ARP Spoofing

Second Assignment NS

Lorenzo Cappetti Matricola:7165929

November 2024

1 Introduction

In this paper, I decided to delve into **ARP Spoofing**, a cyber attack targeting **local area networks (LAN)** by exploiting inherent vulnerabilities in the **Address Resolution Protocol (ARP)**. The primary objective was to understand the mechanisms behind this type of attack, analyzing its effects and the potential implications for network security.

My choice to focus on ARP Spoofing was driven by several reasons. Firstly, although this attack is relatively simple to execute, it is highly effective and serves as a foundation for more complex attacks, such as **Man-in-the-Middle (MITM)**. These attacks enable an attacker to intercept, alter, or block network traffic between two devices, thereby compromising the confidentiality and integrity of communications.

Moreover, ARP Spoofing exploits **by design** vulnerabilities in the ARP protocol, which lacks any form of authentication for incoming responses. This absence of security mechanisms makes the protocol particularly susceptible to manipulation, allowing an attacker to impersonate other devices on the network with relative ease.

I chose this attack also because, being relatively lightweight in terms of computational resources and technical complexity, it seemed like an excellent starting point for exploring the field of cybersecurity, given that I am still a beginner in this domain. The ease of executing the attack allowed me to focus on analyzing the results and exploring potential countermeasures to protect local networks from such threats.

2 ARP Protocol

The **Address Resolution Protocol (ARP)** is a fundamental protocol within **local area networks (LAN)**, used to resolve IP addresses into **MAC (Media Access Control)** addresses, thereby enabling communication between devices

within the same network. When a device (e.g., a computer) needs to send data to another device on the same LAN but does not know the destination MAC address, it sends an ARP broadcast request to all devices on the network, asking, "Who has the IP address X.X.X.X?". The device with that IP address responds by providing its own MAC address, thus allowing the transmission of packets.

Despite its crucial role, the ARP protocol suffers from **by design** vulnerabilities. One of the primary weaknesses is that it does not include any authentication mechanisms to verify the validity of received ARP responses. This means that devices within the network passively accept any ARP response, without checking if it was solicited by an actual request. This lack of verification paves the way for an attack known as **ARP Spoofing**.

2.1 ARP Spoofing

ARP Spoofing [5], also known as **ARP Poisoning**, is an attack in which an attacker sends falsified ARP packets into the local network, with the intent of tricking target devices into believing that the attacker's MAC address corresponds to a legitimate IP address (usually that of the network gateway). The attack generally unfolds in three stages:

1. **Cache Poisoning Phase:** The attacker sends fake ARP responses to both the gateway and the victim device, associating their own MAC address with the IP address of the gateway (or another critical device on the network). This causes the target devices to update their ARP caches, leading them to send packets to the attacker instead of the legitimate recipient.

2. **Traffic Interception Phase (Man-in-the-Middle):** Once the ARP caches are poisoned, the attacker positions themselves in the middle of the communication flow between the victim and the gateway. This allows the attacker to intercept, modify, or block network traffic. The attacker can choose to forward the packets to the original gateway (remaining undetected) or manipulate the content of the communications.

3. **Secondary Attack Phase:** After gaining control over the traffic, the attacker can carry out further attacks, such as credential sniffing.

3 Attack and Documentation

After studying the theoretical background of the attack, I proceeded to execute it in a simulated environment.

I conducted the entire process on a Linux operating system using **Oracle VirtualBox** [4], in order to create the virtual machines required for the simulation.

I set up two virtual machines, both running **Ubuntu 24.10** with a network interface set to NAT mode.

