Faculty of Computer Science and Information Technology

WEB APP BUG HUNTING ASSISTANT (WABUHA): A COMPREHENSIVE TOOLKIT FOR GUIDED RECONNAISSANCE AND EXPLOITATION FOR BEGINNER PENETRATION TESTERS

Lim Hong Yang

Bachelor in Computer Science (Network Computing) with Honours

2025

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE WEB APP BUG HUNTING ASSISTANT (WABUHA): A COMPREHENSIVE TOOLKIT FOR GUIDED RECONNAISSANCE AND EXPLOITATION FOR BEGINNER PENETRATION TESTERS

ACADEMIC SESSION: 2024/2025

LIM HONG YANG

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

☐	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
☐	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
☑	UNRESTRICTED	

林宏陽

(AUTHOR'S SIGNATURE)

Validated by

(SUPERVISOR'S SIGNATURE)

Permanent Address

NO 76 TAMAN HUI SING
LORONG 2/2 JALAN STAMPIN
93350, KUCHING, SARAWAK

Date: 22 JUNE 2025

Date: 21 JULY 2025

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

WEB APP BUG HUNTING ASSISTANT (WABUHA): A COMPREHENSIVE TOOLKIT FOR GUIDED RECONNAISSANCE AND EXPLOITATION FOR BEGINNER PENETRATION TESTERS

LIM HONG YANG

This project is submitted in partial fulfilment of

the requirements for the degree of Bachelor of Computer Science with Honours

(Network Computing)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2025

WEB APP BUG HUNTING ASSISTANT (WABUHA): A COMPREHENSIVE TOOLKIT FOR GUIDED RECONNAISSANCE AND EXPLOITATION FOR BEGINNER PENETRATION TESTERS

LIM HONG YANG

Projek ini merupakan salah satu keperluan untuk

Ijazah Sarjana Muda Sains Komputer dengan Kepujian

(Pengkomputeran Rangkaian)

Fakulti Sains Komputer dan Teknologi Maklumat

UNIVERSITI MALAYSIA SARAWAK

2025

DECLARATION

I hereby declare that this project is entirely my own work and has been completed independently, except for portions clearly cited or referenced in the text. I have not copied any part of it from another student, nor has any section been written on my behalf by someone else.

林宏陽

Date: 22 June 2025

Lim HongYang (79880)
Network Computing
Faculty of Science Computer and Information Technology
Universiti Malaysia Sarawak

ACKNOWLEDGEMENT

To begin with, I want to thank everyone who has supported me, directly or indirectly, in bringing my Final Year Project to its successful end. My heartfelt appreciation goes to my supervisor, Dr Norfadzlan bin Yusup, for their invaluable guidance, support, and constructive feedback throughout the development of my project, Web App Bug Hunting Assistant (WABUHA). It has been their knowledge and advice that has been key to me being able to finish this project in time.

Then I express my deepest gratitude to Universiti Malaysia Sarawak (UNIMAS) for its conducive learning facilities which enables me to further explore and enhance my skills. I would like to thank the Faculty Computer Science Information Technology (FCSIT) for giving me the chance to work on this challenging project, which in addition to enriching my background, trained me to face future challenges in cybersecurity.

Finally, I am sincerely grateful to my parents and siblings for their continued presence, encouragement, and tolerance through this process. Their faith in me has always been a driving force. I appreciate for their cooperation, brainstorming and moral support to my project. To all those who have helped achieve the goal of this thesis, thank you for your care and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BWAPP	Buggy Web Application
CFAA	Computer Fraud and Abuse Act
CIA	Confidentiality, Integrity, and Availability
CLI	Command-line Interface
CSV	Comma Separator Value
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DNS	Domain Name System
DVWA	Damn Vulnerable Web Application
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
OWASP	Open Worldwide Application Security Project
PDF	Portable Document Format
PDPA	Personal Data Protection Act
PTES	Penetration Testing Execution Standard
RECON	Reconnaissance
SQLi	Structured Query Language Injection
UML	Unified Modelling Language
WABUHA	Web App Bug Hunting Assistant
XSS	Cross Site Scripting

ABSTRACT

Cybersecurity threats to web applications are constantly evolving, requiring robust penetration testing to identify and remediate vulnerabilities. Traditional penetration testing tools often provide a steeper learning curve for beginner penetration testers, causing them difficulty in performing thorough assessments. This project addresses this challenge by developing the Web App Bug Hunting Assistant (WABUHA). It is a comprehensive toolkit designed to provide guided reconnaissance and exploitation specifically for beginner penetration testers. WABUHA integrates streamlined processes for scope management, information gathering, vulnerability scanning, exploitation, and reporting. Key features include risk categorisation, Common Weakness Enumeration (CWE) mapping for better understanding, and explanations of detected vulnerabilities are powered by the Gemini 2.0 Flash model. The developed tool simplifies complex penetration testing workflows, making it more convenient and in progressive manner. The automated identification and successful exploitation of common web vulnerabilities by WABUHA demonstrated its effectiveness in guiding beginners through the reconnaissance and exploitation phases. This enhances their learning experience and improves the quality of web application security assessments. This tool significantly contributes to guiding new talent in the cybersecurity field by lowering the entry barrier to practical penetration testing.

Keywords: Web Application Security, Penetration Testing, Vulnerability Assessment, Cybersecurity, Artificial Intelligence, WABUHA

ABSTRAK

Ancaman keselamatan siber terhadap aplikasi web sentiasa berkembang, justeru memerlukan ujian pencerobohan yang mantap bagi mengenal pasti dan membaiki kelemahan sistem. Namun begitu, alat ujian pencerobohan tradisional selalunya sukar digunakan oleh penggiat keselamatan siber yang masih baharu, menyukarkan penilaian yang menyeluruh. Projek ini membangunkan Web App Bug Hunting Assistant (WABUHA), iaitu satu set alat menyeluruh yang direka khas untuk memberi panduan dalam proses peninjauan dan eksploitasi bagi penguji pencerobohan peringkat permulaan. WABUHA menggabungkan pengurusan skop, pengumpulan maklumat, pengimbasan kelemahan, eksploitasi, dan pelaporan dalam satu aliran kerja yang dipermudahkan. Ciri utama termasuk pengelasan risiko yang intuitif, pemetaan kepada Common Weakness Enumeration (CWE) untuk kefahaman yang lebih baik, serta penjelasan kelemahan berasaskan Gemini 2.0 Flash Model. Alat ini memudahkan proses ujian pencerobohan yang kompleks agar lebih mudah diakses secara progresif. Pengenalpastian automatik dan eksploitasi berjaya terhadap kelemahan lazim dalam aplikasi web oleh WABUHA membuktikan efektif dalam membimbing pengguna baharu melalui fasa peninjauan dan eksploitasi. Pendekatan ini bukan sahaja meningkatkan pengalaman pembelajaran mereka, malah turut mempertingkat kualiti penilaian keselamatan aplikasi web. Alat ini memberikan sumbangan yang signifikan dalam membimbing bakat baharu dalam bidang keselamatan siber dengan merendahkan halangan kemasukan kepada ujian pencerobohan secara praktikal.

Chapter 1: Introduction

1.1 Background

As our dependence on web applications grows, so does the need for strong cybersecurity. It's crucial to identify and fix vulnerabilities before they can be exploited by malicious actors. For aspiring web application penetration testers, the number of available tools can be overwhelming, especially when the tools are mostly console-based and require significant technical expertise and command-line proficiency.

The Web Application Bug Hunting Assistant (WABUHA) addresses these challenges through an integrated toolkit that simplifies the penetration testing process while enhancing the users understanding and practical skills. This report outlines the development of WABUHA, a tool that combines popular reconnaissance and exploitation tools—for system discovery and vulnerability assessment respectively—into a user-friendly graphical user interface (GUI).

Key aspects examined in this report include the need for a structured approach to web application penetration testing, the importance of ethical guidelines, and the integration of a Common Weakness Enumeration (CWE) to map relevant vulnerabilities. The report will also cover the phases of the toolkit's development, emphasising the Agile methodology to ensure continuous improvement based on the results of vulnerability accuracy testing.

However, it is important to note that WABUHA is designed primarily for educational purposes and may not encompass every tool or technique used in professional penetration testing. The focus is on creating a simplified, beginner-friendly environment, which means advanced features and tools may be excluded. This limitation aims to ensure that users can build foundational skills without feeling overwhelmed by complexity.

1.2 Problem Statements

The wide range of web application reconnaissance and exploitation tools available can be overwhelming for beginners. Therefore, selecting the most appropriate tools can be difficult for beginners (Hoffman, 2024). Furthermore, the general availability of console-based tools introduces additional complexity, making it more difficult for novices to use these tools efficiently in web application penetration testing.

There is a significant gap in educational resources that provide comprehensive, hands-on guidance for using these tools (Oluwatosin Islamiyat Yusuf, 2024). Beginners often lack access to detailed explanations and contextual information about the commands and options required to maximise the potential of the tools, hindering their ability to develop essential practical skills.

The disjointed nature of existing web application security tools, which frequently operate in isolation across separate platforms (Hoffman, 2024), complicates the learning process for beginners. This fragmentation prevents them from consolidating findings, tracking progress, and understanding the broader security context, leading to incomplete assessments and limiting their ability to communicate vulnerabilities effectively.

1.3 Objectives

1. To enhance the bug-hunting process for beginners by integrating popular reconnaissance and exploitation tools into an accessible graphical user interface (GUI).
2. To develop a structured and intuitive workflow for web application penetration testing, guiding beginners through each testing phase.
3. To integrate findings and track progress systematically across multiple stages of web application penetration testing, enhancing organisation and efficiency.
4. To enhance vulnerability comprehension through AI-powered explanations generated by the Gemini 2.0 Flash model, supporting beginner understanding and remediation efforts.

1.4 Brief Methodology

The Agile methodology used in the development of WABUHA allows for iterative changes through ongoing testing and feedback (Agile Alliance, 2001). Planning, Design, Development, Testing, and Launching are the five main stages of the methodology, which incorporates a simplified version of the Open Worldwide Application Security Project (OWASP) Penetration Testing Execution Standard (PTES) (OWASP, 2023). The goals of each step are to maintain a user-friendly workflow, ensure usability, and improve the functionality of WABUHA. For novice web application penetration testers, this procedure encourages progressive improvement, flexibility, and alignment with learning objectives.

1.5 Project Scope

The WABUHA is designed for aspiring penetration testers, students, and individuals interested in developing their skills in web application penetration testing. Recognising the challenges faced by novices, WABUHA is tailored for those who are unfamiliar with the complexities of penetration testing tools and methodologies. This toolkit aims to provide an accessible entry point into the world of ethical hacking. By offering a user-friendly interface and structured guidance, WABUHA empowers users to engage in the bug-hunting process effectively, bridging the gap between theoretical knowledge and practical application.

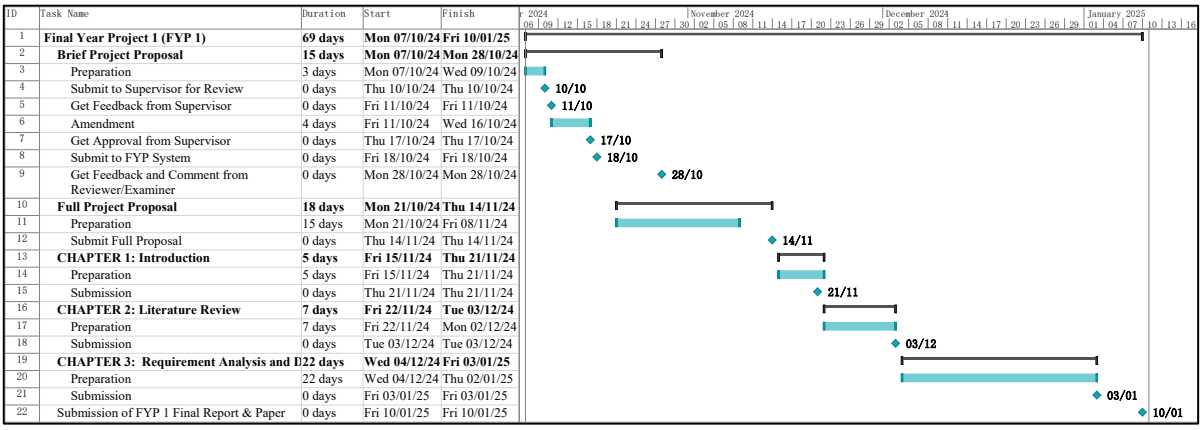
The WABUHA project encompasses the development of a comprehensive toolkit that integrates popular reconnaissance and exploitation tools into a cohesive, graphical user interface (GUI). This toolkit will streamline the penetration testing process by providing users with a guided workflow that covers essential stages of web application penetration testing. The project aims to centralise various functionalities, allowing users to easily navigate through the reconnaissance, exploitation, and analysis phases. By consolidating findings and enabling progress tracking. WABUHA will map to the latest Common Weakness Enumeration (CWE) database to identify any relevant weaknesses found during the exploitation stage. It enhances the educational experience and supports users in developing their skills in a systematic manner. Ultimately, the toolkit serves as a vital resource for beginners, equipping them with the knowledge and practical experience necessary to conduct thorough web application assessments. Furthermore, WABUHA empowers users to define their target scope, including domain, subdomain, and HTTP user-agent settings, ensuring that ethical boundaries are respected throughout the testing process. In the testing phase, Docker environments will be set up to simulate attacks on vulnerable web applications, ensuring the features function effectively before release.

1.6 Significance of Project

The WABUHA project is important for cybersecurity education, particularly for novice penetration testers. The user-friendly GUI and guided workflows make it easier for learners to understand the principles of web application penetration testing. WABUHA assists users in developing practical skills by providing an integrated toolkit that streamlines the reconnaissance and exploitation procedures in a straightforward and organised manner. It also allows users to define testing scopes and maintain ethical boundaries encouraging responsible practices among users. Overall, this project aims to help build a new generation of ethical hackers, improving the security landscape and contributing to the protection of web applications against emerging threats.

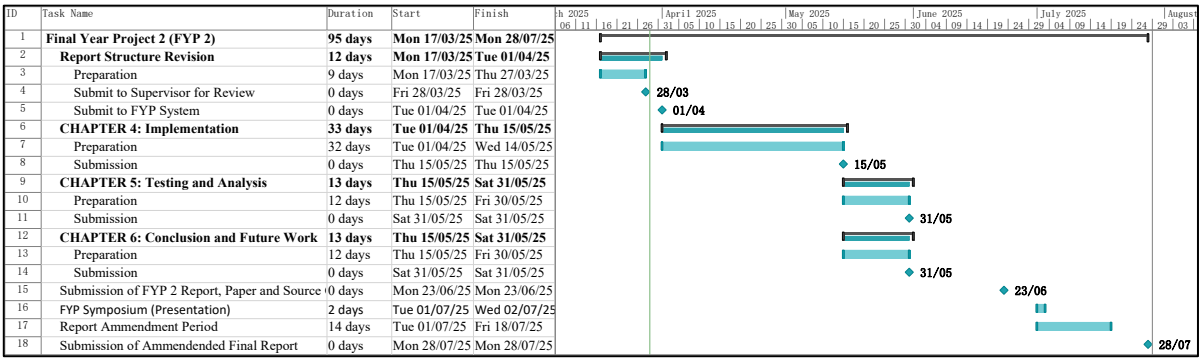
1.7 Project Schedule

Figures 1.1 and 1.2 present the Gantt charts for the WABUHA project, outlining the planned timeline and key milestones. The project spanned from 7th October 2024 to 28th July 2025. The chart illustrates the dependencies between tasks, for example, the testing phase could not begin until the implementation phase was finalised. While some minor adjustments were made to the timeline due to unforeseen challenges, the project was completed within the allocated timeframe. The Gantt charts serve as visual representation of the project's progress and the effective project management strategies employed.



ID	Task Name	Duration	Start	Finish
1	**Final Year Project 1 (FYP 1)**	**69 days**	**Mon 07/10/24**	**Fri 10/01/25**
2	**Brief Project Proposal**	**15 days**	**Mon 07/10/24**	**Mon 28/10/24**
3	Preparation	3 days	Mon 07/10/24	Wed 09/10/24
4	Submit to Supervisor for Review	0 days	Thu 10/10/24	Thu 10/10/24
5	Get Feedback from Supervisor	0 days	Fri 11/10/24	Fri 11/10/24
6	Amendment	4 days	Fri 11/10/24	Wed 16/10/24
7	Get Approval from Supervisor	0 days	Thu 17/10/24	Thu 17/10/24
8	Submit to FYP System	0 days	Fri 18/10/24	Fri 18/10/24
9	Get Feedback and Comment from Reviewer/Examiner	0 days	Mon 28/10/24	Mon 28/10/24
10	**Full Project Proposal**	**18 days**	**Mon 21/10/24**	**Thu 14/11/24**
11	Preparation	15 days	Mon 21/10/24	Fri 08/11/24
12	Submit Full Proposal	0 days	Thu 14/11/24	Thu 14/11/24
13	**CHAPTER 1: Introduction**	**5 days**	**Fri 15/11/24**	**Thu 21/11/24**
14	Preparation	5 days	Fri 15/11/24	Thu 21/11/24
15	Submission	0 days	Thu 21/11/24	Thu 21/11/24
16	**CHAPTER 2: Literature Review**	**7 days**	**Fri 22/11/24**	**Tue 03/12/24**
17	Preparation	7 days	Fri 22/11/24	Mon 02/12/24
18	Submission	0 days	Tue 03/12/24	Tue 03/12/24
19	**CHAPTER 3: Requirement Analysis and D**	**22 days**	**Wed 04/12/24**	**Fri 03/01/25**
20	Preparation	22 days	Wed 04/12/24	Thu 02/01/25
21	Submission	0 days	Fri 03/01/25	Fri 03/01/25
22	Submission of FYP 1 Final Report & Paper	0 days	Fri 10/01/25	Fri 10/01/25

Figure 1.1: Final Year Project 1 (FYP 1) Gantt Chart.



ID	Task Name	Duration	Start	Finish
1	**Final Year Project 2 (FYP 2)**	**95 days**	**Mon 17/03/25**	**Mon 28/07/25**
2	**Report Structure Revision**	**12 days**	**Mon 17/03/25**	**Tue 01/04/25**
3	Preparation	9 days	Mon 17/03/25	Thu 27/03/25
4	Submit to Supervisor for Review	0 days	Fri 28/03/25	Fri 28/03/25
5	Submit to FYP System	0 days	Tue 01/04/25	Tue 01/04/25
6	**CHAPTER 4: Implementation**	**33 days**	**Tue 01/04/25**	**Thu 15/05/25**
7	Preparation	32 days	Tue 01/04/25	Wed 14/05/25
8	Submission	0 days	Thu 15/05/25	Thu 15/05/25
9	**CHAPTER 5: Testing and Analysis**	**13 days**	**Thu 15/05/25**	**Sat 31/05/25**
10	Preparation	12 days	Thu 15/05/25	Fri 30/05/25
11	Submission	0 days	Sat 31/05/25	Sat 31/05/25
12	**CHAPTER 6: Conclusion and Future Work**	**13 days**	**Thu 15/05/25**	**Sat 31/05/25**
13	Preparation	12 days	Thu 15/05/25	Fri 30/05/25
14	Submission	0 days	Sat 31/05/25	Sat 31/05/25
15	Submission of FYP 2 Report, Paper and Source	0 days	Mon 23/06/25	Mon 23/06/25
16	FYP Symposium (Presentation)	2 days	Tue 01/07/25	Wed 02/07/25
17	Report Ammendment Period	14 days	Tue 01/07/25	Fri 18/07/25
18	Submission of Ammendended Final Report	0 days	Mon 28/07/25	Mon 28/07/25

Figure 1.2: Final Year Project 2 (FYP 2) Gantt Chart.

1.8 Expected Outcome

i. Promote responsible penetration testing by educating users on defining testing scopes and respecting ethical boundaries.

ii. Users will acquire practical skills and a basic understanding of web application penetration testing methodologies.

iii. The user-friendly GUI and guided workflows will simplify the bug-hunting process, helping novices navigate key stages and progress tracking.

1.9 Summary

Chapter One introduces the Web Application Bug Hunting Assistant (WABUHA), a beginner-friendly toolkit designed to simplify web application penetration testing through an integrated graphical interface. It addresses challenges faced by novices, such as tool complexity and fragmented resources, by providing guided workflows, practical learning, and systematic progress tracking. The project leverages the Agile methodology and aligns with a simplified OWASP standard to ensure iterative improvements, focusing on educational value and ethical testing practices. By consolidating popular tools and map to the CWE database, WABUHA aims to make bug hunting accessible while equipping users with essential skills and knowledge.

1.10 Project Report Outline

Chapter One: Introduction

This chapter provides an overview of the project, including its title, background, problem statements, objectives, methodology, scope, significance, and expected outcomes. It discusses the reason the Web Application Bug Hunting Assistant (WABUHA) was developed and what its goal is as a friendly toolkit for web application penetration testing.

Chapter Two: Literature Review

The features, benefits, and drawbacks of three similar tools to WABUHA are examined in this chapter. A comparison demonstrates how WABUHA bridges the gaps in these tools and illustrates its distinct benefits to penetration testing for novices.

Chapter Three: Requirement Analysis and Design

The phases of the methodology, system and user requirements, and functional and non-functional requirements are covered in detail in this chapter. It describes the systematic approach used to create the toolkit, with a focus on the requirements of inexperienced users.

Chapter Four: Implementation

The system architecture, environment configuration, and implementation specifics of WABUHA are covered in this chapter. It demonstrates a seamless and straightforward web application penetration testing experience.

Chapter Five: Testing and Analysis

The testing and analysis process for WABUHA is covered in this chapter, along with specific test cases, their outcomes and user acceptance testing methodology. It describes how the tool was assessed to ensure accuracy, reliability, and beginner-friendliness.

Chapter Six: Conclusion and Future Work

This chapter summarises the project's accomplishments and contributions, looks at its limitations, and considers possible directions for future development and growth to increase the capabilities of WABUHA.

Chapter 2: Literature Review

2.1 Introduction

This chapter aims to look into fundamental ideas, techniques, and resources that support the Web Application Bug Hunting Assistant (WABUHA). This chapter identifies the gaps and difficulties that WABUHA seeks to solve through reviewing existing literature, tools, and techniques. It links theoretical knowledge with real-world applications and gives beginner web penetration testers a thorough understanding of web penetration testing (bug hunting). The review focuses on three primary themes, the theoretical background, existing key penetration testing techniques, and tools and their limitations that later define the architecture of WABUHA.

2.2 Theoretical Background

2.2.1 Penetration Testing Methodologies

Penetration testing methodologies such as the OWASP Penetration Testing Execution Standard (PTES) provide a structured framework for identifying and exploiting vulnerabilities in web applications (OWASP, 2023). As illustrated in Figure 2.1, pre-engagement interactions, intelligence collection, threat modelling, vulnerability analysis, exploitation, post-exploitation, and reporting are the seven stages of the technique (OWASP, 2023). These phases provide an organised approach, which WABUHA will later simplify to a much more beginner-friendly approach.

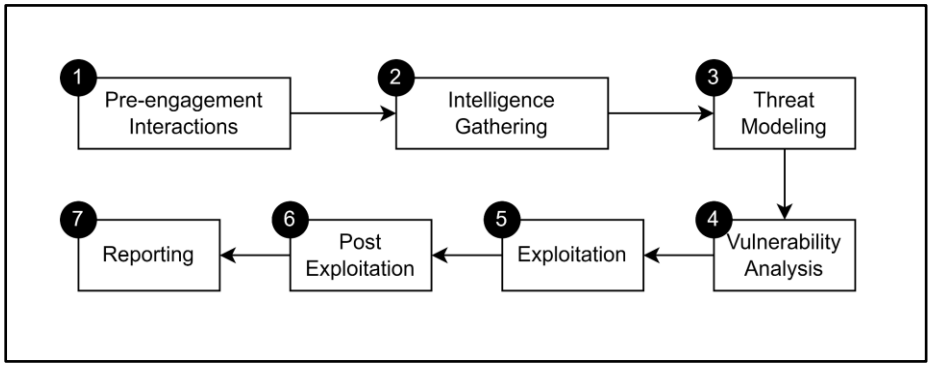


Figure 2.1: Seven Phases of OWASP PTES.

Pre-engagement interactions. The foundation for the entire penetration testing process is laid during the pre-engagement phase. The scope, goals, and limitations of the engagement are discussed and decided upon by the penetration tester and the client at this phase. This involves understanding the timeline, identifying networks, systems, or applications that are within or outside of scope, and clarifying ethical and legal issues. By ensuring everyone is on the same page, effective pre-engagement lowers the possibility of miscommunications (Astrida et al., 2022). For WABUHA, this stage can be made easier for novices by providing guided steps and instructions to set up the parameters for the engagement.

Intelligence collection. The second stage of the penetration testing lifecycle is gathering information about the target, often known as reconnaissance or **recon** for short (Astrida et al., 2022). Without making direct contact with the target system or application, it means gathering as much relevant data as possible about the target (also known as passive recon). The objective is to map out the structure of the targeted website, weaknesses, and potential vulnerabilities before making an active effort to exploit them. Usually, information is obtained from publicly accessible sources like search engines using a technique called Google Dorking (Jeawan Naayagar Kanakasabai et al., 2023), social networking sites, and domain registration information. Network reconnaissance methods like Domain Name System (DNS) searches and WHOIS lookups are also used. This stage is crucial for comprehending the attack

surface and creating a successful attack plan later. For WABUHA, simplifying the recon might involve streamlining the use of common tools and offering an easy single-click feature for beginners on how to perform passive reconnaissance but in a structural form of presentation.

Threat modelling. The process of identifying, understanding, and evaluating potential threats and attack routes to the target system is known as threat modelling (Cisco, 2024). Pentesters assess the target system at this step to guess how threat actors will possibly take advantage of the web security flaws and the potential impacts on the organisation. This phase also defines the assets to be protected, taking into account the skills and motives of possible threat actors, and defining the most likely attack scenarios. According to NIST (2022), pentesters may rate the vulnerability severity (not a risk measurement) according to the possible harm they could create in this step through the Common Vulnerability Scoring System (CVSS). By offering templates that assist pentesters in identifying assets and outlining potential threat scenarios in an organised way, WABUHA can make this process more straightforward for novices.

Vulnerability analysis. Finding vulnerabilities that could be used in a potential attack is the primary objective of this stage. To find vulnerabilities in networks, system setups, or applications, involves actively scanning and analysing target systems. Vulnerabilities like outdated software, improper security configuration, or insecure coding techniques are found using both automated tools and manual testing.

Exploitation. According to Lachkov et al (2022), during this stage, the penetration tester tries to compromise the target system by attacking the vulnerabilities that have been found. Verifying if these flaws may be used to escalate privileges, carry out malicious operations, or obtain unauthorised access. Depending on the kind of vulnerability different techniques are used such as buffer overflow, Cross-site Scripting (XSS) or SQL injection

(SQLi). Successful exploitation demonstrates that the vulnerability can be used in real-world scenarios as well as theoretical ones. Execution must be done carefully to prevent system damage (Lachkov et al., 2022). WABUHA makes this process easier by combining common tools and helpful instructions for regulated and safe exploitation.

Post-exploitation. The actions taken after a vulnerability has been successfully exploited are referred to as post-exploitation (Lachkov et al., 2022). The objectives of this phase are to maintain access to the target machine, obtain additional information, and determine the scope of the compromise (Lachkov et al., 2022). In addition to mapping out network settings and determining how extensively they can penetrate the system, pentesters also try to elevate privileges or switch to other systems within the network. The primary objective is to mimic the actions of an attacker once they have access, including the assets they would target and potential methods of evasion. By providing instructions on how to carry out post-exploitation activities safely and ethically, with a focus on discovering the impact of the compromise.

Reporting. By recording and communicating the results to the client, the reporting phase brings the penetration testing process to a close. An overview of the test is included in the report, along with details about the vulnerabilities discovered exploitation procedures, impacts, and suggested mitigations or solutions (Lachkov et al., 2022). Its goal is to assist the client in identifying and fixing their security flaws. Effective risk management requires reports that are concise, easily understood, and actionable.

2.2.2 Vulnerability, Threat and Risk

The terms vulnerability, risk, and threat are fundamental concepts in understanding the nature of security weaknesses. According to OWASP (n.d.), vulnerabilities are weaknesses or faults in a system that could be used by threat actors to obtain unauthorised access, cause harm, or carry out unexpected actions. These flaws often occur due to either outdated components, incorrect setups, or incorrect development techniques. However, a vulnerability itself is not harmful until it is exploited by a threat actor.

Threats, on the other hand, are the actors or events that exploit vulnerabilities to cause harm (Nash, 2023). These could be malicious hackers or even natural disasters. It might then lead to data breaches or system failures. Threats can be internal or external depending on the type of attacker. They can range from stealing sensitive data to disrupting services (Nash, 2023). To identify and analyse vulnerabilities, pentesters often use tools like the OWASP Top 10, a collection of the most critical web application security risks (OWASP, 2024).

Furthermore, determining the possibility and possible consequences of a vulnerability being exploited involves the use of the concept of risk. According to Kidd (2022), risk is a combination of the probability that a vulnerability will be exploited with the potential damage an exploit may cause. Organisations can prioritise vulnerabilities and focus on those that pose the biggest risk of harm due to this relationship.

2.2.3 OWASP Top Ten

The OWASP Top 10 is a widely acknowledged, popularised standard for describing the most critical security vulnerabilities of web applications. It is a fundamental tool for identification, prioritisation, and resolution of threats, providing developers and penetration testers with guidance. OWASP Top 10 security vulnerabilities in 2021 identified injection flaws, broken authentication and sensitive data exposure as some of the most prevalent and damaging vulnerabilities (OWASP, 2024).

Injection attacks, particularly SQL injection (SQLi), have long been a serious threat to web application security and are today among the most common and destructive weaknesses. Broken authentication is another one of the top problems when a web application cannot correctly authenticate users or securely handle user passwords.

When using the OWASP Top 10 penetration testers will be able to guarantee that they are addressing the most critical attack risks during security tests. For WABUHA, this framework can provide a guided approach for novices to identify and exploit these common vulnerabilities safely.

2.2.4 CVE vs CWE

For monitoring and remediating vulnerabilities, the Common Vulnerabilities and Exposures (CVE) library and the Common Weakness Enumeration (CWE) database are widely adopted frameworks. The CVE system offers a separate unique identifier of publicly disclosed vulnerabilities to exchange specified vulnerabilities among different platforms (MITRE, 2019a). Each CVE entry usually contains data for the software, version, and the flaws through the description of the vulnerability itself.

On the other hand, CWE emphasises on the root vulnerabilities of software which are prone to exploitation. Although CVE identifies specific vulnerabilities, CWE covers common weakness categories that may potentially give rise to security vulnerabilities (MITRE, 2019b). For example, an entry for a CVE might specify a vulnerability caused by insufficient input validation in a web server, whereas the corresponding entry for the CWE might describe this as a "Improper Input Validation" weakness.

In order to evaluate the security of a system, pentesters typically apply both CVE and CWE in combination. A CVE lookup can be used to identify if a particular software version is vulnerable, whereas a CWE lookup can be used to relate the flaw to a more general class of weaknesses that may impact other systems or applications.

2.2.5 Risk Rating Method (traditional vs CVSS)

Risk assessment is a core component of penetration testing, as it enables pentesters to prioritise vulnerabilities in order of the likely severity and probability of exploitation. According to Talabis & Martin (2013), Traditional risk assessment methods often involve a subjective analysis of the severity of a vulnerability, considering factors such as the potential harm, the ease of exploitation, and the business context. However, a more standardised method, such as the Common Vulnerability Scoring System (CVSS), offers a quantitative way to assess the severity of vulnerabilities (NIST, 2022).

The CVSS assigns a score to vulnerabilities on the basis of a number of metrics, namely, how easily the weakness can be exploited, the damage caused by a successful attack, and how difficult it is to exploit the weakness. The CVSS score is from 0 to 10, where more significant vulnerabilities will have higher scores. This scoring scheme allows organisations to rank their

remediation activities and to begin with highest and critical vulnerabilities which would pose the greatest risk to their systems (NIST, 2022).

WABUHA additionally includes an aspect of the CIA triad when assessing the severity of the vulnerability. This ensures that evaluators will also think about the impact not only of how technical a weakness is, but also whether the weakness will compromise the organisation's fundamental values of security. Regardless of the Weighted CVSS or conventional risk assessment approach, WABUHA is designed to help the novices to understand the value of each vulnerability and what it could potentially achieve.

2.2.6 Ethical Considerations in Penetration Testing

Conducting penetration testing with the consent of the organisation or person who owns the system or application under test is referred to as ethical hacking (OWASP, 2018). The main objective of ethical hacking is to discover and patch vulnerabilities in security before threat actors take advantage of them. Getting consent from the owners before performing any testing and ensuring that any vulnerabilities found are appropriately reported are two fundamentals of ethical hacking. Maintaining trust between penetration testers and system owners is necessary, as it avoids the legal or reputational risks that come with unauthorised access.

Pentesters must always be mindful of the CIA triad when conducting their work. CIA is referred to as Confidentiality (C), Integrity (I), and Availability (A). Confidentiality ensures that sensitive data is protected from unauthorised access, while Integrity guarantees that data is not tampered with or altered maliciously. Availability ensures that systems and data remain accessible and functional when needed. These three principles form the foundation of cybersecurity and must guide all ethical hacking activities. Penetration testers should conduct their tests in ways that do not compromise these principles. For instance, ethical hackers should

avoid causing disruptions to services or inadvertently exposing sensitive information during testing.

The responsible disclosure of vulnerabilities is one of the most important ethical factors in penetration testing. The tester has a responsibility to quickly inform the vendor or system owner of any vulnerabilities they find. Disclosure should take place before the vulnerability is made public, allowing the owner or vendor time to patch the security flaws. This ensures that the responsible parties can resolve the issue before it causes harm and helps stop threat actors from taking advantage of the vulnerability. Poor vulnerability disclosure can have significant consequences such as possible data breaches, financial losses, and harm to system users.

Penetration testers have to manage the legal frameworks that limit their line of work along with ethical principles (Dani, 2024). Laws like the General Data Protection Regulation (GDPR) in the European Union, the Computer Fraud and Abuse Act (CFAA) in the United States, and Malaysia's Personal Data Protection Act (PDPA) 2010 clearly define what behaviours are permitted and prohibited in penetration testing (European Union, 2016; Pejabat Pesuruhjaya Perlindungan Data Peribadi Malaysia, 2024; U.S. Department of Justice, 2015). For example, the PDPA protects the privacy of data subjects by regulating the gathering, use, and distribution of personal data. Unauthorised access to computer systems is illegal under the CFAA, and the GDPR places strict limits on how personal data is processed and stored, particularly for organisations based in the European Union (European Union, 2016; Pejabat Pesuruhjaya Perlindungan Data Peribadi Malaysia, 2024; U.S. Department of Justice, 2015).

Terms of service and contracts are also important that pentesters must adhere to when performing tests (Teichmann & Boticiu, 2023). These agreements outline the scope of the testing, specify which systems or applications are subject to testing, and establish the permissions granted by the system owner. Failure to follow these contracts can lead to legal

actions, financial penalties, or loss of professional credibility. Furthermore, adherence to frameworks like the OWASP Top 10 can guide testers in identifying the most critical security vulnerabilities in web applications (OWASP, 2024). This framework provides a list of the top risks and common vulnerabilities that should be prioritised during penetration testing, ensuring that the process aligns with industry standards and ethical guidelines.

For beginners, understanding and following ethical standards is a fundamental part of becoming proficient in penetration testing (Singh, 2023). The purpose of WABUHA is to assist novices in navigating these morally challenging situations with guided reconnaissance and exploitation. For instance, users can configure the target domain, subdomains and ports to ensure the test is within the scope defined by the website owners. WABUHA ensures that novices learn technical skills and ethical concepts that should guide them by integrating elements that walk users through the appropriate and legal web application penetration testing processes. WABUHA can assist novices in making ethical decisions during pentesting by providing them with clear prompts and instructional resources highlighting the significance of responsible vulnerability disclosure and legal compliance.

2.2.7 Impact of LLMs on Cybersecurity Education

Large Language Models as in the case of Gemini 2.0 Flash are to transform cybersecurity education by filling the skills gap and improving what we have in terms of training methods (Tuan, 2025). These models can create realistic and varied attack scenarios and offer personalised learning experiences, which are crucial for developing practical skills. LLMs also make it easier to develop resources, allowing educational institutions to quickly adapt to new threats (Alnajim et al., 2023).

Additionally, integrating LLMs presents challenges such as ensuring accuracy, preventing algorithmic bias, and managing hallucinations in educational content (Sirangi, 2025). This requires teaching students the AI literacy so they can evaluate the information generated by the AI effectively. Furthermore, LLMs are vulnerable to various attacks like prompt injection and data poisoning, and they raise concerns about intellectual property and data privacy (Sirangi, 2025). Therefore, a balanced approach is essential. Combining the benefits of LLMs with strong security measures, continuous evaluation, and a commitment to ethical AI development to foster a skilled and adaptable cybersecurity workforce.

2.3 Key Concepts and Techniques

This section discusses the fundamental concepts and techniques that underpin penetration testing. The fundamental procedures include scope definition, reconnaissance, vulnerability exploitation, and assessments and creating a controlled environment for educational purposes when finding vulnerabilities in either live or local web applications.

2.3.1 Scope Definition

A crucial first step in penetration testing is scope definition, where goals and limitations are set. During this phase, systems, networks, or applications within scope are identified, while prohibited areas are excluded. An appropriate scope definition ensures effective resource allocation during testing, reduces ethical and legal risks, and matches testing objectives with client expectations (Seidl & Chapple, 2022, pp. 31–57).

2.3.2 Reconnaissance (Recon)

Reconnaissance is aimed at information gathering about a target system to discover system vulnerabilities. It is classifiable both as Active and Passive recon, both using different ways and degrees of interaction with the target.

2.3.2.1 Active Recon

Active recon obtains comprehensive information, including open ports, services, and configurations, by engaging with target systems directly. These interactions are made easier by tools like Nmap and SQLmap, which scan networks and databases to detect vulnerabilities (Shah et al., 2019). Active reconnaissance is very instructive depending on the scanning aggressiveness, but it also has the potential of being discovered by defensive systems, therefore it must be done with permission and care.

2.3.2.2 Passive Recon

Passive recon gathers information without directly engaging the target. Techniques involve using programs like Maltego, Recon-ng, WHOIS, Shodan, or Sublist3r to analyse records, domain registrations, and publicly accessible data. This approach reduces the detection risks and may provide the target attack surface overview structure (Mossé Cyber Security Institute (MCSI), 2022).

2.3.3 Exploitation (Attack)

Exploitation attempts to obtain unauthorised access or carry out malicious actions to determine vulnerabilities. Buffer overflow attacks, SQLi, and XSS are common methods. This phase demonstrates the practical impact of theoretical vulnerabilities, transforming potential risks into actual threats (Nancy Bou Ghannam et al., 2022).

