p-secure violation
      * Errdisable recovery interval [seconds]
* Switch spoofing
* In VLANs:
  + Access – Carries frames for one VLAN configured on port
  + Trunk – Carries frames for multiple VLANs on a single physical link
  + Dynamic Trunking Protocol (DTP)
    - Default mode: Dynamic Auto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mode | **Access** | **Dynamic Auto** | **Trunk** | **Dynamic Desirable** |
| **Access** | Access | Access | DO NOT USE | Access |
| **Dynamic Auto** | Access | Access | Trunk | Trunk |
| **Trunk** | DO NOT USE | Trunk | Trunk | Trunk |
| **Dynamic Desirable** | Access | Trunk | Trunk | Trunk |

* + Yersinia (Kali Linux)
    - Attacker injecting DTP messages to form trunks on other unused ports still in Dynamic Auto
    - Once switch is spoofed, attacker can receive and also send more spoofed frames to other switches
    - Sudo Yersinia -I (interactive) for GUI mode
* Preventing Switch Spoofing
  + Disable DTP negotiations
  + Manually configure ports to access / trunk explicitly
    - Switchport nonegotiate
  + Shutdown unused ports and placed them in an unused VLAN
    - Switchport access VLAN 100
    - Switchport trunk allowed vlan remove 100

**Topic 2**

* Double Tagging attack
* In trunk ports, different VLAN traffic are tagged differently (IEEE 802.1Q standard)
  + 1 native VLAN – frames are not tagged
  + All other VLANs – frames are tagged with appropriate VLAN ID
  + The tag is also used to support VOIP
    - To support such traffic switch ports need to be configured with a data vlan and auxiliary vlan
    - Switchport mode access
    - Switchport access vlan 10
    - Timeline

      Description automatically generatedSwitchport voice vlan 11Text

      Description automatically generated with low confidence
* Double 802.1Q tags are later developed and supported for metropolitan networks for VLANs going across an ISP to another VLAN (Q-in-Q)
  + Contains an outer 802.1Q tag of ISP & inner 802.1Q tag of the communicating VLANs
* As native traffic is NOT tagged, VLAN IDs are stripped off
* However an attacker can send a frame tagged for VLAN 10 (1st tag) with an additional tag for VLAN 20 to hop traffic
  + This attack ONLY works if attacker’s access port has the same VLAN ID as the native VLAN of the trunk, default is VLAN 1
* Yersinia can also be used to conduct such an attack
* Prevention of Double tagging
  + Configure a dedicated native VLAN (instead of the default vlan 1) for all trunk ports that are not assigned to any VLANs / access ports
    - Switchport trunk native vlan 500
  + Some switches also have a feature where it forces all traffic to be tagged including native VLAN
    - Vlan dot1q tag native
* STP attack
  + Switches are configured with STP (802.1D) for interconnectivity and redundancy to prevent loops
  + However, there is no security in this configuration
  + Attackers can inject Bridge Protocol Data Unit (BPDU) with lower BID to take over the root switch and capture frames
* SPAN
  + Frames can be copied from a switch to a laptop using Switch Port Analyzer (SPAN)
    - Can be configured to capture only received frames, only transmitted frames, or both, from a source port
    - Monitor session [x] source interface [port] both
    - Monitor session [x] destination interface [port]
    - X is the monitor session number and it can be any number from 1 to 18
  + Verifying SPAN
    - Show monitor session all
    - Show monitor detail
* Preventing STP attacks
  + Enable/apply BPDU guard to each interface individually that is NOT supposed to received BPDU messages
    - Spanning-tree bpduguard enable
  + Enable/apply BPDU guard on all Portfast-enabled ports (Layer 2 LAN interfaces should NOT receive BPDU)
    - Spanning-tree portfast bpduguard default
  + BPDU guard enabled ports when violated will be moved to err-disabled state. Recovery similar to Topic 1 port-security
  + Ports transition from listening to forwarding before being either placed into the forwarding or blocking state to prevent loops
    - The transitioning phase can be speed-up by configuring portfast
      * Spanning-tree portfast

|  |  |  |  |
| --- | --- | --- | --- |
| State | Forward Data Frames | Learns MAC address | Stable / Transitory |
| Disabled | No | No | Stable |
| Blocking | No | No | Stable |
| Listening | No | No | Transitory |
| Learning | No | Yes | Transitory |
| Forwarding | Yes | Yes | Stable |

* + Additionally, root switches can be configured with root guard to prevent them from being taken over by a rogue root switch
    - Can be configured on the distribution layer switch port connected to the access layer switch (since access switches should never be the root bridge)
    - Spanning-tree guard root
    - If root guard port receives superior BPDUs:
      * Transits to root inconsistent state (equivalent to listening state which stops forwarding traffic)
      * Recovers automatically when superior BPDUs stop
* Cisco Discovery Protocol
  + Device automatically sends CDP frames to advertise itself
  + Runs on all Cisco network devices which can reveal information that could be useful for attackers
    - OS version,
    - VLAN ID,
    - Auxiliary voice VLAN ID,
    - IP addresses,
    - etc
  + Has NO security
  + By listening to CDP frames, the switch’s neighbors can be determined
    - Show cdp neighbors
    - Show cdp neighbors detail

