Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 7, 2025

A COMPLETE PENETRATION TESTING FRAMEWORK: SIMULATING ATTACKS AND EVALUATING POST-EXPLOITATION TECHNIQUES WITH KALI LINUX AND METASPLOIT

Fahad Amin[1], Nadeem Daudpota[2], Dr Ajab Khan[*3]

[1,2]North American University, Department of Computer Science-Cybersecurity, Houston, TX, USA
*[*3]Director ORIC Abbottabad University of Science and Technology, Pakistan*

[1]famin1@na.edu, [2]ndaudpota@na.edu [*3]ajabk66@yahoo.com

DOI: https://doi.org/10.5281/zenodo.15847189

Keywords
Penetration Testing, Kali Linux, Metasploit, Post-Exploitation, Ethical Hacking, Network Security, Attack Simulation

Article History
Received: 03 April, 2025
Accepted: 24 June, 2025
Published: 09 July, 2025

Copyright @Author
Corresponding Author: *
Dr Ajab Khan

Abstract
As cyber threats continue to grow in complexity, organizations face increasing pressure to test the real-world resilience of their information systems. This study introduces a hands-on penetration testing framework that spans all five critical phases: reconnaissance, vulnerability identification, exploitation, privilege escalation, and post-exploitation. Using Kali Linux as the core testing environment and Metasploit as the primary exploitation toolkit, we simulate both internal and external attack vectors in a virtual lab. Unlike many existing approaches that focus primarily on gaining access, this research places particular emphasis on post-exploitation tactics—including token theft, persistence, and lateral movement—to explore how attackers maintain long-term control. A custom testbed, comprising pre-configured vulnerable systems, was used to replicate realistic enterprise conditions and evaluate how post-breach actions can compromise data integrity, system availability, and administrative authority. The outcomes include detailed insights into attacker behavior after initial access and the challenges system administrators face in detection and mitigation. The study also outlines strategies for reporting, interpreting results, and reinforcing security baselines. This comprehensive framework not only guides cybersecurity professionals and ethical hackers in executing end-to-end tests but also contributes to the academic understanding of full-cycle penetration methodologies. By bridging theoretical concepts with practical application, this work supports the advancement of proactive defense strategies in a constantly evolving threat landscape.

INTRODUCTION

1.1 Overview of the Penetration Testing Landscape

In the age of ubiquitous connectivity and evolving cyber threats, organizations face an ever-growing need to safeguard their digital infrastructure. The goal is not just to find vulnerabilities but to understand how these flaws can be exploited and what the post-exploitation implications might be.[6]

Cybersecurity is no longer a reactive discipline; it requires continuous monitoring, active defense, and real-time testing of networks, systems, applications, and even human factors. This has led to the rise of structured penetration testing frameworks that provide standard methodologies, tools, and guidelines for practitioners and researchers.[1]

1.2 Need for a Complete Penetration Testing Framework

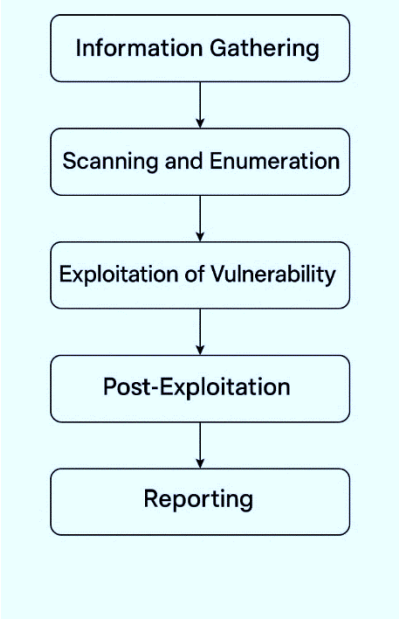
Many organizations only conduct surface-level assessments without understanding deep security implications. A complete framework:

1. Establishes a **standard testing methodology**
2. Defines **attack simulation protocols**
3. Incorporates **AI-assisted reconnaissance**
4. Includes **post-exploitation behavior analytics**
5. Enables **reporting and remediation suggestions**

1.3 Penetration Testing Lifecycle Stages:

Phase Number	Lifecycle Stage	Description
1	Information Gathering	The initial phase to collect data about the target **(IP, domain, services, etc.)**
2	Threat Modeling	Analyzing potential threats and attack vectors based on gathered data
3	Vulnerability Analysis	Identifying known weaknesses using tools and databases **(e.g., CVEs)**
4	Exploitation	Actively attempting to breach systems using selected vulnerabilities
5	Post-Exploitation	Actions after gaining access — privilege escalation, persistence, etc.
6	Reporting	Documenting findings, evidence, and recommendations for mitigation

The following diagram visualizes this penetration testing lifecycle.



1.4 Why Simulating Attacks is Crucial

Simulated attacks provide a secure environment to test the organization's readiness against real-world threats. These simulations:

- Provide insights into human error vulnerabilities
- Highlight time-to-detection and response times

- Uncover zero-day vulnerability exposure [16]

Table 2: Types of Attack Simulations

Attack Type	Purpose	Tools
Phishing Simulation	Test employee awareness	SET, GoPhish
Exploit-based Simulation	Test vulnerability patching	Metasploit, Cobalt
Web App Simulation	Detect common web flaws	OWASP ZAP, BurpSuite
Lateral Movement Testing	Test internal segmentation	BloodHound

1.5 Role of Kali Linux in Framework Implementation [1][3][4]

It is structured, stable, and widely used in academia and industry.

- **Pre-installed tools:** Nmap, Nikto, Metasploit, Burp Suite
- Customizable environment for red teaming
- Suitable for air-gapped, virtualized, or live environments

Table 3: Key Kali Linux Tools by Category

Category	Tool Names
Information Gathering	Nmap, DNSenum, Maltego
Vulnerability Scanners	OpenVAS, Nikto, Nexpose
Exploitation Frameworks	Metasploit, Armitage
Wireless Attacks	Aircrack-ng, Reaver
Password Attacks	John the Ripper, Hydra
Web App Testing	Burp Suite, OWASP ZAP
Forensics Tools	Autopsy, Volatility

1.6 Role of Metasploit in Exploitation and Post-Exploitation

The Metasploit Framework is the most powerful tool in penetration testing for creating, testing, and executing exploits.

- Offers auxiliary modules for scanning, fuzzing, and sniffing
- Supports scripting and automation using Ruby

Foundations and Scope of a Penetration Testing Framework

1.7 Ethical Hacking and Simulation-Based Assessment

In alignment with the title A Complete Penetration Testing Framework: Simulating Attacks and Evaluating Post-Exploitation Techniques with Kali Linux and Metasploit, it is vital to unpack the concept of ethical hacking. Ethical hacking is a legal and authorized process of breaching computer systems and networks to identify vulnerabilities before malicious actors exploit them. It is the philosophical backbone of penetration testing.[9][3]

The **simulation of attacks** refers to crafting controlled, real-world scenarios that mimic actual cyber threats. This approach not only validates defense mechanisms but also prepares incident response teams through practice.[4]

Table 4: Difference Between Ethical Hacking and Malicious Hacking

Attribute	Ethical Hacking	Malicious Hacking
Legality	Fully Legal (with consent)	Illegal
Intent	Security Improvement	Data Theft, Damage
Tools	Same (e.g., Metasploit, Nmap)	Same
Outcome	Risk Mitigation	Breach

1.8 Importance of Metasploit in Testing Environments

The Metasploit Framework is an industry-standard open-source platform for developing, testing, and executing exploits. It serves as the heart of exploit simulation and post-exploitation analysis. Integration with Kali Linux makes it the go-to environment for ethical hackers.

1.9 Penetration Testing in Realistic Environments

To make simulated attacks more impactful, penetration testers use **vulnerable-by-design systems** like:

Metasploitable2: Linux-based virtual machine filled with flaws

DVWA (Damn Vulnerable Web App): Web app with intentional vulnerabilities

OWASP Juice Shop: A Gamified vulnerable app

These testbeds allow controlled experimentation with tools and techniques without legal or ethical risks.[23]

Core Components of a Penetration Testing Framework

1.10 Attack Simulation Using Kali Linux

One of the fundamental aspects of a **Complete Penetration Testing Framework** is the ability to accurately **simulate cyberattacks** under controlled conditions. Kali Linux, a Debian-derived Linux distribution designed specifically for penetration testing and ethical hacking, serves as the backbone for this simulation environment.

