ANALYSING OF PCAP FILE

AKANNI OLUWATOBI ELIJAH

STUDENTID:22060310

UNIVERSITY OF HERTFORDSHIRE

CYBER OPERATION

7COM1069-0901-2023

14th of December 2023

TABLE OF CONTENT

TABLE OF FIGURES

ABSTRACT

This report covers how the analysis of the PCAP was done and how to mitigate the attack found when analysed. Organisations can secure their network from any attack by incorporating input validation, access controls, secure coding practices, and network-level security measures. Organisations can create a defence-in-depth strategy that provides multiple layers of protection by combining input validation, access controls, safe coding practices, and network-level security measures. Regular monitoring, testing, and upgrading of security mechanisms is also required to keep ahead of developing threats and assure web application security.

INTRODUCTION

MI4, a company with questionable behaviour on its network, has requested that I, as a cyber analyst, analyse a specific PCAP file. Networks can be defined by their connections, number of nodes, and connectivity patterns. Network analysis involves studying the relationships between nodes and understanding how the network operates. PCAP files are created using a program and contain network packet data, which is used to analyse network characteristics, control network traffic, and determine network

1.1.Purpose

The analysis aims to assist MI4 Corporation in comprehending the suspicious activity on their network, identifying potential threats, and evaluating the security level of their network. Examining the PCAP file aims to provide insights into the origin of the threat and develop strategies to mitigate existing and future threats. This analysis will aid the corporation in enhancing network security and safeguarding critical assets.

1.2.Scope

The scope of the analysis is to investigate a given PCAP file from the MI4 Corporation's network to find any suspicious activity and potential threats. The investigation will involve various filtering techniques to isolate communication-related to critical assets and find any anomalies or unusual activities.

1.3.Asset Identification

Using Wireshark, a network traffic analysis tool on Kali Linux, I examined the PCAP file to identify potential threat assets. Wireshark allows for specific network traffic filtering, such as by IP address, port number, protocol, data transfer, and login attempts. By utilising the code "Ip.addr == [IP Address], tcp.port == [port number], HTTP, (tcp.flags.syn == 1 and tcp.flags.ack == 0)", I was able to filter and analyse the captured traffic, communication, and potential threats.

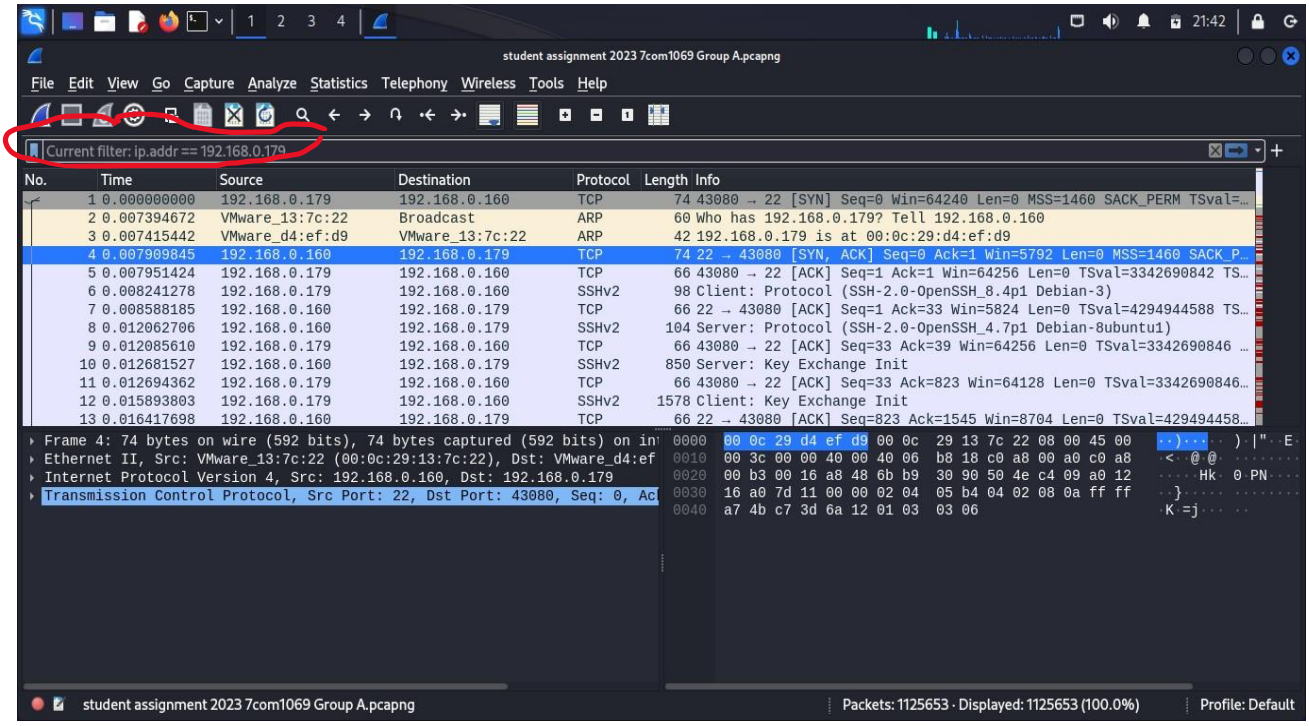


Figure 1. It shows the overview of pcap file

By examining the network packet capture with Wireshark and applying specific filters, we can identify potential threats and anomalies within the captured data. These could include suspicious HTTP requests, unusual user agent strings, or unexpected file transfer patterns that may risk critical assets.

Once potential threats and anomalies are identified, they can be classified according to their severity and impact.

For instance, abnormal patterns in HTTP requests may be classified as a medium-severity threat with a moderate effect on the availability of the web server.

Based on the findings from the PCAP file and the fictional critical assets, a threat model can be created to explain the identified threats. For example, the threat model can outline the potential threat of a DDoS attack on the web server and its impact on the website's availability. Recommendations for mitigating these threats, such as implementing DDoS protection measures or monitoring for unusual HTTP traffic related to critical assets, can also be included in the threat model.