2.3.4 Vulnerability Identification

The purpose of vulnerability identification is to find weaknesses in security and use Common Vulnerabilities and Exposures (CVE) lookups and evaluations to match them to known vulnerabilities (MITRE Corporation, 2023). This process is streamlined by tools such as OWASP ZAP, Nessus, and Metasploit, which compare vulnerability databases with issues discovered. Risks are ranked according to their potential impact through assessments, which direct actions for remediation.

The CVE system provides unique identifiers for certain security vulnerabilities and is crucial in vulnerability assessment by encouraging effective tracking, communication, and remediation. The CVE system ensures that users may efficiently prioritise and handle risks by offering a centralised repository (MITRE Corporation, 2023).

2.3.5 Testing Environments with Docker

Docker is an essential technology in pentesting particularly for creating secure, controlled environments in which security testing may be carried out. Pentesters may simulate different operating systems and application environments using Docker containers instead of requiring resource-intensive tools like virtual machines. This lightweight solution ensures that

tests are isolated and performed without risking damage to production systems or data (Chandel, 2019; Yin et al., 2019).

Docker is the best option for novices and for the WABUHA testing phase since it makes the process of configuring testing environments easier. According to Yin et al. (2019), no matter what hardware is being used, users may conduct tests and simulate real-world scenarios with consistent results by using prebuilt container images, such as Damn Vulnerable Web Application (DVWA) and Buggy Web Application (BWAPP). It is also easy to test multiple applications or vulnerabilities at the same time because Docker containers may be quickly spun up or destroyed.

Docker is essential to WABUHA because it offers a consistent and secure environment for users to practice penetration testing. It ensures the testing environment remains isolated from the rest of the system and allows users to mimic attacks on web apps without risking the live website and data.

2.4 Usability Challenges in Cybersecurity Tools: CLI vs GUI

Command-line interfaces (CLIs) and graphical user interfaces (GUIs) are widely used in cybersecurity tools, although they offer varying levels of usability for novice users. Research has demonstrated that although CLIs are both versatile and powerful, CLIs can be a substantial bottleneck to beginners in cybersecurity or computing (Andrews & Straub, 2021; Svabensky et al., 2021). However, GUIs are typically easier to use and to access, providing a more straightforward entry point for people who lack technical background.

Research highlights that GUIs are more intuitive and simpler to use, especially for people who have no experience of working with CLIs. GUIs help users navigate tasks without requiring extensive prior knowledge. On the other hand, CLIs are highly specific to the known commands and their syntax, thus they are difficult for novices. For example, a research by Andrews & Straub (2021) shows that the participants exhibited a high false negative rate when using a CLI because of incorrect typed commands or because they did not understand the correct sequence of steps to take. However, in the same study no such problems were reported when participants used GUI-based tools and it reduced the number of errors.

The common challenge associated with CLIs is the overwhelming complexity they present to new users (Andrews & Straub, 2021). Novice users are sometimes required to memorise a collection of commands and to become familiar with the hierarchical nature of tools. This complexity can discourage learners from engaging with cybersecurity tools, especially those who rate themselves as having lower computer proficiency. In contrast, GUIs provide visual aids and a structured interface, which reduces the need for memorisation to allow users to focus on the core tasks. For example, study participants reported that they could complete tasks more quickly with a GUI, while CLI tasks often required additional guidance

and time to execute properly (Andrews & Straub, 2021). This demonstrates the clear advantage of GUIs in simplifying processes for inexperienced users.

Another major problem is that the learning curve for CLIs is steeper. According to the research by Andrews & Straub (2021), CLI can demand more technical expertise and preparation compared with GUIs in many cases. Beginners must invest substantial time and effort into learning specific commands and their applications. Participants in usability studies reported the CLI to be confusing and hard to use, whereas the GUI was reported to be more familiar and easily navigated. However, most of the participants reported that they could repeat the GUI-based tasks by themselves, but not after initial exposure CLI-based tasks still needed additional instructions.

In a nutshell, while CLI-based access is the norm for most professional penetration testing tools like ASM, jok3r, and Darkarmy, this approach presents usability barriers for beginners. WABUHA adopts a less common but increasingly recommended method. This design choice aligns with modern cybersecurity education trends that emphasize accessibility and learning outcomes. Although not yet the industry standard, this method is gaining recognition for its ability to reduce the steep learning curve and encourage ethical, responsible engagement in cybersecurity practices.

2.5 Review of Existing Similar Work or Tools

2.5.1 Attack Surface Mapper (ASM)

Attack Surface Mapper (ASM) is developed by *superhedgy* and its community on GitHub (superhedgy, 2019). It is an open-source reconnaissance Command Line Interface (CLI) tool designed to help penetration testers find and understand a target online application's attack surface or weak points to simplify (superhedgy, 2019), refer to Figure 2.2 for the demo screenshot. Also, ASM focuses on automating passive reconnaissance tasks, to speed up and improve the efficiency of information gathering (0x1, 2020).

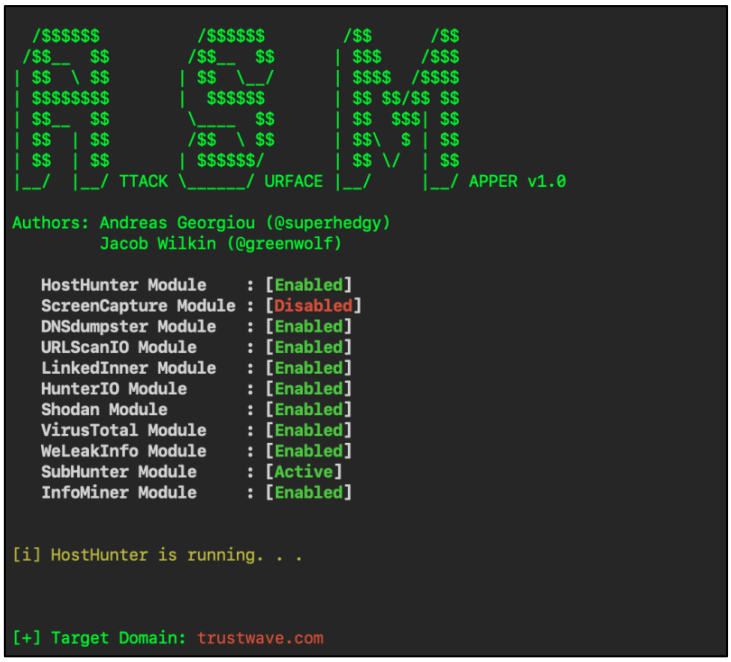


Figure 2.2: Demo Screenshot for ASM.

ASM has been designed to help web security researchers and penetration testers obtain vital information about web applications or targets (superhedgy, 2019). It offers an approachable method of automating reconnaissance tasks without requiring a high level of technical competence, making it suitable for both novices and experts. ASM automation ensures consistency, which is frequently challenging to get in manual procedures.

ASM combines several modules that work together to extract different kinds of information from a target application. In addition to basic analysis of possible security misconfigurations, it offers DNS enumeration, subdomain discovery, and open port identification. The findings will then shown in a clear and organised manner that makes automated exploitation or additional manual analysis easier (superhedgy, 2019).

ASM reduces the possibility of overlooking important information that could be used against organisations in later penetration testing phases by simplifying reconnaissance operations. It is helpful during the reconnaissance stage when learning as much as possible about the target.

In short, the time and effort needed during the reconnaissance phase can be greatly reduced by using ASM to automate tedious procedures. However, ASM does not include any exploitation tools but concentrates on reconnaissance. Furthermore, because it is open-source, community contributions are encouraged, which promotes ongoing development and threat adaptability.

2.5.2 jok3r Framework

The jok3r framework is an open-source beta web application penetration testing toolkit developed by *koutto* available on GitHub (koutto, 2023). This framework automates the detection and exploitation of web application vulnerabilities. It is well known for being modular, enabling users to apply a single tool to do a variety of tests (koutto, 2018b).

The goal of the Jok3r framework is to make web application testing easier and more efficient by integrating different tools and methods into a single framework. This reduces the burden of alternating between tools, which is a frequent problem for penetration testers. But as

it only supports CLI, users must have at least an intermediate familiarity with shell scripting, refer to Figure 2.3 for the demo screenshot (koutto, 2018a).

Figure 2.3: Demo Screenshot for Jok3r Framework.

Jok3r has modules for vulnerability detection, exploitation, and reconnaissance. It can automate password strength tests, directory brute-forcing, SQLi and XSS, for instance (koutto, 2023). It is a flexible option for testers seeking to find a variety of vulnerabilities because of its ability to combine several tools, including Shodan, Nmap, Nikto, and many more (koutto, 2023).

Due to the framework modularity, users may customise their testing strategy to meet particular project needs (koutto, 2018b). It may be used for both targeted vulnerability testing and generic assessments because of its adaptability. Additionally, Jok3r offers thorough reports that assist testers in ranking vulnerabilities according to their impact and seriousness (koutto, 2023).

Jok3r greatly improves the effectiveness of web application testing by automating certain testing lifecycle steps. It is a vital tool for testers because it integrates popular tools, which ensures thorough coverage of potential vulnerabilities.

2.5.3 Darkarmy

Darkarmy is a robust and comprehensive penetration testing toolkit that integrates a wide range of tools into a single framework, it is developed by *D4RK-4RMY* on GitHub (D4RK-4RMY, 2023). It supports every phase of penetration testing, from reconnaissance to post-exploitation, and is intended for ethical hackers and security researchers (D4RK-4RMY, 2023).

The toolkit aims to provide testers with a centralised solution that eliminates the need to switch between multiple tools. This integration makes handling several testing environments easier and ensures a smooth process. Modules for reconnaissance, vulnerability assessment, exploitation, and reporting features are all included in Darkarmy, a total of 11 different categories (GeeksforGeeks, 2023). It has tools for assessing the web application vulnerabilities including SQLi and XSS, as well as subdomain enumeration and network scanning (D4RK-4RMY, 2023). Its reporting module arranges findings to make communication with stakeholders and decision-making easier. However, users can only interact with the tool through CLI, refer to Figure 2.4 for the demo screenshot (D4RK-4RMY, 2023).

Figure 2.4: Demo Screenshot for Darkarmy

Darkarmy is a flexible tool for penetration testers because of its broad feature set and automation capabilities. It assures that users can complete evaluations on a single platform by supporting multiple necessary testing phases (D4RK-4RMY, 2023).

The all-in-one design of Darkarmy simplifies testing, preserving accuracy while reducing time and labour. It also promotes creativity and personalisation, which ensures that the tool will continue to be useful in the rapidly evolving field of cybersecurity.

2.6 Comparative Tools Analysis: Strengths and Weaknesses

Attack Surface Mapper, jok3r Framework and Darkarmyare the three tools being compared where each has unique benefits and drawbacks for web application penetration testing tasks.

Attack Surface Mapper (ASM) is a lightweight and effective reconnaissance tool. Although it is quite good at passive recon scans, using it for thorough penetration testing is limited by its lack of advanced features and exploitation support. For instance, it does not provide CVE or CWE references for vulnerabilities which users need to determine themselves. Additionally, novices may find it difficult to use as they can only interact with the tool through CLI. ASM is lightweight since it only performs basic operations with simple functions, using fewer resources than more complicated frameworks. It only requires minimal setup and uses passive scanning techniques, which often use less computational resources (Raghu, 2023).

The jok3r Framework offers a more extensive, modular toolkit that covers various penetration testing phases. Its adaptability and active community assure ongoing support. However, novices may find its complex nature, resource requirements, and CLI interface overwhelming (koutto, 2018b).

Darkarmy is a robust all-in-one framework for reconnaissance and exploitation, with many customisation capabilities for experienced web application penetration testers. Although it has many features, its high learning curve and resource requirements can make it challenging for novices and less appropriate for quick assessments (D4RK-4RMY, 2023).

In short, ASM, Jok3r, and Darkarmy each address distinct penetration testing requirements. ASM lacks advanced features but is lightweight and excellent for reconnaissance. Jok3r is extensive and modular, yet it requires a lot of resources and is complex. Although Darkarmy has many features, it requires significant computational power and a steep learning

curve. Every tool is appropriate for particular testing circumstances and skill levels. Table 2.1 summarises the strengths and weaknesses of Attack Surface Mapper (ASM), jok3r framework, and Darkarmy:

Table 2.1: Strengths and Weaknesses of Similar Existing Tools.

Tools	**Strength**	**Weakness**
ASM	Efficient, lightweight, focused reconnaissance	Limited scope, lacks exploitation, CLI-based, no CVE or detailed reports
jok3r	Comprehensive, modular, supports multiple phases	Requires prior knowledge, CLI-based, resource-intensive
Darkarmy	All-in-one, customisable, broad functionalities	Steep learning curve, CLI, resource-heavy, complex

2.7 Comparison with Proposed Tool

Although WABUHA takes inspiration from the tools under study, it also has a number of special features designed to appeal to novice web application penetration testers. Table 2.2 shows the comparison of similar existing tools with the proposed tool.

Table 2.2: Comparison of WABUHA with Similar Existing Tools.

Parameter/Tool	**Attack Surface Mapper (ASM)**	**jok3r Framework**	**Darkarmy**	**Proposed Tool (WABUHA)**
User Interface	CLI	CLI	CLI	**GUI**
Guided Workflow	No	No	No	Yes
Recon Tools	Yes	Yes	Yes	Yes
Exploitation Tools	No	Yes	Yes	Yes
Automation Level	High	High	High	High
CWE Database Mapping	No	Yes	No	**Yes**
Output and Reporting	Basic	Detailed	Detailed	Basic but informational
AI-powered	No	No	No	Yes
Target Users	Experienced users	Beginner and experienced users	Intermediate to advanced users	Beginners
Performance	Fast and efficient	Efficient but depends on modules	Heavy and resource-intensive	Optimised for efficiency

2.8 Critical Analysis

Similar existing web application penetration testing tools are robust but often prioritise advanced functionality over accessibility. Beginners face significant barriers due to the complexity of CLI. Tools like ASM, jok3r framework and Darkarmy are powerful but lack the structured guidance for novices to build foundational skills. WABUHA addresses these issues by offering:

- A user-friendly graphical interface (GUI)
- Structural web app penetration testing workflows by implementing the simplified PTES framework
- CWE Mapping
- AI-powered vulnerability explanation

2.9 Summary of Findings

This chapter explored fundamental ideas including defining scope, reconnaissance, exploitation, and vulnerability discovery while reviewing important penetration testing techniques, such as OWASP PTES. Analysis of the distinct benefits and drawbacks of similar existing tools to WABUHA such as Attack Surface Mapper (ASM), jok3r framework, and Darkarmy revealed gaps that the proposed WABUHA framework addresses. The findings also provide a solid foundation for developing beginner-friendly, structured bug-hunting solutions by combining theoretical understanding with practical evaluations.

Chapter 3: Requirements Analysis and Design

3.1 Introduction

The requirements analysis and design phase of the WABUHA is described in this chapter. The methodology used, the requirements specification obtained through questionnaires, and the design considerations and user needs are all thoroughly covered in this section. It also provides the foundation and reasoning for the next design phase by going over the hardware and software requirements for the system.

3.2 Agile Scrum Methodology

The Agile Scrum framework is applied when developing WABUHA. Flexible and iterative, agile work methods are widely recognised to allow developers to focus on what users need and be flexible in the case of evolving requirements (Cohn, 2009). The concepts of Agile Scrum are as useful for solo developers overseeing complicated projects, where the goals and scope may change over time, even if it was first created.

In this approach, development is broken down into a series of time-constrained iterations known as sprints. Each sprint focuses on completing a set of features or functionality, and therefore a rolling deliverable. This nature is also consistent with the goals of WABUHA as it could be repeated to test and iteratively explore.

The benefit of Agile for WABUHA is its ability to adapt changes. The development process is an iterative process with changes, including requirements refinement, feedback from the users, or overcoming technical constraints. Therefore, we can adapt these changes without it affecting workflow in any way. This flexibility is crucial as the application is to integrate a number of tools, APIs, and features.

The Agile Scrum approach can further manage the project time well (Cohn, 2009). Through the well-defined goals for each sprint and by focussing on achieving specific results, to ensure that steady progress is achieved with a consistent overview of particularly important work.

In summary, the Agile Scrum methodology is well-suited for the development of WABUHA due to its iterative, feature-centric approach, adaptability to changes, and emphasis on continuous improvement.

3.3 Simplified PTES Framework

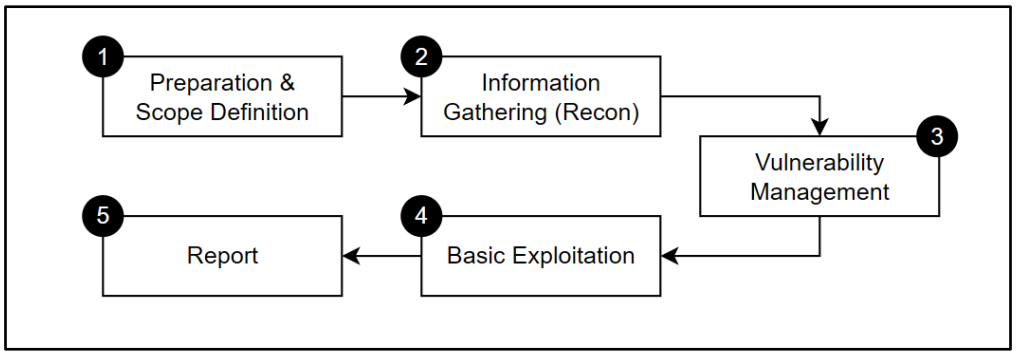


Figure 3.1: Simplified PTES Framework.

Preparation Scope Definition, Information Gathering, Vulnerability Management, Basic Exploitation, and Reporting. All phases play an important role in making a complete and efficient security survey of the target system.

Preparation and Scope Definition is the first stage where the rules of engagement for the penetration test are established. In this stage, the target domain, subdomain, IP addresses, and other parameters need to be defined. Appropriate preparation and scope definition form the base for the entirety of the process, so expectations are aligned where testing falls within the definition of the scope.

Information Gathering, or reconnaissance, is the process of obtaining information about the system or network to be targeted. In this phase, both passive and active methods are used for finding possible weaknesses. Successful information collection is a key step towards grasping the target environment and understanding the vulnerabilities that may be exploited in the subsequent stages. The data gathered in this phase feeds into the strategies and the tools that will be utilised in the vulnerability management and exploitation phases.

Vulnerability Management comprises discovery, analysis, and prioritisation of vulnerabilities encountered in the recon phase. The goal is to assess the risk associated with each vulnerability and determine the most effective remediation strategies. Effective vulnerability management plays a key role in reducing risk and enhancing the security state of the target system, ensuring that the vulnerabilities are addressed in a systematic and efficient manner.

The intention of the basic exploitation is to take a chance on the discovered vulnerabilities to achieve at least some level of control or unauthorised access to the target system. This phase aims to prove the existence of vulnerabilities and provide information on their potential impact. Basic exploitation offers important insights for enhancing defenses and helps in understanding the practical impacts of security flaws. It is a test where multiple exploits are executed to understand how vulnerable the target system is to various attack methods.

Documenting the findings, methodologies, and recommendations from the penetration test stage is the report stage. The report is expected to acquire a full understanding of weaknesses and the steps that need to be taken to fix them, and it should be concise and actionable, by giving details of weaknesses and suitable mitigations.

3.4 Requirements Specification

This section outlines the questionnaire that was completed, as well as the analysis process that was used to determine the system requirements related to WABUHA. It includes the hardware and software requirements necessary for the development process to be followed and justification as to how the survey outcomes relate to the goals of WABUHA.

3.4.1 Survey and Questionnaire

Surveys and questionnaires are popular instruments for collecting quantitatively and qualitatively data, and especially valuable for the user understanding of users' needs, wants and problems. A literature review, presented in Chapter 2, helped to define the gaps in the nature of penetration testing tools and techniques, especially for novices. Given the above background, there was a selection of questionnaire as data collection tool for the present study. The questionnaire was hosted on the internet via Google forms, which allows its easy access to the subjects and also facilitates collecting the data efficiently. A total of 15 respondents participated in the survey, consisting mainly of students and aspiring penetration testers, providing insights into the challenges faced by beginners in the field. The complete questionnaire can be found in *Appendix A*.

Respondent demographics. The survey was performed on a population that is younger, aged between 18 and 24 year olds. This demographic is composed mainly of students, together with a less significant number of aspired penetration testers. According to (see Figure 3.2) respondents, most of them stated they did not have professional penetration testing or cybersecurity experience. In terms of past experience, most of those with experience reported less than one year of practice. This is in line with the target user base of WABUHA, which appeals to novice and new users of penetration testing techniques.

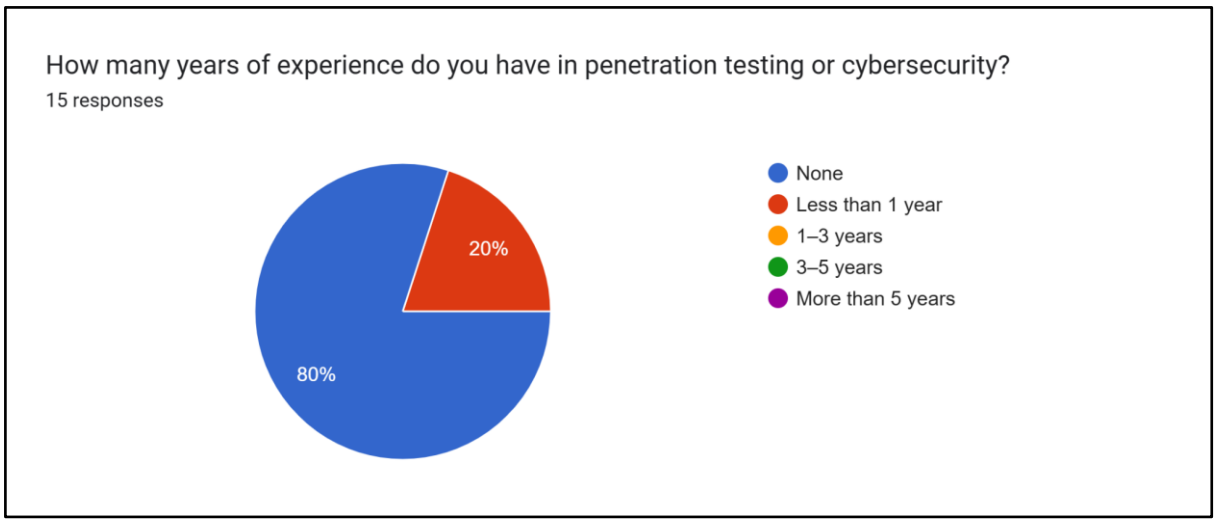


Figure 3.2: Distribution of Respondent Demographics.

Respondents were also asked with respect to their proficiency (in web application penetration testing). Most identified themselves as beginners, with only a few reporting intermediate skills. This is representative of the need to design such a tool as WABUHA with the experience of the novice all in mind and to offer easily available features to aid the learning process and participation. In addition, a large majority of respondents reported that they rarely or never perform penetration testing, thereby underscoring the imbalance in accessibility and WABUHA seeks to tackle this gap.

The questionnaire for user acceptance testing was developed by referencing usability studies and educational goals outlined in the literature review. Key considerations included challenges faced by beginners using CLI-based tools, the need for guided workflows, and the impact of AI-powered vulnerability explanations. Each question was tailored, clear functionality, feature usefulness, and user satisfaction. This ensured the collected feedback was not only reflective of usability but also of how well the tool supports practical learning for beginners.

Challenges faced by beginners. Participants highlighted several challenges associated with penetration testing. Next was the challenge of figuring out the role of the tools individually

and what tools to use in a specific task. This ambiguity arises because there are many tools, some of which are complex, and targeted towards experienced users. Furthermore, a number of respondents complained that there existed a high learning curve for such tools, in particular due to the lack of hands-on educational resources.

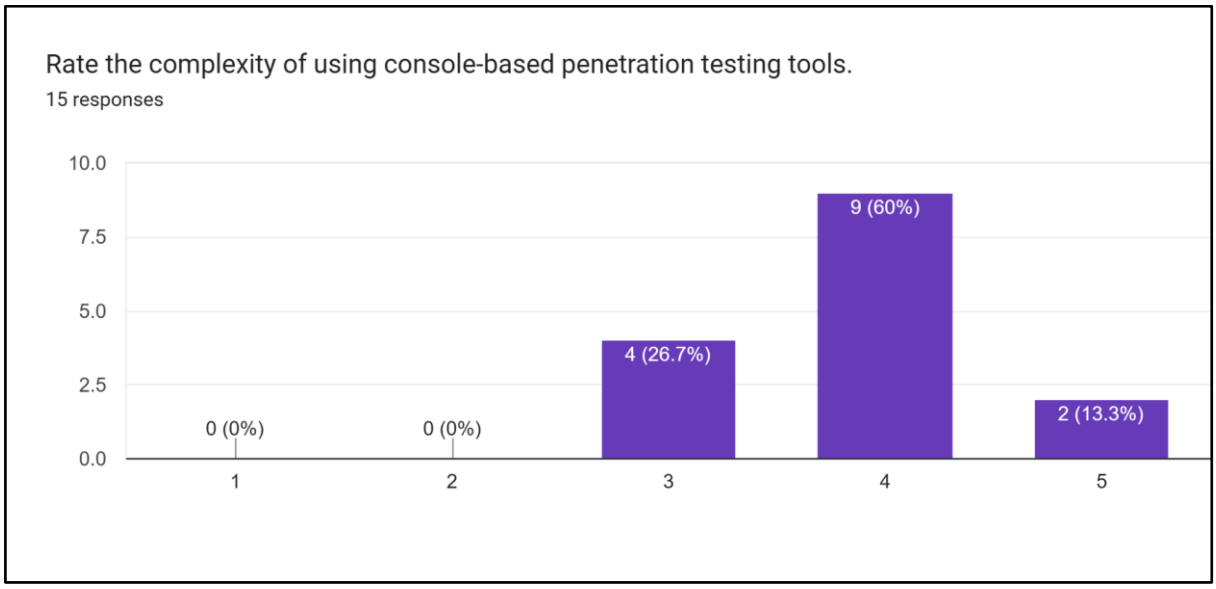


Figure 3.3: Complexity of Using Console-based Pentest Tools.

The complexity of console-based penetration tools was also identified as another major issue. These instruments typically require users to interact with command-line interfaces, which for beginners can be quite daunting. Respondents assessed the complexity of these applications on a scale of 1 to 5 with the ratings generally falling at the range of 3 to 5 (see Figure 3.3). This feedback also highlights the importance of having a graphical user interface (GUI) to reduce the learning curve for the novice.

Time investment was also noted as a hindrance, as beginners often find it time-consuming to learn and effectively use existing tools. Some respondents pointed out that the lack of centralised management for discovery processes further complicates the penetration testing workflow.

Time investment was also identified as an issue, since early adopters often struggle with time and effectiveness of current tools. Several respondents also noted that, in addition to the absence of a central workflow coordination mechanisms makes the penetration testing workflow even more complex.

Preferred features in penetration testing tools. The questionnaire aimed to determine the characteristics that respondents identified as most relevant when designing a penetration testing tool. First and foremost, a user-friendly interface was overwhelmingly supported. Participants stressed the need for a Graphical User Interface (GUI) which allows an easy way of the interaction and lower the intimidation factor for the novice. Figure 3.4 demonstrates that a guided and organised workflow was one other very common feature, since it gives a user a defined route through each step of penetration testing.

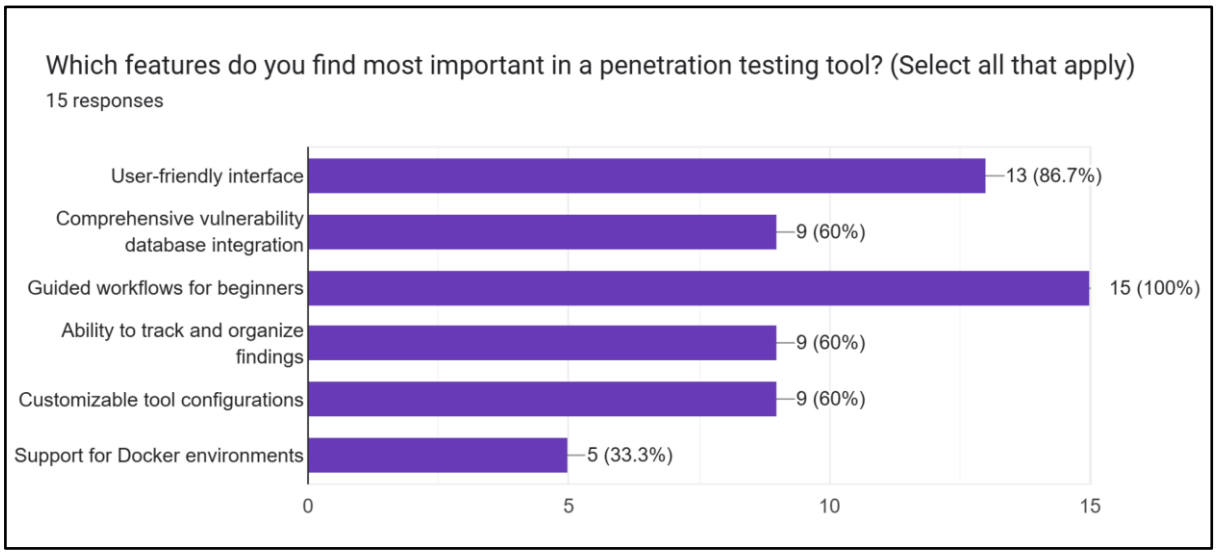


Figure 3.4: Ranking of Preferred Features in Penetration Testing Tools.

Customisability was also a major contributing factor, as users wanted to be able to customise tools for their own needs. For example, setting up the testing parameters or scripts

was presented as an important feature. Finally, support of Docker environments was reported as useful for controlled and safe test environments.

Tracking and organising the findings was often reported. Respondents expressed a need for tools facilitating centralisation of discoveries, so that findings could be documented and referenced easily. Based on Figure 3.5, full integration of databases including the Common Vulnerabilities and Exposures (CVE) and the Common Weakness Enumeration (CWE) databases. This integration assists users in contextualising findings, which improves both the usability and informative value of those findings.

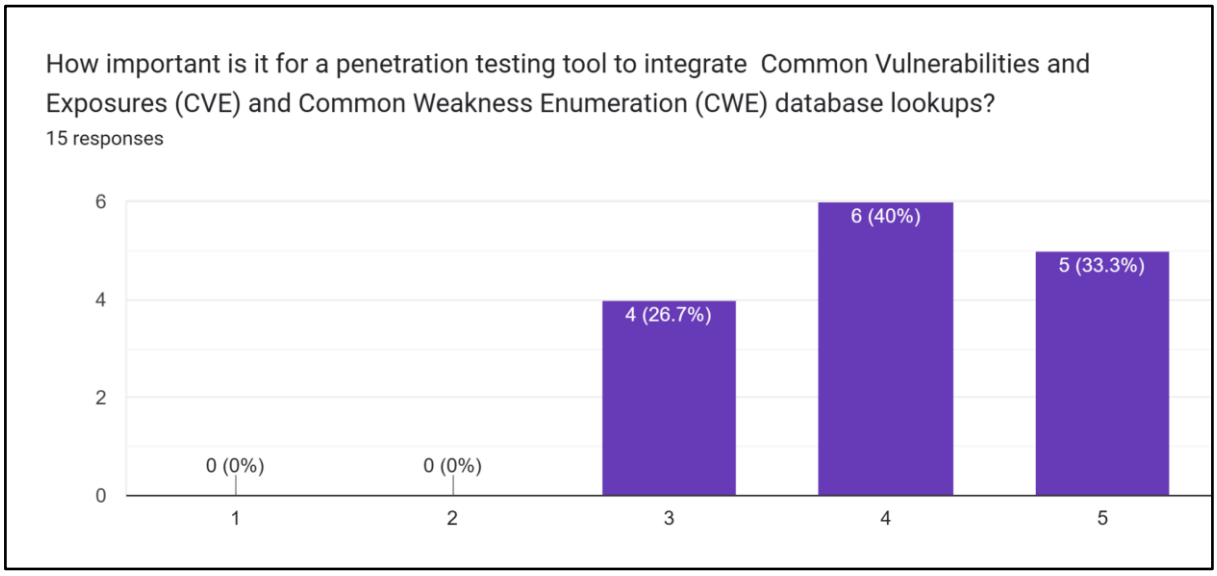


Figure 3.5: Ranking of Preferred Features in Penetration Testing Tools.

Desirable tool characteristics. The survey also investigated features that would increase the usability and attractiveness of penetration testing tools. Accessibility became a primary subject, and respondents agreed strongly that a GUI would allow penetration testing to be made easier for a general audience. Automation was another key consideration. A high percentage of respondents also indicated interest in functionalities such as one-click operation to streamline testing workflows.

Educational assistance was commonly reported in the qualitative responses, with users suggesting software containing tutorials, guided applications, and video instructions. The following characteristics are considered indispensable to bridging the knowledge gap and offering users with concrete skills. Risk assessment capabilities were also deemed important, with respondents requesting tools that categorise vulnerabilities by severity.

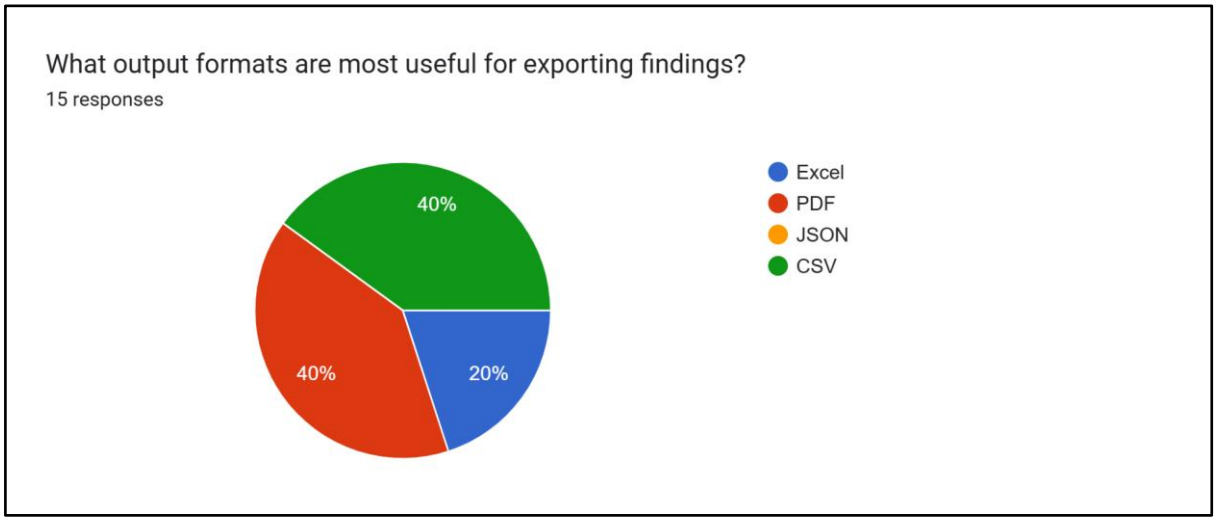


Figure 3.6: Preferred Output Formats for Findings.

Export options and reporting. Respondents were asked on their output format preferences for exporting results. Formats most frequently reported were PDF, CSV and Excel (see Figure 3.6). These formats are common and allow users to produce high-quality reports for documentation and subsequent analysis. Moreover, participants highlighted the value of user configurable reporting, whereby users are able to find a format that best suit these needs.

User expectations of WABUHA. The survey ended with questions measuring user's expectations and the intention regarding WABUHA's adoption. A high proportion of respondents expressed the high likelihood of tool use should the tool satisfy their requirements. Key factors influencing adoption include the presence of guided workflows, educational resources, and an intuitive GUI.

If asked regarding their preferred approaches of learning new tools, the subjects ranked step-by-step tutorial, interactive guide and video demonstrations as the most suitable one. This feedback also confirms the necessity of applying strong educational support into the design of WABUHA.

3.4.2 Development System Requirements

The development of WABUHA requires a specific set of hardware, software, and tools tailored to a solo developer environment. The following outlines the key development requirements for the successful creation, testing, and deployment of the system.

3.4.2.1 Hardware Requirements

These hardware specifications are to ensure smooth operation during development and testing, allowing for seamless execution of the application and related tasks. The hardware specifications for developing WABUHA are shown in Table 3.1.

Table 3.1: Development Hardware Requirement.

Minimum	Requirement	Recommended
2.0 GHz	**Processor**	3.0 GHz or higher.
8 GB	**RAM**	16 GB
50GB	**Storage**	50GB SSD

3.4.2.2 Software Requirements

Table 3.2 shows the software tools and dependencies that are crucial for the development of WABUHA.

Table 3.2: Development Software Requirement.

Software	**Description**
Operating System	A Linux-based operating system, such as Kali Linux, is recommended due to the recon and exploit tools compatibility.
Programming Languages	Python 3.x
Libraries and Frameworks	● PyQt5 Designer for the development of the graphical user interface (GUI). ● SQLite for the local database management ● Requests for managing HTTP requests during the reconnaissance phase. ● Docker for creating isolated environments that simulate penetration testing scenarios.
Version Control	Git for version control, enabling code tracking and management of revisions.
Development Tools	VSCode. It offers features such as debugging, version control integration, and plugin support, making it an ideal choice.
Containerisation	Docker will be used to create controlled environments for simulating real-world penetration testing scenarios, ensuring the application can be tested safely and in isolation.
Documentation and Reporting Tools	Markdown file.

In summary, the development environment for WABUHA has been carefully designed to suit the needs of a solo developer. By leveraging a combination of open-source tools and a flexible coding platform like VSCode, the project can proceed efficiently, with the necessary resources to ensure successful development and deployment.

3.4.3 Requirement Justification

The findings from the survey provide a strong justification for the design and features of WABUHA. Several requirements are directly supported by user feedback.

First and foremost, by having a beginner-friendly GUI in WABUHA. The overwhelming preference for a graphical user interface highlights the need for accessibility and ease of use. By replacing complex console-based interactions with a well-designed GUI, WABUHA can lower the barrier to entry for beginners and enhance user engagement.

Secondly, to have a guided workflow in WABUHA. Structured workflows are repeatedly emphasised as a critical feature. WABUHA is designed to guide users through the simplified penetration testing phase, directly addresses this need and helps reduce confusion for novice users.

Next is the importance of CVE and CWE database integration and mapping is strongly supported by respondents. This feature ensures that users can contextualise findings and access relevant vulnerability information, improving the overall utility of the tool.

Customisability is also a recurring theme, with users requesting flexibility in configuring tools and testing parameters instead of a complete “black box”. "Black box" pentesting refers to a testing scenario where the penetration tester has minimal or no prior knowledge about the target system or network. The customisable configurations in WABUHA will allow users to adapt the tool to their specific requirements.

The demand for risk assessment summaries highlights the need for tools that can categorise vulnerabilities by severity. Displays vulnerabilities categorised by their type and as High, Medium, Low, or Info, directly address this requirement.

3.5 Requirement Analysis

This section presents the detailed requirements analysis for the WABUHA application, focusing on its system design, user interactions, and backlog development.

