GUI-Based Analysis of Penetration on Internal Networks

Muhammad Umer
01-135221-139

Afham Jamil
01-135221-004

Supervisor: Dr. Asim Ali

Bachelor of Science in Computer Science

Department of Computer Science
Bahria University, Islamabad

October 2025

Certificate

We, **Muhammad Umer** (Enrollment No. 01-135221-139) and **Afham Jamil** (Enrollment No. 01-135221-004), hereby declare that the Final Year Project titled **GUI-Based Internal Network Penetration Testing Tool** is our original work. It has not been submitted previously, in whole or in part, for any degree or diploma at this or any other institution. We further declare that all sources of information used in this work have been duly acknowledged.

Approved by . . . :

Supervisor: _____

Internal Examiner: _____

External Examiner: _____

Project Coordinator: _____

Head of the Department: _____

Acknowledgment

First and foremost, we have to thank Almighty Allah for giving us the strength, health, and with patience to complete this Final year Project. We would like to show attention to our suppressor Dr. Asim Ali for his constant guidance and technical insight and encouragement during the course of this work. His invaluable feedback helped make this project into a practical and meaningful contribution. We are also thankful to the faculty and staff of Department of Computer science, Bahria University Islamabad, For the provision of academic foundation and laboratory facilities that are required to perform our experiments. Finally, we do our utmost thanks to our friends and families for their moral support, understanding and constant motivation throughout this journey.

Abstract

Internal computer networks are behind modern organizations and host critical services, sensitive information and internal means of communication; However, these networks are often the victim of attackers exploiting misconfigurations, unpatched weaknesses and poor internal security practices. While powerful open source penetration testing tools e.g. Nmap, Metasploit, Wireshark/Tshark, Bloodhound exist, they are mostly command line driven requiring advanced technical skills, and making them relatively hard to adopt for smaller organisations and beginners This Project brings a GUI-Based Internal Network Penetration Testing Tool that combines a variety of already successful security tools under one unified and easy-to-use interface. The system is implemented in the Python language using the PyQt5 framework and is developed to fix the Kali Linux. It supports the host and port scanning through Nmap, vulnerability exploiting through the Metasploit Framework, Tshark packet capturing, optional Active Directory path analysis using BloodHound, and centralised logging using report generation (SQLite, and export to the web in the form of an .html file). The tool that has been implemented automates typical penetration testing workflows, such as scanning internal hosts, finding exposed services, searching and launching relevant Metasploit modules, network traffic monitoring and risk-aware report generation. A controlled virtual lab was set up using a Kali Linux (attacker machine) and Windows 10 victim system that has a vulnerable Icecast service running on it. The tool was able to successfully detect the vulnerable service - it automatically identified the correct Metasploit module, and set a working exploit succeeded in getting a Meterpreter session Testing and evaluation revealed that the system makes the system more usable, lowers the dependency on manual command line operations, and provides an integrated view to internal network security posture. The work involved in the project

proves that complex penetration testing flows can be encapsulated within a GUI, making it possible to more internally assess networks easy and affordable, particularly for small-sized organizations and cybersecurity students.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Organizations in the contemporary digital world have become highly dependent on in-house computer. networks to monitor the activities, promote communication and store knowledge. These Cloud services, commonly associated with networks are vital to productivity but are becoming major targets of cyber attackers. Research has shown that a large percentage. Cyber incidents of small and medium enterprises occur every year, and internal. one of the most common being network breaches. Internal network will have a common routing network which consists of routers, switches, servers, printers, and employees. workstations. Such systems use local area networks (LANs) or wide. area networks (WANs) and typically secured with perimeter security. as intrusion detection systems and firewalls. Nonetheless, wrong settings, poor. patch control, and bad access control often leave exploitable ports in the system. Organizations also conduct penetration testing in order to preempt these threats. known as ethical hacking. Penetration testing is one that mimics cyberattacks to locate and identify. address security holes before they can be used against. It evaluates how well systems resist attacks like unauthorized access attempts, exploitation of the vulnerabilities, lateral movement, and increase of privileges. The majority of the conventional penetration testing tools, though, are command-line interactive. and profound technical experience. This renders them less available to the entry-level IT personnel. companies that do not have special security units. These tools were manual in nature. also causes time-consuming and allows overlooking the process. To curb these issues, this project will come up with a full user friendly project. graphical user interface (GUI) tool which integrates numerous necessary cybersecurity. Nmap,

Metasploit, Tshark, and BloodHound among others. By unifying multiple tools using a single platform, the suggested system makes it less complicated, more transparent, and. enhances quicker, more formal internal network penetration testing.

1.2 Problem Statement

Internal networks have been one of the most common attack surfaces because of poor system configurations, lack of monitoring and effective penetration testing practices. The actual security cases in the real world have demonstrated the fact that a single vulnerability left unchecked in an internal service can result in severe consequences such as data breach, downtime, or ransomware attacks. Despite the existence of a few effective open-source penetration testing tools, many organizations have difficulties with conducting effective security tests. This challenge is even greater with a lack of trained cybersecurity specialists, the difficulty and high learning curve of command-line tools, and the use of manual testing protocols that are easy to make mistakes on. Also, the integration is inadequate between the various tools, and the absence of a single interface further makes the workflow of the testing more difficult. Consequently, the internal weaknesses can still be critical and go unnoticed. These have identified a need to have a simplified, integrated, GUI based penetration testing framework that can be used to automate routine tasks, provide a real-time analysis and generate meaningful internal network security reports to assist in timely remediation efforts.

1.3 Objectives

This is primarily aimed at producing a comprehensive and viable internal network penetration testing tool, in the form of a GUI. The system is meant to increase accessibility to and ease penetration testing through the integration of several popular open-source security tools into a single graphical interface. Such tools are Nmap which is used to scan the network, Metasploit which is used to exploit the network, Tshark which is used to capture and analyse packets, and BloodHound which is used to analyse the Active Directory. Automation is also another vital goal of the project

because the majority of testing functions like port scanning, enumerating the services, executing the exploits as well as monitoring the traffic will be done automatically without depending on manually typed commands. Another aspect of risk evaluation on which the project is aimed is a scoring system that assists in identifying and prioritizing the high-risk services and vulnerabilities. Moreover, the system offers reporting capabilities to enable users to export logs, risk scores, as well as testing results to facilitate security decision-making. In general, the purpose of the tool is to enhance efficiency by saving time, decreasing human error and simplifying the process of complex testing, as well as having the ability to expand to accommodate internal networks of various size and configuration.

1.4 Scope

This project will involve only the internal network penetration testing in a controlled or a laboratory setting. The system is aimed at incorporating basic open-source security utilities, such as Nmap to scan the network, Metasploit to exploit it, Tshark to record and capture packets, and optional BloodHound integration to analyse an Active Directory. The program is written in Python and the Qt5 framework and is meant to be used in the Kali Linux. The methods of logging and report generation are done with the help of an SQLite database and report generation through HTML. Under the project scope, one of the real-life vulnerabilities, namely the Icecast service exploit, is shown to be working in a virtualized test setup to confirm the weapon that the tool can work. This project is not on web application penetration testing, cloud security testing and mobile application exploitation. Also, the tool should be used in its authorized environment only to conduct educational and ethical testing purposes.

2. Literature Review

2.1 Introduction

The chapter is a review of available penetration testing tools, GUI based security platforms. and pertinent research works pertaining internal network security. The aim is to define the advantages and weaknesses of existing strategies and to inspire the necessity. the proposed penetration testing GUI based tool.

2.2 Overview of Existing Tools

In order to comprehend the discrepancy between the existing practices in penetration testing, it is necessary to assess them. Authenticate the tools widely used in the industry. Each tool plays a specific part in the security evaluation. The tools that are incorporated in the are as follows. proposed system:

2.2.1 Metasploit Framework

Metasploit is a popular framework which is aimed at penetration testing and vulnerability exploitation. It has a large collection of exploits, payloads, encoders, and postexploitation courses. It also enables it to integrate with different databases. and automation by the msfconsole or scripts. Metasploit allows security professionals to emulate attacks on different platforms and evaluate the performance of the different platforms. defensive measures.

- **Primary Functions:**Exploit development, payload delivery, vulnerability val-

idation and privilege escalation simulation.

2.2.2 Nmap

Nmap is an effective open-source application in network discovery and security auditing. diting. It can be able to discover the hosts, detect version of a service, operating system. fingerprinting, and evasion of firewalls. Nmap is compatible with sophisticated scanning. e.g. stealth scans (SYN scan), timing options, and output formatting. It provides a graphical representation of the network and assists testers to detect possible attack vectors.

- **Primary Functions:**Host and port scanning, OS detection, firewall rule detection, service enumeration.

Even though Nmap is very flexible, complex command-line arguments are to be remembered. ments.

2.2.3 Wireshark and Tshark

Wireshark is one of the top network protocol analyzers that enables real time packet capture. and offline analysis. It assists in in-depth examination of many protocols and is powerful. ful filter and search functionality. A graphical schedule of network traffic, and Wireshark assists in identifying the abnormalities like ARP spoofing which are protocol specific. distorted packets, or plain text sensitive data transmissions.

- **Primary Functions:** Packet capture, protocol analysis, network troubleshooting, and anomaly detection.

2.2.4 BloodHound

BloodHound is a software that is used to scan Active Directory (AD) environments and expose them. secret routes to privileged status. Neo4j graph can be used in conjunction with graph theory. database, it assists in locating vulnerable areas in

access control and gives graphical representation. tion of relationships which can be subjugated by attackers. BloodHound is particularly applicable in red-team testing and to lateral movement simulation.

- **Primary Functions:**path discovery of AD privilege escalation, attack graph. visualization, and plan strategy of lateral movement.

2.2.5 Other GUI Tools

here are GUI based tools available such as:

- **Armitage:** Metasploit GUI, allows visualization of both targets and exploits. but is no longer actively cared over.
- **Zenmap:** Nmap frontend graphics; handy in scanning, but not supported. exploitation or packet capture.
- **Sparta:** Lots of enumeration and scanning, but not penetrated to the core. integration.

2.3 Research on Automated and GUI-Based Penetration Testing

Recent studies note that there is an increased requirement of automation in penetration testing to minimise on human error, as well as enhance total coverage of security tests. It has been demonstrated that GUI-based security tools are much easier to use as the learning curve is reduced and the testing can be done more consistently and repeatably. In spite of the advantages, the majority of the available academic and commercial solutions are likely to be restricted to specific aspects, including vulnerability scanning alone, large enterprise or corporate setting, or web application testing and SIEM-based monitoring. Consequently, it has not explored a lot of research, and has no extensive solutions that have assimilated seminal offensive security instruments like Nmap, Metasploit, and Tshark into a solitary lightweight graphic interface. Moreover, such tools as BloodHound have been regarded as discrete entities instead

of being integrated into a single platform of internal network analysis. This is an indication of a distinct research gap that is expected to be filled by this project since these solutions are not integrated, GUI-based internal network penetration testing methods.

2.4 Research Gap

There are a number of research works which have reported effectiveness of automated and GUI-driven penetration testing platform in enhancing the overall security testing procedures. Results reported in IEEE Access and other journals in the domain of cybersecurity show that GUI-based tools can considerably minimize the amount of human error and in some instances lead to a higher testing efficiency of up to 40 percent compared to fully manual methods. Nevertheless, these benefits are not accompanied by a significant share of the absence of all-in-one platforms that would incorporate into one convenient interface popular tools of open-source penetration testing and be applicable to the internal network testing, especially in the case of small and medium-sized businesses. Most of the available solutions are largely based on vulnerability scanning and do not go further on post-exploitation analysis and organized risk reporting. This project seeks to overcome these shortcomings by integrating the necessary security tools into a single interface, automation of the scanning processes and reporting processes, and usability with different degrees of expertise by the end-users. The system further includes fundamental risk scoring and mitigation assistance to help in prioritization of security concerns. In general, the proposed project will help fill this research gap because it will provide a specific, GUI-driven penetration testing tool that is adapted to internal networks and able to automatize the most common attack processes within a controlled and monitored setup.

2.5 Summary

This chapter discussed the present scenario of the penetration testing tools and GUI-based frameworks. It has pointed to the positive and negative sides of current so-

lutions. and found that there exists a distinct requirement of an integrated, user-friendly, and automation-oriented. internal network security assessment tool. The following chapter describes the method- ology to design and execute such a system.

3. Methodology

3.1 Introduction

The chapter specifies the methodology that will be used to design, develop and evaluate. the GUI-Based Internal Network Owing Testing Tool. The approach follows a designed software development life cycle that is specific to security tools integration.

3.2 Requirement Analysis and Design

This project has been able to obtain system requirements after examining the common penetration testing tasks, which included network scanning, exploitation, traffic analysis, and report generation. The selection of open-source security tools based on this assessment is based on the fact that, it is free to use, stable, and has a reputation in the community of cybersecurity. Critical considerations were made regarding the needs of the expected users such as the cybersecurity students, IT teams and small to medium sized businesses to make sure that the system is practical and easier to use. The system was constructed on the basis of the layered architecture in order to emphasize the flexibility and further upgrades and make the interactions in between the various components explicit and enable the extension of the tools by making them abstract.

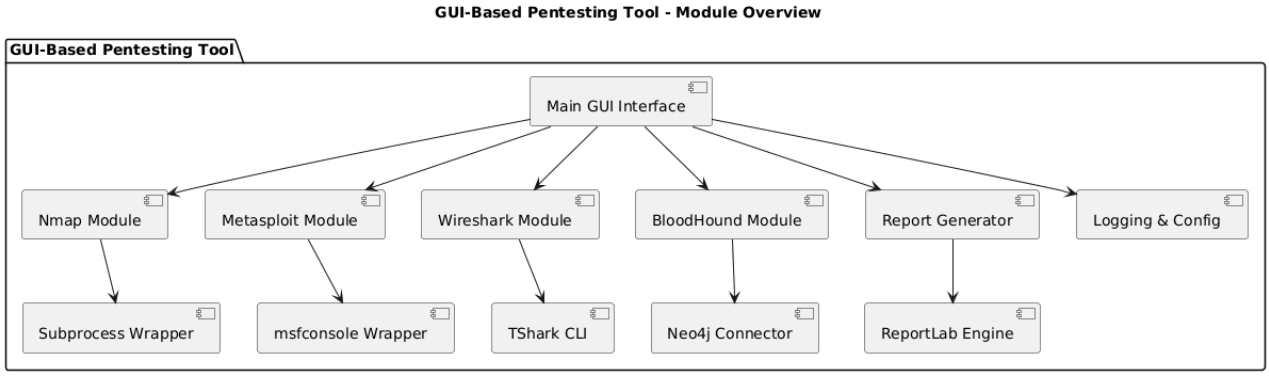

Figure 3.1: System Architecture Diagram

3.3 GUI Development

To make the system responsive and user-friendly, the graphical user interface was written in Python programming and Tkinter, PyQt to have the necessary design. It was made simple in nature and the interface was easy to use and do complex penetration tests with little interaction. Interactive elements like buttons, drop-down menu, logs and status indicators were included to give real time feedback when executing the tool. Also, the visuals and tooltips were included to help the user get better understanding and increase the usability. Built-in logging and report generation are also incorporated in the system to make sure all activities and outcomes are well recorded to be later analyzed.

