Automated Reverse Shell through Undetectable and Obfuscated Techniques

By:

Hamid
35415
Abdul Wahab
36676
Muneeb Ur Rehman
32575

Supervised by:
Mr. Hummayun Raza
Lecturer at Riphah International University

Faculty of Computing
Riphah International University, Islamabad
Fall 2024

A Dissertation Submitted To

Faculty of Computing,

Riphah International University, Islamabad

As a Partial Fulfillment of the Requirement for the Award of the

Degree of

Bachelors of Science in Cyber Security

Faculty of Computing
Riphah International University, Islamabad

Date: 07 December,2024

Final Approval

This is to certify that we have read the report submitted by ***Hamid (35415), Abdul Wahab (36676), and Muneeb Ur Rehman (32575),*** for the partial fulfillment of the requirements for the degree of the Bachelor of Science in Cyber Security (BS CYS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelor of Science in Cyber Security (BS CYS).

Committee:

1 _____

Hummayun Raza
(Supervisor)

2 _____

Dr. Musharraf Ahmed
(Head of Department)

Declaration

We hereby declare that this document "Automated Reverse Shell through Undetectable and Obfuscated Techniques" neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **Mr. Hummayun Raza**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

_____

Hamid

35415

_____

Abdul Wahab

36676

_____

Muneeb Ur Rehman

32575

Dedication

“Automated Reverse Shell through Undetectable and Obfuscated Techniques” belongs to the category of studies dedicated to the retired layers of cybersecurity threats as the digital issues are much more variant, and their development is considerably faster than the improvement of the elementary protection methods.

This project discusses factors of obfuscation, stealth, and continuity to show how one can perpetrate reverse shells that would anyway evade advanced ant-virus systems. It is our honor to present it to the cybersecurity community – to the experimenters, outlaws, and white hats who dedicate themselves to studying, creating, and eradicating complex problems that exist right under our noses. In their constant creation relative to these emerging forms, they progress discipline to make the digital environment safer.

We also dedicate this work to our teachers and advisors, without whom it was impossible to overcome the intricacies of reverse shell generation, as well as obfuscation and penetration testing. Their insight of the weaknesses and new trends have enhanced our knowledge regarding the significance as well as focus to this important area.

This work should have a benefit for penetration testing and would encourage other authors to state in cybersecurity to improve our joint capacity to guard against complicated, invisible dangers step by step.

Acknowledgement

First of all, we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project. Next, our families deserve our gratitude for their constant support and patience while we were working on this project. We sincerely appreciate Mr. Hummayun Raza, our supervisor, for his mentoring and insightful advice. Last but not least, we would also like to thank the entire teaching staff at the Faculty of Computing at Riphah International University for equipping us with the fundamental knowledge and abilities needed for this project.

Hamid

35415

Abdul Wahab

36676

Muneeb Ur Rehman

32575

Abstract

In the modern threat environment, the capacity to efficiently verify and strengthen protection against growing danger is critical. This project presents an enhanced undetectable reverse shell framework capable of setting up automated penetration test with obfuscation features on board. Being a highly developed tool for cumulating arbitrary payload, the tool works invisibly, and among other things it constantly changes its behavior and signs dynamically to avoid detection by modern antivirus systems consistently. In two folds, it allows penetration testers to conduct real attack scenarios, giving useful information of defectivity in the defensive structures. In this regard, it also facilitates the growth of improved antivirus and endpoint protection solutions that provide proactive counteraction against complex concealing and avoiding techniques. Gaining process and in-memory execution and support for process injection and automated deployment, the tool speeds up the creation and deployment of stealthy reverse shells for large scale pen testing. Thus, it plays the crucial role of an efficient tool for both offense and defense in cybersecurity domain and helps to study system vulnerability to contemporary threats and develop system protection strategies.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Chapter 1: Introduction

1.1 Opportunity & Stakeholders

The main opportunity of this project lies within automating the creation and deployment of reverse shell, enabling and equipping penetration testers and cybersecurity professionals to utilize and test frameworks with advanced and improved stealth-oriented methods to conduct system assessments that also avoid detection. This development is very important as modern antivirus solutions available have become more advanced in detecting, recognizing conventional reverse shell tactics, thus there is a need to require advanced, sophisticated evasion strategies. Key Stakeholders involved in this initiative include:

- **Cybersecurity Professionals and Penetration Testers:** Benefiting from automated and obfuscated techniques, these professionals will have a framework and more systematic strategies to bypass antivirus software's with grater effectiveness.
- **Antivirus and security program developers:** The insights from this framework can help in enhancing and upgrading detection capabilities, also helping antivirus solutions against such progressed threats.
- **Educational institutions and Researchers:** The framework offers valuable and practical insights into evasion tactics, serving for academic and research purposes.

