

Security testing in LAN: ARP-Spoofing, MITM Analysis on an exposed OS with countermeasures including network segmentation and VPN tunneling.

1. Project description

In this project, I built an isolated lab network to analyze how ARP-spoofing could affect IP-traffic in a LAN. I used different virtual machines, where one OS was Kali Linux used to simulate attacks, and the other OS was a targeted Windows 10 system. Measurements were done to analyze the consequences of the attack. These results were later used to implement various defenses such as network segmentation, VPN to isolate traffic, and different ARP rules. This project gave me a deeper understanding of LAN network architecture, threat models, and how they can be mitigated.

Note: If you're short of time and want to get a basic understanding of the different outcomes of this project, I would recommend jumping to: 3. *The attack in detail*, 4. *Consequence analysis*, 5. *Countermeasures* and 6. *Summary*.

2. Lab Setup

Entities:

- Type 2 Hypervisor was Virtualbox.
- Attacker: Kali Linux (used for ARP-spoofing and MITM-tests).
- Victim: Windows 10 Pro (22H2)
- Router / default gateway: Both attacker and victim were using the bridged adapter setting in Virtualbox to hop on my home LAN.

Network:

- Local Area Network (LAN), e.g 192.168.0.1/24.
- Router were used as default gateway for both attacker and victim.

3. The attack in detail

ARP-spoofing was performed on a victim host, followed by an analysis (4) made related to the consequences of the attack. Several countermeasures (5) were then implemented and also evaluated to demonstrate in which ways and to what degree the attack can be mitigated.

3.1 Preparations and initiation (see Figure 1 for context)

To demonstrate how ARP-spoofing affects devices on the same Layer-2 network, the attacker machine first identified hosts within the LAN and located both the victim and

the default gateway. For the MITM scenario to work the attacker started IP forwarding to allow traffic to keep being passed on rather than dropped.

Once the attacker had set up the environment, the attacker sent forged ARP replies to both victim and the gateway. These messages associated the attacker's MAC address with the gateway's IP address, causing the victim to route its traffic through the attacker. When it took effect, unencrypted traffic from the victim could be captured and inspected in real time, showing how serious an ARP-based MITM attack can be on an unsecured LAN.

3.2 ARP-spoofing demonstration (MITM)

- After initiation the attacker consistently sends out messages to the victim, portraying itself as the default gateway. The victim now unknowingly sends all its traffic directly to the attacker.

```
felix@kali: ~  
$ arp -a  
$ nmap -sn 192.168.0.1/24  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-21 12:56 CET  
Nmap scan report for 192.168.0.1  
Host is up (0.00056s latency).  
MAC Address: 68:02:B8:27:3E:E2 (Compal Broadband Networks)  
Nmap scan report for 192.168.0.39  
Host is up (0.000062s latency).  
MAC Address: E4:C3:2A:0B:AF:C5 (TP-Link Technologies)  
Nmap scan report for 192.168.0.46  
Host is up (0.00015s latency).  
MAC Address: 08:00:27:ED:8D:63 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)  
Nmap scan report for 192.168.0.254  
Host is up (0.00022s latency).  
MAC Address: 00:50:F1:64:CE:D7 (Maxlinear)  
Nmap scan report for 192.168.0.45  
Host is up.  
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.02 seconds  
$ arp -a  
? (192.168.0.1) at 68:02:b8:27:3e:e2 [ether] on eth0  
$ ping 192.168.0.46  
PING 192.168.0.46 (192.168.0.46) 56(84) bytes of data.  
64 bytes from 192.168.0.46: icmp_seq=1 ttl=128 time=0.399 ms  
64 bytes from 192.168.0.46: icmp_seq=2 ttl=128 time=0.258 ms  
64 bytes from 192.168.0.46: icmp_seq=3 ttl=128 time=0.264 ms  
--- 192.168.0.46 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2050ms  
rtt min/avg/max/mdev = 0.258/0.307/0.399/0.065 ms  
$ arp -a  
? (192.168.0.46) at 08:00:27:ed:8d:63 [ether] on eth0  
? (192.168.0.1) at 68:02:b8:27:3e:e2 [ether] on eth0  
$ sudo sysctl -w net.ipv4.ip_forward=1  
[sudo] password for felix:  
net.ipv4.ip_forward = 1  
$ sudo sysctl net.ipv4.ip_forward=1  
8:0:27:12:1e:23 8:0:27:ed:8d:63 0806 42: arp reply 192.168.0.1 is-at 8:0:27:12:1e:23
```

