ARP Spoofing & DNS MitM with Scapy — Lab Report

Name: Léonie MERY — **Course:** Ethical Hacking

Notes: Example captures, commands, scripts and evidence are provided in the associated GitHub repository.

1 Introduction

This report documents the setup of an isolated virtual lab (Attacker / Victim / Gateway), the Python/Scapy tools developed to perform a Man-in-the-Middle via ARP spoofing, the capture and analysis of network traffic (pcap), and the implementation of selective DNS spoofing. Ethical considerations and mitigation strategies are discussed in the conclusion.

2 Lab environment

2.1 Virtualization and isolation

The lab was built using VirtualBox. All virtual machines were attached to an **internal** VirtualBox network to ensure the experiment remained isolated from the host and any campus/home network.

2.2 Topology and virtual machines

The topology consisted of three VMs on the internal network with static IP addresses:

- **Attacker (Kali)** — IP: `10.0.0.3`
  Network interface: `eth0`. Resources allocated: **1 vCPU**, **2 GB RAM**. IP forwarding was enabled on the attacker only during the active tests using `sysctl`.
- **Gateway (Ubuntu Server 24)** — IP: `10.0.0.1`
  Network interface: `enp0s3` (as detected on the VM). Resources allocated: **1 vCPU**, **1 GB RAM**.
- **Victim (Ubuntu 22)** — IP: `10.0.0.2`
  Network interface: `enp0s3`. Resources allocated: **1 vCPU**, **1 GB RAM**. Browser used for interactive tests: **Firefox**. Additional tests were performed with `curl`.

2.3 IP assignment and network configuration

All IP addresses were assigned statically (manual configuration inside each VM). IP forwarding on the attacker was toggled as required for the experiments using:

```
# enable forwarding (during tests)
sudo sysctl -w net.ipv4.ip_forward=1
# disable forwarding (after tests)
sudo sysctl -w net.ipv4.ip_forward=0
```

3 Task 1 — ARP spoofing tool

3.1 Overview

An ARP spoofing tool ("`arp_spoof.py`") was used to poison the ARP caches of the victim and the gateway to establish a transparent Man-in-the-Middle position. The command used to launch the attack is shown below.

3.2 Command used

The ARP spoofing tool was started on the attacker VM with the following command:

```
sudo python3 arp_spoof.py -t 10.0.0.2 -g 10.0.0.1 -f 2
```

Explanation of the flags used (as observed from the invocation):

- `-t 10.0.0.2` : target / victim IP address
- `-g 10.0.0.1` : gateway IP address
- `-f 2` : interval / frequency parameter (the script was run with a 2 second interval between poisoning packets)

3.3 IP forwarding

IP forwarding on the attacker was **not** enabled automatically by the script; it was controlled manually outside the script when required. For the experiments the attacker's IP forwarding was enabled only while the attack was active using:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

After finishing the tests forwarding was disabled with:

```
sudo sysctl -w net.ipv4.ip_forward=0
```

3.4 Collecting evidence

To prove that traffic was routed through the attacker and to document ARP changes, the following evidence was collected:

ARP tables (before / after) Take screenshots of the ARP/cache table on the victim and on the gateway prior to starting the attack and while the attack is active. Useful commands:

```
# On Linux (victim and gateway)
ip neigh show
# or
arp -n
```

Save screenshots as, for example:

- `evidence/victim_arp_before.png`
- `evidence/victim_arp_during.png`
- `evidence/victim_arp_after.png`

PCAP of the attacker While the ARP spoofing script was running, a tcpdump capture was started on the attacker to record the network traffic and the ARP activity. The exact command used in the experiment was:

```
sudo tcpdump -i eth0 -w ~/pcaps/attacker_capture.pcap
```

This pcap file contains ARP requests/replies and the redirected traffic proving the MitM position. Reference this file in the deliverables as: `pcaps/attacker_capture.pcap`.

3.5 End of the attack

Stop the ARP spoofing script by sending a keyboard interrupt (press `Ctrl+C` in the terminal where the script is running). If a tcpdump capture was started to record traffic, stop it as well (press `Ctrl+C` in the tcpdump terminal) so the pcap file is closed and saved.

Finally, disable IP forwarding on the attacker with:

```
sudo sysctl -w net.ipv4.ip_forward=0
```

4 Task 2 — Traffic capture & analysis

4.1 Goal

Task 2 aims to capture network traffic during the experiments and to extract evidence: visited HTTP URLs, DNS queries, top talkers and protocol counts. These outputs support the claim that the attacker was in a Man-in-the-Middle position.