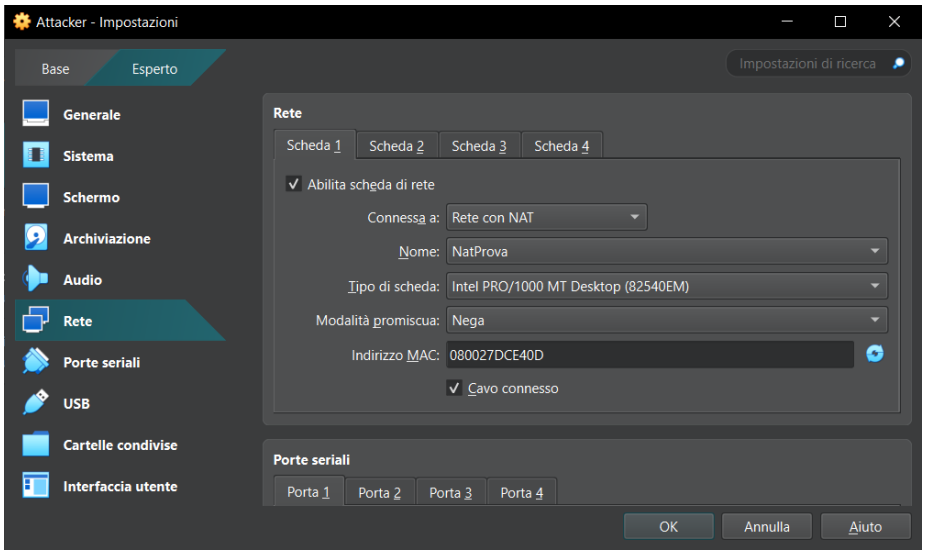

Figure 1: Network Settings

This configuration ensures that the virtual machines are on the same subnet and are also able to connect to the internet.

3.1 Simulation Setup

After completing this step, I requested the IP address of the machine, which we will refer to as the victim, using the following command:

```
1 ip a
```

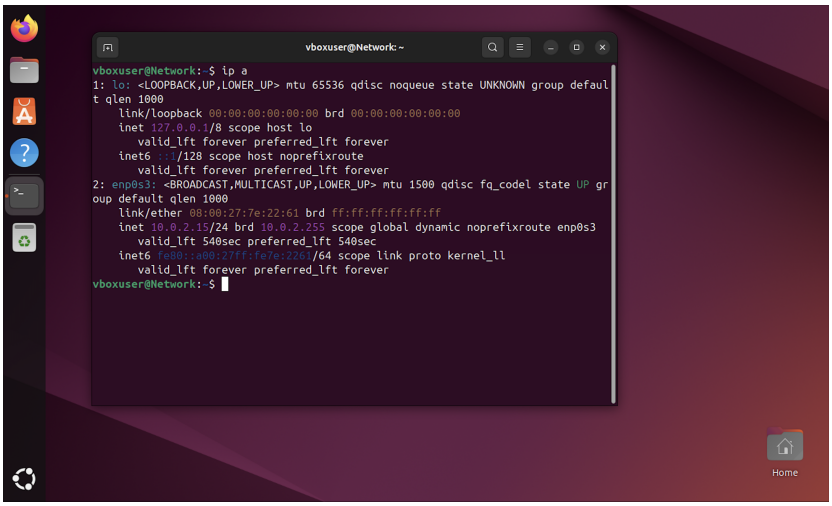

Figure 2: Victim's IP address

Once the IP address of the victim was obtained, which in our case is **10.0.2.15**, we proceeded to work on the attacker's machine.

I then executed the following series of commands in order: [1] [3]

1. ```
   sudo bettercap
   ```

   This command starts **Bettercap**, a cybersecurity tool used for penetration testing and network monitoring.

2. ```
   net.probe on
   ```

   This activates the **net.probe** module, which performs a network scan to identify all connected devices.

3. ```
   set arp.spoof.fullduplex true
   ```

   With this configuration, I enable ARP spoofing in **full-duplex** mode, meaning ARP packets are sent to both the victim and the router. In other words, I position myself between the victim and the router, intercepting all bidirectional traffic.

4. ```
   set.arp.spoof.targets 10.0.2.15
   ```

   I set a specific **target** for the ARP spoofing attack.

5. ```
   set net.sniff.local true
   ```

   This command enables local traffic capture; Bettercap will sniff traffic passing through the network interface of the device it is running on.
   ```

6.
```
arp.spoof on
```

This activates the ARP Spoofing module, causing Bettercap to send falsified ARP packets to deceive target devices into thinking the attacker's device is the network gateway (router).

7.
```
net.sniff on
```

This command activates Bettercap's network sniffer, capturing all visible network traffic through your network interface. This traffic may include passwords, login credentials, and other sensitive data.