1.2 Motivations and Challenges

The motivation behind this project stems from the limitation of existing tools, that often require extensive manual efforts which can be extremely challenging and show inconsistent success rates in evading antivirus software. Existing frameworks like Metasploit, Villain, and Netcat lack automating for mass and broad deployment, relies totally on reiterative, manual, hands-on adjustments. Some of the key challenges are listed below:

- **Evasion Consistency:** The evasion consistency of the reverse shell is a big challenge, ensuring that reverse shell constantly bypass detection mechanisms of different antiviruses.
- **Automation:** Automation is another big challenge, developing a scalable framework that automates creation, obfuscation, achieving persistence and deployment. Combining all these processes to reduce or minimize manual workload.

- **Persistence and Stealth:** Maintaing the ability of payload to be undetected, prolonged access on targeted system is also another challenge which is being addressed in this framework, while remaining undetectable by advanced antivirus programs.

1.3 Significance of study

This research is very important as it addresses the significance and evolving complexity of antivirus software, highlighting and demanding the need of developing new and more sophisticated evasion strategies. The creation and development of an automated reverse shell, employing different obfuscation techniques, helps in enhancing defensive security by allowing antivirus developers to identify and mitigate potential vulnerabilities, while it also benefits offensive security personals and pentesters by supporting their assessments. Furthermore, this framework also contributes to support of cybersecurity research and fosters the practical application of evasion strategies in real-world environments.

1.4 Goals and Objectives

The main objectives and goals of this project are as follows:

- **Improving Evasion Techniques:** This improvement in evasion techniques is being depicted by taking deepen insights into antivirus vulnerabilities. This is done by emulating sophisticated evasion tactics akin and resemble those employed by threat actors or adversaries.
- **Automating Reverse Shell Deployment:** Automating Reverse Shell Deployment is to improve, optimize and streamline penetration testing efficiency and accuracy. This is done through automating the creation and deployment of obfuscated reverse shell, thereby increasing both factors.
- **Achieving Persistent Access:** The persistent access is to devise and create strategies to enable sustained, covert and long-term access to compromised systems while facilitating through uninterrupted Pentesting.

1.5 Scope of project

The project aims to develop and create an automated, stealthy reverse shell that is highly capable of circumventing, evading contemporary antivirus software. The project's scope highlights the following:

- **Automated payload Generation:** The process of generating an automated payload is by using high-level languages that are then compiled into executable files. After that reverse shell payloads are created.
- **Obfuscation Techniques:** Obfuscation Techniques like in-memory execution of PowerShell, string manipulation, variable obfuscation, and Execution Policy Bypass are being used to effectively evade detection mechanisms.
- **Payload Delivery:** The Delivery process of Payload is by applying and utilizing tactics like Man-in-the-Middle (MITM) attacks or embedding payload within authentic files for successful delivery.
- **Persistence:** Persistence is achieved for establishing prolonged, concealed, hidden access to compromised system for an extended period and for sustained testing. This is being done by different processes and techniques like Process manipulation and hidden execution.

1.6 Chapter Summary

- This chapter has outlined and focus on the primary aim of developing an automated framework that can deploy undetectable reverse shell, encapsulated in our project title, "Automated Reverse Shell using Undetectable and Obfuscated Techniques". The project is aimed at tailoring the needs of cybersecurity professionals, offensive security personals and pentesters, who are in need of, seeking, sophisticated tools to avoid detection by contemporary, available antivirus software.
- This is how we highlight the broader significance of developing evasion techniques that not only bolster offensive security capabilities but also shed light on antivirus vulnerabilities. Thereby the project benefits the defensive cybersecurity, by identifying important stakeholders such as cybersecurity specialist, antivirus software developers, and our researchers in academia.
- The chapter examines the reasons and difficulties underlying with current tools, exploring the limitations and challenges, and the obstacles inherited, which are not automated,

scalable or capable of consistent evasion. This is the gap that the project seeks to solve. This research provides significant value in improving the procedures of cybersecurity, also benefitting both offensive and defensive security initiatives.

- Among the stated goals of the project were achieving persistence, undetected system access, automated reverse shell deployment, and refining evasion strategies. The chapter achieved, setting a foundation for in-depth technical discussion in subsequent chapters. The chapter ended, delineating the project's scope, along with highlighting essential technical elements payload generation, obfuscation, stealthy delivery, and persistence.

Chapter 2:
Market Survey

Chapter 2: Market Survey

2.1 Introduction

This chapter delves into the current landscape of available reverse shell frameworks and associated technologies, with a focus on analysis of existing tools employed in penetration testing for deploying remote access mechanisms. The aim is to compare functionalities, strengths, advantages and disadvantages, and limitations of these frameworks. Especially within the context of obfuscation, automation, and antivirus evasion capabilities.

This chapter establishes a foundation, uncovering the groundwork for identifying gaps, recognizing deficiencies, and understanding where existing solutions fall short, furthermore underscoring the need for a more sophisticated, advanced, undetectable reverse shell framework that meets and tailored to the dynamic requirements of modern cybersecurity testing.