THREAT MODELLING AND THREAT ACTOR PROFILING

2.1. Threat Modelling

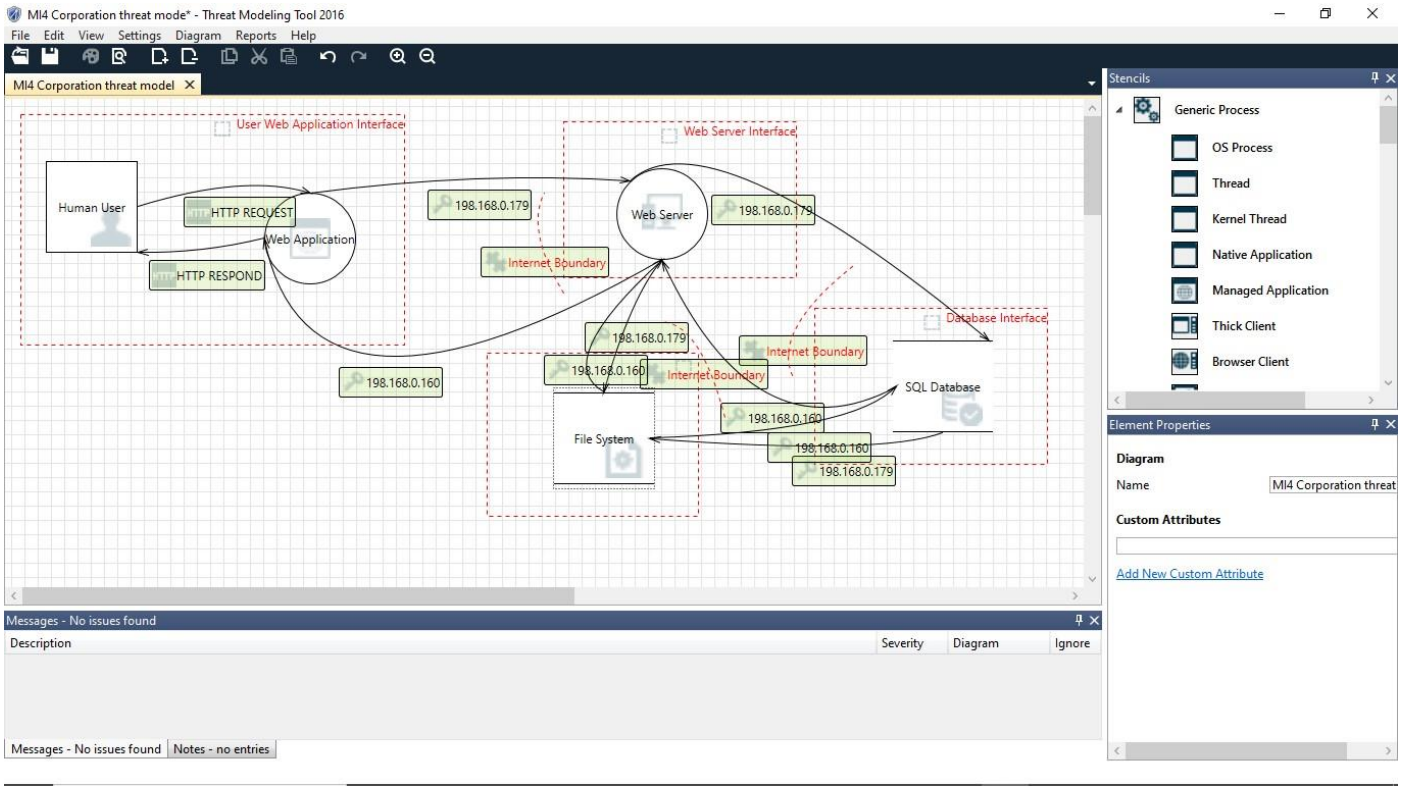


Figure 2. Show the threat modelling

The diagram above depicts the hacker's efforts to extract information from the web server. The hacker, shown as the human user in the graph, initiated a request to the web application using the specified IP address 198.168.0179. The hacker tried to exploit path traversal/directory traversal via the web application to access the web server but was unsuccessful, receiving a 404 NOT FOUND response from the web server through the web application. The arrows in the model graph indicate that path traversal may be accessible through the web server *(see Figure 8)*.

Subsequently, the hacker made another request to the web server via the web application, attempting a local file inclusion attack but again receiving a 404 NOT FOUND response *(see Figure 9)*.

The hacker also attempted to access the Server's database using SQL injection, resulting in a response in the PCAP file with the number 669046 *(see Figure 10)*.

Finally, the hacker uses the command injection technique to execute arbitrary commands on the host OS. The attacker launched an attack against the web server, which was ultimately rejected *(see Figure 11)*.

2.2. Threat Actor Profiling

A cyber threat actor is an individual or group that presents a security risk. These actors are responsible for carrying out cyberattacks and are often categorised based on factors such as motive, method of attack, and the targeted industry. Threat actors include hacktivists, cybercriminals, thrill seekers, cyber terrorists, insider threats, and nation-state actors. (Richard, 2001)

Thrill seekers are individuals who attack computers and information systems for enjoyment. Some aim to steal sensitive information or data, while others seek to understand better how networks and computer systems operate through hacking *(see Figure 2: it shows what the attack did)*.

The attacker must gain more technical expertise in web application development, system administration, network security, and database management. They have limited knowledge of web application vulnerabilities and how to exploit them and may have experience with SQL and command injection vulnerabilities. The attacker aims to gain unauthorised access to sensitive files, execute commands on servers, and extract sensitive information from databases, posing a significant threat to web applications and system security.

A thrill seeker targets computer systems for self-validation, learning, or experimentation. Between the 1980s and 1990s, they were commonly called "hackers" or white-hat hackers. Although thrill seekers do not intend to cause harm to systems, they are curious about understanding how things operate and may inadvertently create unexpected issues for systems and products. In contrast, well-structured vulnerability bounty program problems may be made by thrill seekers, which may transform these individuals into cost-effective testing engineers. (Mirko, et al., 2020)

STATISTICAL ANALYSIS

3.1. PCAP File Properties

The properties of the capture file are utilised to provide comprehensive details about the PCAP file, revealing various aspects of the data. Wireshark's capture file properties can be used to retrieve information regarding different attributes, such as identifying the source and destination IP addresses, displaying the port within the captured network, showing the protocol versions used in the captured network, indicating the duration and time range of the captured network, presenting details about the size and number of packets captured, and verifying the file format and version of the captured data.
In summary, Wireshark's PCAP file attributes offer valuable information about the captured network traffic that can be used for analysis, troubleshooting, and security investigations. The collected PCAP file contains 1125653 packets *(see Figure 3)*.

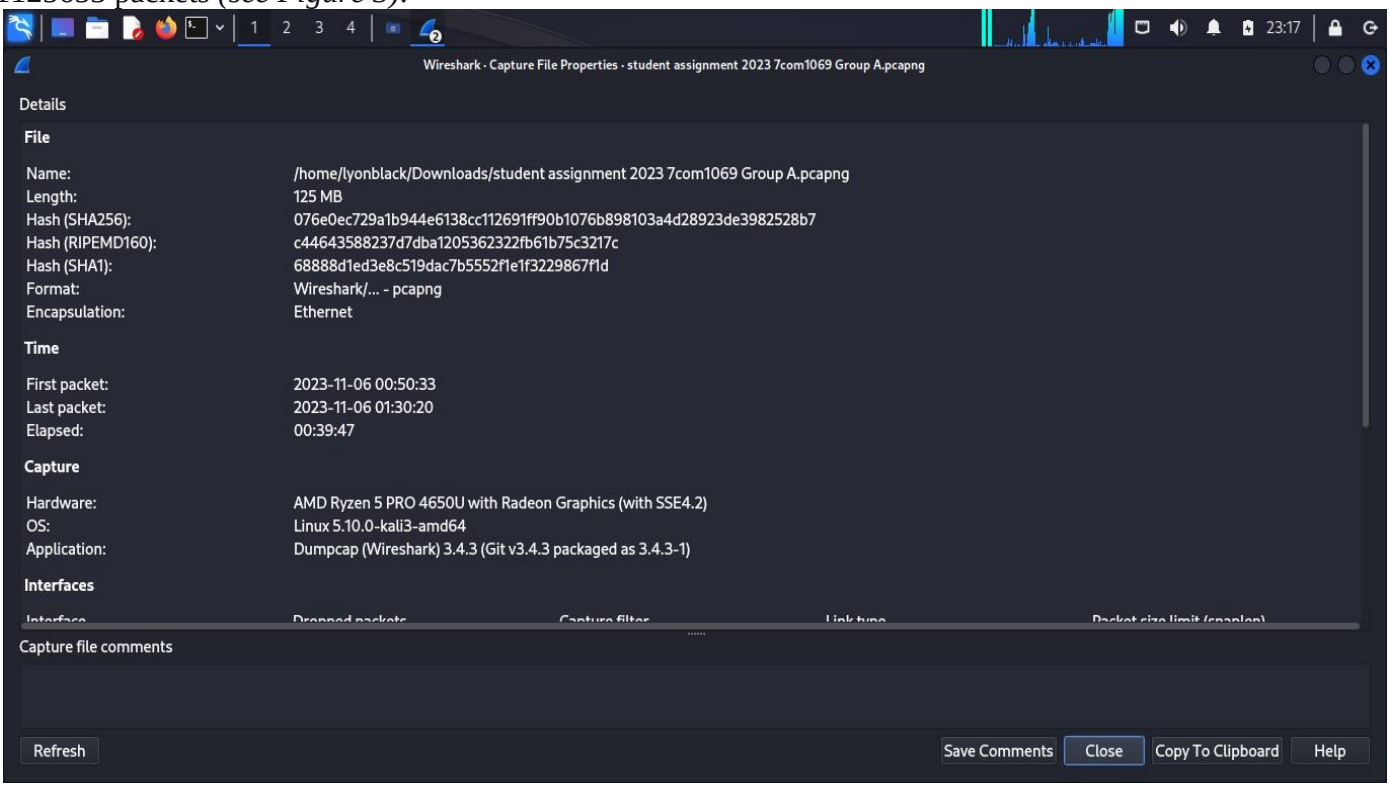
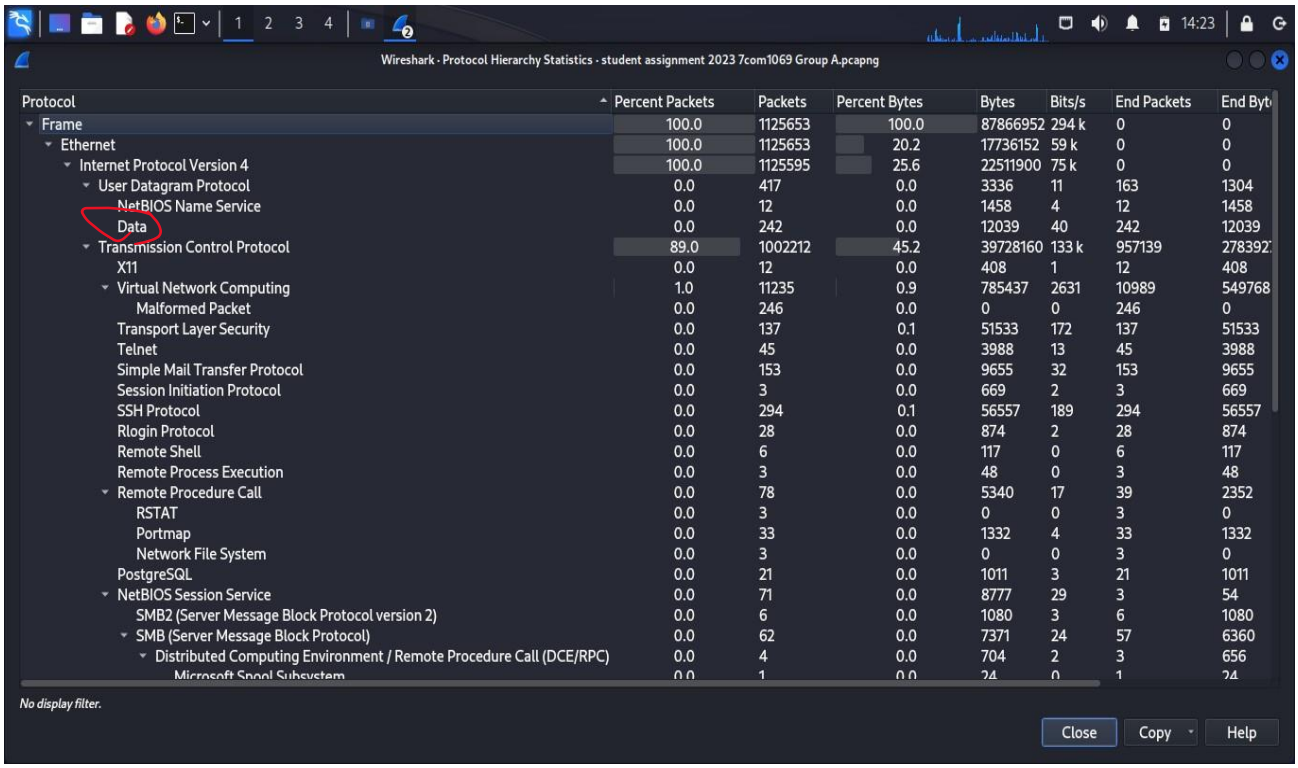


