RIPHAH INTERNATIONAL UNIVERSITY

Faculty of Computing

FINAL YEAR PROJECT PROPOSAL & PLAN

Automated Reverse Shell through Undetectable and Obfuscated techniques

Project Team

Full Name of Student	**SAP Id**	**Program**	**Contact Number**	**Email Address**
Muneeb Ur Rehman	**32575**	**BSCY**	**0310-6161831**	**32575@students.riphah.edu.pk**
Abdul Wahab	**36676**	**BSCY**	**0311-1912045**	**36676@students.riphah.edu.pk**
Hamid	**35415**	**BSCY**	**0341-5117009**	**35415@students.riphah.edu.pk**

Hummayun Raza
Lecturer at Riphah International University

Automated Reverse Shell through Undetectable and Obfuscated techniques

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
Abdul Wahab, Muneeb ur Rehman, Hamid	1.0	August 28, 2024	Original Draft	[illegible]
Abdul Wahab, Muneeb ur Rehman, Hamid	1.1	September 02, 2024	New Project Title Assigned	

Project Proposal

Project Title: Automated Reverse Shell through Undetectable and Obfuscated techniques

Introduction and Background: To develop an undetectable reverse shell, the focus should be on employing obfuscation techniques to effectively evade modern antivirus detection. This includes automating the creation and deployment of the reverse shell to streamline the process. Techniques such as string manipulation, variable obfuscation, and script alteration should be utilized to bypass detection mechanisms. Additionally, it is essential to evaluate the stealth of the payload by testing it against current, live, and updated antivirus software versions, ensuring its continued effectiveness in evading detection.

As cybersecurity advances, attackers continually refine techniques to bypass detection systems. Reverse shells allow remote control of compromised systems, but modern antivirus and endpoint detection tools have become adept at neutralizing them. To counter this, obfuscation methods like string manipulation, variable obfuscation, and script alteration are used to conceal malicious payloads from both signature and behavior-based detection. Automating the creation and deployment of these reverse shells enables rapid generation of new variants, enhancing their ability to evade detection. Regular testing against updated antivirus software ensures continued stealth and effectiveness.

Existing Systems/ Survey/ Literature Review: Current frameworks for payload creation, such as Metasploit, Netcat, and Veil-Evasion, still rely heavily on manual configuration for creating, customizing, and obfuscating payloads. This process can be time-consuming and prone to human error, especially in large-scale testing scenarios. Despite their modularity, these tools lack seamless automation for deploying payloads across multiple targets simultaneously, forcing penetration testers to perform repetitive tasks manually. Additionally, the fragmented nature of these tools requires testers to switch between different frameworks, slowing down the workflow and reducing overall efficiency.

Another significant gap is the inconsistent success rate in bypassing antivirus solutions. Different antivirus programs respond variably to the same payloads, and existing frameworks do not guarantee consistent detection evasion. As antivirus software evolves, static payloads generated by these tools often become ineffective, requiring testers to manually tweak and re-obfuscate them. Moreover, many frameworks lack real-time testing capabilities against live, updated antivirus programs, meaning testers must manually test payloads, further complicating the process and limiting the effectiveness of these frameworks.

Problem Statement: Existing tool surveys reveal that the exploitation process in current frameworks is often lengthy, time-consuming, and requires extensive manual work. The steps involved in creating, customizing, and obfuscating payloads to bypass antivirus detection extend the overall process, significantly increasing resource utilization. This not only demands considerable time and effort from penetration testers but also reduces operational efficiency, especially when dealing with large-scale testing across multiple systems. The need to manually configure and adjust payloads, test against various antivirus programs, and tweak obfuscation techniques to ensure success adds to the complexity and further slows down the workflow.

Furthermore, the lack of automation in these frameworks limits their scalability, making it difficult for testers to handle multiple targets simultaneously. As a result, the process becomes repetitive, increasing the chances of human error and lowering the effectiveness of the penetration testing efforts. Given these challenges, there is a clear need for an automated framework that can streamline the entire process—from bypassing antivirus detection to creating and deploying reverse shells—while ensuring scalability, efficiency, and reduced resource consumption.