3.4 Tool Integration

The system was integrated with Nmap application via the means of subprocesses and commands execution to conduct real-time network scanning with customizable parameters being chosen in the GUI. The inclusion of Metasploit functionality was achieved by integrating msfconsole commands through automation which allowed users to search, configure, and execute the exploit modules within the graphical interface without having to interact with the command-line interface. Tshark was associated with the Wireshark functionality by implementing system calls to facilitate live network traffic capturing and analysis in a controlled environment. Besides, Data

importation and exportation mechanisms were also added as well as the support of BloodHound, which was analyzed and visualized with Neo4j graph database to find possible paths of Active Directory privilege escalation and attacks.

3.5 Automated Penetration Testing

Port scanning, service enumeration, simple vulnerability checks, and other routine penetration tests were automated in the system in an attempt to cut down manual work. The logic of scripting was put in place to transform the multi-step testing processes into simplified operations that need little user inputs. The app also gives real time scan progress tracking and logs in detail to maintain transparency in the implementation process. To assist the novices, scan presets were provided, whereby new users could complete ordinary security tests without the technical expertise.

3.6 Risk Assessment and Reporting

The system uses the Common Vulnerability Scoring System (CVSS) in identifying threats and vulnerabilities to establish the level of severity of the reported threat and vulnerability. A report generation system was developed that is automated to generate structured security reports containing threat classifications, visual summaries and proposed prevention measures. Such reports are easy to share and document using the HTML format. Moreover, the system keeps elaborated audit logs containing timestamps to document all the user actions and tests to facilitate accountability and traceability.

3.7 Testing Methodology

The developed tool was functionalized by the virtualized environment simulating a real internal enterprise network. The system was tested in a normal usage state to determine its performance, usability and the stability of the system. Various testing methods were used such as the unit test, integration test as well as full system test to

make sure that each of the modules worked properly as well as in the entire system. According to the testing, the integration layers were further refined in order to meet low execution latency and high accuracy when interacting with the tool and processing data.

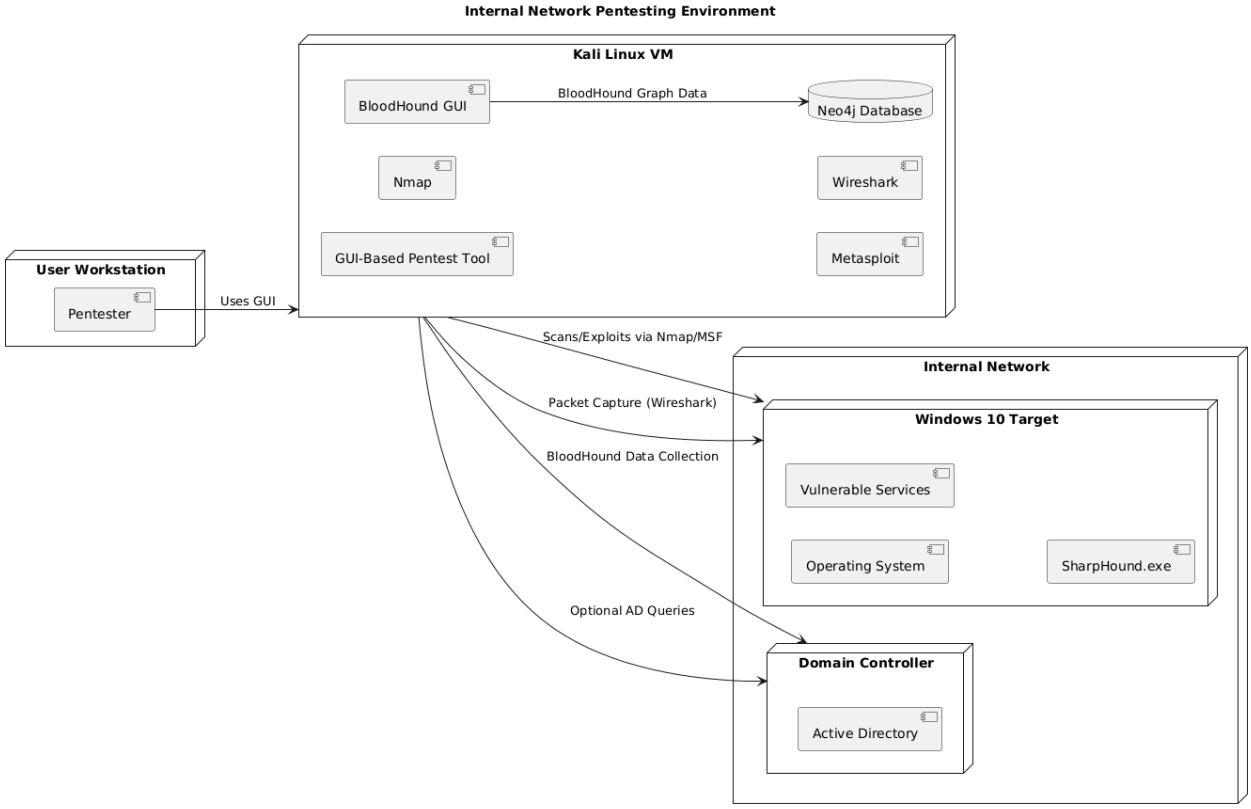

Figure 3.2: System Architecture Diagram

3.8 Deployment

To be deployed, pyinstaller was used to make standalone installation packages to make installation and execution easier. The application was made to be compatible with windows and Linux operating systems making it usable in more environments. A detailed documentation was made on installation procedures, user operation procedures and maintenance procedures to be followed in future. Besides this, all the dependencies required were bundled with the application and troubleshooting manuals were created to enable end users to solve the common hustles effectively.

3.9 Summary

The entire procedure of how to construct the GUI-based penetration was detailed in this chapter.

4. System Design

4.1 Introduction

This chapter gives out the system design of the GUI-Based Internal Network Penetration Testing Tool. The design process is aimed at modularity and tool integration. support automation, usability, and support automation. Specifically, current cybersecurity solu- tions underlines the need of GUI based penetration testing systems to minimize. elaborate and improve working performance. A 2023 study on GUI-based pene- traction testing platforms prove that visual interfaces offer a considerable decrease in user. error and faster scanning and enumerating operations (Scribd – GUI-Based Penetra- tion Testing Tool). On the same note, the scholarly literature reveals that integrated frame- works enhance consistency of tasks and reduce the context-switching overhead (iieta.org – SecureGUI Tool). Results of these findings are included in the design of the proposed system. to make a sensible, research-oriented strategy that can be used to assess the internal network. comments.

4.2 System Architecture

To provide modularity, maintainability, and effective operation, the system architecture of the proposed tool is split into several layers that are distinctly defined. The User Interface Layer is created based on PyQt5 and offers the user a friendly graphical interface through which they can easily scan the network, exploit it, capture traffic and even generate reports. The Integration Layer serves as an interface between the GUI and the security tools underneath by doing the communication between the GUI and

the security tools via QProcess and make sure that the command is safely executed and the tool outputs are appropriately handled. The Tool Layer is a collection of inbuilt penetration testing tools, such as Nmap, Metasploit Framework, Tshark, and BloodHound, and they execute the fundamental security test procedures. The Data Layer is created with the help of the SQLite database that will store scan logs, exploitation information, timestamps, and risk scores calculated to be used in the future. Lastly, it is the mandate of the Reporting Layer to produce structured HTML reports that provide formatted results and embedded findings to allow clear documentation and analysis of security assessment.

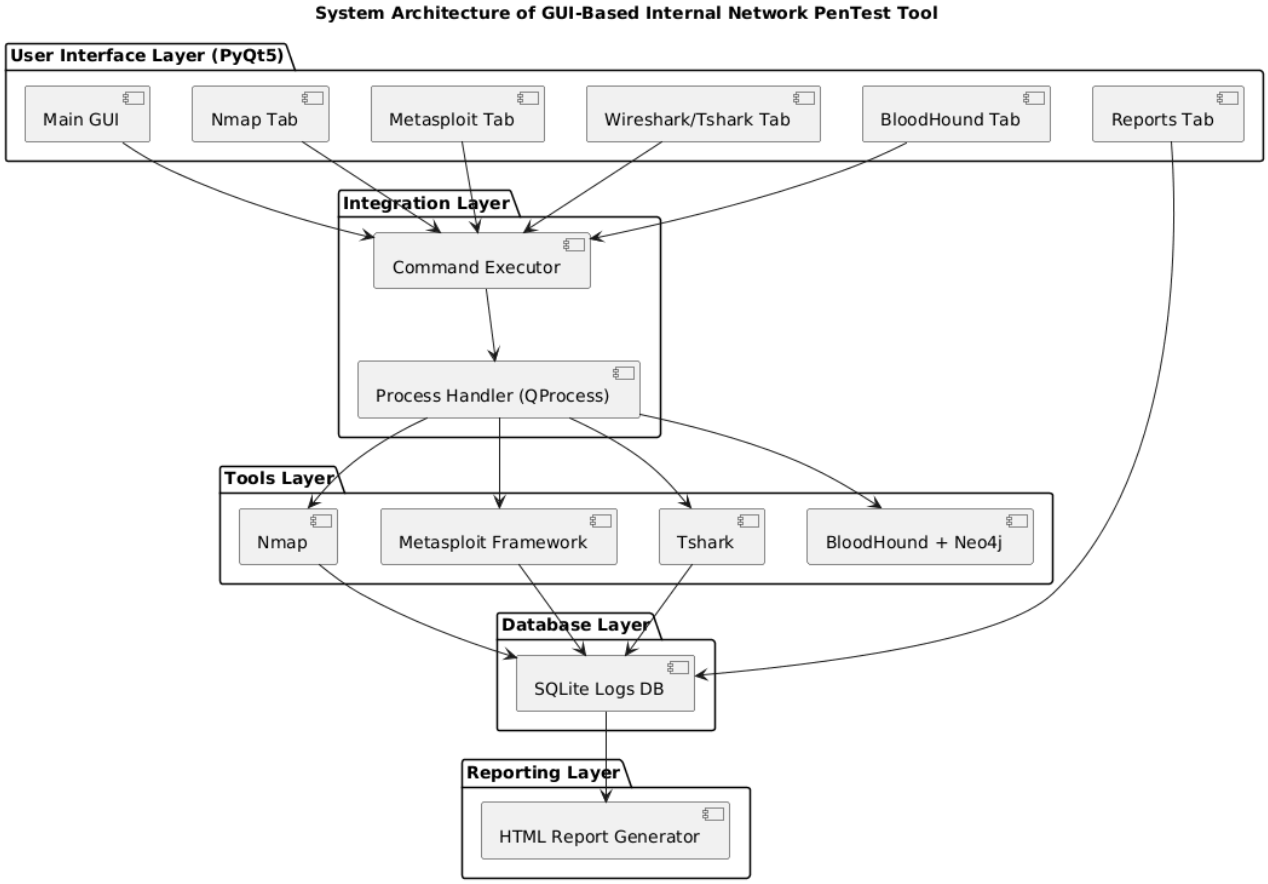

Figure 4.1: System Architecture Diagram

4.3 GUI Design

The GUI will be in such a way that it limits complexity and barriers to entry (especially in matters of novice). cybersecurity practitioners. It has been shown that pentesting tools based on a GUI are sig- significantly enhance the understanding of the users and lessen the complexity of operation (Scribd). Research). The tool comprises of five principal tabs:

- Nmap Scan Tab
- Metasploit Tab
- Tshark Capture Tab
- BloodHound Tab (optional)
- Reports Tab

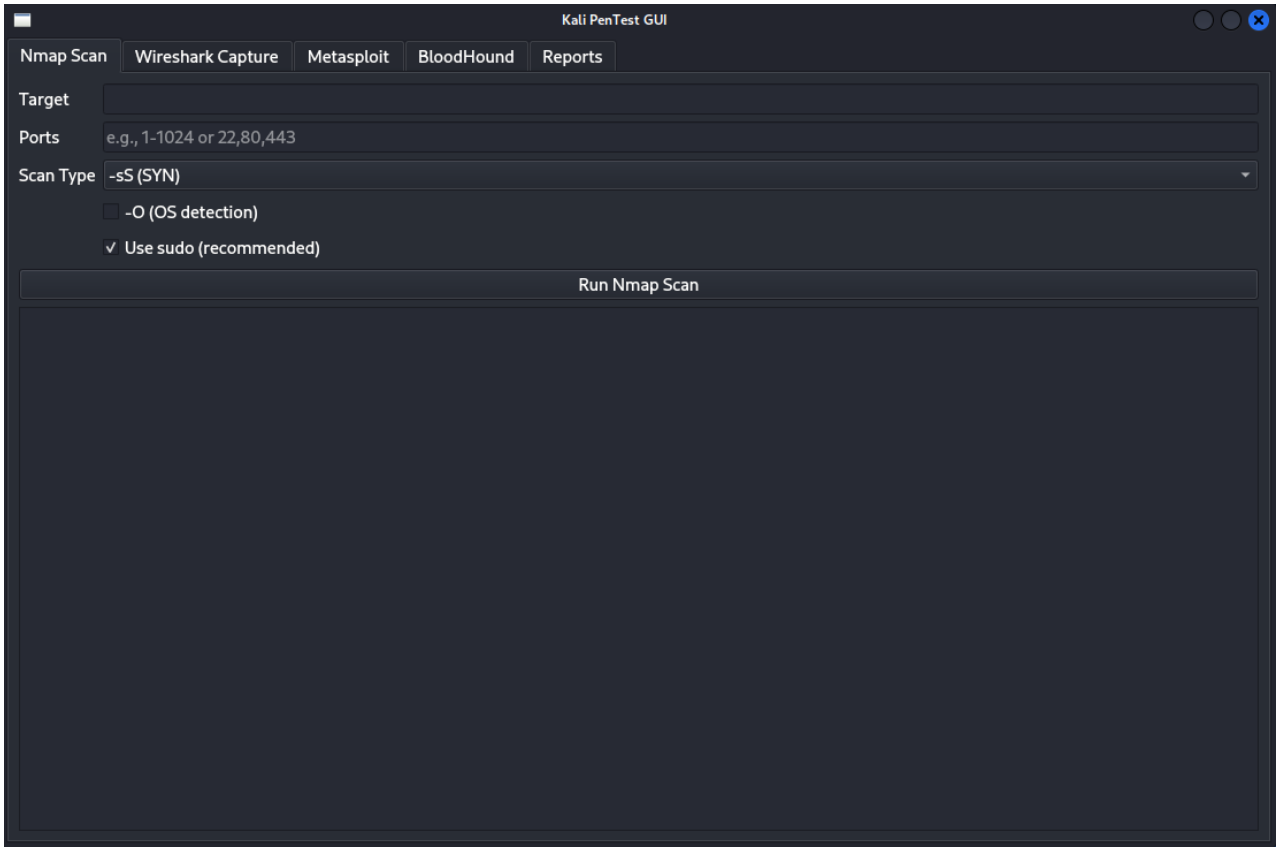

Figure 4.2: System Architecture Diagram

Each tab has clearly labeled fields and a dedicated output pane.