3.5.1 Use Case Diagram

A Use Case diagram is a graphical representation, which shows the relationship between the system and a user (actor), representing the system's capabilities and how these are used. It is a type of Unified Modeling Language (UML). It offers an abstract overview of the workflows of the system, with emphasising the functional roles of different actions and their corresponding actors (GeeksforGeeks, 2023). Figure 3.7 shows the WABUHA use case diagram, which represents the system's main features and operations. This approach makes it easier to understand how WABUHA works.

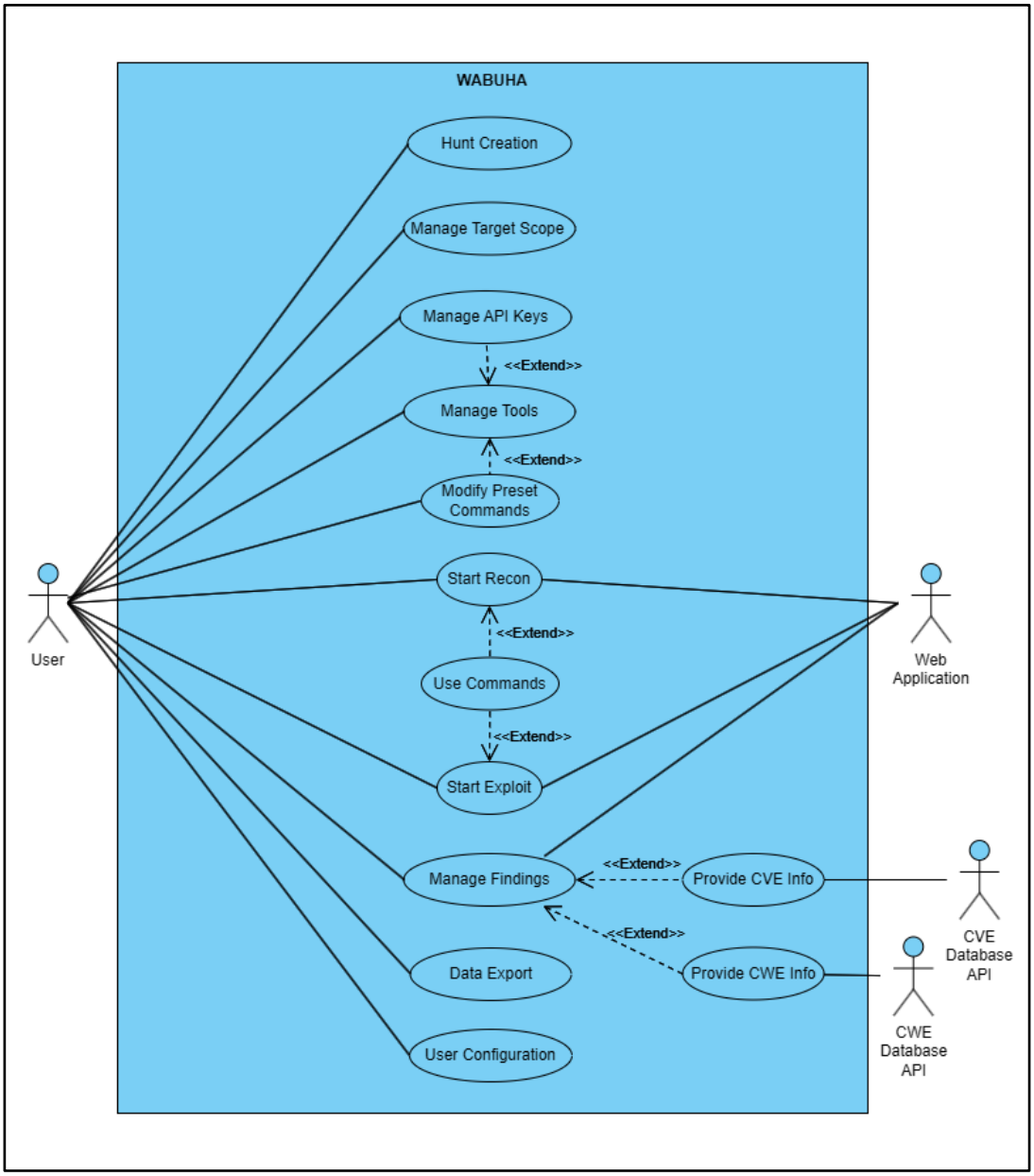


Figure 3.7: A Use Case Diagram of WABUHA.

Figure 3.7 presents a structured representation of the interactions between the user and the WABUHA system in a Use Case diagram. It effectively captures the system's core functionalities, emphasising the role of the user in executing and managing various web application penetration testing processes. The diagram outlines several essential use cases, each representing a critical aspect of the system's functionality, Table 3.3 is the description and explanation of the use cases.

Table 3.3: Use Case Explanations.

No	Use Case	Explanation
1	Hunt Creation	Allows the user to create a new testing session, organising activities under a specific session.
2	Manage Target Scope	Allow the user to define testing parameters such as domains, IP ranges, and subdomains.
3	Manage API Keys	Allows secure configuration and usage of API keys for external tools and services.
4	Manage Tools	Facilitates selection, configuration, and customisation of tools for reconnaissance and exploitation.
5	Modify Preset Commands	Provides functionality to customise default configurations and commands for the tools.
6	Start Recon	Initiates the reconnaissance phase to gather information about the target web application.
7	Use Commands	Enables execution of specific tool commands during reconnaissance and exploitation phases.
8	Start Exploit	Launches the exploitation phase to identify vulnerabilities in the target web application.
9	Manage Findings	Aggregates and organises findings from the reconnaissance and exploitation phases for analysis.
10	Provide CVE Info	Queries the CVE Database API to retrieve detailed information about identified vulnerabilities.
11	Provide CWE Info	Queries the CWE Database API to provide information about the associated weaknesses of the vulnerabilities.
12	Data Export	Enables exporting findings and reports into structured formats such as Excel for documentation.
13	User Configuration	Allows customisation of application settings to improve user experience and usability.

The diagram highlights how the user interacts with WABUHA to perform various testing tasks, with the ultimate goal of collecting actionable insights about the web application's vulnerabilities.

The use case diagram provides a clear and concise overview of WABUHA's functionalities and user interactions. It is a valuable resource for understanding the system's design and operation, ensuring that the user can effectively navigate the tool to conduct ethical and efficient penetration testing.

3.5.2 Use Case Specification

Table 3.4: Use Case Specification for Hunt Creation.

Use Case	Hunt Creation
Actor	User
Pre-Condition	● User has access to WABUHA system ● User has necessary permissions
Post-Condition	● New hunt is created in the system ● Session logging level is configured
Main flow	1. User launches WABUHA application 2. User selects "Create New Hunt" option 3. User enters hunt name 4. User selects session log verbosity level 5. System validates input 6. System creates new hunt session 7. System confirms hunt creation
Alternative flow(s)	A1: Invalid hunt name 1. System displays error message 2. User enters different hunt name 3. Returns to main flow step 5
Exception flow(s)	E1: System unable to create hunt 1. System displays error message 2. System logs error 3. User returns to main screen

Table 3.5: Use Case Specification for Manage Target Scope.

Use Case	Manage Target Scope
Actor	User
Pre-Condition	● Hunt has been created ● User is in Config tab
Post-Condition	● Target scope is defined and saved ● Scope parameters are validated
Main flow	1. User navigates to Define Scope section 2. User enters target domain 3. User specifies additional scope parameters (subdomains, IPs, ports, URLs) 4. User marks allowed/restricted areas 5. System validates scope parameters 6. System saves scope configuration
Alternative flow(s)	A1: Invalid domain format 1. System highlights invalid domain 2. User corrects domain format 3. Returns to main flow step 5
Exception flow(s)	E1: Scope validation failure 1. System displays validation errors 2. System prevents scope saving 3. User must correct errors

Table 3.6: Use Case Specification for Manage Tools.

Use Case	Manage Tools
Actor	User
Pre-Condition	● Hunt is created ● Target scope is defined ● User has configured API keys
Post-Condition	● Tools are selected and configured ● Tool configurations are saved
Main flow	1. User navigates to Tools Selection section 2. System displays available reconnaissance tools 3. System displays available exploitation tools 4. User selects desired tools 5. User reviews tool configurations 6. System validates tool selection 7. System saves tool configuration

Alternative flow(s)	A1: Missing API keys 1. System identifies tools requiring API keys 2. User navigates to API Keys section 3. Returns to main flow step 4
Exception flow(s)	E1: Tool compatibility issues 1. System displays compatibility warnings 2. User must adjust selection 3. Returns to tool selection

Table 3.7: Use Case Specification for Start Recon.

Use Case	Start Recon
Actor	User
Pre-Condition	● Tools are selected and configured ● Target scope is defined
Post-Condition	● Reconnaissance tasks are initiated ● Findings are stored in database
Main flow	1. User navigates to Recon tab 2. User reviews selected tools 3. User clicks Start Recon button 4. System initiates async tool execution 5. System displays progress 6. Targeted web application returns the query results 7. System stores findings in database 8. System displays findings in Finding Panel
Alternative flow(s)	A1: Custom script execution 1. User adds custom scripts 2. System validates scripts 3. Returns to main flow step 4
Exception flow(s)	E1: Tool execution failure 1. System logs error 2. System notifies user 3. System continues with remaining tools

Table 3.8: Use Case Specification for Start Exploit.

Use Case	Start Exploit
Actor	User
Pre-Condition	● Reconnaissance phase completed ● Findings are reviewed and filtered
Post-Condition	● Exploitation results are mapped to CVE/CWE ● Results are stored in the database
Main flow	1. User navigates to Exploit tab 2. User reviews selected exploitation tools 3. User clicks Start Exploit button 4. System executes tools asynchronously 5. Targeted web application returns the query results 6. System maps results to CVE/CWE databases 7. System stores results 8. System displays findings
Alternative flow(s)	A1: Custom exploit scripts 1. User adds custom exploit scripts 2. System validates scripts 3. Returns to main flow step 4
Exception flow(s)	E1: CVE/CWE mapping failure 1. System logs mapping error 2. System continues without mapping 3. User can manually map later

Table 3.9: Use Case Specification for View Findings.

Use Case	Manage Findings
Actor	User, CVE Database API, CWE Database API
Pre-Condition	● Exploitation phase completed ● Results are mapped and stored
Post-Condition	● Findings are displayed with risk categories ● Detailed vulnerability information is available
Main flow	1. User navigates to Report tab 2. System retrieves findings from database 3. System displays risk count categories 4. System displays CVE/CWE lists 5. User clicks on specific finding 6. System retrieves detailed information 7. System displays vulnerability details

Alternative flow(s)	A1: Filter findings 1. User applies filters 2. System updates display 3. Returns to main flow step 4
Exception flow(s)	E1: API connection failure 1. System displays cached information 2. System notifies user of connection issue 3. User can retry API connection

Table 3.10: Use Case Specification for Export Data.

Use Case	Export Data
Actor	User
Pre-Condition	● Findings are available ● Summary is generated
Post-Condition	● Results are exported to Excel file ● Export is validated
Main flow	1. User selects Export Data option 2. System prepares data for export 3. System generates Excel file 4. System includes all findings and results 5. System saves file to user-specified location 6. System confirms successful export
Alternative flow(s)	A1: Custom export format 1. User selects specific data to export 2. System filters export content 3. Returns to main flow step 3
Exception flow(s)	E1: Export failure 1. System displays error message 2. System logs error details 3. User can retry export

Table 3.11: Use case specification for User Configuration.

Use Case	User Configuration
Actor	User
Pre-Condition	● User has access to WABUHA system
Post-Condition	● User preferences are saved ● API keys are stored securely
Main flow	1. User navigates to Config tab 2. User enters/updates API keys 3. User sets preferences 4. System validates configuration 5. System encrypts sensitive data 6. System saves configuration
Alternative flow(s)	A1: Invalid API key 1. System validates API key 2. System notifies user of invalid key 3. Returns to main flow step 2
Exception flow(s)	E1: Configuration save failure 1. System retains previous configuration 2. System notifies user of save failure 3. User can retry saving

3.5.3 Scrum Product Backlog

According to Cohn (2009), the Scrum Product Backlog is a ranked list of all of the features, functions and requirements of the WABUHA application. It captures the objectives and requirements of the project. The backlog is constantly updated with new features, and tasks are ranked according to their importance and timeliness. Every item on the backlog, called a user story, consists in an explanation of the feature, the acceptance criteria (the conditions that must be fulfilled for the feature to be considered finished), estimated story points (an ordinal measure of the amount of work involved in implementing the feature) and priority (the importance level attached to work among other backlog items). The product backlog is critical for the development process.

Table 3.12: Story 1.

ID	1
Story Description	As a user, I want to create a new hunt with a custom name and log verbosity.
Acceptance Criteria	● The system allows the creation of a hunt. ● A custom name can be provided. ● Log verbosity is selectable.
Story Points	3
Priority	High

Table 3.13: Story 2.

ID	2
Story Description	As a user, I want to define the target scope such as domain, subdomains, IPs, URLs and ports.
Acceptance Criteria	● The user can input domains, subdomains, IPs, URLs, and ports. ● Scope parameters are saved for later use.
Story Points	5
Priority	High

Table 3.14: Story 3.

ID	3
Story Description	As a user, I want to manage sessions for web pages during reconnaissance and exploitation.
Acceptance Criteria	● The system allows the user to manage sessions (cookies, headers, etc.). ● Session data is maintained for the web pages.
Story Points	5
Priority	High

Table 3.15: Story 4.

ID	4
Story Description	As a user, I want to select reconnaissance tools for use during the recon phase.
Acceptance Criteria	● The user can select tools from a list of checkboxes. ● Tools are saved and available in the recon phase.
Story Points	5
Priority	High

Table 3.16: Story 5.

ID	5
Story Description	As a user, I want to run reconnaissance tools asynchronously during the recon phase.
Acceptance Criteria	● Tools are executed concurrently in background. ● Recon results are displayed in the General Information panel.
Story Points	8
Priority	High

Table 3.17: Story 6.

ID	6
Story Description	As a user, I want to view reconnaissance findings and manage them (exclude, save).
Acceptance Criteria	● Findings from the recon phase are displayed. ● Users can exclude findings from exploitation. ● Changes are saved.
Story Points	5
Priority	High

Table 3.18: Story 7.

ID	7
Story Description	As a user, I want to start exploitation with selected tools.
Acceptance Criteria	● The user can select exploitation tools. ● The tools execute and map results to CVE/CWE databases.
Story Points	8
Priority	High

Table 3.19: Story 8.

ID	8
Story Description	As a user, I want to select a user-agent string for use in reconnaissance and exploitation.
Acceptance Criteria	● Users can select from a list of user-agent strings. ● The system applies the selected string to requests.
Story Points	3
Priority	Medium

Table 3.20: Story 9.

ID	9
Story Description	As a user, I want to manage API keys for tools.
Acceptance Criteria	● The system allows the user to input and store API keys. ● API keys can be edited or deleted.
Story Points	3
Priority	Medium

Table 3.21: Story 10.

ID	10
Story Description	As a user, I want to view exploitation findings linked to CVE and CWE details.
Acceptance Criteria	● Exploitation findings are shown. ● Links to CVE and CWE databases are accessible.
Story Points	8
Priority	Medium

Table 3.22: Story 11.

ID	11
Story Description	As a user, I want to modify preset commands for reconnaissance and exploitation tools.
Acceptance Criteria	● Users can customise preset commands. ● Changes are saved and applied during tool execution.
Story Points	3
Priority	Medium

Table 3.23: Story 12.

ID	12
Story Description	As a user, I want to view all tool details (name, type, description, origin, commands).
Acceptance Criteria	● Tool details are displayed for each selected tool. ● Users can view and understand tool functions.
Story Points	5
Priority	Medium

Table 3.24: Story 13.

ID	13
Story Description	As a user, I want the system to track and manage session logs throughout the process.
Acceptance Criteria	● All session logs are saved. ● Logs can be accessed from the interface based on verbosity settings.
Story Points	5
Priority	Medium

Table 3.25: Story 14.

ID	14
Story Description	As a user, I want to see a summary of risk counts categorised as High, Medium, Low, and Info.
Acceptance Criteria	● Risk counts are categorised. ● The summary view displays categorised risks.
Story Points	5
Priority	High

Table 3.26: Story 15

ID	15
Story Description	As a user, I want to export the findings and results into an Excel file.
Acceptance Criteria	● Users can export data into an Excel file. ● File includes all relevant findings and results.
Story Points	3
Priority	Low

3.5.4 Scrum Sprint Backlog Weekly Plan

The following sprint backlog planning for WABUHA shown in Table 3.27 is derived from the product backlog, with each task referenced by its corresponding story ID. The project is structured over 3 months, with each month consisting of a 4-week sprint. Throughout each sprint, tasks will focus on key activities such as refining the user interface (UI/UX), optimising the database, implementing new features, and conducting comprehensive testing for each developed feature.

Table 3.27: Sprint Backlog Planning.

ID	Priority	Task	Sprint 1				Sprint 2				Sprint 3			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1	High	Refine UI/UX	✓											
		Optimise database	✓											
		Implement features		✓										
		Testing			✓									
2	High	Refine UI/UX	✓											
		Optimise database	✓											
		Implement features		✓										
		Testing			✓									
3	High	Refine UI/UX	✓											
		Optimise database	✓											
		Implement features		✓										
		Testing			✓									
4	High	Refine UI/UX	✓				✓							
		Optimise database	✓				✓							
		Implement features		✓	✓			✓	✓			✓	✓	
		Testing				✓				✓				✓
5	High	Refine UI/UX	✓				✓							
		Optimise database	✓				✓							
		Implement features						✓	✓					
		Testing								✓				
6	High	Refine UI/UX	✓				✓							

ID	Priority	Task	Sprint 1				Sprint 2				Sprint 3			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
		Optimise database	✓				✓							
		Implement features						✓	✓					
		Testing								✓				
7	High	Refine UI/UX	✓				✓							
		Optimise database	✓				✓							
		Implement features						✓	✓					
		Testing								✓				
8	Medium	Refine UI/UX	✓				✓							
		Optimise database	✓				✓							
		Implement features						✓	✓					
		Testing								✓				
9	Medium	Refine UI/UX	✓				✓							
		Optimise database					✓							
		Implement features						✓	✓					
		Testing								✓				
10	Medium	Refine UI/UX	✓				✓							
		Optimise database					✓							
		Implement features						✓	✓			✓	✓	
		Testing								✓				✓
11	Medium	Refine UI/UX	✓				✓							
		Optimise database					✓							
		Implement features						✓						

ID	Priority	Task	Sprint 1				Sprint 2				Sprint 3			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
		Testing							✓					
12	Medium	Refine UI/UX	✓								✓			
		Optimise database									✓			
		Implement features										✓	✓	
		Testing												✓
13	Medium	Refine UI/UX	✓								✓			
		Optimise database									✓			
		Implement features										✓	✓	
		Testing												✓
14	Medium	Refine UI/UX	✓								✓			
		Optimise database									✓			
		Implement features										✓	✓	
		Testing												✓
15	Low	Refine UI/UX	✓								✓			
		Optimise database									✓			
		Implement features										✓	✓	
		Testing												✓

3.6 System Design

This section elaborates on the design aspects of WABUHA, focusing on how the system is structured. The design flow ensures that the application is time efficient and easy to use, meanwhile being modular and scalable. Important design features including database schema, data flow, and user interfaces mockups.

3.6.1 Database Design

Design of the database is of fundamental importance to WABUHA, since it provides storage, retrieval and organisation of data that is necessary for undertaking penetration testing activities. SQLite was selected as the database management system for WABUHA, as it was a lightweight and self-contained database management system. It does not require a separate server, making it easy to set up and deploy while maintaining high reliability and performance for local data management.

SQLite is an open-source relational database management system with a serverless architecture. This implies that all the information is contained in a single file, which makes both portability and backup much easier (Priy, 2018).

3.6.1.1 Crow's Foot Diagram

Since WABUHA supports data management, a good database design is essential. The high-level conceptual design shown in Figure 3.8 effectively outlines the fundamental structure of the WABUHA database, emphasising the primary entities and their interactions. This schematic serves as a foundational blueprint, guiding the subsequent detailed design and implementation phases of the database, thereby ensuring the robust and efficient operation of the application.

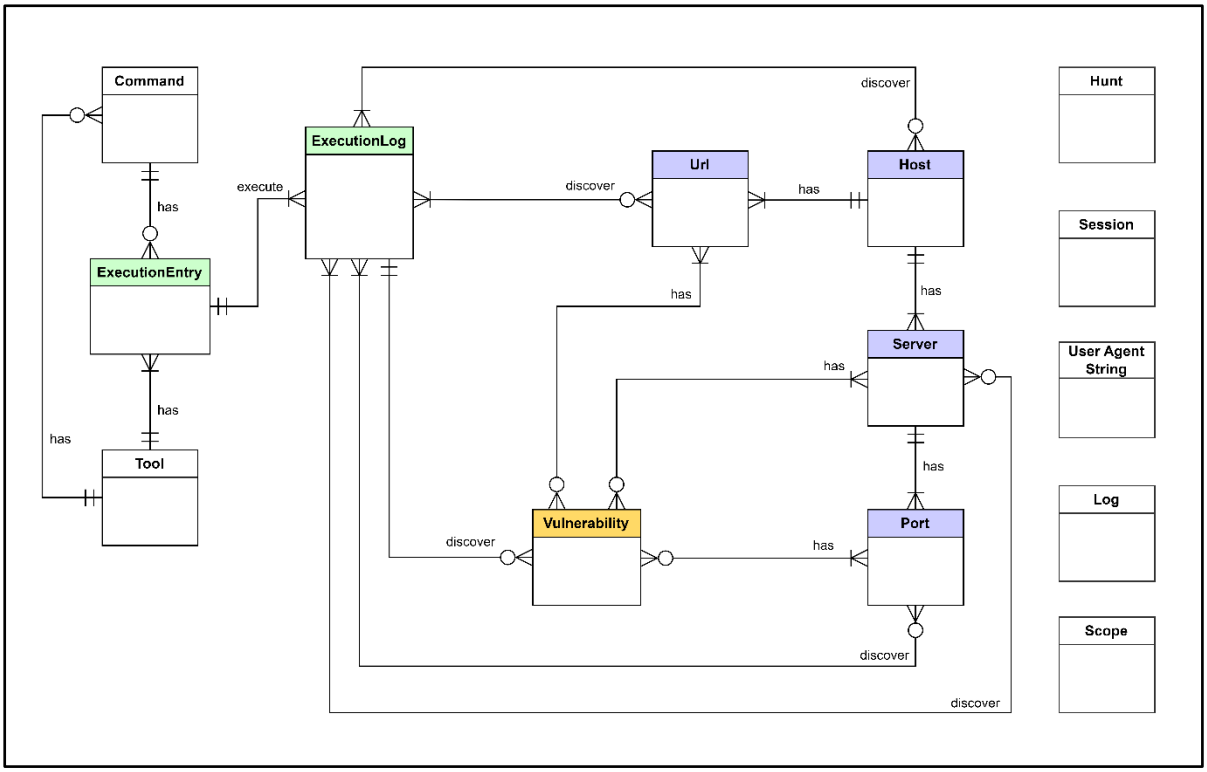


Figure 3.8: Conceptual Model.

Figure 3.9 illustrates the complete Crow's Foot diagram with listed attributes for each entity (table) and data type. "User Agent String" refers to the User-Agent string where users can select or customise to mimic network header traffic.

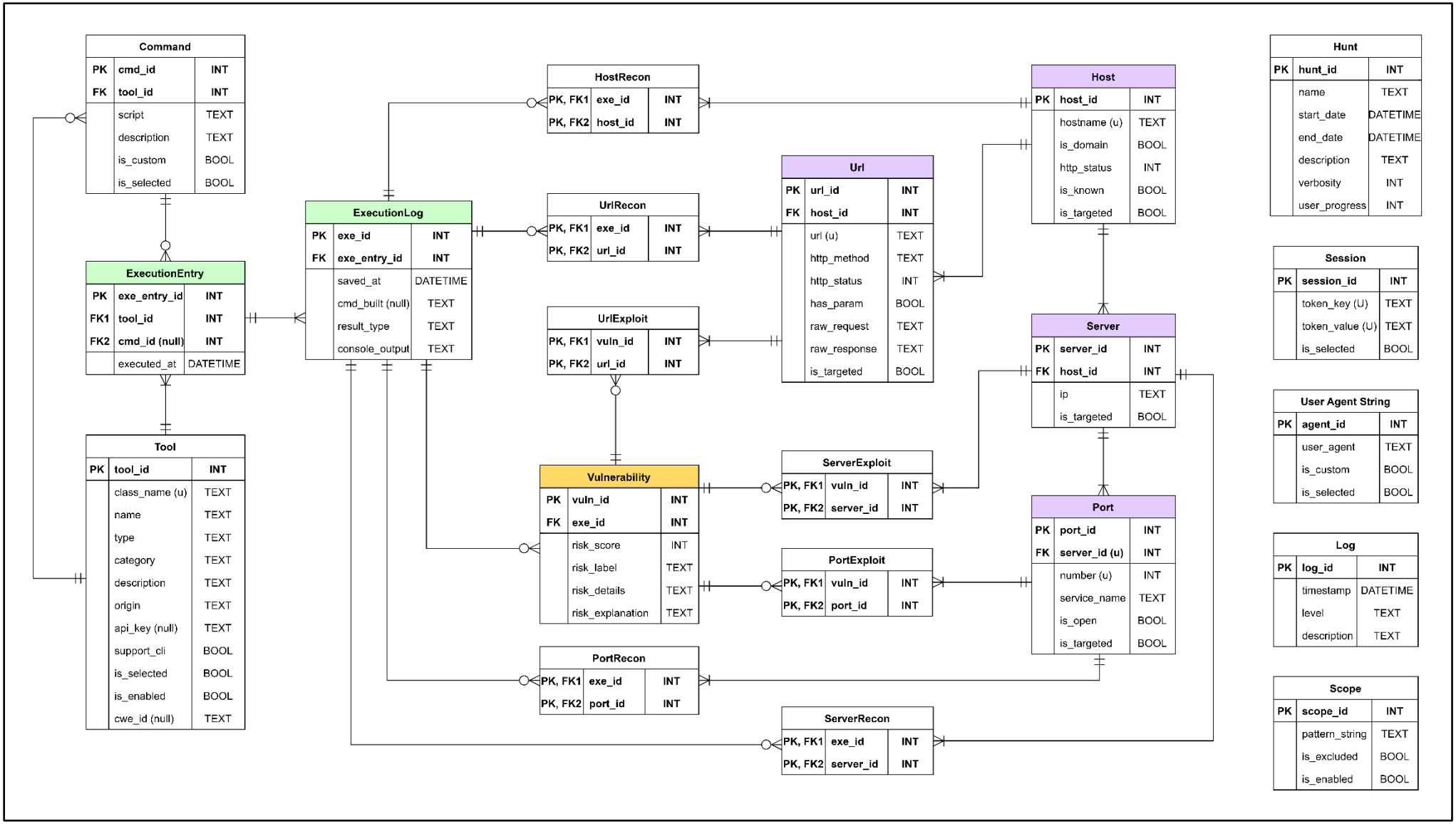


Figure 3.9: Detailed Crow's Foot ERD.

3.6.1.2 Data Dictionary

This section describes the data dictionary, an essential component of the data model, by drawing on Crow's Foot Diagram in Figure 3.9. The data dictionary provides a guide to the structure, attributes and dependencies of the database tables that are accessed in WABUHA. Representing the data dictionary as a table facilitates documentation of the schema, including the names of the fields, data type, description and constraint for each table.

Table 3.28: Hunt Table.

Table Name	Hunt			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
hunt_id	INT	Unique ID for each new hunt	PK	
name	TEXT	Hunt name		
start_date	DATETIME	Date start hunting		
end_date	DATETIME	Date complete hunting		
description	TEXT	Hunting details	Nullable	
verbosity	INT	Logging level to display		
user_progress	INT	Track user tab progression		

Table 3.29: User Agent String Table.

Table Name	User Agent String			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
agent_id	INT	Unique ID for each new user agent string	PK	
user_agent	TEXT	User-agent string		
is_custom	BOOLEAN	True if it is a custom user agent string provided by the user		
is_selected	BOOLEAN	True if it is selected to use as a user agent string.		

Table 3.30: Session Table

Table Name	Session			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
session_id	INT	Unique ID for each session	PK	
timestamp	DATETIME	Timestamp of the session		
token_key	TEXT	Key (e.g. *PHPSESSID*)		
token_value	TEXT	Key's value		

Table 3.31: Log Table.

Table Name	Log			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
log_id	INT	Unique ID for each new log activity	PK	
level	TEXT	Log level		
description	TEXT	Activity details		

Table 3.32: Scope Table.

Table Name	Scope			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
scope_id	INT	Unique ID for each new scope	PK	
pattern_string	TEXT	Log level		
is_excluded	BOOLEAN	Check if it is an in or out of scope pattern string		
is_enabled	BOOLEAN	Does the pattern used for validation		

Table 3.33: Tool Table.

Table Name	Tool			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
tool_id	INT	Unique ID for each tool	PK	
classs_name	TEXT	Unique name for system	Unique	
name	TEXT	Name of the tool		
type	TEXT	Type of tool	R or X	
description	TEXT	Information about the tool	Nullable	
origin	TEXT	Link to project		
api_key	TEXT	API key	Nullable	
support_cli	BOOLEAN	Does the tool support CLI		
is_enabled	BOOLEAN	True if the tool is allowed to be used for the hunt		
is_selected	BOOLEAN	True if the tool is chosen to involve the test later		
cwe_id	TEXT	Relative CWE	Nullable	

Table 3.34: Command Table.

Table Name	Command			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
cmd_id	INT	Unique ID for each tool	PK	
script	TEXT	Name of the tool		
description	TEXT	Script explanation/note	Nullable	
is_custom	BOOLEAN	Type of tool • **r**: recon tool • **x**: exploitation tool		
is_selected	BOOLEAN	True if selected for execution		
tool_id	INT	Tool ID	FK	Tool

Table 3.35: Execution Entry Table.

Table Name	Execution Entry			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_entry_id	INT	Unique ID for each tool	PK	
executed_at	DATETIME	Command execution timestamp		
tool_id	INT	Tool ID	FK	Tool
cmd_id (null)	INT	Command ID	FK	Command

Table 3.36: Execution Log Table.

Table Name	Execution Log			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_id	INT	Unique ID for each tool	PK	
saved_at	DATETIME	Command execution timestamp		
cmd_built	INT	Complete script that got executed	Nullable	
result_type	TEXT	To determine which table to map later	Host/Server /Port/URL	
console_output	TEXT	The console output		
exe_entry_id	INT	Execution Entry ID	FK	Execution Entry

Table 3.37: Host Table.

Table Name	Host			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
id	INT	Unique ID for each tool	PK	
hostname	TEXT	Domain or subdomain	Unique	
is_domain	BOOLEAN	Is a domain or subdomain		
http_status	INT	The host HTTP status (0 = unreachable)		
is_known	BOOLEAN	It is a known hostname keyed in by the user		
is_targeted	BOOLEAN	True if the domain or subdomain is chosen to be tested		

Table 3.38: Server Table.

Table Name	Server			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
server_id	INT	Host ID	PK, FK1	Host
host_id	INT	Tool ID	PK, FK2	Tool
ip	TEXT	Server IP Address		
is_targeted	BOOLEAN	True if is chosen to be tested		

Table 3.39: Port Table.

Table Name	Port			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
port_id	INT	Unique ID for each port	PK	
number	INT	The port number	Unique	
service_name	TEXT	Port service name		
is_open	BOOLEAN	True if the port is open		
is_targeted	BOOLEAN	True if is chosen to be tested		
server_id	INT	Server ID	FK	Server

Table 3.40: URL Table.

Table Name	Url			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
id	INT	Unique ID for each URL	PK	
url	TEXT	Uniform resource locator	Unique	
http_method	TEXT	URL HTTP request method		
http_status	INT	URL HTTP status		
has_param	BOOLEAN	True if URL consist of parameters/body		
raw_request	TEXT	URL raw HTTP request		
raw_response	TEXT	URL raw HTTP response		
is_targeted	BOOLEAN	True if is chosen to be tested		
host_id	INT	Host ID	FK	Host

Table 3.41: Host Recon Table.

Table Name	Host Recon			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_id	INT	Execution Entry ID	PK, FK1	Execution Entry
host_id	INT	Host ID	PK, FK2	Host

Table 3.42: Server Recon Table.

Table Name	Server Recon			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_id	INT	Execution Entry ID	PK, FK1	Execution Entry
server_id	INT	Server ID	PK, FK2	Server

Table 3.43: Port Recon Table.

Table Name	Port Recon			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_id	INT	Execution Entry ID	PK, FK1	Execution Entry
port_id	INT	Port ID	PK, FK2	Port

Table 3.44: Url Recon Table.

Table Name	Url Recon			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
exe_id	INT	Execution Entry ID	PK, FK1	Execution Entry
url_id	INT	Url ID	PK, FK2	Url

Table 3.45: Url Exploit Table.

Table Name	Url Exploit			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
vuln_id	INT	Vulnerability ID	PK, FK1	Vulnerability
url_id	INT	Url ID	PK, FK2	Url

Table 3.46: Vulnerability Table

Table Name	Session			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
vuln_id	INT	Vulnerability ID	PK	
risk_score	INT	0 to 5		
risk_label	TEXT	CVSS V4.0 Severity		
risk_details	TEXT	Proof-of-concept		
risk_explanation	TEXT	Vulnerability explanation		
exe_id	INT	Execution Log ID	FK	Execution Log

Table 3.47: Host Exploit Table.

Table Name	Host Exploit			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
vuln_id	INT	Vulnerability ID	PK, FK1	Vulnerability
host_id	INT	Host ID	PK, FK2	Host

Table 3.48: Server Exploit Table.

Table Name	Server Exploit			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
vuln_id	INT	Vulnerability ID	PK, FK1	Vulnerability
server_id	INT	Server ID	PK, FK2	Server

Table 3.49: Port Exploit Table.

Table Name	Port Exploit			
Field Name	**Data Type**	**Description**	**Constraint**	**Reference Table**
vuln_id	INT	Vulnerability ID	PK, FK1	Vulnerability
port_id	INT	Port ID	PK, FK2	Port

3.6.2 Activity Diagram

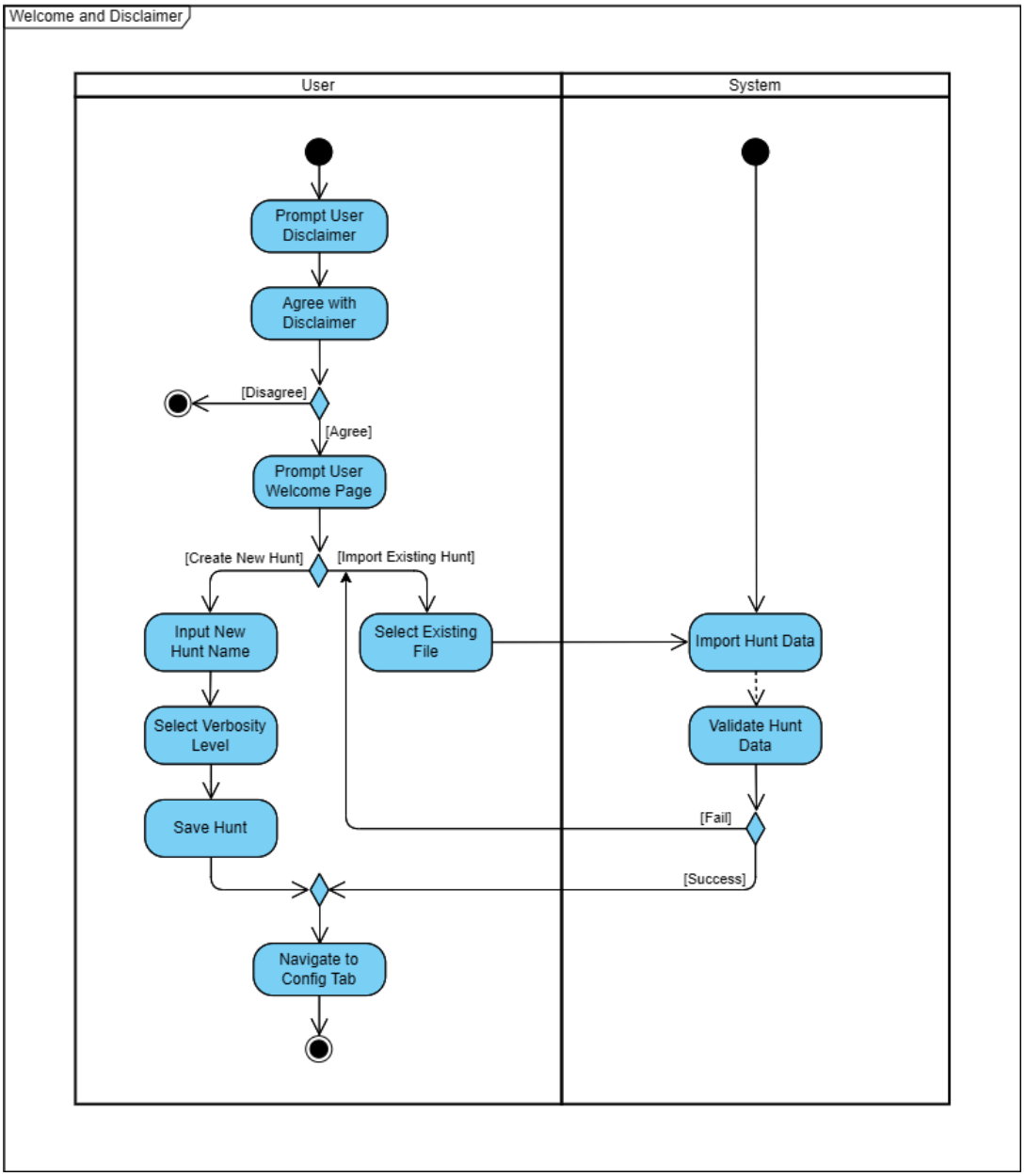


Figure 3.8: Welcome and Disclaimer Activities.

Figure 3.8 illustrates the initial workflow of the WABUHA system, starting with a welcome page that presents a disclaimer to the user. Upon acknowledging and agreeing to the disclaimer, the user is redirected to the main welcome page. Here, they are either to create a new hunt or import an existing one. If a new hunt is chosen, the user is prompted to input a hunt name and select a verbosity level. Then, the system saves the hunt and navigates the user

to the “Config” tab. Alternatively, if the user chooses to import an existing hunt, they are required to select the corresponding file. The system then proceeds to validate the imported hunt data. If the validation is successful, the user is redirected to the “Config” tab, else, an error message is displayed.

