ARP Spoofing & DNS MitM with Scapy

AUCHET ROMAIN

SCENARIO (SHORT) :

IN AN ISOLATED LAB NETWORK, BUILD SCAPY-BASED TOOLS TO PERFORM ARP SPOOFING TO BECOME A MAN-IN-THE-MIDDLE AND IMPLEMENT SELECTIVE DNS SPOOFING TO REDIRECT VICTIMS TO ATTACKER-CONTROLLED PAGES. CAPTURE EVIDENCE, ANALYSE INTERCEPTED TRAFFIC, AND PROPOSE MITIGATIONS.

SETUP SUMMARY

To start the lab we first need to setup 3 virtual machines :

- One Gateway running simple web server—Apache/NGINX—and a local DNS
- One Victim Vm running on Unbuntu
- One Attacker Vm running on Kali

I used Oracle Virtualbox to create this 3 VM.

First we set up the network for the VMs to NAT, we install all the necessary packages :

For the Victim : curl

For the Attacker : all the scripts we published in github

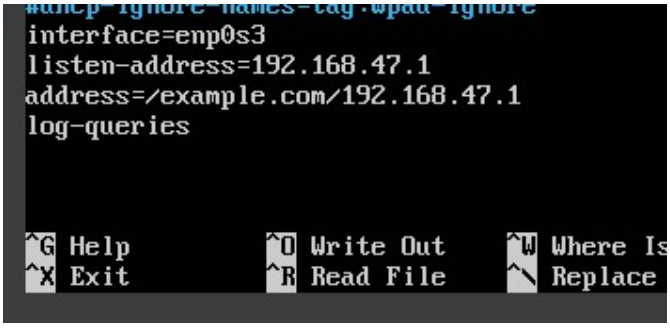
For the Gateway : nginx, apache2, dnsmasq

When it's done we need to isolate the lab : we put the Vms network to an internal network for example 'lab', to prevent Unauthorized attacks in real network.

The next step is to setup the gateway :

« sudo systemctl start nginx »

« sudo nano /etc/dnsmasq.conf »



We need for each VM to apply a static IP, for this lab I decided to go for:

- 192.168.47.1 for the gateway;
- 192.168.47.10 for the attacker
- 192.168.47.20 for the victim

```
  GNU nano 8.4
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.47.10
        netmask 255.255.255.0
        gateway 192.168.47.1
        dns-nameservers 192.168.47.1 8.8.8.8
```

Now that each VM can communicate between them and a nginx server is running with the dns resolution 'example.com' we can start the tasks for this lab.

METHODOLOGY

TASK 1

For the first task we have a arp_spoof.py script on the attacker VM.

We can check before applying the script the arp tables :

```
┌──(romain㉿kali)-[~]
└─$ arp -n
Adresse                  TypeMap AdresseMat          Indicateurs           Iface
192.168.47.20            ether   08:00:27:43:c8:da   C                     eth0
192.168.47.1             ether   08:00:27:3b:b7:8b   C                     eth0

┌──(romain㉿kali)-[~]
└─$
```

```
romain@ubuntu:~$ ip neigh
192.168.47.1 dev enp0s3 lladdr 08:00:27:58:e6:5c REACHABLE
192.168.47.10 dev enp0s3 lladdr 08:00:27:58:e6:5c STALE
```

```
^Cromain@ubuntuserv:~$ ip neigh
192.168.47.10 dev enp0s3 lladdr 08:00:27:58:e6:5c STALE
192.168.47.20 dev enp0s3 lladdr 08:00:27:58:e6:5c REACHABLE
romain@ubuntuserv:~$ _
```

Now we can on the Attacker VM run a capture in one terminal :

« sudo tcpdump -i etho -w capture_task1.pcap »

And on an other CMD we can run the python script to poison the arp tables :

<sudo python3 arp_spoof.py –victim_ip 192.168.47.20 –gateway_ip 192.168.47.1 –interface etho -v -f »