2.2 Market Survey/Technologies & Products Overview

Given where there is a rising demand for automated and undetectable reverse shell in cybersecurity, there are also several frameworks that have been developed and emerged to support and facilitate creation, deployment and obfuscation. Widely recognized frameworks include Metasploit, Hoax Shell, and Veil-Evasion. These frameworks have been playing a crucial role and have been proven essential for penetration testing; however, each of them exhibits limitation in automation, consistency in antivirus evasion, and scalability across multiple targets. These tools often necessitate manual configuration and adjustments, also lacking the streamlined, large-scaled and efficient deployment capabilities that are demanded by modern penetration testing environment. Moreover, varying and inconsistent detection rates across different antivirus solutions poses further complications in the reliability of these frameworks.

Existing Systems

1. **Metasploit Framework**

Functionality:

Metasploit is a comprehensive penetration testing and offensive security tool with an extensive library of exploit modules, payloads, and obfuscation options. It enables pentesters or security

personals to deploy revers shell and various other exploits. This provides versatility for a variety of cybersecurity tests.

Limitations:

- **Detection**: Due to its wide use, many antivirus solutions have been able to flag and recognize Metasploit's payloads, leading them to be detected easily.
- **Manual Obfuscation**: Although Metasploit provides some obfuscation options, they often require additional manual adjustments, reducing the payloads efficiency of bypassing antivirus software.
- **Scalability**: Metasploit lacks fully automation for large-scale deployments across multiple systems, necessitating repetition of configuration across multiple systems.

Problems:

- **Inconsistent Evasion**: Ever since Metasploit's payloads are commonly flagged, it means that testers must spend time manually reconfiguring and re-obfuscating payloads to avoid their detection from antivirus solutions, slowing down workflow.
- **Time-Consuming for Large-Scale Testing**: Without Metasploit's multi-target deployment feature, pentesters must individually configure and deploy payloads for each target, which in most cases, lead to delays and higher chances of errors.

2. **HoaxShell**

Functionality: HoaxShell is a lightweight reverse shell tool/framework designed for low detectability. It offers a simple setup and quick deployment. It has been ideal framework for scenarios where minimal footprints and basic reverse shell functionality is required and sufficient.

Limitations:

- **Limited Functionality**: HoaxShell provides all the basic reverse shell functionalities, but it lacks in advance features like obfuscation, persistence, and multi-target management.
- **Obfuscation Constraints**: HoaxShell does not support in-memory execution or dynamic payload alteration/modification, limiting it against more sophisticated antivirus solutions.
- **Not Scalable**: HoaxShell primarily suits single-target operations, it lacks and is not suited for large-scale deployments.

Problems:

- **Reduced Effectiveness Against Advanced AVs**: Being lacking in obfuscation and advanced stealth techniques, HoaxShell payloads are more likely to be detected by modern AV solutions.
- **Labor-Intensive for Broader Testing**: Since HoaxShell lacks automations for handling multi-target, this framework is not efficient in testing scenarios that involves multiple systems, increasing the workload on testers.

3. Villain

Functionality: Villain is tailored for handling and managing multiple reverse shell sessions, allowing simultaneous connections across diverse targets. It includes remote command execution feature, making it suitable for environments where pentesters and security personals requires control over multiple systems.

Limitations:

- **Manual Evasion Techniques**: Villain provides limited obfuscation options and lacks automated evasion. It means that, even that it handles and manages multiple reverse shells connections, but it still requires significant manual inputs to avoid detection.
- **Niche Focus on Session Management**: Villain emphasizes and prioritizes multiple session management, rather than of stealth, thus limiting its effectiveness in environments where undetectable shells are needed.
- **Limited Persistence**: Villian does not emphasize techniques for maintaining persistence, and having persistent access, thus reducing its efficacy for long-term control.

Problems:

- **Increased Manual Configuration**: with only basic obfuscation technique, pentesters and security personnels often need to manually adjust payloads for each session created. This can be time consuming in environments with active AV solutions, or environments where more than one AV solution is active.

- **Reduced Stealth**: the absence and lacking advanced evasion and persistence features make Villian less effective in environments where long-term, undetected access is critical and needed.

3. Veil-Evasion

Functionality: Veil-Evasion is a tool designed and built to help penetration testers and offensive security personals, for creating obfuscated payloads that is capable of evading antivirus detection. This framework uses a variety of encoding and manipulation techniques to improve stealth.

Limitations:

- **Manual Obfuscation**: Veil-Evasion requires pentesters and security personals to manually apply obfuscation techniques, thus making it labor-intensive for frequent or testing over a large-scale.
- **Inconsistent Evasion**: By the time, AV solutions are being regularly updated, thus Veil-Evasion payloads often become ineffective, being flagged over time. Then these payloads require frequent re-obfuscation.
- **Single Payload Focus**: Veil-Evasion is limited to single payload creation. This is a big flaw for pentesters and security personnel as they would need to handle multiple targets at a time.

