NETWORK EXPLOIT

SUBJECT NAME: CRYPTOGRAPHY AND NETWORK SECURITY

SUBJECT CODE: CS6008

MODULE: 4

NAME	BHUVANESHWAR S
REG.NO	2019103513
DATE	05/04/2022

AIM:

Using ARP Poison network and detect using Wireshark.

TOOLS INVOLVED:

- Kali linux VM (2)
- Ettercap
- Wireshark
- Terminal
- python
- Oracle virtual box

PROBLEM DESCRIPTION:

The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address. This mapping is a critical function in the Internet protocol suite. The devices of the network peel the header of the data link layer from the protocol data unit (PDU) called frame and transfer the packet to the network layer (layer 3 of OSI) where the network ID of the packet is validated with the destination IP's network ID of the packet and if it's equal then it responds to the source with the MAC address of the destination, else the packet reaches the gateway of the network and broadcasts packet to the devices it is connected with and validates their network ID.The above process continues till the second last network device in the path reaches the destination where it gets validated and ARP, in turn, responds with the destination MAC address.

An ARP spoofing, also known as ARP poisoning, is a Man in the Middle (MitM) attack that allows attackers to intercept communication between network devices. The attack works as follows:

The attacker must have access to the network. They scan the network to determine the IP addresses of at least two devices—let's say these are a workstation and a router.

The attacker uses a spoofing tool, such as Arpspoof or Driftnet, to send out forged ARP responses.

The forged responses advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address. This fools both router and workstation to connect to the attacker's machine, instead of to each other.

The two devices update their ARP cache entries and from that point onwards, communicate with the attacker instead of directly with each other.

The attacker is now secretly in the middle of all communications.

INPUT:

The user search or visit the insecure website.

OUTPUT:

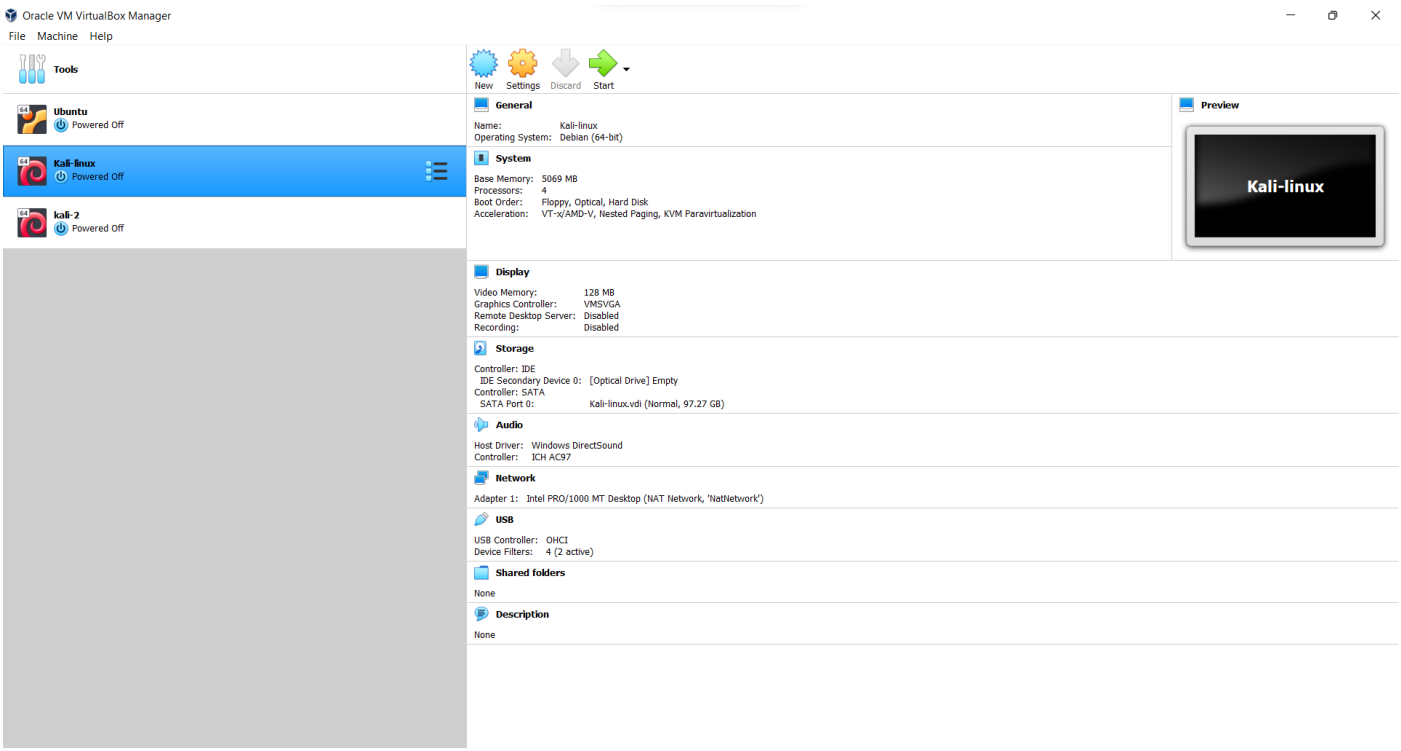
Stole personal information from the user.