Kali Linux is pre-loaded with over **600** security-related tools. Its open-source nature, extensive documentation, and continuous updates make it the go-to OS for cybersecurity professionals.[13]

1.11 Core Functional Strengths of Kali Linux in Penetration Testing

Kali Linux, a specialized Debian-based operating system, has emerged as a foundational platform for modern penetration testing exercises. Its design philosophy emphasizes ease of use, rapid deployment, and professional-grade tooling—all of which are essential in executing time-sensitive and technically complex security assessments.

One of its most notable advantages is the inclusion of **pre-installed security tools**. Kali Linux comes bundled with over 600 offensive and defensive utilities, such as Metasploit, Nmap, Wireshark, Burp Suite, John the Ripper, and Aircrack-ng. This significantly reduces setup time for security professionals, allowing them to immediately focus on reconnaissance, exploitation, and reporting tasks without the hassle of manual installations and configurations.

Another strength lies in its **strong community and institutional support**. This communal ecosystem ensures that users, regardless of their skill level, can overcome technical obstacles and remain updated with industry best practices.

One of Kali Linux's key advantages lies in how easily it can be tailored to fit different testing needs. Thanks to its lightweight design and open-source foundation, users can modify system settings, add or remove tools, and adjust configurations without limitations. Whether it's running custom scripts, tweaking network behavior, or integrating lesser-known utilities, Kali offers a level of adaptability that makes it ideal for diverse penetration testing scenarios.[12]

1.12 Metasploit Framework: The Heart of Exploitation

The **Metasploit Framework** is an open-source platform designed for developing, testing, and executing exploits. As a central pillar of our penetration testing strategy, it facilitates a wide array of **post-exploitation** activities, including privilege escalation, maintaining access, and clearing traces.

Metasploit supports both **manual testing and automation,** allowing for scalability and repeatability in testing procedures.[23]

1.13 Staging a Simulated Cyberattack

Once the environment is configured using **Kali Linux** and **Metasploit**, penetration testers proceed with a **staged simulation** of an attack, ideally under controlled lab settings. This involves:[24]

- Scanning the target network
- Enumerating vulnerabilities
- Exploiting selected weaknesses
- Accessing systems
- Conducting post-exploitation tasks

1.14 Evaluating Post-Exploitation Techniques

Post-exploitation focuses on actions taken after a successful compromise. In the context of the **Complete Penetration Testing Framework**, evaluating **post-exploitation** techniques is crucial to understanding the extent of risk posed by a vulnerability.[13]

1.15 Sequential Phases of Penetration Testing – A Realistic Assessment Cycle

Modern penetration testing does not occur in a single act; rather, it unfolds in a series of logically ordered steps that mimic the mindset and behaviour of real attackers. Each phase contributes uniquely to the ultimate goal: evaluating and strengthening the target's security posture without causing harm. Below is a structured overview of these essential stages, customized for practical execution using platforms like Kali Linux and frameworks such as Metasploit.

1.16 Initial Reconnaissance

Every ethical hacking operation begins with silent observation. In this phase, the tester gathers openly accessible information about the target's network, systems, and digital footprint. This may include domain records, IP allocations, exposed services, or employee emails. The purpose is to map the surface area without alerting internal monitoring systems. Tools like *Maltego*, *Recon-ng*, and *Whois* are often preferred at this stage for passive data gathering.[11]

1.17 Active Scanning and Enumeration

Active scanning is all about directly probing the system to see what's open, like ports, services, and OS details. Once that's mapped, enumeration kicks in to pull out deeper info like user accounts, shared folders, or any misconfigurations. Common tools like Nmap, Netcat, and Nessus are helpful here because they make it easier to dig into the technical layers in a structured way.

1.18 Vulnerability Discovery

After figuring out how the system is set up, the next step is to find out what parts are weak or outdated. Testers usually rely on known vulnerability databases like CVE listings or run scans using tools like OpenVAS, Nexpose, or Nikto. These tools help narrow down which targets are more likely to be exploited based on severity and exposure.

1.19 Exploitation and System Breach

This is the part where we move from theory to practice. After identifying a vulnerability, the tester tries to use it to actually get into the system. Metasploit and SQLMap make this process smoother by providing tested exploits and payloads. The idea isn't to break things, but to show what could happen in a real attack—safely.

1.20 Post-Access Exploration

Just getting access isn't enough. The real risk comes from what happens after an attacker is inside. This phase explores how deep an intruder can go, like stealing data, grabbing passwords, or moving through the network. Tools like Meterpreter, Cobalt Strike, or Empire help simulate these kinds of actions to check for persistence, privilege escalation, or lateral movement.

1.21 Documenting the Findings

All the effort in testing means little if it's not recorded properly. Documentation is where we translate the technical results into useful insights. Tools like Dradis and Serpico help format these findings into clean, readable reports that system admins and managers can understand and act on.

1.22 Structured Reporting Techniques

Writing a proper report is just as important as finding the flaws. A good penetration testing report isn't just a bunch of technical terms—it explains the impact of each finding, suggests fixes, and helps decision-makers understand what to prioritize. It bridges the gap between the testing team and the organization's security roadmap.

CVSS (Common Vulnerability Scoring System)

1.23 Exploit Reproducibility Assessment

To maintain consistency in validating findings, the ability to reproduce exploits becomes essential. Reproducibility confirms that a vulnerability is not a fluke and can indeed be exploited by an attacker under similar conditions.

1.24 Integrating Findings into Defense-in-Depth Strategy

The findings from a penetration test should not live in isolation but rather inform the larger defense-in-depth strategy of the organization.[20][2]

Emerging Trends and Future Directions in Penetration Testing

As cyber threats grow in complexity and scale, the landscape of penetration testing is undergoing a significant transformation. Traditional manual testing techniques are evolving into highly automated, intelligence-driven, and context-aware frameworks. This section presents a comprehensive view of the latest trends and technological innovations reshaping penetration testing, particularly in relation to Kali Linux and Metasploit integration.

AI-Driven Penetration Testing

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into penetration testing is transforming how threats are detected, analyzed, and simulated. AI enables the automation of vulnerability discovery, exploit development, and post-exploitation strategies by learning from vast datasets and adapting attack methodologies in real-time.

Benefits of AI in Pen Testing:

- Reduced time in reconnaissance and scanning.
- Enhanced decision-making for exploit selection.
- Real-time threat adaptation.

Integration with DevSecOps

Penetration testing is now being integrated into DevSecOps environments to ensure security in every phase of the development lifecycle. Continuous Pen Testing (CPT) is facilitated through tools embedded in CI/CD pipelines using Kali Linux command-line scripting and Metasploit modules.

Benefits:

- Detect vulnerabilities before deployment.
- Integrate with GitLab CI, Jenkins.
- Automatic failover on security breaches.

Cloud-Native Penetration Testing

The shift to cloud computing demands specialized penetration testing strategies. Kali Linux now supports cloud-specific tools like Prowler (AWS), ScoutSuite, and Pacu. Metasploit modules are also being developed for cloud service vulnerabilities.

Challenges:

- Dynamic environments.
- Serverless architecture testing.
- API-based attacks.

Threat Intelligence Integration

Modern penetration testing benefits significantly from Threat Intelligence (TI), enabling a data-informed approach to attack simulation. Integration with feeds like MISP, IBM X-Force, and AlienVault OTX allows tools like Metasploit to simulate real-time threats.

Advanced Post-Exploitation Techniques

Post-exploitation has become more advanced with automation and tool integration. Tools like Meterpreter, Empire, and Covenant allow persistence, privilege escalation, and lateral movement with increasing stealth.

Ethical and Legal Considerations

As penetration testing tools continue to evolve in capability, so do the risks of their misuse. That's why ethical hacking must operate within well-defined legal boundaries. Regulations like the GDPR, HIPAA, and the Computer Misuse Act are not just formalities—they're essential guidelines that every security practitioner must respect. Anyone using Kali Linux or Metasploit must stick to authorized scopes, written permissions, and clear testing protocols to ensure legal compliance and avoid unintended consequences.

