SWITCH ATTACKS AND MITIGATION

Daniel Ge
Period 678

CISCO

CCNP LAB REPORT

Purpose

This lab provides instructions for conducting three different layer 2 attacks: DHCP pool starvation attack, CAM table overflow attack, and ARP spoofing attack. The first two attacks are Denial of Service attacks which prevent new devices from using the network through manipulating packets. The third attack is a man-in-the-middle attack to intercept packets. Working on the Linux platform, we will run the attacks through the console and observe the results. Afterwards, we will show the commands used to prevent these three attacks through configuring security measures on the switches. Ultimately, this will serve as instructions both for stress testing our network and securing our network to mitigate the risk of network attacks.

Background Information

Beginning with the 1988 Morris Worm internet attack, the first major internet attack in which a self-duplicating virus propagated through the networking infrastructure of the United States to infect computers across the nation, malicious actors, pranksters, and activists have used a variety of sophisticated tools to take down networking devices or hijack it for their gain. Although these attacks have grown in scale and complexity in the past decades, organizations may arm themselves against attackers by enacting a variety of security countermeasures developed by companies like CISCO or Cloudflare.

Given the ubiquity and necessity of layer 2 networking switches throughout the networks, one of the most common types of attacks are those which exploit security weaknesses in the networking switch devices. In this area, three of the easiest attacks on layer 2 switches are DHCP Starvation attacks, CAM table overflow attacks, and ARP spoofing attacks.

The Dynamic Host Configuration Protocol (DHCP) is a 1993 protocol in which DHCP servers scattered throughout the network automatically assign IP addresses to devices. In the past, when new devices joined the network, users manually configure their IP addresses. However, in addition to being an inconvenient task, such a system would be vulnerable to duplicate IP addresses. Furthermore, it would be hard for organizations to keep track of the IP addresses used by the devices. With DHCP, however, a central server would automatically keep track of available addresses left in a pre-defined IP address pool. When new devices join the network, they would broadcast a DHCP Discover message using a temporary IP address. When a DHCP server receives the message, they respond by offering an IP address. After confirmation, the IP address is assigned to the new device.

When a DHCP server, usually a networking router, runs out of addresses, they would not be able to offer new IP addresses to new devices. As a result, new devices would not be able to access the internet. Therefore, network attackers can exploit DHCP's system by intentionally draining the router of its DHCP address pool. Using a program such as Yersinia, attackers would send packets mimicking official DHCP Discover packets from new devices and trick the DHCP

server into giving out an address to a non-existent device. With enough packets sent out, the DHCP server's address pool would instantly drain.

A similar packet spoofing denial of service attack would attack the CAM table on switches directly. To remember the devices that are connected to each port, each switch keeps a CAM table in which the MAC addresses of devices connected to each port are stored. When a new packet arrives, it will read its destination MAC address and send it through the corresponding port. However, this CAM table on the switch has a limit too, and only a limited number of addresses may be memorized at a time. As a result, attackers can once again send spoofed packets to the switch from non-existent devices with random MAC addresses to fill up the table. When new legitimate devices try to connect to the network through the switch, the switch can no longer accommodate them because its table is already full.

In addition to denial-of-service attacks, layer 2 attacks may also be used for man-in-the-middle attacks in which the attacker intercept messages sent between adjacent devices. The ARP spoofing attack is one attack which precisely tricks devise into relaying messages through the attacker. Networking devices use the ARP table to find each other on a network by matching IP addresses to MAC addresses. For Device A to send a packet to Device B's IP, it will look up the MAC address corresponding to the destination IP and tell the switch to send the packet to that MAC address. However, attackers can use spoofed packets to manipulate the table by replacing the MAC address of the true destination with its own address. As a result, packets being sent between devices will go to the attacker instead. If the attacker then programs their computer to then relay the message to the correct destination as if nothing had happened, they would be able to steal the information without detection.

Thankfully, though these attacks may have severe repercussions on networking operations, CISCO has developed countermeasures protecting the network from hostile attackers. For DHCP starvation attacks, we may use DHCP snooping to check for fraudulent packets being sent across the network and ensure that only the router is offering DHCP addresses to new device. To prevent CAM table overflow attacks, CISCO Switch port security offers the ability to restrict each switch port to only a set limit of MAC addresses learned. When attackers try to run a CAM table overflow attack with port security on, the switch shuts the port automatically once the limit is reached. To mitigate ARP spoofing, we can run Dynamic ARP Inspection on the switch to eliminate spoofed ARP messages from being sent through the network.

Lab Summary

Through the steps of the lab, we will set up a network with a switch connected to two hosts at F0/1, F0/2 respectively, and a router at F0/4. The router on the network will provide DHCP service to the two client hosts. Working on a Linux Ubuntu operating system, we will then install three applications used to run network attacks: yersinia for DHCP spoofing, dsniff for MAC overflow, and arpspoof for the ARP spoofing attack. The three applications will create fake network packets and send them out through the interface to the switch, crippling the network.

After each attack, we will enact security measures on the switch to mitigate that type of attack, then reattempt the attack to verify that the security measures work.

Lab Commands

The new commands introduced in this lab are as follows.

DHCP Attack

To create a new DHCP pool on the router, we apply,

```
ip dhcp pool POOL-A
```

Configure the address pool and default gateway given out as follows.

```
network [network address] [subnet mask]
```

```
default-router [router port ip address]
```

To reset the bindings, we use,

```
clear ip dhcp bindings *
```

To show the bindings, we use,

```
show ip dhcp binding
```

To install an application on Linux, we run `sudo apt install [application name]` on Linux console

To enable DHCP snooping on a VLAN of the switch, run,

```
ip dhcp snooping
```

```
ip dhcp snoop vlan [vlan-number]
```

To mark certain ports as trusted, run,

```
interface [trusted interface]
```

```
ip dhcp snooping trust
```

To set a rate limit on messages from a port, apply,

```
ip dhcp snooping limit rate [limit per second]
```

CAM Table Overflow Attack

After installing dsniff, to conduct a CAM table overflow attack, run,

```
sudo macof -i [interface name]
```

To show the mac address table on a switch, use,

```
show mac address-table
```

To clear the mac address table on a switch, use,

```
clear mac address-table dynamic
```

To limit one user to a port on the switch, apply,

```
switchport mode access
```

```
switchport port-security maximum 1
```

ARP Spoofing Attack

To run an arpspoof attack, run the following command through Linux console

```
sudo arpspoof -i [interface name] -t [victim IP] [Router IP]
```

To enable dynamic arp inspection on a switch, use,

```
ip arp inspection vlan [vlan number]
```

To label a port as trusted on a switch for purposes of arp inspection, use,

```
ip arp inspection trust
```

Lab Instructions (DHCP Pool Starvation Attack)

Open two computers running Linux and Windows respectively. The computer running Linux will be referred to as PC2, and the one running Windows PC1.

Connect S1 to PC1 at F0/1, PC2 at F0/2, and R3 at F0/4 to G0/0/1 respectively. We may now set up a DHCP server on R3 with PC1 and PC2 as clients.

Configure the IP Address for the g0/0/1 port on R3 as 192.168.1.1/24

```
R1(config)#int g0/0/1
R1(config-if)#ip address 192.168.1.1 255.255.255.0
```

Create a DHCP Pool with the following commands

```
R1(config)#ip dhcp pool POOL-A
*Dec 19 22:22:13.543: %LINK-3-UPDOWN: Interface GigabitEth
*Dec 19 22:22:14.543: %LINEPROTO-5-UPDOWN: Line protocol c
R1(dhcp-config)#network 192.168.1.0 255.255.255.0
R1(dhcp-config)#default-router 192.168.1.1
R1(dhcp-config)#exit
```

Verify that PC1 was able to receive an IP addresses

```
C:\Users\Daniel>ipconfig /renew

Windows IP Configuration

No operation can be performed on Ethernet 2 while it has its media disconnected.
No operation can be performed on Wi-Fi while it has its media disconnected.
No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Local Area Connection* 10 while it has its media disconnected.
No operation can be performed on Bluetooth Network Connection while it has its media disconnected.

Ethernet adapter Ethernet 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . : nhstechnet.edu

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 10:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.2
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.1
```

Disconnect PC1 from the network temporarily.