4.3.1 Nmap Module

Responsible for:

- Validating target input.
- Building scan commands.
- Executing Nmap using QProcess.
- Parsing open ports and assigning a risk score.
- Logging results.

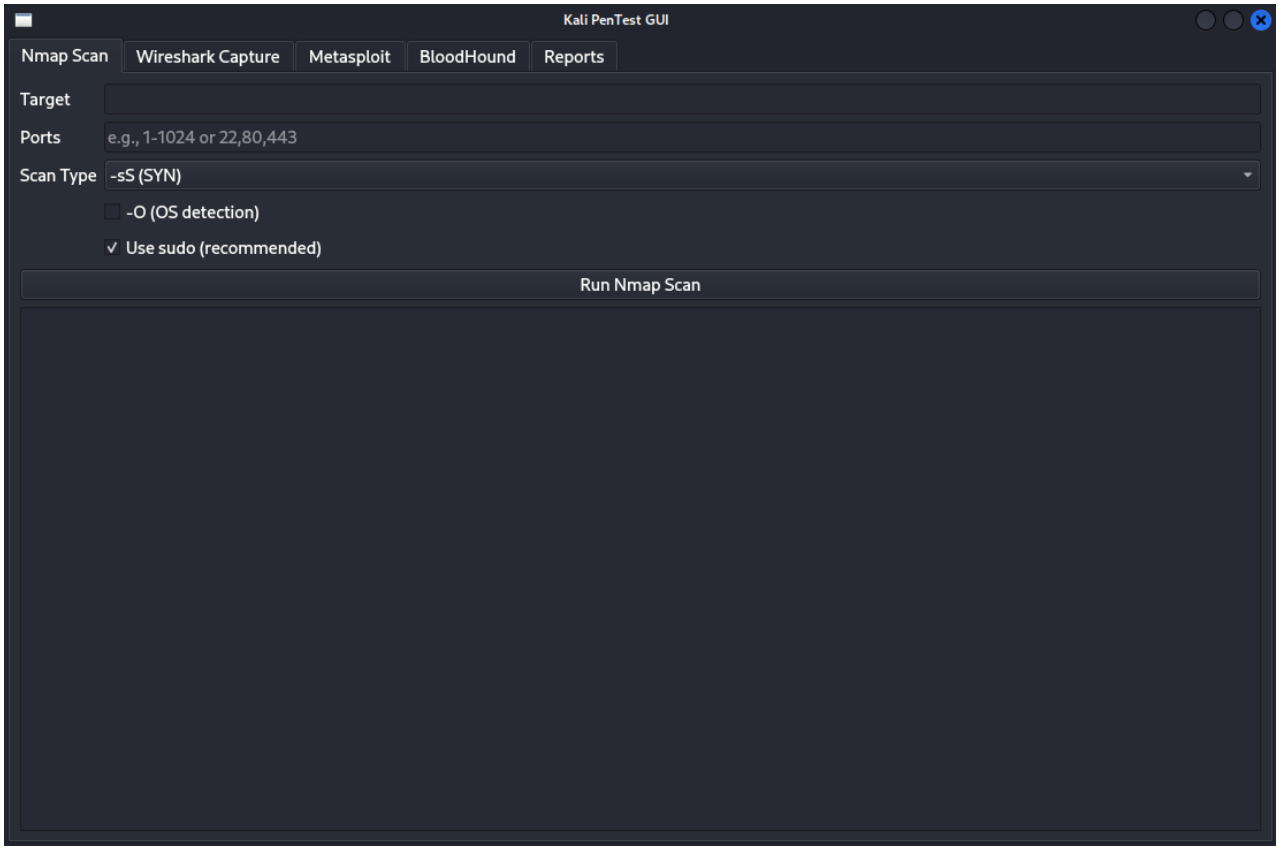

Figure 4.3: System Architecture Diagram

4.3.2 Metasploit Module

Responsible for:

- Searching for modules by keyword (e.g., “icecast”).
- Extracting exploit paths using regular expressions.
- Allowing the user to set RHOSTS, RPORT, and PAYLOAD.
- Running exploitation commands in `msfconsole`.
- Logging exploit output and session information.

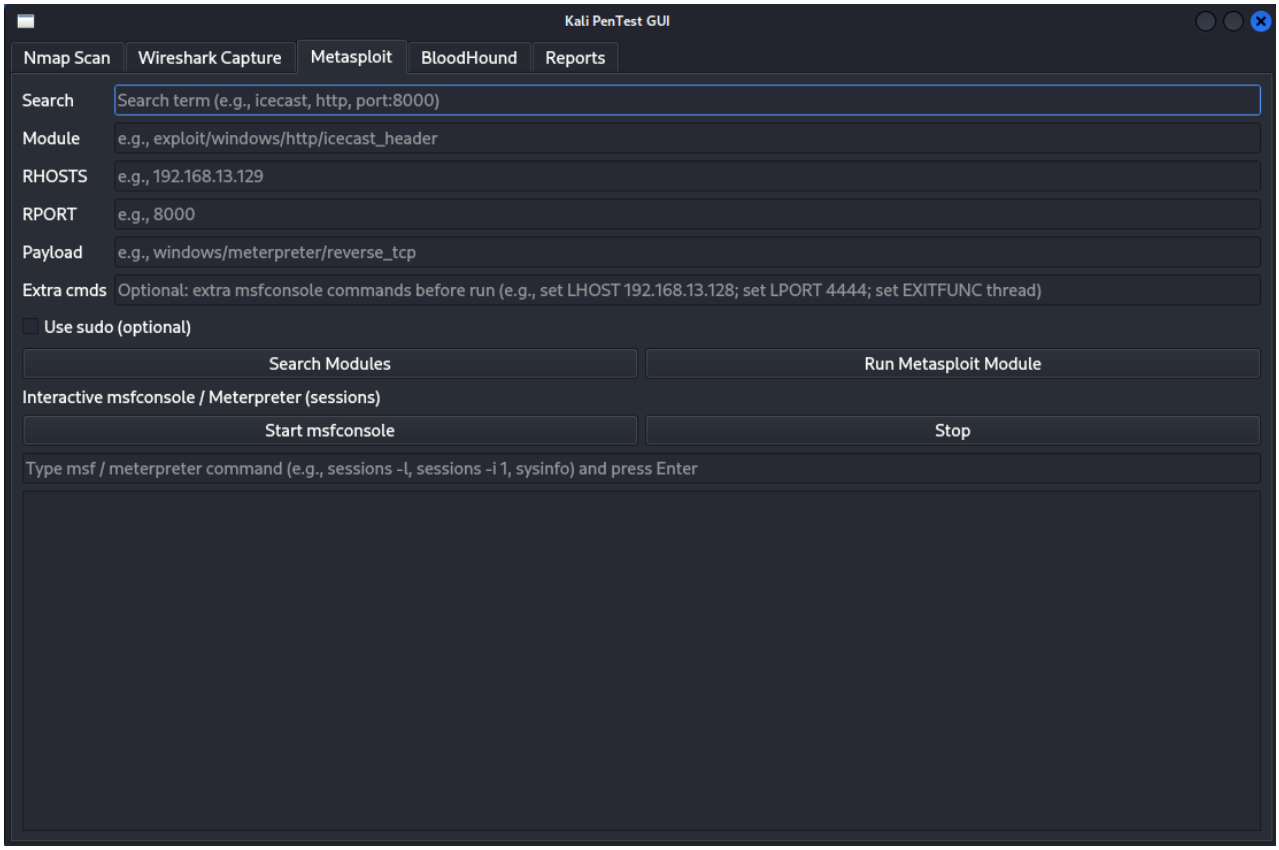

Figure 4.4: Metasploit

4.3.3 Tshark Module

Provides:

- Interface discovery via `tshark -D`.
- Packet capture using appropriate filters and duration.
- Live streaming of packet data to the GUI.
- Logging captured output.

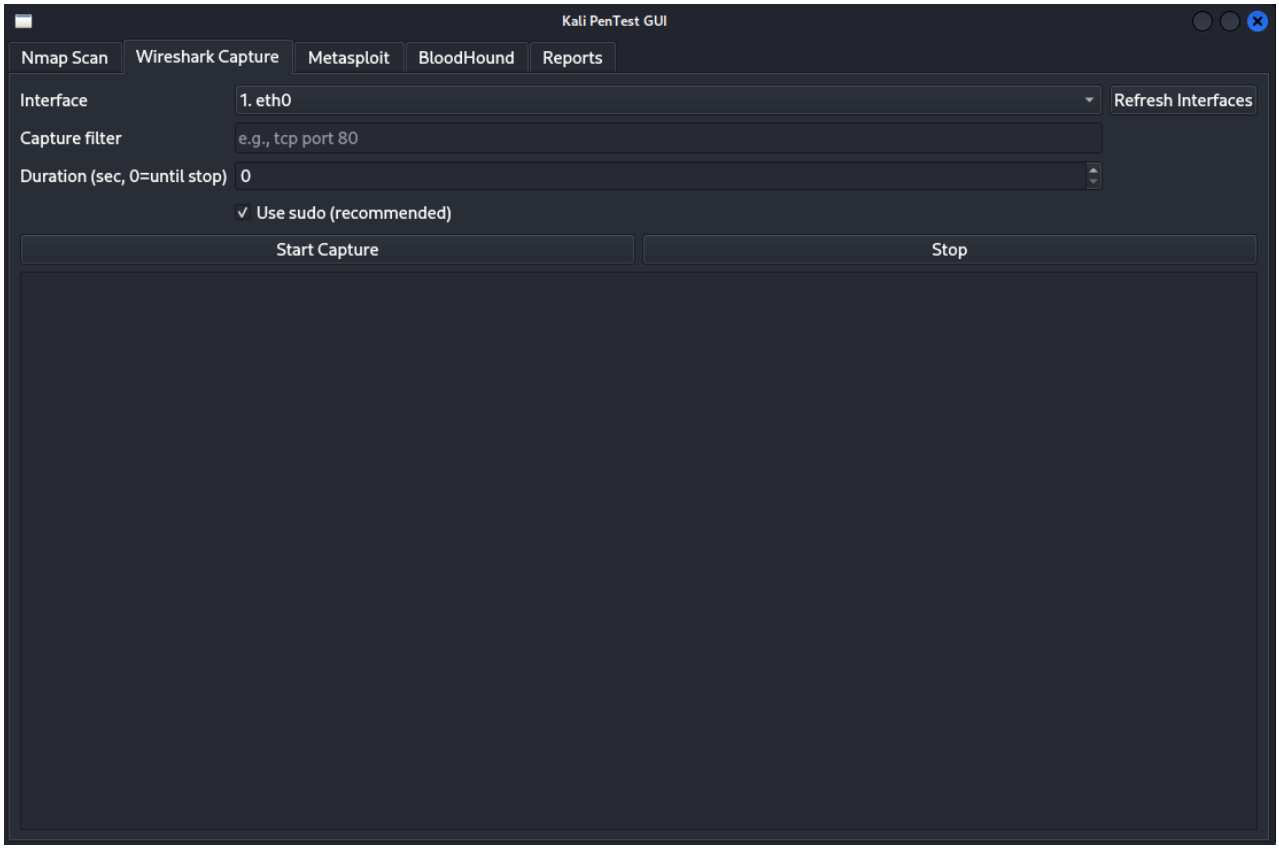

Figure 4.5: Tshark Module

4.3.4 BloodHound Module (Optional)

Supports:

- Domain and credential input.
- Running `bloodhound-python` ingestor.
- Saving ingested data for offline AD analysis.