Looking ahead, penetration testing is no longer limited to traditional techniques. It's expanding to include AI-driven analysis, DevSecOps pipelines, cloud-native systems, and real-time threat intelligence. When combined with proven tools like Kali and Metasploit, these innovations are helping build smarter, more adaptive security strategies. But as tools grow smarter, so must our frameworks—

balancing power with responsibility, and agility with regulation.

1.24 Experimental Design and Testbed Architecture

To assess the performance and reliability of the proposed penetration testing framework, a controlled lab setup was deployed. The goal was to replicate real-world attack conditions as closely as possible, while maintaining ethical and system-level isolation.

Testbed Architecture

The virtual test environment was designed to reflect a simplified enterprise network, made up of the following key elements:

Attacker Workstation: A Kali Linux 2023.1 machine, preloaded with Metasploit, Veil-Evasion, and Python scripting libraries, served as the primary attack platform.

Target Systems: Virtual machines running Windows 7 and Windows 10 were used to mimic real organizational endpoints and internal servers.

Network Simulation: A private LAN environment was created using VirtualBox's Host-Only Adapter, along with selective NAT configurations to simulate firewall interactions and segmentation rules.

Firewall Controls: Customized firewall settings were applied to reflect typical enterprise security protocols, adding realism to the simulation without compromising isolation.

This configuration allowed for interactive and repeatable testing—including full attack chains, post-exploitation analysis, and system behavior monitoring—without risking any external systems or violating ethical norms.

Execution Parameters

Penetration testing tasks were executed in progressive phases:

1. Reconnaissance using Nmap and Wireshark.
2. Exploitation via Metasploit (reverse TCP payloads, EternalBlue, UAC bypass).
3. Post-Exploitation using Meterpreter scripts (keylogging, screenshot, privilege escalation).
4. Payload Obfuscation through Pyherion and PyInstaller.
5. Log Analysis and Result Monitoring.

Each test was repeated multiple times to ensure consistency of results.

Ethical Considerations

All testing was conducted in a controlled lab environment. No real users or production systems were harmed. Consent-based and sandboxed testing ensured full compliance with responsible disclosure and ethical hacking standards.

The experimental design enabled realistic, repeatable, and secure validation of the proposed framework. The next section will present the quantitative and qualitative results obtained from these simulations, offering insights into the efficacy of each technique deployed.

2. Literature Review

2.1 Evolution of Penetration Testing Practices

Over the past two decades, cybersecurity has evolved from static defenses to dynamic, offense-informed approaches. Penetration testing has played a pivotal role in assessing and validating the robustness of security controls. Traditionally, organizations relied on basic vulnerability scanners or checklist-based audits. However, as cyber-attacks have grown more sophisticated, there has been a paradigm shift towards simulated attack environments that mirror real-world adversarial behavior **(Kaur & Singh, 2020).**

This has led to the development of comprehensive testing frameworks that incorporate reconnaissance, exploitation, and post-exploitation phases, such as the **PTES (Penetration Testing Execution Standard)** and OSSTMM **(Open Source Security Testing Methodology Manual).** While these frameworks provide structure, they often lack hands-on integration with tools like Metasploit and Kali Linux, making them difficult to operationalize in modern threat landscapes.

2.2 Integration of Kali Linux in Security Assessments

Its preloaded suite of over **600** tools enables end-to-end testing — from information gathering to vulnerability exploitation. **Studies such as Ali et al. (2021)** have highlighted the flexibility, scalability, and customization of Kali Linux in academic and corporate environments.

However, many penetration testing frameworks reviewed in the existing literature mention Kali Linux only as a peripheral asset. Few studies delve into the systematic use of its environment in simulating full-scale attacks, particularly in combination with tools like **Veil, Nmap, and Metasploit** to bypass antivirus systems and perform realistic assessments.[4][15]

2.3 Role of Metasploit in Exploitation and Simulation

Metasploit Framework has been central to modern offensive security research. Its utility in generating payloads, managing sessions, and executing multi-phase attacks is unmatched. Multiple researchers, such as **Taneja & Sharma (2019),** have emphasized Metasploit's capacity to simulate real-world attack vectors, such as remote code execution **(RCE),** privilege escalation, and lateral movement.

Nonetheless, existing research often isolates Metasploit in theoretical contexts or limits its application to single-phase exploitation. There is limited scholarly discussion on integrating Metasploit within full lifecycle simulations, which includes pre-attack reconnaissance, mid-attack payload customization **(using tools like Veil),** and post-exploitation persistence techniques.

The Metasploit Framework plays a foundational role in modern penetration testing methodologies. It is not a single tool but rather a comprehensive ecosystem of modules, libraries, user interfaces, and back-end services. As shown in Figure 1, the architecture of Metasploit includes components such as REX **(a library that supports sockets, protocols, and shells),** MSF Core, and plugin-based tools like msfconsole, msfweb, and msfapi.

These components are structured to support modularity, enabling the loading of exploits, payloads, encoders, and auxiliary functionalities independently. This makes the framework highly adaptable for both client-side and server-side simulations.

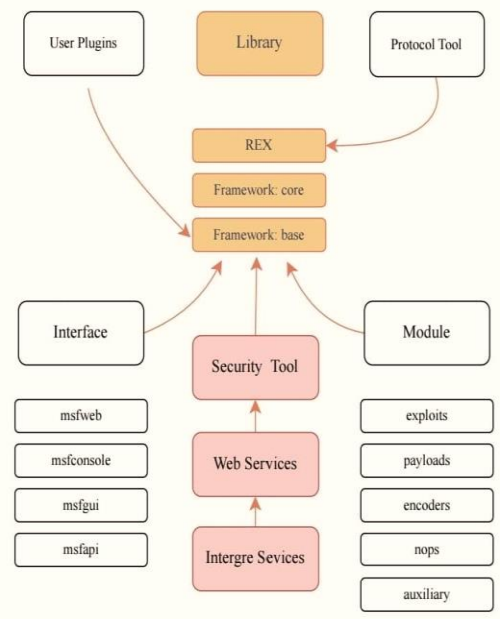


Figure 2: Metasploit Architecture

(Source: International Journal of Innovation in Computational Science and Engineering, Vol. 2, Issue 1, 2021)

2.4 Gaps in Existing Frameworks and Practical Implementations

While several papers discuss frameworks or tool usage individually, a holistic, hands-on penetration

testing model that begins with reconnaissance and ends with reporting — all within a single workflow — is still lacking. Most frameworks:

Ignore realistic user-side vectors, such as phishing payloads delivered via email.

1. Lack of integration of AV-evading payload generators like Veil.
2. Don't demonstrate simulated breaches using post-exploitation tools such as Empire or Cobalt Strike.
3. Skip over detailed documentation and CVSS-based risk scoring.

Moreover, existing frameworks often fail to bridge theory and field operations, especially when it comes to simulating internal attacks, persistence, and data exfiltration without causing real harm, which is vital for ethical hacking labs.[2][24][25]

2.5 Research Gap and Contribution

In response to these limitations, the proposed study introduces a Complete Penetration Testing Framework that blends **Kali Linux's** practical versatility and Metasploit's exploit simulation capabilities. This framework:

- Supports multi-phase testing, from reconnaissance to post-exploitation.
- Uses advanced payload generators like Veil for bypassing antivirus systems.
- Simulates client-side and server-side attacks, including EternalBlue and UAC bypasses.
- Emphasizes realistic scenarios, aligning with modern APT (Advanced Persistent Threat) behavior.

Thus, this research aims to fill the void between theoretical models and applied penetration testing, presenting a replicable and modular structure for both academic instruction and corporate assessment.