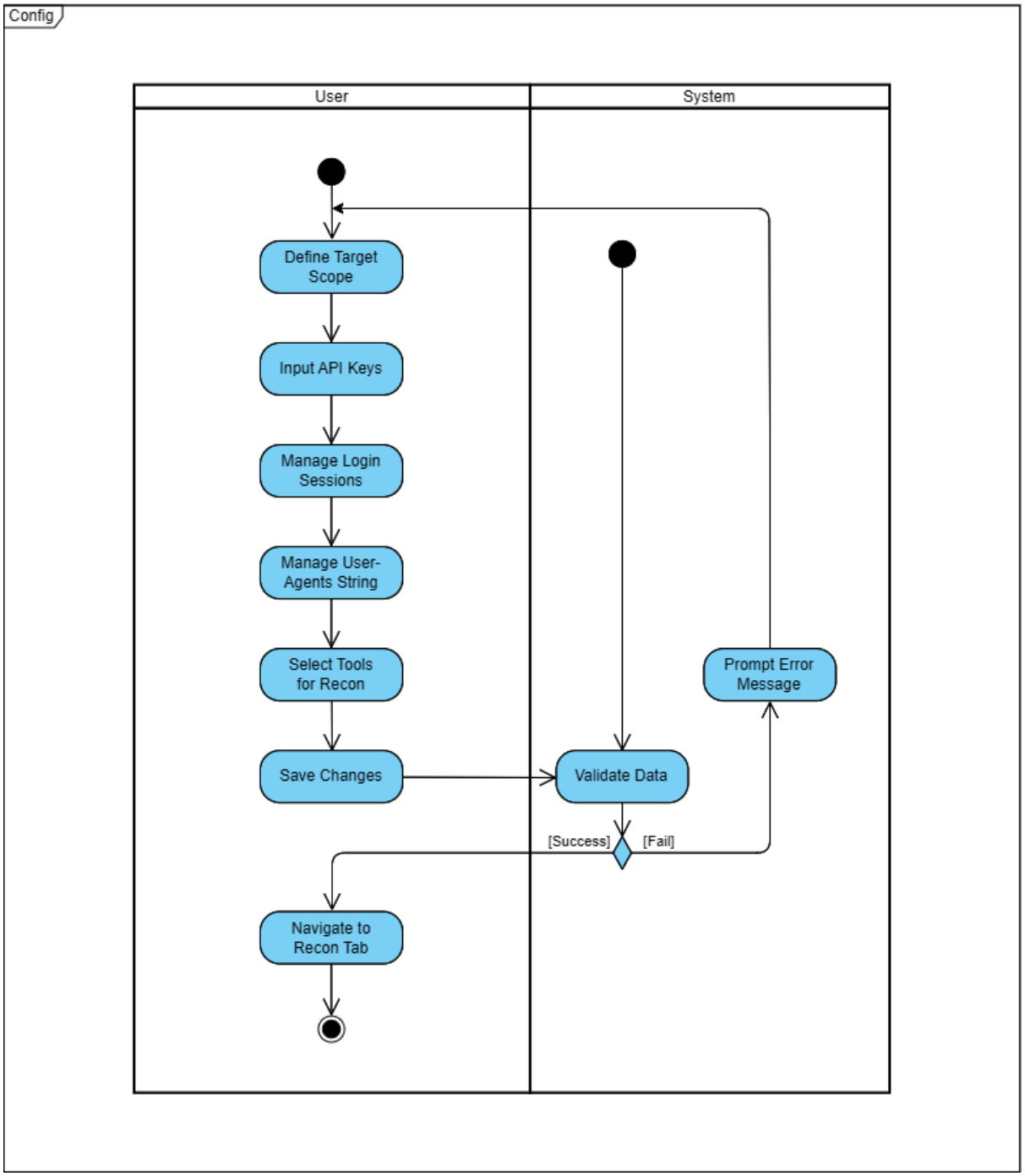


Figure 3.9: Configuration Activities.

Figure 3.9 illustrates the workflow within the "Config" tab. The user begins to specify the target scope and provide API keys. Next, they control the web application logins and user-agent strings. The user makes a selection of the tools used in the subsequent recon stage. After setting these configurations, the system checks the input data. If the validation is successful, the user is directed to the "Recon" tab, otherwise an error message is presented.

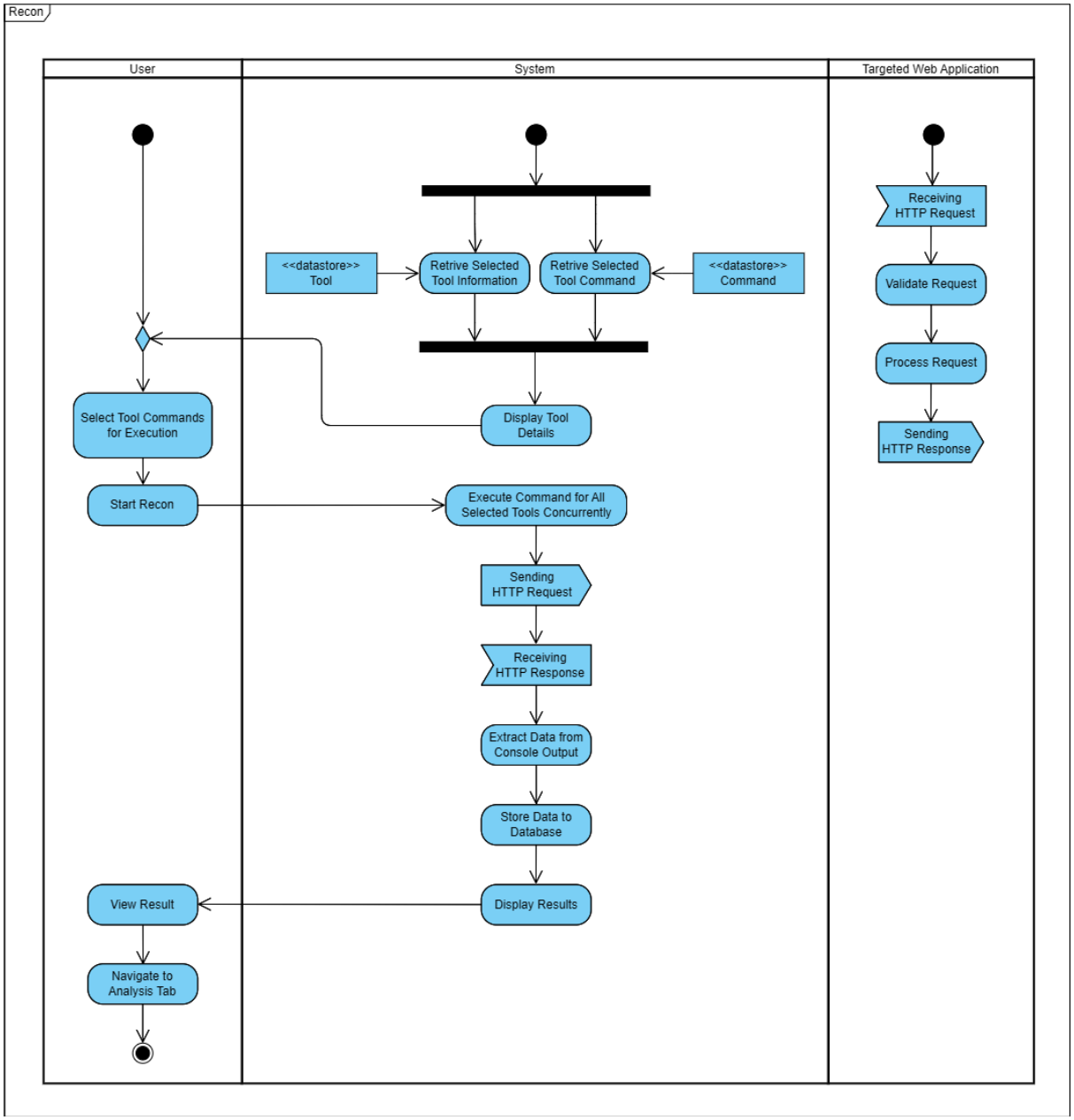


Figure 3.10: Recon Activities.

Figure 3.10 illustrates the workflow within the "Recon" tab. The user first selects the tool commands to be executed and begins the reconnaissance procedure. At the same time, the system accesses and gathers chosen tool information and commands from the datastore. Then the system provides with the selected commands executing the commands, processes the incoming HTTP requests and responses. Then the system extracts the needed information from the console output. This extracted information is saved to the database and is made available to the user. At last, the user is allowed to access the "Analysis" tab to make a deeper analysis of the results.

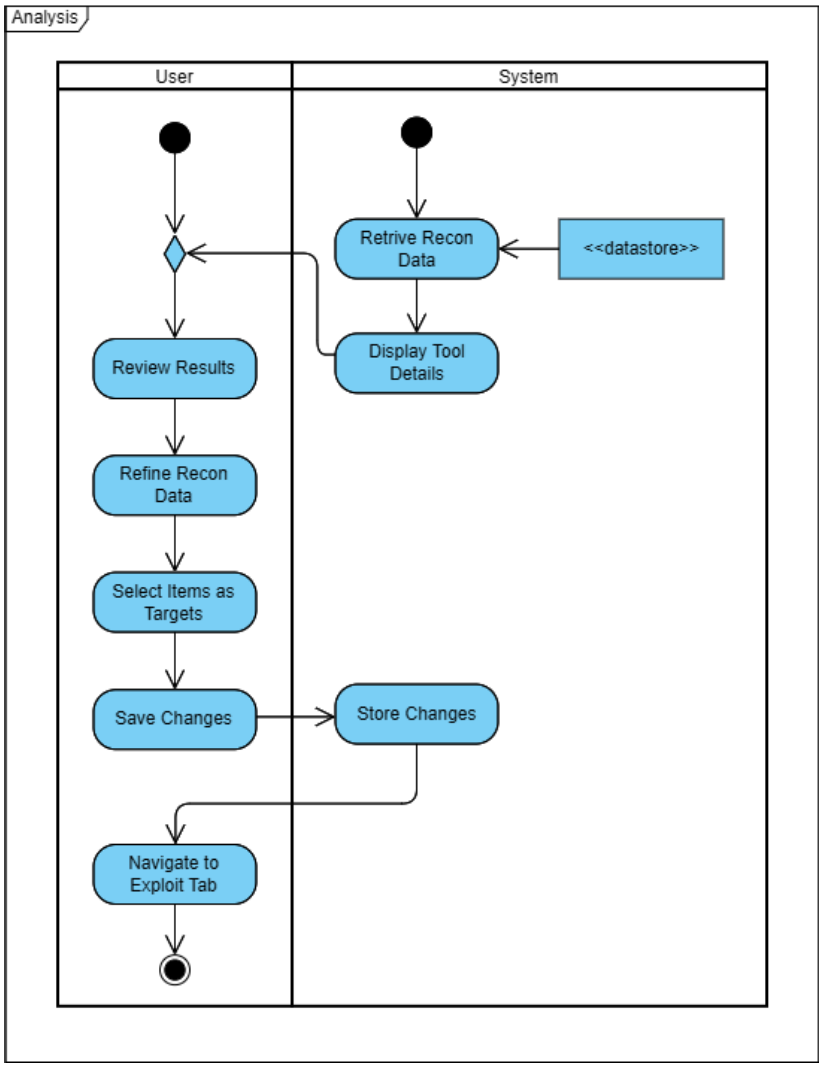


Figure 3.11: Analysis Activities

Figure 3.11 illustrates the workflow within the "Analysis" tab. The user first checks the reconnaissance data, as obtained from the datastore. The tool information is presented in the system to help the user's analysis. The user can then upscale the reconnaissance data and pick certain objects as targets. At last, the changes are saved by the user, which are stored in the system. The user can then go to the "Exploit" tab where further actions can be taken.

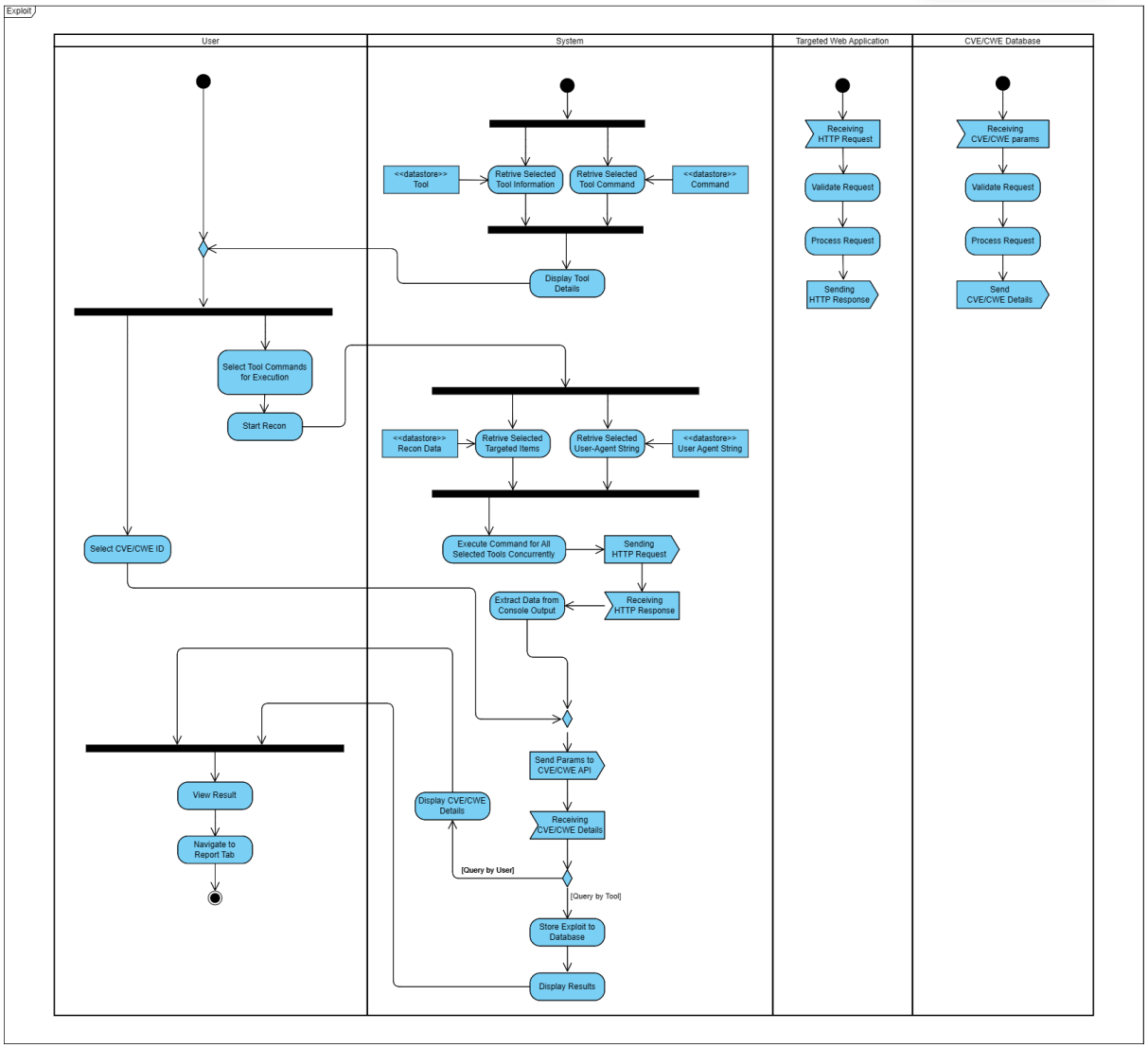


Figure 3.12: Exploit Activities.

Figure 3.12 illustrates the workflow within the "Exploit" tab. User can execute the exploitation process, the system obtains the selected tool details and commands. The system then implements the desired commands and the target web page processes the resulting HTTP

requests and HTTP responses. Users can also select a CVE/CWE ID for viewing. The system reads information from the console output and feeds parameters to the CVE/CWE API. When the CVE/CWE information is received, the system stores the exploit into the database and shows the output to the user. At last, the report of "Report" can be accessed by the user for further exploration.

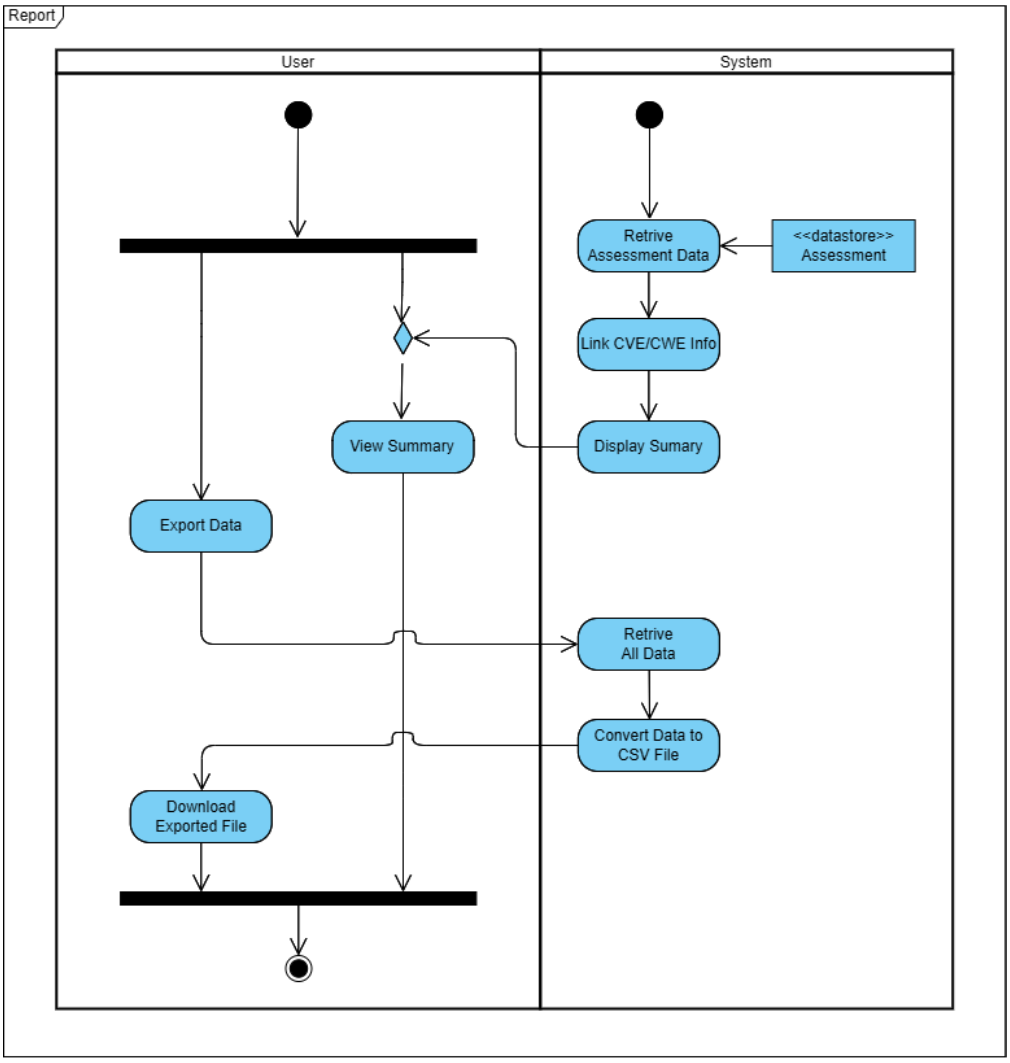


Figure 3.13: Report Activities.

Finally, Figure 3.13 illustrates the workflow within the "Report" tab. The user can either view a summary of the assessment or export the data. If the user chooses to export the data, the system retrieves all assessment data from the datastore, converts it into a CSV file, and allows the user to download the file. If the user chooses to view the summary, the system retrieves the

assessment data, links CVE/CWE information, and displays a summary of the assessment to the user.

3.6.3 User Interface Mockups

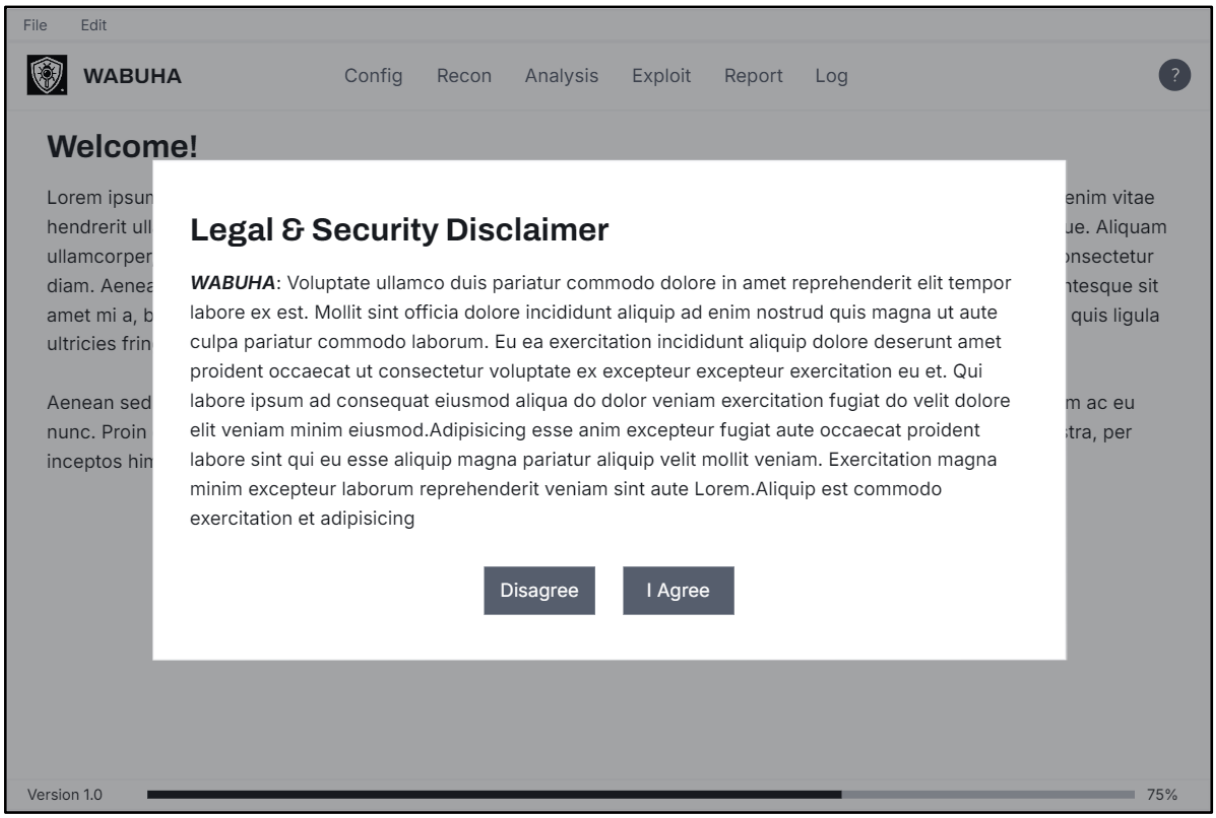


Figure 3.14: Legal & Security Disclaimer Popup.

When launching the WABUHA system, the user is shown a Legal and Security Disclaimer stating the use terms and conditions, shown in Figure 3.14. It provides such users with information on how data is to be handled and kept private according to the privacy policy. By accepting the user agreements, users agree to be bound by the conditions. Individuals who disagree have the choice to leave by clicking "Disagree. Refer to the full Legal and Security Disclaimer in *Appendix B*.

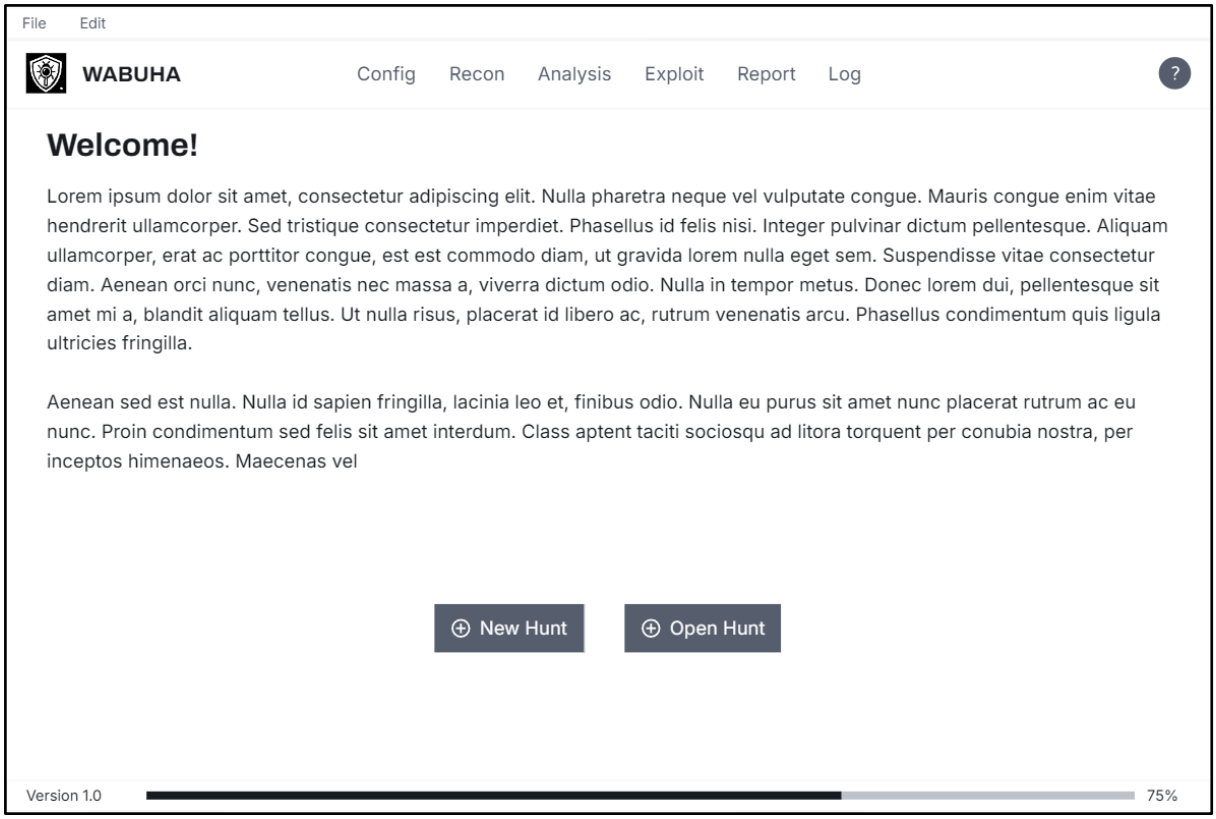


Figure 3.15: Welcome Page.

After accepting disclaimers, the "Welcome Page" (shown in Figure 3.15). The header contains main navigation options like Config, Recon, Analysis, Exploit, Report, and Log, which could be used to quickly gain access to various parts of the system. Below the header, a prominent "Welcome!" message is displayed. At the end of this text, there are two buttons with the labels "New Hunt" and "Open Hunt," allowing users to start a new operation or refine an ongoing operation.

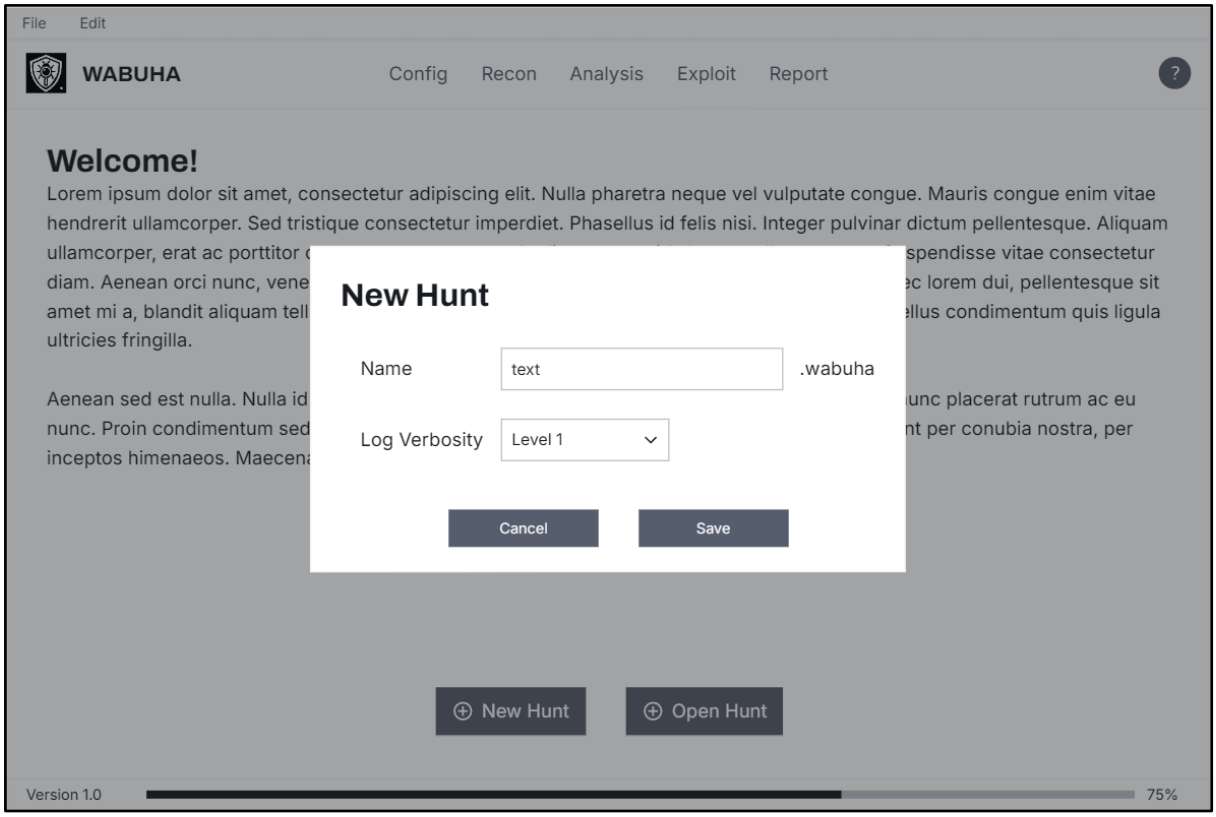


Figure 3.16: New Hunt Popup.

Upon clicking the "New Hunt" button on the Welcome Page of the WABUHA interface, a modal window titled "New Hunt" appears shown in Figure 3.16. This modal window allows the user to input the name of the hunt and select the session log verbosity level from a dropdown menu. The interface is designed with fields for required information and two buttons at the bottom labeled "Cancel" and "Save".

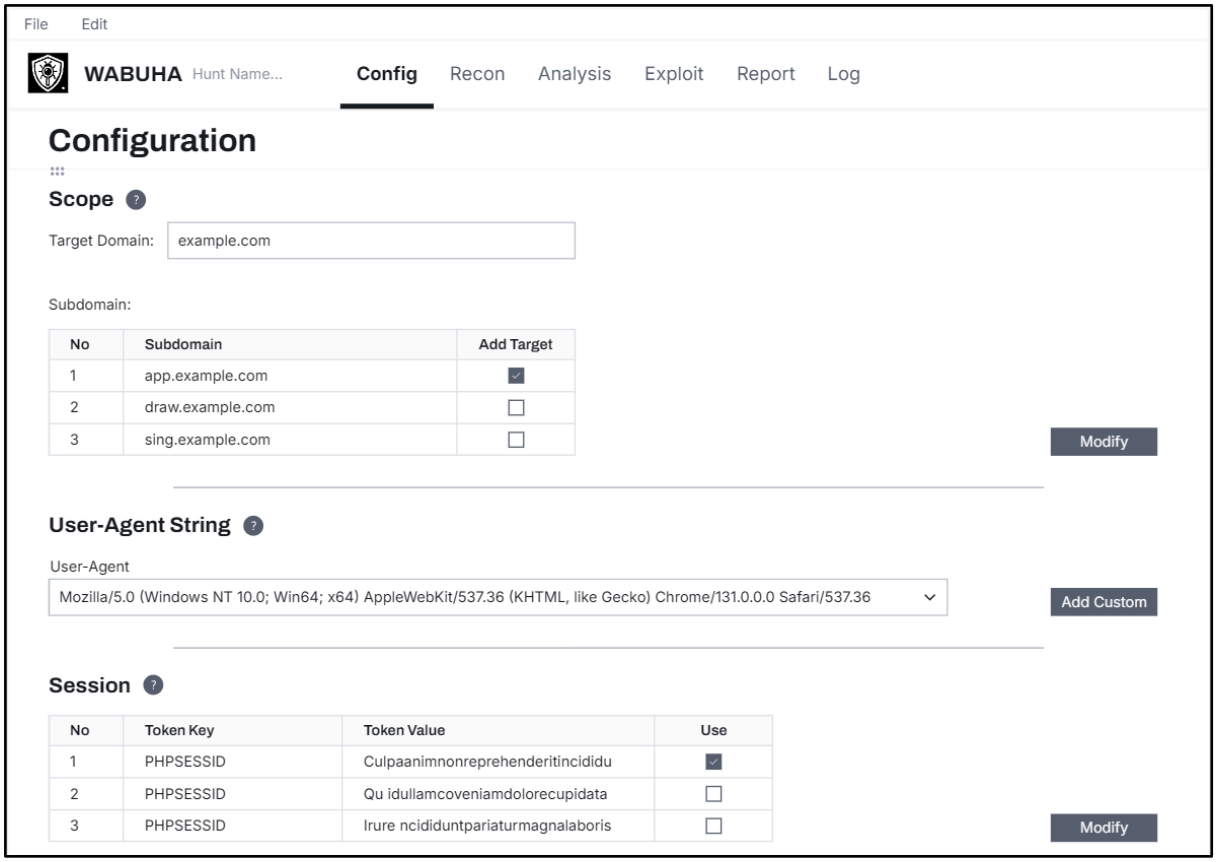


Figure 3.17: Config Tab (Part 1).

Upon saving the new hunt details, users are directed to the "Config" tab shown in Figure 3.17. This tab is structured to facilitate the configuration for the hunt. It is divided into Scope, User-Agent String, and Session, API keys and Tools Selection. In the Scope section, users define the target domain and control subdomains. The User-Agent String section enables customisation of the string with both a default option and user option. Then the Session part is used for controlling session tokens, in the Session section there is the option of setting the token keys and values.

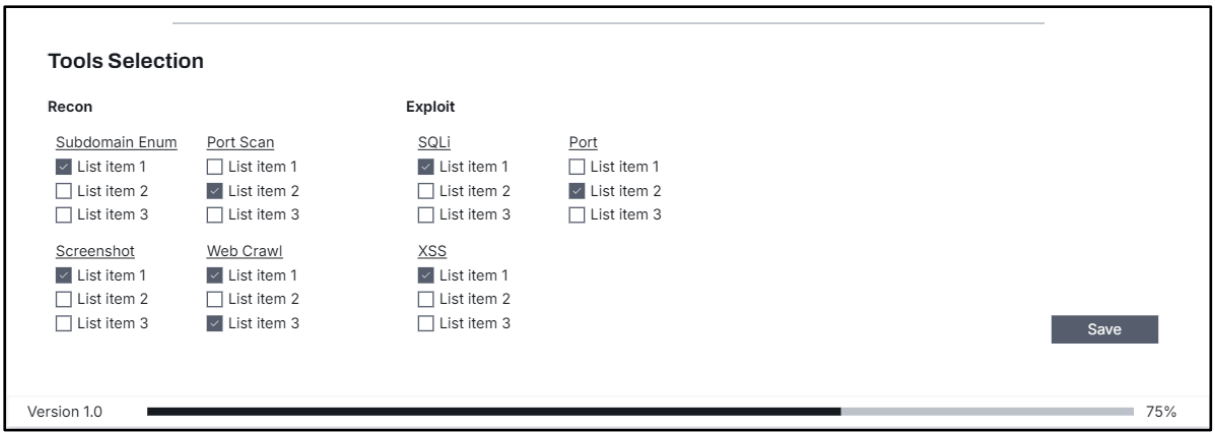


Figure 3.18: Config Tab (Part 2).

Then, users proceed to the Tools Selection shown in Figure 3.18. The Tools Selection section is divided into Reconnaissance and Exploitation categories. Users can select the tools they plan to use in each phase by checking the corresponding boxes (at least one tool per category). This page also includes a Save button for each section to apply the configurations.

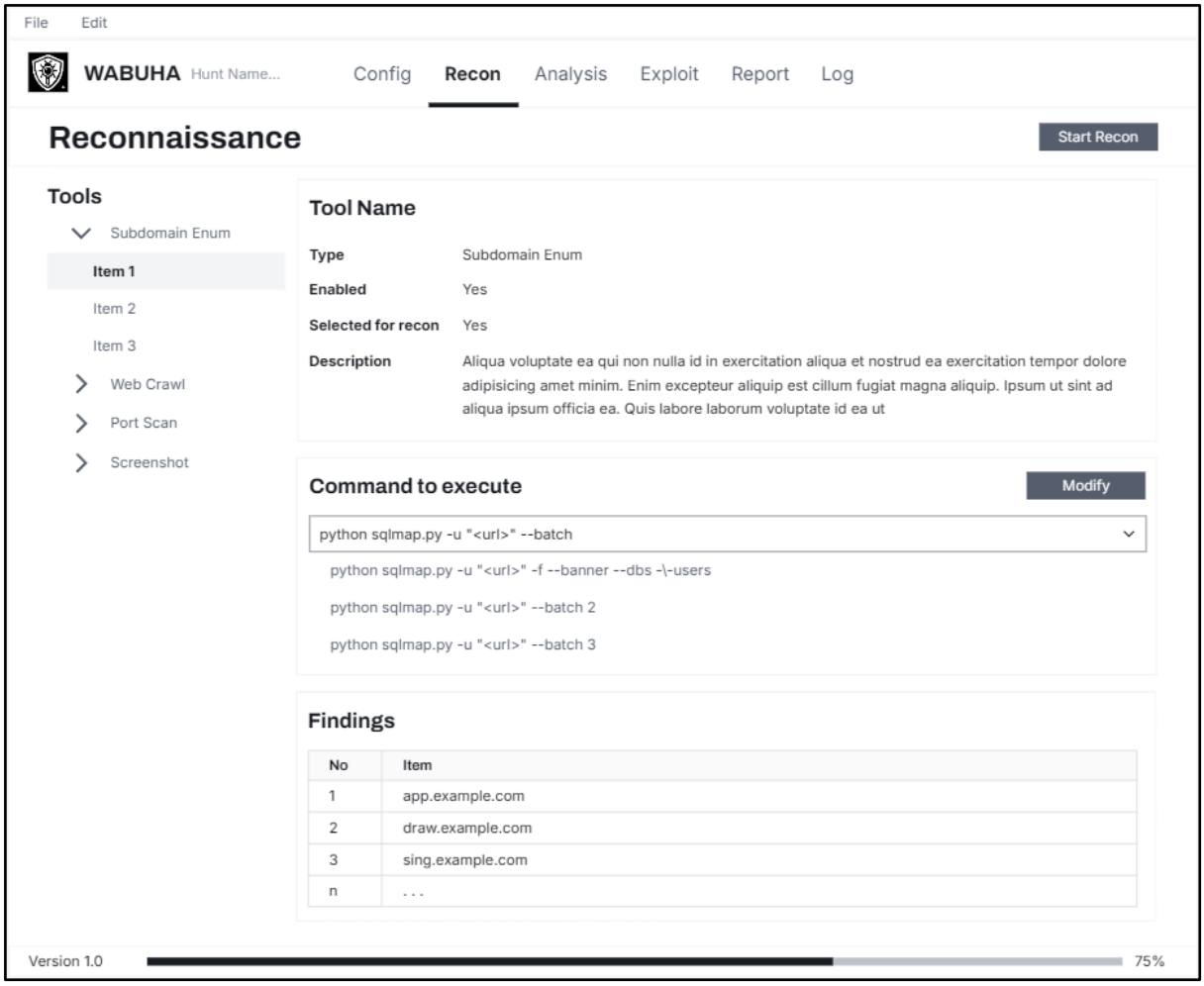


Figure 3.19: Recon Tab.