Objectives: Our primary objective is to enhance cybersecurity defenses and streamline penetration-testing efforts through two key initiatives:

Improve Antivirus Detection: We aim to advance antivirus detection capabilities by simulating sophisticated evasion techniques used by attackers. By identifying and analyzing vulnerabilities in current AV solutions, we will provide actionable insights that help AV vendors strengthen their defenses and better protect against emerging threats.

Automate Evasion Techniques: We seek to optimize the process of bypassing modern AV solutions by automating advanced evasion techniques. Our goal is to create a tool that reduces the manual effort required for effective penetration testing and red teaming, thereby increasing the accuracy and efficiency of security assessments while overcoming the limitations of existing manual methods.

Proposed Solution: 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.

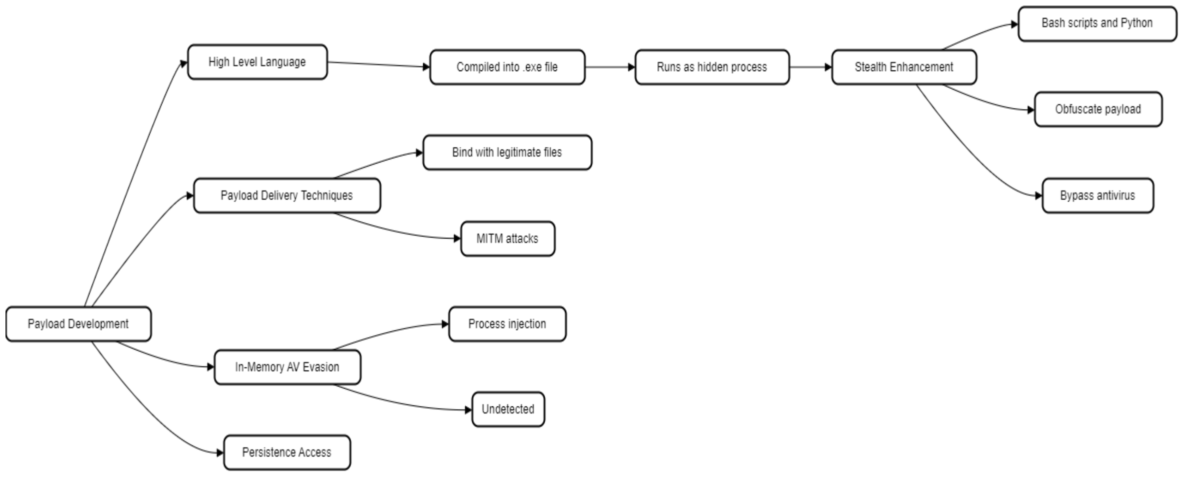

Figure 1: Proposed Solution

Methodology: Our project involves creating a reverse shell payload designed to establish a connection with a command and control (C2) server, enabling remote access to the target system. The payload will be written in high level language and compiled into an .exe file that runs as a hidden process without user interaction. To enhance stealth, we will use bash scripts and Python to automate the obfuscation process, dynamically altering the file's signature or hash with each execution to bypass popular antivirus (**Windows Defender**).

We will also incorporate in-memory AV evasion techniques, such as process injection into trusted system processes, ensuring the payload remains undetected. Persistence will be achieved through various methods to maintain long-term access to the compromised system.

For payload delivery, we will explore techniques like binding the reverse shell payload with legitimate files or resources, as well as employing Man-in-the-Middle (MITM) attacks to deliver the payload without raising suspicion.

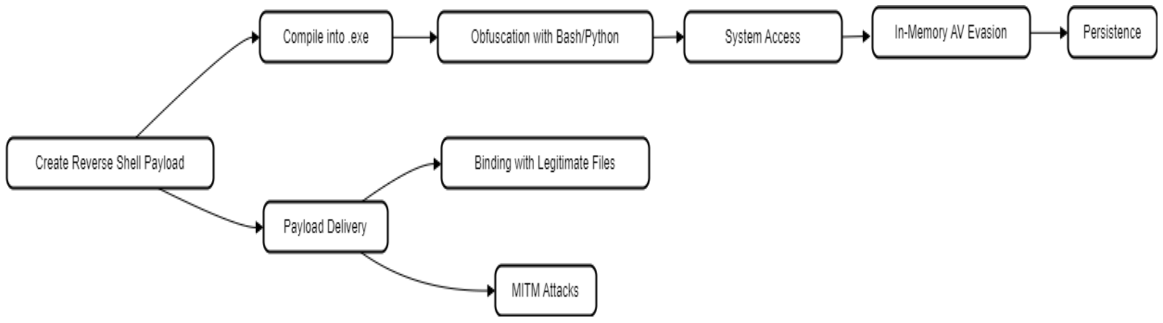

Figure 2: Methodology

Implementation Plan:

Domain 1: Payload Creation & Obfuscation

We will begin by creating a reverse shell payload that establishes a connection back to our system. The payload will be written in High Level Language and compiled into an .exe file. To ensure stealth, it will run as a hidden process, with no terminal or GUI prompts visible to the target. We will use bash scripting and Python automation to obfuscate the payload, employing techniques that dynamically change the file's signature or hash with each execution to bypass Windows Defender. Additionally, we will implement command and control (C2) functionality to maintain communication with the server. To further evade detection, we will explore in-memory AV evasion techniques such as process injection into trusted processes, ensuring the antivirus is fully bypassed. Finally, we will ensure persistence by utilizing methods that grant long-term access to the compromised system.

Domain 2: Payload Delivery

In the payload delivery phase, we will first establish a Man-in-the-Middle (MITM) connection between the target machine and an access point. When the target downloads legitimate resources from the internet, we will seamlessly bind the reverse shell payload with these files. This strategy allows for discreet delivery without raising suspicion. By leveraging network-based delivery methods, we can ensure the payload infiltrates the target system undetected.

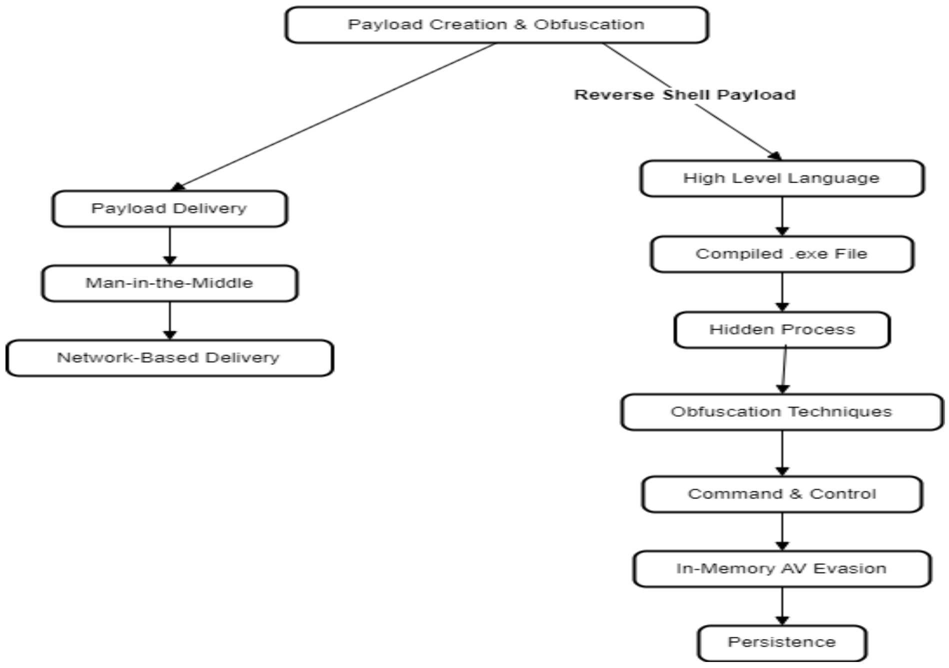

Figure 3: Implementation Plan

Evaluation Plan:

To evaluate the effectiveness of the payload, it must first be tested against live and updated antivirus software by deploying it in a controlled environment. This will help assess its ability to bypass the latest protection mechanisms in real-world scenarios. Additionally, testing should be conducted across renowned antivirus solution (**Microsoft Defender**) to determine how well the payload evades detection by a wide range of leading security product. The success criteria for these tests should include several key factors. First, the detection evasion rate must be high. Second, the payload should have minimal impact on system performance, ensuring it operates efficiently without slowing down the target system. Third, the payload must demonstrate persistence, maintaining its operation without being flagged or removed by an antivirus solution. Lastly, the payload's functionality is critical, and it should reliably execute its intended functions without disruption, ensuring it remains an effective tool for penetration testing.