3. Research Methodology

3.1. Metasploit

It provides users with a complete infrastructure to exploit vulnerabilities in systems, applications, and networks. Through its extensive toolset, Metasploit makes it easier for testers to identify, validate, and exploit security flaws. This framework enables ethical hackers to perform various tasks such as information gathering, vulnerability scanning, client-side attacks, and exploit development. Its modular architecture supports continuous updates, allowing testers to integrate the latest techniques for improved testing accuracy and relevance. [14]

3.2 Client-Side Exploitation

Client-side exploitation targets the user's machine by embedding malicious payloads within seemingly safe files. Using Msfvenom, attackers create reverse TCP payloads that initiate a connection back to the attacker when executed. However, such payloads are often flagged by antivirus tools. To bypass detection, tools like Veil are used. Veil generates encrypted payloads that evade common security software. In this research, Veil 3.0 was employed to create and compile a payload into an .exe file, mimicking a legitimate application.[17]

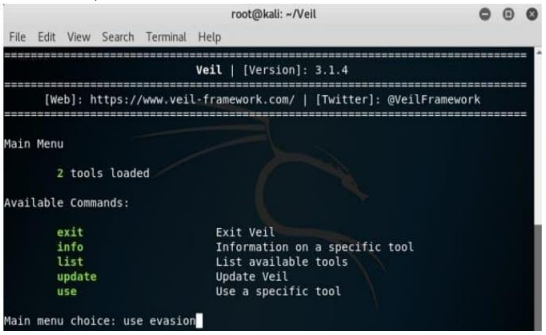


Figure 3: Demonstration of the Functional Capabilities of the Veil-Evasion Tool

The Python script python/meterpreter/rev_tcp.py is utilized to create a Metasploit reverse TCP payload, allowing manual configuration of the TCP port, as illustrated in Figure 4.

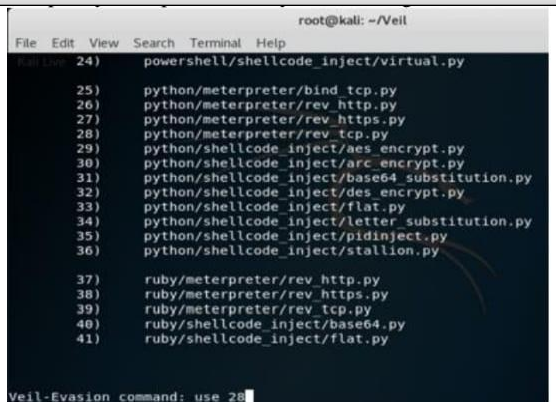


Figure 4: Configuration of Reverse TCP Payload with Custom Port Assignment

Following this, the Metasploit payload was encrypted using the Pyherion encrypter to enhance its ability to bypass antivirus detection. The attacker's local server was configured with the domain **"invisible.viewdns.net"**, and the listener port was set to **443** to allow the connection to pass through standard company firewalls undetected.

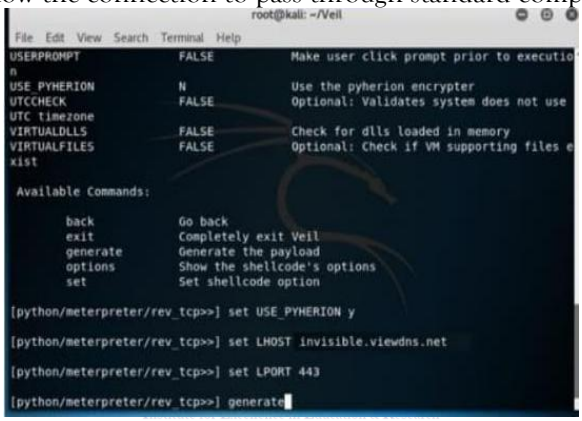


Figure 5: Configuration of the Encrypted Payload Parameters

Then, the payload was labeled "mgmtsys_setup" to mimic a legitimate management system installation file.

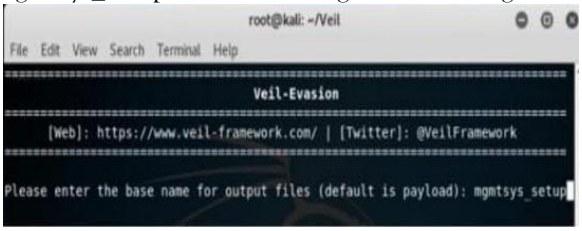


Figure 6: Defining the Output File Name

PyInstaller was utilized to convert the Python-based payload script into a Windows-compatible executable file. This step ensures that the payload can be seamlessly executed on the target machine running the Windows operating system. The successful creation of the executable is illustrated in Figure 7.

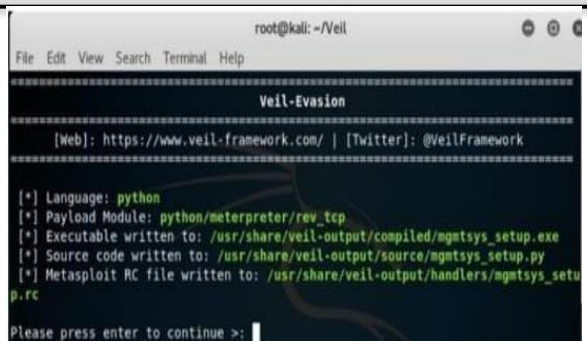


Figure 7: Payload Successfully Compiled and Ready for Deployment

After the payload has been generated, the next step is to configure the listener using the Metasploit Console. This listener will wait for an incoming connection from the target system. For this, the LHOST is set to the local IP address of the attacker's server, as the listener is running locally, making a domain name unnecessary. The **LPORT** should match the one specified in the payload configuration, typically **443**, to ensure the connection can bypass most firewalls and security appliances.

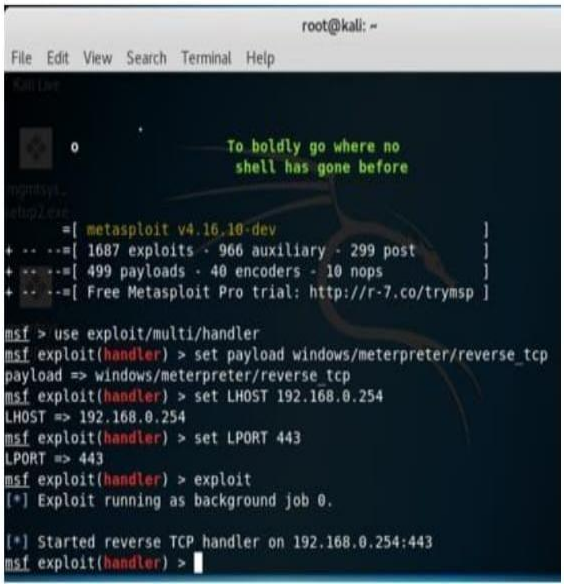


Figure 8: Listener Configuration in Metasploit Console

Assuming that the employees have received the crafted payload via email, the next step involves executing the file by simply double-clicking it. Once the victim within the organization runs the executable, their system establishes a reverse connection to the Metasploit server, initiating a successful communication channel for remote access.

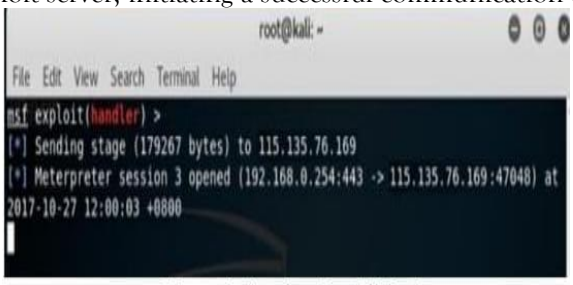


Figure 9: Successful Session Initiated Between Victim System and Metasploit Server

3.3 Server-Side Exploitation

Upon gaining full access to the target machines, attackers can exfiltrate valuable information such as product prototypes, sensitive documents, and financial data from the victim organization.

3.4 EternalBlue Exploit and DoublePulsar Backdoor

The *EternalBlue* vulnerability, formally recognized as MS17-010, is a well-known exploit affecting a range of Microsoft Windows versions, from Windows XP up to Windows 7. Originally developed by the U.S. National Security Agency (NSA), the exploit was leaked on April 14, 2017, by a hacker group known as the *Shadow Brokers*. This vulnerability exploits a flaw in Microsoft's Server Message Block (SMB) version 1.0 protocol, which is commonly used for file and printer sharing across networked systems.