SCREENSHOT:

Two kali linux machine are used to do ARP spoofing.

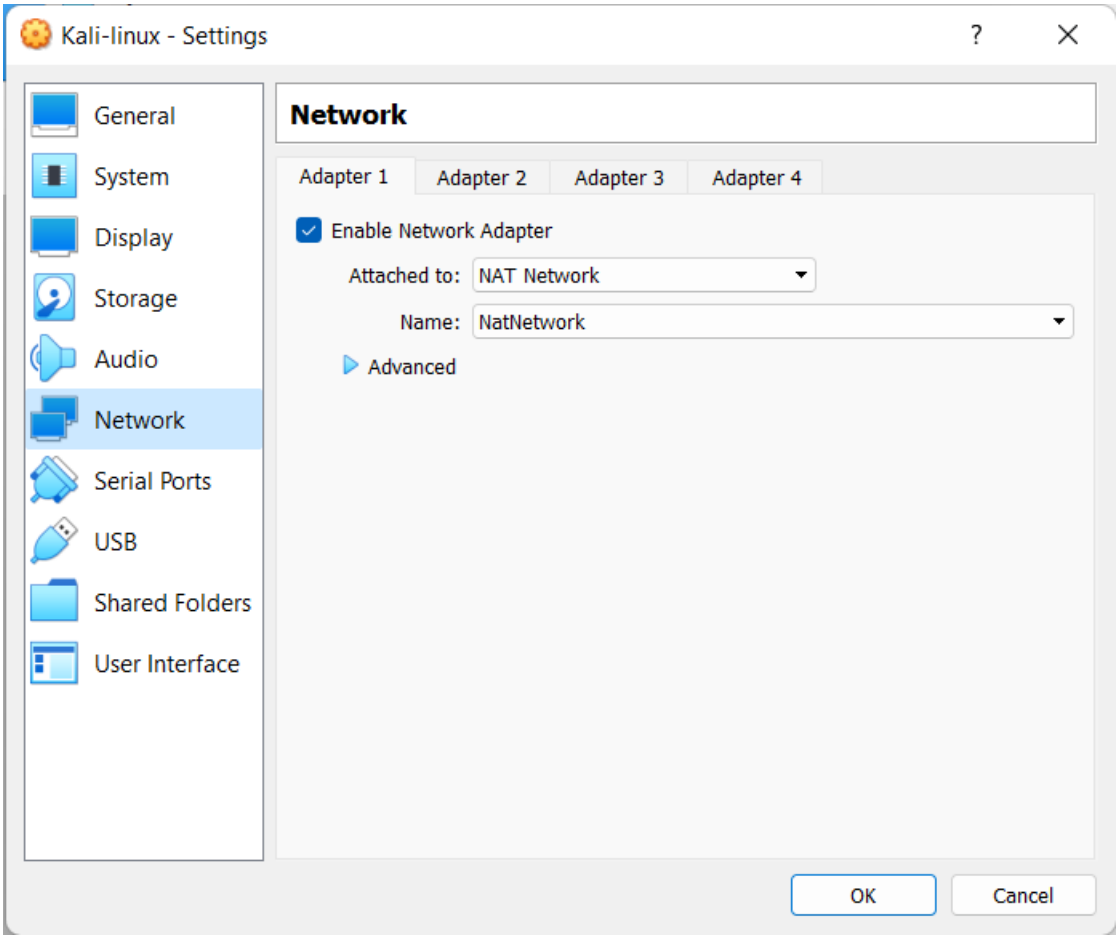
Machine 1 -> attacker

Machine 2 -> user



Here machine 1 is kali-linux and machine 2 is kali 2.