Figure 3. Image showing PCAP properties

3.2. Protocol Hierarchy Statistics

Protocol Hierarchy Statistics in Wireshark show the different network protocols and their usage within captured network traffic. This feature allows users to analyse network traffic distribution based on the protocols used, providing valuable insights into the types of traffic flowing through the network.
The Protocol Hierarchy Statistics in Wireshark categories network traffic based on the protocol stack, including Ethernet, IP, TCP, UDP, HTTP, DNS, and many others. The statistics include the number of packets, bytes, and percentage of total traffic for each protocol, allowing users to identify the most prevalent protocols and potential areas of concern.
By analysing Protocol Hierarchy Statistics, users can better understand the types of traffic on their network, identify any anomalies or excessive usage of specific protocols, and troubleshoot potential network issues.
This information can also be used for capacity planning, security monitoring, and performance optimisation.
Overall, Protocol Hierarchy Statistics in Wireshark is a valuable tool for network administrators and analysts to gain insights into network traffic composition and make informed decisions about network management and optimisation.

In addition, there is suspicious data under the user datagram protocol; any attempt the network cannot recognise will automatically convert it to data *(see Figure 4)*.



Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Byt
Frame	100.0	1125653	100.0	87866952	294 k	0	0
Ethernet	100.0	1125653	20.2	17736152	59 k	0	0
Internet Protocol Version 4	100.0	1125595	25.6	22511900	75 k	0	0
User Datagram Protocol	0.0	417	0.0	3336	11	163	1304
NetBIOS Name Service	0.0	12	0.0	1458	4	12	1458
Data	0.0	242	0.0	12039	40	242	12039
Transmission Control Protocol	89.0	1002212	45.2	39728160	133 k	957139	278392
X11	0.0	12	0.0	408	1	12	408
Virtual Network Computing	1.0	11235	0.9	785437	2631	10989	549768
Malformed Packet	0.0	246	0.0	0	0	246	0
Transport Layer Security	0.0	137	0.1	51533	172	137	51533
Telnet	0.0	45	0.0	3988	13	45	3988
Simple Mail Transfer Protocol	0.0	153	0.0	9655	32	153	9655
Session Initiation Protocol	0.0	3	0.0	669	2	3	669
SSH Protocol	0.0	294	0.1	56557	189	294	56557
Rlogin Protocol	0.0	28	0.0	874	2	28	874
Remote Shell	0.0	6	0.0	117	0	6	117
Remote Process Execution	0.0	3	0.0	48	0	3	48
Remote Procedure Call	0.0	78	0.0	5340	17	39	2352
RSTAT	0.0	3	0.0	0	0	3	0
Portmap	0.0	33	0.0	1332	4	33	1332
Network File System	0.0	3	0.0	0	0	3	0
PostgreSQL	0.0	21	0.0	1011	3	21	1011
NetBIOS Session Service	0.0	71	0.0	8777	29	3	54
SMB2 (Server Message Block Protocol version 2)	0.0	6	0.0	1080	3	6	1080
SMB (Server Message Block Protocol)	0.0	62	0.0	7371	24	57	6360
Distributed Computing Environment / Remote Procedure Call (DCE/RPC)	0.0	4	0.0	704	2	3	656
Microsoft Spool Subsystem	0.0	1	0.0	24	0	1	24

Figure 4. Image showing Protocol Hierarchy Statistics

3.3. Endpoint

According to a packet conversation analysis, the communication traffic associated with the IP address 192.168.1.200 is considerable. The IP address 146.90.197.173 is related to the traffic *(see Figure 5)*. High traffic is not cause for alarm on its own because it might suggest that one of the hosts is a server and the other is a client. Despite this, the high volume of communication identified between the two devices raises the possibility of an attack.

The endpoint is a Wireshark conversation analysis tool used to investigate communication traffic; the tool discloses that the IP address 192.168.0.160 has the most talks, consistent with the IP address 192.168.0.179. Although high traffic on a single IP address might be concerning,

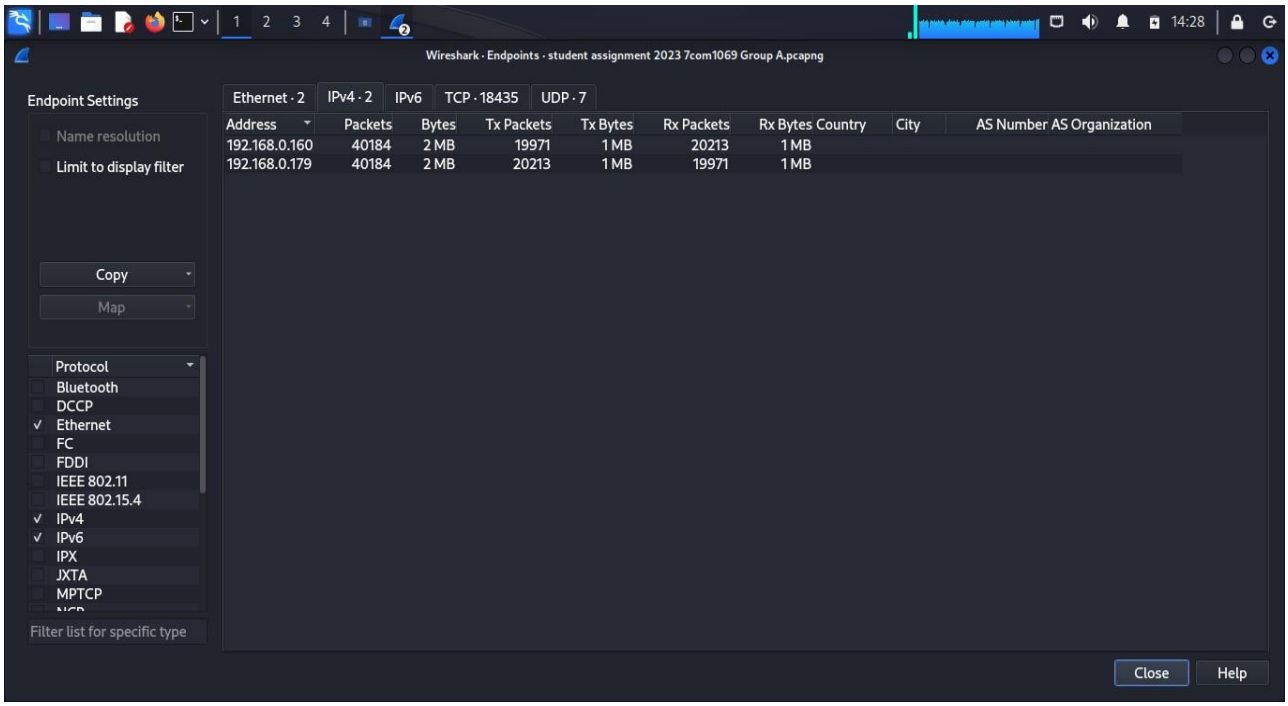


Figure 5. Image showing Endpoints

3.5. Conversation

They are investigating the interplay of the IP numbers 192.168.0.179 and 19.169.0.160. It indicates that the two addresses communicate for 23885 seconds and use 2MB of data *(see Figure 6)*.