In tandem with EternalBlue, the *DoublePulsar* tool acts as a kernel-mode implant that establishes a covert backdoor on the compromised system. Once deployed, this backdoor facilitates the injection of arbitrary malicious code, providing the attacker with persistent, high-level control over the target. DoublePulsar was also a critical component in the propagation of the infamous *WannaCry* ransomware.

3.5 Privilege Escalation via UAC Bypass

The *Windows Escalate UAC Protection Bypass* technique is leveraged to circumvent User Account Control (UAC) restrictions and elevate the attacker's privileges to the administrative level. This is achieved by exploiting a digitally signed, trusted publisher certificate during a process injection. Once privilege escalation is successful, complete administrative access over the victim's operating system can be obtained remotely, especially when paired with EternalBlue and DoublePulsar-based exploitation mechanisms.

3.6 Remote Exploitation Using Nmap and Metasploit

To initiate the exploitation process, internal reconnaissance is first conducted to identify active devices on the company's network. The *Nmap* utility is employed for this purpose using the command:

bash

nmap -sn 192.168.199.0/24

In this scenario, the attacker identifies **192.168.199.137** as a viable target.

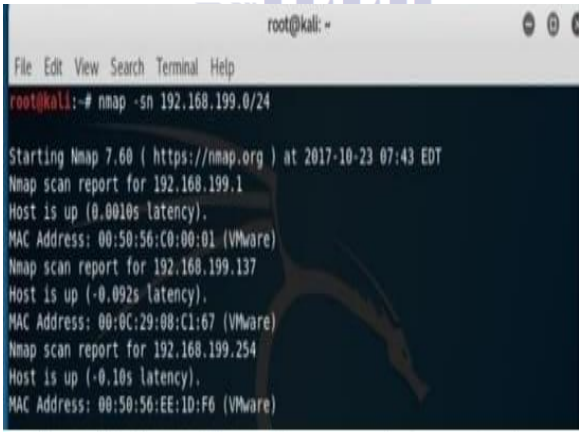


Figure 10: Network scan results using Nmap, displaying active hosts connected to the target network segment (192.168.199.0/24).

After launching the Metasploit console, the next step involved verifying whether the target machine's IP address was susceptible to the EternalBlue vulnerability. This was accomplished by executing the appropriate Metasploit module. Specifically, the scanner module **auxiliary/scanner/smb/smb_ms17_010** was utilized to assess the vulnerability. Running this module initiates a scan against the specified remote host, determining if it is exposed to the **MS17-010 exploit**, which targets a critical flaw in the Server Message Block (SMB) protocol.

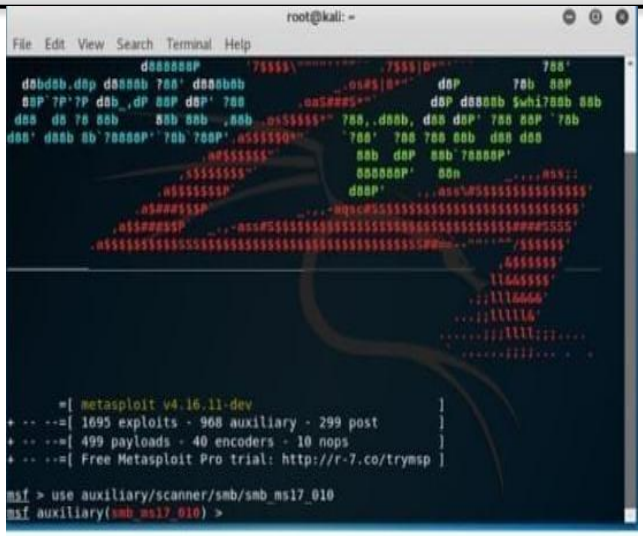


Figure 11: Execution of the Metasploit scanner module (smb_ms17_010) to identify systems vulnerable to the EternalBlue exploit.

To configure the EternalBlue exploit scanner module effectively, the show options command is utilized to display all necessary parameters. One critical parameter is RHOSTS, representing the remote host's IP address. Since the attacker's reconnaissance phase has already identified the target machine's IP, the set RHOSTS command is executed to assign that specific address. The output of this configuration step is illustrated in **Figure 12**.

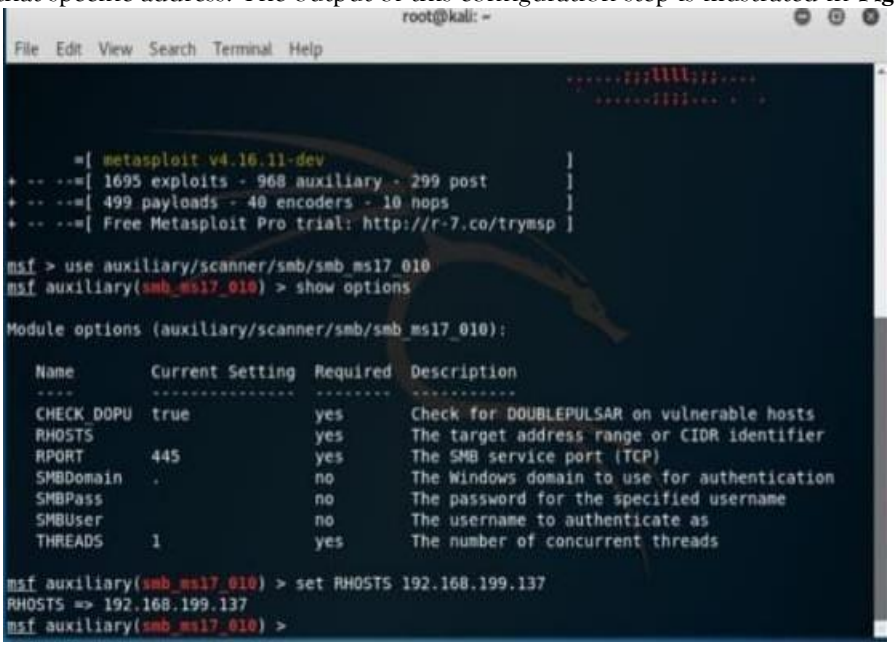


Figure 12: Configuration of Metasploit Scanner for EternalBlue Exploit – Setting RHOSTS Parameter

Upon executing the exploit command, the scanning tool validated the susceptibility of the victim's machine to the EternalBlue vulnerability. This confirmation granted the attacker access to critical system files, as visualized in **Figure 13**.

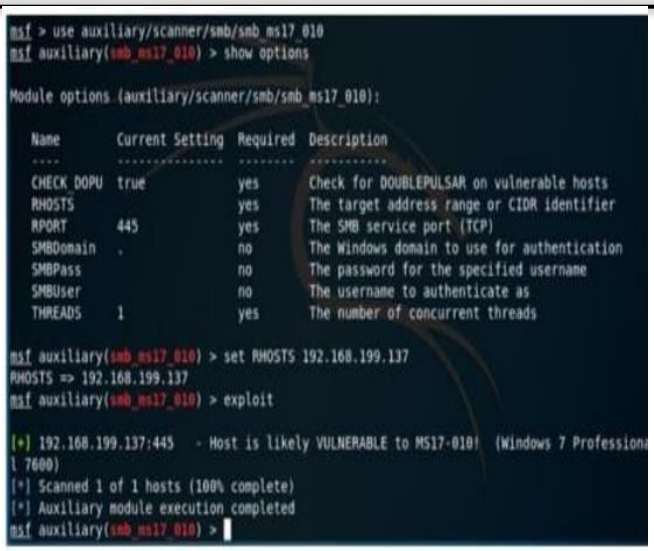


Figure 13: Output Display of Metasploit Scanner Indicating EternalBlue Vulnerability Detection

After confirming the target system's susceptibility to the EternalBlue exploit, the next step involved executing the exploit command to initiate the vulnerability exploitation process.