NAT network is used in both virtual machines.



Using the **ifconfig** command, to see the ip address , mac address and default gateway.

MACHINE 1

```
┌──(bhuvan㉿bhuvan-kali)-[~]
└─$ 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:feec:29c3  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:ec:29:c3  txqueuelen 1000  (Ethernet)
        RX packets 2  bytes 650 (650.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 25  bytes 3311 (3.2 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
```

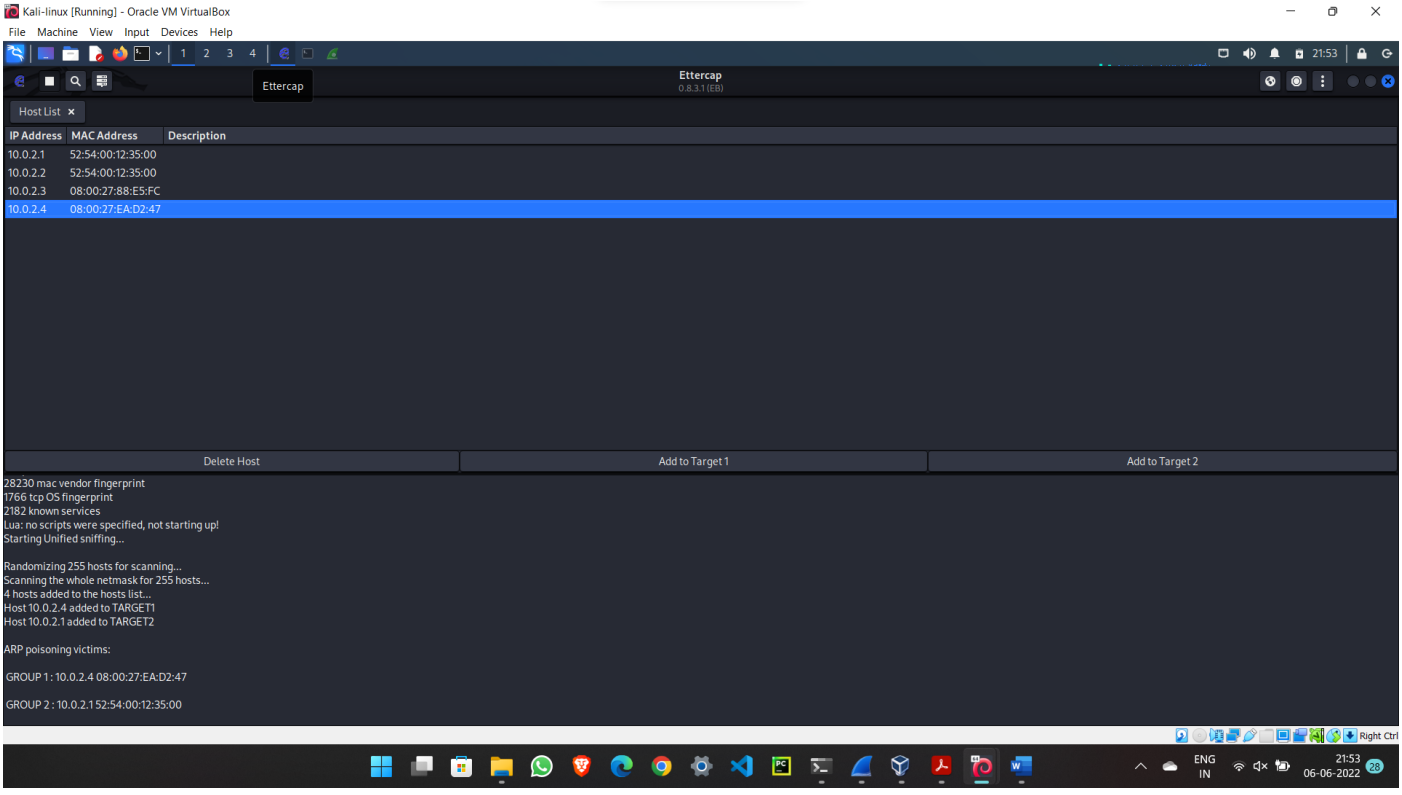
MACHINE 2

```
┌──(kali㉿kali)-[~]
└─$ 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 fd17:625c:f037:2:9443:9396:54d:9413  prefixlen 64  scopeid 0x0<
global>
        inet6 fd17:625c:f037:2:a00:27ff:fe1d:7431  prefixlen 64  scopeid 0x0<
global>
        inet6 fe80::a00:27ff:fe1d:7431  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:1d:74:31  txqueuelen 1000  (Ethernet)
        RX packets 18  bytes 3890 (3.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30  bytes 5320 (5.1 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 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
```

