ARP Spoofing using Ubuntu

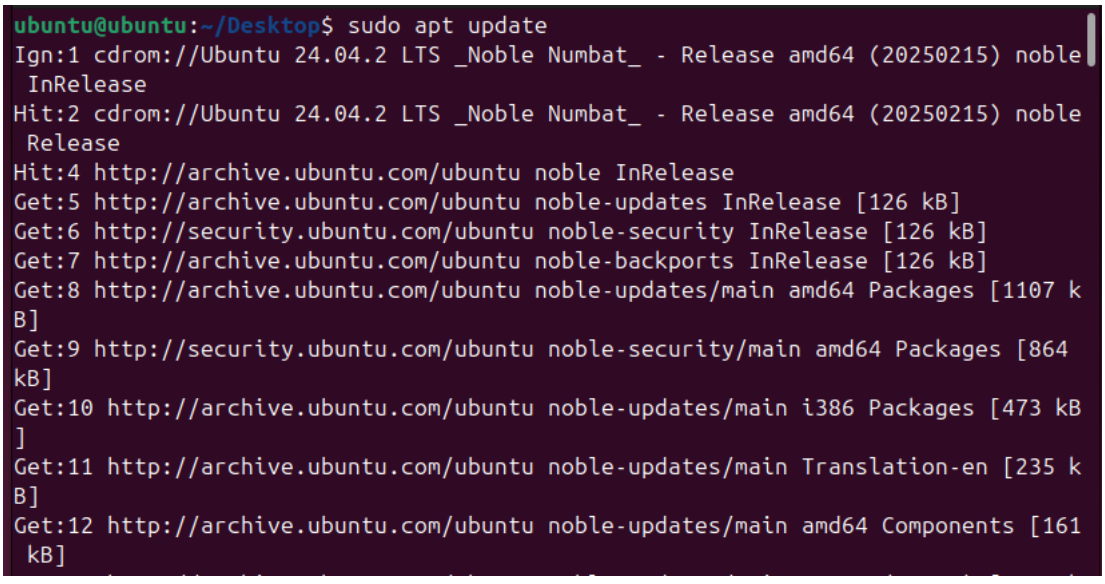
Introduction to ARP Spoofing

ARP (Address Resolution Protocol) Spoofing is a type of cyberattack where an attacker sends fake ARP messages on a local network. This tricks other devices into believing that the attacker's machine is the router or another trusted device. As a result, the attacker can intercept, modify, or stop data sent between devices. ARP Spoofing is commonly used for Man-in-the-Middle (MITM) attacks and helps demonstrate the importance of network security and encryption.

Step-by-Step Execution of ARP Spoofing

Step 1: Installation of Required Tools

In the first step, necessary tools like dsniff arp-scan net-tools were installed on the Ubuntu system. These tools are essential for scanning the network and launching the ARP spoofing attack. The installation ensured that all required software components were available to perform the attack successfully.



```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
283 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ubuntu:~/Desktop$ sudo apt install dsniff arp-scan net-tools
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libencode-perl libnet1 libnids1.21t64 libtext-csv-perl libtext-csv-xs-perl
The following NEW packages will be installed:
  arp-scan dsniff libencode-perl libnet1 libnids1.21t64 libtext-csv-perl
  libtext-csv-xs-perl net-tools
0 upgraded, 8 newly installed, 0 to remove and 283 not upgraded.
Need to get 2823 kB of archives.
After this operation, 14.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 libnet1 amd64 1.1.6+dfsg
-3.2build1 [44.5 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble/universe amd64 libnids1.21t64 amd64
 1.26-2.1build2 [21.9 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble/universe amd64 libtext-csv-perl all
 2.04-1 [109 kB]
```