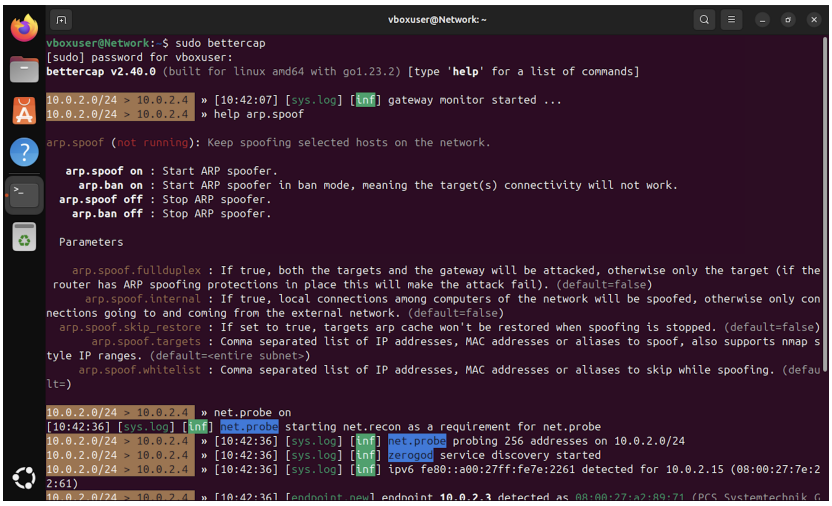

Figure 3: Attack sterup

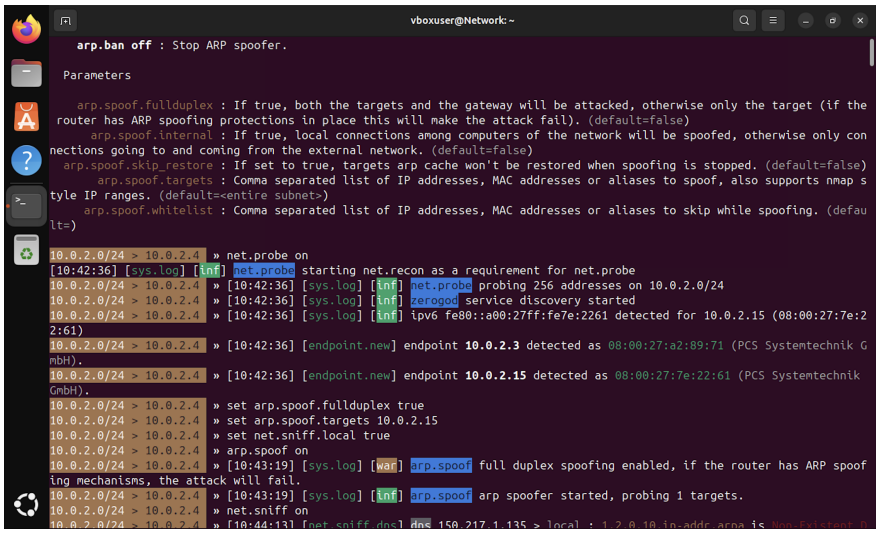

Figure 4: Attack sterup

3.2 Attack Simulation and Results

Once the attacking machine was prepared, which at this point was already ready to collect information from the victim, I searched for a webpage using the victim's browser.

The page searched was the Amazon homepage.

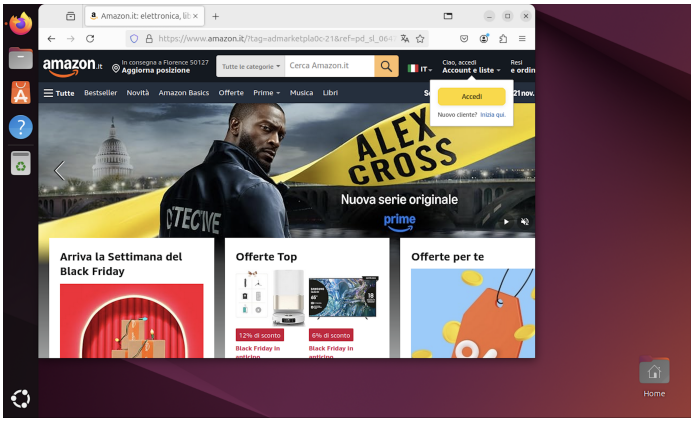

Figure 5: Amazon page search on the victim's VM

At this point, I checked if it was displayed on the attacking machine's terminal.

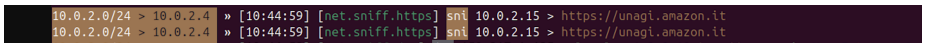

Figure 6: Sniff of the Amazon page

We can clearly see from the image that the attacking machine receives the traffic from the victim and is able to view the page that the victim searched for, in our case, Amazon.

To complete the attack and its verification, I checked if the victim's login credentials were also **sniffed**.