Ettercap is a comprehensive suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis

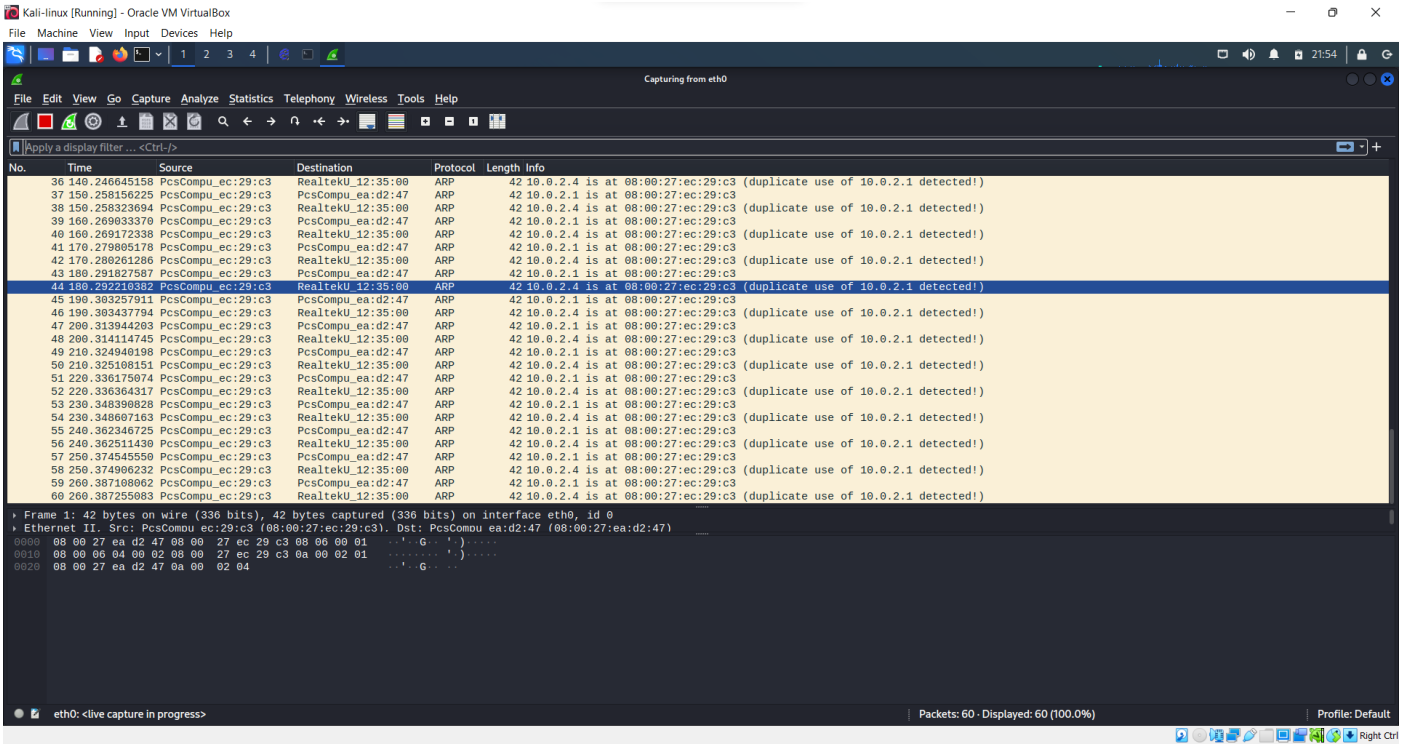
ETTERCAP – MACHINE 1 (ATTACKER)



