Ασφάλεια Δικτύων 2025

Κωνσταντίνος Πάνου 3140298

Report: ARP Poisoning Laboratory Exercise

Introduction

This report documents the steps, methodologies, and results from the ARP Poisoning laboratory exercise. The main goal was to perform a Man-in-the-Middle attack using ARP cache poisoning in a virtualized environment and confirm its success with appropriate tools.

Objectives

- To perform ARP cache poisoning to intercept communication between two machines.
- To validate the success of the MitM attack by analyzing network traffic and ARP table modifications.

Prerequisites

- Virtualization Software: VirtualBox
- Operating Systems:
  - ☐ Attacker: Kali Linux
  - ☐ Victim: Ubuntu

Tools:

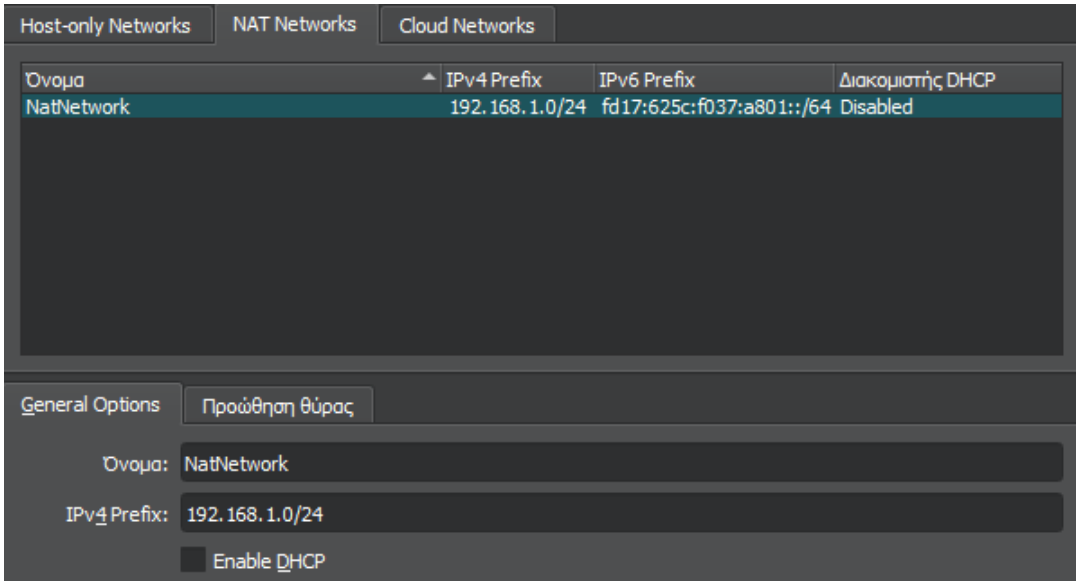
- arpspoof
- urlsnarf
- Wireshark

Methodology

Step 1: Virtual Network Setup

A custom NAT network was created with VirtualBox:

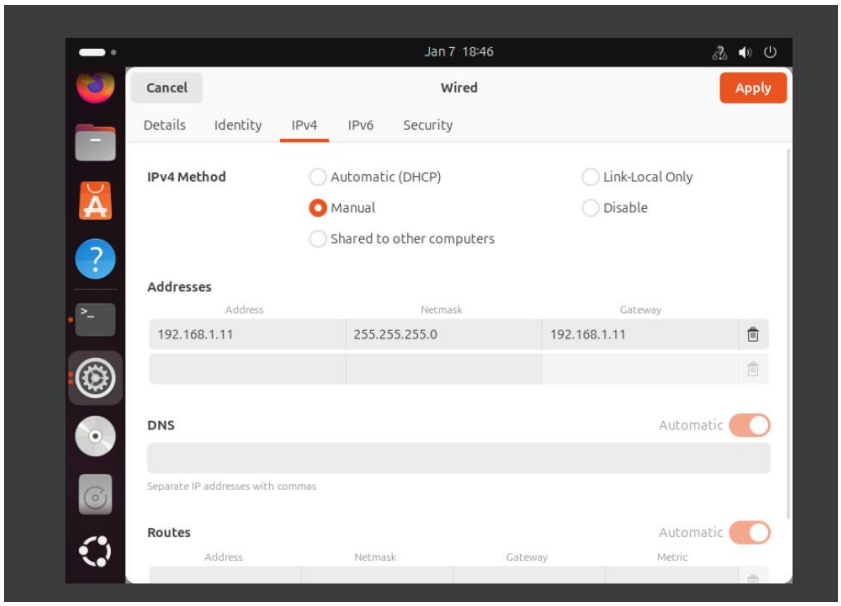
- IP range: 192.168.1.0/24
- DHCP server: Disabled
- Each VM adapter was set to the NAT network.