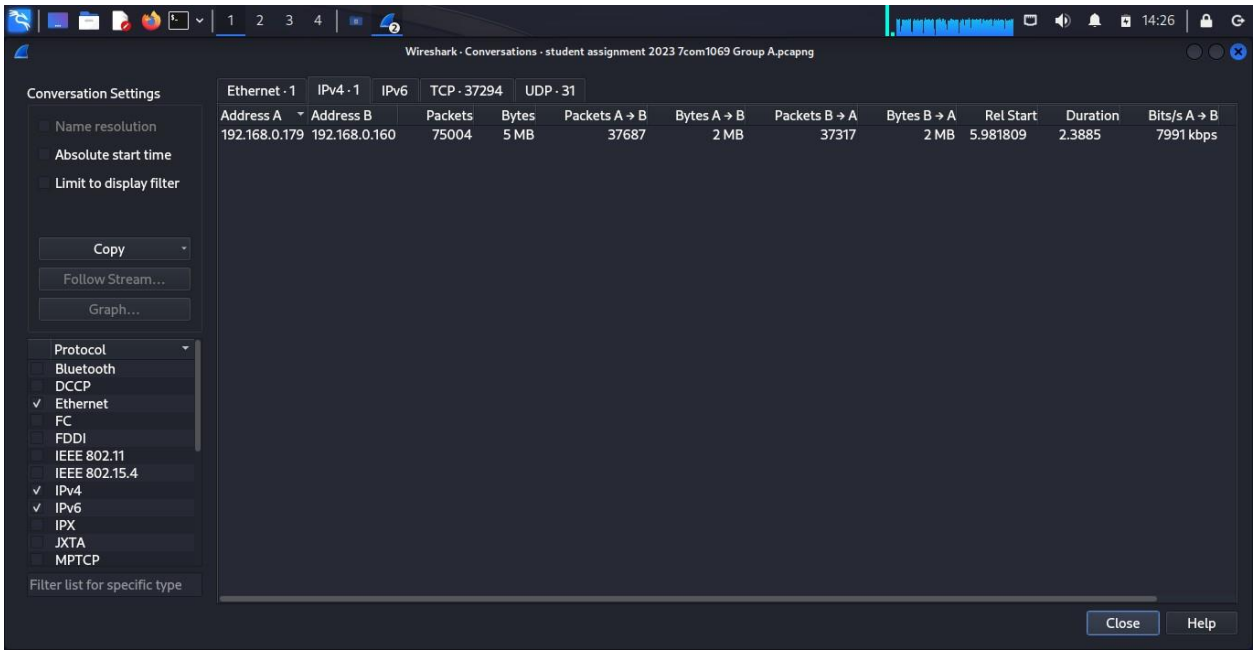


Figure 6. Image showing Conversation

3.6. I/O Graphs

The tcp packets exhibit an error *(see Figure 7)*, according to an I/O graph analysis of the device's communication. This indicates an issue with the transmission control protocol (TCP), a core communication

protocol. TCP errors can arise in various ways, including network congestion, packet loss, misconfigured network devices, connection timeout (when a connection request is not accepted within a specific interval), or software defects.

After filtering the tcp.analysis.flags, it shows that the client reused the port, which did not make the request go through.

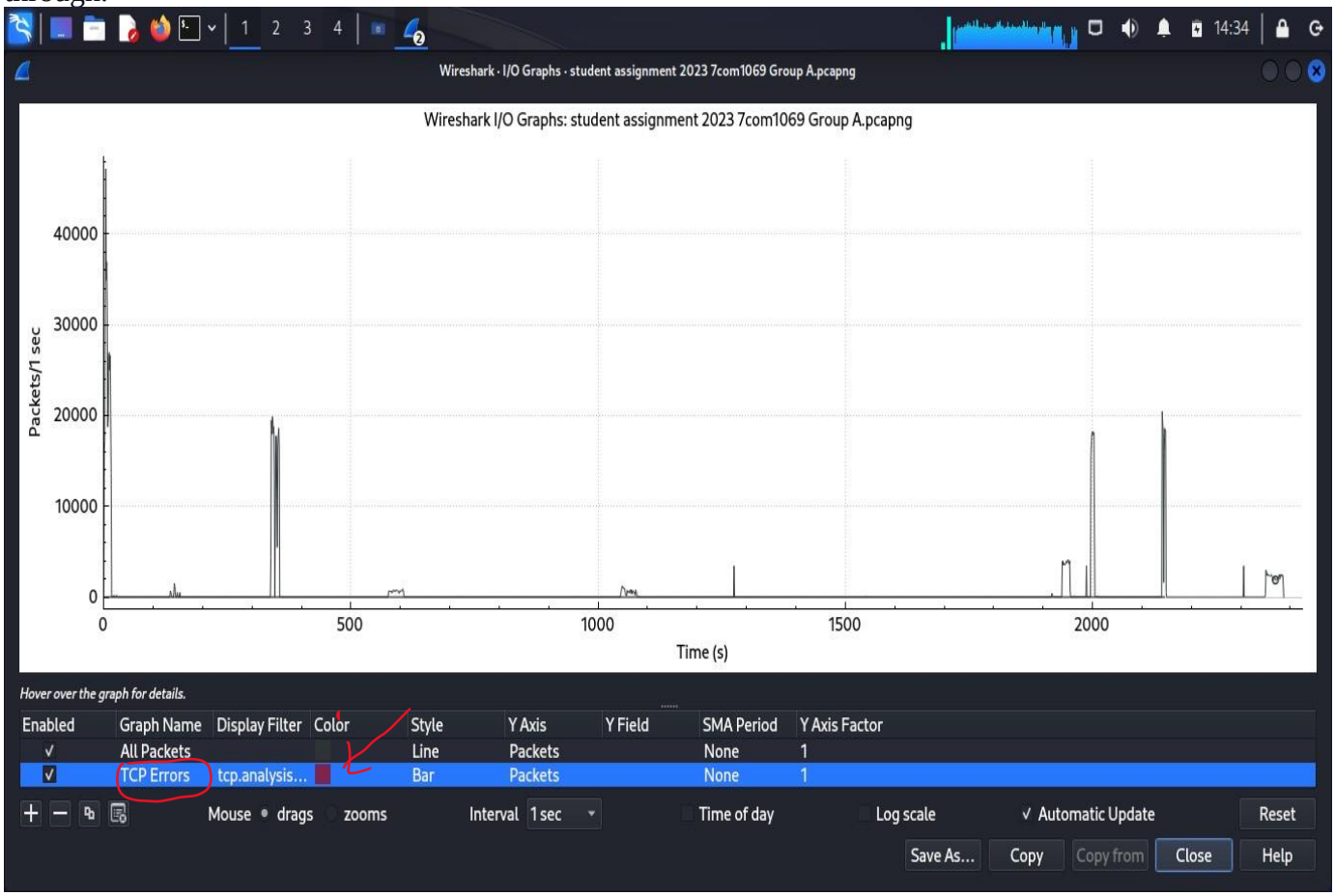


Figure 7. Image showing I/O Graphs

ACTIVITY NARRATIVE AND ATTACK IDENTIFICATION

4.1. Activity Narrative

The PCAP file properties provide comprehensive details about the captured network traffic, including source and destination IP addresses, protocol versions, duration and time range, packet size and number, and file format (see Figure 3). This information is valuable for analysis, troubleshooting, and security investigations. The Protocol Hierarchy Statistics in Wireshark categorise network traffic based on the protocols used, allowing users to identify prevalent protocols and potential areas of concern (see Figure 4). This information can be used for capacity planning, security monitoring, and performance optimisation. The Endpoint tool in Wireshark reveals significant communication traffic associated with specific IP addresses, which could indicate a potential attack (see Figure 5). The Conversation analysis tool investigates the interaction between IP addresses, providing insights into the duration and data usage of the communication (see Figure 6). Finally, the I/O Graphs tool can identify errors in TCP packets, which may indicate issues with network congestion, packet loss, misconfigured devices, or software defects (see Figure 7). These tools and analyses are essential for understanding network traffic composition, identifying anomalies, and troubleshooting potential issues.

4.2. Attack Identification

After examining specific packets, it was discovered that various attacks targeted the web server. The destination IP address was identified as VMware_13:7c:22, while the source IP address was VMware_d4:ef:d9. The communication between these two devices revealed that the source IP address (VMware_d4:ef:d9) attempted different attack techniques, including path traversal/directory traversal, local file inclusion, SQL injection, and command injection against the destination IP address (VMware_13:7c:22). These attacks resulted in the error message "404 NOT FOUND" (see figure 8,9,10, and 11).