Now all the arp tables are poisonned :

```
romain@ubuntu:~$ ip neigh
192.168.47.1 dev enp0s3 lladdr 08:00:27:58:e6:5c REACHABLE
192.168.47.10 dev enp0s3 lladdr 08:00:27:58:e6:5c STALE
```

```
^Cromain@ubuntuserv:~$ ip neigh
192.168.47.10 dev enp0s3 lladdr 08:00:27:58:e6:5c STALE
192.168.47.20 dev enp0s3 lladdr 08:00:27:58:e6:5c REACHABLE
romain@ubuntuserv:~$ _
```

We can then run sudo tcpdump -i etho -w capture_task1_test.pcap

And run http query from the victim VM to ensure that the the traffic flows through the attacker.

TASK 2

For this task we need to capture common protocols and analyze them with a script.

Now we can on the Attacker VM run a capture in one terminal :

« sudo tcpdump -i etho -w capture_task2_test.pcap »

And then we can run the analyzer script :

```
└─$ sudo python3 traffic_interceptor.py ../capture_task2_test2.pcap
```

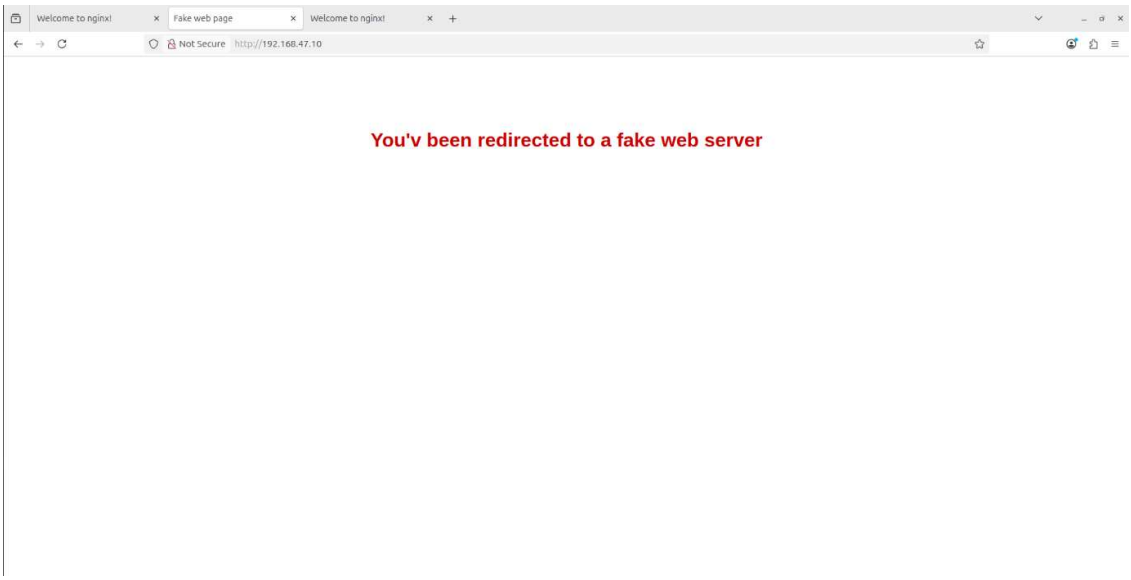
TASK 3

For this task we need to to intercept victim DNS queries and reply with attacker-controlled IPs for selected domains. We have a dns_spoof.py script on the attacker. But also a server.py script to run a smal fake server with a html file.

First we run the server.py script :

```
└─$ sudo python3 server.py
[*] Fake web server started on Port : 80 (http://192.168.47.10)
```

We get this website :



On an other terminal we run the arp_spoof.py script.