After defining the scope and completing the system configuration, users are directed to the "Recon" tab shown in Figure 3.19. This tab is organised to facilitate the reconnaissance phase of the workflow. It consists of Tools, Tool Name, Command to Execute, and Findings sections. In the Tools section, a list of the available reconnaissance tools can be chosen by the users. In the Command to execute section, users can change and choose other commands from the menu provided, which includes a dropdown list of different commands to execute. In the end, the Findings section presents the reconnaissance outcome by listing discovered subdomains, IPs, ports, and URLs. This design ensures that the users can effectively organise and carry out reconnaissance tasks, an essential step to discover possible flaws in the target system.

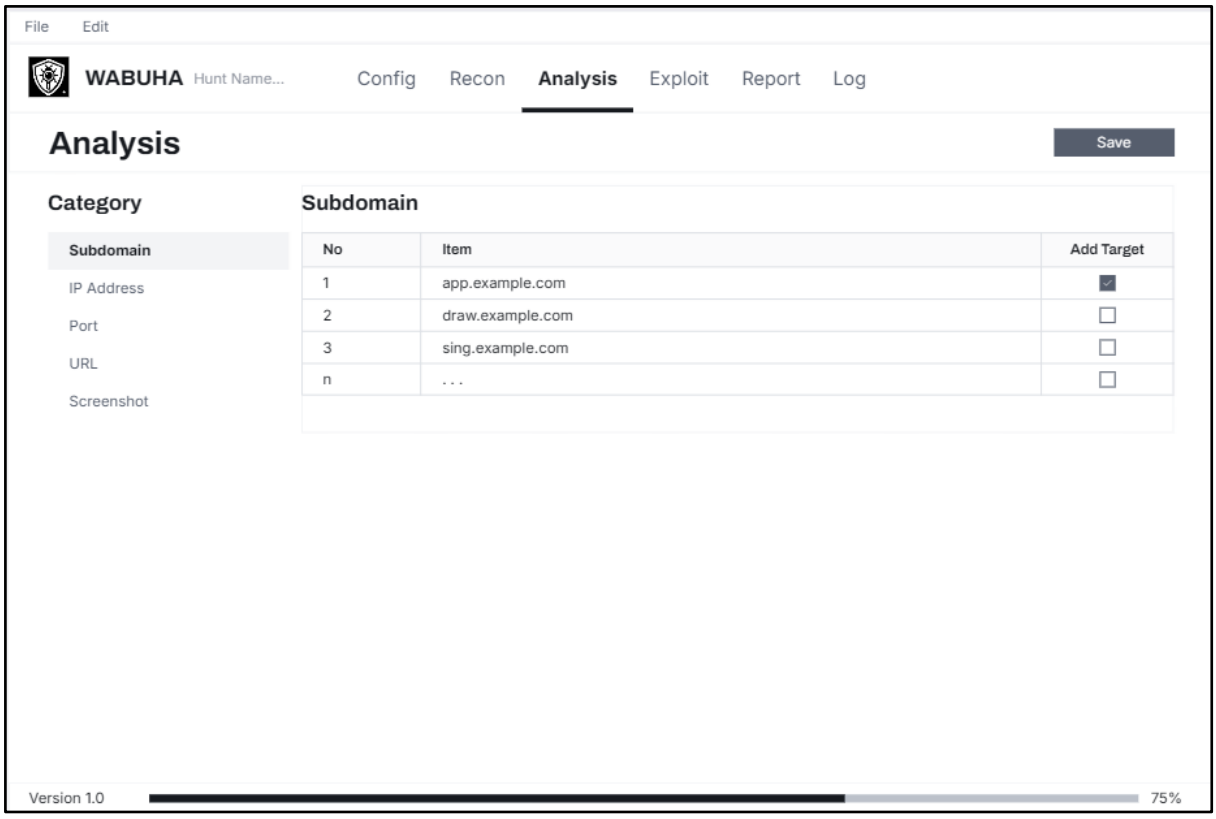


Figure 3.20: Analysis Tab.

Once done with reconnaissance, the user can navigate to the "Analysis" tab as presented in Figure 3.20. It is further segmented into several classes such as Subdomain, IP Address, Port, URL, and Screenshot. In this example, the Subdomain category is selected. The page shows a table with the subdomains scanned during the reconnaissance phase and the user can check the items to be used as targets for exploitation in a later stage. There is a "Save" button to save the user's selection. This section is of high relevance for users to be able to analyse and choose particular subdomains for exploitation.

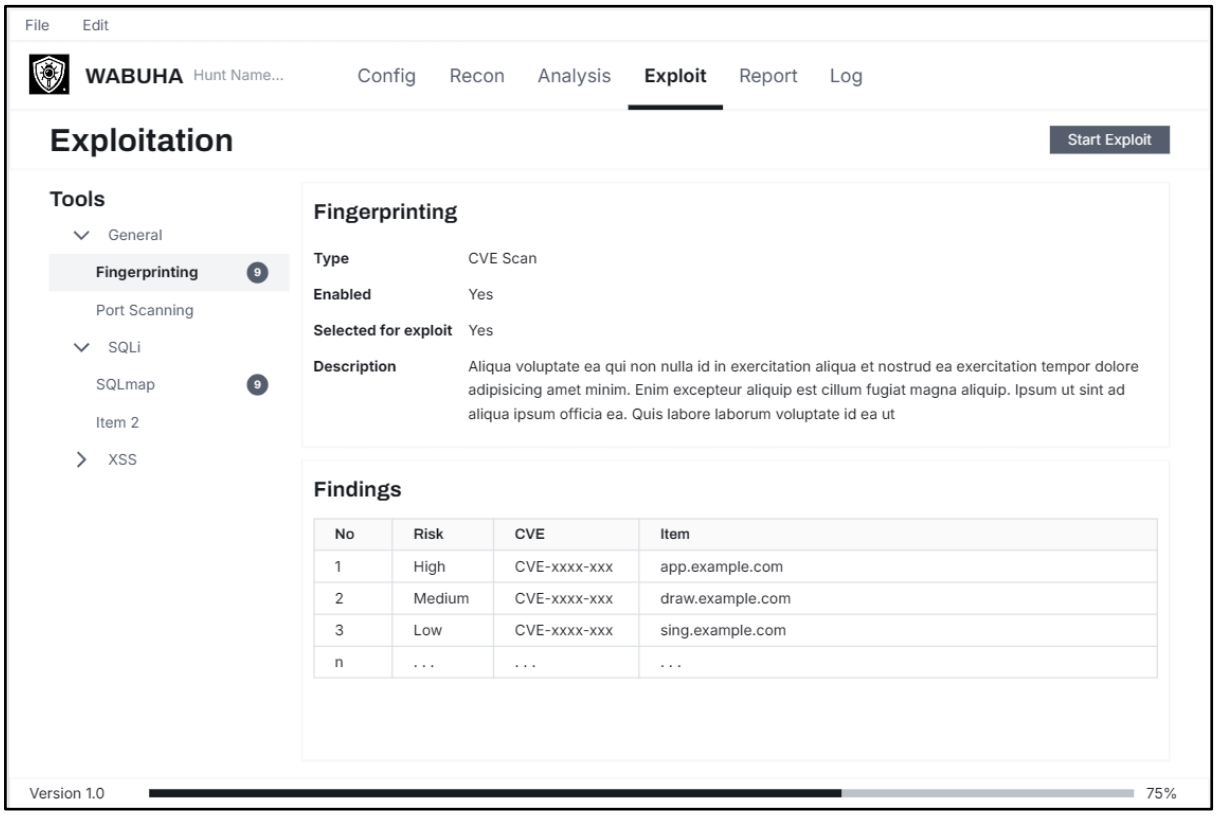


Figure 3.21: Exploit Tab (Part 1).

Once the Analysis phase is finished and targets for exploitation are included as objects on the "Exploit" tab (see Figure 3.21). On the left, a list of tools (Fingerprinting, Port Scanning, SQLi, and XSS) are provided in a navigation pane. The selected tool is shown in the main part on the right side with sufficient detailed data. For instance, if the Fingerprinting tool is chosen, information is revealed about its type (CVE Scan), its state (enabled), and whether it is on the list of tools that may be used for exploiting, along with a placeholder text area to which a description for the tool may be added. Furthermore, there is a "Start Exploit" button available to trigger the exploitation attack. Following the exploitation, a table "Findings" presents known vulnerabilities, annotated by risk level (High, Medium, Low), CVE identifier and content item.

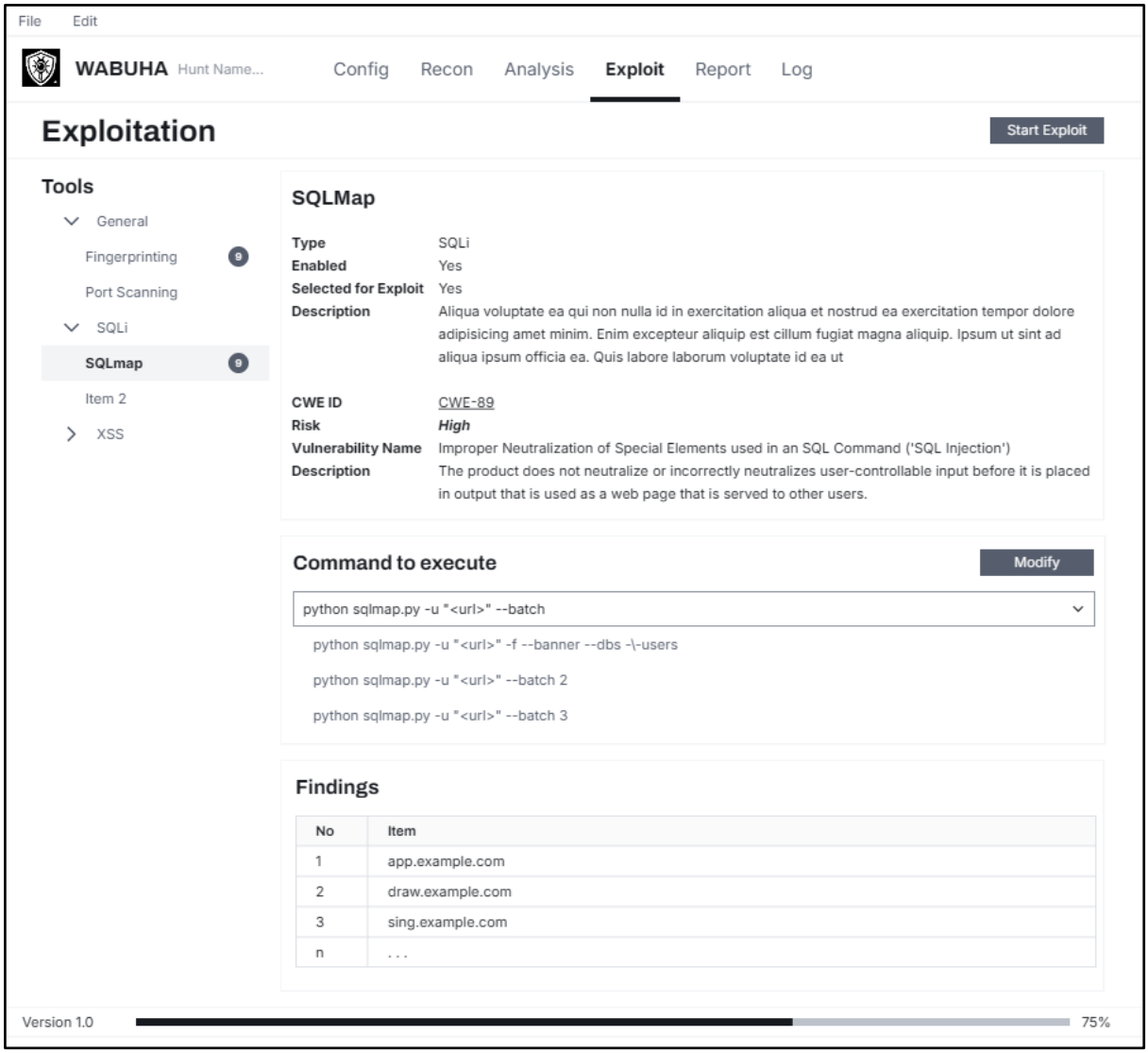


Figure 3.22: Exploit Tab (Part 2).

The users, who select the SQLmap tool from the side list under the "Exploit" tab as shown in Figure 3.22, are then directed to a specific interface to carry out the SQL injection (SQLi) exploitation details. In the body of the page, a detailed description of the tool is shown. Furthermore, the interface outputs the CWE ID (CWE-89) of SQL injection attacks, which specifies the type of the vulnerability. Below this, a fixed set of commands from which the user can choose are shown, options to execute desired commands. If the user did not choose any command, the default command will be used. A “Findings” table with identified vulnerabilities is presented in the "Findings" section.

Figure 3.23: Exploit Tab (Part 3).

If the user chooses the CWE ID of the tool in the previous figure, the background system will redirect user to the selected CWE website shown in Figure 3.23. Such information is presented in the popup interface, which shows users the complete information about the detected vulnerability. For instance, selecting CWE-89, relevant to SQL Injection, provides the user with a comprehensive explanatory text of the vulnerability.

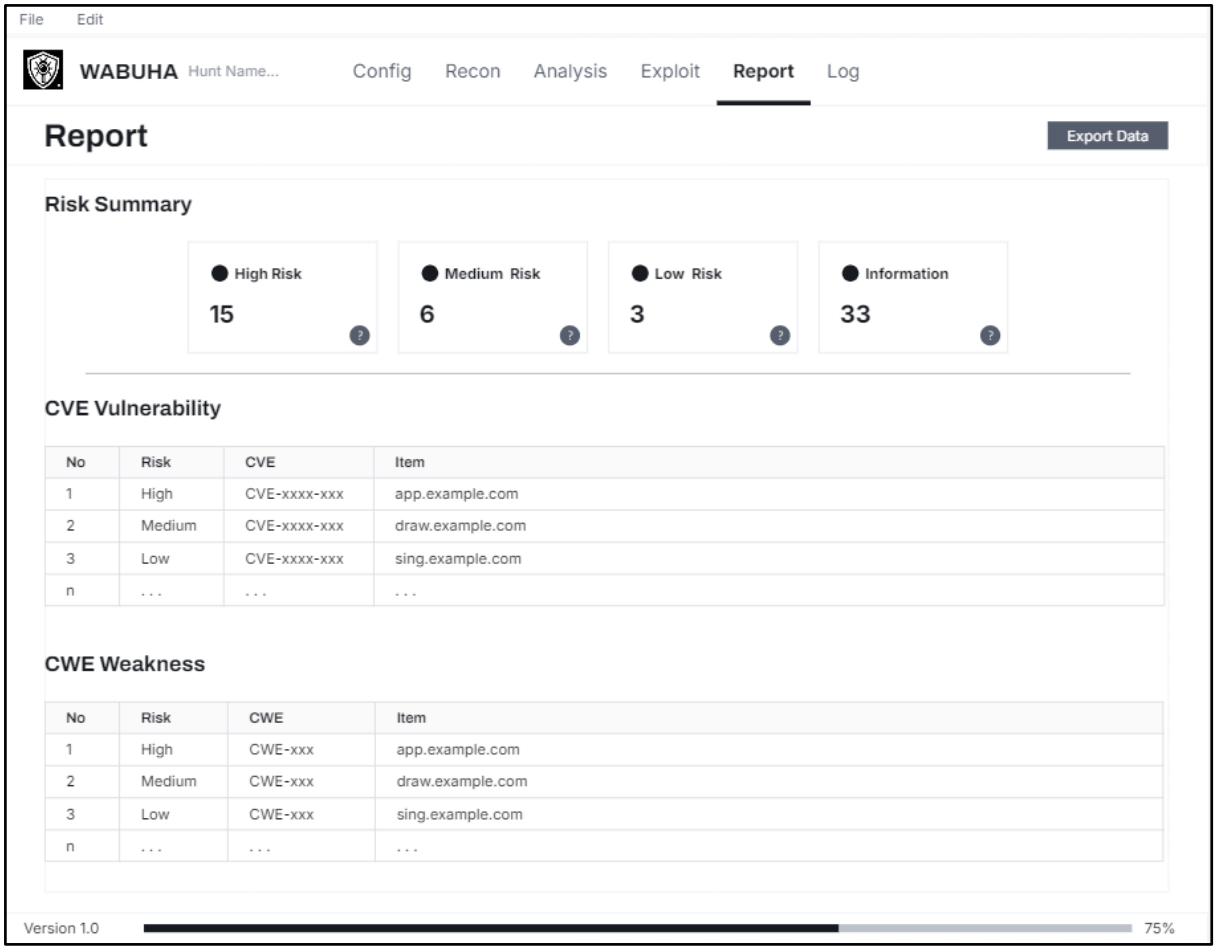


Figure 3.24: Report Tab.

Once the exploitation has been finished, users access the menu "Report" as displayed in Figure 3.24. In this section a formalised description has been given of the reported weaknesses. The upper part, labelled with "Risk Summary", provides a brief state of the risk levels, which are refined in 'High', 'Medium', 'Low' and 'Information' categories with their count values. These tables present each vulnerability or defect with its risk level and related item. Using these rich results, users can read them and mine information to figure out the security state of the client system they are interested in.

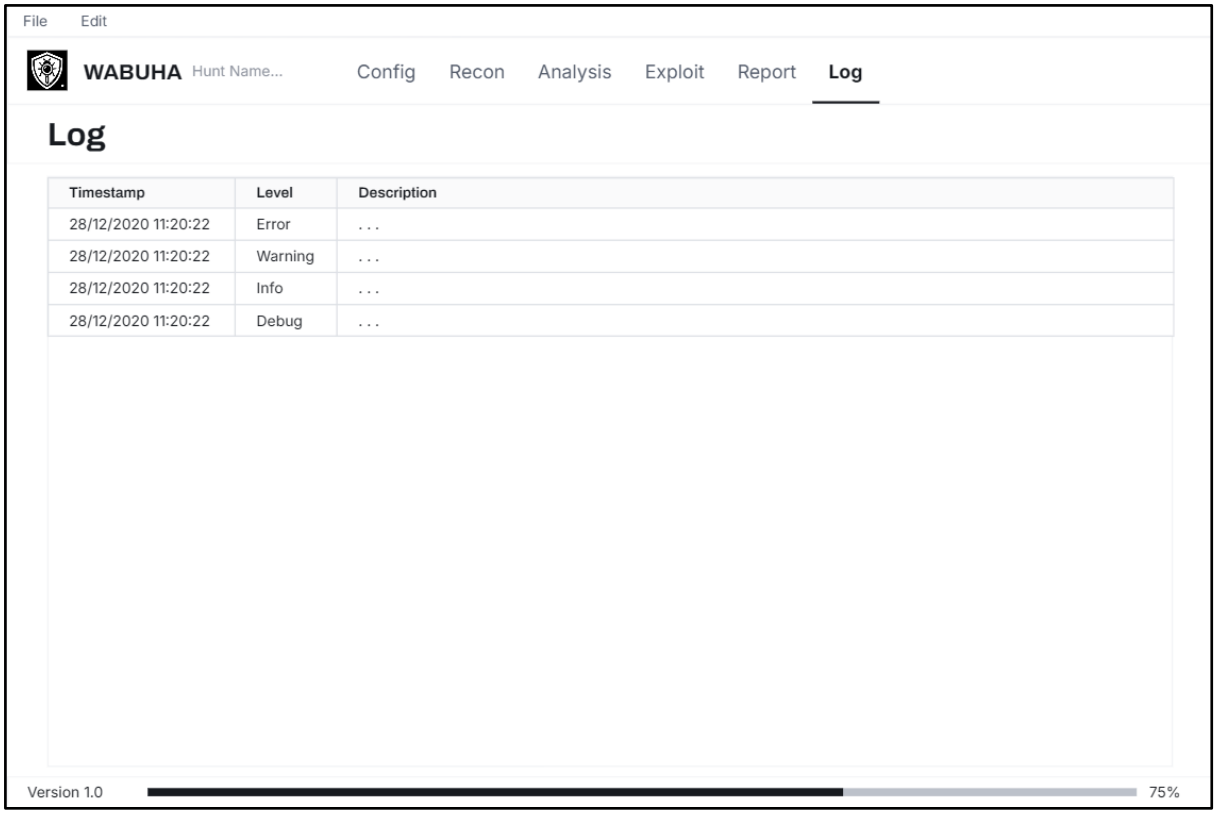


Figure 3.25: Log Tab.

Once the exploit findings have been managed within the Report tab, users go to the "Log" tab shown in Figure 3.25. This tab contains a detailed list of all application activities in the format of a table. The table has columns for Timestamp, Level, and Description. Log entries are classified according to level (Error, Warning, Info, Debug). This functionality is very important for monitoring and debugging, because it provides the possibility to look back into the complete history of actions.

3.7 Summary

Chapter 3 outlined the requirements analysis and design for WABUHA. The Agile Scrum methodology was adopted to enable iterative development. A simplified PTES framework is proposed for WABUHA, covering the core penetration testing phases which are Scope Definition, Reconnaissance (Recon), Vulnerability Management (Analysis), Exploitation, and Reporting.

Requirements were collected from questionnaires, with a focus on ease of use of a GUI, the presence of guided workflow, database integration, and possibilities of customisation. Technical specifications, such as hardware, software, and tools were described to facilitate efficient development.

The chapter presented the system design through use case diagrams, data flow diagrams, database schemas, activity diagrams, and interface mockups. This is to ensure modularity, scalability, and user-centric functionality. It also ensures that WABUHA achieves its aims of accessibility, ethical use, and effective novice penetration testing.

Chapter 4: Implementation

4.1 Introduction

This chapter discusses the implementation process of the WABUHA. It outlines the development environment, supported platforms, and the integration of third-party libraries and tools that enable reconnaissance and exploitation functionalities. Each component is discussed in detail to provide comprehension into how they contribute to the system functionality.

4.2 System Implementation

The implementation of WABUHA was carried out using Visual Studio Code (VSCode) as the Integrated Development Environment (IDE), chosen for its extensibility, built-in terminal support, and compatibility with Python development workflows (Microsoft, 2024).

The development of WABUHA involved the integration of several third-party Python libraries as well as open-source tools for reconnaissance and exploitation. These components were selected based on their community trust, stability, and functionality relevant to beginner penetration testers.

4.2.1 Platform Compatibility and OS Dependencies

WABUHA is currently designed to run and tested exclusively on **Kali Linux** (Debian-based Linux distribution), leveraging its pre-configured penetration testing environment and support for wide variety of security tools (Kali Linux, 2024). Subprocess calls assume a Unix-like shell and filesystem structure, are how the system communicates with these tools. Since certain of the tools are inaccessible or function differently in a Windows environment, WABUHA is now incompatible with Windows OS due to this design.

4.2.2 System Requirements

To ensure proper functionality and compatibility, WABUHA requires a system that meets the following minimum specifications in Table 4.1:

Table 4.1: System Minimum Requirements.

Operating System	Kali Linux 2023 or later, or any Debian-based distribution with similar capabilities.
Processor	At least 2 Cores.
Memory (RAM)	At least 4 GB (8 GB recommended).
Storage	Minimum 2 GB of free disk space for installation and data storage.
Python Version	Python 3.8 or higher.
Internet Connection	Required for API lookups (e.g. CWE mapping or AI supported explanation) and accessing certain reconnaissance tools.

Additionally, administrative (root) privileges may be necessary to run certain scripts that WABUHA integrates. The system should also have default Kali Linux tools pre-installed, or equivalent packages manually configured to ensure seamless subprocess execution.

4.2.3 Third-party Python Library

WABUHA employs several third-party Python libraries. The most significant libraries used are PyQt5, keyring, and mitmproxy. Each playing a crucial role in providing the graphical interface, credential management and traffic interception capabilities respectively.

4.2.3.1 PyQt5

PyQt5 is the core library used to implement WABUHA's graphical user interface (GUI). It provides a rich set of widgets and layout tools, enabling the creation of a clean, intuitive and user-friendly interface. Its signal-slot mechanism facilitates asynchronous operations, allowing

tool outputs to be captured and displayed dynamically without blocking the interface (Riverbank Computing Limited, 2025).

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1080, 720)
        MainWindow.setMinimumSize(QtCore.QSize(1080, 720))
        MainWindow.setWindowTitle("WABUHA")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setStyleSheet("#centralwidget{\n"
```

Figure 4.1: PyQt5 Main Window Code Snippet.

Figure 4.1 shows a snippet of the Python code demonstrating the use of the PyQt5 library. This segment illustrates the basic setup of the main window, including setting its object name, initial size, minimum size, and title to "WABUHA". The `QtWidgets.QWidget` class is utilized to create the central widget, which serves as the primary container for other UI elements within the main window. This foundational code highlights how PyQt5 classes like `QtCore` and `QtWidgets` are imported and used to build the application's visual interface.

4.2.3.2 keyring

According to jaraco (2024), Keyring is utilised to securely store and retrieve sensitive credentials, serving as an abstraction layer to the operating system's native secure credential storage facilities. By leveraging system-native keyring backends, it ensures that data such as recon, exploit tool, and AI model API keys remain protected during storage and retrieval, thereby enhancing the security posture of the application.

```
api_key = keyring.get_password(service_name, username)

keyring.set_password(service_name, username, api_key)

keyring.delete_password(service_name, username)
```

Figure 4.2: Keyring Code Snippet.

Figure 4.2 showcases code demonstrating the keyring library, used in WABUHA for secure credential management. It illustrates `keyring.get_password()` to retrieve, `keyring.set_password()` to store, and `keyring.delete_password()` to remove sensitive data like API keys. This ensures secure storage by leveraging the operating system's keyring services.

4.2.3.3 mitmproxy

Mitmproxy is integrated into WABUHA to support the analysis of HTTP traffic during the reconnaissance (Recon) stage. It allows the user to inspect and intercept HTTP requests and responses, providing valuable insights into how the target application behaves during various attack vectors (mitmproxy, 2025). This will also ease the user when setting cookies and recycle them for later use. Additionally, user can inspect the HTTP POST body data on-the-fly.

```
async def _start_server(self):
    try:
        options = Options(
            listen_host=self.listen_host,
            listen_port=self.listen_port,
            ssl_insecure=True,
        )

        self._master = DumpMaster(options, with_termlog=False, with_dumper=False)
        self._master.addons.add(
            InterceptorAddon(
                result_callback=self._on_result, error_callback=self._on_error
            )
        )
        self._other_master_options(self._master)

        self._on_start()
        await self._master.run()

    except Exception as e:
        msg = f"Error in mitmproxy thread: {e}\n\n{traceback.format_exc()}"
        self._on_error(msg)
```

Figure 4.3: Mitmproxy Code Snippet.

Figure 4.3 presents a code snippet showcasing the integration of Mitmproxy within the WABUHA application. This asynchronous function initialises and configures a Mitmproxy instance to act as a programmable proxy. It sets up `Options` for the proxy, specifying the listening host and port, and enabling `ssl_insecure` for SSL interception. A `DumpMaster` instance is created, and a custom `InterceptorAddon` is added to its addons, allowing for the interception and processing of network traffic. This setup enables WABUHA to monitor and interact with network requests and responses, with defined callbacks to handle intercepted data or any exceptions during the proxy operation.

4.2.4 Database Integration

WABUHA integrates SQLite database as its local data storage solution. SQLite was chosen for its simplicity and seamless integration with Python, making it well-suited for desktop-based applications like WABUHA. The database is used to persistently store data generated throughout the bug hunting process, including reconnaissance results, exploitation findings, tool configurations, session logs, and user-defined parameters. Hence, this will enable users to revisit previous hunts and historical data, thereby enhancing its functionality as an educational and practical tool for beginner penetration testers. Figure 4.4 below is the code snippet of how WABUHA will establish connection to the SQLite database.

```
def _connect(self):
    """Establish a connection to the SQLite database."""
    try:
        conn = sqlite3.connect(self.db_path, check_same_thread=False)
        conn.execute("PRAGMA foreign_keys = ON")  # Enable foreign key support
        conn.execute("PRAGMA journal_mode = WAL")  # Enable Write-Ahead Logging
        conn.row_factory = sqlite3.Row
        return conn
    except Exception as e:
        raise Exception(f"Error connecting to database: {e}")
```

Figure 4.4: SQLite Database Implementation Code Snippet.

4.2.5 Integration of Common Reconnaissance Tools

Reconnaissance is a crucial phase in penetration testing, and WABUHA integrates several widely used tools to assist users in this process. These tools were selected for their popularity, open-source nature, and effectiveness in discovering application surfaces.

4.2.5.1 Subdomain Enumeration

- **Sublist3r**: Developed by aboul3la, it is a fast and simple subdomain enumeration tool that uses multiple public sources to gather subdomains of a given domain (Aboul-Ela, 2021).
- **Subfinder**: Created by ProjectDiscovery, it performs passive subdomain enumeration using a curated set of reliable online sources and APIs (projectdiscovery, 2023).

4.2.5.2 Reverse DNS and Port Scan

- **Nmap**: A well-established network scanning tool used to discover hosts and services on a network. In WABUHA, Nmap is used for both reverse DNS lookups and comprehensive port scanning, providing visibility into the attack surface of discovered IP addresses (Nmap, 2024).

4.2.5.3 Web Crawl / Spider

- **Katana**: Created by ProjectDiscovery, it is a web crawling and URL discovery tool. It is capable of extracting endpoints, JavaScript files, and other useful resources from a target web application, which can be useful for identifying hidden or unlinked pages (projectdiscovery, 2021).

4.2.6 Integration of Common Exploitation Tools

The exploitation phase in WABUHA enables users to simulate attacks and identify real vulnerabilities. The tools integrated under this phase assist in identifying critical issues such as SQL injection and Cross-Site Scripting (XSS). However, WABUHA does not implement destructive command template as the aim is to provide a safe learning environment for beginner penetration testers.

4.2.6.1 SQL Injection

- **SQLmap**: Developed by the sqlmapproject, it automates the process of detecting and exploiting SQL injection vulnerabilities in web applications. It supports various database management systems and offers features such as database fingerprinting and data extraction (sqlmapproject, 2019).
- **Nuclei**: Created by ProjectDiscovery, it is a customisable vulnerability scanner that uses payload templates to detect a wide range of known issues, including SQL injections. WABUHA leverages Nuclei templates during exploitation phases to validate findings (projectdiscovery, 2022).

4.2.6.2 Cross Site Scripting (XSS)

- **Dalfox**: Developed by hahwul, it is a fast and powerful XSS scanning tool. It performs both passive and active analysis to detect XSS vulnerabilities in web applications. Dalfox is integrated into WABUHA to allow users to simulate and detect reflected and stored XSS attacks efficiently (hahwul, 2023).

4.2.7 Gemini 2.0 Flash Model

```
def generate_content(self, prompt):
    headers = {"Content-Type": "application/json"}
    params = {"key": self.api_key}
    data = {"contents": [{"parts": [{"text": prompt}]}]}
    try:
        response = requests.post(
            self.API_URL, headers=headers, params=params, json=data, timeout=30
        )
        if response.status_code == 200:
            result = response.json()
            try:
                return result["candidates"][0]["content"]["parts"][0]["text"]
            except (KeyError, IndexError):
                return "No response from Gemini AI."
        elif response.status_code == 401:
            return "Unauthorized: Invalid API key. Please update your API key."
        elif response.status_code == 429:
            retry_after = response.headers.get("Retry-After")
            msg = "Rate limit exceeded. Please wait before making more requests."
            if retry_after:
                msg += f" Retry after {retry_after} seconds."
            return msg
        else:
            return f"Error: {response.status_code} - {response.text}"
    except requests.RequestException as e:
        return f"Network error: {e}"
```

Figure 4.5: Gemini 2.0 Flash Model Implementation Code Snippet.

Figure 4.5 presents the key code snippet detailing the utilisation of the Gemini 2.0 Flash Model API. This integration enables the dynamic generation of vulnerability explanations. A 30 second timeout is implemented for the API request to avoid potential indefinite waiting states, thereby enhancing user experience. To ensure the uniformity and accuracy of the generated explanations, WABUHA leverages the following predefined prompt:

```
Explain this web vulnerability for beginners in <{Word_Count} words,
using the format below. Start each section with a square-bracket header:

[Definition]
One-sentence, beginner-friendly explanation.

[Why {Risk_Level}]
Justify the risk level (e.g., High = easy + severe).

[Example]
Describe an attack using {PoC}.

[Impact]
State one worst-case result (e.g., Steal passwords).

[Mitigation]
Suggest one basic fix.

Input:
- Risk: {Risk_Level}
- Type: {Tool_Category}
- Found via: {Tool_Name}
- Command: {Cmd_Executed}
- PoC: {PoC}