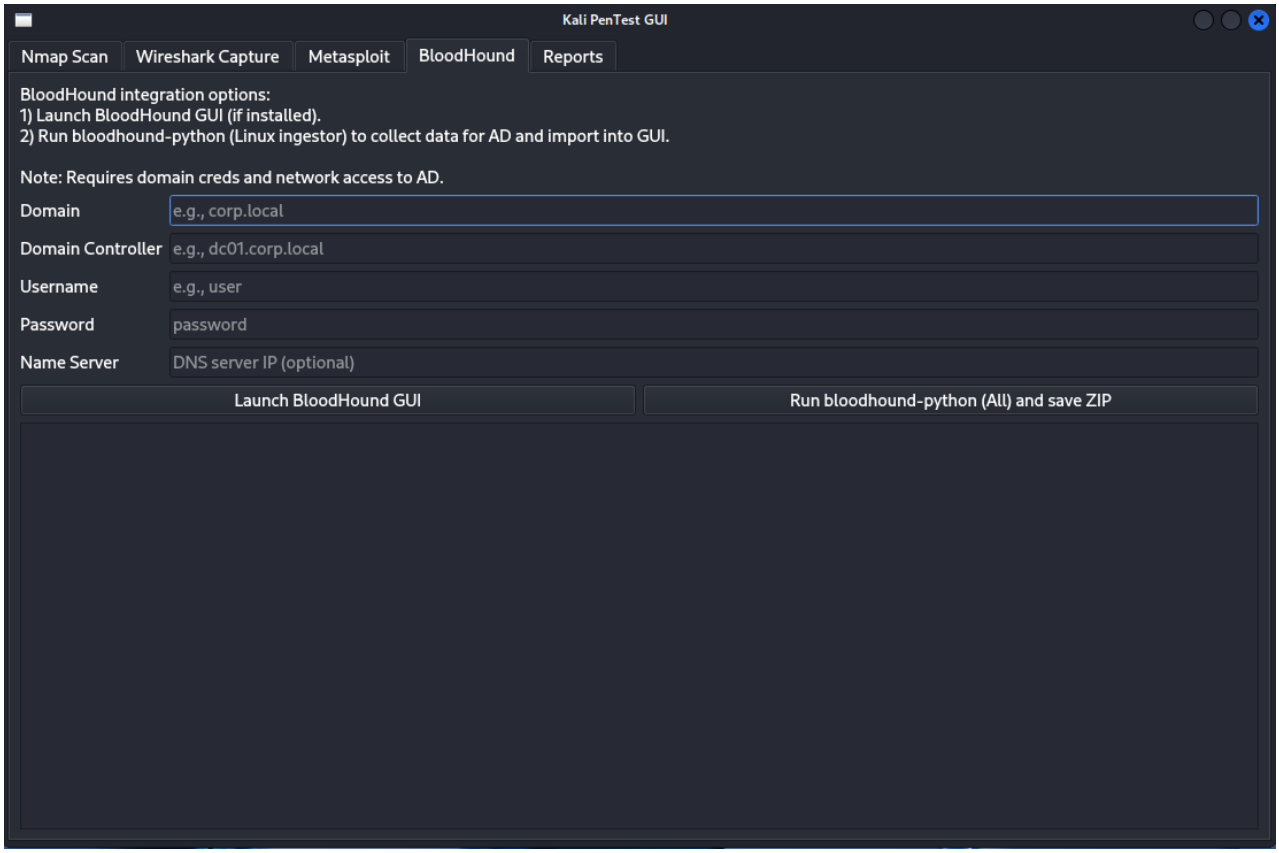

Figure 4.6: BloodHound Module

4.3.5 Logging and Reporting Module

Writes all relevant outputs to an SQLite database and generates HTML reports with:

- Tool name.
- Target.
- Parameters.
- Risk score (if applicable).
- Tool output.

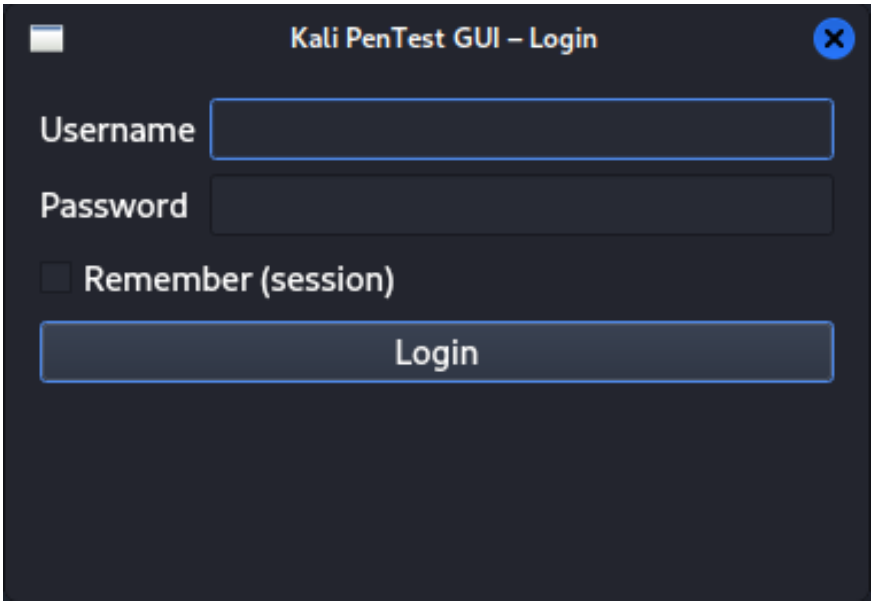

Figure 4.7: Logging and Reporting Module

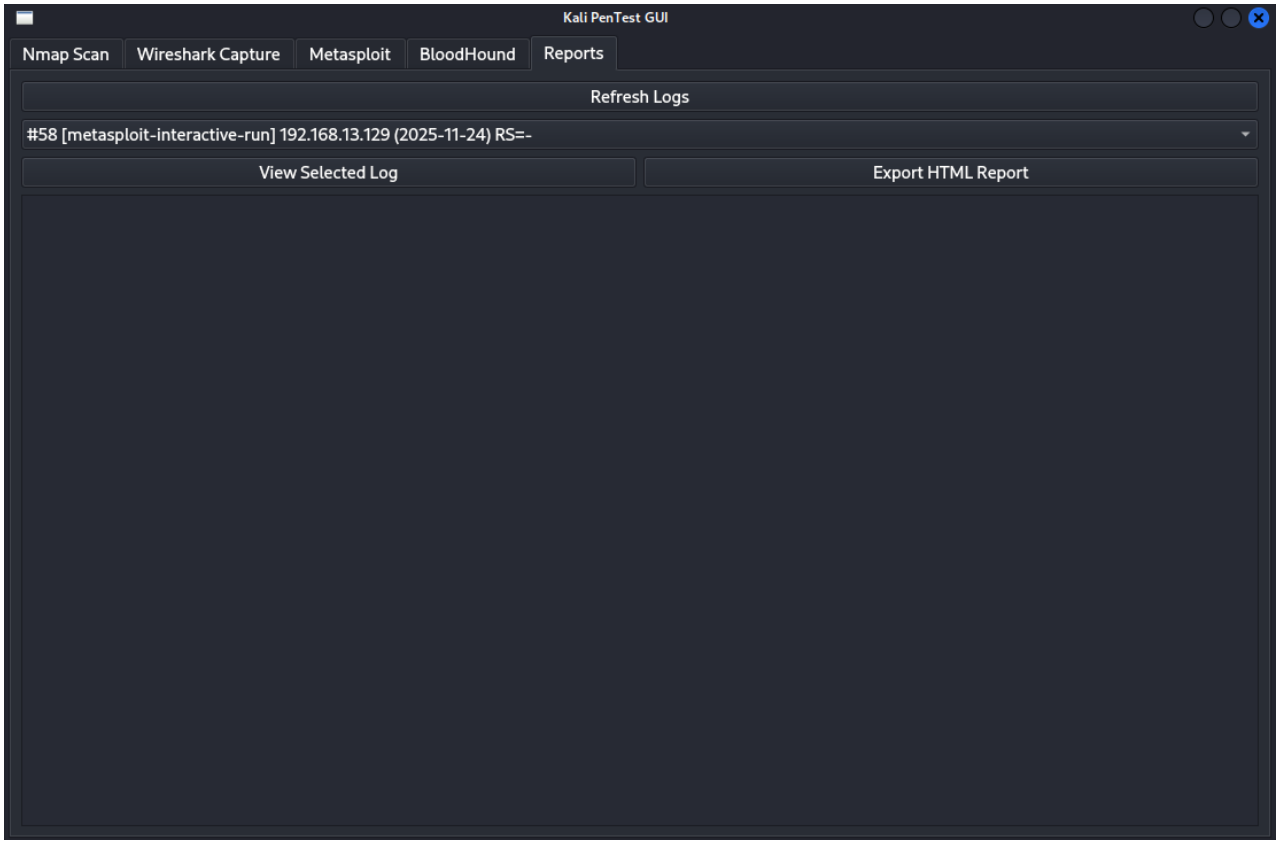

Figure 4.8: Logging and Reporting Module

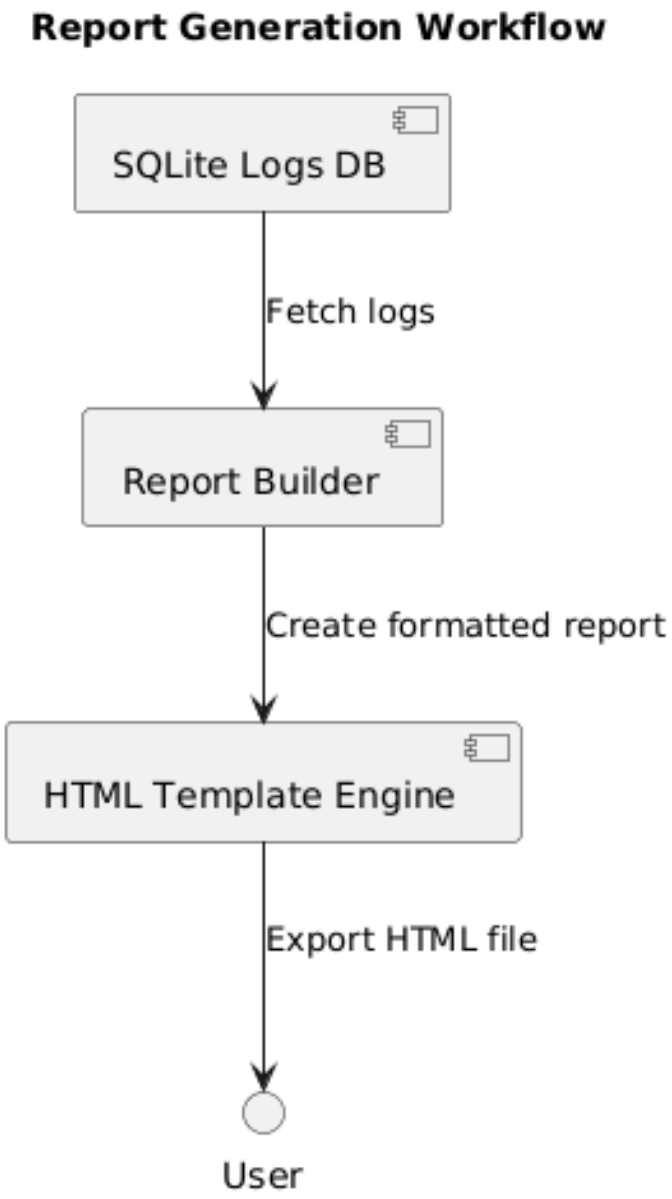

Figure 4.9: Report Generation Workflow

The proposed system report generation process is reflected in figure 4.9. The workflow begins with the retrieval of the object of the SQLite logs database with scan and exploitation data that were previously stored. The report builder then works on this information and arranges the data into a structured format. The formatted data is sent to the HTML template engine that creates a readable and well-formatted report. Lastly, the report is generated under HTML file so that users can gain access to it. This is an automated mechanism that ensures proper recording of the penetration test results with minimum efforts by the user.

4.4 Module Design

Data Flow Design The data flow architecture of the GUI-Based Analysis of Pene-

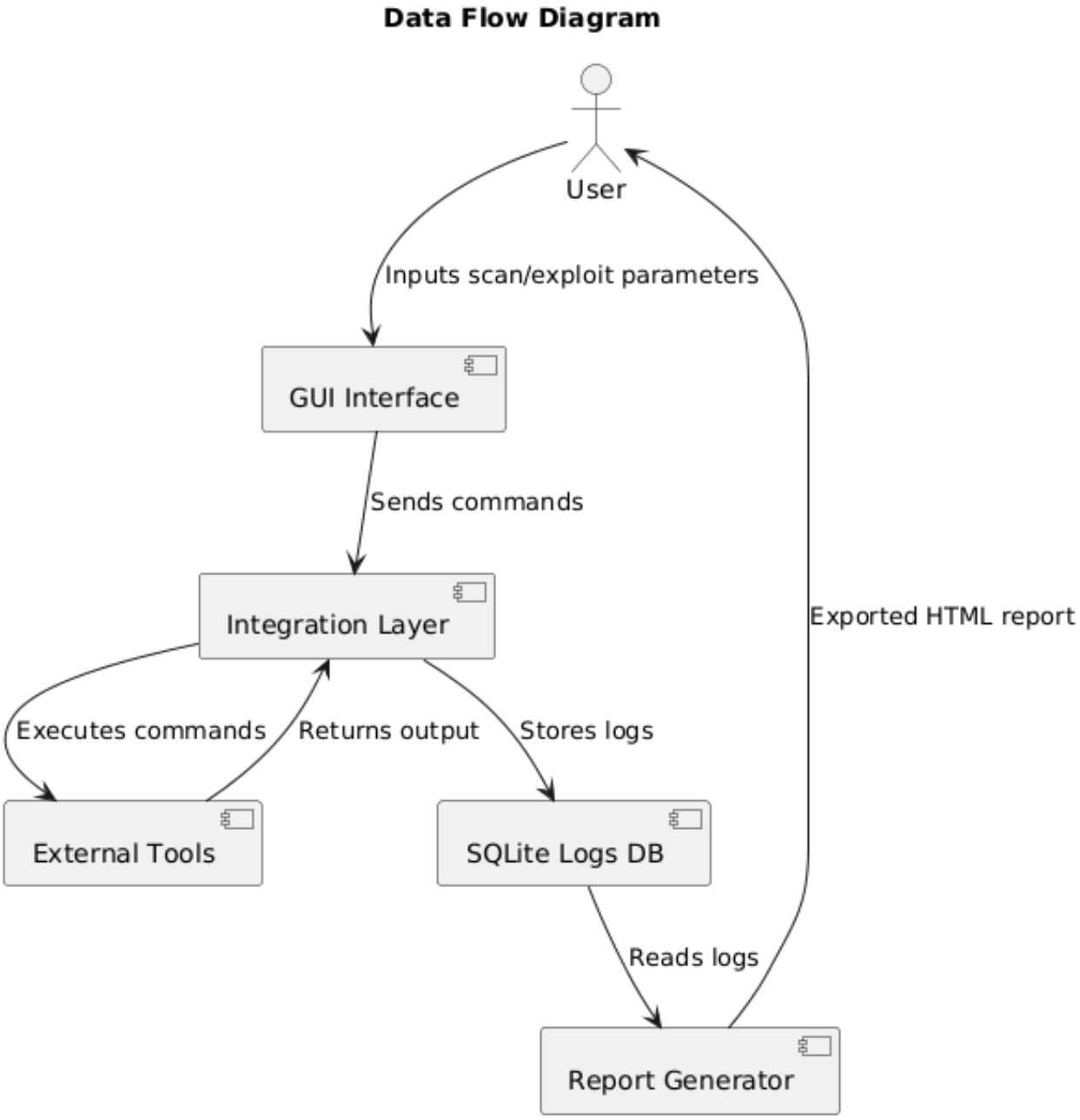

Figure 4.10: Module Design

tration on Internal Networks is shown in the following figure (4.10). It starts when the user input scan or exploit parameters by the user interface. These inputs are sent into the integration layer that performs the necessary commands with the help of external security tools like Nmap or Metasploit. The results produced by the tools are sent

back to the GUI to be displayed and the appropriate results and logs are saved to the SQLite database. Report generator reads the logged data and generates a report that can be exported to the user, when the user requests a report. This architecture will facilitate smooth flow of data, adequate logging and an automated reporting system in the system.