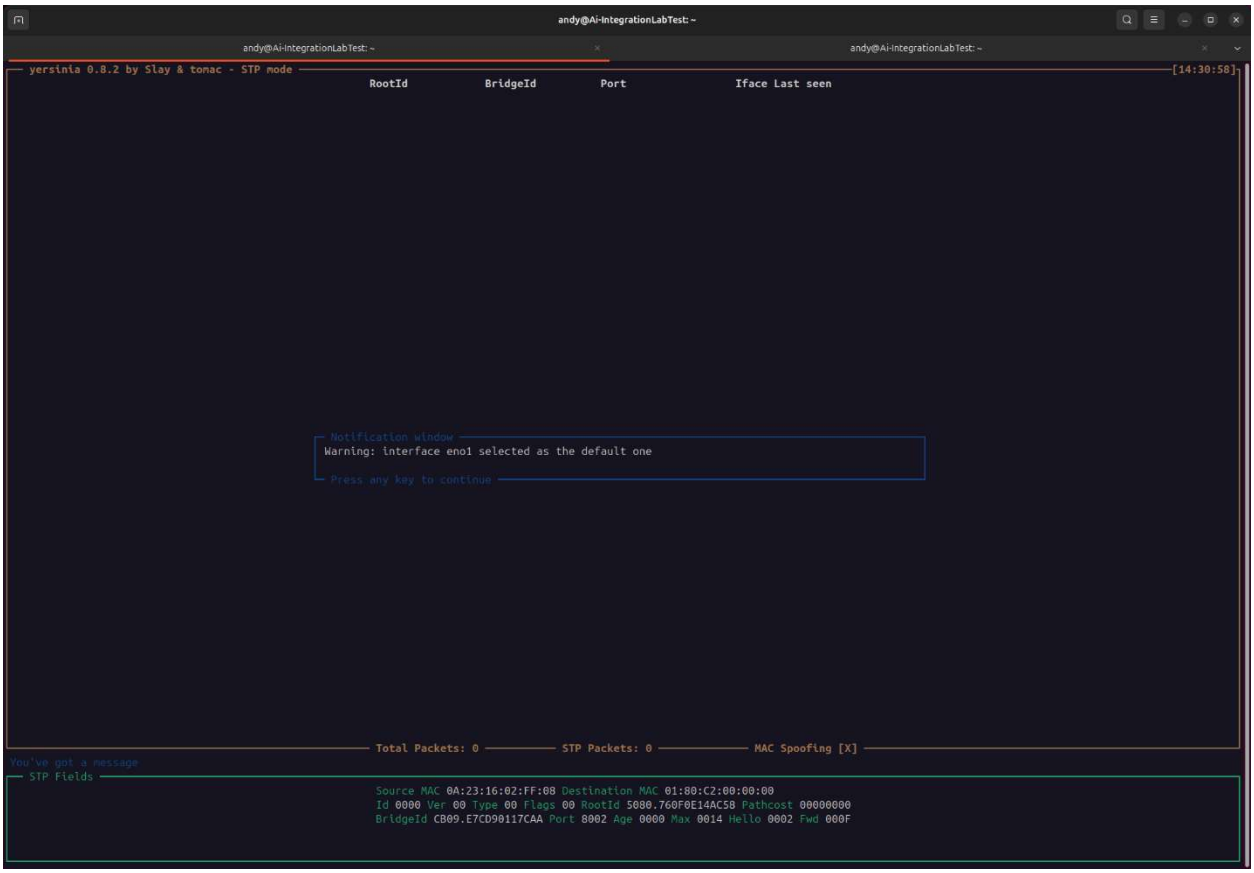
Run `clear ip dhcp binding *` to reset the bindings

```
R1#clear ip dhcp binding *
```

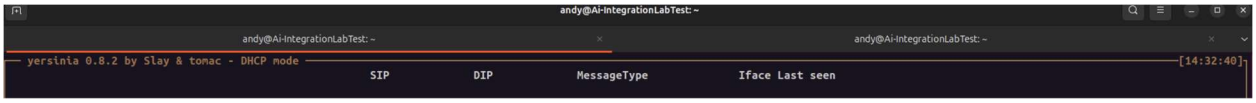
On PC2, run `sudo apt install yersinia` to install Yersinia

```
andy@Ai-IntegrationLabTest:~$ sudo apt install yersinia
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
yersinia is already the newest version (0.8.2-2.2build2).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
```

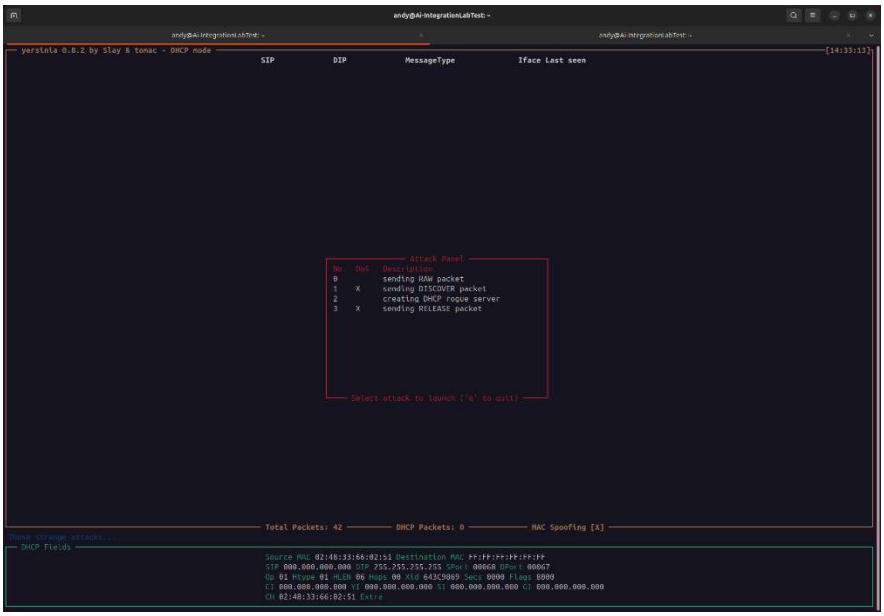
Run `sudo yersinia -I` to run yersinia in interface mode



Press **F2** to enter DHCP attack mode



Press **X** to open attack option menu, then **1** to spam DISCOVER packets



Returning to the router, run `show ip dhcp binding` to verify the DHCP pool is exhausted

```
R1#
*Dec 19 22:24:29.895: %DHCPD-4-PING_CONFLICT: DHCP address conflict:  server pinged 192.168.1.3.
R1#show ip dhcp binding
Bindings from all pools not associated with VRF:
IP address      Client-ID/              Lease expiration        Type       State      Interface
                Hardware address/
                User name
192.168.1.2     dee3.7148.d24f          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.4     8608.0b08.57fc          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.5     ace3.7d06.6d4b          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.6     a468.e261.1054          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.7     fcf4.fb50.2c0b          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.8     f208.146b.ead7          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.9     9634.2559.f987          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.10    56d0.ce07.1dcf          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.11    5c2e.a17b.68dd          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
192.168.1.12    54e6.ec70.3503          Dec 19 2025 10:29 PM    Automatic  Selecting  GigabitEthernet0/0/1
```

Now, reconnect PC1 to the network and run `ipconfig /renew` to verify it may no longer get an IP address

```
C:\Users\Daniel>ipconfig /renew

Windows IP Configuration

No operation can be performed on Ethernet 2 while it has its media disconnected.
No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Local Area Connection* 10 while it has its media disconnected.
```

Lab Instructions (DHCP Starvation Attack Mitigation)

Run `clear ip dhcp binding *` to undo the attack

```
R1#clear ip dhcp binding *
```

On the switch, enable DHCP Snooping

```
S1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)#ip dhcp snooping
S1(config)#ip dhcp snoop vlan 1
```