Rules:
- Max 100 words
- No off-topic info
- No jargon without explanation
- Use analogies if and only if helpful
- Be clear, not technical
```

4.3 WABUHA Features

This section provides a thorough explanation of the main components and operations within WABUHA. Each part of the user interface and its corresponding function will be clarified. The objective is to outline how each feature contributes to the overall utility and workflow for WABUHA, ensuring a complete understanding of the capabilities.

4.3.1 Disclaimer Notice

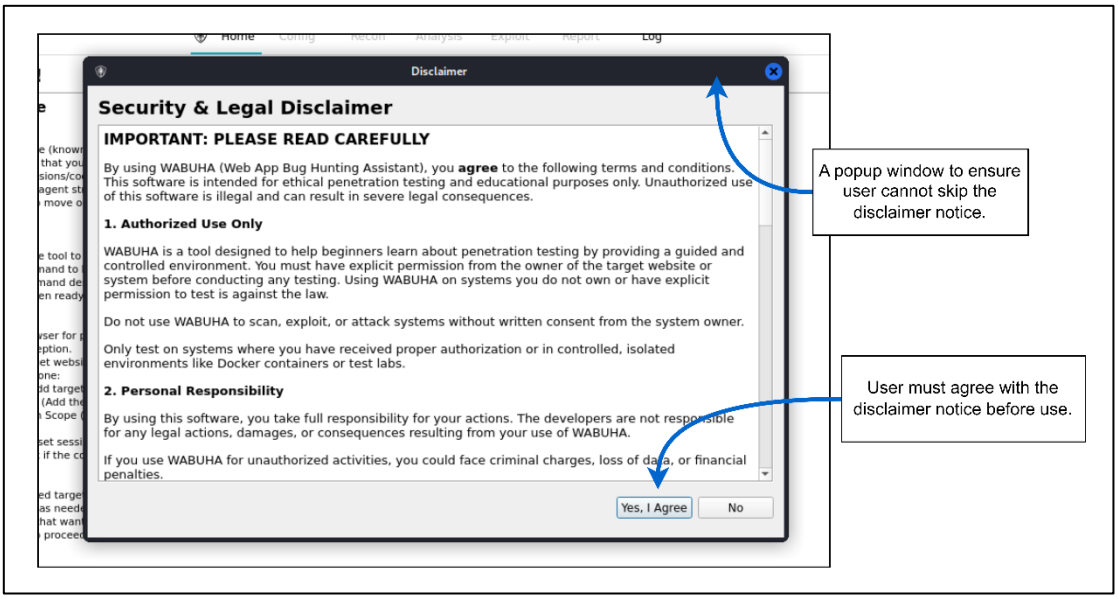


Figure 4.6: Disclaimer Dialog.

Figure 4.6 displays the “Disclaimer” popup dialog, which is presented to the user each time the application is launched. This dialog outlines important security and legal disclaimers. Users are required to read and accept the disclaimer before proceeding. If the user declines or does not agree, the application will terminate automatically. Access to and use of WABUHA is permitted only upon the user’s explicit acceptance of the disclaimer, thereby ensuring acknowledgment of ethical and legal responsibilities associated with its usage.

4.3.2 Home Tab

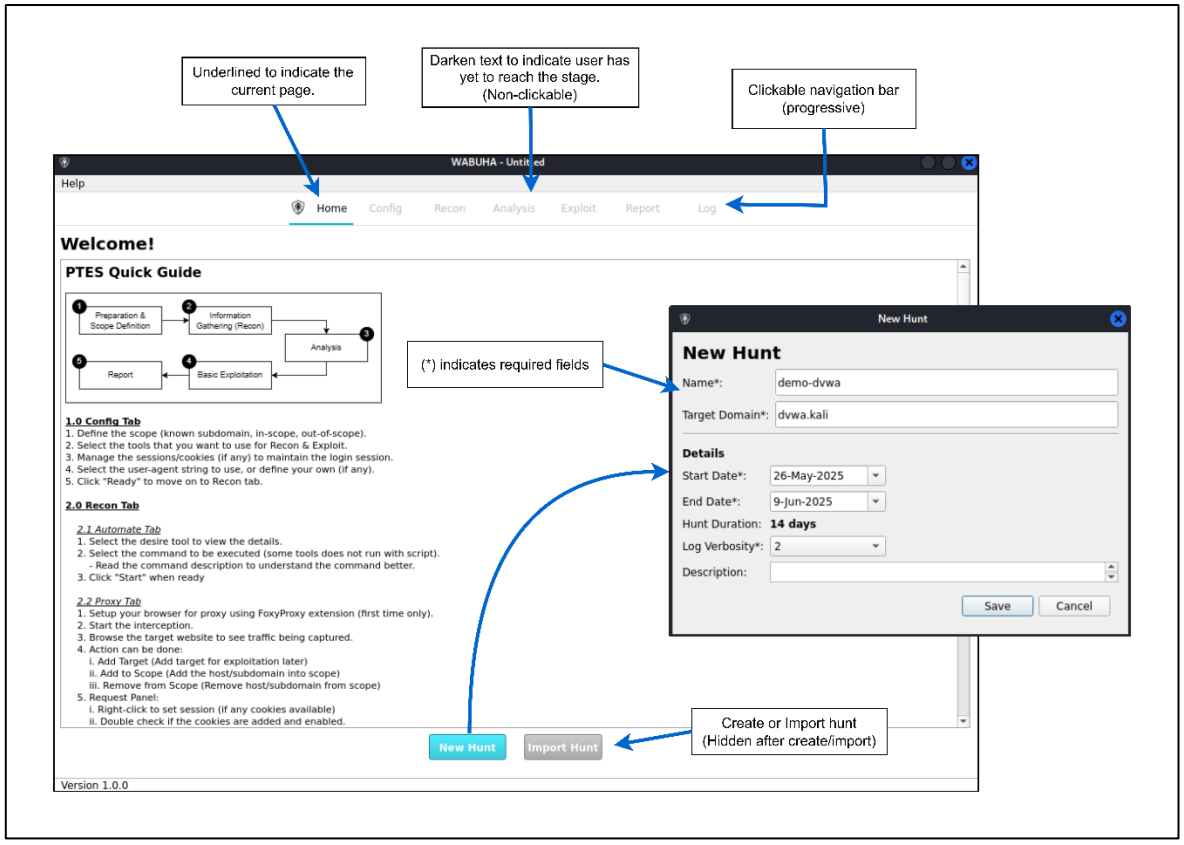


Figure 4.7: Home Tab and New Hunt Creation Dialog.

Figure 4.7 illustrates the Home tab of the WABUHA application. Positioned below the "Welcome" banner is a quick-start guide designed to assist users in navigating the main features of the application. Directly beneath the guide are two primary action buttons: "New Hunt" and "Import Hunt."

Clicking the "New Hunt" button prompts the user to input the details for a new hunt. Fields marked with an asterisk (*) are mandatory. The form includes "Start" and "End" date selectors, enabling users to define the time frame for their bug bounty engagement. This helps ensure that testing is conducted within the authorised testing window, particularly for time-bound programs. The hunt duration is automatically calculated based on the selected dates, with a default period of 14 days.

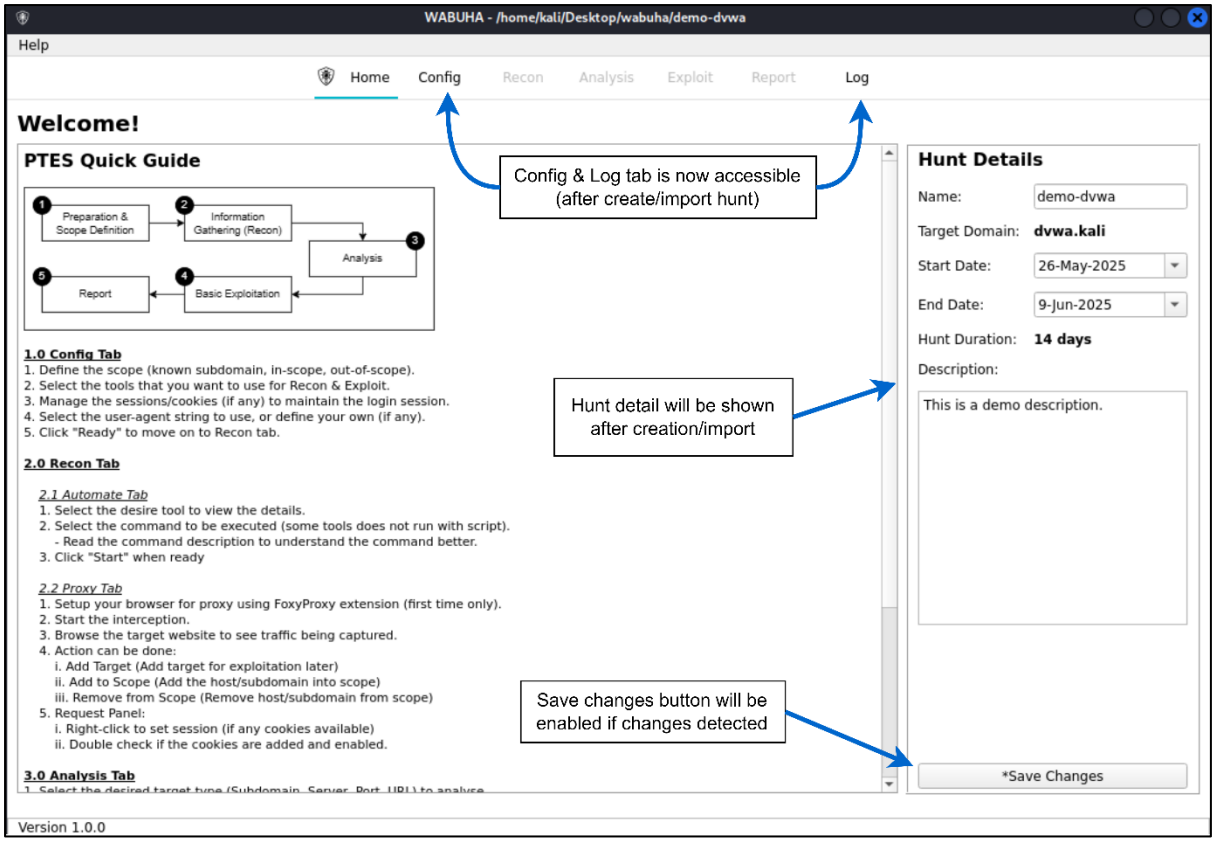


Figure 4.8: Home Tab After Hunt Created or Imported.

After creating a new hunt or importing an existing one, the corresponding hunt details are displayed in a side panel, as illustrated in Figure 4.8. Upon successful hunt initialisation, the Config and Log tabs become accessible to the user, allowing further configuration and activity monitoring. If any modifications are made to the hunt details, the “Save Changes” button is automatically enabled, prompting the user to confirm and store the updated information.

4.3.3 Configuration Tab

The Config tab is organised into four sub-tabs: Scope, Tool Selection, Session, and More. Each sub-tab serves a distinct purpose, designed to logically group related configuration settings in a structured and progressive manner. This modular design enhances usability by guiding users through the setup process in a clear and methodical sequence.

4.3.3.1 Scope

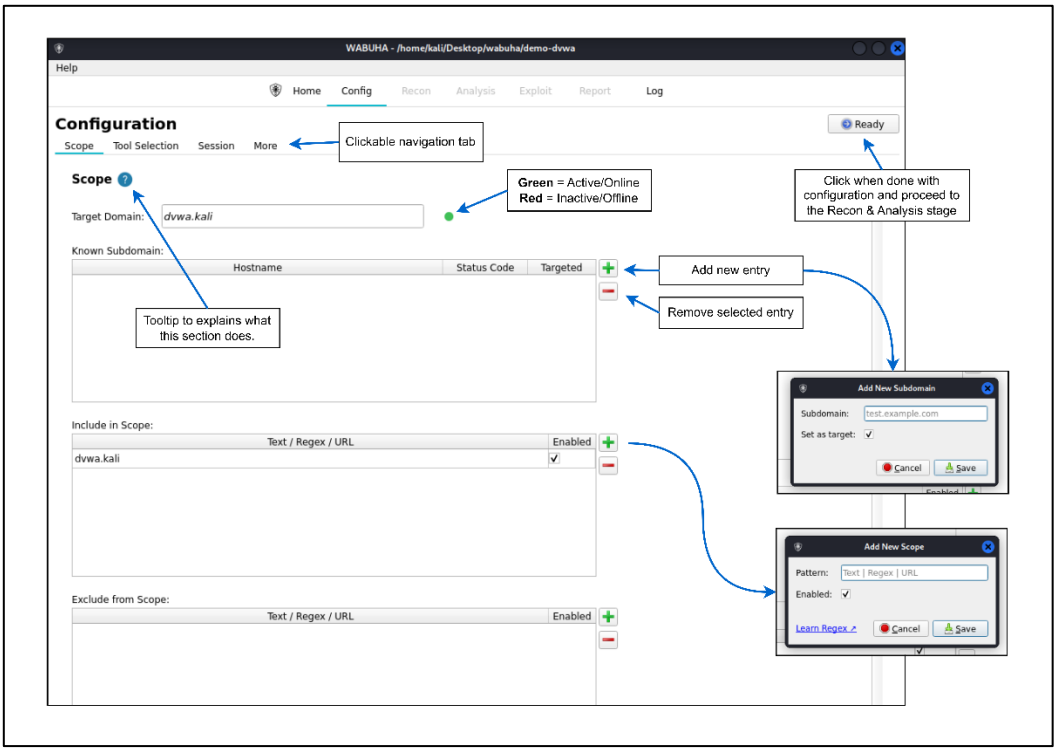


Figure 4.9: Scope Sub-tab in Config Tab.

Figure 4.9 presents the Scope tab, which comprises four key sections: Target Domain, Known Subdomains, Include in Scope, and Exclude from Scope. The Target Domain field is non-editable, as it is predefined during the initial hunt creation. A status indicator icon as green for active and red for inactive is displayed beside the domain text box to visually reflect the current availability of the domain.

The Known Subdomains section allows users to manually input any known subdomains associated with the target. This is particularly useful in cases where automated tools may fail to enumerate all available subdomains during the reconnaissance phase.

The Include in Scope and Exclude from Scope sections accept inputs in both plain text and regular expression (regex) formats. These inputs help define the boundaries of the assessment by matching specific patterns associated with the target assets. This ensures that WABUHA performs testing within the intended and authorised scope, thereby promoting ethical and controlled penetration testing practices.

4.3.3.2 Tool Selection

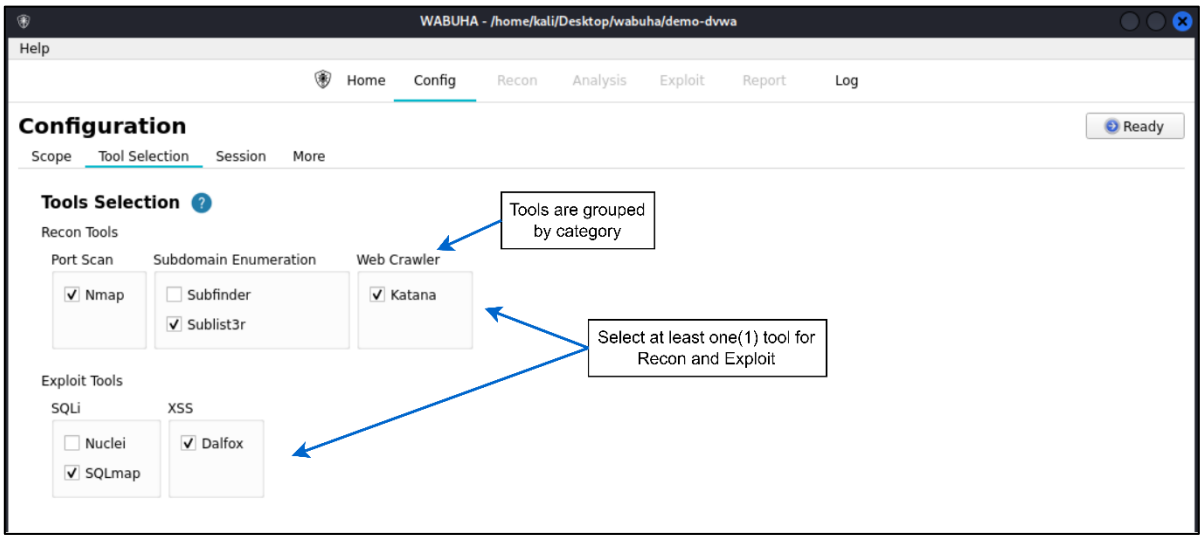


Figure 4.10: Tool Selection Sub-tab in Config Tab.

Figure 4.10 shows the selected tool(s) for recon and exploit, but during the recon and exploit, the selection panel will be disabled and will be non-editable until the process is finished. Tools can be selected by default by changing the “is_selected” option in the “tools.json” file to “True” (refer Figure 4.11 below).

```
"tools": {
    "recon": [
        {
            "category": "Subdomain Enumeration",
            "tools": [
                {
                    "class_name": "Sublist3r",
                    "name": "Sublist3r",
                    "description": "Sublist3r is a python tool designed to enumerate subdomains of websites using OSINT.",
                    "origin": "https://github.com/aboul3la/Sublist3r",
                    "is_enabled": true,
                    "is_selected": true,
                    "api_key": "",
                    "support_cli": true,
                    "command": [ ···
                    ]
```


Figure 4.11: Sample tool.json for Tool Details.

4.3.3.3 Session Management

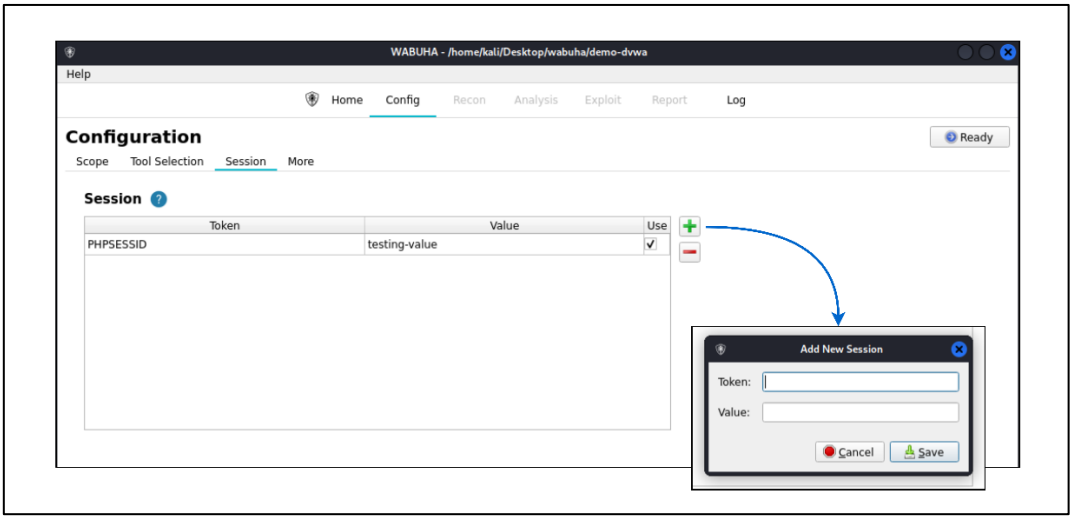


Figure 4.12: Session Sub-tab in Config Tab

Figure 4.12 shows the Session tab. User can manually add the cookies here, or add them easily via the Proxy feature under the Recon tab (refer to section 4.3.4.2). The Session tab is to let user easily manage the cookies by using in the recon or exploit later by checking the “use” box.

4.3.3.4 More Configuration

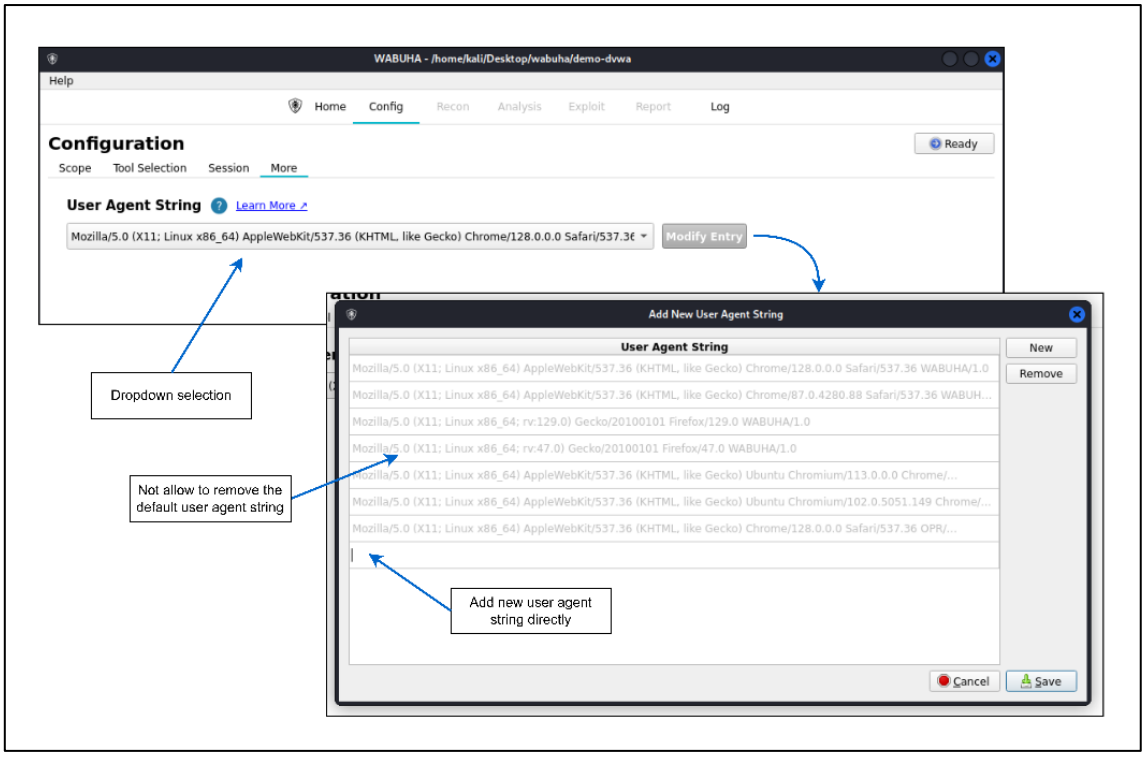


Figure 4.13: More Sub-tab in Config Tab.

Figure 4.13 shows the “More” tab and the user agent management window upon clicking the “Modify Entry” button. The default user agent strings will always have a trailing “WABUHA/1.0” to indicate the HTTP traffic is coming from WABUHA application and to allow firewall to whitelist the traffic by the vendor (if needed). However, user may add custom user agent string for system flexibility.

```
def config_ready(self):

    # enable until analysis tab
    self.main_ui.tabMain.setTabEnabled(Constants.TabNumber.RECON, True)
    self.main_ui.tabMain.setTabEnabled(Constants.TabNumber.ANALYSIS, True)
    self.app.set_user_progress(Constants.TabNumber.ANALYSIS)

    # Hide config ready button and move to recon tab
    self.main_ui.configReady_btn.setHidden(True)
    self.main_ui.tabMain.setCurrentIndex(Constants.TabNumber.RECON)