Problems:

- **Constant Need for Updates**: Due to so much updating of antivirus solutions, Veil-Evasion payloads may lose their effectiveness and become detectable by these AV solutions, thus requiring pentesters and security personnel to frequently adjust their payloads.
- **Time and Labor Intensive**: Lack of Veil-Evasion in both automation and multi-target support, demanding repetitive manual work, which can slow down penetration testing, particularly in large environments.

5. Shellter

Functionality: Shellter is another powerful tool, used in making reverse shell connections. It allows shell code injection into windows executables, enhancing the payload's stealth by embedding the payload within seemingly legitimate processes or applications.

Limitations:

- **Compatibility Constraints**: Shellter only operates with windows executables. By this, it limits its utility in Unix operating systems like Linux and Ubuntu.
- **Manual Injection Process**: Shell code injection requires intensive manual setup for each payload, which is often repetitive and time consuming.
- **Lack of Automation**: Shellter does not support automated obfuscation across multiple targets, hindering and impacting on scalability.

Problems:

- **Platform Dependency**: Being Windows exclusive and limited, Shellter is not suitable for non-Windows environments like Linux and Ubuntu. Limiting it to pentesters and security personnel.
- **Increased Configuration Time**: In larger testing scenarios and environments, manually setting up everything and lack of automation can significantly slow down deployments, making Shellter less efficient for quick and scalable testing.

6. Empire

Functionality: Empire is a post-exploitation framework which focuses on maintaining access and control over target system. Empire has been very helpful for pentesters and offensive security personnel. It supports PowerShell based reverse shell with additional persistence and lateral movement capabilities.

Limitations:

- **High Detection Risk**: Empire totally relies on PowerShell which makes it highly and easily detectable by modern antivirus solutions and endpoint monitoring mechanisms, particularly in environments where PowerShell movements are actively monitored.
- **Manual Customization Required**: Empire does not fully automate the configuration process. Thus, additional manual adjustments for each reverse shell payload.

- **Resource Intensive**: Empire is oriented and designed towards post exploitation rather than initial stealth. This makes Empire resource-heavy for scenarios focused purely on evasion.

Problems:

- **Limited Stealth in Modern Environments**: Since PowerShell activities are being actively monitored in many environments, Empire payloads are often flagged, which also reduces their effectiveness.
- **Reduced Efficiency in Large-Scale Testing**: Empire is always in need of manual adjustments for each session. Furthermore, Empire emphasizes post-exploitation rather than evasion. This makes it less suitable for scenarios where the delivery of an undetectable, automated and obfuscated payload is required.

2.3 Comparative Analysis

Table 2.1 Comparative Analysis of Existing Frameworks

Framework	Primary Focus	Key Features	Limitations	Problems
Metasploit	Comprehensive penetration testing with large exploit library	Exploit modules, payloads, obfuscation options	High detection rate, manual obfuscation, lacks scalability	Inconsistent evasion, time-consuming for large-scale tests
HoaxShell	Lightweight, minimal footprint reverse shell	Simple configuration, low detection, minimal footprint	Limited functionality, no obfuscation, single-target	Reduced AV evasion, labor-intensive for broader testing
Villain	Multi-session management for multiple targets	Simultaneous connections, remote command execution	Basic obfuscation, no automated evasion, limited persistence	Increased manual config, lacks stealth in high-security setups
Veil-Evasion	Obfuscated payload creation for AV evasion	Encoding and manipulation for AV evasion	Manual obfuscation, frequent re-obfuscation needed, single-target	Constant updates required, labor-intensive for large environments
Shellter	Shellcode injection into Windows executables	Embed shellcode into legitimate apps	Windows-only, manual injection, no multi-target automation	Platform dependency on Windows, slow deployment for scalability

Empire	Post-exploitation with persistence and control	PowerShell reverse shells, persistence, lateral movement	High detection risk, manual setup, resource-intensive	Limited stealth in monitored environments, inefficient for evasion-focused tasks

2.4 Research Gaps

The existing frameworks in the current landscape, demonstrates and revels several significant gaps, some of which are as follows:

- **Challenges in Stealth and Obfuscation**: Existing all the frameworks have limited, inconsistent evasion capabilities. This leads to payloads to frequent detection by
- **Insufficient Automation**: As we have analyzed above, many tools automated processes for obfuscation techniques or multi-targeted deployments, thus leading to inefficient and increased time requirements for workflows.
- **Ineffective Updates for Evasion AV**: With the AV solutions evolving, updating and patching frequently, the effectiveness of evasion strategies and techniques found in these frameworks, diminishing frequently. They mainly lack providing sophisticated, automated and adaptive obfuscation methods, making it challenging for pentesters and security personnel to maintain prolonged and undetectable access to the target machines.
- **High Detection Rates**: Most of these common frameworks are easily flagged and detected by antivirus systems due to the widespread use and limited obfuscation capabilities.
- **Resource-Intensive Configuration**: There is a need for manual setup, which in some cases, is necessary for each target or payload. In scenarios where large-scale testing is involved, this intensive resource configuration slows down the processes.
- **Limited Persistence Mechanisms**: As we have seen earlier, while some tools addresses session management, there is a little emphasis on implementing stealthy, automated persistence techniques. Enhancing such mechanisms would provide pentesters and security personnel with enhanced control over targeted systems.