Figure 1: Shows address of 1. 192.168.0.46 = victim. 2. 192.168.0.1 = default gateway 3. Attacker executing command to forward packets to router 4. Attacker tricking the victim that it is now the router.

- We can verify in the victims ARP-table that the MAC-addresses are manipulated.

The screenshot shows a Windows Command Prompt window with the following content:

```

C:\Windows\system32>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : home
    IPv6 Address. . . . . : fd59:e58a:512b:534c:8374:667f:ad8d:a382
    Temporary IPv6 Address. . . . . : fd59:e58a:512b:534c:2d13:60f7:101e:e3ea
    Link-local IPv6 Address . . . . . : fe80::8ef2:cb1e:5797:10df%3
    IPv4 Address. . . . . : 192.168.0.46
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.0.1

C:\Windows\system32>arp -a

Interface: 192.168.0.46 --- 0x3

Internet Address      Physical Address      Type
192.168.0.1           68-02-b8-27-3e-e2    dynamic
192.168.0.45          08-00-27-12-1e-23    dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251           01-00-5e-00-00-fb    static
224.0.0.252           01-00-5e-00-00-fc    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Windows\system32>arp -a

Interface: 192.168.0.46 --- 0x3

Internet Address      Physical Address      Type
192.168.0.1           08-00-27-12-1e-23    dynamic
192.168.0.45          08-00-27-12-1e-23    dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251           01-00-5e-00-00-fb    static
224.0.0.252           01-00-5e-00-00-fc    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Windows\system32>

```

Handwritten annotations in the image include:

- Router**: A yellow arrow pointing to the Default Gateway (192.168.0.1) in the first IP configuration output.
- Before**: A blue circle around the first ARP table output, with a blue arrow pointing to the MAC address of 192.168.0.1 (68-02-b8-27-3e-e2).
- After**: A red circle around the second ARP table output, with a red arrow pointing to the MAC address of 192.168.0.1 (08-00-27-12-1e-23).

Figure 2: We can see that the default gateway has IP-address 192.168.0.1. We can see that the MAC address has been manipulated since before the attack it was 68-02-b8-27-3e-e2. While after the attack it had a new MAC address 08-00-27-12-1e-23, which is equivalent to the mac address of IP-address 192.168.0.45, which indicates that this must be the attacker.

3.3 The attacker collecting information: sniffing of HTTP-credentials

- The victim enters an unencrypted server, such as an HTTP server.
- Writes in username and password.
- The attacker captures this traffic in Wireshark and filters for login attempts as seen in Figure 3.