Project Scope: The scope of this project is to design and implement an automated reverse shell capable of evading modern antivirus detection using obfuscation techniques. It will focus on automating payload creation, obfuscation, and deployment to reduce manual effort and improve scalability across multiple systems. The reverse shell payload will be written in High Level Language, compiled into an executable, and run as a hidden process, ensuring stealth and minimal system performance impact. Advanced techniques such as process injection and in-memory antivirus evasion will be explored to keep the payload undetected while maintaining persistence on compromised systems.

Expected Outcomes: The project aims to develop a scalable tool for penetration testers, allowing efficient large-scale assessments. Expected outcomes include a fully automated framework for creating and deploying undetectable reverse shells that bypass popular antivirus solutions like Windows Defender. Methods like MITM attacks for seamless payload delivery and real-time testing against updated antivirus software will ensure the payload remains effective and reliable. This project will reduce manual effort and improve overall efficiency in penetration testing and detection evasion.

Conclusion and Future Work: This project centers on developing an automated reverse shell using obfuscation techniques to bypass modern antivirus detection while ensuring stealth and persistence on compromised systems. Automating the creation, obfuscation, and deployment processes reduces manual effort, minimizes human error, and enhances scalability for large-scale penetration testing. Advanced methods like process injection and in-memory antivirus evasion ensure that the payload remains undetected, even by widely used antivirus solutions such as Windows Defender.

For future work, post-exploitation efforts will focus on maintaining control over the compromised system by implementing features like keylogging, camera access, and system screenshots. Additionally, sensitive data such as passwords and credentials will be gathered, with continuous access ensured while clearing traces to avoid detection. This comprehensive approach strengthens the overall effectiveness of penetration testing, ensuring both immediate and long-term control over compromised systems while evading detection.

References:

- Y. Kolli, T. K. Mohd and A. Y. Javaid, "Remote Desktop Backdoor Implementation with Reverse TCP Payload using Open Source Tools for Instructional Use," *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2018, pp. 444-450, doi: 10.1109/IEMCON.2018.8614801.
- A. Viticchié *et al*., "Assessment of Source Code Obfuscation Techniques," *2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, Raleigh, NC, USA, 2016, pp. 11-20, doi: 10.1109/SCAM.2016.17.
- M. U. Rana, M. Ali Shah and O. Ellahi, "Malware Persistence and Obfuscation: An Analysis on Concealed Strategies," *2021 26th International Conference on Automation and Computing (ICAC)*, Portsmouth, United Kingdom, 2021, pp. 1-6, doi: 10.23919/ICAC50006.2021.9594197.
- theseus.fi/bitstream/handle/10024/798038/Opinnaytetyo_Luoma-aho_Mika_YTC19S1.pdf?sequence=2&isAllowed=y

Appendices: The framework for creating undetectable reverse shell payloads involves several crucial steps. First, the process of creating the payload is outlined, focusing on how it is developed and obfuscated to remain undetectable. Techniques to hide the payload's true nature are employed, ensuring it can bypass antivirus detection by altering its signature or using in-memory execution methods. The script used to execute the payload is provided, along with the compiled executable file that contains the payload, ensuring seamless deployment. A man-in-the-middle attack setup is used between the target and the attacker's machine, allowing the payload to be delivered discreetly. The payload is then combined with a legitimate file or method to enhance stealth before being sent to the target machine. This process ensures efficient delivery while maintaining undetected access to the compromised system.

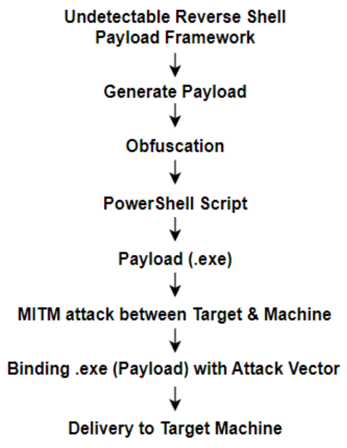
Undetectable Reverse Shell Payload Framework
Generate Payload
Obfuscation
PowerShell Script
Payload (.exe)
MITM attack between Target & Machine
Binding .exe (Payload) with Attack Vector
Delivery to Target Machine