**Network Probing & Scanning**

Diagram

Description automatically generated

**Topic 3**

* DHCP Starvation
  + Dynamic Host Configuration Protocol (DHCP) is used for hosts to automatically obtain IP addresses, default gateway, DNS.
    - DHCPDiscover
    - DHCPOffer
    - DHCPRequest
    - DHCPAck
  + DHCP has no security however;
    - Attacker may generate large number of DHCP discover messages to seize all IP addresses given by DHCP server
    - AKA Starvation / Exhaustion / DoS
    - In such attacks only DHCP Discover messages are sent without any DHCP Offer. Thus, the IP addresses offered will be reclaimed after a period of time (a few minutes)
    - This reclaim time is different from the lease time when the IP address is accepted and leased out to hosts (default 24 hours lease time once accepted)
  + Again, Yersinia can be used to conduct such attacks
    - Port security is NOT enough to stop such attacks
    - Attacker may still generate many DHCP Discover messages without violating the port security
      * DHCP is at application layer, layers are removed before server receives final DHCP message (Packets go through a network go from Physical to Application layer)
      * The source MAC address at application layer is a random
      * The source MAC address at data link layer (layer 2) is fixed
    - After the starvation, the attacker may conduct other attacks
* DHCP Spoofing
  + Metasploit tool in Kali Linux
    - Use auxiliary/server/dhcp
    - Set SVRHOST / NETMASK / DHCPIPSTART / DHCPIPEND / ROUTER / DNSSERVER
    - Show options
* Rogue DNS
  + After spoofing the DHCP server, the attacker can also setup a rogue DNS Server and create a spoofed website to masquerade as an authorized user using the stolen information
  + DNSChef tool in Kali Linux
    - Dnschef - - interface [listening interface ip address] - - fakeip [spoofed website/attacker’s IP]
  + To Show DNS information on Linux
    - cat /etc/resolv.conf
* Preventing DHCP Spoofing
  + DHCP Snooping enables switches to perform deep-packet inspection
  + Verifying the layer 2 source MAC addresses to match the application layer DHCP client hardware address (chaddr)
    - Drop the packet if no match
  + All ports are untrusted by default
    - DHCP Offer & Ack messages not allowed
    - Connected ports of authorized devices must be manually configured to be trusted
* Global Config mode
  + Ip dhcp snooping
    - Enable snooping
  + Ip dhcp snooping verify mac-address
    - Prevent DHCP starvation
* Interface Config mode
  + Ip dhcp snooping trust
    - All ports by default are not trusted so should configure the correct port to trust that connects to the server to prevent spoofing
  + Ip dhcp snooping limit rate [rate]
    - Port becomes err-disabled if limit is exceeded (1-2048 pps)
* Option-82 (relay agent information option)
  + This field will be automatically added when snooping is enabled
  + Contains circuit ID (VLAN & port number) & remote ID (base MAC address) of the switch
  + These are information about the location of the client that sends the DHCP Discover message
  + However, the switch is not a relay agent and the DHCP packet with Option-82 (giaddr) with relay agent IP address zeroes will be discarded by the next DHCP agent / server
  + Do **disable Option-82** for the DHCP server to trust DHCP messages
    - no ip dhcp snooping information option
  + Do enable the router to trust DHCP messages
    - ip dhcp relay information trust-all OR
    - ip dhcp relay information trusted
* Binding table
  + Used by switches to support DHCP snooping
  + Contains DHCP assigned addresses for untrusted ports (to defend from other attacks)
  + Upon receiving DHCPACK, new entry is added
  + Upon receiving DHCPNack, DHCPRelease, DHCPDecline, corresponding entry is removed
* ARP Poisoning
  + Hosts will update its ARP cache with sender’s IP/MAC upon receiving an ARP Request / Reply
  + There is a GARP (gratuitous ARP) that attackers can use for ARP announcement (of information) or unsolicited ARP replies WITHOUT an ARP request —> Poisoning the ARP cache of victim
  + Can lead the victim into Man-in-the-Middle (MitM) attacks (after changing default gateway, etc.)
  + Bettercap tool in Kali Linux
    - Sudo bettercap
    - Net.probe [on/off]
    - Net.show
    - Set arp.spoof.targets [victim IP]
    - Set arp.spoof.fullduplex true
    - Arp.spoof on
  + Enabling sniffing (to capture credentials if not encrypted)
    - Set net.sniff.local true
    - Net.sniff on
  + Driftnet tool in Kali Linux
    - Sniff & display images victim is viewing
    - Sudo driftnet -i [interface]
    - Files are stored in /tmp/driftnet
* HTTP Strict Transport Security
  + HTTP Strict Transport Security (HSTS) is a security mechanism that is used for web sites to declare to browsers to access via secure connections only
  + Attackers may bypass HSTS or perform SSL stripping
    - Direct users to wwww.google.com/ instead of https://www.google.com/
* Preventing ARP Poisoning attacks
  + Switches with DHCP Snooping enabled can be further configured with Dynamic ARP Inspection (DAI) to drop invalid ARP traffic
  + After enabling DAI, all ports are untrusted, similarly to DHCP snooping
  + IP & MAC addresses in ARP packets received on untrusted ports will be validated against the DHCP binding table
  + Switch with DHCP Snooping & DAI
  + Each port can also be configured with an ARP rate-limit
    - Default value of 15 pps
    - If violated, port goes to err-disabled state
    - Ip arp inspection limit [rate in pps] [burst interval in secs]
  + Additionally, DAI can be configured to validate contents in ARP reply messages
    - Ip arp inspection validate [src-mac] [dst-mac] [ip]
      * **src-mac** – validate layer 2 source MAC against sender hardware address
      * **dst-mac** – validate layer 2 destination MAC against target hardware address
      * **ip** – validate invalid and unexpected IP addresses (multicast addresses, 0.0.0.0, 255.255.255.255)
* DAI in Non-DHCP environment
  + ARP ACL can be used to configure IP to MAC mapping manually
  + The static keyword is optional and used to indicate to check ARP ACP only
  + By default, intercepted ARP on untrusted ports are first matched with ARP ACL, and if no match, then the DHCP snooping table
* IP & MAC address spoofing
  + Access Control Lists (ACLs) can be used to filter traffic in VLAN routing.
    - E.g. Bob is the only one allowed to access Admin VLAN
  + Attackers still can access the server by performing IP/MAC address spoofing
  + Devices will just forward and updates its table accordingly
    - E.g. Bob with MAC address was at port 10, however device received new packet from “Bob” with MAC address at port 20 from attacker, device will just forward the packet as follows, and update its table
    - Show mac-address
* Preventing IP & MAC address spoofing
  + Switches configured with DHCP snooping can be further configured with IP source guard (IPSG)
  + When IPSG is configured on untrusted ports, all traffic is blocked except DHCP
  + Any packet with different source IP will be dropped
  + It will only allow traffic after the IP address is assigned by DHCP server, or statically configured
    - Interface [interface]
    - Ip verify source
    - Show ip verify source
  + The example below is for both IP & MAC addresses
    - Interface [interface]
    - Ip verify source port-security
    - Show ip verify source
    - Option-82 must be ENABLED for the server to not drop the DHCP server reply and client will not receive its IP address
  + To implement IPSG, the switch maintains an IP source binding table
    - Automatically learnt from dhcp snooping table
    - Statically configured
    - Ip source binding [mac-address] vlan [vlan-id] [ipaddress] interface [interface]
    - Show ip source binding