Step 2: Checking Network Information

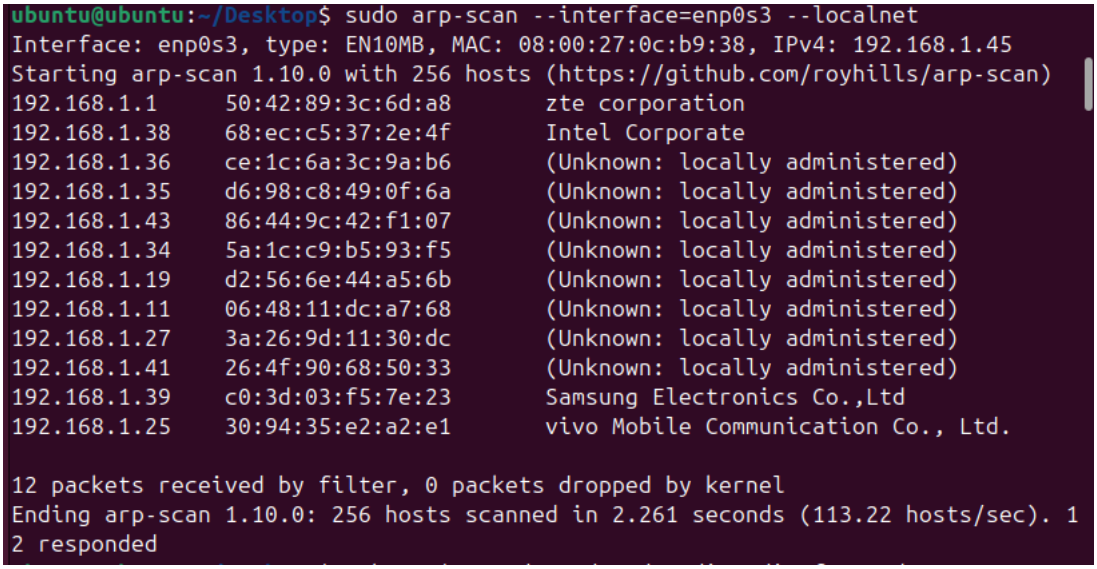
In this step, the network details of the attacker's system were identified. This included the IP address of the attacker's machine, the default gateway (router), and the name of the network interface (such as Wi-Fi or Ethernet). Knowing this information is important to identify where the attack should be directed.

```
ubuntu@ubuntu:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:0c:b9:38 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.45/24 brd 192.168.1.255 scope global dynamic noprefixroute en
p0s3
       valid_lft 86041sec preferred_lft 86041sec
    inet6 fe80::a00:27ff:fe0c:b938/64 scope link
       valid_lft forever preferred_lft forever
```

```
ubuntu@ubuntu:~/Desktop$ ip route | grep default
default via 192.168.1.1 dev enp0s3 proto dhcp src 192.168.1.45 metric 100
```

Step 3: Scanning the Network

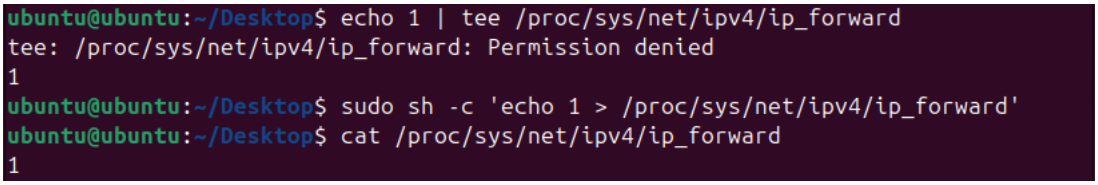
The third step involved scanning the local network to identify all connected devices. The purpose was to find a suitable target device (such as a mobile phone or another computer) on the same network. The IP and MAC address of the target were recorded for the attack.



In this we successfully locate our targeted mobile device which is the second-last device having **Ip-address of 192.168.1.39** and **Mac address of c0:3d:03:f5:7e:23** having name **Samsung Electronics Co. ,Ltd.**

Step 4: Enabling IP Forwarding

Before launching the attack, IP forwarding was enabled on the attacker's system. This allows the attacker to act as a bridge between the target and the router, forwarding traffic while secretly observing or manipulating it. It is a necessary step to maintain communication flow during the spoofing.