    self.logger.info("Configuration tab is ready.")
```

Figure 4.14: Update User Progression in PTES Stages.

After configuring all the necessaries and making sure everything is in-scope, user can proceed to the Recon and Analysis tab by clicking on the “Ready” button at the top-right. In Figure 4.13, the user progress will also be saved; the next import of the hunt will restore back the user progress.

4.3.4 Reconnaissance Tab

The Reconnaissance (Recon) tab contains two sub-tabs: "Automate" and "Proxy." The "Automate" tab provides access to several commonly used reconnaissance tools. Some of these tools may require session cookies to function properly and maintain user authentication.

The "Proxy" tab, on the other hand, serves as a lightweight HTTP traffic capture utility, allowing users to intercept and inspect raw HTTP requests and responses during manual browsing.

4.3.4.1 Automate

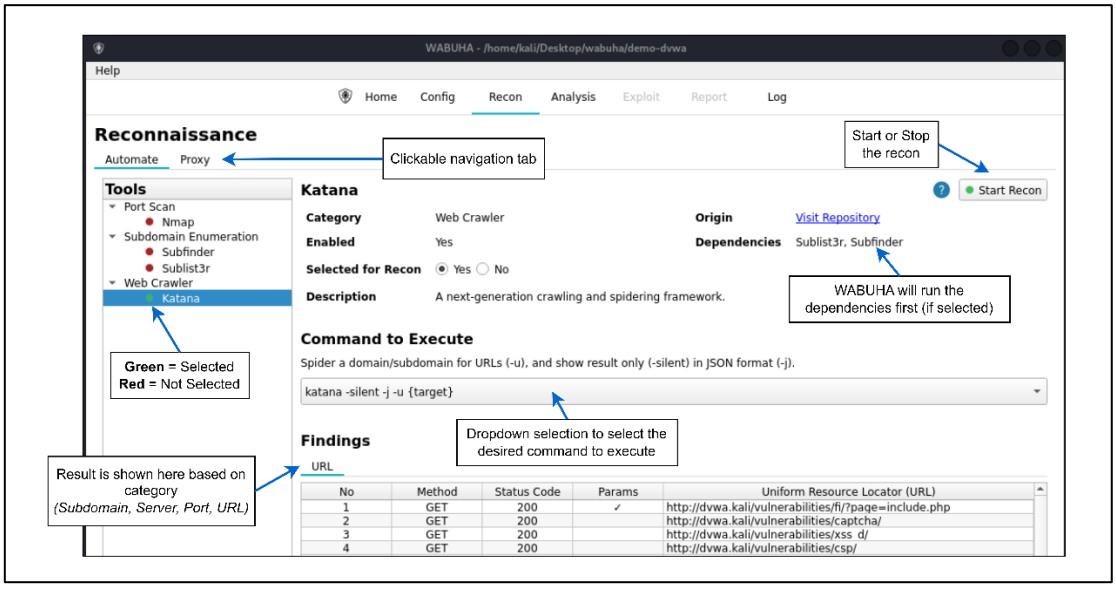


Figure 4.15: Automate Sub-tab in Recon Tab.

Figure 4.15 illustrates the "Automate" tab, which features a left sidebar displaying reconnaissance tools categorised into Port Scanning, Subdomain Enumeration, and Web Crawling. To use a tool, the user must set the "Selected for Recon" option to "Yes", indicated by a green dot icon in the sidebar. The user can then choose the desired command from the dropdown, with each command accompanied by a brief explanation. Before executing the main tool, WABUHA automatically checks for and runs any required dependency tools that have been selected. Currently, WABUHA supports Nmap for reverse DNS and port scan, Subfinder and Sublist3r for subdomain enumeration, and Katana for (active) web crawler. The right panel is divided into three sections: Tool Information, Command to Execute, and Risk Found.

If the target domain is a public-facing application (i.e., no authentication is required to access the website), the user can start the recon phase immediately. However, for internal-facing applications that require authentication, the user may need to provide valid session cookies to maintain access. These cookies can be added manually through the Config tab under the Session section or captured and set via the Proxy tab. User will see this popup as shown in Figure 4.16 once the recon is Completed(C), Error(E), or Killed(K).

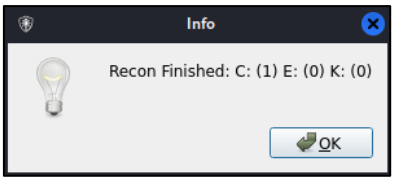


Figure 4.16: Recon and Exploitation Completion View.

4.3.4.2 Proxy

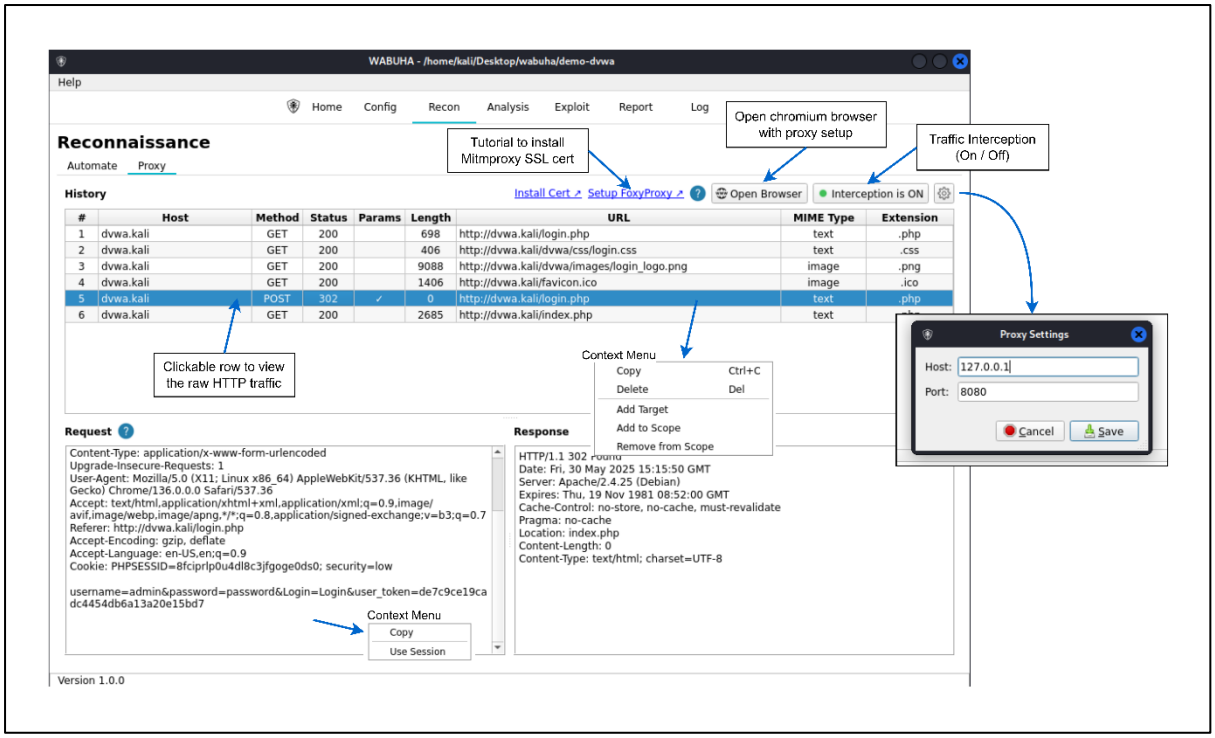


Figure 4.17: Proxy Sub-tab in Recon Tab.

Figure 4.17 illustrates the “Proxy” tab. According to mitmproxy (2025), the user is required to setup their browser to trust the Mitmproxy Certificate Authority (CA) certificate and install the FoxyProxy browser extension. Instructions are provided highlighted in blue and underlined. Alternatively, user can just click on the “Open Browser” button to launch a Chromium browser with the CA Cert installed and proxy setup.

After successfully setup the pre-requisites, user can start the interception (indicated with a green dot). Then, browse the target as usual to see the traffic being captured. User may right-click at any row to add the URI as target or add and remove from scope.

4.3.5 Analysis Tab

The Analysis tab consist of four sub-tabs: “Subdomain”, “Server”, “Port” and “URL”. This is to simplify the view for the user and easier to manage and filter later. The Analysis stage is essentially important for both Recon and Exploitation stages. User can set the asset as target or remove them from target to ensure it is test within the scope.

4.3.5.1 Subdomain

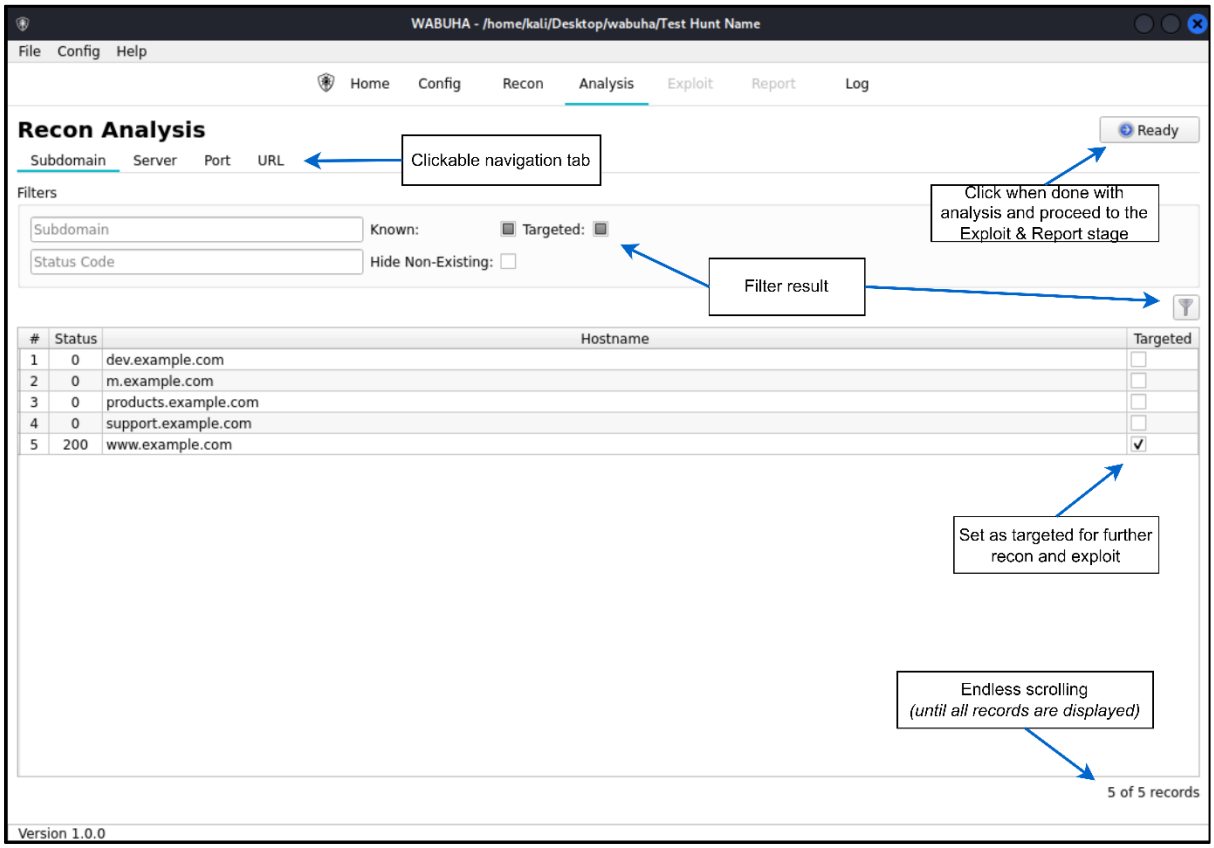


Figure 4.18: Subdomain Sub-tab in Analysis Tab.

The Subdomain sub-tab, as shown in Figure 4.18, consists of two main panels: the "Filter" panel and the "Findings" panel. The Filter panel can be collapsed by clicking the filter icon. Users can filter results based on subdomain name, HTTP status code, whether the

subdomain is known (predefined in the Config tab), non-existent (status code 0), or marked as a target.

In the Findings panel, users can perform extended (multi-row) selections and apply bulk actions such as copy, delete, or browse. A total record count is displayed at the bottom-right corner of the panel. The table supports infinite scrolling and will automatically load more entries as the user scrolls near the bottom.

4.3.5.2 Server

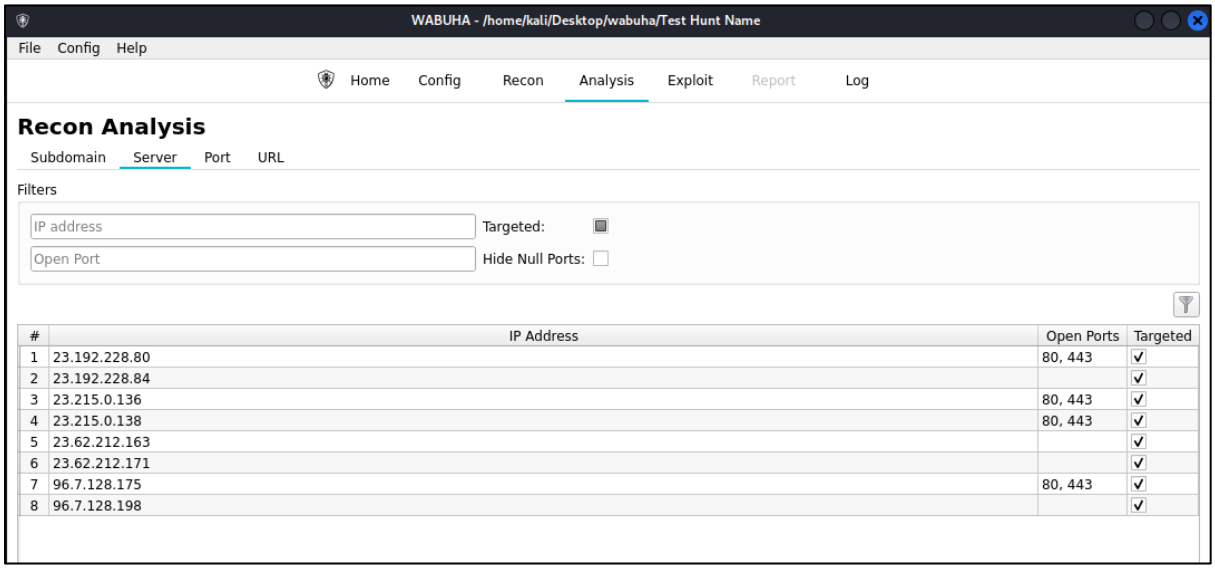


Figure 4.19: Server Sub-tab in Analysis Tab.

The "Server" sub-tab, shown in Figure 4.19, allows users to filter findings based on IP address, open ports, null ports, or whether the entry is marked as a target. This tab provides informational context, listing the available IP addresses and their associated open ports discovered during reconnaissance.

Currently, this server-related information is not used in the exploitation phase, as port-based attacks can be potentially destructive. WABUHA is designed as a learning tool for

beginners, with a focus on safe and ethical practices, rather than conducting real-world destructive testing. However, the targeted selection is prepared for future development and improvement.

4.3.5.3 Port

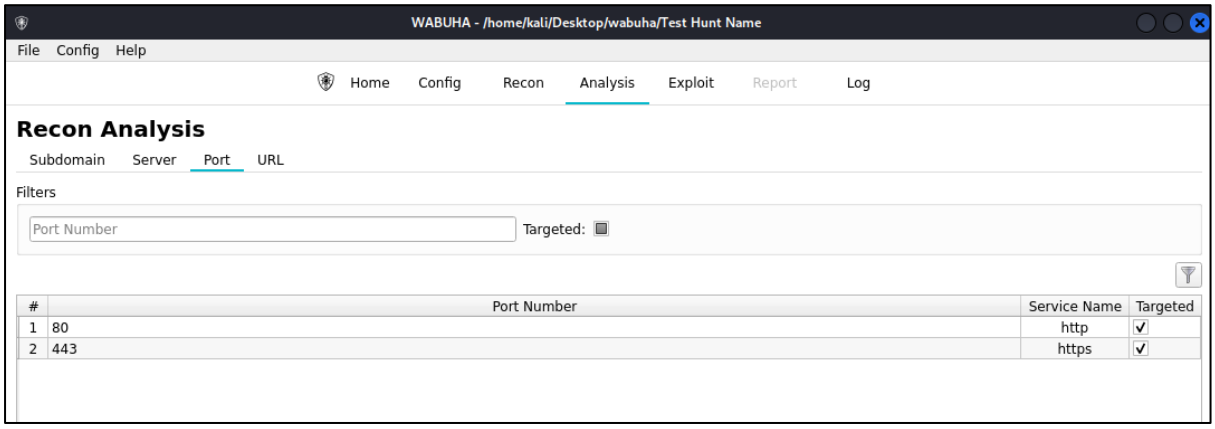


Figure 4.20: Port Sub-tab in Analysis Tab.

The “Port” sub-tab shown in Figure 4.20 allows user to filter by open port number or whether the entry is marked as a target. Same as the “Server” sub-tab, this port information is not used in the exploitation phase, as port-based attacks can be potentially destructive.

4.3.5.4 URL (Uniform Resource Locator)

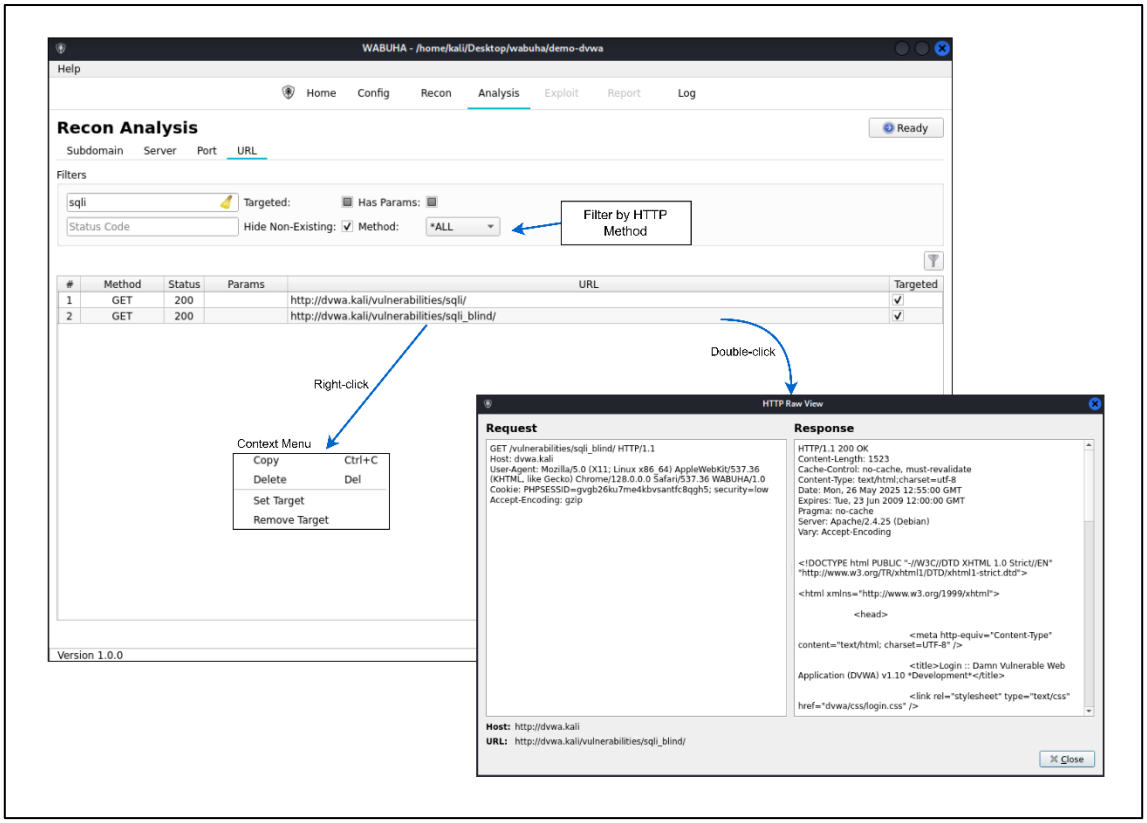


Figure 4.21: URL Sub-tab in Analysis Tab.

Figure 4.21 shows the "URL" sub-tab, which displays all reconnaissance data collected from the web crawler or manually added by the user via the "Proxy" tab. Users can filter the findings based on various criteria, including the URL, HTTP status code, existence (e.g., non-existent resources), target status, presence of parameters (determined by query strings in the URL or parameters in the POST body), and HTTP method.

Users can double-click any row in the table to view the raw HTTP request and response. Additionally, right clicking an entry provides options to copy, delete, set as target, or remove from target.

The "Has Param" column is particularly important for the exploitation phase, as it helps identify URLs that include parameters, it a key indicator for potential vulnerabilities such as SQL Injection and Cross-Site Scripting (XSS).

4.3.6 Exploitation Tab

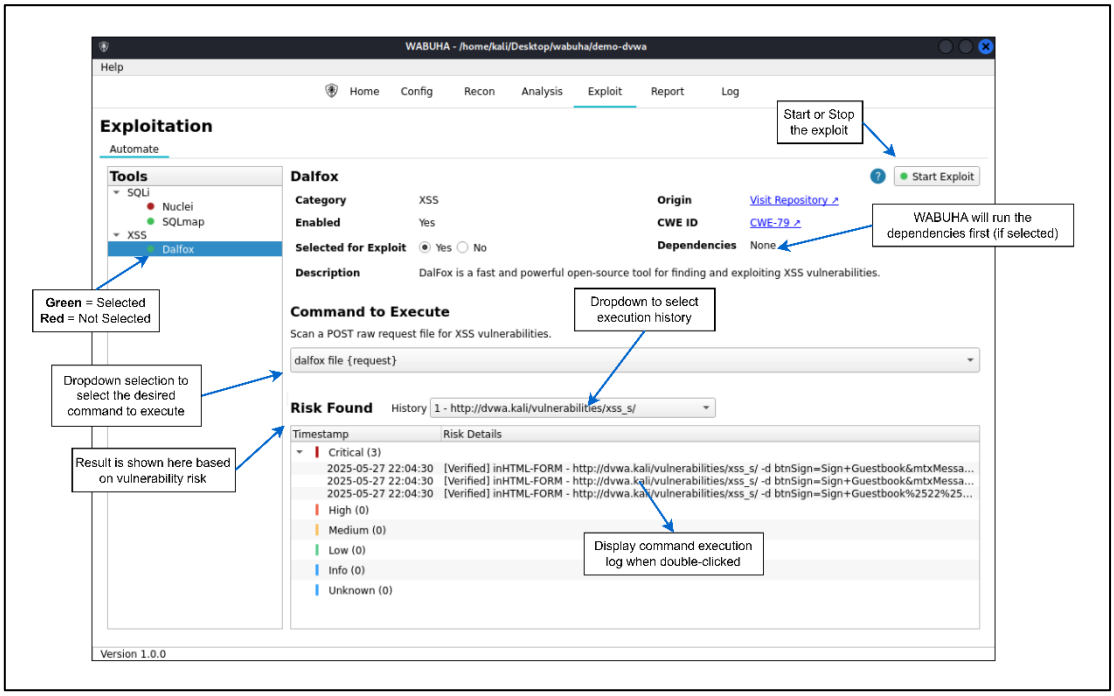


Figure 4.22: Automate Sub-tab in Exploit Tab

Figure 4.22 illustrates the Exploitation (Exploit) tab, which includes the "Automate" sub-tab and a command execution log dialog. Similar to the Recon tab, a "Start Exploit" button is provided to initiate the exploitation process using the selected tools. The left sidebar displays the available exploitation tools, categorised into two primary types: SQL Injection (SQLi) and Cross-Site Scripting (XSS).

Currently, WABUHA supports SQLmap and Nuclei for SQLi, and Dalfox for XSS. The right panel is divided into three sections: Tool Information, Command to Execute, and Risk Found.

In the Tool Information section, details are provided regarding the tool's category (SQLi or XSS), whether it is enabled, its origin (linked to the official project website), the associated CWE identifier, whether it has been selected for exploitation (as determined by the user), any required tool dependencies, and a brief description of the tool. Once the exploitation process begins, the selection button for enabling or disabling tools is disabled to prevent corruption of the ongoing process.

The Command to Execute section allows users to select a predefined command template from a dropdown menu. A description of the selected command is displayed to provide users with better understanding of its purpose and behaviour. This section is also disabled during exploitation to maintain the integrity of the process.

Upon completion of the exploitation phase, both the tool selection and command execution sections are re-enabled. Any identified vulnerabilities are displayed in the Risk Found section. Vulnerabilities are categorised into six severity levels: Critical, High, Medium, Low, Info, and Unknown.

Each command executed against a target is logged and stored in the execution history. This allows users to trace which specific command identified a vulnerability and which target was affected. Users can double-click on a history entry to view the complete execution log along with details of the vulnerability discovered, as illustrated in Figure 4.24.

4.3.7 Report Tab

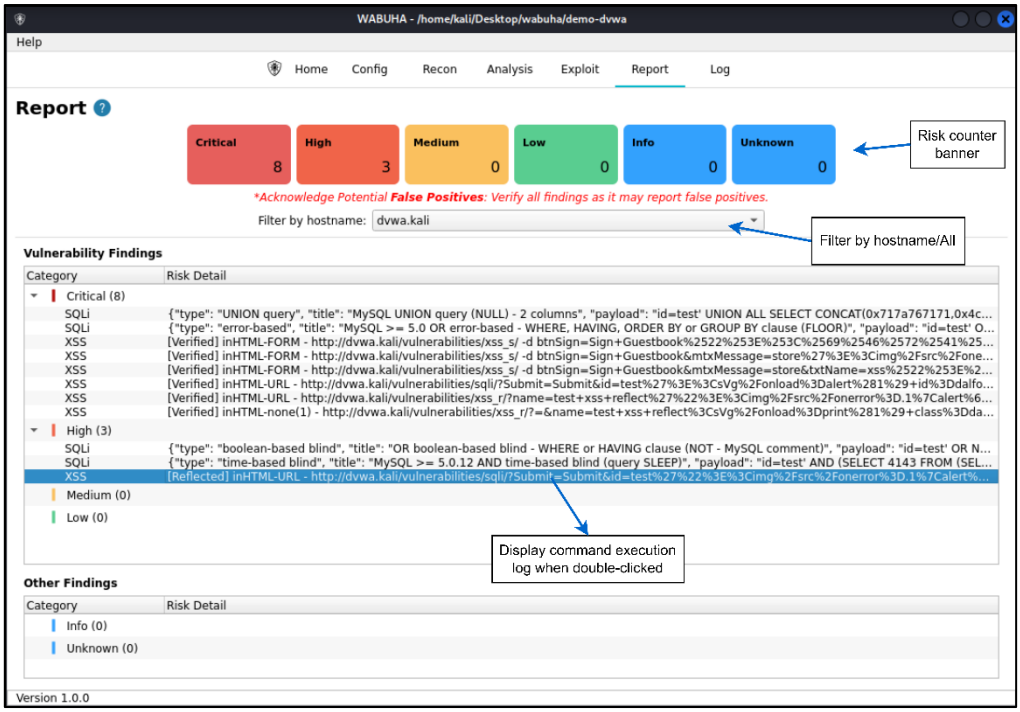


Figure 4.23: Report Tab.

Figure 4.23 presents the “Report” tab, which is organised into three main sections: a risk summary banner, vulnerability findings (categorised as Critical, High, Medium, and Low), and other findings (classified as Informational and Unknown). Users may filter the displayed results based on hostname or opt to view all unique vulnerabilities discovered during the exploitation phase. Additionally, users can double-click any row entry to open the corresponding Command Execution Log dialog (refer to Figure 4.24), which is similar in functionality to the log view found in the Exploit tab.

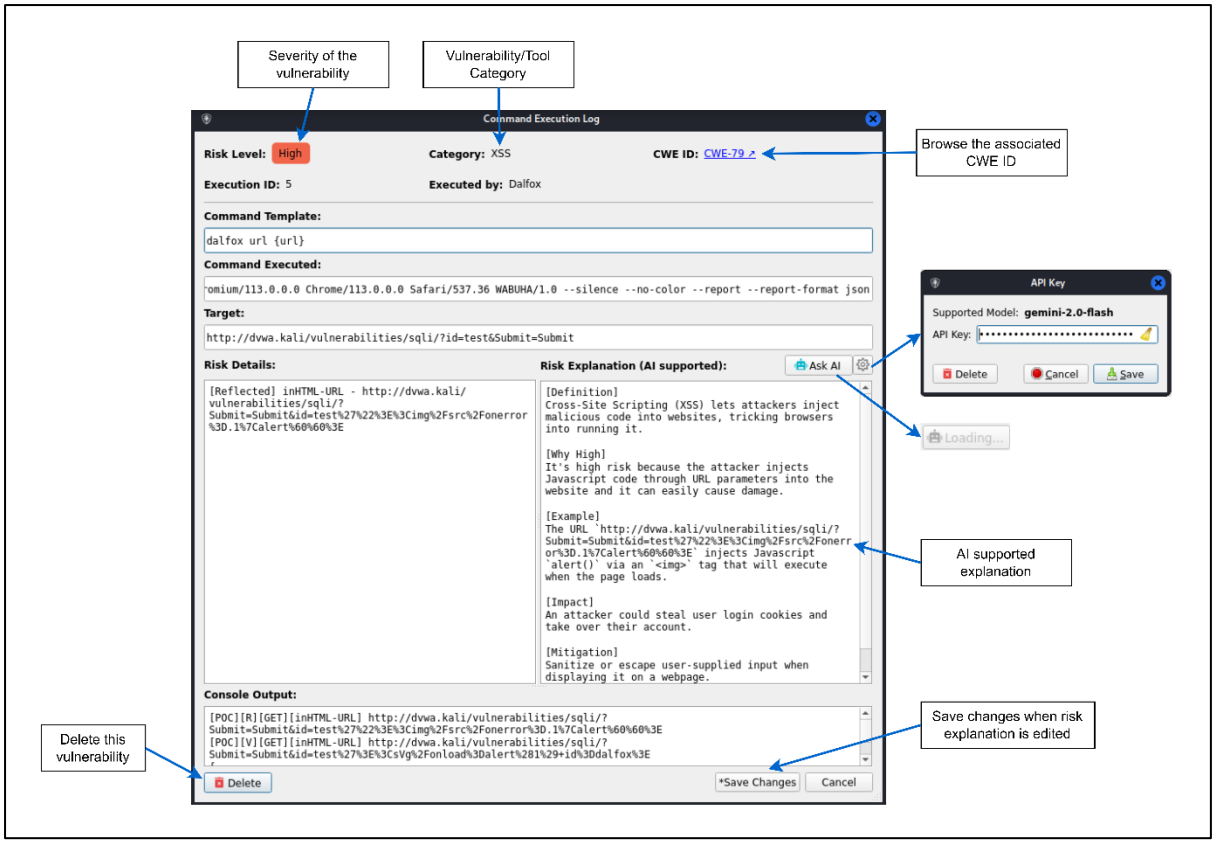


Figure 4.24: Command Execution Log Dialog.

Figure 4.24 shows the Command Execution Log dialog. This dialog provides comprehensive details of each executed command and its associated vulnerability. The displayed information includes risk level, tool category, CWE identifier, execution ID, tool used, command template, actual command executed, risk details, AI-generated risk explanation, and the full console output.

The AI-supported risk explanation feature enhances user understanding by summarising the risk in natural language. However, to utilise this functionality, users must supply a valid AI API key. Currently, only the Gemini 2.0 Flash model is supported for this feature. The AI-powered explanation is based on a predefined prompt template explained in the previous section 4.2.7.

4.3.8 Log Tab

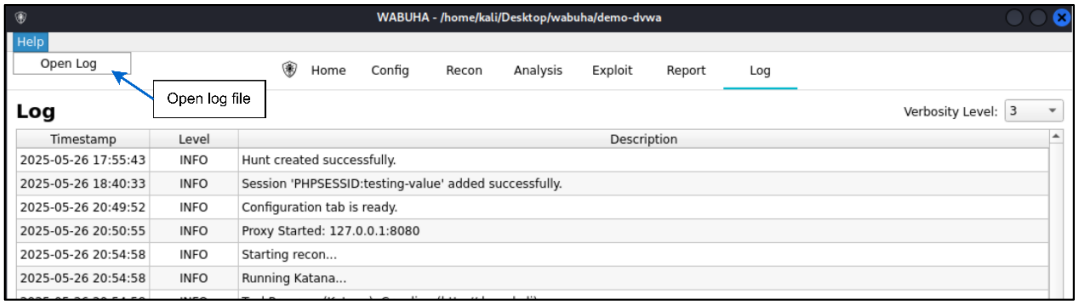


Figure 4.25: Log Tab.

Figure 4.25 illustrates the “Log” tab, which allows users to monitor system activity based on adjustable verbosity levels. A dropdown menu enables the user to select one of three levels of log verbosity:

- Level 1: Displays only error messages,
- Level 2: Displays both error and warning messages,
- Level 3: Displays error, warning, and informational messages.

This feature is designed to help users track the system’s operational progress and status in varying levels of detail according to their preference or troubleshooting needs. For access to a more detailed and persistent record of system activity, users can open the full log file via the “Help” menu located in the top-left corner of the application window by selecting “Open Log”.

4.4 Summary

This chapter presented the implementation of WABUHA, including its development environment, supported platform, the third-party components integrated and database integration into the system. By leveraging open-source tools and Python libraries, WABUHA provides a streamlined and educational penetration testing experience for beginners. The next chapter will evaluate the system’s performance and usability based on functional testing and user acceptance test with feedback.

Chapter 5: Testing and Analysis

5.1 Introduction

This chapter details both the testing methodology and the results obtained, providing insights into operational stability and user experience of WABUHA. This is also to ensure the system meets its requirements and performs as intended in real-world scenarios. Two main categories of software testing were employed, functional testing and user acceptance testing (UAT).

5.2 Software Testing

Software testing was carried out to assess WABUHA's functional correctness and to detect any potential defects before deployment (Sandler et al., 2013). The goal was to ensure that each component of the system performs according to the predefined specifications. Functional testing was chosen as the primary method for validating the accuracy and completeness of WABUHA.

The evaluation methods for WABUHA were derived from the toolkit focus on technical functionality and beginner usability. Functional Testing was structured according to system requirements and use cases developed during the design phase. These verified that each module of the system meets the expectations (as shown in *Appendix C*).

Meanwhile, User Acceptance Testing (UAT) focused on user experience, learnability, and clarity of workflow, which aligned with the objectives of educating and guiding beginner penetration testers. UAT feedback was collected via structured Likert-scale questionnaires. These methods ensure WABUHA is both technically sound and appropriate for beginners.

5.2.1 Functional Testing

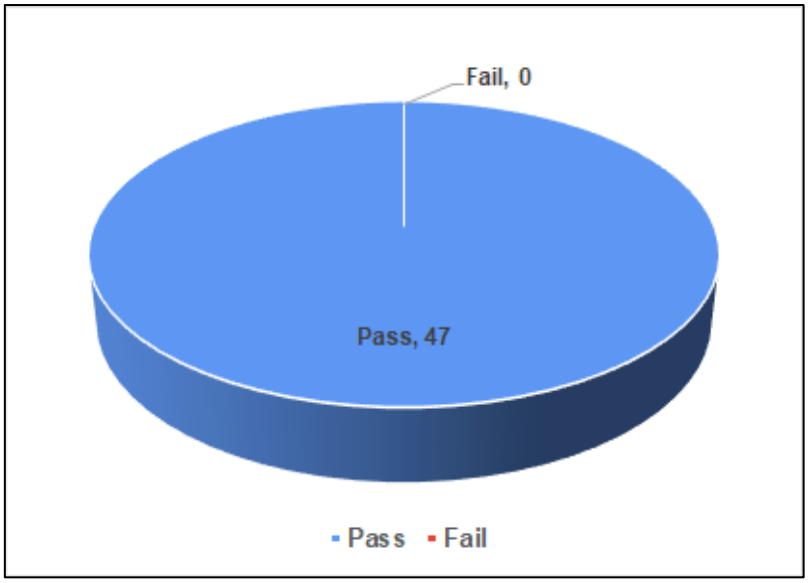


Figure 5.1: Summary of Functional Test Results

Figure 5.1 shows the result summary of the functional tests conducted on WABUHA, demonstrating its successful operation. Out of 47 test cases executed, all 47 passed (100% success rate). The detailed procedure for these functional tests can be found in *Appendix C*. Hence, it confirms WABUHA meets its specified requirements and performs all intended functions as designed, providing a robust and reliable foundation for its deployment and further development. The absence of any failed tests highlights the stability and correctness of the implemented features.

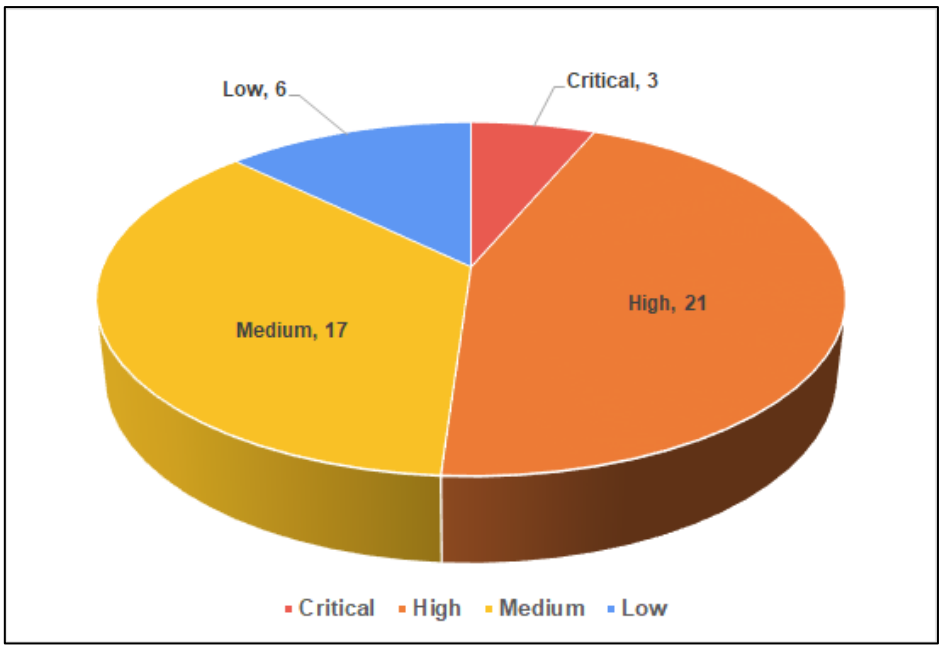


Figure 5.2: Summary of Test Cases Severities

Figure 5.2 illustrates a pie chart for software test case severities: Critical (3 test cases) signifies complete system failure; High (21 test cases) indicates significant functional impairment; Medium (17 test cases) denotes impacts on non-critical functionalities; and Low (8 test cases) represents minor cosmetic issues. This hierarchical classification is essential for prioritising defect resolution and assessing software stability.

5.2.1.1 Platform Testing Results

WABUHA was tested on three different operating systems to check if it works properly across platforms. The system ran successfully on **Kali Linux** and **Ubuntu 22.04** (Debian-based Linux distribution). These platforms provided the environment and tools needed for WABUHA to function as expected.

However, WABUHA did not work on Windows OS. Some of the tools used in WABUHA could not run or behaved differently, and parts of the program that rely on the Linux command-line system did not function properly. This shows that WABUHA is currently only suitable for use on Debian-based Linux distribution systems. These results further highlight the need for further development if support for other operating systems is required in the future.

5.2.2 User Acceptance Testing

In this section, the User Acceptance Testing (UAT) is carried out in accordance with a feedback-based approach to see how well the 15 users (respondents) accept and interact with WABUHA (Joshi et al., 2015)The testing includes a 24 questions Likert scale questionnaire (refer *Appendix D*) to measure user satisfaction, along with five (5) open-ended questions to gather detailed opinions and suggestions. This helps provide both measurable and descriptive insights into the user experience.

5.2.2.1 Likert Scale Questionnaires

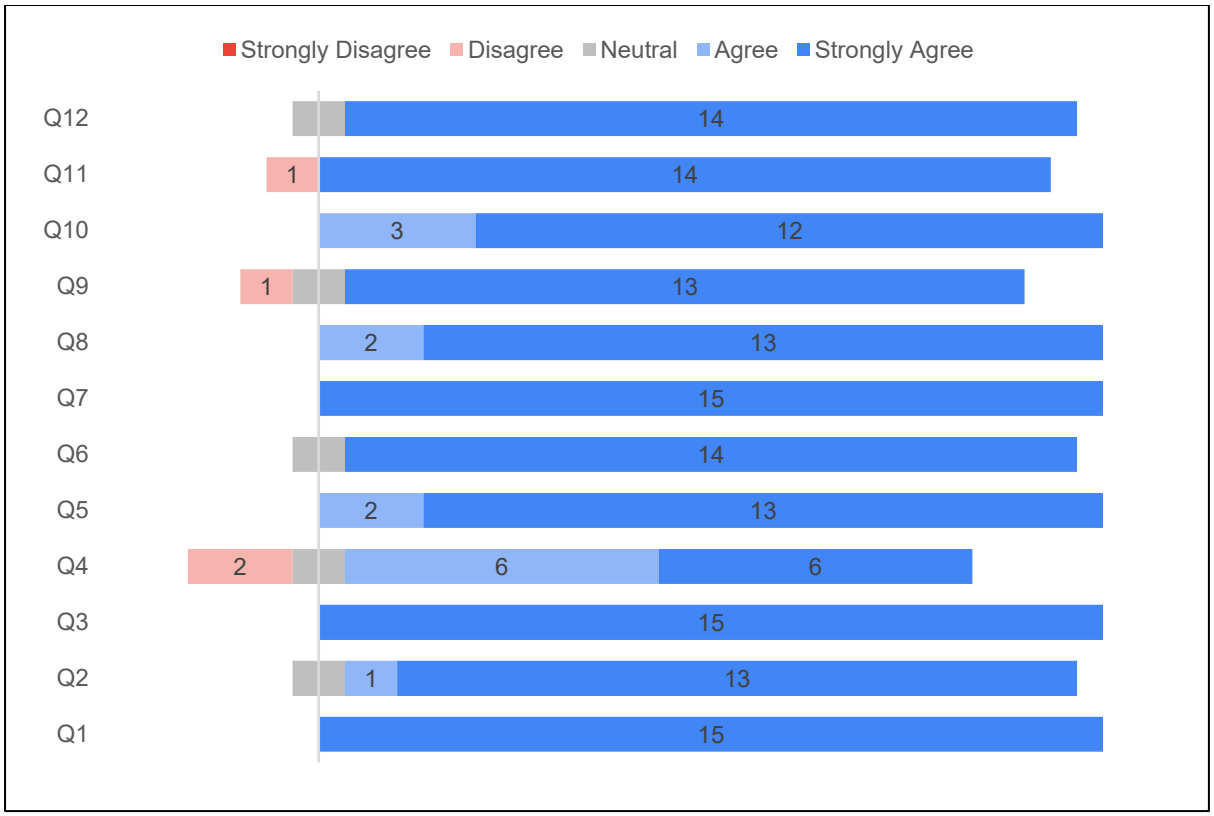


Figure 5.3: Summary Result for Q1-Q12 Questionnaires.

Based on the Figure 5.3, with most responses being "Agree" or "Strongly Agree," the system's early phases, such as configuration and early reconnaissance or exploitation steps (Q1-Q3, Q5-Q8, Q10, Q12), received mostly positive feedback. Questions Q1, Q3 and Q7 received overwhelmingly "Strongly Agree" responses. This suggests that users are highly satisfied with the system's usability and intuitiveness.

Questions about feedback or status updates during reconnaissance (Q9), which also received a few unfavourable replies, and managing session settings (Q4), where a small percentage of respondents indicated "Disagree" or "Neutral" sentiment, were minor areas that could use improvement. Likewise, one "Disagree" comment was received for eliminating findings (Q11). Even while these are only slight deviations from the generally encouraging trend, they highlight certain areas that may be strengthened to provide clarity for all users.

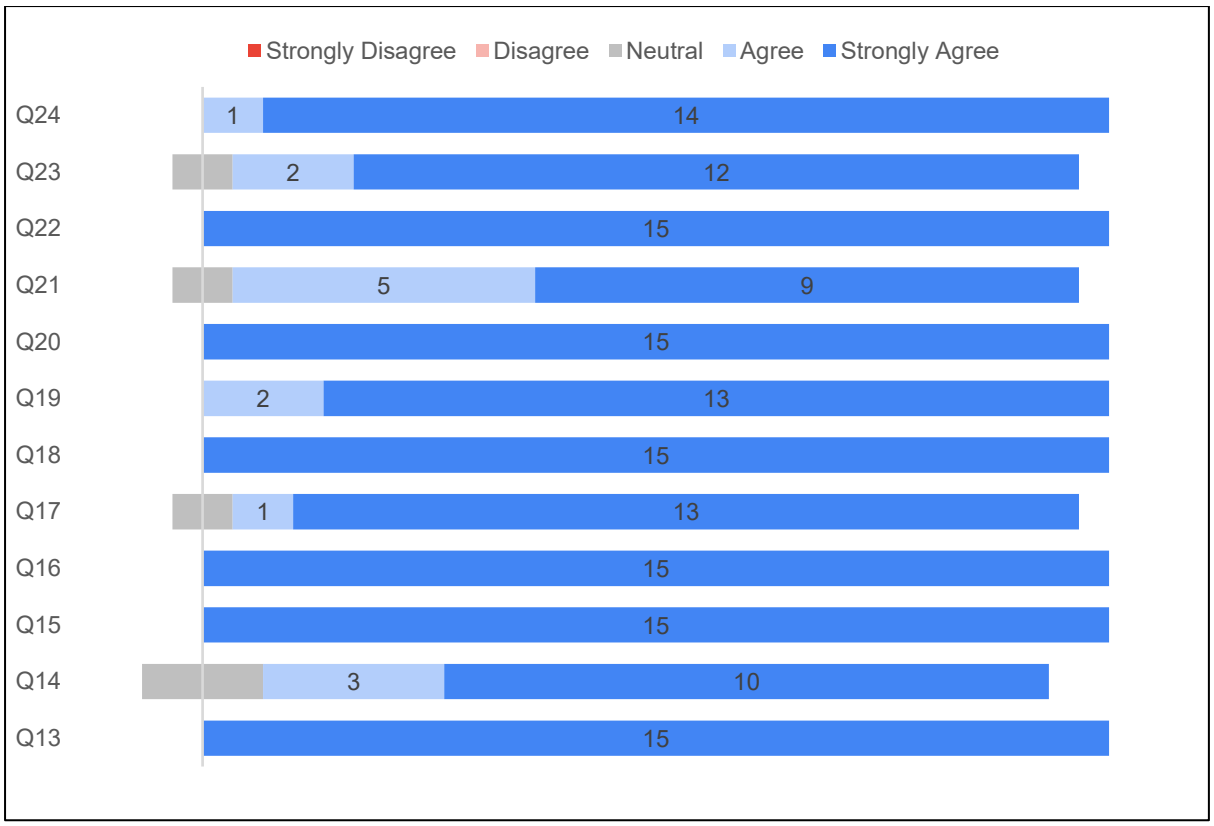


Figure 5.4: Summary Result for Q13-Q24 Questionnaires.

Figure 5.4 shows that users were very happy with the advanced features. Everyone strongly agreed that mapping findings to common weakness types (Q13), clear risk categories (Q15), intuitive viewing of detailed logs (Q16), helpful AI explanations (Q18), and easy management of AI settings (Q20) were excellent.

Most users also found the technical information in the logs meaningful (Q17) and the AI keys settings (Q19). The system generally helped users stay organised (Q22) and made complex tasks easier to understand (Q23), with strong recommendations for new users (Q24).

A small number of users were neutral about how informative and easy to access vulnerability details were (Q14) and the clarity of the report summary (Q21), suggesting these areas, while mostly good, could be slightly improved.

5.2.2.2 Open-Ended Questionnaires

To gain deeper insights into user perceptions and experiences, a set of five open-ended questions was included in the evaluation. These questions aimed to capture qualitative feedback on WABUHA's usability, functionality, and areas for improvement. The responses were analysed and are summarized in Tables 5.1 to 5.5, each addressing a specific aspect of the user experience.

Table 5.1: User feedback on most valuable features in WABUHA.

Q1	**What feature or functionality did you find most valuable in WABUHA, and why?**
R1	The risk categorization were helpful and it gave me a clear picture of my findings without needing to dig through all the raw data and AI supported explanation which is really helpful
R2	The ability to run recon and exploit tools asynchronously saved a lot of time and made the workflow more efficient
R3	I like the AI supported vulnerability explanation.
R4	The tabbed interface was very clean and easy to follow. I never felt lost while navigating through the different phases
R5	Having AI-generated explanations for the vulnerabilities made it easier to understand what was going on. However, it needs our own API key which might need us to subscribe when heavy use.

As tabulated in Table 5.1, the question aimed to identify which features of WABUHA users found most valuable and why. The responses highlighted several key strengths of the application. Most of the respondents appreciated the AI-supported explanations of vulnerabilities, which helped them better understand findings without needing extensive manual research. Additionally, users mentioned the risk categorisation feature for providing a clear, high-level overview of the findings, allowing for quicker prioritization. Another commonly valued aspect was the ability to run reconnaissance and exploitation tools asynchronously, which improved workflow efficiency. The intuitive tabbed interface also received positive feedback for enhancing navigation and user experience. However, one user

noted that requiring an external API key for AI features could be a limitation, especially in high-usage scenarios.

Table 5.2: Reported Moments of Confusion or Uncertainty During WABUHA Usage.

Q2	**Were there any moments during your usage where you felt unsure or confused? Please describe the situation.**
R1	I was not sure if my proxy settings had applied correctly. It might help to have a tutorial video.
R2	Only once, I got confused between whether a finding had already been excluded from the exploitation phase—it was a minor UI clarity issue
R3	At first I not sure how the AI API key worked, but the error prompt explained it pretty clearly
R4	I don't know the format to use for defining the scope, but the examples helped clarify that quickly.
R5	There was a slight learning curve on how session cookies were managed, but the interface was still manageable overall.

In Table 5.2, the question explored the confusion or uncertainty users experienced while using WABUHA. The feedback revealed minor usability challenges rather than critical issues. Some users faced initial confusion about specific configurations, such as proxy settings and the format for defining the target scope. Other issues related to session cookie management or determining whether certain findings had been excluded from the exploitation phase. A few respondents also mentioned uncertainty about the AI API key, although they reported that the prompts helped resolve this. Overall, users encountered manageable learning curves, often resolved through existing tooltips, but suggestions such as adding a tutorial video were proposed to improve onboarding.

Table 5.3: User Suggestions for Improving WABUHA's Functionality and Usability.

Q3	What improvements or new features would you suggest to make WABUHA more effective or user-friendly?
R1	Maybe a dark mode version would be great.
R2	It would be nice to have more tooltips on hover for more advanced tool options or terminology
R3	Can include more tools.
R4	Perhaps the ability to have custom command templates for different types of assessments would speed things up even more.
R5	An option to generate a PDF report in addition to Excel would be helpful for sharing with clients

The third question shown in Table 5.3 asked users to suggest improvements or new features to enhance WABUHA and their responses. The responses reflected both usability enhancements and functional additions. Users recommended features such as a dark mode, more tooltips for complex options, and PDF report generation to complement the existing Excel export, particularly for client sharing. In terms of functionality, respondents suggested the inclusion of more integrated tools and support for custom command templates, which could streamline different types of web security assessments. These suggestions indicate that while WABUHA is already effective, users see opportunities for further customisation and improvements.

Table 5.4: Impact of AI-Generated Vulnerability Explanations on User Understanding.

Q4	How did the AI-generated vulnerability explanation impact your understanding of the findings?
R1	It's more beginner-friendly than tools like Burp Suite, and the workflow feels much more guided
R2	WABUHA feels more organized compared to other scripts and toolkits I've tried. The structured tabs really helped
R3	The integration of AI explanation with potential mitigation is helpful.
R4	It's not as feature rich as some enterprise solutions, but it's definitely more intuitive and less overwhelming
R5	The experience was smoother overall.

The fourth question shown in Table 5.4 assessed how the AI-generated vulnerability explanations influenced the understanding of security findings. Responses were generally positive, highlighting the clarity, structure, and accessibility provided by the AI integration. Users found WABUHA to be more beginner-friendly and intuitive compared to traditional tools like Burp Suite or complex enterprise solutions. Several participants appreciated that the AI explanations were accompanied by potential mitigation suggestions, which enhanced their practical understanding. The structured tabbed interface was also mentioned for making the workflow feel more guided and organized. Overall, the AI-supported explanations contributed to a smoother and more educational experience.

Table 5.5: Overall User Feedback on the WABUHA Experience.

Q5	**Do you have any other feedback about your overall experience using WABUHA?**
R1	Really impressed with how seamless the whole process was. Great job.
R2	Appreciate the attention to detail, especially in the summary and reporting section. It made my final review so much easier
R3	It's a great tool for anyone getting into web app security. I would definitely recommend it to classmates or junior pentesters
R4	Thanks for building something that makes security assessment approachable and not intimidating.
R5	Looking forward to future updates. This already feels like a strong foundation to grow on.

Based on Table 5.5, the final open-ended question inquired the overall user experience with WABUHA. The responses were very positive, reflecting high satisfaction with the tool's usability, structure, and accessibility. Users described the workflow as seamless, which simplified final result reviews. Several respondents also noted that WABUHA is well-suited for beginners and junior penetration testers, appreciating its approachable design compared to more complex tools. The feedback suggested that users saw WABUHA as a solid, reliable foundation with potential for future enhancements. Some expressed enthusiasm about recommending the tool to peers, highlighting its value in educational and practical contexts.

5.3 Real-World Testing on Live and Simulated Targets

Table 5.6: System Feasibilities on Live Bug Hunting Scenarios.

No.	Target (Scenario)	Type	Vulnerability Found	
			SQLi	XSS
1	http://dvwa.kali	Local	3 Critical, 2 High	6 Critical, 2 High
2	http://bwapp.kali	Local	4 Critical, 4 High	3 Critical, 8 High
3	https://testphp.vulnweb.com	Live	9 Critical, 7 High	8 Critical, 13 High

Table 5.6 shows the summary of evidence and feasibility of WABUHA finding vulnerabilities in two local websites and one live target. The testing was conducted on different target environments to assess capability of WABUHA in different scenarios.

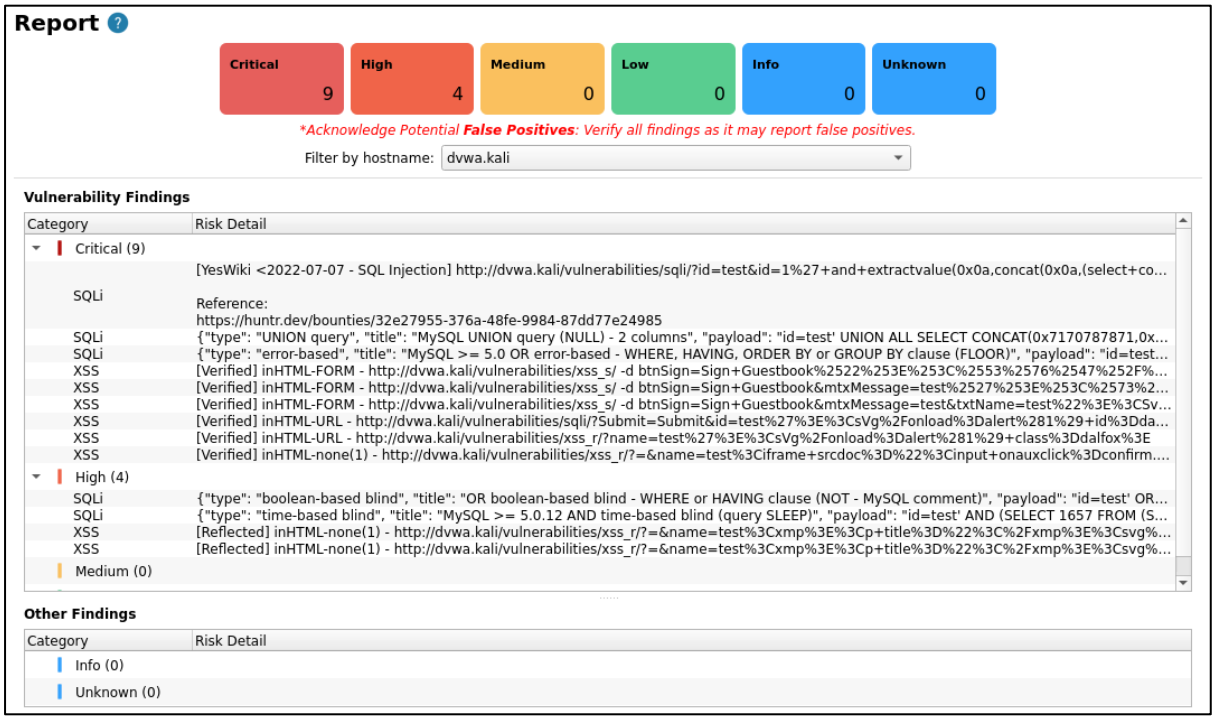


Figure 5.5: Vulnerability Report for DVWA.

For the first local target, http://dvwa.kali (Damn Vulnerable Web Application), WABUHA successfully identified a total of 9 critical and 4 high vulnerabilities. Specifically, the findings included 3 critical and 2 high-severity SQLi vulnerabilities, as well as 6 critical and 2 high-severity XSS vulnerabilities. Figure 5.5 provides a detailed report of these findings,

indicating specific SQLi payloads like 'UNION query' and 'error-based' alongside various reflected and verified XSS findings.

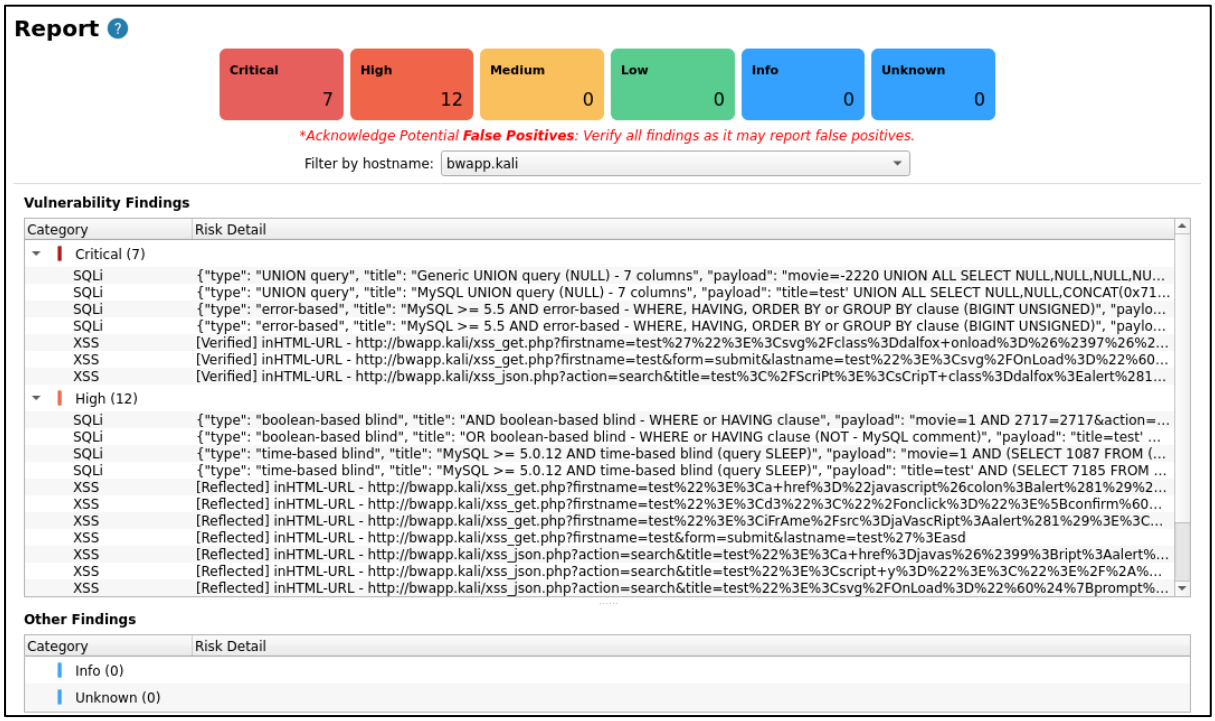


Figure 5.6: Vulnerability Report for bWAPP.

On the second local target, http://bwapp.kali (Buggy Web Application), WABUHA demonstrated robust detection by uncovering 7 critical and 12 high-severity vulnerabilities. This included 4 critical and 4 high-severity SQLi vulnerabilities, and 3 critical alongside 8 high-severity XSS vulnerabilities. The report shown in Figure 5.6 listed these findings, showcasing "boolean-based blind" and "time-based blind" SQLi types, as well as several "reflected" and "verified" XSS instances within the URL and HTML forms in the application.

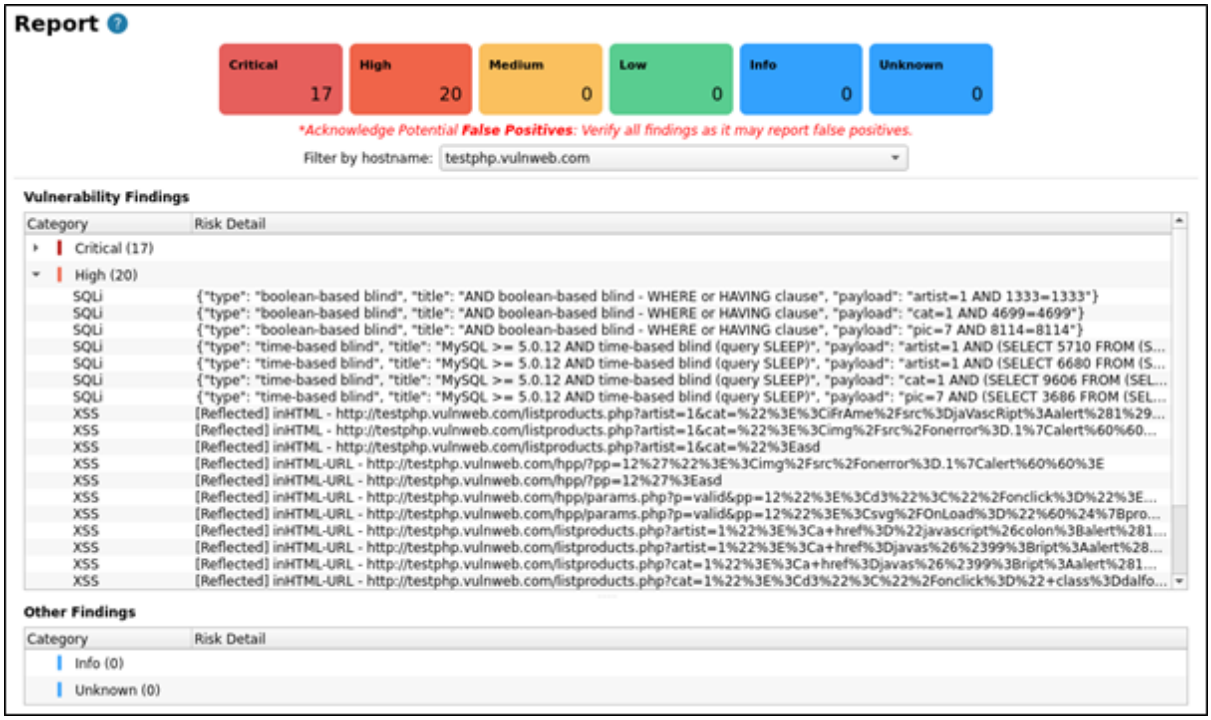


Figure 5.7: Vulnerability Report for Acunetix.

Finally, for the live target, https://testphp.vulnweb.com (provided by Acunetix), WABUHA proved its efficacy in real-world scenarios by identifying a significant number of vulnerabilities: 17 critical and 20 high. This comprised 9 critical and 7 high-severity SQLi vulnerabilities, and 8 critical along with 13 high-severity XSS vulnerabilities. As shown in Figure 5.7, the report details a variety of "boolean-based blind" and "time-based blind" SQLi attacks, in addition to numerous "reflected" XSS vulnerabilities affecting different parameters and pages of the live website.

These results demonstrate the ability of WABUHA to effectively detect a range of critical and high-severity SQLi and XSS vulnerabilities across both controlled local environments and a publicly accessible live web application, thus validating its practical feasibility as a bug-hunting tool.

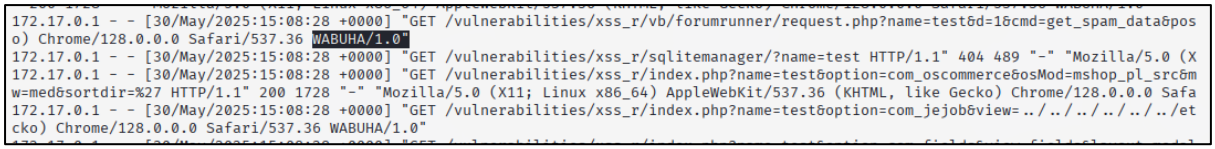


Figure 5.8: User-Agent String in HTTP Requests.

An important aspect of WABUHA's operation is its ability to customise the "User-Agent" string for its web requests. This means that when WABUHA sends requests to a website, it identifies itself with a specific signature. As seen in Figure 5.8 console output, WABUHA was configured to use "WABUHA/1.0" as its User-Agent. This demonstrates that the tool successfully adopts the string set by the user, which can be useful for various testing methodologies, including scenarios where a user might want to test the effectiveness of firewall whitelisting or blacklisting rules based on the inbound traffic from WABUHA.

5.4 Summary

In summary, the testing phase confirmed that WABUHA functions effectively and meets its design objectives. The software testing, particularly functional testing, demonstrated a 100% success rate across all 45 test cases, indicating strong system reliability and correct implementation of features.

Additionally, the user acceptance testing further reinforced these findings, with mostly positive responses from respondents regarding system usability, clarity of features, and the integration of AI-supported explanations. Minor areas for improvement were also identified. Overall, the test results validate WABUHA’s readiness for practical deployment.

Chapter 6: Conclusion and Future Works

This chapter provides an overview of the accomplishments for WABUHA, emphasising how the system meets its initial objectives. Along with discussing the limitations during development and testing, the proposed roadmap for next enhancements and possible collaborations.

6.1 Achievements

The development of WABUHA successfully addressed its core objectives, providing a robust and intuitive platform for beginners in web application penetration testing.

Firstly, the integration of popular reconnaissance and exploitation tools into an intuitive GUI greatly improved the bug hunting experience for beginners. As stated in Chapter 4, reconnaissance tools like Sublist3r, Subfinder, Nmap, and Katana, as well as exploitation tools like SQLmap, Nuclei, and Dalfox, were easily integrated. With the help of the PyQt5 framework, this integration converted the complex command-line processes into simple clicks actions into the interface. The usability and intuitiveness of the system were verified by the outcomes of UAT, where the feedback repeatedly found that the tool to is simple to use and comprehend.

Secondly, WABUHA successfully developed a structured and intuitive workflow for web application penetration testing, guiding beginners through each stage. The design of the system aligns with a simplified PTES methodology, presenting distinct phases through a clear tabbed interface. This progressive structure, as illustrated in Chapter 4 ensures users are guided step-by-step. Feedback from UAT, particularly regarding the structured tabs and guided workflow further assured that users felt more organised and less overwhelmed compared to traditional approach.

Lastly, WABUHA achieved its objective of integrating findings and tracking progress systematically across multiple stages of web application penetration testing, enhancing organisation and efficiency. The integration of SQLite database enables persistent storage of all reconnaissance results, exploitation findings, tool configurations, and system logs. The Report tab, detailed in Chapter 4, provides a clear summary of vulnerabilities categorised by severity (Critical, High, Medium, Low, Info, Unknown), allowing for overview and prioritisation. The Command Execution Log further enhances traceability by linking findings directly to the executed commands and offering Gemini AI-generated explanations. UAT responses highlighted the effectiveness categorising risks in the system, providing easy access to detailed logs and helping users stay organised throughout the assessment.

Additionally, the real-world testing on live and simulated targets, demonstrated WABUHA's practical feasibility in detecting a range of severity vulnerabilities, validating its capability to systematically track and report findings.

6.2 Limitations

Despite the achievements, WABUHA currently has several limitations that are opportunities for future development.

6.2.1 API Key Dependency

One notable limitation is the dependency on an external API key for the AI-generated vulnerability explanations. While this feature, powered by the Gemini Flash 2.0 model, significantly enhances the understanding of complex findings, requiring users to supply their own API key can be an issue. As highlighted by user feedback this dependency might require subscription for heavy usage, potentially limiting accessibility for some users who prefer a fully self-contained solution.

6.2.2 Limited Reporting Options