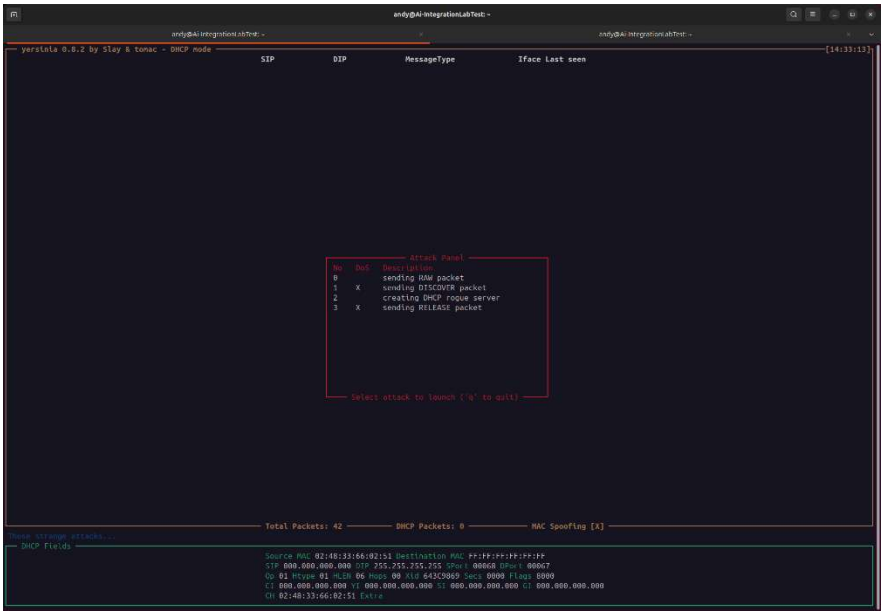
Mark the port connected to the router as a trusted port

```
S1(config)#int f0/4
S1(config-if)#ip dhcp snooping trust
S1(config-if)#exit
```

For user interfaces, rate limit DHCP messages to 100 per second

```
S1(config)#int range f0/1-2
S1(config-if-range)#ip dhcp snooping limit rate 100
```

Repeat the steps to launch the DHCP attack

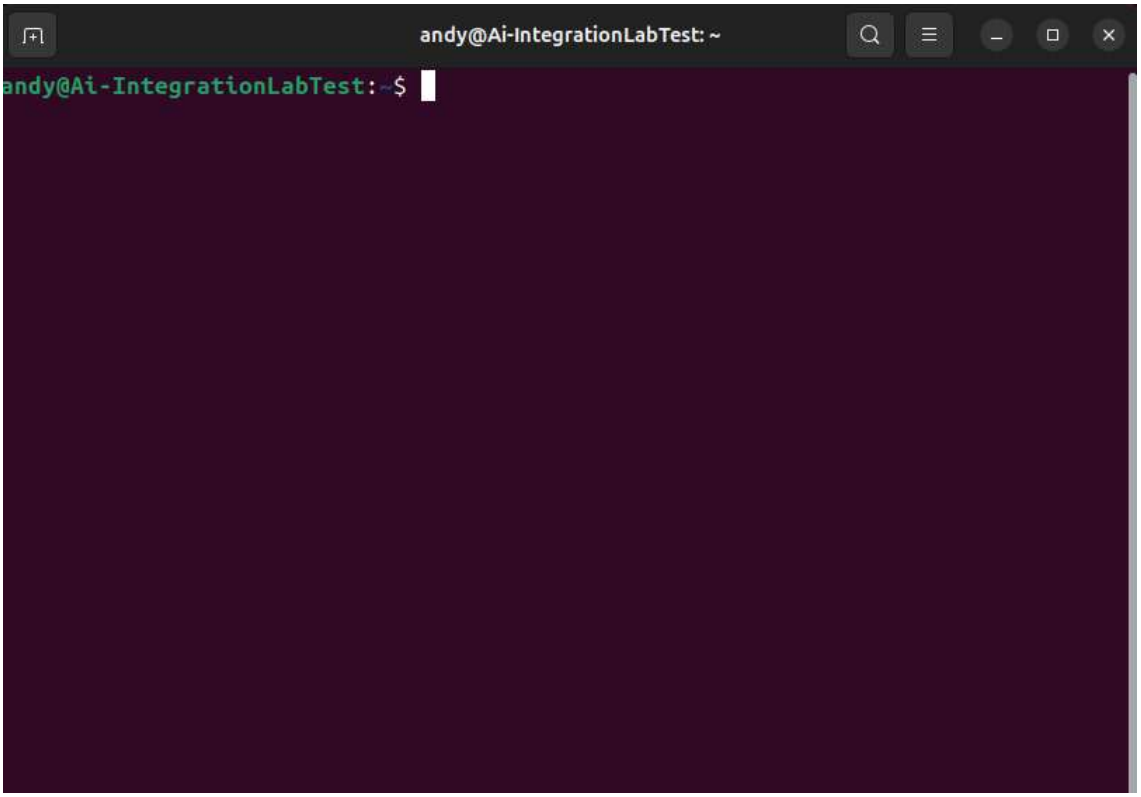


Verify that the Switch automatically shuts down the port upon violation

```
*Apr  6 08:07:39.961: %DHCP_SNOOPING-4-DHCP_SNOOPING_ERRDISABLE_WARNING: DHCP Snooping received 10 DHCP
packets on interface Fa0/2
*Apr  6 08:07:39.961: %PM-4-ERR_DISABLE: dhcp-rate-limit error detected on Fa0/2, putting Fa0/2 in err-d
isable state
*Apr  6 08:07:39.978: %DHCP_SNOOPING-4-DHCP_SNOOPING_RATE_LIMIT_EXCEEDED: The interface Fa0/2 is receivi
ng more than the threshold set
*Apr  6 08:07:41.127: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/2, changed state to
down
*Apr  6 08:07:41.982: %LINK-3-UPDOWN: Interface FastEthernet0/2, changed state to down
```

Lab Instructions (CAM Table Overflow Attack)

On PC2, open the terminal



Run `sudo apt-get install dsniff –fix-missing` to get the relevant commands for the attack

```
andy@Ai-IntegrationLabTest:~$ sudo apt-get install dsniff --fix-missing
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnet1 libnids1.21t64
The following NEW packages will be installed:
  dsniff libnet1 libnids1.21t64
0 upgraded, 3 newly installed, 0 to remove and 62 not upgraded.
Need to get 170 kB of archives.
After this operation, 656 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libnet1 amd64 1.1.6+dfsg-3.2build1 [44.5 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu noble/universe amd64 libnids1.21t64 amd64 1.26-2.1build2 [21.9 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu noble/universe amd64 dsniff amd64 2.4b1+debian-32build2 [103 kB]
Fetched 170 kB in 1s (230 kB/s)
Selecting previously unselected package libnet1:amd64.
(Reading database ... 226123 files and directories currently installed.)
Preparing to unpack .../libnet1_1.1.6+dfsg-3.2build1_amd64.deb ...
Unpacking libnet1:amd64 (1.1.6+dfsg-3.2build1) ...
Selecting previously unselected package libnids1.21t64:amd64.
Preparing to unpack .../libnids1.21t64_1.26-2.1build2_amd64.deb ...
Unpacking libnids1.21t64:amd64 (1.26-2.1build2) ...
Selecting previously unselected package dsniff.
Preparing to unpack .../dsniff_2.4b1+debian-32build2_amd64.deb ...
Unpacking dsniff (2.4b1+debian-32build2) ...
Setting up libnet1:amd64 (1.1.6+dfsg-3.2build1) ...
Setting up libnids1.21t64:amd64 (1.26-2.1build2) ...
Setting up dsniff (2.4b1+debian-32build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Processing triggers for man-db (2.12.0-4build2) ...
andy@Ai-IntegrationLabTest:~$ sudo macof -i eth0
macof: libnet_check_iface() ioctl: No such device
```