2.5 Problem Statement

The exploitation process employed in existing frameworks are often protracted, resource-intensive, and inefficient. They can be identified by modern AV solutions. Generation,

Customization, and obfuscating payloads, to bypass these modern antiviruses detection can significantly extends once time and budget. The time required increases resource consumption. This not only demands considerable effort from penetration testers but also reduces operational efficiency. Diminishing operational efficiency, particularly when conducting large-scale assessments across multiple systems.

Furthermore, the need for reliance on manual configuration and adjustment of payloads, combined with repeated testing against various AV systems and continual refinement of obfuscation techniques, adds further complexity and delays. These limitations hinder and undermine the overall effectiveness and scalability of penetration testing efforts.

2.6 Chapter Summary

This chapter offers an in-depth market survey, along with examination of established reverse shell frameworks, including Metasploit, HoaxShell, Villain, Veil-Evasion, Shellter, and Empire, with a focus on their utility in cybersecurity, Pentesting, and for offensive security personnel. Although these tools are being used in creating payloads, they demonstrate significant shortcomings in areas where there is more need for things such as automation, stealth, and scalability. One major limitation and a common challenge among all of them lies in their high detection rates, which stem from widely recognized payload signatures and inadequate obfuscation methods. These issues often require frequent manual adjustments for any of the related personnel to bypass antivirus defenses.

Additionally, frameworks lack the ability of automation for multi-target deployments which restricts the effectiveness of these frameworks in large-scale testing environments. The analysis also identifies other key limitations which are sufficiently considered around Pentesting, including platform-specific dependencies, insufficient persistence mechanisms, and the need for extensive manual configuration. Such constraints not only hinder the effectiveness for one, but also complicate cross-platform testing, increase the effort required from penetration testers, and adversely impact productivity and scalability.

The chapter concludes by emphasizing the necessity for a more advanced framework that can integrate these multiple functionalities like automated, undetectable, and adaptive reverse shell functionalities.

Chapter 3:
Requirements and System Design

Chapter 3: Requirements and System Design

3.1 Introduction

Requirement engineering forms and serves the foundation/cornerstone for identifying a system's goals, functional capabilities, purpose, and limitations. It ensures that the final design meets our security professional and Pentesting requirements while fulfilling operational expectations.

3.2 System Architecture

System architecture is a key component which displays how our model will proposedly work in developing a reliable and effective design. Figure 3.1 illustrates the proposed model's architecture, offering a detailed depiction of our operational framework and workflow.