Step 2: IP and MAC Address Configuration

Static IPs were assigned as follows:

- **Attacker (Kali Linux):** 192.168.1.10
- **Victim (Ubuntu):** 192.168.1.11



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::95a7:88e:dbac:1701  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:6e:13:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120  bytes 37935 (37.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$
```

Step 3: Validating Connectivity

Both machines successfully communicated using ping commands to confirm connectivity.

```
kapsoules@kapsoules-VirtualBox:~$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.857 ms
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=0.746 ms
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=1.15 ms
64 bytes from 192.168.1.10: icmp_seq=4 ttl=64 time=0.960 ms
64 bytes from 192.168.1.10: icmp_seq=5 ttl=64 time=0.976 ms
64 bytes from 192.168.1.10: icmp_seq=6 ttl=64 time=1.15 ms
64 bytes from 192.168.1.10: icmp_seq=7 ttl=64 time=0.507 ms
64 bytes from 192.168.1.10: icmp_seq=8 ttl=64 time=0.531 ms
64 bytes from 192.168.1.10: icmp_seq=9 ttl=64 time=0.875 ms
64 bytes from 192.168.1.10: icmp_seq=10 ttl=64 time=0.354 ms
64 bytes from 192.168.1.10: icmp_seq=11 ttl=64 time=1.28 ms
64 bytes from 192.168.1.10: icmp_seq=12 ttl=64 time=0.545 ms
64 bytes from 192.168.1.10: icmp_seq=13 ttl=64 time=0.852 ms
64 bytes from 192.168.1.10: icmp_seq=14 ttl=64 time=0.441 ms
64 bytes from 192.168.1.10: icmp_seq=15 ttl=64 time=1.03 ms
```

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.10  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::95a7:88e:dbac:1701  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:6e:13:6e  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 120  bytes 37935 (37.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$ ping 192.168.1.11

PING 192.168.1.11 (192.168.1.11) 56(84) bytes of data.
64 bytes from 192.168.1.11: icmp_seq=1 ttl=64 time=0.581 ms
64 bytes from 192.168.1.11: icmp_seq=2 ttl=64 time=0.601 ms
64 bytes from 192.168.1.11: icmp_seq=3 ttl=64 time=0.571 ms
64 bytes from 192.168.1.11: icmp_seq=4 ttl=64 time=1.04 ms
64 bytes from 192.168.1.11: icmp_seq=5 ttl=64 time=1.29 ms
64 bytes from 192.168.1.11: icmp_seq=6 ttl=64 time=0.794 ms
64 bytes from 192.168.1.11: icmp_seq=7 ttl=64 time=1.01 ms
64 bytes from 192.168.1.11: icmp_seq=8 ttl=64 time=0.631 ms
64 bytes from 192.168.1.11: icmp_seq=9 ttl=64 time=1.12 ms
64 bytes from 192.168.1.11: icmp_seq=10 ttl=64 time=1.20 ms
64 bytes from 192.168.1.11: icmp_seq=11 ttl=64 time=0.506 ms
64 bytes from 192.168.1.11: icmp_seq=12 ttl=64 time=1.05 ms
64 bytes from 192.168.1.11: icmp_seq=13 ttl=64 time=1.07 ms
64 bytes from 192.168.1.11: icmp_seq=14 ttl=64 time=0.919 ms
64 bytes from 192.168.1.11: icmp_seq=15 ttl=64 time=1.17 ms
64 bytes from 192.168.1.11: icmp_seq=16 ttl=64 time=0.717 ms
^[[F64 bytes from 192.168.1.11: icmp_seq=17 ttl=64 time=1.07 ms
64 bytes from 192.168.1.11: icmp_seq=18 ttl=64 time=0.743 ms
64 bytes from 192.168.1.11: icmp_seq=19 ttl=64 time=1.12 ms
64 bytes from 192.168.1.11: icmp_seq=20 ttl=64 time=1.22 ms
64 bytes from 192.168.1.11: icmp_seq=21 ttl=64 time=1.09 ms
```