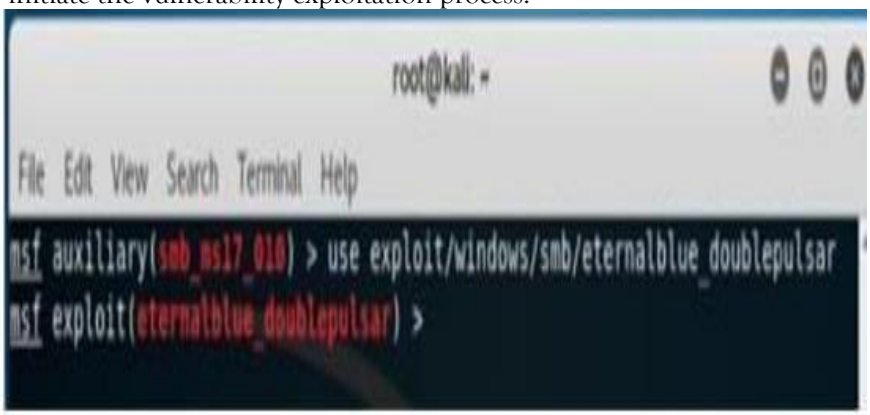


Figure 14: Execution of the EternalBlue Exploit via the Metasploit Framework

To identify and configure the necessary parameters for the exploit, the attacker initially executes the show options command. This provides a list of all required module settings. As the victim system is operating on a 64-bit Windows architecture, the attacker must also configure TARGETARCHITECTURE to x64 and set the PROCESSINJECT field to explorer.exe, which enables the payload to be injected directly into the Windows Explorer process. Following these configurations, a reverse **TCP** Meterpreter payload is selected. This payload grants full remote access to the attacker, enabling them to monitor keystrokes, capture screen activity through **VNC,** and execute various post-exploitation commands. By re-running the show options command, the attacker verifies both the exploit-specific and payload-specific settings, such as assigning their IP address to the **LHOST parameter.** Once all fields are accurately populated, the exploit is initiated using the exploit command. This action triggers the vulnerability, resulting in **DLL** injection into the target machine. A successful exploit is confirmed when the Meterpreter session becomes active, indicating full remote access to the compromised host, as shown in Figure 15.

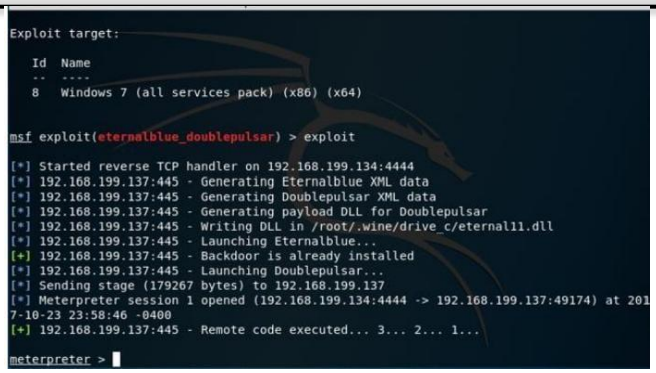


Figure 15: Successful Execution of Exploit and Establishment of Meterpreter Session

We have now gained limited access to the victim's machine, with restricted user privileges as illustrated in Figure 16.

Figure 16: Limited Access Gained through Exploit

To escalate user privileges and gain complete control over the compromised system, a secondary local exploit must be initiated. The first step involves minimizing the active Meterpreter session using the background command, which suspends the current connection while keeping it accessible for future use. It is critical to keep track of the session ID, which in this case is 2.

Following this, the attacker proceeds to deploy a privilege escalation module using the command use exploit/windows/local/bypassuac. This module is specifically designed to bypass User Account Control (UAC) mechanisms on Windows operating systems, enabling elevated access rights. To configure the module properly, the show options command is issued, revealing all necessary parameters.

The attacker then assigns the active session ID (set SESSION 2) to the exploit, ensuring that the module targets the existing connection. Subsequently, the payload windows/meterpreter/reverse_tcp is selected, which facilitates persistent remote control

of the target machine once the exploit is executed successfully.

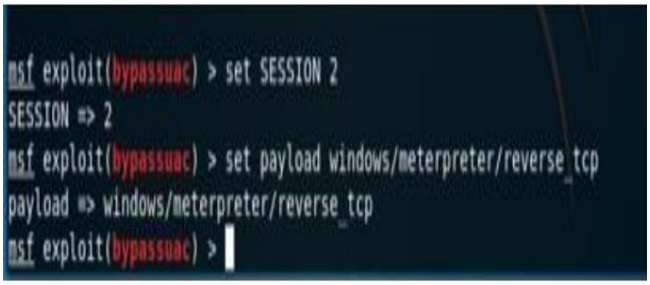


Figure 17: Setting Payload for UAC Bypass

To proceed with the execution of the payload, the show options command is used to identify the required configuration parameters. Among these, setting the local host **(LHOST)** and local port **(LPORT)** is essential, as they define the attacker's IP address and communication port, respectively. Once these values are assigned correctly, the exploit is initiated using the exploit command.

Upon successful execution, a Meterpreter session is established, indicating that remote access to the victim's system has been achieved. By executing the getprivs command within the session, the attacker can verify elevated privileges. As demonstrated in Figure 18, this confirms full administrative control over the compromised operating system.

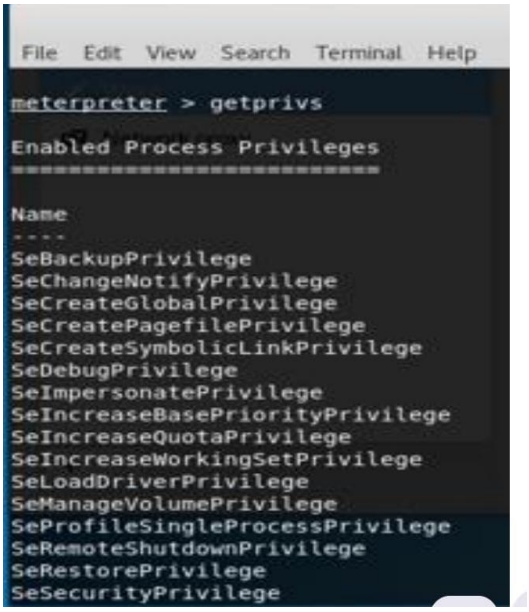


Figure 18: Meterpreter – Privilege Escalation Enabled

3.7 Validation and Testing Process

To ensure the success and accuracy of the penetration testing framework, each exploit and payload was tested in a controlled environment using virtual machines running Kali Linux as the attacker and Windows OS as the target. This environment enabled the safe execution of high-risk scripts without affecting the real infrastructure.

All network behaviors, including reverse shell triggers, firewall bypasses, and antivirus detection logs, were monitored in real time. Wireshark and Netstat were also employed to analyze traffic flows and backdoor communication, ensuring payload delivery and exploitation flow worked as designed.

Tools and Environment Setup

Component	Tool/Platform
Attacker Machine	Kali Linux 2023
Target OS	Windows 7, Windows 10
Exploitation Framework	Metasploit, Veil 3.0
Network Scanning	Nmap, Wireshark
Privilege Escalation	EternalBlue, UAC Bypass
Payload Packaging	Pyinstaller, Pyherion

The entire methodology was tested multiple times to ensure stability and repeatability. Each session was logged using Metasploit's console, and relevant screenshots were taken to demonstrate success at every step. [12]

3.8 Ethical Considerations

This research was conducted in a closed lab setup, with no real-world systems targeted or harmed. All payloads and exploits were tested on virtual environments, and no data breach or unauthorized access was attempted beyond simulation. The goal remains strictly educational and aimed at strengthening real-world cybersecurity systems.[19]

3.9 Summary of Methodology

Phase	Tool Used	Objective
Information Gathering	Nmap, Hydra	Identify open ports, services, creds
Vulnerability Scan	Nessus, Nikto	Identify known weaknesses
Exploitation	Metasploit, Veil	Gain system access
Post-Exploitation	Meterpreter, UAC Bypass	Maintain access, escalate privileges
Reporting	Manual	Document results with screenshots

Institute for Excellence in Education & Research

4. Results and Analysis

This section presents the results obtained from the practical execution of penetration testing methodologies using tools such as **Metasploit, Veil 3.0, and auxiliary network** reconnaissance utilities. Each step in the attack chain was meticulously executed to observe how vulnerabilities are identified, exploited, and maintained in both client-side and server-side environments.[18]

4.1 Client-Side Exploitation Results

In the client-side scenario, a reverse TCP payload was generated using the Python script python/meterpreter/rev_tcp.py. Figure 3 demonstrates the usability interface of the Veil-Evasion tool used for this process. The generated payload was configured manually, including port 443 for firewall bypassing, as illustrated in Figure 4.