4.5 Tool Integration Design

The combination of external tools is done through QProcess to take up output in real time. This approach is based on the best practice suggested in the literature of automated security systems. (iieta.org — SecureGUI tool).

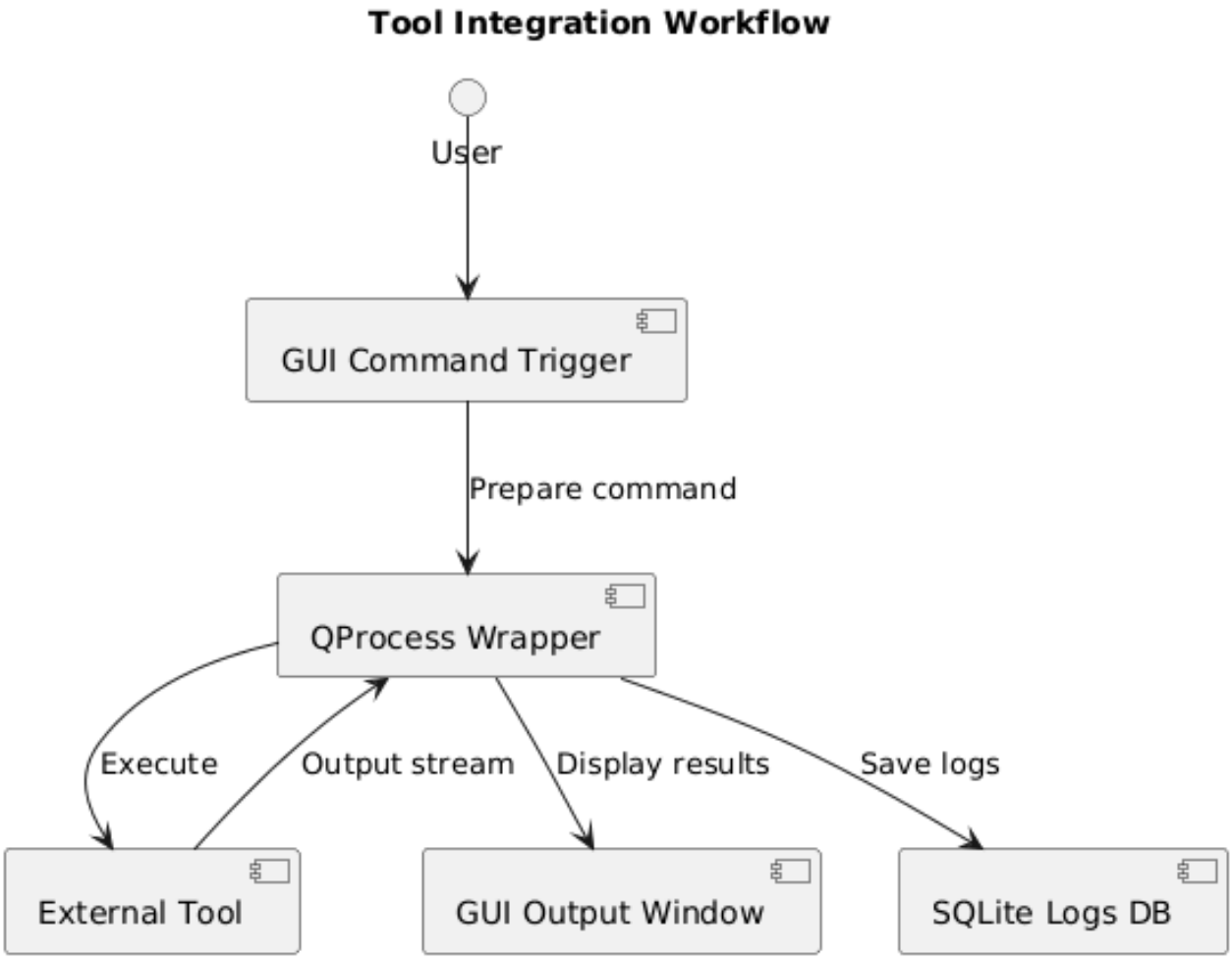

Figure 4.11: Tool Integration Design

Figure 4.11 shows the external security tools as they are integrated into the system.

This begins when the user initiates a command with the help of the GUI. The system constructs the necessary command and sends it to the QProcess wrapper which executes the chosen external tool. The output created in the process of execution is concurrently shown in the GUI output window and logged in the SQLite logs database. This workflow has limited execution, real-time feedback to the user, and results logging to be used in reporting.

4.6 Use Case Modeling

Representative use cases are summarized below.

Use Case: UC-01 – Login

Table 4.1: Use Case UC-01 – Launch and Login

Use Case: UC-01 – Launch and Login	
Use Case ID	UC-01
Use Case Name	Launch and authenticate user into the GUI.
Primary Actor	Penetration Tester / Security Analyst
Preconditions	Application is installed; user account exists in SQLite database.
Postconditions	User is logged in and main GUI tabs are displayed.
Main Flow	1. User launches the application. 2. Login window appears. 3. User enters valid username and password. 4. System verifies credentials. 5. Main GUI dashboard opens.
Alternate Flow	Invalid credentials trigger an error message and login is denied.

The Use Case UC-01 explains the process of logging in and launching an application of the GUI-Based Internal Network Penetration Testing Tool. This use case describes the way, in which an authorized user opens the application and enters the system. The penetration tester or security analyst is the main participant and the precondition is that the application is already installed and the user credentials are already stored in the SQLite database. The primary flow illustrates the standard procedure of logging in, beginning with the opening of the application and ending with a successful log in and entering the main dashboard. When successful the GUI tabs can be used to carry out the tasks of penetration testing. The post-

condition is the best way to ensure that the user is a verified user, and it has shown the main interface. The alternative flow is used to deal with the invalid login attempts through displaying an error message and rejecting the access to ensure a basic access control and system security.

Use Case: UC-02 – Perform Nmap Scan

Table 4.2: Use Case UC-02 – Perform Nmap Scan

Use Case: UC-02 – Perform Nmap Scan	
Use Case ID	UC-02
Use Case Name	Perform Nmap Scan
Primary Actor	Penetration Tester
Preconditions	Nmap is installed and target machine is reachable.
Postconditions	Scan results are displayed and stored in logs.
Main Flow	1. User enters target IP. 2. User selects scan type and options. 3. User clicks "Run Scan". 4. Output is displayed and logged.
Alternate Flow	Invalid IP address triggers an error message.

Use Case UC-02 is the description of the way in which the system conducts an Nmap scan by using the graphical interface. In this use case, the penetration tester will start a network scan by launching it with an IP address of the target and selecting the type and options of the scan. The prerequisite is to have Nmap tool ready and available in the system. The inclusion of the main flow explains how the user initiates the scan using the GUI and how the system carries out the scan in the background, shows real time output and logs the results. The postcondition validates that scan results, as well as the calculated risk score are saved in the database after successful execution. The alternate flow also manages the invalid inputs or unreachable target by showing a suitable system error message, which makes the system robust.

Use Case: UC-03 – Run Metasploit Module

Table 4.3: Use Case UC-03 – Run Metasploit Module

Use Case: UC-03 – Run Metasploit Module	
Use Case ID	UC-03
Use Case Name	Search and execute a Metasploit exploit module.
Primary Actor	Penetration Tester
Preconditions	Metasploit Framework is installed and the target is reachable.
Postconditions	Exploit output is displayed and stored; sessions remain active.
Main Flow	1. User opens the Metasploit tab. 2. User enters a search term (e.g., "icecast"). 3. System returns matching modules. 4. User selects module and sets RHOSTS, RPORT, payload, etc. 5. User clicks "Run Module". 6. Output is displayed and saved.
Alternate Flow	1. Module not found → error message shown. 2. Exploit fails → output still logged but no session created.

UC-03 provides the exploitation procedure with using the Metasploit framework in the GUI. The main actor picks out a Metasploit module, sets up any necessary parameters including target host, port, payload settings as well as listener settings and launches the exploit. The precondition presupposes the installation of Metasploit and access to the target system. The primary flow describes the way the system runs Metasploit commands within the system, introduces the exploit, and controls background jobs/sessions. When successful, the postcondition will be used to check that an active Meterpreter session has been established and logged. The other flow is the alternate flow which deals with unsuccessful exploitation

caused by wrong configuration or target protection, where the exploitation is controlled and errors are handled.

Use Case: UC-04 – Capture Traffic

Table 4.4: Use Case UC-04 – Capture Network Traffic

Use Case: UC-04 – Capture Network Traffic	
Use Case ID	UC-04
Use Case Name	Capture packets using Tshark.
Primary Actor	Penetration Tester / Network Analyst
Preconditions	Tshark is installed; user has permission to capture.
Postconditions	Capture results are displayed and stored.
Main Flow	1. User opens Wireshark/Tshark tab. 2. User refreshes and selects an interface. 3. User clicks "Start Capture". 4. Live output is displayed. 5. User clicks "Stop Capture". 6. Log is saved.
Alternate Flow	Tshark not installed → error message displayed.

The Use Case UC-04 describes the packet capture feature offered by integrated Wireshark/Tshark. In this application, the penetration tester picks a network interface, turns on optional capture filters and commences packet capture with the GUI. That precondition is that Tshark is in place and there is enough permission. The overall flow of the application explains the process of initiating traffic capture, monitoring and depiction in the application interface. After the capture is terminated, it is postconditioned that captured traffic data is saved and logged. The other flow is used to manage the invalid interface selection or permission, by displaying to the user the suitable error messages.

Use Case: UC-05 – Run BloodHound Ingest (Optional)

Table 4.5: Use Case UC-05 – Run BloodHound Ingest

Use Case: UC-05 – Run BloodHound Ingest	
Use Case ID	UC-05
Use Case Name	Execute bloodhound-python to collect AD data.
Primary Actor	Penetration Tester / Red Team Analyst
Preconditions	bloodhound-python installed; valid domain credentials.
Postconditions	Output logged; collection files generated.
Main Flow	1. User opens BloodHound tab. 2. User enters domain, credentials, and DC information. 3. User clicks "Run bloodhound-python". 4. Ingest data is collected. 5. Output is displayed and logged.
Alternate Flow	Incorrect credentials or unreachable domain → failure message logged.

Use Case UC-05 describes Active Directory analysis with the help of BloodHound. Data collection involves the use of the GUI by providing domain credentials by the penetration tester. The precondition presupposes that BloodHound tools and valid domain access are installed properly. The principal flow describes the way the system executes BloodHound data collector, collects privilege and relationship data and makes it ready to be visualized. The postcondition will indicate that data have been collected successfully and can be further analyzed. The alternate flow is used to process incorrect credentials or problems in connectivity, and to make sure that it is done in a secure and controlled manner.

Table 4.6: Use Case UC-06 – View Logs and Export Report

Use Case: UC-06 – View Logs and Export Report	
Use Case ID	UC-06
Use Case Name	View logs and export HTML report.
Primary Actor	Penetration Tester / Manager
Preconditions	Logs exist in SQLite database.
Postconditions	Selected log displayed; report exported to HTML.
Main Flow	1. User opens Reports tab. 2. User clicks "Refresh Logs". 3. Logs appear in dropdown. 4. User selects a log. 5. User clicks "View Selected Log". 6. User clicks "Export HTML Report".
Alternate Flow	No logs available → export disabled.

The Use Case UC-06 covers the mechanism of the report generation in the system. In this application, the user picks one of the scan or activity logs that have been done on the system. The precondition is used to guarantee that the testing data is stored in the database. The primary flow describes the steps used to retrieve logs and the data and create structured HTML security report by the system. The confirmation of the postcondition is that the report is generated and stored on the local system successfully. The alternative stream addresses the situation of unavailable logs or export failure, which guarantees appropriate responses to the user.

4.7 Risk Scoring Design

The design of a simple heuristic risk scoring is based on:

- Number of open ports.
- Presence of high-risk ports (23, 445, 3389, 21, 110, 139).
- Occurrence of vulnerability indicators (e.g., CVE identifiers).

This score is put on a normal basis of 0-10 scale and stored against each Nmap log record.

4.8 Summary

The chapter constituted a detailed design of the GUI-Based Internal Network. Penetration Testing Tool based on existing research and best practices related to cybersecurity. tices. A well-planned data flow, intuitively designed GUI layout, and a modular architecture. out make the tool available, easy to scale, and efficient. The design facilitates easy. integration of open-source security tools that were generally trusted, forming an integrated and au- penetration testing environment based on the automated environment.

5. System Implementation

5.1 Introduction

The present chapter shows the implementation of the GUI-Based Analysis of Penetration on Internal Networksin detail. The implementation, based on the architectural and design of the system discussed in Chapter 4, integrates various open-source security tools into one graphical user interface created in Python and using the PyQt5 framework. The implementation is centered around the need to achieve good interoperability among tools, modularity, real-time command execution and structured logging and automated report generation. The user-friendliness and feedback mechanisms were also given a special concern to make sure that the users have access to clear and timely information when running the tools. The system greatly simplifies the complexity of the operation of all functions in one simplified GUI application, which also opens penetration testing of the internal network to users with different degrees of technical skills.

5.2 Development Environment

The system was implemented and tested on:

- Operating System: Kali Linux 2023.4
- Programming Language: Python 3
- Framework: PyQt5
- Database: SQLite3
- **External Tools:**

- Nmap
- Metasploit Framework
- Tshark (command-line engine for Wireshark)
- BloodHound + bloodhound-python

This selection aligns with recommendations from contemporary cybersecurity research advocating the use of widely adopted open-source security tools for internal penetration testing workflows

5.3 Module-Based Implementation

Its implementation is done in a modular shape to facilitate support and scaling. ability. The major functions are separated and placed in separate GUI tab:

- Nmap Scanning Module
- Wireshark/Tshark Capture Module
- Metasploit Automation Module
- BloodHound Enumeration Module
- Reporting and Log Management Module

PyQt5 QTabWidget is employed in the GUI in order to give features a clean separation.

5.4 Nmap Scanning Module Implementation

5.4.1 Overview

The Nmap module of the proposed system enables the fundamental functions of network scanning, which are discovery of the host, port scanning, version detection of services, and operating system fingerprints. The graphical interface also lets users specify the parameters of the scanning to suit their needs, including defining the port range, the type of scan to be undertaken, the option to detect operating system as

well as whether the scan should be carried out at elevated privileges using a sudo button. These parameters are dynamically loaded to the Nmap engine and give an opportunity to flexibly and controlled scans and provide real-time output to the GUI.