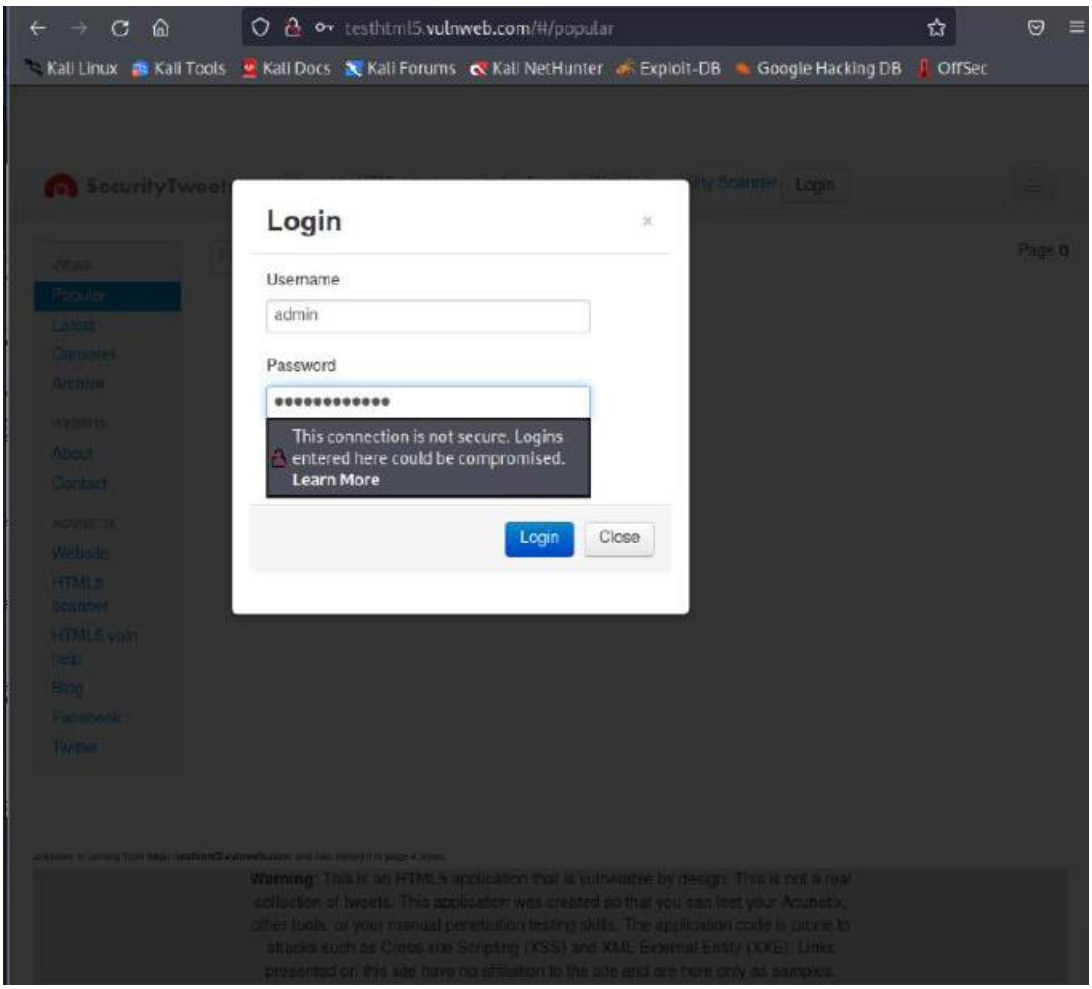
Poisoning the user ip address

WIRESHARK – MACHINE 1 (ATTACKER)