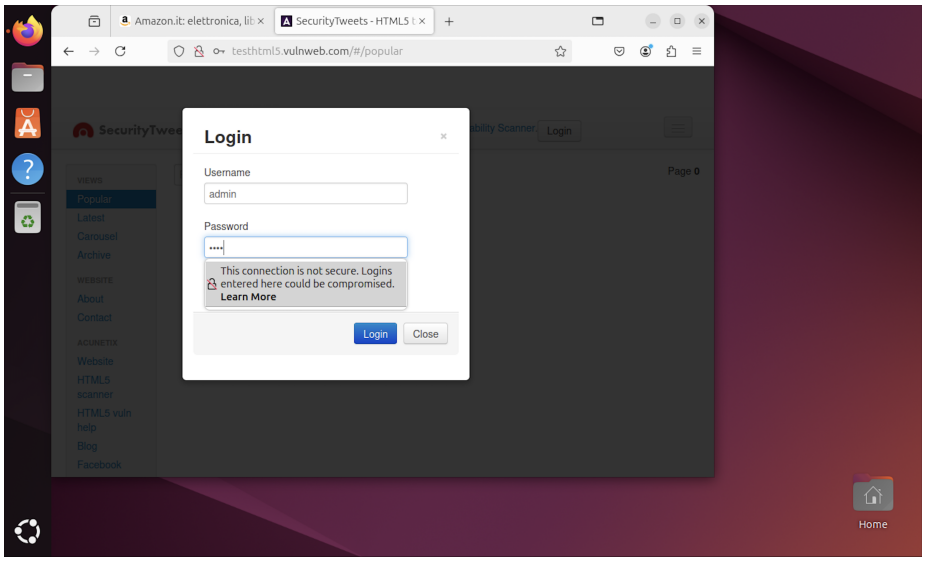

Figure 7: Page search and login

For this test, I used the website:

```
1 http://testhtml5.vulnweb.com/#/popular
```

and logged in with the credentials: **admin** and **test**.

At this point, I checked to see if these credentials could be seen in the attacking machine's terminal.

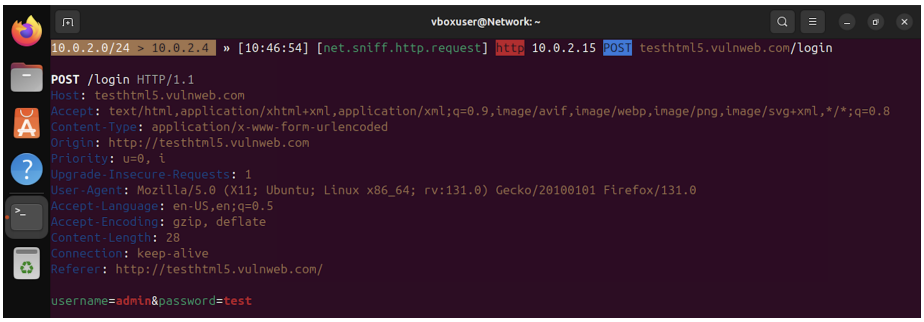

Figure 8: Sniff of the credentials

As we can clearly see, we were able to sniff the login credentials.

The simulation was successful, and we were able to sniff both the web pages visited by the victim and the credentials they entered for login on a webpage.

4 Conclusion and Mitigation

In conclusion, the **ARP Spoofing** attack represents a significant threat to the security of local area networks (LANs), as it exploits inherent vulnerabilities in the ARP protocol. Through the stages of ARP cache poisoning, traffic interception, and potential secondary attacks, an attacker can compromise the confidentiality and integrity of data transmitted over the network, putting sensitive information at risk, such as login credentials and personal data.

The experiment conducted highlighted how easy it is to execute an ARP Spoofing attack using common tools like **Bettercap**.

To mitigate the effects of an ARP Spoofing attack, several countermeasures can be implemented, including [2]:

- **Using Static ARP**: Manually configure the ARP tables on network devices so that critical IP addresses (such as those of the gateway) are associated with known and trusted MAC addresses. However, this solution may be impractical in large networks.
- **Enabling security features on switches**: Many modern switches offer features such as *Dynamic ARP Inspection (DAI)*, which can detect and block suspicious ARP packets, preventing cache poisoning.
- **Implementing firewalls and intrusion detection systems (IDS)**: Tools like *Snort* can be configured to detect suspicious ARP Spoofing activity and generate real-time alerts.

In summary, while ARP Spoofing demonstrates how vulnerable a local network can be due to weaknesses in the ARP protocol, several strategies can be

adopted to strengthen network security. The adoption of these measures, combined with continuous training and updates to security policies, is essential to protect network infrastructures from ever-evolving cyber threats.

References

[1] YouTube, *Bettercap ARP Spoofing Man In The Middle - MITM 15.2*, `https://www.youtube.com/watch?v=kUQqTk3SOq8`

[2] ChatGPT, OpenAI, *Conversazione con ChatGPT*, 15 Novembre 2024.

[3] Bettercap, *Bettercap - Network Manipulation Tool*, `https://www.bettercap.org`

[4] Oracle Virtualbox, *Oracle Virtualbox*, `https://www.virtualbox.org`

[5] ARP Spoofing—Veracode, `https://www.veracode.com/security/arp-spoofing`