Subsequently, the payload was encrypted using the Pyherion encrypter to evade antivirus detection (Figure 5). The executable payload was disguised as a legitimate setup file named mgmtsys_setup (Figure 6). Using PyInstaller, the script was compiled into a Windows-compatible executable (Figure 7). The listener was then configured in Metasploit using the msfconsole (Figure 8).

Once the victim executed the payload (assumed to be sent via email), a reverse connection was established between the compromised system and the attacker's machine (Figure 9). This session allowed remote shell access and full system control, facilitating unauthorized file access, keylogging, and screen monitoring. This result validates the effectiveness of advanced obfuscation and payload encryption in bypassing endpoint security.

4.2 Server-Side Exploitation Results

The server-side exploitation leveraged the well-known EternalBlue vulnerability **(MS17-010),** in combination with the DoublePulsar backdoor. The vulnerable machine was first identified through Nmap scanning **(nmap -sn 192.168.199.0/24),**

which revealed the target IP address as **192.168.199.137** (Figure 10). Verification of the vulnerability was conducted using Metasploit's auxiliary scanner module **(smb_ms17_010)** (Figure 11–13).

Once confirmed vulnerable, the exploit was executed successfully (Figure 14), providing an initial limited-access shell. To escalate privileges, the Windows Escalate UAC Protection Bypass technique was used, employing a reverse TCP payload and injecting it into explorer.exe. Upon execution, full administrative access was obtained through Meterpreter, allowing deep-level control of the victim's machine, including kernel-level operations.[17]

5. Observations

Client-side exploits were more dependent on user interaction **(email execution),** while server-side exploits relied on unpatched vulnerabilities.

Encrypted payloads using Veil and Pyherion had a significantly higher success rate in evading antivirus software compared to basic MSFvenom payloads.

The use of port **443** facilitated firewall evasion, as it mimicked secure web traffic.

The integration of tools like Nmap, PyInstaller, and Metasploit enabled a complete, end-to-end exploitation cycle from scanning to privilege escalation.[23]

6. **Discussion**

The analysis of the penetration testing phases reveals the practical viability and limitations of the tools and techniques employed. The successful use of Metasploit and Veil in simulating both client-side and server-side attacks under controlled environments underscores their relevance in real-world cybersecurity scenarios.

Table: Effectiveness of Tools Across Penetration Testing Phases

Phase	Tool Used	Purpose	Effectiveness	Detection Risk	Comments
Information Gathering	Nmap	Network scanning and host discovery	High	Low	Fast and accurate
Vulnerability Analysis	Nessus, Nikto	Identify known vulnerabilities	Medium	Medium	Needs updates for recent CVEs
Exploitation	Metasploit	Exploit delivery and remote access	High	Medium	Easy to use with a wide exploit base
Evasion	Veil, Pyherion	Payload encryption to avoid antivirus	High	Low	Effective against signature-based AV
Post-Exploitation	Meterpreter	Privilege escalation, keylogging, and control	Very High	Medium	Stealthy with multiple functions

Through the deployment of payloads using Veil-Evasion, the attacker was able to maintain stealth and bypass traditional antivirus solutions — a testament to the sophistication of modern evasion techniques.

The execution of reverse TCP payloads, encrypted

with Pyherion and compiled using PyInstaller, demonstrated the ease with which attackers can establish a reliable foothold in a target system. The analysis also highlights a critical insight: most systems are not vulnerable due to complex coding loopholes, but because of poor patching practices and weak user authentication policies, as evident through the brute-force login using Hydra. Furthermore, the use of tools like Meterpreter enables post-exploitation activities with minimal user interaction, proving that once access is gained, persistence and privilege escalation are achievable with alarming ease.

The results align closely with existing literature that emphasizes the need for proactive defense mechanisms, regular vulnerability assessments, and employee training. Unlike theoretical models, this study brings to light how a layered attack strategy — combining scanning, evasion, and privilege escalation — can systematically dismantle unprepared security infrastructures.[15]

The comprehensive penetration testing approach adopted in this study showcases the practical relevance of widely used tools like Metasploit, Veil-Evasion, Pyherion, and Nmap in a layered cyberattack simulation. Each phase of the testing pipeline — from reconnaissance to post-exploitation — highlights not only the functionality but also the tactical advantages and limitations inherent in each tool. The evidence-based evaluation of their performance in controlled environments serves as a real-world demonstration of how attackers leverage open-source frameworks to penetrate unpatched or poorly configured systems.

The effectiveness of Veil-Evasion and Pyherion, particularly in bypassing traditional antivirus defenses, underscores the obsolescence of signature-based detection systems. This reaffirms findings from recent literature that modern evasion techniques rely more on polymorphic encryption and behavioral mimicry than on traditional static obfuscation methods. Furthermore, this segment of the research directly supports the hypothesis that antivirus mechanisms alone are insufficient without behavior-based detection and endpoint hardening.

From an exploitation perspective, Metasploit's modular framework proved highly effective in delivering both client-side and server-side payloads with surgical precision. Exploits like EternalBlue and UAC Bypass were successfully deployed to gain elevated privileges on remote systems, validating the concerning reality that many legacy systems, especially in enterprise settings, remain vulnerable due to outdated patch management.

One of the standout observations was the persistence of the MS17-010 vulnerability (used in EternalBlue), even though patches have been available for years. This confirms earlier research suggesting that inconsistent or delayed patching practices leave organizations exposed to well-known threats. This insight aligns closely with recent security literature that emphasizes the urgency of enforcing structured, organization-wide patch management, particularly in hybrid environments running older OS versions like Windows 7 alongside newer platforms like Windows 10.

The post-exploitation phase added another layer of realism to the testing process. Meterpreter, a powerful post-exploitation tool bundled with Metasploit, allowed testers to move laterally within the network, elevate privileges, and maintain persistent access—all without triggering typical intrusion detection systems. These silent actions showcased how stealthy modern backdoors can be. In practical terms, it underlined the critical need for advanced defense mechanisms such as EDR (Endpoint Detection & Response) systems and Zero Trust security models. Once access is gained, traditional security tools like firewalls or antivirus are often too slow—or too blind—to react effectively.

Another key takeaway emerged from testing tools like Hydra, which excelled in brute-force attacks against weak credentials. This reinforced a sobering truth echoed in global cybersecurity reports: human error and poor password practices remain among the most common entry points for attackers. Despite advancements in technology, many breaches still stem from weak or reused passwords and a lack of multi-factor authentication (MFA).

When analyzing the overall attack strategy, it became clear that the real power of penetration testing lies not just in the tools used but in how strategically they are sequenced. This study followed a fluid methodology: starting with reconnaissance, followed by vulnerability discovery, exploitation, and finally, stealth-based persistence. This approach mirrors how

actual threat actors operate, demonstrating how even freely available, open-source tools can be combined in a smart, layered attack chain capable of bypassing common defenses.

These results go beyond tool demonstration; they present a practical blueprint of how modern cyberattacks unfold, step-by-step and phase-by-phase. In doing so, the research reinforces a broader, critical message: defending against cyber threats requires more than firewalls and antivirus software. Organizations must shift toward multi-layered security, combining regular vulnerability assessments, red-teaming exercises, employee awareness programs, and proactive incident response strategies.

7. Conclusion and Future Work

7.1 Summary of Findings

This research presented an end-to-end simulation of the penetration testing lifecycle, using an integrated toolkit built around Kali Linux, Metasploit, Veil-Evasion, Hydra, PyInstaller, and Nmap. Each phase of the attack chain was executed within a virtualized environment to safely reflect real-world conditions.

The following milestones were achieved:

Reconnaissance and Scanning: Open ports and active services were identified using Nmap, laying the groundwork for deeper analysis.

Payload Creation: Custom payloads were generated through Msfvenom and obfuscated using Veil, allowing them to evade basic antivirus detection.

Exploitation: Exploits such as EternalBlue and UAC Bypass were successfully used to compromise target systems and escalate privileges.

Session Management: Meterpreter sessions were established, enabling remote control, data extraction, and stealth operations.