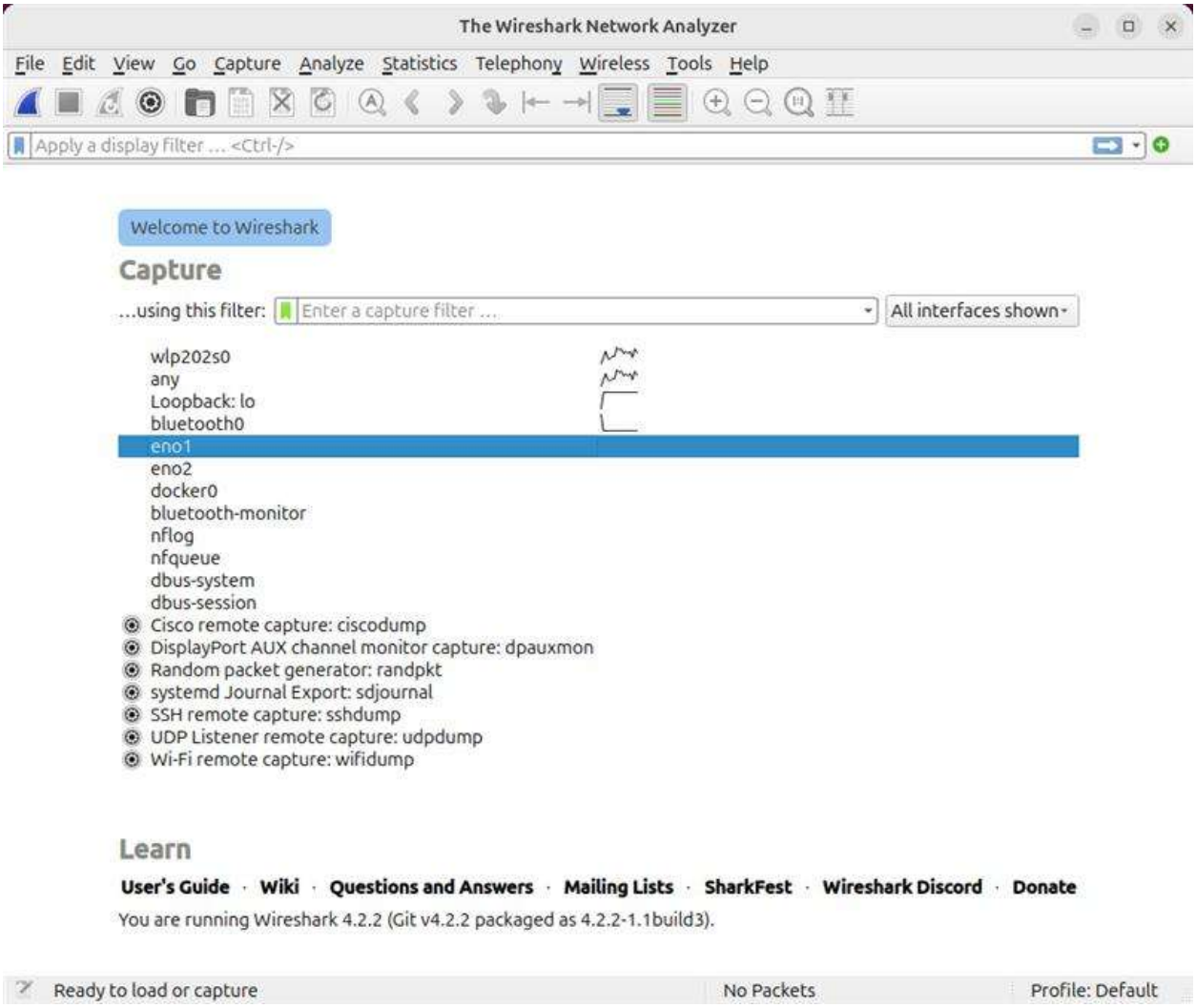
Run `sudo apt install wireshark` to download Wireshark

```
andy@Ai-IntegrationLabTest:~$ sudo apt install wireshark
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaacs0 libavformat60 libb2-1 libbcg729-0 libbdplus0 libbluray2
  libchromaprint1 libcjson1 libdouble-conversion3 libgme0 liblua5.2-0
  libmbedcrypto7t64 libmd4c0 libminizip1t64 libnghttp3-3 libnorm1t64
  libopencore-amrnb0 libopenmpt0t64 libpcre2-16-0 libpgm-5.3-0t64
  libqt6core5compat6 libqt6core6t64 libqt6dbus6t64 libqt6gui6t64
  libqt6multimedia6 libqt6network6t64 libqt6opengl6t64 libqt6printsupport6t64
  libqt6qml6 libqt6qmlmodels6 libqt6quick6 libqt6svg6 libqt6waylandclient6
  libqt6waylandcompositor6 libqt6waylandeglclienthwintegration6
  libqt6waylandeglcompositorhwintegration6 libqt6widgets6t64
  libqt6wlshellintegration6 librabbitmq4 librist4 libsmi2t64 libspandsp2t64
  libsrt1.5-gnutls libssh-gcrypt-4 libswscale7 libts0t64 libudfread0
  libwireshark-data libwireshark17t64 libwiretap14t64 libwsutil15t64 libzmq5
  qt6-gtk-platformtheme qt6-qpa-plugins qt6-translations-l10n qt6-wayland
  wireshark-common
Suggested packages:
  libbluray-bdj qt6-qmltooling-plugins snmp-mibs-downloader geoipupdate
  geoip-database geoip-database-extra libjs-leaflet
  libjs-leaflet.markercluster wireshark-doc
The following NEW packages will be installed:
```

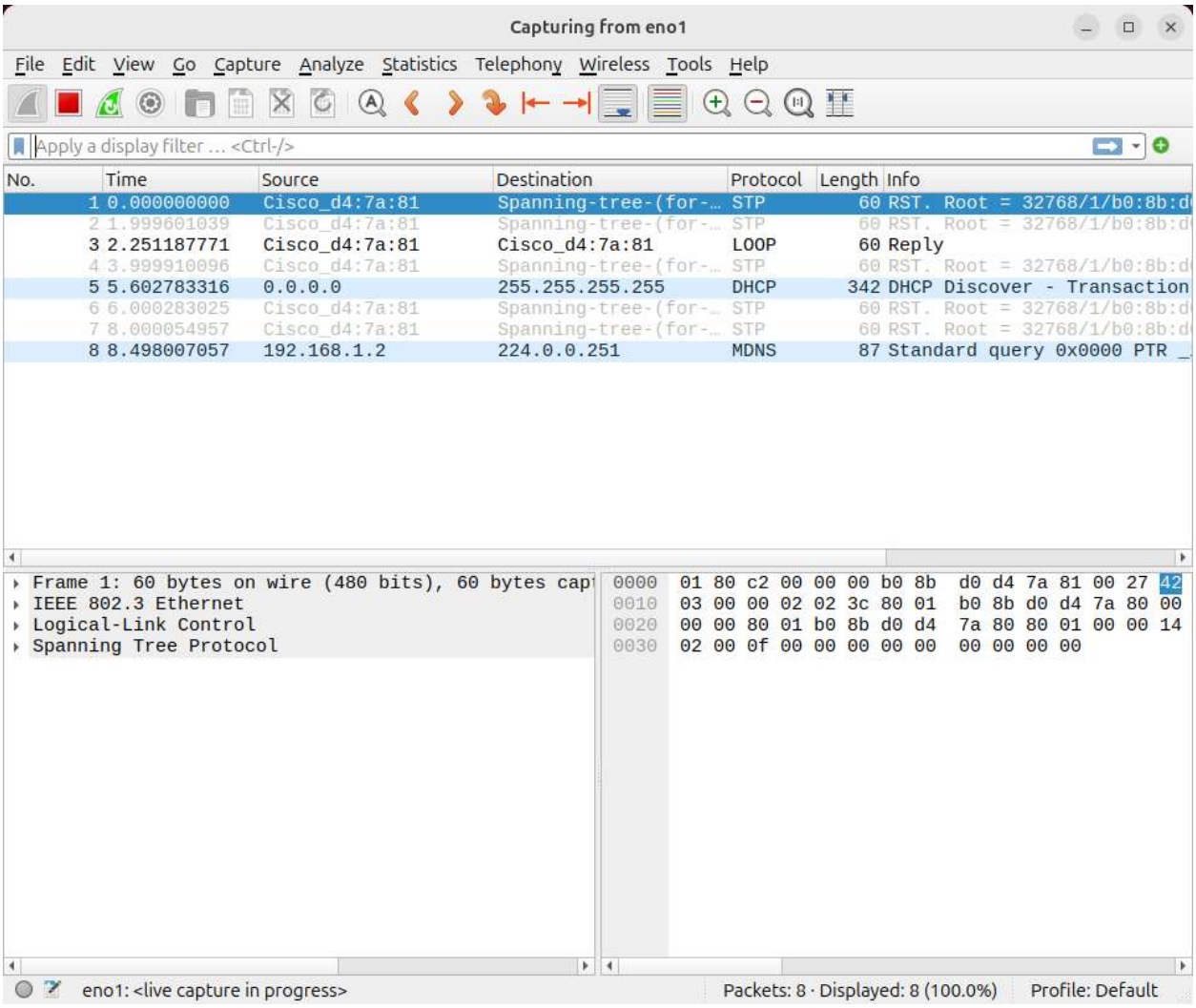
Now, on PC2, open a second Terminal window and run `sudo macof -i eno1`

```
andy@Ai-IntegrationLabTest:~$ sudo macof -i eno1
```

