

DAT505 – ARP Spoofing DNS MitM with Scapy

Enrico Alberti

Student Number: 289195, mail: `enalb9899@uis.no`

PROJECT REPO

Contents

1	Setup	2
2	Task 1 - ARP Spoofing Tool	2
3	Task 2 — Traffic Capture Analysis	3
4	Task 3 — DNS Spoofing	4
5	Possible mitigation	6
6	Ethics	6

1 Setup

The assignment requires to use three virtual machines.

The VMs have been all set in an environment with a custom, completely isolated network.

Following is their description:

1. **Victim:** Ubuntu Linux

- *IP:* 192.168.200.129
- *MAC:* 00:0c:29:b1:65:fc

2. **Gateway:** Ubuntu Linux

- *IP:* 192.168.200.131
- *MAC:* 00:0c:29:c5:6c:fb

3. **Attacker:** Kali Linux

- *IP:* 192.168.00.128
- *MAC:* 00:0c:29:4b:e0:94

Following *ip a* command showing these properties:

Victim:
• IP: 192.168.200.129
• MAC: 00:0c:29:b1:65:fc

```
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP>
    link/ether 00:0c:29:b1:65:fc brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.129/24 brd 192.168.200.255
        valid_lft 1785sec preferred_lft 1785sec
        inet6 fe80::20c:29ff:feb1:65fc/64 scope link
            valid_lft forever preferred_lft forever
```

Gateway:
• IP: 192.168.200.131
• MAC: 00:0c:29:c5:6c:fb

```
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP>
    link/ether 00:0c:29:c5:6c:fb brd ff:ff:ff:ff:ff:ff
    altname enp2s0
    altname enx000c29c56cfb
    inet 192.168.200.131/24 brd 192.168.200.255
        valid_lft 1796sec preferred_lft 1796sec
        inet6 fe80::20c:29ff:fe4b:e094/64
            valid_lft forever preferred_lft forever
```