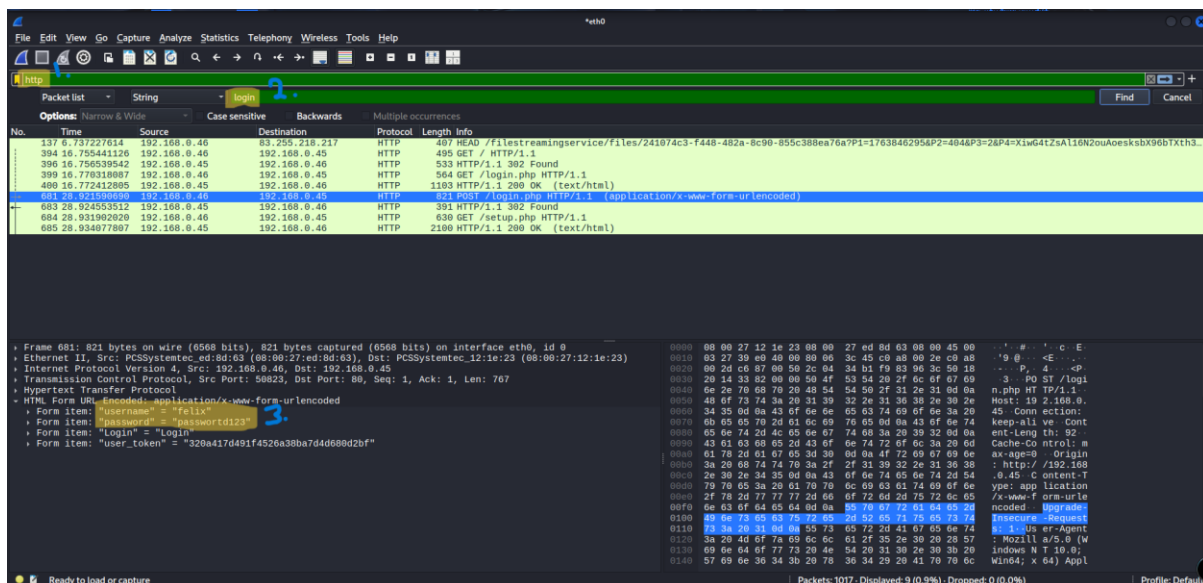


Figure 3: Shows a Wireshark capture that is: 1. Filtered for HTTP requests. 2. Filtered for string login. 3. Attacker finds the received packet with login credentials such as username and password of the victim.

4. Consequence analysis (attack summary)

The ARP-spoofing attack allowed the attacker to analyze confidential information related to the victim device. For example:

- The victim used an unencrypted HTTP login form, which contributed to login credentials being completely visible in Wireshark. This shows that any sensitive data sent without encryption can be intercepted and read by an attacker on the same LAN.
- All traffic from the victim is rerouted through the attacker, therefore any unencrypted protocol such as HTTP, DNS, FTP etc becomes entirely exposed.

5. Countermeasures

Based on the consequence analysis, several measures for defense were tested to understand how the ARP-based attacks could be mitigated in the case described earlier. The countermeasures were applied and verified in Linux and Windows 10 within the LAN.

1. Static ARP-posts

One effective mitigation strategy is to manually register the router's MAC address on every device connected to the network. This ensures that each device consistently recognizes the legitimate router, preventing ARP-spoofing attacks from altering the network. However, this approach requires manual configuration on each individual device, which can be time consuming and quite impractical, especially in environments with many clients or frequently changing devices.

2. Network segmentation:

Another good mitigation tactic is network segmentation. Since ARP spoofing requires the attacker to be in the same data link layer as the victim, the LAN was split into separate segments using OPNsense. Basically, 2 new virtual LANs were created, both using OPNsense as the default gateway acting as a central of control or firewall for traffic between the two segments.