Figure 4: Appendices

List of Faculty Proposed Changes

Automated Reverse Shell through Undetectable and Obfuscated techniques

Supervisor's Signature: [illegible]

Proposed Change	**Proposed By**	**Supervisor's Decision**
What is the target environment for the reverse shell? Will it be developed for specific operating systems (Windows, Linux, etc.), or will it be cross-platform? What methods will you use to obfuscate the reverse shell code? Will encryption, packing, polymorphism,or steganography be used to hide the code or communication? What development tools and programming languages will be used? Will the reverse shell be written in C,Python,or another language known for creating shellcode? What benchmarks will be used to determine the success of the obfuscation techniques?	Dr. Jawaid Iqbal	- OS-window - no-encryption - but packing will be used yes-in C, → Bypassing window defender. [illegible]
Sufficient work	Mr. Yawar Abbas	✓
.	Mr. Osama Raza	✓
No Suggestion	Mr. Awais Nawaz	✓
Good idea. Accepted	Mr. Tabbasum Javed	✓
Define the encryption algorithm relevant to your project.	Mr. Mueed Mirza	(out of scope) [illegible]
No	Mr. Hummayun Raza	✓

Roles & Responsibility Matrix:

The purpose of roles & responsibility matrix is to identify who will do what.

WBS #	WBS Deliverable	Activity #	Activity to Complete the Deliverable	Duration (# of Days)	Responsible Team Member(s) & Role(s)
1	Project Plan	1	Draft project plan outline		Abdul Wahab, Muneeb ur Rehman.
2	Project Plan	2	Review and Finalize project plan		Hamid
3	Project Presentation	1	Writing and Designing Presentation		Abdul Wahab, Muneeb ur Rehman, Hamid
4	Malware Analysis Techniques	1	Study malware analysis techniques to understand how malware analysts and antivirus systems detect malware on a system.		Abdul Wahab, Muneeb ur Rehman
5	Sandbox Testing Malware	1	Collect various malware samples and run them in a sandbox environment to observe how the sandbox detects their artifacts and behavior.		Muneeb ur Rehman
6	Malware Evasion Research	1	Research both traditional and advanced malware evasion techniques.		Abdul Wahab, Muneeb ur Rehman, Hamid
7	Obfuscation Tools Study	1	Explore different obfuscation methods and tools (e.g., Invoke-Obfuscation, Chimera Obfuscation).		Abdul Wahab
8	Obfuscating Reverse Shells	1	Test raw reverse shell payloads with manual and tool-based obfuscation techniques.		Abdul Wahab, Muneeb ur Rehman, Hamid
9	Malware Writing Skills	1	Learn malware writing techniques and programming principles.		Abdul Wahab, Muneeb ur Rehman, Hamid

10	PowerShell Automation	1	Develop skills in PowerShell scripting and automation, including modifying PowerShell scripts.		Abdul Wahab, Muneeb ur Rehman, Hamid
11	Hidden Process Execution	1	Execute malware as a hidden process with no user interaction or visible interface (GUI), hiding its execution from the victim.		Muneeb ur Rehman, Hamid
12	Python Obfuscation Automation	1	Automate obfuscation processes using Python scripting.		Abdul Wahab, Hamid
13	Windows API Execution	1	Write a C program that uses the Windows API to execute PowerShell commands.		Abdul Wahab, Muneeb ur Rehman, Hamid
14	Undetectable Malware Creation	1	Generate and compile undetectable malware that bypasses Microsoft Defender.	Week # 16 FYP-P1	Abdul Wahab, Muneeb ur Rehman, Hamid
15	Man-in-the-Middle (MITM) Connection	1	Identify Target Network, Configure MITM Attack Tool, Capture Legitimate Traffic		Abdul Wahab, Muneeb ur Rehman, Hamid
16	Bind Payload to Legitimate Resources	1	Analyze Captured Traffic, Modify Legitimate Resources, Test Payload Delivery		Abdul Wahab, Muneeb ur Rehman, Hamid
17	Final Report	1	Research Work	Week # 28 FYP-P2	Abdul Wahab, Muneeb ur Rehman, Hamid