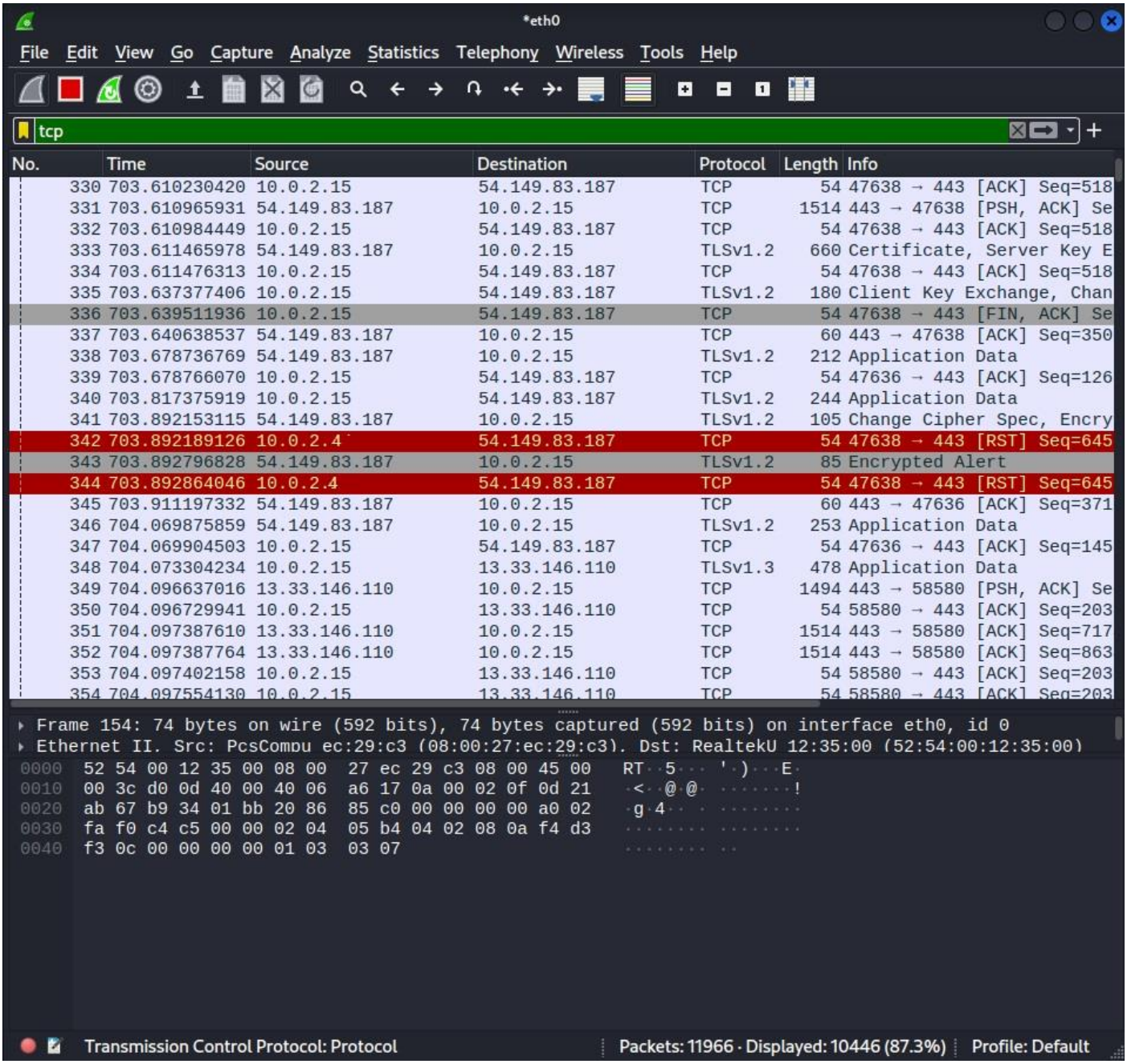
Wireshark capturing of the spoof packets



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



Attacker can see the credential information from the user using wireshark.



We can create our own arp spoof detection. So I wrote in python.

```
import scapy.all as scapy

def mac(ipadd):
    arp_request = scapy.ARP(pdst=ipadd)
    br = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_req_br = br / arp_request
    list_1 = scapy.srp(arp_req_br, timeout=5, verbose=False)[0]
    return list_1[0][1].hwsrc

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

def process_sniffed_packet(packet):
```

```
    if packet.haslayer(scapy.ARP) and packet[scapy.ARP].op == 2:
        originalmac = mac(packet[scapy.ARP].psrc)
        responsemac = packet[scapy.ARP].hwsrc

        if originalmac != responsemac:
            print("[*] ALERT!! You are under attack, the ARP table is being poisoned.!")

sniff("eth0")
```

OUTPUT