And on one last terminal we run the dns_spoof.py script :

```
┌──(romain㉿kali)-[~/Téléchargements]
└─$ sudo python3 dns_spoof.py --iface eth0 --config domain.json --victim 192.168.47.20 --forward
```

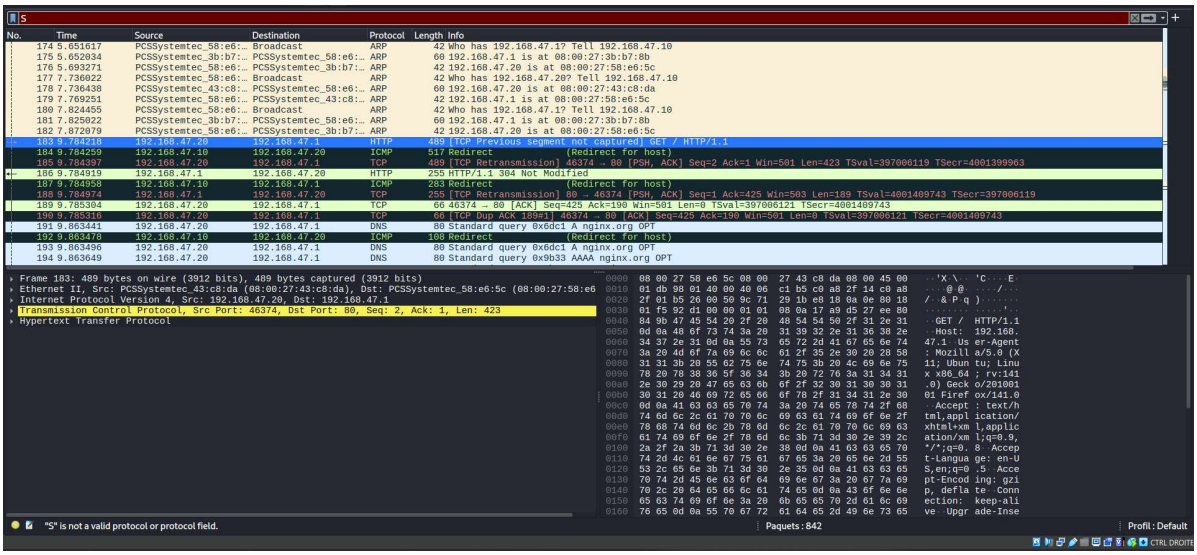
To run this script we need the domain.json it's this config file :

```
{
    "mode": "blacklist",
    "upstream_dns": "192.168.47.1",
    "targets": {
        "example.com": "192.168.47.10"
    }
}
```

When example.com will be called in the victim VM it will spoof the response with our fake service