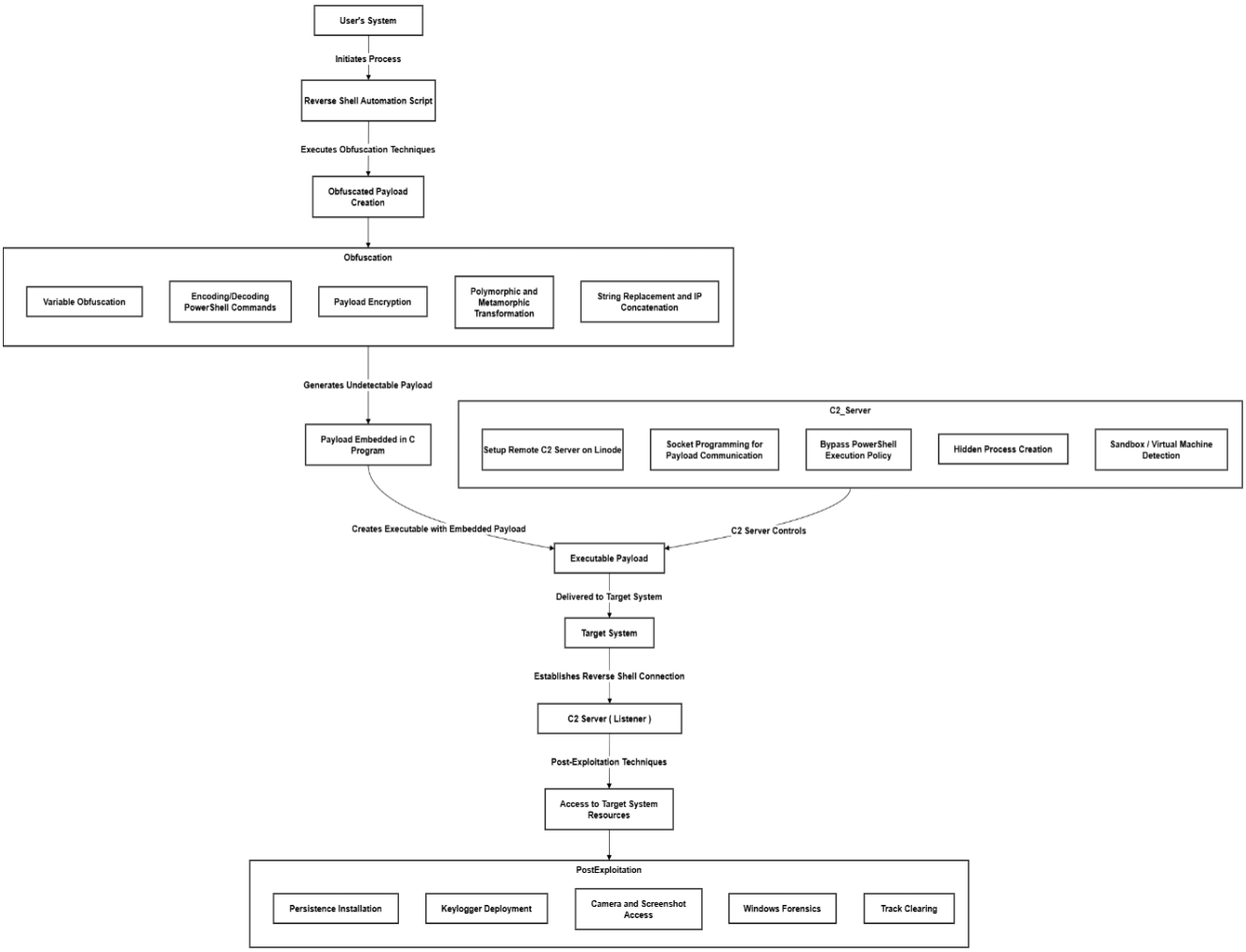

Figure 3.1 System Architecture

3.3 Functional Requirements

Outlined below are the main functional requirements for our proposed reverse shell framework:

3.3.1 The framework shall generate reverse shell payloads equipped and integrated with obfuscation techniques to bypass antivirus software and endpoint detection systems.

3.3.2 The payload shall detect itself whether execution within virtual machines or sandboxed environments or not, refraining itself and avoiding running under such conditions to avoid automated analysis.

3.3.3 The framework shall support multi-target deployment, enabling automated configuration and dissemination of payloads across multiple systems.

3.3.4 The framework shall implement adaptive obfuscation, providing dynamically altering payloads to reduce the likelihood of detection by antivirus solutions.

3.3.5 The framework shall enable in-memory execution, so that payloads shall be executed directly in memory, minimizing traces left on disk and enhancing stealth capabilities.

3.3.6 The framework shall include functionality for persistence. Persistence mechanisms must be incorporated to maintain long-term access to target systems while avoiding detection over extended periods.

3.3.7 The framework shall offer and should facilitate remote command execution across multiple active sessions, allowing for simultaneous control of several targeted systems.

3.3.8 The system shall provide real-time feedback on the detection status of payloads, enabling users to adjust and refine obfuscation strategies as needed.

3.4 Non-Functional Requirements

The non-functional requirements, which outline the framework's quality attributes, are as follows:

3.4.1 The framework shall exhibit and achieve a high degree of stealth/undetectability, minimizing false positives and detection rates from antivirus and endpoint security solutions.

3.4.2 The framework shall maintain efficiency in terms of performance. It must deliver efficient performance, ensuring rapid payload generation and deployment without imposing excessive computational overheads.

3.4.3 The framework shall be resilient and adapted to evolving antivirus and security measures, regularly updating and adapting its obfuscation and evasion techniques.

3.4.4 The framework shall offer an intuitive and user-friendly interface be, simplifying the configuration and management of reverse shell operations and deployment for penetration testers.

3.4.5 The framework shall prioritize its own security, preventing unauthorized access or modification/tempering of its payloads.

3.4.6 The framework shall be scalable to support large-scale penetration testing operations. Scalability is essential, enabling the framework to support multiple concurrent targets without compromising performance.

3.4.7 The framework shall provide reliability in operations; it must be ensuring consistent performance across diverse environments and under varying network conditions.

3.4.8 The framework shall minimize resource usage when deployed over a targeted system, prevent detection, maintaining a low operational footprint in terms of CPU and memory consumption on target systems

3.5 Dataflow Diagram

Figure 3.1 Dataflow Diagram

3.6 Hardware and Software Requirements

Oblivioun Strike has the following hardware as well as software requirements:

Hardware Requirements

CPU: 2-core processor (Intel i3 or equivalent)

RAM: 4 GB

Storage: 20 GB free space

Network: Basic Ethernet or Wi-Fi connection

Software Requirements

Software Requirements:

Operating Systems:

Linux (Kali Linux, Ubuntu, Debian, Parrot OS)

Windows (7, 8.1, 10)

PowerShell: Version 5.1 or later

Linode: Nanode 1 GB

Virtual Machine Configuration:

VM Software: VMware Workstation or Oracle VirtualBox

Recommended VM Settings:

Processor (2 cores), Memory (2 GB per VM), Storage (20 GB dynamically allocated)

3.7 Threat Scenarios

Threat scenarios referencing the "Automated Reverse Shell through Undetectable and Obfuscated Techniques" project include threats which are a threat to the stability or purity of the tool. The following threat scenarios address possible weaknesses and exploitation paths that attackers or malicious entities could leverage:

3.7.1 Violation of the Framework for the purpose of Harm

An adversary could get into the framework and use it to create invisible reverse shell payloads for misuse. This could lead to widespread abuse of the facility and this can lead to theft of data, unauthorized intrusion into computer systems or even tampering with essential structures. Left uncovered, the framework could unwittingly aid attackers: if there are no restrictions on access to these frameworks or software meters are not appropriately monitored, a threat actor will have their work made easier.

3.7.2 Payload Detection by Advanced Antivirus Software

Even though the project used obfuscation techniques so that the payload wouldn't be caught, future improvements on antivirus and endpoint protection systems might catch it. This could enable the detection and counter measure of the tool and rendering it inoperable for penetration testing and expose the operation of it to the defender of the target system.

3.7.3 Unintended Target Compromise

An unintended target may be compromised by the payload delivery mechanism, in this case from MITM attacks, or by binding payloads into legitimate files. As an example, if the manipulated file, resource is used while a legitimate user accesses it, this could negatively impact the system of that penetration tester, creating ethical and legal repercussion.

3.7.4 Persistent Access is Utilized by Adversaries

This is especially true when the reverse shell has laid down its tent on a given system in a compromised manner; in other words, it can be manipulated by other unauthorized third parties, or malicious insiders. This could result in prolonged unauthorized control of the target system, and, therefore, higher probability of data leakage or a disruption of services.

3.7.5 Unsecured command and control (C2) Communication

The communication between the established reverse shell and C2 server may be easily sniffed by the skilled attackers if the concept of encryption or using secure channel is not employed. It could enable the attackers to recognize and seize the C2 session or even listen to the querent and respondent's interchange of incriminating data compromising the tester.

3.7.6 Legal and Ethical Risks in Testing

Misuse of the framework in the wrong way like sending out payloads without express permission of the owner of the target system it was directed to could attract legal consequences. Further, the exploitation of this tool in unethical contexts proved dangerous to the credibility of pen testers or the businesses that employ the tool.

3.7.7 Lack of capability to elude an Endpoint Detection and Response (EDR) system

The current versions of EDR use sophisticated algorithms to identify the actions that are performed by malware. In some cases, if the payload does not modify its behavior to new heuristics or does not possess optimal levels of invasiveness and behavioral disguise, the system will detect it, hindering successful deployment or execution.

3.8 Threat Modeling Techniques

The process of recognizing and evaluating possible security risks to a system, comprehending the viewpoint of an attacker, and implementing preventative measures is known as threat modelling. We have adopted the STRIDE model to detect and lessen risks for our framework which is displayed in the Table 3.1: STRIDE Model below:

Table 3.1 STRIDE Model

STRIDE | Threats	Payload Tampering	Antivirus Evasion	MITM Attack Delivery	Persistence
Spoofing			√	
Tampering	√		√	
Repudiation	√			
Info disclosure		√		
Denial of service				√
Elevation of privilege				√

3.9 Threat Resistance Model

That is why the multi-layered defense concept has been used to protect the "Automated Reverse Shell through Undetectable and Obfuscated Techniques" project from existing threats. This model also has preventive, detective and corrective control to minimize risks and improve security in an organization. Below is an overview of how the system resists various security threats:

3.9.1 Preventive Actions

Payload Encryption and Obfuscation: The reverse shell payload also undergoes encryption and obfuscation to bypass all antivirus detection systems in addition to vandalism. This case advanced string manipulation as well as script alteration techniques are utilized.

Access Control: Testers/pagans are the only ones who have the ability to browse and release payloads. Standard security measures are put in place to unsafe signs in systems to offer protection against impostors.

Delivery Validation: A particular kind of verification tests is performed on the payload delivery process to check for any modifications to the payload.

3.9.2 Detective Actions

Real-time Monitoring: Payload modifications are checked to see if they are made by any other host than the senders and MITM traffic is also checked to see if it is abnormal in any way.

Antivirus Bypass Testing: The academic payloads are scanned in a real-world environment against fresh AV signatures in the search of effective ways to bypass them.

Network Traffic Analysis: A pattern comparing traffic flow related to payload transfer or reverse shell running is conducted to identify potential interception attempts by an analyst.

3.9.3 Corrective Actions

Dynamic Re-obfuscation: If a payload is found or raised then new re-Obfuscation methods are used to create a new payload signature so that it continues to remain undetectable.

Command and Control (C2) Redundancy: In case of a disruption, there are configured fallback C2 servers to ensure communication is kept up.