5.4.2 Workflow

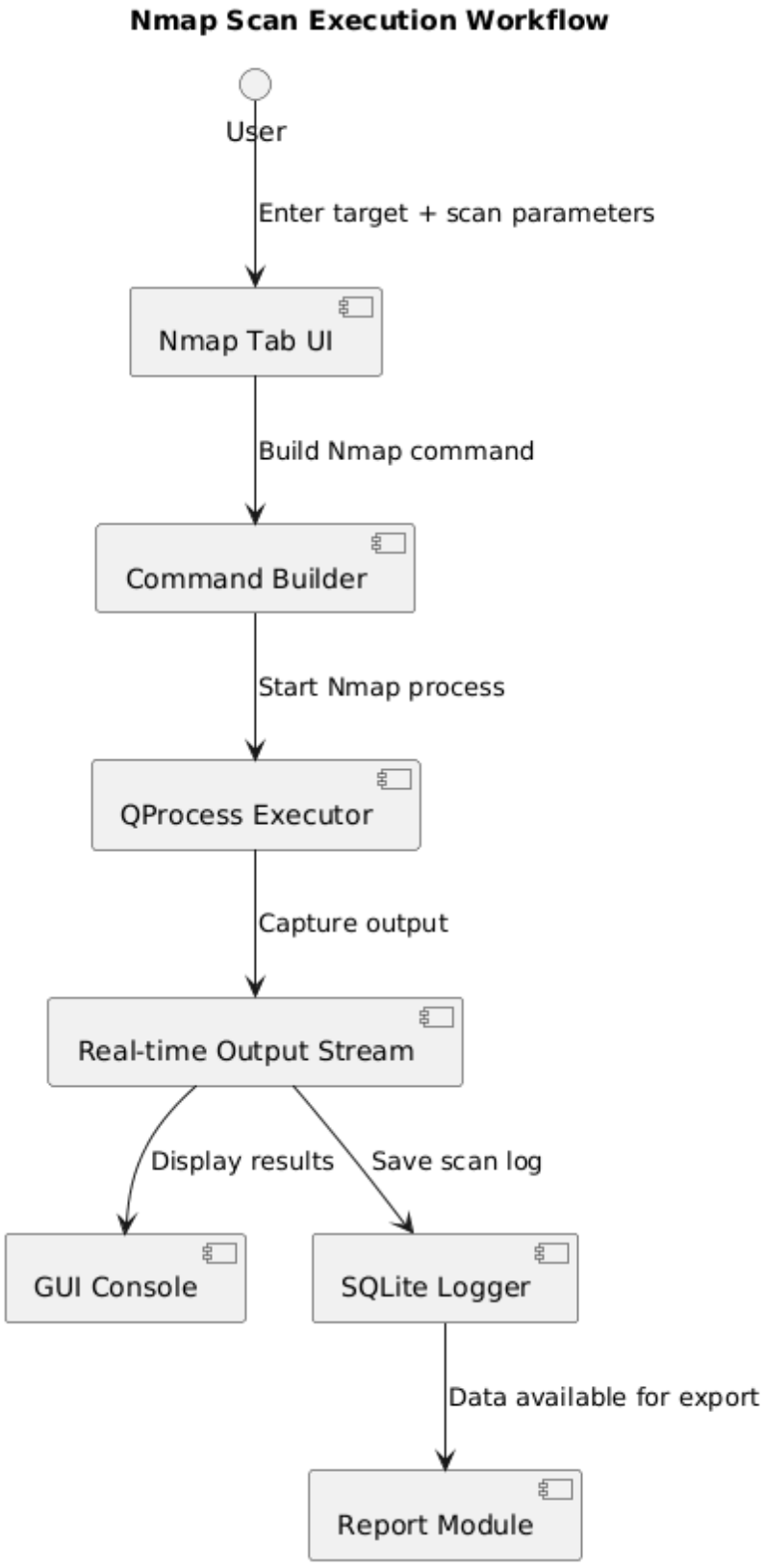

Figure 5.1: Nmap Scan Execution Workflow

There is a graph of the entire execution of the Nmap scanning module as shown in Figure 5.1 that is a part of the GUI-Based Internal Network Penetration Testing Tool. It starts with the user entering the target address and the scan parameters that he wants through the Nmap tab on the graphical interface. Such inputs are scan type, port range and optional flags as they enable the user to tailor the scanning behavior to suit the testing need.

After the input has been captured, the Command Builder component is dynamically assembled to build the relevant Nmap command in accordance with the options that have been chosen. This makes the scan to be flawlessly carried out without the user to have to concern with the command line. The resulting constructed command is then forwarded to the QProcess Executor which starts the Nmap process in the background without being connected with the GUI thread.

In the execution process, the system saves the scan output in real time by using an output stream handler. This live output is also shown in the GUI console where the user can follow scan progress and scan results in real time as they appear. Simultaneously, the scan data is recorded in a logging system based on SQLite, and the results are stored in an unlimited system to be referred to later.

Lastly, the information on the logged scans is made available to the module of generating the reports so that structured and exportable security reports can be generated. This workflow shows how the system consists of user interaction, automated command execution, real time feedback and structured data storage have been incorporated into a smooth and efficient scanning process.

5.4.3 Risk Scoring

A built-in heuristic assigns a risk score based on:

- Number of open ports
- Presence of high-risk ports (e.g., 445, 3389, 21)
- Detection of potential vulnerabilities in output

This aligns with CVSS-inspired scoring frameworks widely referenced in research.

5.5 Wireshark/Tshark Capture Module Implementation

5.5.1 Overview

Tshark is used for:

- Live packet capture
- Filtering based on BPF syntax
- Timed or continuous capture

5.5.2 Workflow

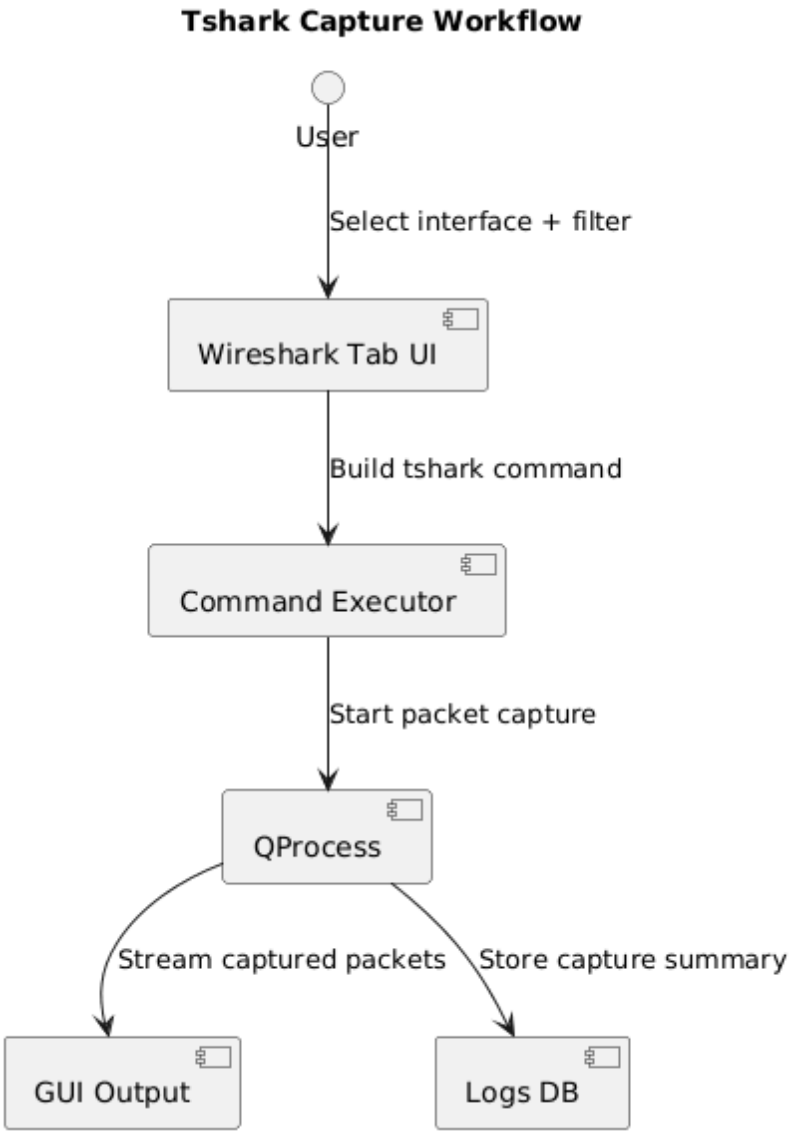

Figure 5.2: Wireshark/Tshark Workflow

The Fig. 5.2 illustrates the operations of the packet capture with Wireshark/Tshark in the system. This process starts once the user chooses the network interface as well as capture filters via the GUI. The system then constructs the relevant Tshark command and runs the same command with the help of QProcess. The packets captured are sent in real time through the GUI output and a summarized capture log is written into the database. This strategy will facilitate efficient monitoring of traffic, real-time analysis, and record keeping to be reviewed later.

5.6 Metasploit Automation Module Implementation

5.6.1 Overview

The Metasploit module automates:

1. Module search
2. Exploitation
3. Payload configuration
4. Remote command execution
5. Logging

This module uses Metasploit's msfconsole with the -x flag for automated command execution.

5.6.2 Module Search Implementation

The user enters a keyword (e.g., `icecast`, `ftp`, `ssh`).

A Metasploit search command is executed:

```
msfconsole -q -x "search <term>; exit"
```

The system parses module paths using regex patterns such as:

```
exploit/windows/http/icecast_header
auxiliary/scanner/ftp/ftp_version
```