Attacker:
• IP: 192.168.200.128
• MAC: 00:0c:29:4b:e0:94

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>
    link/ether 00:0c:29:4b:e0:94 brd ff:ff:ff:ff:ff:ff
    altname ens160
    altname enx000c294b4e094
    inet 192.168.200.128/24 brd 192.168.200.255
        valid_lft forever preferred_lft forever
        inet6 fe80::20c:29ff:fe4b:e094/64
            valid_lft forever preferred_lft forever
```

Figure 1: VMs properties

2 Task 1 - ARP Spoofing Tool

The goal of this task is to poison the arp caches of both victim and gateway. To do so the script *arp_spoof.py*, which can be found here, finds MAC addresses in the network (by sending ARP who-has via *get_mac*). It enables IP forwarding (optionally) so traffic is routed through the attacker. The script repeatedly sends forged ARP replies (ARP (op=2)) to the victim/gateway claiming gateway/victim → attacker MAC.

Because both endpoints update their ARP tables, traffic between victim and gateway is forwarded through the attacker (MitM).

On exit the script restore previous tables.

This can be shown in Figure 2 (one frame of tcdump capture opened with wireshark) where we can see how a forged ARP (seen from attacker) claims that the mac address for the gateway is the one of the attacker.

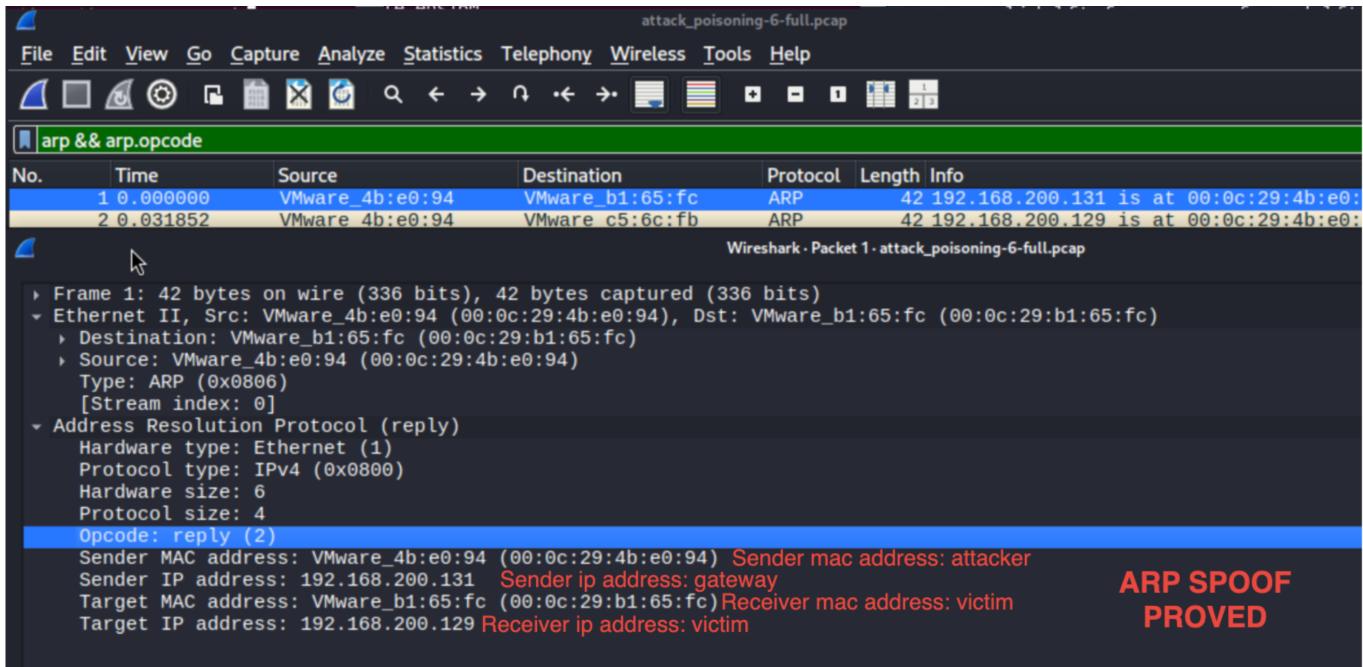


Figure 2: tcdump capture during ARP spoofing

Looking at the pre and post-forging ARP tables of Victim and Gateway (Figure 2) it is clear how the script worked correctly.