Run `sudo wireshark` to launch Wireshark in admin mode



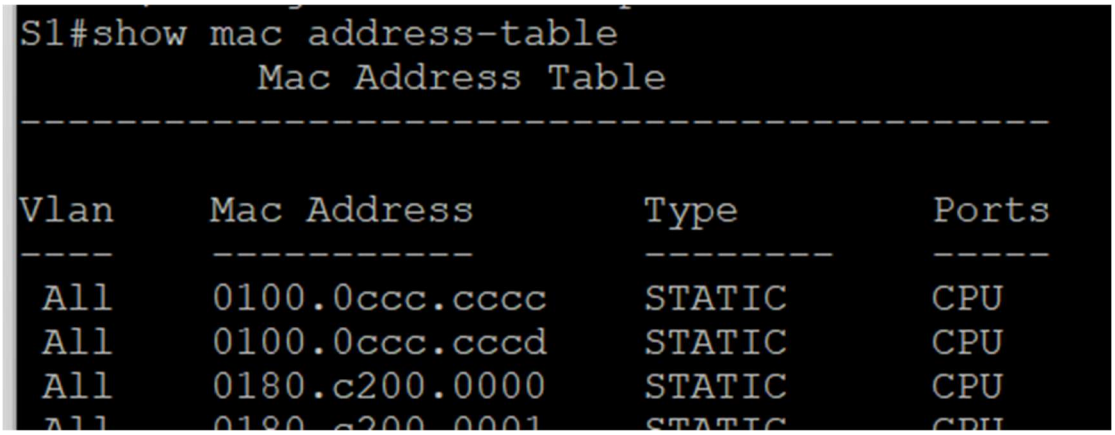
Open the option for **eno1**



Connect the PC2 and PC1 to a switch through an Ethernet connection on **port eno1**

Open Putty on PC1 and use COM1 to open the switch

Enter privileged EXEC mode through the `enable` command, then enter `show mac address-table`

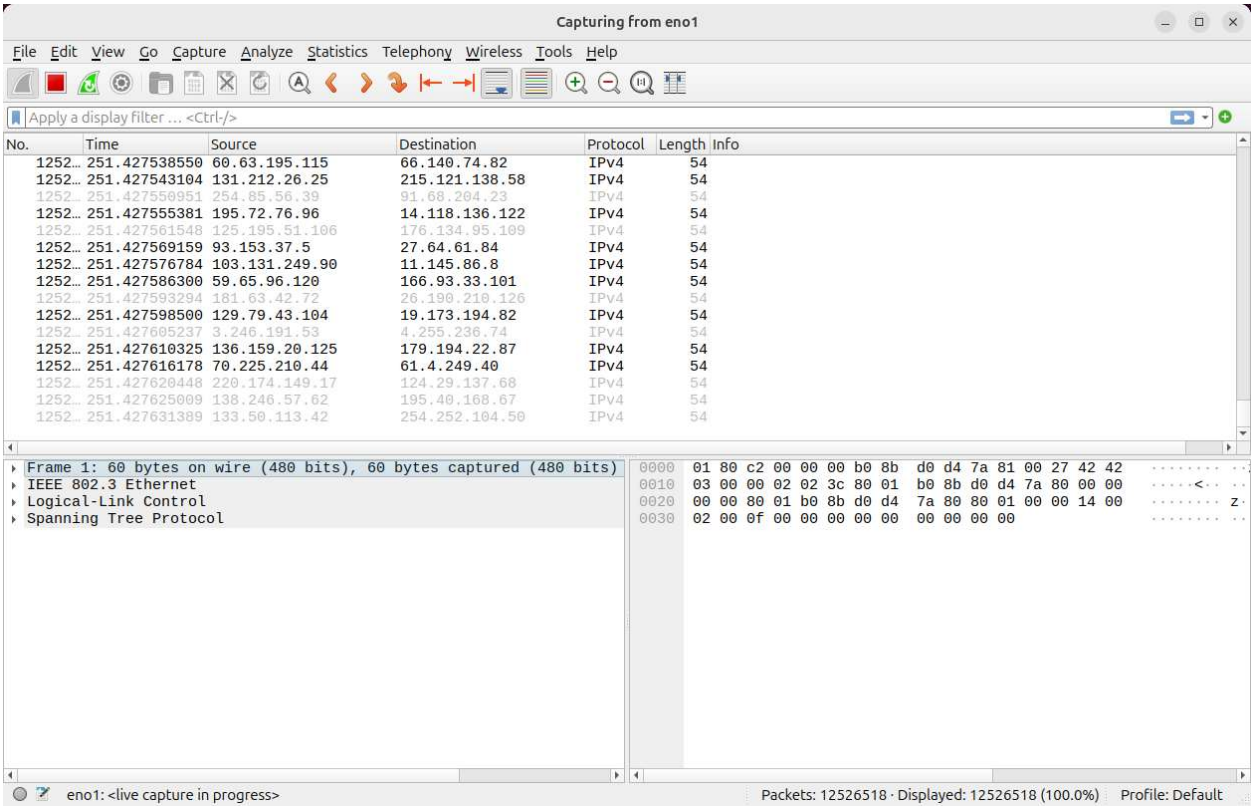


Verify that the MAC address-table has only two DYNAMIC addresses, namely the MAC addresses of the two connected PC2 and PC1

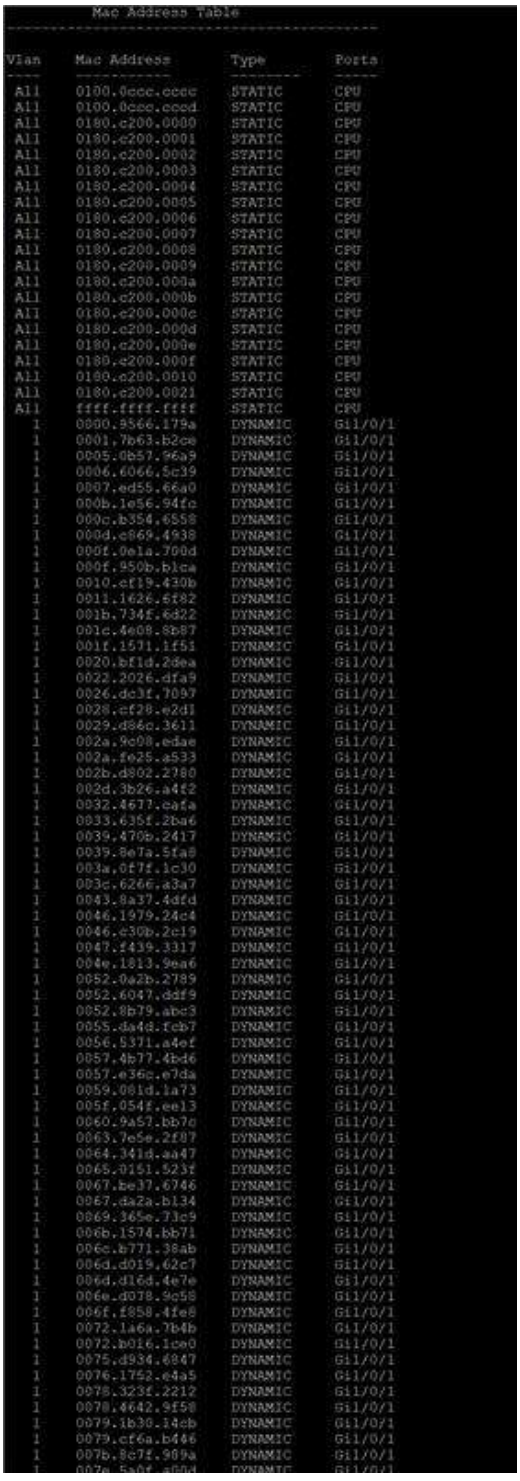
Now, on PC2, open a second Terminal window and run `sudo macof -i eno1`

```
andy@Ai-IntegrationLabTest:~$ sudo macof -i eno1
```