Step 5: Performing the ARP Spoofing Attack

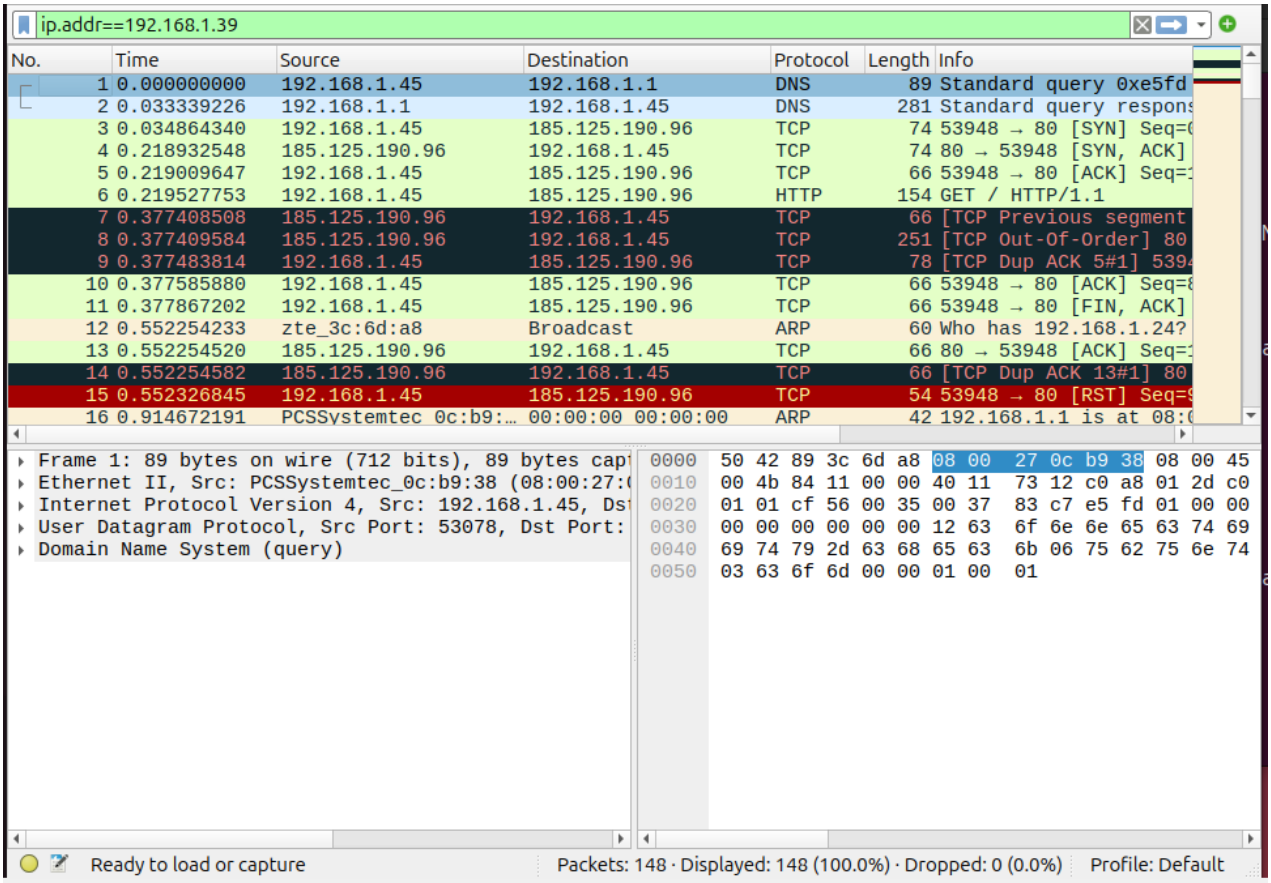
The core of the process was executed in this step. The attacker impersonated the router to the target device by sending fake ARP responses. This caused the target device to unknowingly send its internet traffic to the attacker instead of directly to the router, enabling a Man-in-the-Middle position.

Step 6: Observing and Cleaning Up

Once I started the ARP spoofing attack, I monitored the victim's device to see if there were any visible changes. Since I had enabled IP forwarding on my Ubuntu system, the victim's internet connection remained active, and everything appeared normal to them. However, behind the scenes, all their network traffic was passing through my machine.

To confirm this, I opened Wireshark and began capturing packets. I could see traffic coming from the victim's IP address, including DNS requests and other data, which proved that I was successfully acting as a Man-in-the-Middle. This step showed me how attackers can quietly intercept sensitive data without the victim even knowing.

After observing and taking all the necessary screenshots, I decided to end the attack. I stopped the ARP spoofing tool and disabled IP forwarding. This restored normal communication between the victim and the router. Cleaning up was important to make sure the network returned to its original state and no devices were affected after the test.




```
ubuntu@ubuntu:~/Desktop$ sudo wireshark
 ** (wireshark:7060) 17:37:40.021798 [GUI WARNING] -- QStandardPaths: XDG_RUNTIM
E_DIR not set, defaulting to '/tmp/runtime-root'
 ** (wireshark:7060) 17:40:43.160812 [Capture MESSAGE] -- Capture Start ...
 ** (wireshark:7060) 17:40:43.239119 [Capture MESSAGE] -- Capture started
 ** (wireshark:7060) 17:40:43.239484 [Capture MESSAGE] -- File: "/tmp/wireshark_
enp0s3SOSK72.pcapng"
 ** (wireshark:7060) 17:42:46.506699 [Capture MESSAGE] -- Capture Stop ...
 ** (wireshark:7060) 17:42:46.564431 [Capture MESSAGE] -- Capture stopped.
 ** (wireshark:7060) 17:42:46.564498 [Capture WARNING] ./ui/capture.c:722 -- cap
ture_input_closed():
 ** (wireshark:7060) 17:43:49.368445 [Capture MESSAGE] -- Capture Start ...
 ** (wireshark:7060) 17:43:49.461287 [Capture MESSAGE] -- Capture started
 ** (wireshark:7060) 17:43:49.461592 [Capture MESSAGE] -- File: "/tmp/wireshark_
enp0s3YK3F72.pcapng"
 ** (wireshark:7060) 17:46:23.723335 [Capture MESSAGE] -- Capture Stop ...
 ** (wireshark:7060) 17:46:23.750544 [Capture MESSAGE] -- Capture stopped.
 ** (wireshark:7060) 17:46:23.750704 [Capture WARNING] ./ui/capture.c:722 -- cap
ture_input_closed():
```

Conclusion

This activity provided me the practical experience in conducting a basic ARP spoofing attack using Ubuntu. Through each step, the attack demonstrated how unprotected networks can be vulnerable to manipulation and data interception. It emphasized the importance of network-level security practices such as encryption, strong authentication, and ARP protection techniques. Understanding such attacks is essential for learning how to defend against them in real-world scenarios.

The End