System Cleanup: When the payload is damaged, there are recovery scripts which wipe records off the target system and prevent the adversary from further investigations.

3.10 Chapter Summary

In this chapter, the specifications, structure, and security measures of the conceptualized Automated Reverse Shell through Undetectable and Obfuscated Techniques framework are discussed. The functional and non-functional specifications were preset: self-generated payload; evasive capability avoiding perception; operating environments: secure settings; scalability of targets. Implementation plans for technologies, such as a processing subsystem, a storage subsystem, a Linux/Windows/MAC virtual environment, and endpoints, were defined with due reference to the desired performances in hardware and software.

We also evaluated threat scenarios by using more formalized STRIDE analysis to assess and counter existing threats. Presenting a Threat Resistance Model, possibility of layered protection was discussed to improve system's capability to protect against potential threats. Taken in their entirety, this chapter offers a solid ground for human understanding of how it is possible to arrange and protect an effective reverse shell environment without the detriment of speed, capacity and operational invisibility.

Chapter 4:
Proposed Solution

Chapter 4: Proposed Solution

4.1 Introduction

The complexity of risks is growing year by year thus demanding the usage of progressive aggressive security tools to analyze and strengthen protection mechanisms. Of all these, reverse shells have emerged as one of the most essential tools in pentesting because such hackers can get access to target systems from other locations. This project is all about creating an extremely stealthy and fully automated reverse shell framework along with the chosen command & control (C2) server.

The payload used in the proposed solution incorporates numerous new elements, such as various forms of payload obfuscation, in-memory operation, and persistence methods, to contribute to the efficiency and invisibility of the payload. Combined with process injection techniques and changing the payload signatures on the fly, the framework plans on being able to evade most of today's antivirus solutions (Microsoft Defender). The goal is to give pen testers a solid tool and, in the process, show just how dangerous similar techniques would be for the same organizations if used by malicious entities.

4.2 Proposed Model

Create a reverse shell payload to establish a connection with a command and control (C2) server for remote access to the target system.

- **Payload Development:**
    - Written in High Level Language.
    - Compiled into an .exe file.
    - Runs as a hidden process without user interaction.
- **Stealth Enhancement:**
    - Use bash scripts and Python for automation.
    - Obfuscate the payload to dynamically alter its signature or hash with each execution.
    - Aim to bypass popular antivirus solutions, including Windows Defender.
- **Payload Delivery Techniques:**
    - Bind the reverse shell payload with legitimate files or resources.

    - Employ Man-in-the-Middle (MITM) attacks to deliver the payload without raising suspicion.
- **In-Memory AV Evasion:**
    - Process injection into trusted system processes.
    - Ensure the payload remains undetected.
- **Persistence Access:**
    - Maintain long-term access to the compromised system.

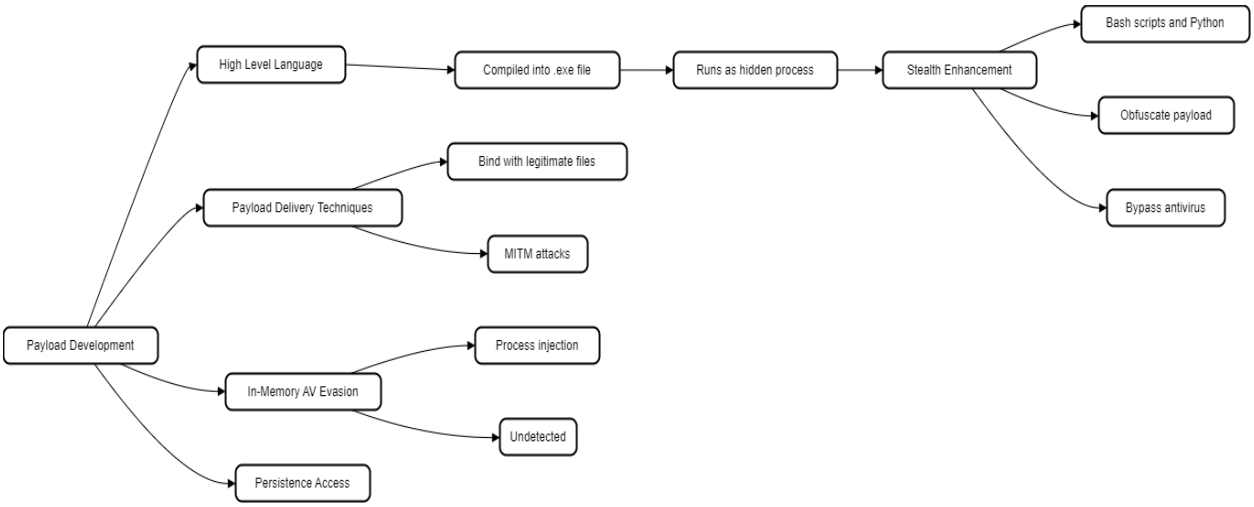

Figure 4.1: Proposed Solution