After roughly 60 seconds, verify the attack is successful by checking if there is indeed a flood of packets flowing through Wireshark from random IP addresses



Verify the switch is affected by the attack by running `show mac address-table` again



```
          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
 All    0100.0ccc.cccc    STATIC      CPU
 All    0100.0ccc.cccd    STATIC      CPU
 All    0180.c200.0000    STATIC      CPU
 All    0180.c200.0001    STATIC      CPU
 All    0180.c200.0002    STATIC      CPU
 All    0180.c200.0003    STATIC      CPU
 All    0180.c200.0004    STATIC      CPU
 All    0180.c200.0005    STATIC      CPU
 All    0180.c200.0006    STATIC      CPU
 All    0180.c200.0007    STATIC      CPU
 All    0180.c200.0008    STATIC      CPU
 All    0180.c200.0009    STATIC      CPU
 All    0180.c200.000a    STATIC      CPU
 All    0180.c200.000b    STATIC      CPU
 All    0180.c200.000c    STATIC      CPU
 All    0180.c200.000d    STATIC      CPU
 All    0180.c200.000e    STATIC      CPU
 All    0180.c200.000f    STATIC      CPU
 All    0180.c200.0010    STATIC      CPU
 All    0180.c200.0021    STATIC      CPU
 All    ffff.ffff.ffff    STATIC      CPU
   1    0000.9566.179a    DYNAMIC     Gi1/0/1
   1    0001.7b63.b2ce    DYNAMIC     Gi1/0/1
   1    0005.0b57.96a9    DYNAMIC     Gi1/0/1
   1    0006.6066.5c39    DYNAMIC     Gi1/0/1
   1    0007.ed55.66a0    DYNAMIC     Gi1/0/1
   1    000b.1e56.94fc    DYNAMIC     Gi1/0/1
   1    000c.b354.6558    DYNAMIC     Gi1/0/1
   1    000d.e869.4938    DYNAMIC     Gi1/0/1
   1    000f.0e1a.700d    DYNAMIC     Gi1/0/1
   1    000f.950b.b1ca    DYNAMIC     Gi1/0/1
   1    0010.ef19.430b    DYNAMIC     Gi1/0/1
   1    0011.1626.6f82    DYNAMIC     Gi1/0/1
   1    001b.734f.6d22    DYNAMIC     Gi1/0/1
   1    001c.4e08.8b87    DYNAMIC     Gi1/0/1
   1    001f.1571.1f51    DYNAMIC     Gi1/0/1
   1    0020.bf1d.2dea    DYNAMIC     Gi1/0/1
   1    0022.2026.dfa9    DYNAMIC     Gi1/0/1
   1    0026.dc3f.7097    DYNAMIC     Gi1/0/1
   1    0028.ef28.e2d1    DYNAMIC     Gi1/0/1
   1    0029.d86c.3611    DYNAMIC     Gi1/0/1
   1    002a.9c08.edae    DYNAMIC     Gi1/0/1
   1    002a.fe25.a533    DYNAMIC     Gi1/0/1
   1    002b.d802.2780    DYNAMIC     Gi1/0/1
   1    002d.3b26.a4f2    DYNAMIC     Gi1/0/1
   1    0032.4677.eafa    DYNAMIC     Gi1/0/1
   1    0033.635f.2ba6    DYNAMIC     Gi1/0/1
   1    0039.470b.2417    DYNAMIC     Gi1/0/1
   1    0039.8e7a.5fa8    DYNAMIC     Gi1/0/1
   1    003a.0f7f.1c30    DYNAMIC     Gi1/0/1
   1    003c.6266.a3a7    DYNAMIC     Gi1/0/1
   1    0043.8a37.4dfd    DYNAMIC     Gi1/0/1
   1    0046.1979.24c4    DYNAMIC     Gi1/0/1
   1    0046.e30b.2c19    DYNAMIC     Gi1/0/1
   1    0047.f439.3317    DYNAMIC     Gi1/0/1
   1    004e.1013.9ea6    DYNAMIC     Gi1/0/1
   1    0052.0a2b.2789    DYNAMIC     Gi1/0/1
   1    0052.6047.ddf9    DYNAMIC     Gi1/0/1
   1    0052.8b79.abc3    DYNAMIC     Gi1/0/1
   1    0055.da4d.feb7    DYNAMIC     Gi1/0/1
   1    0056.5371.a4ef    DYNAMIC     Gi1/0/1
   1    0057.4b77.4bd6    DYNAMIC     Gi1/0/1
   1    0057.e36c.e7da    DYNAMIC     Gi1/0/1
   1    0059.081d.1a73    DYNAMIC     Gi1/0/1
   1    005f.054f.ee13    DYNAMIC     Gi1/0/1
   1    0060.9a57.bb7c    DYNAMIC     Gi1/0/1
   1    0063.7e5e.2f87    DYNAMIC     Gi1/0/1
   1    0064.341d.aa47    DYNAMIC     Gi1/0/1
   1    0065.0151.523f    DYNAMIC     Gi1/0/1
   1    0067.be37.6746    DYNAMIC     Gi1/0/1
   1    0067.da2a.b134    DYNAMIC     Gi1/0/1
   1    0069.365e.73c9    DYNAMIC     Gi1/0/1
   1    006b.1574.bb71    DYNAMIC     Gi1/0/1
   1    006c.b771.38ab    DYNAMIC     Gi1/0/1
   1    006d.d019.62c7    DYNAMIC     Gi1/0/1
   1    006d.d16d.4e7e    DYNAMIC     Gi1/0/1
   1    006e.d078.9c58    DYNAMIC     Gi1/0/1
   1    006f.f858.4fe8    DYNAMIC     Gi1/0/1
   1    0072.1a6a.7b4b    DYNAMIC     Gi1/0/1
   1    0072.b016.1ce0    DYNAMIC     Gi1/0/1
   1    0075.d934.6847    DYNAMIC     Gi1/0/1
   1    0076.1752.e4a5    DYNAMIC     Gi1/0/1
   1    0078.323f.2212    DYNAMIC     Gi1/0/1
   1    0078.4642.9f58    DYNAMIC     Gi1/0/1
   1    0079.1b30.14eb    DYNAMIC     Gi1/0/1
   1    0079.cf6a.b446    DYNAMIC     Gi1/0/1
   1    007b.8c7f.989a    DYNAMIC     Gi1/0/1
```

After a few minutes, the switch should report that the MAC address limit has been reached, indicating a successful attack; no more devices can use the switch

```
*Dec  8 21:56:10.356: %MATM-3-MAX_ENTRIES: Switch 1 R0/0: fed: The maximum number of MAC addresses has been reached:16384
```

Lab Instructions (CAM Table Overflow Mitigation)

Start by entering `clear mac address-table dynamic` on the switch to undo the effect of the attack

```
S1#clear mac address-table dynamic
```