4.2.1. Path traversal/directory traversal

Path traversal/directory traversal involves using "../../". (Awad, et al., 2022) in the input, indicating an attempt to access directories. The attacker also attempted to execute code on HOST, HASH, password, and Windows. This suggests an effort to gather information about passwords and determine the targeted system's operating system (OS) (see Figure 8).

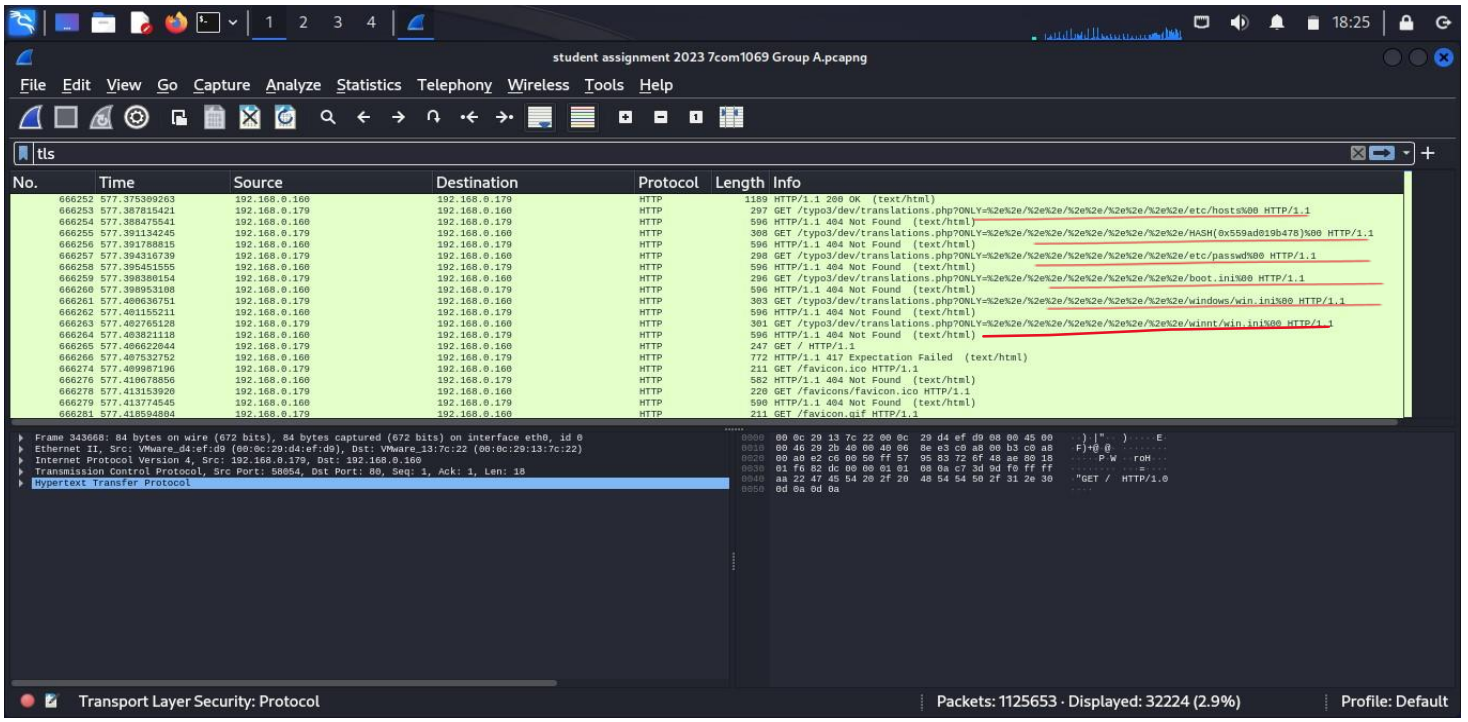


Figure 8. Image showing Path traversal (directory traversal) attack by the attacker

4.2.2. Local file inclusion

Watch out for user input that tries to include local files in the application. Pay attention to things like "../../../../../../../../etc/passwd" *(see Figure 9)* or other attempts to access sensitive system files. Keep an eye on server logs for any suspicious file inclusion attempts. **(Awad, et al., 2022)**

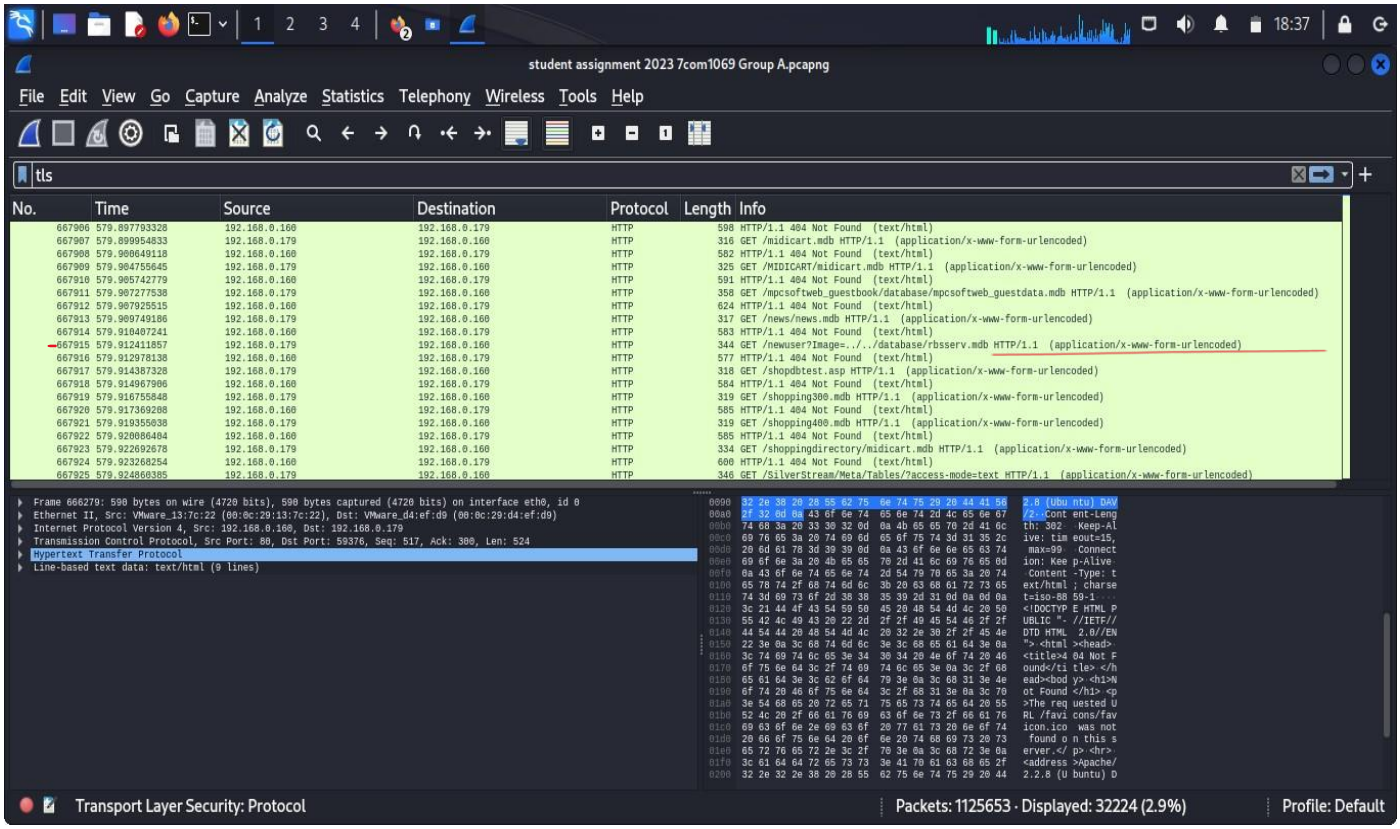


Figure 9. Image showing **Local file inclusion** *attack by the attacker*

4.2.3. Structured query language (SQL) injection

Always filter user input before using it in Structured query language (SQL) queries to avoid Structured query language (SQL) injection attacks. Watch for odd trends in user input and strange requests in database logs. (Limei, et al., 2019)