4.2 PCAP capture

PCAPs were recorded while the ARP spoofing attack was active. Example commands used in the lab:

```
# Attacker (primary capture)
sudo tcpdump -i eth0 -w pcaps/attacker_capture.pcap

# Victim
sudo tcpdump -i enp0s3 -w pcaps/victim_capture.pcap
```

4.3 Parser and outputs

A parser script `traffic_interceptor.py` was run against the attacker pcap. The script writes its outputs into the `evidence/` folder. Command used:

```
python3 traffic_interceptor.py pcaps/attacker_capture.pcap
```

Files produced (in `evidence/`):

- `evidence/urls.csv` - extracted HTTP requests (columns: timestamp, src_ip, dst_ip, method, host, path)
- `evidence/dns_queries.csv` - extracted DNS queries (columns: timestamp, src_ip, qname, qtype)
- `evidence/top_talkers.csv` - top talkers by packet count and bytes (ip, packets, bytes)
- `evidence/protocol_counts.csv` - counts per protocol (DNS, HTTP, TLS, ARP, etc.)

4.4 Annotated Wireshark screenshots

Two annotated Wireshark screenshots were produced as evidence:

1. **DNS screenshot** (`evidence/wireshark_dns.png`) — shows a DNS query and reply; the answer field is highlighted to demonstrate the resolved IP.
2. **HTTP screenshot** (`evidence/wireshark_http.png`) — shows an intercepted HTTP GET from the victim captured on the attacker; the Host header and source/destination are visible to prove interception.

5 Task 3 — Selective DNS spoofing

5.1 Objective

The objective of this task is to perform selective DNS spoofing: the attacker replies to DNS queries from the victim for configured domains with attacker-controlled IP addresses so the victim is redirected to an attacker-hosted page.

5.2 Summary of actions performed

The selective DNS spoofing experiment was executed using the following simple workflow:

1. Start an attacker web server (serves the fake page):
```
sudo python3 -m http.server 80
```
2. Start ARP spoofing to become MitM (left running in a terminal):
```
sudo python3 arp_spoof.py -t 10.0.0.2 -g 10.0.0.1 -f 2
```
3. Start packet capture (tcpdump) to record evidence (left running during the test).
```
sudo tcpdump -i eth0 -w ~/mitm_lab/pcaps/dns_mitm_attacker.pcap
```
4. Run the DNS spoofer with the hosts mapping file:
```
sudo python3 dns_spoof.py -i eth0 -t 10.0.0.2 -c hosts.json -r 6 -d 0.01
```
5. On the victim, open the target domain in the browser (example):
```
# In victim browser:
http://example.com
```
The victim loaded the attacker-provided page, confirming successful redirection.

5.3 Observed result

When the victim navigated to `http://example.com`, the browser displayed the page served by the attacker web server, confirming that DNS resolution for the targeted domain was successfully redirected to the attacker IP.

6 Results and analysis

The captures and parser outputs show DNS queries and HTTP requests consistent with a successful Man-in-the-Middle attack. DNS requests for the targeted domain(s) were observed and the victim loaded the attacker's page (confirmed by the browser screenshot and webserver log). PCAPs and extracted CSV files are provided in the repository as evidence.

7 Mitigation

To prevent ARP and DNS spoofing attacks, several countermeasures can be applied:

- Use static ARP entries or enable dynamic ARP inspection on switches.
- Prefer HTTPS and DNSSEC to prevent forged DNS responses.
- Segment the network and limit broadcast domains to reduce ARP spoofing risks.

These protections make it harder for an attacker to become a Man-in-the-Middle or to inject fake DNS replies.

8 Ethics and safety

All experiments were executed within an isolated virtual network under controlled conditions. Performing these techniques on unauthorized networks is illegal and unethical. The purpose of this lab is educational and defensive.

9 Deliverables

The GitHub repository contains:

```
project/
|-- arp_spoof.py
|-- dns_spoof.py
|-- traffic_parser.py
|-- hosts.json
|-- requirements.txt
|-- README.md
|-- pcap_files/
|   |-- attacker_capture.pcap
|   ‘-- dns_attack.pcap
|-- evidence/
|   |-- attacker_iface.png
|   |-- attacker_ip.png
|   |-- dns_queries.csv
|   ‘-- ...
‘-- report.pdf    (this document)
```