```
chicco@chicco-VMware20-1:~$ arp -a
? (192.168.200.254) at 00:50:56:f3:ec:ec [ether] on ens160
? (192.168.200.129) at 00:0c:29:b1:65:fc [ether] on ens160
? (192.168.200.128) at 00:0c:29:4b:e0:94 [ether] on ens160
? (192.168.200.1) at be:d0:74:91:7c:67 [ether] on ens160
chicco@chicco-VMware20-1:~$ arp -a
? (192.168.200.254) at 00:50:56:f3:ec:ec [ether] on ens160
? (192.168.200.129) at 00:0c:29:4b:e0:94 [ether] on ens160
? (192.168.200.128) at 00:0c:29:4b:e0:94 [ether] on ens160
chicco@chicco-VMware20-1:~$ arp -a
? (192.168.200.1) at be:d0:74:91:7c:67 [ether] on ens160
? (192.168.200.254) at 00:0c:29:c5:6c:fb [ether] on ens160
? (192.168.200.128) at 00:0c:29:4b:e0:94 [ether] on ens160
? (192.168.200.131) at 00:0c:29:4b:e0:94 [ether] on ens160
```

Figure 3: Pre-Post ARP spoofing for gateway (left) and victim (right)

This can be also seen in tcp dump while requesting arp via arping tool:

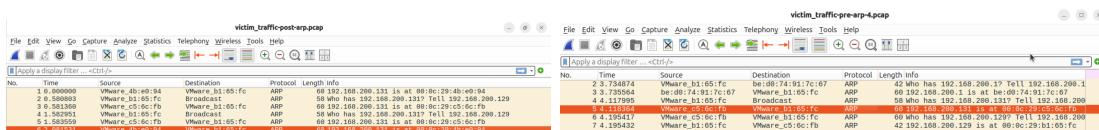


Figure 4: Pre-Post ARP spoofing, as seen in victim tcpdump

3 Task 2 — Traffic Capture Analysis

PCAPs logs can be found here, together with stats generated by the script `traffic_interceptor` which takes an input pcap file and computes 4 csv file results:

- urls: all urls found from pcap
- dns_queries: all dns queries
- top_talkers: the ips with the most packets sent
- proto_counts: which counts the occurrence of each protocol

The first annotated wireshark capture, which analyzes the ARP can be found in Figure 2. Following a HTTP capture analyzed with wireshark from a pcap file capture:

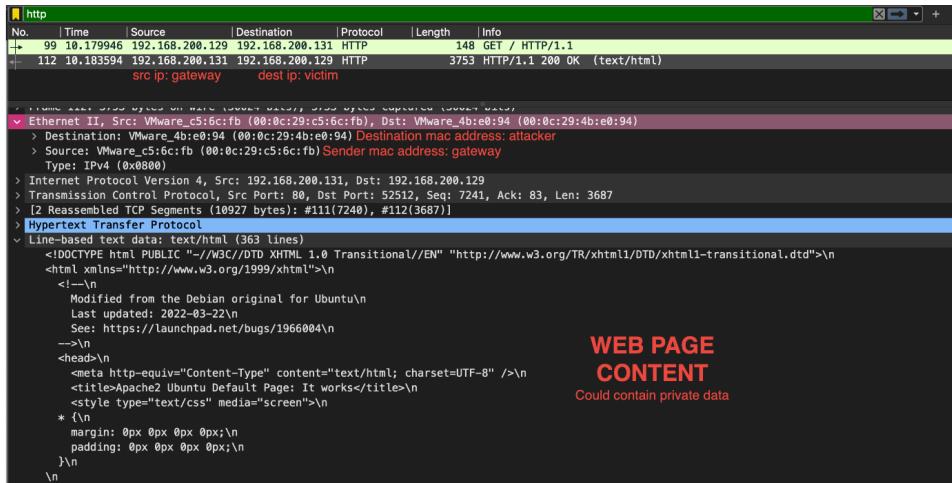


Figure 5: HTTP getter from victim, read by attacker

Before the ARP spoof we could not see this content. Now we can fully see the requested apache web page, which could contain sensible data from the victim.

4 Task 3 — DNS Spoofing

As requested by the assignment, a local DNS was set up on the gateway, providing resolution for a url which gave access to a Apache web server page, in our case `www.dat505.alberti`

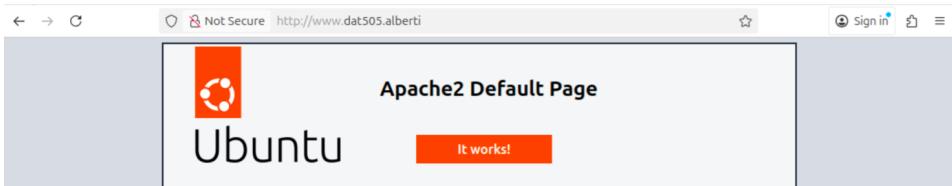


Figure 6: Victim connecting to apache web server with DNS resolved to Gateway apache server

The goal of this task is to intercept victim DNS queries and reply with attacker-controlled IPs for selected domains. The script to do so is `dns_spoof.py`. It intercepts DNS UDP/53 queries on the configured network interface and, for matched domain patterns, forges DNS A replies that map the queried name to attacker_ip while pretending to come from dns_server. Runs a background sniffer thread and sends multiple crafted Ethernet/IP/UDP/DNS packets to the client.

The result of this can be seen in the following wireshark analysis:

The figure consists of two NetworkMiner captures. The top capture, titled '4-dns-spoofing-redirection.pcap', shows forged DNS responses. A red circle highlights a row where the source is 192.168.200.128 and the destination is 192.168.200.131. An annotation points to this row with the text 'forged dns response points to attacker IP'. The bottom capture, also titled '4-dns-spoofing-redirection.pcap', shows authentic DNS responses. A red circle highlights a row where the source is 192.168.200.128 and the destination is 192.168.200.131. An annotation points to this row with the text 'authentic dns response points to gateway IP'.

No.	Time	Source	Destination	Protocol	Length	Info
22	2016-05-22T07:40:15.100000000Z	VMware_4b:e0:94	192.168.200.131	DNS	101	Standard query 0x10fa A www.dat505.alberti OPT request for www.dat505.alberti
23	2016-05-22T07:40:15.100000000Z	192.168.200.128	192.168.200.131	DNS	101	Standard query 0x10fa A www.dat505.alberti OPT request for www.dat505.alberti
24	2016-05-22T07:40:15.100000000Z	192.168.200.128	192.168.200.129	ICMP	129	Redirect (Redirect for host)
25	2016-05-22T07:40:15.100000000Z	192.168.200.129	192.168.200.131	DNS	101	Standard query 0x10fa A www.dat505.alberti OPT
26	2016-05-22T07:40:15.100000000Z	192.168.200.129	192.168.200.131	DNS	133	Standard query response 0x10fa A www.dat505.alberti A 192.168.200.131 OPT
27	2016-05-22T07:40:15.100000000Z	192.168.200.128	192.168.200.131	ICMP	161	Redirect (Redirect for host)
28	2016-05-22T07:40:15.100000000Z	192.168.200.131	192.168.200.129	DNS	133	Standard query response 0x10fa A www.dat505.alberti A 192.168.200.131 OPT

No.	Time	Source	Destination	Protocol	Length	Info
61	22.840419	VMware_4b:e0:94	VMware_b1:65:fc	ARP	42	192.168.200.131 is at 00:0c:29:4b:e0:94
62	22.879786	VMware_4b:e0:94	VMware_c5:6c:fb	ARP	42	192.168.200.129 is at 00:0c:29:4b:e0:94 (duplicate use of 192.168.200.131 detected)
63	24.370843	192.168.200.129	192.168.200.131	DNS	101	Standard query 0x02d3 A www.dat505.alberti OPT request for www.dat505.alberti
64	24.370892	192.168.200.128	192.168.200.129	ICMP	129	Redirect (Redirect for host)
65	24.370915	192.168.200.129	192.168.200.131	DNS	101	Standard query 0x02d3 A www.dat505.alberti OPT
66	24.395295	192.168.200.131	192.168.200.129	DNS	112	Standard query response 0x02d3 A www.dat505.alberti A 192.168.200.128
67	24.439467	192.168.200.131	192.168.200.129	DNS	112	Standard query response 0x02d3 A www.dat505.alberti A 192.168.200.128
68	24.439821	192.168.200.129	192.168.200.131	ICMP	140	Destination unreachable (Port unreachable)
69	24.439834	192.168.200.129	192.168.200.131	ICMP	140	Destination unreachable (Port unreachable)
70	24.487484	192.168.200.131	192.168.200.129	DNS	112	Standard query response 0x02d3 A www.dat505.alberti A 192.168.200.128

Figure 7: Pre and post DNS redirection packets

And we can obviously see the results when the victim tries to access `www.dat505.alberti`, which redirects to web page hosted by attacker, without the victims knowledge:

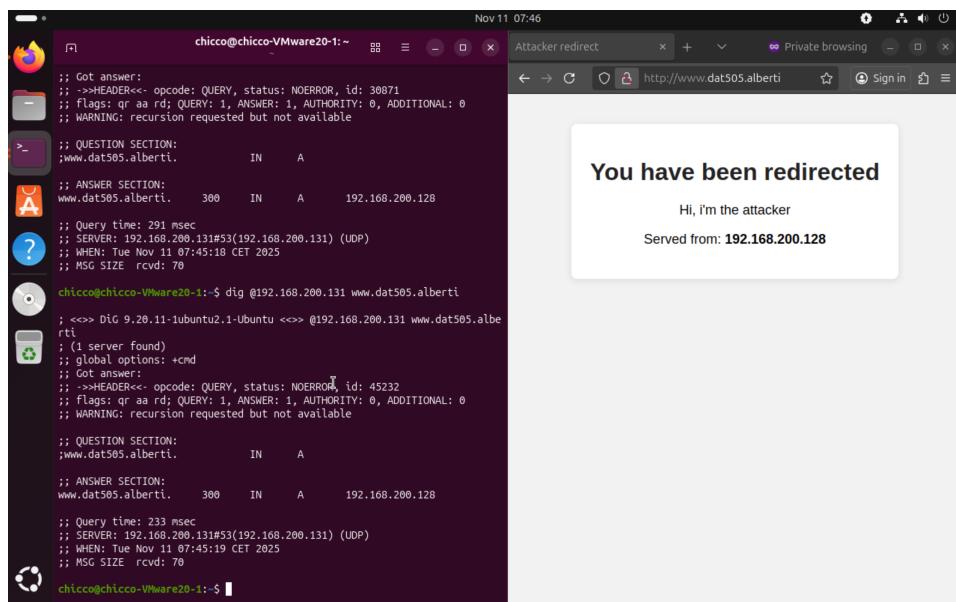


Figure 8: Redirection on web and dig

The script also enable two modes:

- Blacklist: all but the urls in this list get redirection
- Whitelist: the urls in this list get redirection

These are to be set in the config file given as input command with json format:

```
#dns_spoof_config.json
{
  "mode": "whitelist",
  "domains": ["www.dat505.alberti", "www.non.existent"],
```

```

    "forward_non_target": true
}

```

In this case those two links get redirected to attacker page. It is important to notice that `www.non.existent` is not present in the DNS resolutor. This means that we can redirect non existant pages, making use of possible typing errors by users. The results with this configuration is showed in Figure 4

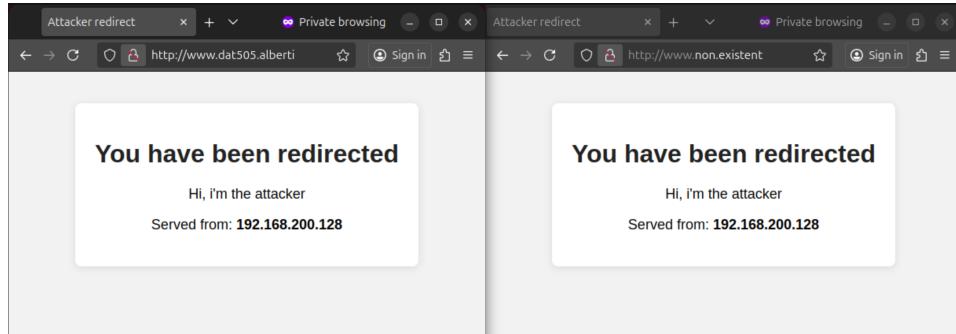


Figure 9: Redirection on multiple websites

5 Possible mitigation

- **Static ARP Tables:** It is often important to manually assign IP-MAC mappings. This way hosts will ignore spoofed ARP replies. However this could be time consuming for large networks, so best used for small, critical, networks.
- **Switch Security:** Some modern switches contain DAI (Dynamic ARP inspection) that validate ARP packets against known bindings, discarding the malicious ones
- **Physical Security:** Limiting physical access to network ports. Also implementing 802.1X auth ensures that only trusted endpoints can connect to the network.
- **Encryption:** Encryption does not stop ARP spoofing itself, but it limits its impact. In our case we were able to read web page content requested by victim since it used http and was clearly readable with a tcp dump, if it used HTTPS instead, the content would have been encrypted and impossible to read even in a MitM situation such as it was simulated here.

6 Ethics

All the operations done for this assignment were done in a controlled network, completely shut off from outside. Doing this *on field* could have dangerous ripercussions.