Step 4: Enabling IP Forwarding on Attacker Machine

IP forwarding was enabled on the attacker using the following command:

sudo sysctl -w net.ipv4.ip_forward=1

```
┌──(kali㉿kali)-[~]
└─$ sudo sysctl -w net.ipv4.ip_forward=1
[sudo] password for kali:
net.ipv4.ip_forward = 1

┌──(kali㉿kali)-[~]
└─$
```

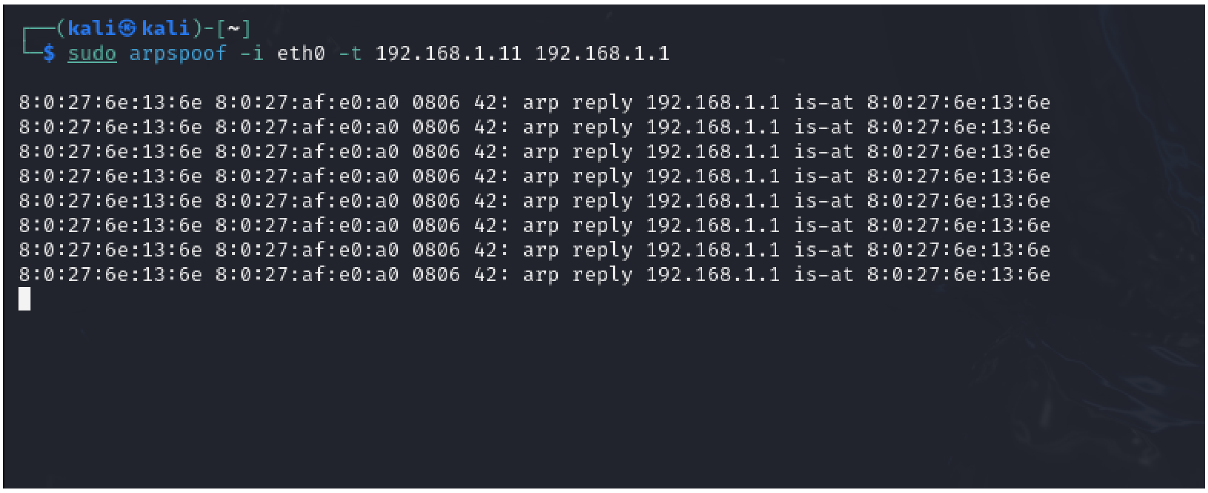
Step 5: Executing ARP Poisoning

The following command was executed to spoof ARP entries:

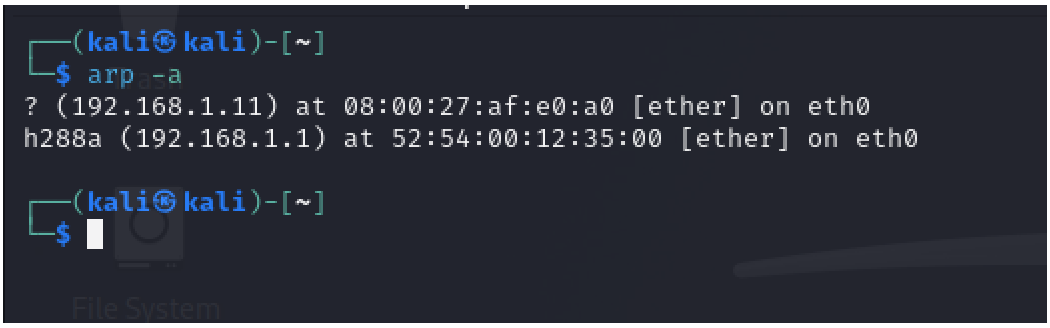
arpspoof -i eth0 -t 192.168.1.11 192.168.1.1

This redirected traffic between the victim and the router through the attacker**:**