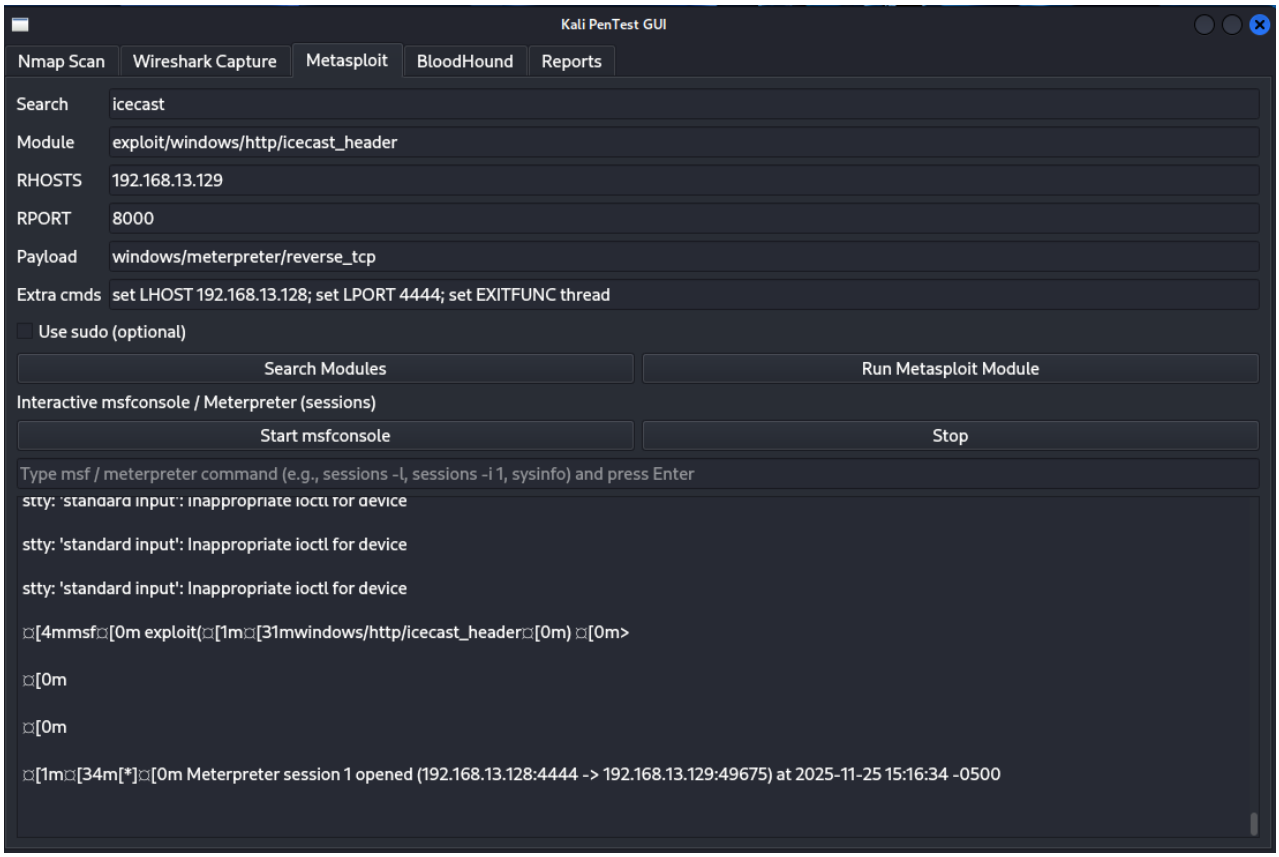

Figure 5.3: Exploitation

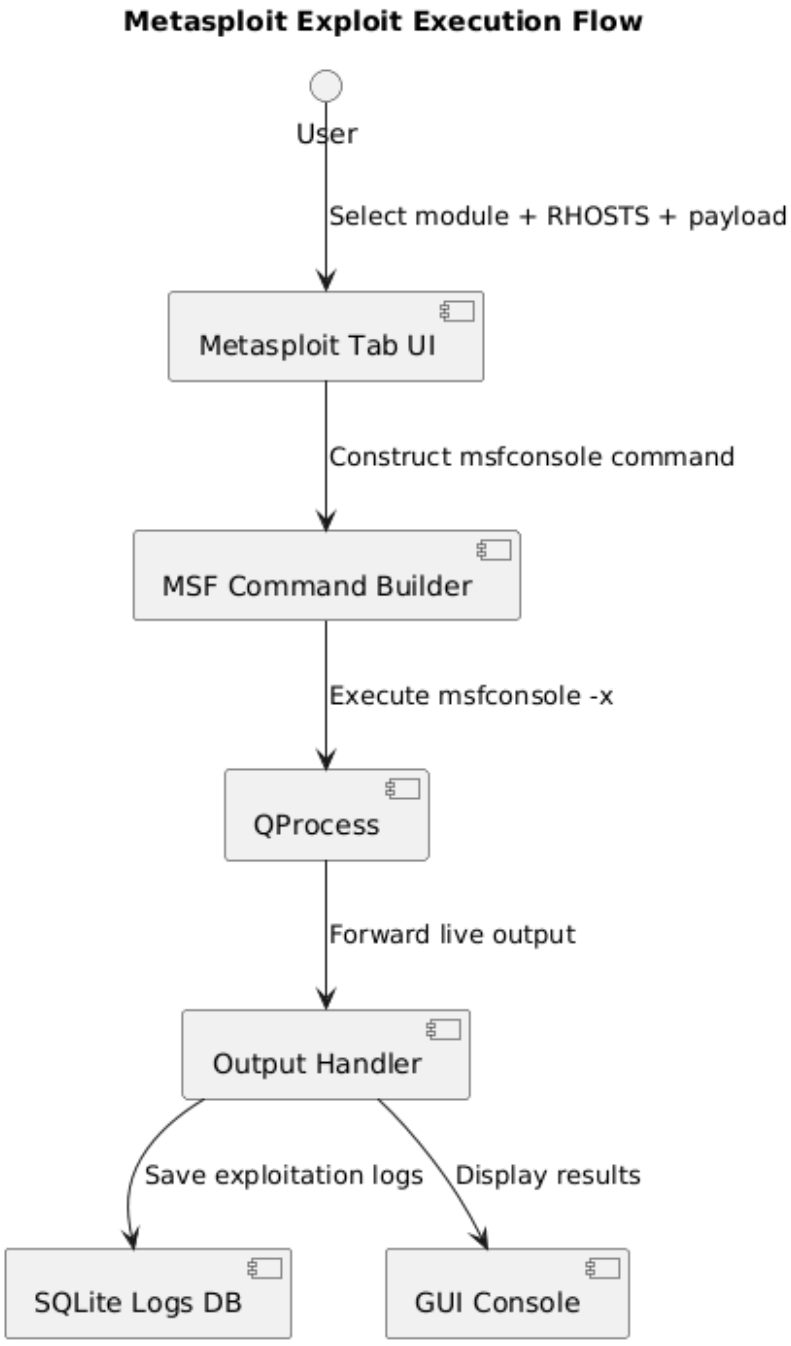

Figure 5.4: Exploit Execution Workflow

Figure 5.4 shows the manner in which the exploit execution is done using the Metasploit module. The GUI allows the user to select the exploit, target address (RHOSTS) and payload. The system builds up the msfconsole command needed and runs it by use of QProcess. Live exploit output is sent to the GUI console and logs on exploitation are stored in the SQLite logs database. The workflow enables monitored execution of exploits, real time feedback and suitable documentation of exploitation findings.

5.7 BloodHound Integration Module Implementation

5.7.1 Overview

BloodHound integration includes:

- Runs `tshark -D` to list interfaces.
- Allows the user to select an interface and optional filter.
- Captures packets using QProcess and appends output to the GUI.

5.7.2 Workflow

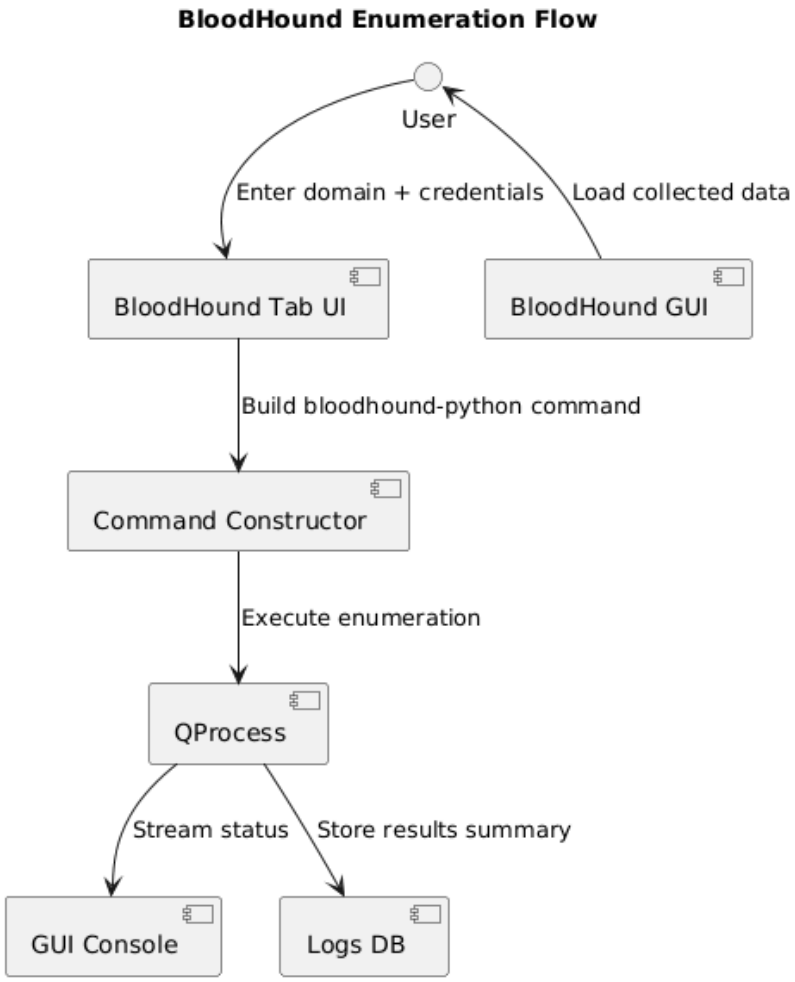

Figure 5.5: BloodHound Enumeration Flow

The BloodHound enumeration process is shown in figure 5.5 as being integrated into the GUI. The user enters domain credentials on the BloodHound tab and the system prepares and runs BloodHound-Python command on QProcess. Status of the enumeration is continuously updated to the GUI console and summarized enumeration results are logged to the database records. Data gathered can be imported into the BloodHound GUI and visual analysis of the Active Directory relationships is performed.

5.8 Logging and Report Generation

5.8.1 Logging Implementation

Every module writes to a centralized SQLite database:

- Tool name
- Target
- Command parameters
- Timestamp
- Risk score
- Raw output

5.8.2 HTML Report Generation

Reports include:

- Summary section
- Tool-specific findings
- Risk score
- Raw logs
- Timestamp

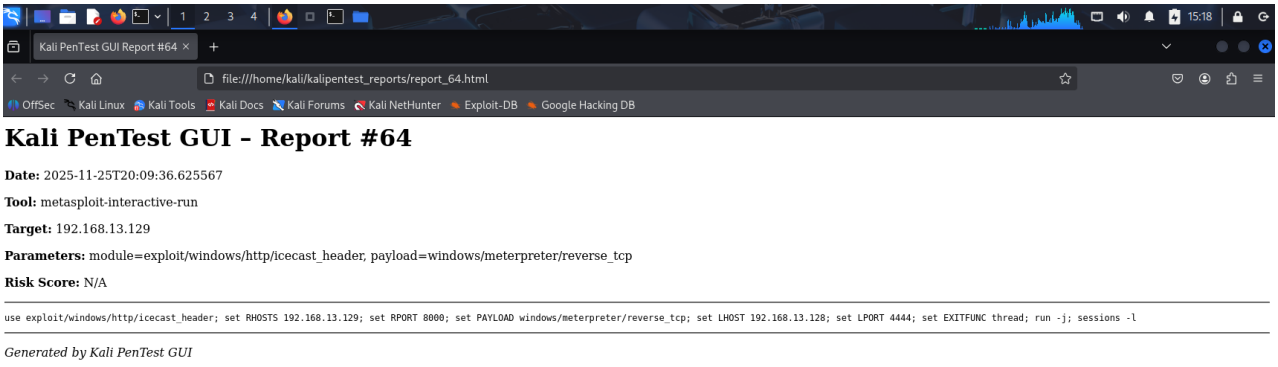

Figure 5.6: HTML Report

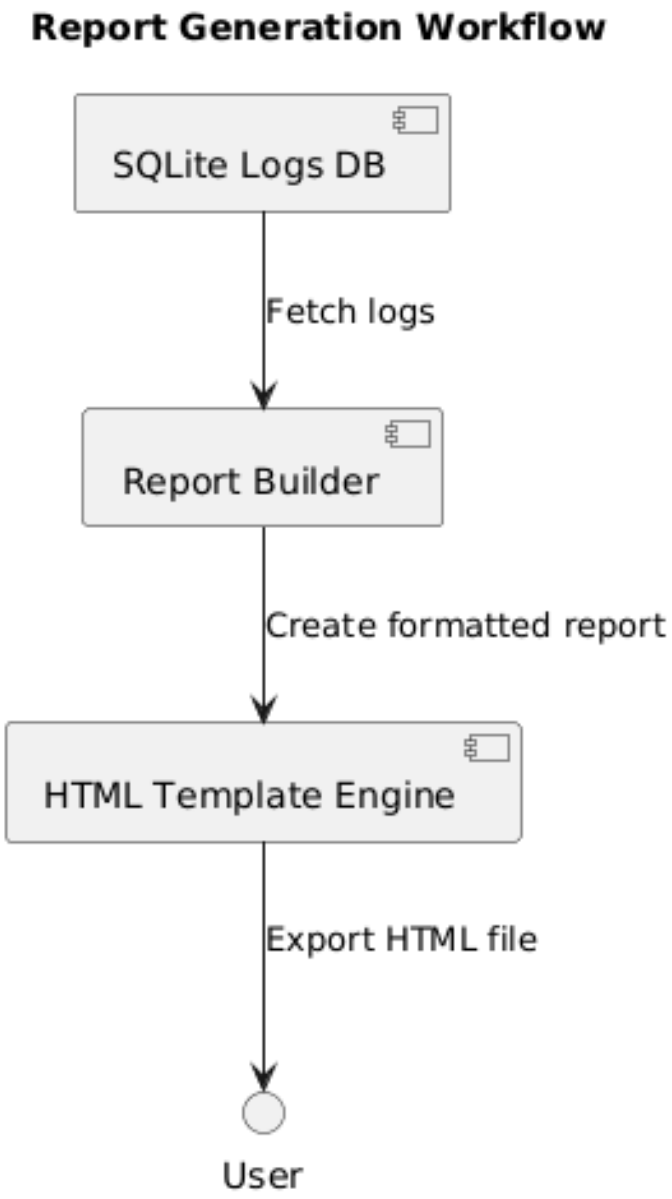

Figure 5.7: Report Generation Workflow

The Reports tab gives a user an option of a log and produces an export in the form of an HTML. Reports have an overview, configuration, risk score, and raw output in already structured blocks.

5.9 Logging and Reporting Implementation

Each executed command stores:

- Summary section
- Tool-specific findings
- Risk score
- Raw logs
- Timestamp

5.10 Error Handling and User Feedback

The GUI provides:

- Pop-up alerts for missing fields
- Warnings for missing tools
- Validation messages
- Real-time command output
- Process completion notifications

This improves usability and aligns with recommended GUI testing guidelines discussed in the literature.

5.11 Summary

The chapter reflected the penetration testing tool in the form of GUI implemented. emphasizing the use of the modular method of integrating industry-standard security tools. into a unified interface. Underuse of Python, PyQt5, QProcess, and SQLite will provide. portability, extended and consistency. The internal is illustrated through workflow diagrams. logic of every module giving a clear picture on the way the tools interact. data flows.

6. System Testing and Evaluation

6.1 Introduction

The chapter includes testing and evaluation of the implemented system including usability, integration, functional, and performance testing. A controlled virtual lab was to test the tool using realistic conditions of internal network.

6.2 Testing Strategy

Testing of the project entailed a variety of tests to make sure that the system was reliable and as envisaged. All tabs and functionalities were tested separately and their proper functioning was checked. Cross-testing also had to be done to ensure that integrated tools and modules would be used without conflict. The usability testing was conducted during the development of the interface to make sure that the interface was user-intuitive. Moreover, performance testing was performed to determine the efficacy and the responsiveness of scanning and exploitation operations.

6.3 Test Environment

As part of the experimental design of a controlled virtual environment, the internal network penetration testing was safely demonstrated. The attacker machine was a Kali Linux virtual machine that had the developed GUI based penetration testing tool. The system used to simulate a real world internal service weakness was a windows virtual machine with an exposed icecast server. The two machines were linked with

a host-only internal network, and they were not connected with the outside world networks, which made the machines isolated but in a real-life testing environment.

6.4 Functional Test Cases

Representative functional test cases are summarized below.

Test Case: TC-01 – Successful Login

Table 6.1: Test Case TC-01 – Successful Login

Test Case: TC-01 – Successful Login	
Test Case ID	TC-01
Related Use Case	UC-01 – Launch and Login
Description	Verify that a valid username and password allow the user to log in and open the main GUI.
Preconditions	Application is installed; default user (e.g., `admin/admin`) exists in SQLite database.
Test Steps	1. Launch the application. 2. Enter valid username and password. 3. Click the "Login" button.
Expected Result	Login dialog closes and the main tabbed interface is displayed without errors.
Actual Result	To be filled after execution.

Test Case: TC-02 – Login with Invalid Credentials

Table 6.2: Test Case TC-02 – Invalid Login

Test Case: TC-02 – Invalid Login	
Test Case ID	TC-02
Related Use Case	UC-01 – Launch and Login
Description	Verify that invalid credentials are rejected and user is not logged in.
Preconditions	Application is installed and login dialog is visible.
Test Steps	1. Run the application. 2. Enter `admin` as username. 3. Enter a wrong password (e.g., `1234`). 4. Click "Login".
Expected Result	A critical error message "Incorrect password." is shown and the login dialog remains open. Main GUI is not displayed.
Actual Result	To be filled after execution.

Table 6.3: Test Case TC-03 – Nmap Scan with Valid Target

Test Case: TC-03 – Nmap Scan with Valid Target	
Test Case ID	TC-03
Related Use Case	UC-02 – Perform Nmap Scan
Description	Verify that Nmap scan runs correctly when a valid target IP is provided.
Preconditions	• Nmap is installed and in PATH. • User is logged in. • Target host (e.g., vulnerable Windows VM) is powered on and reachable.
Test Steps	1. Open the "Nmap Scan" tab. 2. In "Target" field, enter a valid IP (e.g., `192.168.13.129`). 3. Leave "Ports" empty to use default. 4. Choose scan type "-sS (SYN)". 5. Keep "Use sudo" checked. 6. Click "Run Nmap Scan".
Expected Result	• Command line (starting with `$ sudo nmap ...`) appears in the output box. • After execution, open ports are listed. • A heuristic risk score line like "[+] Heuristic risk score: X/10.0 (saved to logs)" is appended. • A new entry is stored in the `logs` table with tool = "nmap" and the correct target IP.
Actual Result	To be filled after execution.