**Topic 4**

* Access Control Lists (ACLs)
* ACLs are a sequential list of one or more Access Control Entry (ACE) to filter network traffic
  + Consists of action (permit/deny) and test condition (standard / extended)
  + Processed from top to bottom in order
  + By default there is a default deny all statement to drop any other traffic
  + Sequence numbers (priority) are automatically inserted by IOS (in 10s)
    - Established keyword can be used to protect internal hosts and still allow them to access the internet
  + Recommended practices when configuring ACLs
    - One ACL per direction
    - One ACL per interface
    - One ACL per protocol
  + Verifying ACLs
    - Show ip interface [interface]
    - Show ip access-lists
  + Clear ACLs
    - Clear access-list counters
  + Editing ACLs (After showing ip access-list)
    - No [sequence number]
    - [sequence number] [permit/deny] [ip] (inserting new ACE)
    - Ip access-list [name/number] [starting sequence number] [increment step] (rearranging ACEs in ACLs)

**Topic 5**

**Table

Description automatically generated**

**Ingress Filtering**

**Text

Description automatically generated**

**Smurf attack with hping3**

* Hping3 -1 - -flood -a [target-ip] [directed broadcast address]
* Disable routers from supporting direct broadcasts
* Disable hosts from replying to broadcast ping

Text

Description automatically generated

Diagram

Description automatically generatedTable

Description automatically generated**Ports & Protocols Time-based ACLs**

**Macof**

* Install dsniff on Kali Linux
* Flood switched LAN with random MAC addresses
* Values for any options left unspecified will be generated randomly.

|  |  |
| --- | --- |
| **-i** (interface) | Specify the interface to send on. |
| **-s** (src) | Specify source IP address |
| **-d** dst | Specify destination IP address |
| **-e** tha | Specify target hardware address |
| **-x** sport | Specify TCP source port |
| **-y** dport | Specify TCP destination port |
| **-n** times | Specify the number of packets to send. |