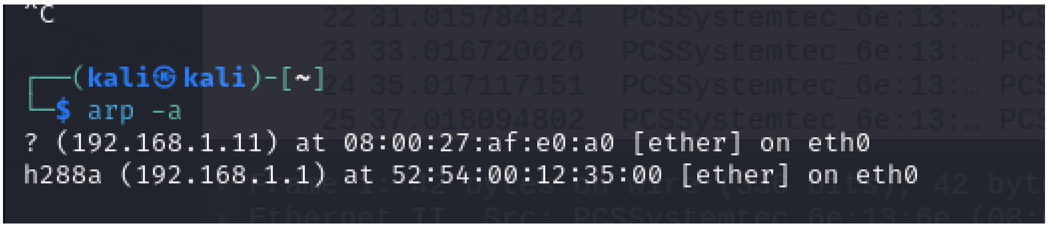
```
┌──(kali㉿kali)-[~]
└─$ arpspoof -i eth0 -t 192.168.1.11 192.168.1.1

arpspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW required
```



ARP tables before poisoning.



ARP tables after poisoning.



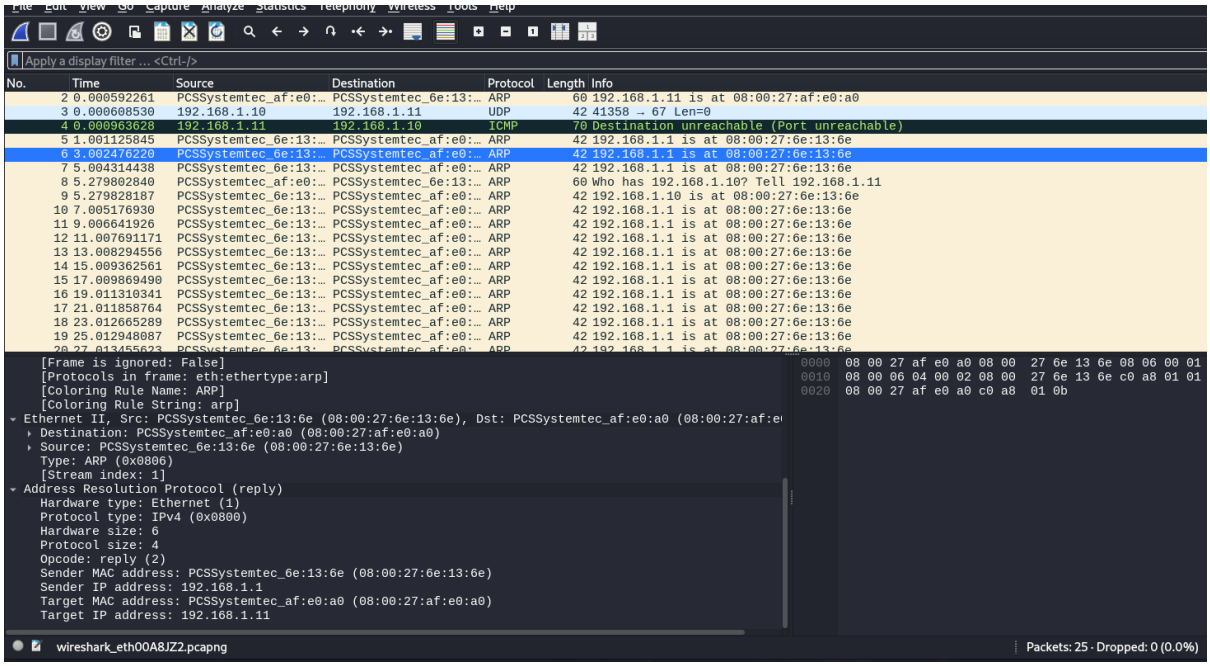
Flushing ARP cache and observing changes.