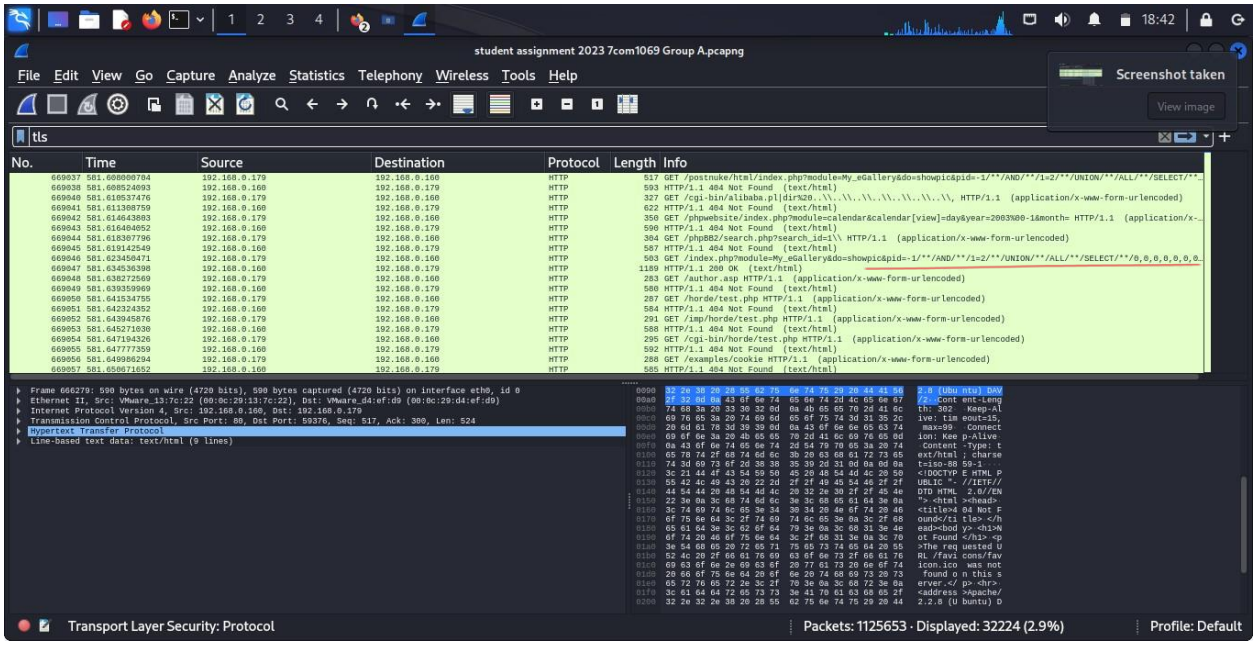


Figure 10. Image showing* SQL injection *attack by the attacker

4.2.4. Command injection

To avoid command injection attacks, it is necessary to properly check and verify user input before using it for system instructions. This involves searching for any irregular or unexpected patterns in the user input that could suggest an effort to introduce extra commands, such as using characters like " " or *";" (see Figure 11)*. Monitoring server logs for any suspicious or unusual command behaviour can also aid in detecting and preventing possible attacks. Manipulate the system and protect it from potential security threats. (Awad, et al., 2022)

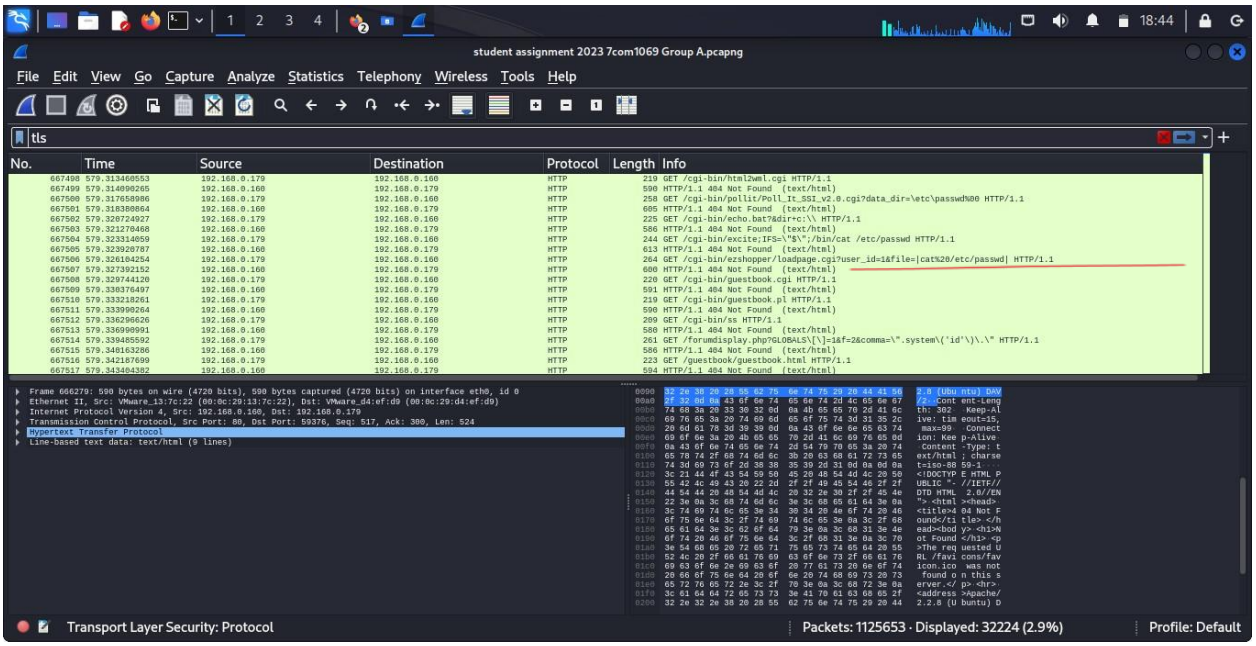


Figure 11. Image showing Command injection attack by the attacker.

The images indicate that the attacker utilised various web-based attack techniques, including path traversal/directory traversal, local file inclusion, SQL injection, and command injection. However, the attacks were unsuccessful, resulting in a "404 NOT FOUND" error message. The user agent used in the attacker's request was Mozilla/5.0 and accessed Nikto/2.1.6, a web tool designed to scan for vulnerabilities on web servers. This suggests that the attacker attempted to exploit vulnerabilities and gather information from the Server. Despite the attempts, the attacker was unable to execute the attacks successfully.

ATTACK TECHNICAL EXPLANATION

Path traversal: Path traversal (directory traversal) attack:
The attacker can access files and directories outside the web server's root directory to use web service input parameters *(See Figure 8)*. This exploits flaws in the web service's input validation and sanitisation procedures, allowing them to access sensitive files on the Server's operating system (OS). This attack's known vulnerabilities include **CVE-2023-6577**.

Local file inclusion (LFI) attack:
The attacker may exploit a web application by including local files, such as configuration files, on the web server. *(see figure 9)* They accomplish this by using the web service's poor input validation and sanitisation, which allows them to include and run local files on the Server. **CVE-2023-4591** is linked to this type of attack.

Structured query language (SQL) injection attack:
The attacker can compromise a web application by inserting malicious Structured query language (SQL) code into the input boxes, causing the database to malfunction *(See Figure 10)*. They do this because the online service needs to adequately validate the input, allowing them to run SQL statements that they should not be able to. **CVE2023-6655** has similar issues.

Command injection attack:
The attacker can compromise a web application by entering malicious commands into the input fields, which can disrupt the Server's operating system (OS) *(See Figure 11)*. They do this because the web service does not adequately validate the input, allowing them to run commands they should not be able to. **CVE-2023-4746** is a similar issue.

A path traversal attack, or a directory traversal attack, tries to access files and directories not in the web server's root directory. Figure 8 shows a web service receiving the argument "file path," which corresponds to the name of a file in the Server's directory. Attackers can utilise this option to access files on the system, which might lead to unauthorised access to sensitive data. Attackers might get access to arbitrary files and directories stored on the machine's operating system (OS) by modifying the file path variable to include sequences like "../../" or by using absolute file paths. (Awad, et al., 2022)
In the operating system (OS), an arbitrary command is sent by command injection attacks through a web application or vulnerable program. This occurs when the application sends secure user-supplied data to a particular system.
The attacker's OS instructions are routinely performed with all critical privileges granted to the vulnerable application. (Stasinopoulos, et al., 2020)
Structured query language **(**SQL) injection attacks work because the most different methodists perform, give calculations, and dynamically build code run by another system or component in diverse applications. (Justin, 2009)

ATTACK MITIGATION STRATEGY ATTACK MITIGATION

6.1. Attack Mitigation

Path traversal:
All user input, particularly file paths, must be validated and sanitised to prevent path traversal threats. Validate user input to guarantee that it cannot be used to navigate outside the desired directory structure.
Consider utilising an allowlist technique to restrict access to specified, recognised file paths. *(see figure 13)*
Local file inclusion:
Avoid leveraging user input to directly include files in the program to avoid local file inclusion attacks. Instead, utilise secure ways for having files, such as absolute file paths or preset inclusions. Implement effective input validation and sanitise user input to avoid introducing unauthorised files. *(see figure 14)*
SQL injection:
When communicating with the database, utilise parameterised queries or prepared statements to avoid SQL injection attacks. This can aid in the prevention of malicious SQL code being inserted into user input. Input validation and sanitisation should also be implemented to guarantee that user input does not include any dangerous SQL code. *(see figure 15).*
Command injection:
Avoid utilising user input to execute system instructions directly to combat command injection attacks. Instead, employ secure mechanisms for command execution, such as parameterised commands or preconfigured instructions. Implement adequate input validation and sanitisation to prevent unauthorised instructions from being executed. Consider adding tight access restrictions to limit attackers' ability to conduct damaging operations.