Over all ports of the switch, enter the commands

- `switchport mode access`
- `switchport port-security maximum 1`

```
S1(config)#int range f0/1-24
S1(config-if-range)#switchport mode access
S1(config-if-range)#switchport port-security maximum 1
```

Now, if we recreate the attack, the switch will shut down the violating port after learning more than one address from the port

```
Switch#
*Dec  8 22:05:39.281: %PM-4-ERR_DISABLE: psecure-violation error detected on Gi1/0/1, putting Gi1/0/1 in err-disable state
*Dec  8 22:05:39.284: %PORT_SECURITY-2-PSECURE_VIOLATION: Security violation occurred, caused by MAC address d0bf.2159.8640 on port GigabitEthernet1/0/1.
*Dec  8 22:05:40.282: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet1/0/1, changed state to down
*Dec  8 22:05:41.281: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/1, changed state to down
```

Lab Instructions (ARP Spoofing)

Reconnect the network

On PC2, run `sudo arpspoof -i eno1 -t [PC1 IP] [Router IP]` to remap the router MAC address to PC2

```
andy@Ai-IntegrationLabTest:~$ sudo arpspoof -i eno1 -t 192.168.1.2 192.168.1.1
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at 4:d9:c8:ba:24:cc
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at 4:d9:c8:ba:24:cc
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at 4:d9:c8:ba:24:cc
```

Run `sudo arpspoof -i eno1 -t [Router IP] [PC1 IP]` to remap the PC2 MAC address to R1

```
andy@Ai-IntegrationLabTest:~$ sudo arpspoof -i eno1 -t 192.168.1.1 192.168.1.2
4:d9:c8:ba:24:cc b4:a8:b9:1:b7:51 0806 42: arp reply 192.168.1.2 is-at 4:d9:c8:ba:24:cc
4:d9:c8:ba:24:cc b4:a8:b9:1:b7:51 0806 42: arp reply 192.168.1.2 is-at 4:d9:c8:ba:24:cc
4:d9:c8:ba:24:cc b4:a8:b9:1:b7:51 0806 42: arp reply 192.168.1.2 is-at 4:d9:c8:ba:24:cc
```

Run `arp -a` on to verify the MAC address of the router and PC2 are now the same

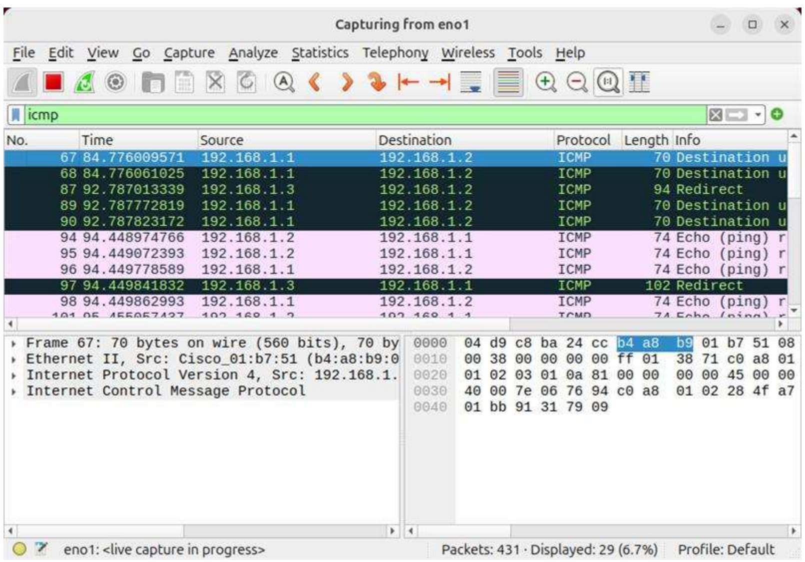
```
Interface: 192.168.1.2 --- 0x13
  Internet Address      Physical Address      Type
  192.168.1.1           04-d9-c8-ba-24-cc     dynamic
  192.168.1.3           04-d9-c8-ba-24-cc     dynamic
```

Ping the router from PC1

```
C:\Users\Daniel>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
Reply from 192.168.1.1: bytes=32 time=1ms TTL=254
```

Open Wireshark on PC2 and start capturing on eno1 once again, observe that PC2 is now able to see the message



Lab Instructions (ARP Spoofing Mitigation)

Enter Ctrl+C on PC2 terminal to undo the attack

```
^CCleaning up and re-arping targets...
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at b4:a8:b9:1:b7:51
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at b4:a8:b9:1:b7:51
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at b4:a8:b9:1:b7:51
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at b4:a8:b9:1:b7:51
4:d9:c8:ba:24:cc 4:d9:c8:ba:24:6e 0806 42: arp reply 192.168.1.1 is-at b4:a8:b9:1:b7:51
andy@Ai-IntegrationLabTest:~$
```

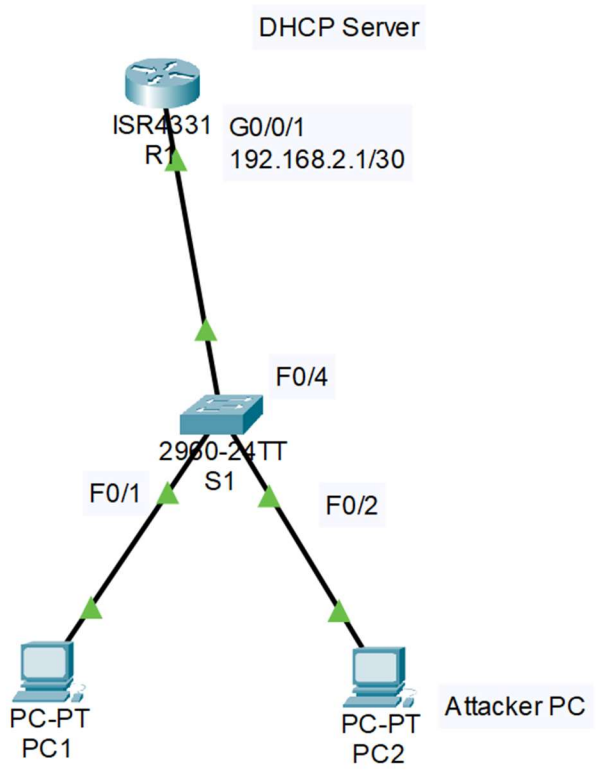
On S1, use `ip arp inspection vlan 1` and `ip arp inspection trust` on the port to R1 to enable ARP inspection

```
S1(config)#ip arp inspection vlan 1
S1(config)#int f0/4
S1(config-if)#ip arp inspection trust
```

When we reattempt the attack, the spoofed ARP messages are dropped

```
*Mar  1 01:18:08.090: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Fa0/2, vlan 1.([04d9.c8ba.24cc/192.168.1.
2/b4a8.b901.b751/192.168.1.1/01:18:07 UTC Mon Mar 1 1993])
*Mar  1 01:18:10.104: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Fa0/2, vlan 1.([04d9.c8ba.24cc/192.168.1.
2/b4a8.b901.b751/192.168.1.1/01:18:09 UTC Mon Mar 1 1993])
*Mar  1 01:18:12.117: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Fa0/2, vlan 1.([04d9.c8ba.24cc/192.168.1.
2/b4a8.b901.b751/192.168.1.1/01:18:11 UTC Mon Mar 1 1993])
*Mar  1 01:18:14.130: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Fa0/2, vlan 1.([04d9.c8ba.24cc/192.168.1.
2/b4a8.b901.b751/192.168.1.1/01:18:13 UTC Mon Mar 1 1993])
```

Network Diagram



Configurations

Below are the configurations of the two network devices after all the protections have been added.