The positive thing with using network segmentation like this, is that not only do we completely erase the possibility to arpspoof (because of the different segmentations), but we can also assert different rules for traffic coming in and out of our now controlled network segmentations. For example, it is possible to block all traffic from the Kali VM's VLAN to the windows VLAN, or vice versa, but we could also specifically deny communication from the Kali host to a specific IP address inside the Windows VLAN, even if other hosts in the same segment would be allowed.

```

Kali (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
felix@kali:~$ ip r
default via 192.168.50.1 dev eth0 proto dhcp src 192.168.50.124 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.50.0/24 dev eth0 proto kernel scope link src 192.168.50.124 metric 100
felix@kali:~$ ip netns exec OPNsense.internal (192.168.50.1) at 00:00:27:b6:ac:5c [ether] on eth0
felix@kali:~$ sudo systemctl --no-wait --no-drop net.ipv4.ip_forward=1
[sudo] password for felix:
net.ipv4.ip_forward = 1
felix@kali:~$ sudo arpspoof -i eth0 -t 192.168.60.11 192.168.50.1
arpspoof: couldn't arp for host 192.168.60.11
felix@kali:~$ sudo systemctl --no-wait --no-drop net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
felix@kali:~$ ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data:
64 bytes from 192.168.60.11: icmp_seq=1 ttl=127 time=0.539 ms
64 bytes from 192.168.60.11: icmp_seq=2 ttl=127 time=0.477 ms
64 bytes from 192.168.60.11: icmp_seq=3 ttl=127 time=0.508 ms
64 bytes from 192.168.60.11: icmp_seq=4 ttl=127 time=0.585 ms
64 bytes from 192.168.60.11: icmp_seq=5 ttl=127 time=0.492 ms
64 bytes from 192.168.60.11: icmp_seq=6 ttl=127 time=0.467 ms
^C
--- 192.168.60.11 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5117ms
rtt min/avg/max/mdev = 0.467/0.496/0.539/0.022 ms
felix@kali:~$ ping 192.168.60.11
PING 192.168.60.11 (192.168.60.11) 56(84) bytes of data:
^C
--- 192.168.60.11 ping statistics ---
20 packets transmitted, 0 received, 100% packet loss, time 27753ms

Windows10 (Running) - Oracle VM VirtualBox
File Machine View Input Devices Help
Administrator: Command Prompt
C:\Windows\system32>ipconfig

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : internal
Link-local IPv6 Address . . . . . : fe80::8ef2:cba5:5797:18d9%3
IPv4 Address. . . . . : 192.168.60.11
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.60.1
  
```

Figure 4: (1–2) Windows host in VLAN 60 with IP 192.168.60.11 and default gateway 192.168.60.1. (3–4) Kali host in VLAN 50 with IP 192.168.50.124 and gateway 192.168.50.1. (5) arpspoof against 192.168.60.11 fails because the target is on a different L2 segment. (6) ICMP ping works while inter-VLAN traffic is allowed. (7) After an OPNsense rule is added, ICMP from Kali to Windows is blocked.

3. HTTPS / TLS

Another simple way would be to just use HTTPS / TLS protocol everywhere, to make sure that all traffic is encrypted. This makes it very hard for the attacker to read anything even if they manage to get a hold of the traffic. However, it doesn't really address the problem.

4. VPN tunnel via WireGuard

We can also setup a VPN tunnel via WireGuard where we transfer all traffic from the victim through a designated VPN tunnel. In this case the victim only communicates through the VPN tunnel, which makes it useless for the attacker to listen to any traffic between the victim and the router. If the attacker finds the IP-address of the VPN setup the attacker will still be

able to execute a MITM attack. However, the packets will be WireGuard encrypted and therefore the information that exists in the packets will almost be useless for the attacker.

This is what the setup looked like, where a Linux VPN had to be set up to act as the WireGuard VPN server.