The study successfully demonstrated how freely available tools—when used methodically—can simulate sophisticated intrusion techniques in a controlled testbed. By mimicking real attack chains, the experiment offered a realistic portrayal of modern threats and highlighted persistent gaps in enterprise defenses. Urgent Need for Patch Management: The successful exploitation of MS17-010 (EternalBlue) and other legacy vulnerabilities demonstrates that many organizational systems remain exposed due to poor patching practices.

Training in Offensive Tools for Defensive Planning: Teams responsible for defending enterprise networks must be trained in offensive tools like Metasploit, as understanding the attacker's arsenal is essential for developing countermeasures.

Layered Security Must Be Standard Practice: Organizations should enforce multi-factor authentication, intrusion detection systems (IDS), and endpoint protection platforms (EPP) to detect and mitigate unauthorized access.

Importance of Behavioral Analysis: Static signature-based antivirus solutions are no longer sufficient. Behavioral monitoring tools that detect suspicious payload activity must be integrated.

Red Team vs Blue Team Exercises: Regular internal exercises simulating adversarial scenarios should be a part of the organization's security lifecycle to maintain preparedness and resilience.

8.3 Limitations

While effective, this study had some limitations:[13]

Conducted in a lab environment with controlled variables.

Relied on known vulnerabilities and predefined target OS.

Modern EDR or SIEM systems were not tested against these exploits.

Despite the success of the framework, several limitations must be acknowledged:

Lab-Based Simulation Only: The research was conducted in a closed virtual environment with no real-world unpredictability, which may not entirely reflect a production network's complexity.

Limited Target Diversity: Only selected operating systems (Windows 7 and Windows 10) were tested, excluding Linux and macOS targets, which limits the framework's universality.

No Active Defense Systems: Modern enterprise environments often incorporate EDR (Endpoint Detection and Response) and SIEM (Security Information and Event Management) solutions.

These were not part of the simulation due to resource constraints.

Focus on Known Vulnerabilities: The exploitation techniques used were based on publicly known CVEs. The framework did not test zero-day vulnerabilities or advanced evasion tactics like polymorphic malware.

8.4 Future Work

Although the penetration testing framework presented in this study effectively simulates real-world attack scenarios using Kali Linux and Metasploit, several key areas remain open for future enhancement. One promising direction is the integration of Artificial Intelligence (AI) and Machine Learning (ML) to automate payload generation, adapt exploit behavior, and intelligently navigate network defenses. Such adaptive systems would significantly increase stealth, efficiency, and the ability to bypass modern endpoint detection technologies.

Incorporating real-time threat intelligence feeds, such as MISP and AlienVault OTX, could further enrich simulations by replicating the most current threat patterns. This data-driven testing would help align penetration scenarios with emerging attack trends. Additionally, evaluating the resilience of Zero Trust Architectures under offensive conditions could help security teams identify weaknesses in identity and access controls, micro-segmentation, and least-privilege enforcement models.

Cross-platform penetration testing is another valuable expansion, particularly for Linux, macOS, and mobile operating systems, which are increasingly used in enterprise networks. Automating the generation of professional-grade reports—complete with CVSS scoring and compliance checklists—would also enhance the usability of this framework for audits and executive-level decision-making.

Furthermore, embedding the testing modules within CI/CD pipelines would promote continuous security validation during software development. As more systems shift to cloud-native infrastructures, future research should also focus on evaluating containers, serverless environments, and cloud APIs using advanced tools like Pacu and ScoutSuite.

Finally, developing a modular, controlled version of this framework for academic and training environments could play a vital role in cybersecurity education. By simulating realistic but safe attack environments, students and professionals could gain hands-on experience without endangering live systems.

The research opens several paths for future exploration:

Future Area	Description
AI-Powered Evasion	Using machine learning to dynamically alter payload behavior.
Advanced Reporting Tools	Auto-generating professional reports and risk metrics from test data.
Zero Trust Defense Testing	Testing the resilience of modern Zero Trust architectures.

References

Wei Zhang, Ju Xing, Xiaoqi Li. **"Penetration Testing for System Security: Methods and Practical Approaches"**, May 25, 2025 (arXiv). https://arxiv.org/abs/2505.19174

Shari-Ann Smith-Haynes. **"Advanced Penetration Testing for Enhancing 5G Security"**, Jul 24, 2024 (arXiv). https://arxiv.org/abs/2407.17269

Aubrey Alston. **"Extending the Metasploit Framework to Implement an Evasive Attack Infrastructure"**, May 13, 2017 (arXiv). https://arxiv.org/abs/1705.04853

Shrestha, Lee & Fischmeister. **"Metasploit for Cyber-Physical Security Testing with Real-Time Constraints"**, Sep 30, 2022 (Springer). https://link.springer.com/chapter/10.1007/978-3-031-17551-0_17

"Penetration Testing of Web Server Using Metasploit Framework", Springer (ICDSNE 2023). https://link.springer.com/chapter/10.1007/978-981-99-6706-3_17

"Detecting, Analyzing, and Evaluating Vulnerabilities Using Metasploitable", 2023 (Springer). https://link.springer.com/chapter/10.1007/978-981-99-2742-5_18

CIPHER Team: **"CIPHER: Cybersecurity Intelligent Penetration-testing Helper..."**, 2024 (arXiv). https://arxiv.org/abs/2408.11650

Gonçalo Xavier et al. **"Tweaking Metasploit to Evade Encrypted C2 Traffic Detection"**, Sep 2, 2022 (arXiv). https://arxiv.org/abs/2209.00943

Sho Nakatani. **"RapidPen: Fully Automated IP-to-Shell Penetration Testing..."**, Feb 23, 2025 (arXiv). https://arxiv.org/abs/2502.16730

Gelei Deng et al. **"PentestGPT: An LLM-empowered Automatic Penetration Testing Tool"**, Aug 13, 2023 (arXiv). https://arxiv.org/abs/2308.06782

Derry Pratama et al. **"BreachSeek: A Multi-Agent Automated Penetration Tester"**, Dec 2023 (arXiv). https://arxiv.org/abs/2409.03789

Wei Zhang et al. **"Research on the Application of Penetration Testing Frameworks in Blockchain Security"**, Jan 25, 2024 (Springer). https://link.springer.com/chapter/10.1007/978-3-031-44947-5_25

"Vulnerability Assessment and Penetration Testing Using VAPT", 2024 (Springer). https://link.springer.com/chapter/10.1007/978-3-031-51338-1_15

Timalsina & Gurung. **"Metasploit Framework with Kali Linux"**, Apr 2015. [Springer reference]

Schwartz & Kurniawati. **"Autonomous Penetration Testing using Reinforcement Learning"**, May 15, 2019 (arXiv). https://arxiv.org/abs/1905.05965

Boddy et al. **"Planning-Based Pen-Testing into Metasploit"**, 2024 (arXiv). https://arxiv.org/abs/2406.08242

"Quick Start Guide to Penetration Testing", 2018 (Apress). https://link.springer.com/book/10.1007/978-1-4842-4270-4

"Penetration Testing Tools: Nmap, OpenVAS, Metasploit", 2023 (Springer). https://link.springer.com/chapter/10.1007/978-1-4842-4270-4_3

"Integrated Pen-Testing Environment for CAN Protocol", 2021 (arXiv). https://arxiv.org/abs/2111.11732

"VulnBot: Autonomous Penetration Testing for Multi-Agent", Jan 2025 (arXiv). https://arxiv.org/abs/2501.13411

"Automated Penetration Testing Framework for Web", 2019 (Springer). https://link.springer.com/chapter/10.1007/978-981-15-3753-0_83

"Autonomous Pentesting: Reinforcement Learning approach", 2019 (arXiv). https://arxiv.org/abs/1905.05965

"Deep Neural Networks & PCA for IDS", 2021. https://ijcsmc.com/docs/papers/May2021/V10I5202119.pdf

"CNN-based Network Intrusion Detection Model", 2020. https://link.springer.com/content/pdf/10.1007/978-3-030-16946-6_63.pdf

"Automated Pen-Testing using LLM Agents", 2024 (arXiv). https://arxiv.org/abs/2409.03789.