Router R1

```
version 16.9
ser
```

```
version 16.9
service timestamps debug datetime msec
service timestamps log datetime msec
```

```
platform qfp utilization monitor load 80
platform punt-keepalive disable-kernel-core
hostname R1
boot-start-marker
boot-end-marker
vrf definition Mgmt-intf
address-family ipv4
exit-address-family
address-family ipv6
exit-address-family
no aaa new-model
ip dhcp pool POOL-A
network 192.168.1.0 255.255.255.0
default-router 192.168.1.1
login on-success log
subscriber templating
vtp domain cisco
vtp mode transparent
multilink bundle-name authenticated
crypto pki trustpoint TP-self-signed-3939337615
enrollment selfsigned
subject-name cn=IOS-Self-Signed-Certificate-3939337615
revocation-check none
rsakeypair TP-self-signed-3939337615
crypto pki certificate chain TP-self-signed-3939337615
certificate self-signed 01
30820330 30820218 A0030201 02020101 300D0609 2A864886 F70D0101 05050030
31312F30 2D060355 04031326 494F532D 53656C66 2D536967 6E65642D 43657274
69666963 6174652D 33393339 33333736 3135301E 170D3236 30313035 32323236
30395A17 0D333030 31303130 30303030 305A3031 312F302D 06035504 03132649
4F532D53 656C662D 5369676E 65642D43 65727469 66696361 74652D33 39333933
33373631 35308201 22300D06 092A8648 86F70D01 01010500 0382010F 00308201
0A028201 0100A103 56E616C4 05A3946D 186611DC 2A6DDBB3 6524BAE8 B2469B81
0693CA95 552C1EE4 0949BD59 42B5C352 4247184E 0E96DB2B 76E2A38E B8F5B0A4
C6D9BDEE 26C9C6C3 1BE69F0C BBBB9D67 40585E70 16243FCE F58D7754 C1E943BD
2F68E1A8 4D671AF3 2758D6B4 83018D24 4A1F5AA6 638C70AC 0555A582 DF9F4E5C
3A280BA9 E96D21F3 C5B289DB C4289B59 5CD170BC CDB3447A 9F7626AB 79BC2201
6B6E201B A7A75DFE 935B4595 72CDC366 53F76724 CA02BF22 B54409F3 613C3E52
95926BC8 EA2BE80C AB6D5746 E18ED301 2E3E71E6 D8E32985 9A57D1B4 EAD3C6A5
FAD2217B 0321A5C9 E8F2F46F 32411AFA 662A79AF FC84D043 27AA53A3 2963E46D
B19141DF 3ACF0203 010001A3 53305130 0F060355 1D130101 FF040530 030101FF
301F0603 551D2304 18301680 14312EC5 2A743A69 365E9BB3 765C78BE 80F6EB2B
A5301D06 03551D0E 04160414 312EC52A 743A6936 5E9BB376 5C78BE80 F6EB2BA5
300D0609 2A864886 F70D0101 05050003 82010100 09BA72CB CBB5D5BC 3DC6DD4B
16CC416B 524ECFBB 45073947 C06C7BEF 508F3786 16D19BC3 DD469BB1 7AC7CC90
47FAD366 8ED9DA77 EBE55E5A 0F82046E A076CB7C BA1AA0C5 4A144E2D 2F4C1D88
84933861 93F7F034 67E06191 EEE649DF 926329CA 159661B1 BAECB40B CE34CA6D
0568EA03 0C124326 1D816FB3 E0E44EFE 4B3B6545 C7F9748F 46EE0504 1F09E529
FC89DAF0 E157BB40 1D33EBF1 D43D9DAF CA4EC730 264F466E 642052F1 BD6451AA
754BEA07 B363130C 9A5F993E 15DD8EFC 76528E3C 2CE8A4CC 2E7CD3ED 050F738E
```

```
B4669D95 15F3730F EE8184B8 EEDEC219 EDAE2231 B2EF1A39 92AFE2E7 8D568DAE
152D7FC8 D7B305C5 0882CCA6 FBE42D9F B33F1A0A
quit
license udi pid ISR4321/K9 sn FDO214420G7
no license smart enable
diagnostic bootup level minimal
spanning-tree extend system-id
redundancy
mode none
interface GigabitEthernet0/0/0
no ip address
negotiation auto
interface GigabitEthernet0/0/1
ip address 192.168.1.1 255.255.255.0
negotiation auto
interface Serial0/1/0
no ip address
shutdown
interface Serial0/1/1
no ip address
shutdown
interface GigabitEthernet0
vrf forwarding Mgmt-intf
no ip address
shutdown
negotiation auto
ip forward-protocol nd
ip http server
ip http authentication local
ip http secure-server
ip tftp source-interface GigabitEthernet0
control-plane
line con 0
transport input none
stopbits 1
line aux 0
stopbits 1
line vty 0 4
login
end
```

Switch S1

```
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
hostname S1
```

```
boot-start-marker
boot-end-marker
no aaa new-model
system mtu routing 1500
authentication mac-move permit
ip subnet-zero
ip dhcp snooping vlan 1
ip dhcp snooping
ip arp inspection vlan 1
spanning-tree mode pvst
spanning-tree etherchannel guard misconfig
spanning-tree extend system-id
vlan internal allocation policy ascending
interface FastEthernet0/1
switchport mode access
switchport port-security maximum 10
switchport port-security
spanning-tree portfast
ip dhcp snooping limit rate 100
interface FastEthernet0/2
switchport mode access
switchport port-security maximum 10
switchport port-security
spanning-tree portfast
ip dhcp snooping limit rate 100
interface FastEthernet0/3
switchport mode access
switchport port-security maximum 10
switchport port-security
spanning-tree portfast
ip dhcp snooping limit rate 100
interface FastEthernet0/4
switchport mode access
switchport port-security maximum 10
switchport port-security
ip arp inspection trust
spanning-tree portfast
ip dhcp snooping trust
interface FastEthernet0/5
interface FastEthernet0/6
interface FastEthernet0/7
interface FastEthernet0/8
interface FastEthernet0/9
interface FastEthernet0/10
interface FastEthernet0/11
interface FastEthernet0/12
interface FastEthernet0/13
interface FastEthernet0/14
interface FastEthernet0/15
interface FastEthernet0/16
```

```
interface FastEthernet0/17
interface FastEthernet0/18
interface FastEthernet0/19
interface FastEthernet0/20
interface FastEthernet0/21
interface FastEthernet0/22
interface FastEthernet0/23
interface FastEthernet0/24
interface GigabitEthernet0/1
interface GigabitEthernet0/2
interface Vlan1
ip address 192.168.1.5 255.255.255.0
ip default-gateway 192.168.1.10
ip classless
ip http server
ip http secure-server
ip sla enable reaction-alerts
line con 0
line vty 5 15
end
```

Problems

During the DHCP starvation attack mitigation, after enabling DHCP snooping, no DHCP addresses were given out at all. As it turns out, I had set the rate limit for DHCP messages to 10 messages per second.

```
S1(config)#int range f0/1,f0/2
S1(config-if-range)#ip dhcp snooping limit rate 10
S1(config-if-range)#exit
```

However, this limit was too tight, since normal DHCP operations caused the limit to trigger. After increasing the limit to 100, we found a better balance where DHCP could operate normally.

While presenting the DHCP starvation attack, I once again ran into an issue where PC1 could not get a DHCP address through a normal request. After checking the DHCP settings, I discovered that I had put the wrong port as the trusted port, meaning DHCPoffer messages from the router were getting dropped. The configuration was fixed and things worked.

When downloading programs such as Wireshark through Linux for the first time, the system initially showed that it was unable to fetch the archives. After investigating the issue, we found that the cause was because the computer was connected to the switch via ethernet but not to the internet via Wi-Fi. As a result, it could not download the program files from the internet. After reconnecting, the installations worked.

Opening Wireshark through the search bar, none of the ports showed up. After Googling this issue, I realized that I needed to run Wireshark with admin mode by opening it through the terminal.

Conclusion

Although the tools used in the lab today may be used for malicious Denial of Service attack purposes, if we familiarize ourselves with these tools and use them responsibly, we may apply them for ethical purposes such as penetration testing on our own network. Though DHCP Starvation Attacks, CAM Table Overflow, and ARP Spoofing attacks are rudimentary methods of launching Denial-of-Service attacks, they still pose a major risk to organizations with weak network security, such as small businesses. Therefore, learning to mitigate these risks through the tools used in the lab is a critical skill. Furthermore, through the experience of encountering problems when working with a new operating system, we grow our capacity to handle unique challenges and the use of internet tools to resolve our problems.

Signoff Page