6.2. Defence-In-Depth Strategy

A defence-in-depth approach entails implementing several security measures to protect against various threats. This may be applicable in the context of web application security:
To discover and correct vulnerabilities, utilise secure coding techniques and conduct frequent code reviews.
Web application firewalls (WAFs) should be used to filter and monitor incoming traffic for possible threats.
Adopt network-level security solutions such as intrusion detection or prevention systems (IDS/IPS) to detect and prevent malicious behaviour.
Routine security assessments and penetration testing to identify and address vulnerabilities in the online application.
It is critical to emphasise that while these mitigation methods are effective, they should be used in tandem to form a complete defence-in-depth strategy. Security measures must be monitored, tested, and updated regularly.

6.3. Write Bash Scripts

```
#!/bin/bash

# Check if the script runs as an admin/root

if [ "(id -u)" != "0" ]; then

    echo "This script must be run as root"

    exit 1

fi

# Extract IP address of the attacker from the provided pcap

file
```

```
attacker_ip=(tcpdump -r file.pcap | awk 'print 3' | sort | uniq |
grep -Eo '([0-9]1,3)3[0-9]1,3')
# Check if the server is up and running
ping -c 1victim_ip > /dev/null
if [ ? -eq 0 ]; then
    # Temporarily disconnect the server from the network
    ifconfig eth0 down
    echo "Server isolated. Investigating the attack..."
else
    echo "Server is already down"
fi
# Blacklist the attacker's IP address
iptables -A INPUT -sattacker_ip -j DROP
echo "Attacker's IP address has been dropped"
# Check if the attacker's IP address has been blacklisted
if iptables -C INPUT -s attacker_ip -j DROP; then
    # Reconnect the server to the network
    ifconfig eth0 up
    echo "Connection reinstated"
fi
 #(see figure 12).
```

CONCLUSION

In conclusion, web application developers and security teams must proactively implement robust mitigation strategies to protect against common attack vectors such as path traversal, local file inclusion, SQL injection, and command injection. By incorporating input validation, access controls, secure coding practices, and network-level security measures, organisations can create a defence-in-depth strategy that provides multiple layers of protection. Regular monitoring, testing, and updating security measures are also essential to stay ahead of evolving threats and ensure the ongoing security of web applications.

References

Awad, A. A., Shailendra, M. & Mohammed, A., 2022. Web Security: Emerging Threats and Defense. *Computer Systems Science & Engineering,* Volume 40, p. 3.

Justin, C.-S., 2009. *SQL injection attacks and defence.* s.l.:Elsevier.

Limei, M., Dongmei, Z., Gao, Y. & Zhao, C., 2019. Research on SQL injection attack and prevention technology based on the web. In: *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA).* s.l.:IEEE, pp. 176--179.

Mirko, S., Outi-Marja, L. & Szanto, A., 2020. Cyber threat actors for the factory of the future. *Applied Sciences,* Volume 10, p. 4334.

Richard, B., 2001. Hackers profiled—who are they, and what are their motivations? *Computer Fraud & Security,* Volume 2001, pp. pp. 14-17.

Stasinopoulos, A., Christoforos, N. & Christos, X., 2020. Detection of malicious command injection attacks on phase shifter control in power systems. *IEEE Transactions on Power Systems,* Volume 36, pp. 271--280.

Appendix I

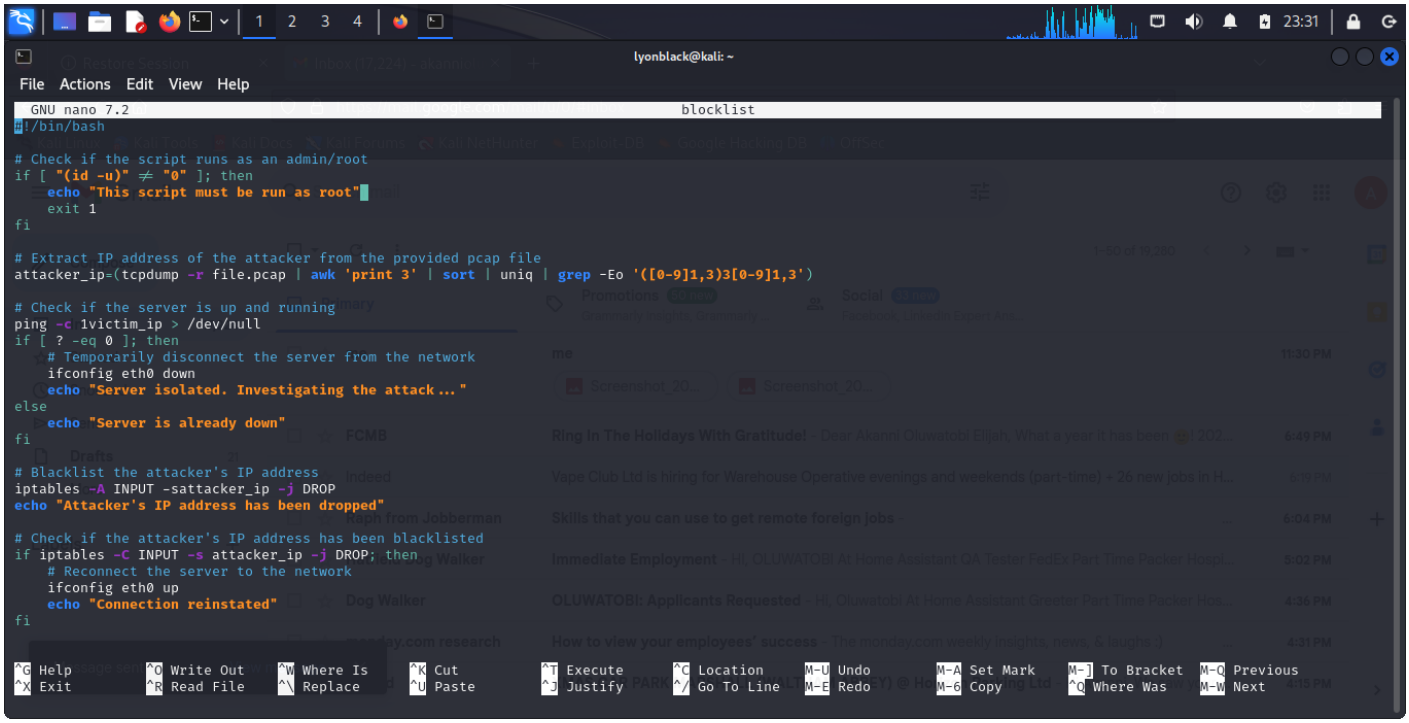


Figure 12. Image showing Bash script command for blacklisting and blocking the attacker's IP

This bash script is designed to be run with administrative privileges and performs actions to isolate and block an attacker's IP address from a server.