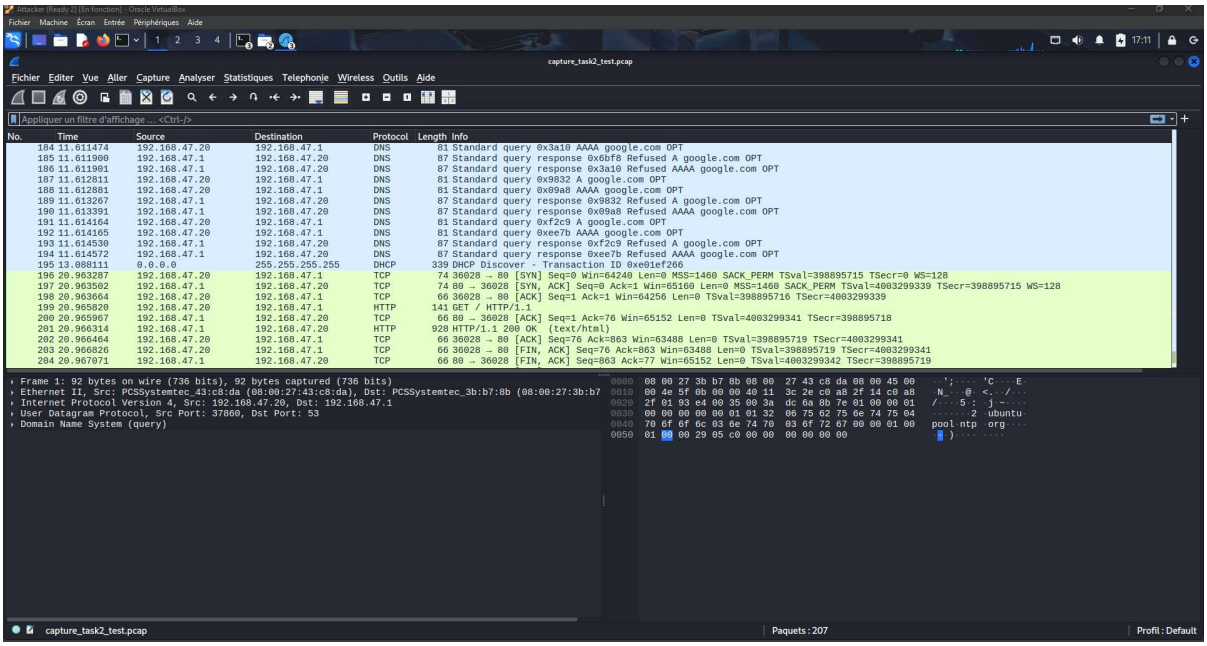
RESULTS

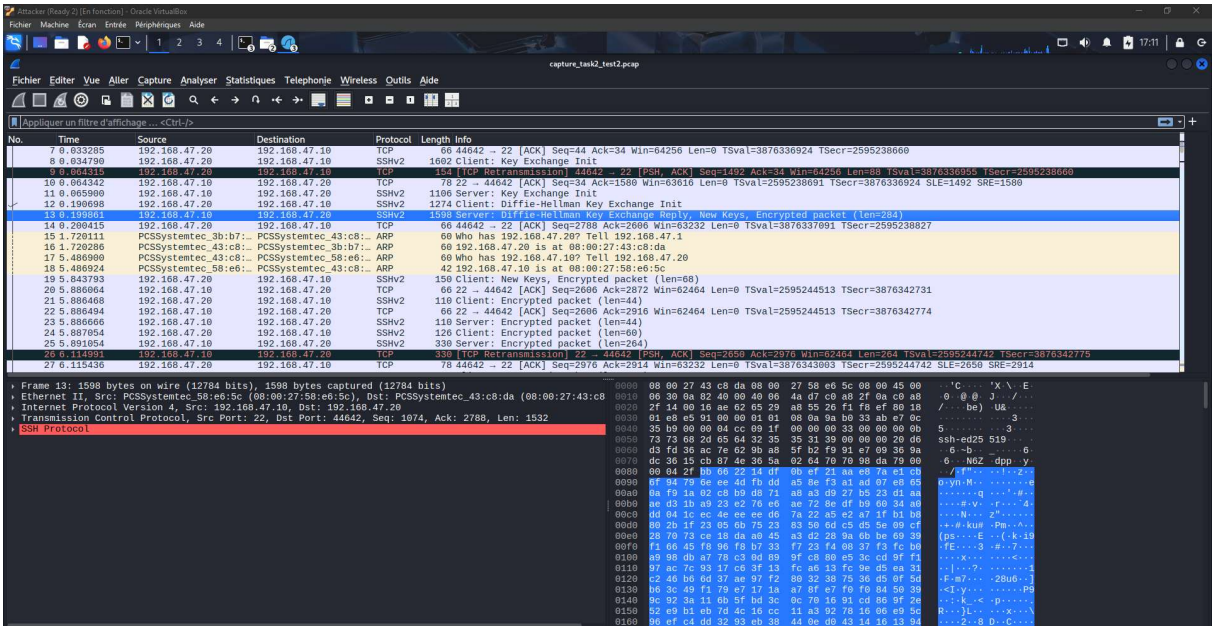
For task 1 :



We can see that the arp table are poisonned and that the the traffic flows through the attacker.