Table 6.4: Test Case TC-04 – Metasploit Search for Icecast Module

Test Case: TC-04 – Metasploit Search for Icecast Module	
Test Case ID	TC-04
Related Use Case	UC-03 – Run Metasploit Module
Description	Verify that searching for "icecast" suggests the correct module path and auto-fills the Module field.
Preconditions	• Metasploit is installed and `msfconsole` is accessible. • User is logged in and Metasploit tab is open.
Test Steps	1. In "Search" field, type `icecast`. 2. Click "Search Modules". 3. Wait for msfconsole search output to appear in the text area.
Expected Result	• Output shows a matching module row for Icecast. • At the end, a section "[+] Parsed module paths:" is printed. • One of the parsed entries is `exploit/windows/http/icecast_header`. • The "Module" textbox is automatically filled with `exploit/windows/http/icecast_header`.
Actual Result	To be filled after execution.

Table 6.5: Test Case TC-05 – Metasploit Icecast Exploit

Test Case: TC-05 – Metasploit Icecast Exploit	
Test Case ID	TC-05
Related Use Case	UC-03 – Run Metasploit Module
Description	Verify that running the Icecast exploit from the GUI can successfully open a Meterpreter session on the vulnerable Windows machine.
Preconditions	• Metasploit installed and working from terminal. • Vulnerable Windows 10 VM with Icecast service running on port 8000. • Correct internal IPs known for attacker (LHOST) and target (RHOSTS).
Test Steps	1. Use TC-05 to search for "icecast" and auto-fill the module. 2. Confirm "Module" = `exploit/windows/http/icecast_header`. 3. Enter target IP (e.g., `192.168.13.129`) in RHOSTS. 4. Enter `8000` in RPORT. 5. Enter attacker IP (e.g., `192.168.13.128`) in LHOST. 6. Enter `4444` as LPORT. 7. Set payload: `windows/meterpreter/reverse_tcp`. 8. Click "Run Metasploit Module".
Expected Result	• Metasploit output in GUI shows exploit steps. • A line indicates "Meterpreter session X opened" with correct IP/port tuple. • `sessions -l` output lists the active session. • A log entry is created in `logs` table with tool = "metasploit" and target IP.
Actual Result	To be filled after execution.

Table 6.6: Test Case TC-06 – Metasploit Icecast Exploit

Test Case: TC-06 – Metasploit Icecast Exploit	
Test Case ID	TC-06
Related Use Case	UC-03 – Run Metasploit Module
Description	Verify that running the Icecast exploit from the GUI can successfully open a Meterpreter session on the vulnerable Windows machine.
Preconditions	• Metasploit installed and working from terminal. • Vulnerable Windows 10 VM with Icecast service running on port 8000. • Correct internal IPs known for attacker (LHOST) and target (RHOSTS).
Test Steps	1. Use TC-05 to search for "icecast" and auto-fill the module. 2. Confirm "Module" = `exploit/windows/http/icecast_header`. 3. Enter target IP (e.g., `192.168.13.129`) in RHOSTS. 4. Enter `8000` in RPORT. 5. Enter attacker IP (e.g., `192.168.13.128`) in LHOST. 6. Enter `4444` as LPORT. 7. Set payload: `windows/meterpreter/reverse_tcp`. 8. Click "Run Metasploit Module".
Expected Result	• Metasploit output in GUI shows exploit steps. • A line indicates "Meterpreter session X opened" with correct IP/port tuple. • `sessions -l` output lists the active session. • A log entry is created in `logs` table with tool = "metasploit" and target IP.
Actual Result	To be filled after execution.

Table 6.7: Test Case TC-07 – Tshark Capture Start/Stop

Test Case: TC-07 – Tshark Capture Start/Stop	
Test Case ID	TC-07
Related Use Case	UC-04 – Capture Network Traffic
Description	Verify that the GUI can start and stop a Tshark capture and store the captured text in logs.
Preconditions	• Tshark is installed. • User is logged in. • Interface with some traffic is available (e.g., `eth0`).
Test Steps	1. Open "Wireshark Capture" tab. 2. Click "Refresh Interfaces" and choose a valid interface. 3. Leave filter empty and duration set to 0 (manual stop). 4. Click "Start Capture". 5. Generate some traffic (e.g., ping or web request). 6. After a few seconds, click "Stop".
Expected Result	• The command line appears (e.g., `$ sudo tshark -i 1 -l`). • Incoming packets are printed in the text area. • After "Stop", QProcess terminates and output stops. • A new log entry with tool = "tshark" is stored in `logs`.
Actual Result	To be filled after execution.

Table 6.8: Test Case TC-08 – View Log Detail

Test Case: TC-08 – View Log Detail	
Test Case ID	TC-08
Related Use Case	UC-06 – View Logs and Export Report
Description	Verify that a selected log entry displays full metadata and tool output.
Preconditions	At least one Nmap, Tshark, or Metasploit run has been executed and stored in `logs`.
Test Steps	1. Open the "Reports" tab. 2. Click "Refresh Logs". 3. Select any log from the combo box. 4. Click "View Selected Log".
Expected Result	The text area shows details: • Log ID, Tool, Target, Params • Created timestamp • Risk Score (if any) • Full saved output
Actual Result	To be filled after execution.

Table 6.9: Test Case TC-09 – Export HTML Report

Test Case: TC-09 – Export HTML Report	
Test Case ID	TC-09
Related Use Case	UC-06 – View Logs and Export Report
Description	Verify that an HTML report file is generated for a selected log.
Preconditions	• At least one log entry exists. • `REPORTS_DIR` folder is writable (e.g., `$HOME/kalipentest_reports`).
Test Steps	1. Open the "Reports" tab. 2. Click "Refresh Logs". 3. Select a log in the combo box. 4. Click "Export HTML Report". 5. Check the reports folder in the file system.
Expected Result	• A message box shows the saved path (e.g., `report_5.html`). • The generated HTML file contains date, tool, target, parameters, risk score (if available) and full output in `<pre>` format.
Actual Result	To be filled after execution.

Table 6.10: Test Case TC-10 – BloodHound Ingest Execution

Test Case: TC-10 – BloodHound Ingest Execution	
Test Case ID	TC-10
Related Use Case	UC-05 – Run BloodHound Ingest
Description	Verify that bloodhound-python runs with given AD credentials and logs output.
Preconditions	• `bloodhound-python` installed. • Test domain and DC reachable. • Valid domain username/password known.
Test Steps	1. Open "BloodHound" tab. 2. Fill "Domain", "Domain Controller", "Username", "Password". 3. Optionally set "Name Server". 4. Click "Run bloodhound-python (All)".
Expected Result	Console output of bloodhound-python is displayed and saved as a log entry with tool = "bloodhound". ZIP files are created for later import in BloodHound GUI.
Actual Result	To be filled after execution.

6.5 Integration Testing

Integration testing was done to ensure that all the modules were integrated to work as a single system. This testingregulated that Metasploit module choice was coordinated to the findings of Nmap scans so that exploitable choices could be made in an educated manner. It also verified that after successful scans Metasploit exploitation and correct session creation occurred. Also, Tshark was tested to capture network traffic when performing activities of exploitation. Lastly, the testing ensured that all the tools outputs were logged and stored appropriately and could be used to generate the consolidated reports.

6.6 Usability Evaluation

The usability feedback was gathered on the basis of peers that had prior familiarity with penetration testing. The feedback showed that the tab-based design was good to follow the workflow of penetration testing and simplified navigation. It was found that clear field labels and place hold text enhance ease of use and minimize errors in input. Also, the automatic storage of logs was welcomed, since there was no necessity to manually copy logs and it made the whole process more efficient.

6.7 Performance Evaluation

Qualitative performance analysis revealed that the system was very efficient with normal conditions of testing. Nmap scans finished in a reasonable time period when scanning a small internal network, which is an acceptable performance. The searches in metasploit modules, and execution of the exploits were reactive and had no noticeable delays. Also, command execution was done by using QProcess that made the graphical interface responsive and it was not frozen whenever the operation was long-running.

6.8 Evaluation Summary

In general, the system addressed its functionality purpose and worked well during the test environment. It was able to support the entire process of internal penetration testing between scanning and exploitation and collection of evidence.

7. Conclusion and Future Work

7.1 Conclusion

The project involved the design and development of GUI Based Internal Network Penetration Testing Tool that incorporated the use of Nmap, Metasploit, Tshark and optional BloodHound as a single desktop program. The main aim of the tool was to decrease the use of command-line interfaces and decrease the barriers of entry to conduct internal network penetration testing. The system was able to demonstrate the major penetration testing tasks, such as scanning the ports and services with Nmap, using the results of the scanning to inform the choice of exploits, and using the Metasploit to search and run modules, including the Icecast exploit. Traffic over the network which occurred in the course of exploitation was also captured on Tshark to facilitate further examination, whereas all the outputs were documented and collated into ordered HTML reports to facilitate documentation. The practical effectiveness and the real-life applicability of the proposed tool was evidenced by the successful exploitation of a vulnerable Icecast service and creation of a Meterpreter session in a controlled laboratory environment.

7.2 Limitations

Although this system is effective, there are some limitations associated with it. The tool is dependent on the proper installation and setup of third-party security tools like Nmap, Metasploit, and Tshark that can result in a functional tool in case of dependency missing or misuse. Moreover, the system also allows exploitation work-

flows, but advanced post-exploitation activity, e.g., full privilege escalation remains not automated but requires human input. In addition, it has optional and basic features (BloodHound analysis of AD) that offers limited analytics, but does not offer enterprise-level analysis of AD.

7.3 Future Work

It is possible to note several improvements that could be useful to increase the functionality and effectiveness of the proposed system. The next generation of the tool can also be equipped with automated checks of privilege escalation and long post exploitation checks to minimize manual intervention following successful exploitation. Machine learning methods integration might also be considered to forecast the exploitability and evaluate the risk levels, considering scan data. It can be improved with the ability to support PDF exports with graphical summaries and visual analytics. Also, real-time notifications and dashboard might be implemented to allow constant monitoring of the internal network. There should be enhanced and more integration with BloodHound and other Active Directory attack path analysis tools to enhance AD tests. Lastly, the tool can be supported with multi-user environment with centralized log management to make it more organizational and team-based.

7.4 Final Remarks

The project shows that even complex penetration-testing workflows can be neatly packaged into a simple, organized GUI application. It provides a practical and accessible platform for learning and performing internal network security assessments, especially in small-scale lab environments.

REFERENCES

[1] Verizon, "2023 Data Breach Investigations Report," Verizon Enterprise, 2023.

[2] IBM Security, "Cost of a Data Breach Report 2023," IBM, 2023.

[3] A. Shameli-Sendi, R. M. Mohammad, A. A. H. N. Zainal, and M. Dagenais, "Taxonomy of intrusion risk assessment and response system," *Computers & Security*, vol. 31, no. 4, pp. 494–508, 2012.

[4] J. Allen et al., "Improving the Security of Networked Systems," Carnegie Mellon University, 2020.

[5] S. K. Sharma, "Penetration testing framework for internal networks," *International Journal of Computer Applications*, vol. 182, no. 45, pp. 1–7, 2021.

[6] W. Stallings, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*. Pearson, 2015.

[7] Cisco, "LAN Design Fundamentals," Cisco Press, 2022.

[8] NIST, "Technical Guide to Information Security Testing and Assessment," NIST SP 800-115, 2021.

[9] S. Northcutt and J. Novak, *Network Intrusion Detection*. New Riders, 2002.

[10] A. B. Tiwari, "Analyzing internal LAN vulnerabilities using open-source scanners," *IEEE Access*, vol. 8, pp. 129331–129340, 2020.

[11] EC-Council, *CEH v12 Official Courseware*. EC-Council Press, 2023.

[12] R. McClure and A. Scambray, *Hacking Exposed*. McGraw-Hill, 2017.

[13] J. Andress, *The Basics of Information Security*. Syngress, 2019.

[14] OWASP, "Penetration Testing Guide," OWASP Foundation, 2022.

[15] Rapid7, "Metasploit Framework Documentation," Rapid7 LLC, 2023.

[16] H. D. Moore, "Metasploit: The penetration tester's platform," *SecurityFocus*, 2007.

[17] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide*. Insecure.Org, 2009.

[18] Nmap Project, "Nmap Reference Guide," 2023.

[19] Wireshark Foundation, "Wireshark User Guide," Version 4.2, 2023.

[20] R. Sanders, "Packet Analysis Using Wireshark," SANS Institute Reading Room, 2022.

[21] R. Wright and A. Spencer, "BloodHound: AD Attack Path Analysis," SpecterOps, 2022.

[22] SpecterOps, "BloodHound Documentation," 2023.

[23] I. Sommerville, *Software Engineering*, 10th ed. Pearson, 2020.

[24] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2015.

[25] M. Summerfield, *Rapid GUI Programming with Python and Qt*. Prentice Hall, 2008.

[26] Riverbank Computing, "PyQt5 Reference Guide," 2023.

[27] FIRST.org, "CVSS v3.1 Specification Document," FIRST, 2022.

[28] U. Parmar, "Automated CVSS-based vulnerability scoring in enterprise environments," *IEEE Trans. Inf. Forensics Security*, vol. 17, no. 3, 2022.

[29] US-CERT, "Vulnerability Note VU#431844 – Icecast header overwrite," 2004.

[30] Rapid7, "Icecast Remote Buffer Overflow (exploit/windows/http/icecast_header)," Metasploit Module Report, 2023.

[31] CVE-2004-1561, "Icecast 2.0.1 Header Processing Buffer Overflow," MITRE, 2023.

[32] I. Kotenko and A. Fedorchenko, "Performance evaluation of network security tools," *Journal of Information Security*, vol. 10, pp. 139–150, 2019.

[33] D. F. Moore, “Automated penetration testing research,” in *IEEE S&P Workshops*, 2020.

[34] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1993.

[35] A. Dix, J. Finlay, G. Abowd, and R. Beale, *Human-Computer Interaction*. Pearson, 2004.

[36] OWASP, “Network Testing Cheat Sheet,” OWASP Foundation, 2023.

[37] MITRE, “ATT&CK Enterprise Matrix,” MITRE Corporation, 2023.

[38] Palo Alto Networks, “Internal Lateral Movement Analysis,” Unit 42 Cyber Reports, 2022.

[39] SANS Institute, “Best Practices for Internal Network Security Audits,” SANS Reading Room, 2021.