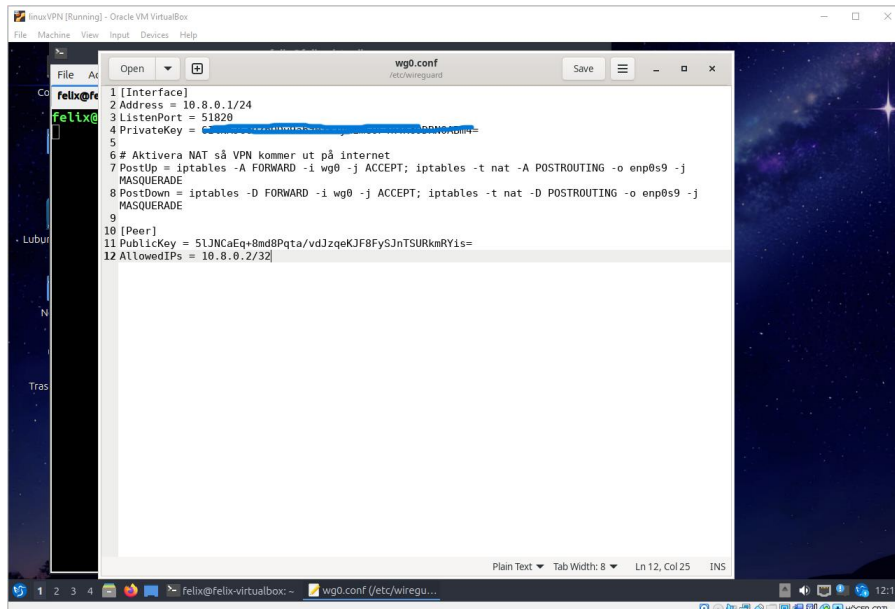


Figure 5: The Linux machine acting as the VPN server on this specific LAN.

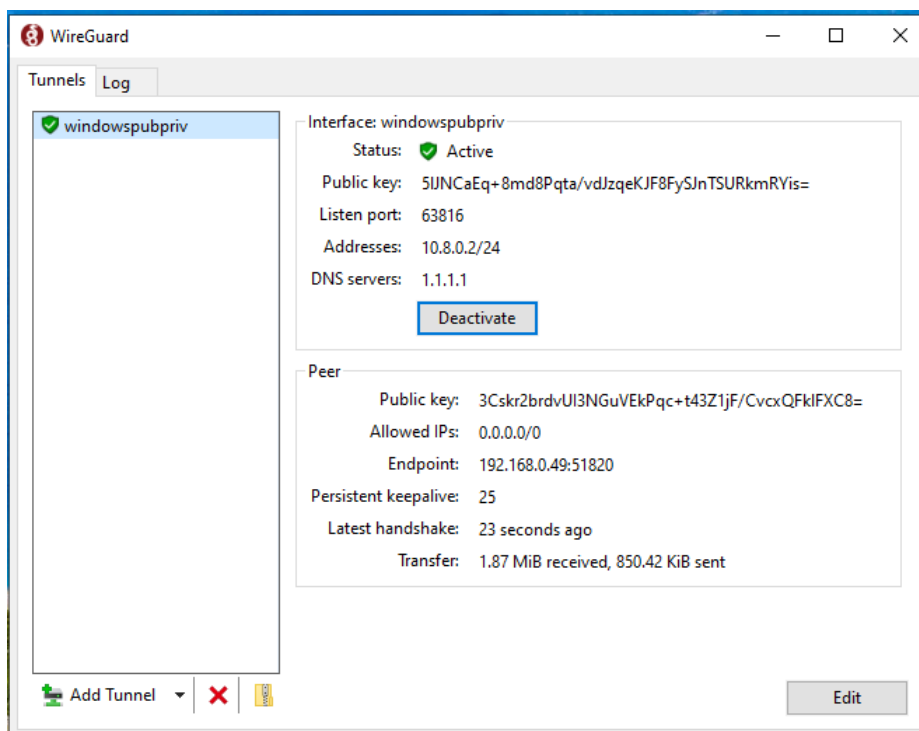


Figure 6: After configuring the VPN server on the Linux machine, we added the server's public key on the Windows client and specified the server's endpoint (IP address and port). This defines where the WireGuard tunnel is established and where the windows machine forwards its traffic.

6. Summary

From the attack shown in 3 we can see that it is quite simple for an attacker to ARP-spoof a victim considering they are on the same LAN and without security measures on the LAN. However there are several countermeasures that can be thought of to mitigate the attack, such as network segmentation, secure protocols, VPN tunneling via WireGuard and much more. All the countermeasures tested in this project provide security measures from different angles and can be useful in different ways depending on what the circumstances are. Sometimes a combination of different countermeasures might be beneficial, such as network segmentation together with different static ARP-rules to provide a wide security and mitigate full exposure.