For task2 :





We can see both of the tested protocols : http and ssh.

And the results of the analyzes :

```
┌──(romain㉿kali)-[~/Téléchargements]
└─$ sudo python3 traffic_interceptor.py ../capture_task2_test.pcap
[*] Analyzing 207 packets from ../capture_task2_test.pcap...

......Analysis Results......

[+] Top Talkers (Source IP):
    192.168.47.20: 102 packets
    192.168.47.1: 100 packets
    0.0.0.0: 1 packets

[+] Protocol Counts (Simple):
    UDP: 193 packets
    TCP: 10 packets

[+] DNS Queries Intercepted:
    0.ubuntu.pool.ntp.org.
    1.ubuntu.pool.ntp.org.
    2.ubuntu.pool.ntp.org.
    3.ubuntu.pool.ntp.org.
    google.co.
    google.com.

[+] HTTP Hosts Visited (via Host Header):
    192.168.47.1
```

```
┌──(romain㉿kali)-[~/Téléchargements]
└─$ sudo python3 traffic_interceptor.py ../capture_task2_test2.pcap
[*] Analyzing 204 packets from ../capture_task2_test2.pcap ...

......Analysis Results......

[+] Top Talkers (Source IP):
    192.168.47.20: 98 packets
    192.168.47.1: 80 packets
    192.168.47.10: 16 packets

[+] Protocol Counts (Simple):
    TCP: 34 packets
    UDP: 160 packets

[+] DNS Queries Intercepted:
    0.ubuntu.pool.ntp.org.
    1.ubuntu.pool.ntp.org.
    2.ubuntu.pool.ntp.org.
    3.ubuntu.pool.ntp.org.

[+] HTTP Hosts Visited (via Host Header):
    No HTTP hosts found.
```

For task 3 :

```
[+] DNS query incoming.telemetry.mozilla.org from 192.168.47.10 → 192.168.47.1, id=30409
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=3217
[=] Sent spoofed answer for example.com → 192.168.47.10
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=3217
[=] Sent spoofed answer for example.com → 192.168.47.10
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=39619
[=] Sent spoofed answer for example.com → 192.168.47.10
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=39619
[=] Sent spoofed answer for example.com → 192.168.47.10
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=21883
[=] Sent spoofed answer for example.com → 192.168.47.10
[+] DNS query example.com from 192.168.47.20 → 192.168.47.1, id=21883
[=] Sent spoofed answer for example.com → 192.168.47.10
```

When Victim tried to navigate to example.com, dns_spoof.py sent a spoofed answer.

We have the logs of the server when executing server.py :

```
┌──(romain㉿kali)-[~/Téléchargements]
└─$ sudo python3 server.py
[*] Fake web server started on Port : 80 (http://192.168.47.10)
[log] Request received: 192.168.47.20 - GET / HTTP/1.1
[log] Request received: 192.168.47.20 - GET /favicon.ico HTTP/1.1
```

MITIGATION IDEAS

To reduce the risk of ARP/DNS MitM attacks on network, we need a layered strategy : isolate and segment the network, enable switch-level protections (DHCP Snooping/Dynamic ARP Inspection), encrypt and make DNS resolution harder (DoT/DoH, DNSSEC), and renforce HTTPS via HSTS.

These combined measures limit the attack surface and make MitM attacks much more difficult to happen.

ETHICS

All scripts were conducted only in an isolated virtual network environment (three VMs on an internal network with no external access). Before each task, snapshots were taken, and only test data was used.

The objective of this lab is strictly educational: to understand the vulnerabilities in order to better propose countermeasures. It's a important step to understand this type of attacks but we should never try to use them with illegal intentions.

The dammages caused by attacks like that can be devastating for people life : login and password lost to hackers, fake website redirections, informations stolen...