Appendix II

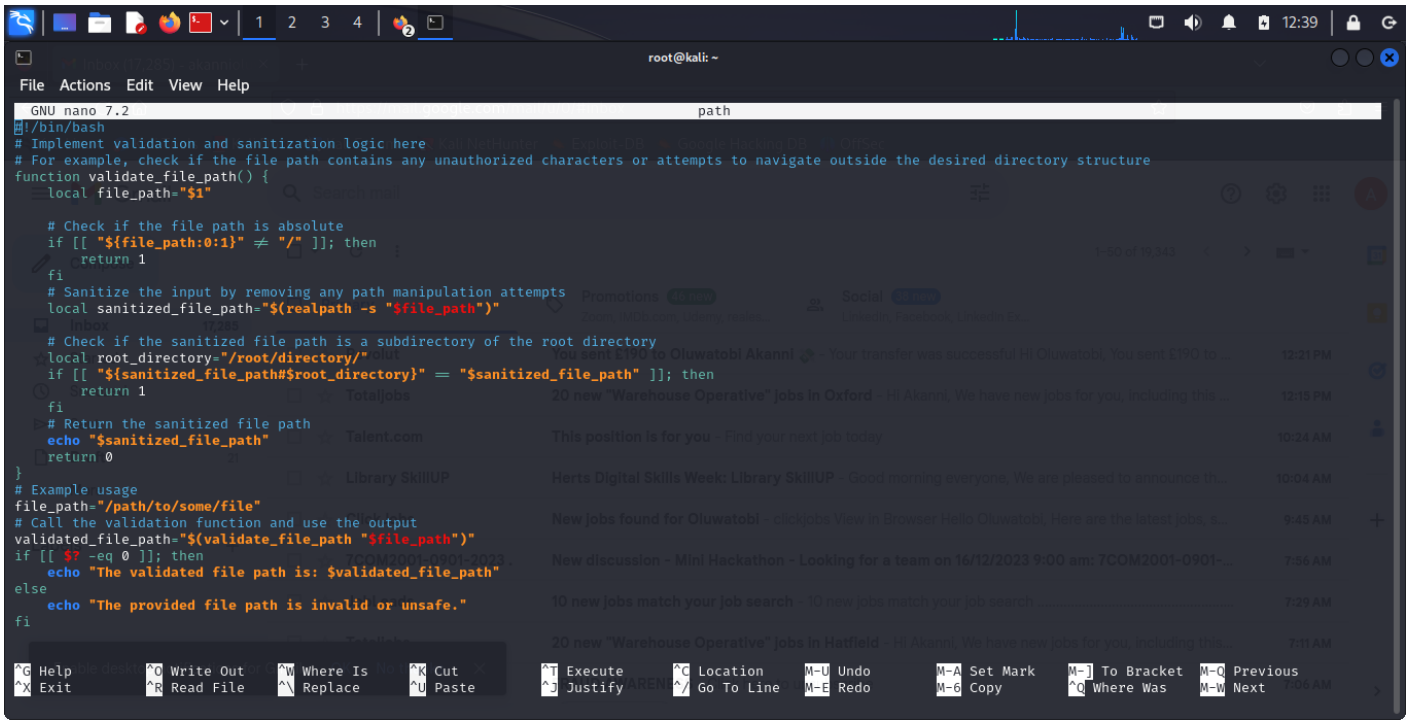


Figure 13. Image showing Bash script command mitigating Path traversal

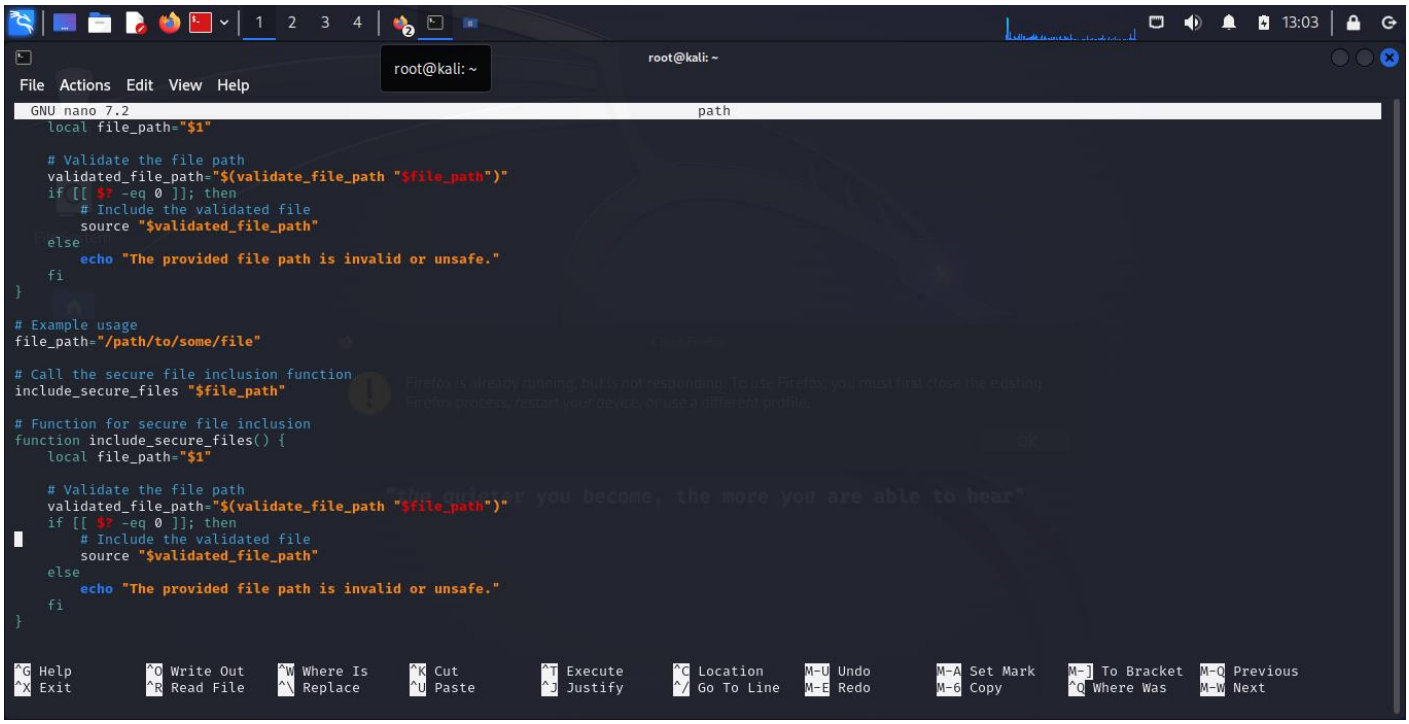


Figure 14. Image showing Bash script command mitigating Local file inclusion (LFI) attack

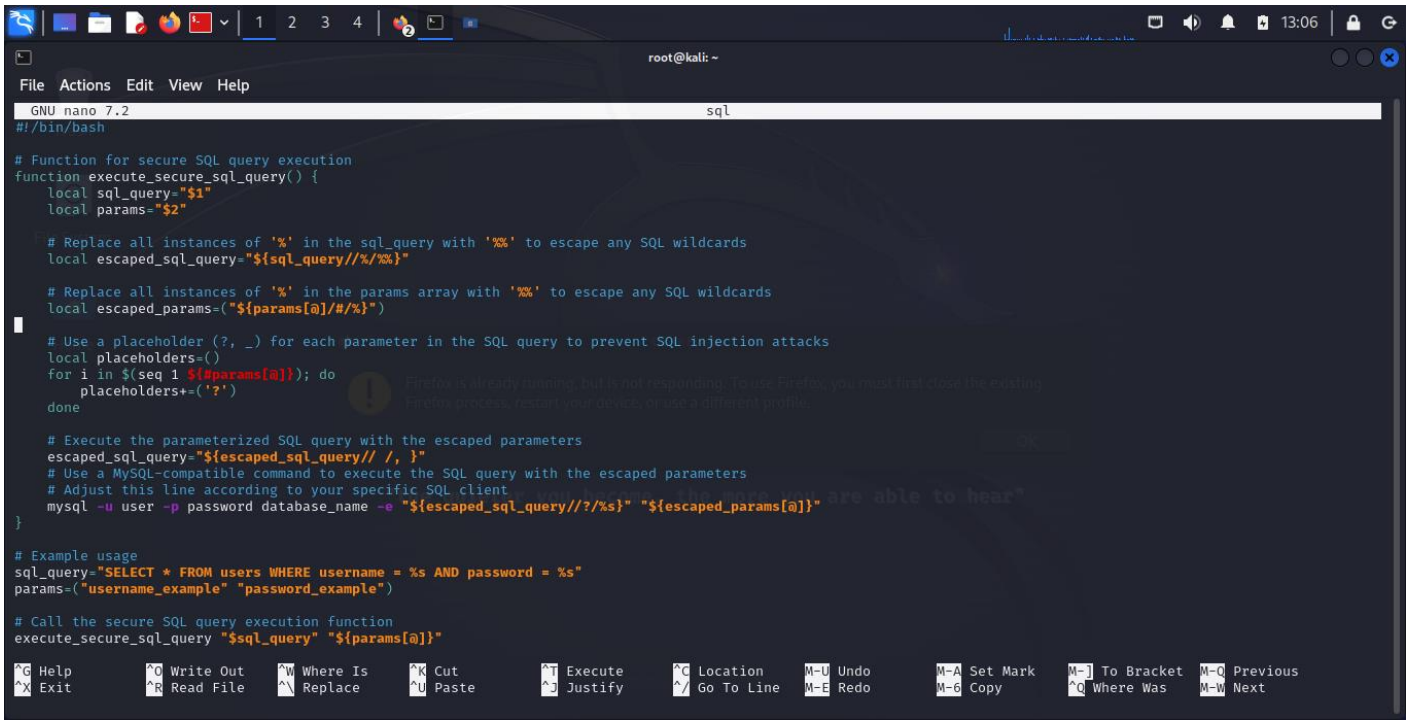


Figure 15. Image showing Bash script command mitigating Command injection