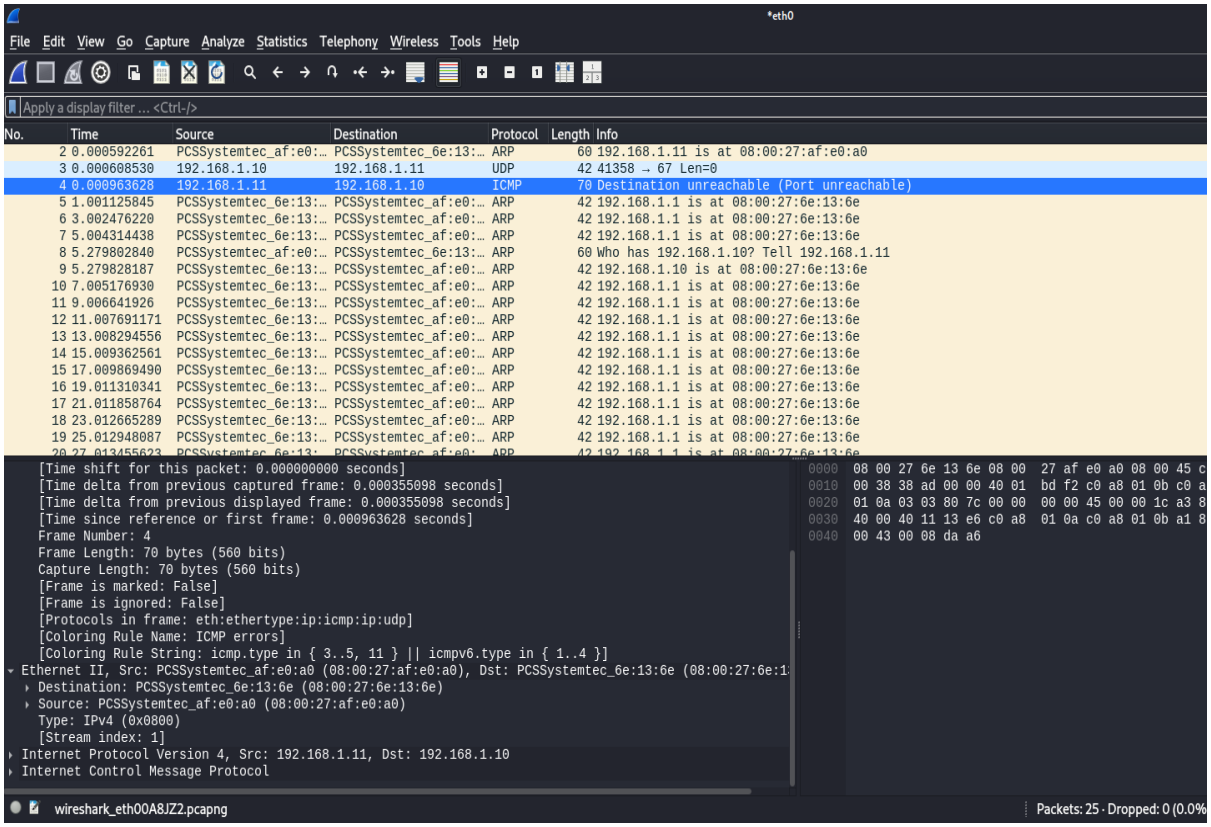
```
┌──(kali㉿kali)-[~]
└─$ sudo ip -s -s neigh flush all

[sudo] password for kali:
192.168.1.11 dev eth0 lladdr 08:00:27:af:e0:a0  used 692/692/646probes
192.168.1.1 dev eth0 lladdr 52:54:00:12:35:00  used 657/657/611probes

*** Round 1, deleting 2 entries ***
*** Flush is complete after 1 round ***

┌──(kali㉿kali)-[~]
└─$
```

Step 6: Validation of Attack

The attack was validated using:

1. **Wireshark**: Capturing packets to verify traffic interception.
2. **urlsnarf**: Confirming HTTP requests from the victim.





Results

1. The ARP tables on the victim machine showed entries modified by the attacker's MAC address.
2. Network traffic was successfully routed through the attacker's machine, validating the MitM attack.
3. Tools like urlsnarf confirmed the interception of HTTP traffic.

Conclusion

This exercise was a practical realization of the vulnerabilities in the ARP protocol and some of the risks imposed by ARP cache poisoning in networking. We can easily intercept the communication and manipulate it with ease between devices with a simulated Man-in-the-Middle attack. The results emphasize the use of best network security practices involving secure communication protocols, enabling ARP spoofing detection tools, and network segmentation. This study emphasizes the duality of network hacking for offense and defense aspects in cybersecurity as a means to provide learned insights into taking practical measures required to harden real-world systems against such attacks.

References

- Kali Linux Documentation
- Wireshark User Guide
- Official VirtualBox Documentation