Currently, WABUHA's reporting capabilities are somewhat limited. While the system effectively summarises findings within the Report tab. User feedback indicated a desire for more versatile reporting formats, specifically the option to generate PDF. This limitation can affect the ease of sharing comprehensive, client-ready reports, which often require a formal and universally document format.

6.2.3 Toolset Customisation

Although WABUHA integrates a selection of popular reconnaissance and exploitation tools, the current level of toolset customisation is not extensive. Users expressed interest in the

inclusion of more integrated tools and the ability to create custom command templates. This suggests that while the existing tools are effective, advanced users or those with specific testing requirements might find the predefined toolset and command options restrictive, limiting the adaptability.

6.2.4 Session Management

The current session management within WABUHA primarily supports cookie-based session handling. While this covers a common authentication mechanism, it represents a limitation in handling more advanced session management techniques. The system has yet to support the use of Authorization Headers, custom headers, or mechanisms for token refresh. This restricts the applicability to web applications that rely on these more complex authentication methods, potentially impede comprehensive testing in modern environments.

6.2.5 Cross-Platform Limitations

A significant limitation of WABUHA is its current lack of cross-platform compatibility. The system is designed and tested exclusively on Kali Linux (Debian-based distributions). This is primarily due to its reliance on subprocess calls that assume a Unix-like shell and filesystem structure, as well as the specific functionalities of certain integrated tools that may not operate correctly or be accessible on other operating systems (non-Debian distribution). This restricts the user base and requires users to operate within a specific Linux environment.

6.3 Future Improvements and Collaborations

Building upon the current foundation, several key areas have been identified for future improvements and potential collaborations to enhance the capabilities of WABUHA and user experience.

6.3.1 Enhanced Integration with AI without External API Dependency

To address the current API key dependency, future efforts will focus on exploring alternative methods for integrating AI-powered explanations. This could involve investigating the use of locally runnable, lightweight AI models or leveraging open-source language models that do not require external API keys. The goal is to provide seamless, on-device AI support for vulnerability explanations, thereby removing the barrier of external API requirements and ensuring consistent accessibility for all users, regardless of their access to commercial AI services.

6.3.2 Improved Onboarding and User Assistance

Based on user feedback, enhancing the onboarding process and providing more comprehensive user assistance is a priority. This includes developing tutorial videos that guide new users through initial setup and key functionalities, as suggested by respondents. Additionally, implementing more extensive tooltips and contextual help within the interface, particularly for advanced options and technical terminology, would further improve usability and reduce learning curves. These improvements aim to make WABUHA even more approachable and self-sufficient for beginners.

6.3.3 Broadened Evaluation and Collaboration

To further validate WABUHA's effectiveness and expand its reach, future work will involve broadening its evaluation to include a wider range of users and scenarios. This could involve collaborating with educational institutions to integrate WABUHA into cybersecurity curricula, allowing students to gain hands-on experience in a controlled environment. Additionally, seeking feedback from experienced penetration testers could provide valuable insights for advanced feature development and refinement. Such collaborations would not only enhance the tool's practical utility but also foster a community-driven approach to its ongoing development.

References

0x1. (2020, January 10). *AttackSurfaceMapper*. 0x1 | Cyber Security Consulting; 0x1. https://0x1.gitlab.io/reconnaissance-tools/AttackSurfaceMapper/

Aboul-Ela, A. (2021, October 31). *aboul3la/Sublist3r*. GitHub. https://github.com/aboul3la/Sublist3r

Agile Alliance. (2001). *Manifesto for Agile software development*. Agile Manifesto. https://agilemanifesto.org/

Andrews, W., & Straub, J. (2021). Comparative analysis of interface usability for cybersecurity applications. *ProQuest Dissertations and Theses*, 54. https://www.proquest.com/dissertations-theses/comparative-analysis-interface-usability/docview/2572548409/se-2?accountid=40705

Alnajim, A. M., Habib, S., Islam, M., AlRawashdeh, H. S., & Wasim, M. (2023). Exploring cybersecurity education and training techniques: A comprehensive review of traditional, virtual reality, and augmented reality approaches. *Symmetry*, *15*(12), 2175. https://doi.org/10.3390/sym15122175

Astrida, D. N., Saputra, A. R., & Assaufi, A. I. (2022). Analysis and evaluation of wireless network security with the penetration testing execution standard (PTES). *Sinkron*, *7*(1), 147–154. https://doi.org/10.33395/sinkron.v7i1.11249

Chandel, R. (2019, September 29). *Web application pentest lab setup using Docker*. Hacking Articles. https://www.hackingarticles.in/web-application-pentest-lab-setup-using-docker/

Cisco. (2024). *What is threat modeling?* Cisco. https://www.cisco.com/c/en/us/products/security/what-is-threat-modeling.html

Cohn, M. (2009). *Succeeding with agile : Software development using Scrum*. Addison-Wesley. https://dl.acm.org/doi/abs/10.5555/1667109

D4RK-4RMY. (2023, January 17). *GitHub - D4RK-4RMY/DARKARMY: DARKARMY hacking tools pack - A penetration testing framework* . GitHub. https://github.com/D4RK-4RMY/DARKARMY

Dani. (2024, June 27). *Ethical considerations in penetration testing - Intrix Cyber Security*. Intrix Cyber Security. https://intrix.com.au/articles/ethical-considerations-in-penetration-testing/

European Union. (2016). *EUR-Lex - 32016R0679 - EN - EUR-Lex*. Europa.eu. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679

GeeksforGeeks. (2023a, June 8). *DARKARMY penetration testing tools*. GeeksforGeeks. https://www.geeksforgeeks.org/darkarmy-penetration-testing-tools/

GeeksforGeeks. (2023b, November 24). *Use Case diagram*. GeeksforGeeks. https://www.geeksforgeeks.org/use-case-diagram/

hahwul. (2023, March 11). *dalfox*. GitHub. https://github.com/hahwul/dalfox

Hoffman, A. (2024). *Web Application Security*. "O'Reilly Media, Inc." https://elib.vku.udn.vn/bitstream/123456789/3382/1/2020.%20Web%20Application%20Security.pdf

jaraco. (2024, December 25). *GitHub - jaraco/keyring*. GitHub. https://github.com/jaraco/keyring

Jeawan Naayagar Kanakasabai, Siti Hajar Othman, Maheyzah Md Siraj, Mohamad Hafiz Rahman, & Zaharudin, M. (2023). Google Dorking commands-based approach for assisting forensic investigators in gender identification of social media text data. *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, 466–471. https://doi.org/10.1109/icicyta60173.2023.10428736

Joshi, A., Kale, S., Chandel, S., & Pal, D. K. (2015). Likert Scale: Explored and explained. *British Journal of Applied Science & Technology*, *7*(4), 396–403. https://doi.org/10.9734/BJAST/2015/14975

Kali Linux. (2024, September 11). *Kali Linux 2024.3 release (multiple transitions)*. Kali Linux; Kali Linux 2024.3 Release (Multiple transitions) | Kali Linux Blog. https://www.kali.org/blog/kali-linux-2024-3-release/

Kidd, C. (2022, September 20). *Security 101: Vulnerabilities, threats & risk explained*. Splunk-Blogs. https://www.splunk.com/en_us/blog/learn/vulnerability-vs-threat-vs-risk.html

koutto. (2018a). *Command info — jok3r 2.0 documentation*. Readthedocs.io. https://jok3r.readthedocs.io/en/latest/command_info.html

koutto. (2018b). *What is Jok3r ? — jok3r 2.0 documentation*. Readthedocs.io. https://jok3r.readthedocs.io/en/latest/what.html

koutto. (2023, November 11). *Jok3r v3 beta*. GitHub. https://github.com/koutto/jok3r

Lachkov, P., Tawalbeh, L., & Bhatt, S. (2022). Vulnerability assessment for applications security through penetration simulation and testing. *Journal of Web Engineering*, *21*(7). https://doi.org/10.13052/jwe1540-9589.2178

Microsoft. (2024). *Visual Studio Code*. Visualstudio.com; Microsoft. https://code.visualstudio.com/

mitmproxy. (2025). *How mitmproxy works*. Mitmproxy.org. https://docs.mitmproxy.org/stable/concepts/how-mitmproxy-works/

MITRE. (2019a). *CVE - Common Vulnerabilities and Exposures (CVE)*. Mitre.org. https://cve.mitre.org/

MITRE. (2019b). *CWE - About CWE*. Mitre.org. https://cwe.mitre.org/about/index.html

Mossé Cyber Security Institute (MCSI). (2022). *The reconnaissance phase in penetration testing engagements*. Library.mosse-Institute.com. https://mcsi-library.readthedocs.io/articles/2022/03/reconnaissance-phase-in-penetration-testing-engagements/reconnaissance-phase-in-penetration-testing-engagements.html

Nancy Bou Ghannam, Salhab, N., & Maher Abdul Rahman. (2022). SQL injection, cross-site scripting and buffer overflow attacks detection using machine learning. *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, 292–296. https://doi.org/10.1109/icdabi56818.2022.10041495

Nash, A. J. (2023, May 25). *External threats vs. internal threats in cybersecurity*. ZeroFox. https://www.zerofox.com/blog/external-threats-vs-internal-threats-in-cybersecurity/

NIST. (2022, September 20). *NVD - vulnerability metrics*. Nist.gov. https://nvd.nist.gov/vuln-metrics/cvss

Nmap. (2024). *Nmap*. Nmap.org. https://nmap.org/

Oluwatosin Islamiyat Yusuf. (2024). Bridging the gap: Aligning cybersecurity education with industry needs. *International Journal of Information Technology and Computer Engineering*, *4*(43), 1–8. https://doi.org/10.55529/ijitc.43.1.8

OWASP. (n.d.). *Vulnerabilities | OWASP*. Owasp.org. Retrieved January 4, 2025, from https://owasp.org/www-community/vulnerabilities/

OWASP. (2018). *WSTG - Stable | OWASP Foundation*. Owasp.org. https://owasp.org/www-project-web-security-testing-guide/stable/2-Introduction/README

OWASP. (2023). *Web Security Testing Guide (WSTG) - Latest | OWASP*. Owasp.org. https://owasp.org/www-project-web-security-testing-guide/latest/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies

OWASP. (2024). *OWASP Top Ten*. Owasp.org; OWASP. https://owasp.org/www-project-top-ten/

Pejabat Pesuruhjaya Perlindungan Data Peribadi Malaysia. (2024). *Perlindungan Data Peribadi – PDP*. Pdp.gov.my. https://www.pdp.gov.my/ppdpv1/

Priy, S. (2018, April 10). *Introduction to SQLite*. GeeksforGeeks. https://www.geeksforgeeks.org/introduction-to-sqlite/

projectdiscovery. (2021). *GitHub - projectdiscovery/katana: A next-generation crawling and spidering framework.* GitHub. https://github.com/projectdiscovery/katana

projectdiscovery. (2022, May 22). *projectdiscovery/nuclei*. GitHub. https://github.com/projectdiscovery/nuclei

projectdiscovery. (2023, January 13). *projectdiscovery/subfinder*. GitHub. https://github.com/projectdiscovery/subfinder

Raghu, A. (2023, July 5). *Trustwave Blog | Trustwave*. Trustwave.com; Trustwave Holdings, Inc. https://doi.org/102279083/1729705714101/module_128102279083_Global-Header

Riverbank Computing Limited. (2025, May 19). *PyQt5 tutorial 2022, create Python GUIs with Qt*. Python GUIs. https://www.pythonguis.com/pyqt5-tutorial/

Sandler, C., Badgett, T., & Myers, G. J. (2013). *The art of software testing* (3rd ed.). Wiley.

Seidl, D., & Chapple, M. (2022). *Planning and scoping penetration tests* (pp. 31–57). Wiley Data and Cybersecurity. https://doi.org/10.1002/9781394177653

Shah, M., Ahmed, S., Saeed, K., Junaid, M., Khan, H., & Ata-ur-rehman. (2019, March). Penetration testing active reconnaissance phase – optimized port scanning with Nmap tool. *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (ICoMET)*. International Conference on Computing, Mathematics and Engineering Technologies ICOMET2019. https://doi.org/10.1109/icomet.2019.8673520

Singh, S. (2023, August 10). *The role of penetration testing in ethical hacking: Identifying vulnerabilities and risks - E&ICT Academy, IIT Kanpur*. E&ICT Academy, IIT Kanpur - E&ICT Academy, IIT Kanpur. https://eicta.iitk.ac.in/knowledge-hub/ethical-hacking/the-role-of-penetration-testing-in-ethical-hacking-identifying-vulnerabilities-and-risks/

Sirangi, A. (2025). LLMs in personalized education: Adaptive learning models. *Journal of Information Systems Engineering and Management*, *10*(47s), 706–715. https://doi.org/10.52783/jisem.v10i47s.9333

sqlmapproject. (2019, September 26). *sqlmapproject/sqlmap*. GitHub. https://github.com/sqlmapproject/sqlmap

superhedgy. (2019). GitHub - superhedgy/AttackSurfaceMapper: AttackSurfaceMapper is a tool that aims to automate the reconnaissance process. *GitHub*. https://doi.org/1067612/0837e700-ba42-11e9-9a7f-e46c4b6b2411

Svabensky, V., Vykopal, J., Tovarnak, D., & Celeda, P. (2021). Toolset for collecting shell commands and its application in hands-on cybersecurity training. *2021 IEEE Frontiers in Education Conference (FIE)*. https://doi.org/10.1109/fie49875.2021.9637052

Talabis, M., & Martin, J. (2013). *Risk assessment framework - An overview | ScienceDirect topics*. Www.sciencedirect.com. https://www.sciencedirect.com/topics/computer-science/risk-assessment-framework

Teichmann, F., & Boticiu, S. (2023). An overview of the benefits, challenges, and legal aspects of penetration testing and red teaming. *International Cybersecurity Law Review*, *4*(4), 1–11. https://doi.org/10.1365/s43439-023-00100-2

Tuan, Y. (2025, June 12). *How to overcome cybersecurity skills gaps in the digital age*. Netcomlearning.com. https://www.netcomlearning.com/blog/How-to-Bridge-the-Cybersecurity-Skills-Gap-in-Today-Threat-Landscape

U.S. Department of Justice. (2015, February 19). *Computer Fraud and Abuse Act*. Www.justice.gov. https://www.justice.gov/jm/jm-9-48000-computer-fraud

Yin, Y., Shao, Y., Wang, X., & Su, Q. (2019). A flexible cyber security experimentation platform architecture based on Docker. *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 413–420. https://doi.org/10.1109/qrs-c.2019.00082

Appendices

Appendix A: Survey and Questionnaire

1. This section collects basic demographic information to help understand the background of the respondents.
    - Student
    - Aspiring Penetration Tester
    - Experienced Penetration Tester
    - IT Professional
    - Other
2. Do you have prior experience with penetration testing?
    - Yes
    - No
3. How many years of experience do you have in penetration testing or cybersecurity?
    - None
    - Less than 1 year
    - 1 - 3 years
    - 3 - 5 years
    - More than 5 years
4. What is your age group?
    - Under 18
    - 18 - 24
    - 25 - 34
    - 35 - 44
    - 45 and above

5. How would you rate your technical skill level in context of web application penetration testing?
    - Beginner
    - Intermediate
    - Advanced
6. How often do you perform penetration testing?
    - Never
    - Occasionally
    - Regularly
7. Existing penetration testing tools are easy to use for beginners.
    - 5 - Strongly Agree
    - 4 - Agree
    - 3 - Neutral
    - 2 - Disagree
    - 1 - Strongly Disagree
8. Rate the complexity of using console-based penetration testing tools.
    - 1 - Very Easy
    - 2 - Easy
    - 3 - Moderate
    - 4 - Difficult
    - 5 - Very Complex
9. What are the biggest challenges you face when using penetration testing tools?

10. Which features do you find most important in a penetration testing tool? (Select all that apply)
    - User-friendly interface
    - Comprehensive vulnerability database integration
    - Guided workflows for beginners
    - Ability to track and organise findings
    - Customisable tool configurations
    - Support for Docker environments
11. A graphical user interface would make penetration testing more accessible to beginners.
    - 5 - Strongly Agree
    - 4 - Agree
    - 3 - Neutral
    - 2 - Disagree
    - 1 - Strongly Disagree
12. How important is it for a penetration testing tool to integrate Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) database lookups?
    - 1 - Not Important
    - 2 - Slightly Important
    - 3 - Moderately Important
    - 4 - Very Important
    - 5 - Extremely Important
13. Would you prefer a tool that automates the penetration testing process?
    - Yes
    - No

14. If you could design the ideal penetration testing tool, what features would you include?
15. What output formats are most useful for exporting findings?
    - Excel
    - PDF
    - JSON
    - CSV
    - Other
16. Should the tool provide a risk assessment summary (e.g., High, Medium, Low)?
    - Yes
    - No
17. Integration with databases like Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE) makes a penetration testing tool more effective?
    - 5 - Strongly Agree
    - 4 - Agree
    - 3 - Neutral
    - 2 - Disagree
    - 1 - Strongly Disagree
18. How useful would it be for the tool to provide predefined attack templates for common vulnerabilities?
    - 5 - Extremely Useful
    - 4 - Very Useful
    - 3 - Moderately Useful
    - 2 - Slightly Useful
    - 1 - Not Useful

19. Tutorials and hands-on guidance are essential in learning penetration testing tools.
    - 5 - Strongly Agree
    - 4 - Agree
    - 3 - Neutral
    - 2 - Disagree
    - 1 - Strongly Disagree
20. How confident are you in using current penetration testing tools without prior training?
    - 5 - Very Confident
    - 4 - Confident
    - 3 - Somewhat Confident
    - 2 - Slightly Confident
    - 1 - Not Confident
21. How do you prefer to learn about new penetration testing tools?
    - Step-by-step tutorials
    - Video demonstrations
    - Interactive guides within the tool
    - Written documentation
    - Community forums
    - Other
22. How likely are you to use a tool like WABUHA if it meets your expectations?
    - 5 - Extremely Likely
    - 4 - Very Likely
    - 3 - Moderately Likely
    - 2 - Slightly Likely
    - 1 - Not Likely

Appendix B: Legal and Security Disclaimer

IMPORTANT: PLEASE READ CAREFULLY

By using WABUHA (Web App Bug Hunting Assistant), you agree to the following terms and conditions. This software is intended for ethical penetration testing and educational purposes only. Unauthorized use of this software is illegal and can result in severe legal consequences.

1. Authorized Use Only

WABUHA is a tool designed to help beginners learn about penetration testing by providing a guided and controlled environment. You must have explicit permission from the owner of the target website or system before conducting any testing. Using WABUHA on systems you do not own or have explicit permission to test is against the law.

Do not use WABUHA to scan, exploit, or attack systems without written consent from the system owner.

Only test on systems where you have received proper authorization or in controlled, isolated environments like Docker containers or test labs.

2. Personal Responsibility

By using this software, you take full responsibility for your actions. The developers are not responsible for any legal actions, damages, or consequences resulting from your use of WABUHA.

If you use WABUHA for unauthorized activities, you could face criminal charges, loss of data, or financial penalties.

Always respect privacy: Do not collect, modify, or disclose personal or sensitive data without permission.

3. Educational Purpose

WABUHA is designed to help beginners understand penetration testing techniques in a safe and educational environment. The tool provides a guided workflow, but it is your responsibility to ensure that you are testing legally and ethically.

You should only use WABUHA in controlled environments like your personal lab, a designated testing network, or with explicit permission from the owner of the website/domain.

The tool includes built-in features for tracking and managing findings to help guide you through the testing process. However, you should never perform tests on real-world systems without authorization.

4. Security Risks

Penetration testing can sometimes lead to unintended consequences, including data loss, system crashes, or unintentional exposure of vulnerabilities. You use WABUHA at your own risk.

Security Testing Is Risky: You may inadvertently cause damage or interruption to services during testing. Ensure that you are only testing on systems that you own or have been authorized to test.

Do not use WABUHA on production systems unless explicitly authorized. It is recommended to use the software in a safe, isolated environment.

5. No Liability

The developers of WABUHA are not responsible for any damages or issues arising from the use of this software. This includes:

- Data loss or corruption
- Damage to systems or applications
- Unauthorized access to systems or data

By using this software, you agree to hold the developers harmless from any and all claims, losses, or damages.

6. Compliance with Laws

You agree to comply with all applicable laws and regulations, including but not limited to:

Cybercrime laws: Unauthorized access to computer systems and networks is illegal in most countries.

Data protection laws: Ensure that you do not violate privacy laws by accessing or disclosing personal data without permission.

Penetration testing laws: Many countries require written consent to perform security testing on systems that are not your own.

7. Usage Limits

While WABUHA can help you identify vulnerabilities, it should never be used for malicious purposes. The tools provided are for learning, research, and educational purposes only.

Test ethically: If you discover vulnerabilities, report them responsibly to the system owner.

Do not share or exploit vulnerabilities discovered using WABUHA without permission.

8. Security of WABUHA

We recommend you keep WABUHA updated to ensure that you have the latest features and security fixes. However, the tool does not guarantee to find all vulnerabilities or work on

every system. Use it as a learning tool, and supplement it with additional resources and research.

9. Consent

By clicking Agree, you confirm that you understand and accept the terms of this disclaimer. If you do not agree to these terms, you must exit the application immediately and refrain from using WABUHA.

Appendix C: Functional Test Cases

Table C.1: Test Cases for Disclaimer Notice Module.

Module:	Disclaimer Notice						
Description:	Verify disclaimer acceptance functionality						
Objective:	Ensure application proceeds only upon user acceptance of disclaimer						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-DIS-001	User accepts disclaimer	1. Launch WABUHA application 2. Click "Yes, I Agree" on disclaimer popup	Positive: Click "Yes, I Agree"	Application proceeds to Home tab	Same as expected result	Pass	High
TC-DIS-002	User declines disclaimer	1. Launch WABUHA application 2. Click "No" on disclaimer popup	Negative: Click "No"	Application terminates immediately	Same as expected result	Pass	Critical
TC-DIS-003	User closes disclaimer without action	1. Launch WABUHA application 2. Close disclaimer popup without clicking "Yes, I Agree" or "No"	Negative: Close popup	Application terminates immediately	Same as expected result	Pass	High

Table C.2: Test Cases for Home Tab Module.

Module:	Home Tab						
Description:	Verify "New Hunt" and "Import Hunt" features						
Objective:	Ensure users can create, update and import hunts successfully						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-HOME-001	Create a new hunt with valid data	1. Click "New Hunt" 2. Fill in all mandatory fields 3. Select valid start and end dates 4. Click "Save"	Positive: Valid hunt name, target domain, start date, end date, description	Hunt is created successfully; Config and Log tabs become accessible	Same as expected result	Pass	High
TC-HOME-002	Create a new hunt with missing data	1. Click "New Hunt" 2. Leave mandatory fields empty 3. Click "Create"	Negative: Missing hunt name, target domain or dates	Error message displayed; hunt not created	Same as expected result	Pass	Medium
TC-HOME-003	Import existing hunt	1. Click "Import Hunt" 2. Select a valid hunt folder 3. Click "Import"	Positive: Valid hunt folder	Hunt is imported successfully; Config and Log tabs become accessible	Same as expected result	Pass	High
TC-HOME-004	Import invalid hunt folder	1. Click "Import Hunt" 2. Select an invalid or corrupted hunt folder 3. Click "Import"	Negative: Invalid hunt folder (missing wabuha.db or wabuha.log)	Error message displayed; hunt not imported or hunt data corrupted	Same as expected result	Pass	Medium
TC-HOME-005	Modify hunt detail	1. Modify the hunt name, dates or description 2. Click "Save Changes"	Positive: Valid hunt name, dates and description	Hunt details are updated successfully	Same as expected result	Pass	Medium
TC-HOME-006	Modify invalid hunt detail	1. Modify the hunt name, dates or description 2. Click "Save Changes"	Negative: Empty hunt name	Error message display: Invalid formatting	Same as expected result	Pass	Medium

Table C.3: Test Cases for Configuration Tab - Scope Module.

Module:	Configuration Tab - Scope						
Description:	Verify scope inclusion and exclusion functionality						
Objective:	Ensure users can define in-scope and out-of-scope targets accurately						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-CONF-001	Add known subdomains manually	1. Navigate to Config > Scope 2. Enter known subdomains in the provided field 3. Click "Save"	Positive: Valid subdomain entries	Subdomains are saved successfully	Same as expected result	Pass	Medium
TC-CONF-002	Include targets in scope using regex	1. Navigate to Config > Scope 2. Enter regex patterns in "Include in Scope" field 3. Click "Save"	Positive: Valid regex patterns	Patterns are saved; targets matching regex are included in scope	Same as expected result	Pass	High
TC-CONF-003	Exclude targets from scope using regex	1. Navigate to Config > Scope 2. Enter regex patterns in "Exclude from Scope" field 3. Click "Save"	Positive: Valid regex patterns	Patterns are saved; targets matching regex are excluded from scope	Same as expected result	Pass	High
TC-CONF-004	Enter invalid regex patterns	1. Navigate to Config > Scope 2. Enter invalid regex patterns in "Include" or "Exclude" fields 3. Click "Save"	Negative: Invalid regex patterns	Error message displayed; patterns not saved	Same as expected result	Pass	Medium

Table C.4: Test Cases for Configuration Tab - Tool Selection Module.

Module:	Configuration Tab - Tool Selection						
Description:	Verify tool selection and saving functionality						
Objective:	Ensure selected tools are saved and used in subsequent phases						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-CONF-005	Select tools for recon and exploitation	1. Navigate to Config > Tool Selection 2. Tick checkboxes for desired tools 3. Click "Save"	Positive: Valid tool selection	Tools saved and available in Recon and Exploitation tabs	Same as expected result	Pass	High
TC-CONF-006	Save with no tools selected	1. Navigate to Config > Tool Selection 2. Deselect all tools 3. Click "Save"	Negative: No tools selected	Error or warning shown; cannot proceed without selecting tools	Same as expected result	Pass	Medium

Table C.5: Test Cases for Configuration Tab - Session Module.

Module:	Configuration Tab - Session						
Description:	Verify session creation and handling						
Objective:	Ensure session headers/cookies are properly configured and stored						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-CONF-007	Add session headers and cookies	1. Navigate to Config > Session 2. Enter valid HTTP headers and cookies 3. Click "Save	Positive: Valid header/cookie key-value pairs	Session saved successfully	Same as expected result	Pass	High
TC-CONF-008	Enter invalid header format	1. Enter malformed headers or cookies 2. Click "Save"	Negative: Leave empty	Error shown; session not saved	Same as expected result	Pass	Medium
TC-CONF-009	Use saved session in Recon	1. Configure session 2. Navigate to Recon 3. Start recon with session applied	Positive: Proper session attached	Session headers/cookies attached to recon requests	Same as expected result	Pass	High

Table C.6: Test Cases for Configuration Tab - More Module.

Module:	Configuration Tab - More						
Description:	Verify User-Agent string configuration						
Objective:	Ensure custom User-Agent strings are saved and used during requests						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-CONF-010	Select predefined user-agent string	1. Navigate to Config > More 2. Select a predefined user-agent string 3. Click "Save"	Positive: Chrome, Firefox, etc.	User-agent string saved; visible in requests	Same as expected result	Pass	Medium
TC-CONF-011	Enter custom user-agent string	1. Enter custom user-agent string 2. Click "Save"	Positive: Custom string like "WABUHA/1.0"	User-agent saved and applied to future requests	Same as expected result	Pass	High
TC-CONF-012	Leave user-agent field empty	1. Leave the field blank 2. Click "Save"	Negative: Empty value	Warning shown; user-agent not saved	Same as expected result	Pass	Low

Table C.7: Test Cases for Reconnaissance Tab - Automate Module.

Module:	Reconnaissance Tab - Automate						
Description:	Verify automated reconnaissance tools functionality						
Objective:	Ensure selected tools execute properly and display results						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-RECON-001	Run selected recon tools	1. Navigate to Recon > Automate 2. Select desired tools 3. Click "Start Recon"	Positive: Valid tool selection	Tools execute; results displayed in respective sections	Same as expected result	Pass	High
TC-RECON-002	Run recon without selecting tools	1. Navigate to Recon > Automate 2. Click "Start Recon" without selecting any tools	Negative: No tools selected	Error message displayed; recon not initiated	Same as expected result	Pass	Medium
TC-RECON-003	Interrupt recon process	1. Start recon process 2. Click "Stop" before completion	Positive: User interrupts process	Recon process stops; partial results displayed	Same as expected result	Pass	Low

Table C.8: Test Cases for Reconnaissance Tab - Proxy Module.

Module:	Reconnaissance Tab - Proxy						
Description:	Verify proxy server configuration and interception						
Objective:	Ensure user can enable proxy, capture traffic, and use browser with WABUHA proxy						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-RECON-004	Start proxy and intercept traffic	1. Navigate to Recon > Proxy 2. Click "Start Proxy" 3. Visit site using browser configured with WABUHA proxy	Positive: Valid proxy setup	HTTP requests appear in proxy capture panel	Same as expected result	Pass	High
TC-RECON-005	Stop proxy after use	1. Stop the interception.	N/A	Proxy stops cleanly and releases port	Same as expected result	Pass	Medium
TC-RECON-006	Start proxy on occupied port	1. Set proxy port to a value already used (e.g., 8080 if already occupied) 2. Click "Start Proxy"	Negative: Port already in use	Error shown: “Port in use”	Same as expected result	Pass	Critical
TC-RECON-007	Launch Chromium browser	1. Navigate to Recon > Proxy 2. Click "Open Browser"	Positive: Click "Open Browser"	A Chromium browser will pop up with proxy configuration ready.	Same as expected result	Pass	Medium
TC-RECON-008	Close the opened Chromium browser	1. Close the browser	N/A	The browser will close and CA cert uninstalled.	Same as expected result	Pass	Low

Table C.9: Test Cases for Exploitation Tab - Automate Module.

Module:	Exploitation Tab - Automate						
Description:	Verify exploitation tools execution						
Objective:	Ensure selected exploitation tools run and identify vulnerabilities						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-EXP-001	Run SQLi tool on target URL	1. Navigate to Exploitation tab 2. Select SQLi tool 3. Choose target URL 4. Click "Start Exploit"	Positive: Valid target URL with SQLi vulnerability	Tool executes; SQLi vulnerability identified and reported	Same as expected result	Pass	Critical
TC-EXP-002	Run XSS tool on target URL	1. Navigate to Exploitation tab 2. Select XSS tool 3. Choose target URL 4. Click "Start Exploit"	Positive: Valid target URL with XSS vulnerability	Tool executes; XSS vulnerability identified and reported	Same as expected result	Pass	High
TC-EXP-003	Run exploit without selecting tools	1. Navigate to Exploitation tab 2. Click "Start Exploit" without selecting any tools	Negative: No tools selected	Warning message displayed; exploitation not initiated	Same as expected result	Pass	Medium

Table C.10: Test Cases for Report Tab Module.

Module:	Report Tab						
Description:	Verify vulnerability summary and AI explanation						
Objective:	Ensure vulnerability summary are generated correctly						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-REP-001	View summary of the vulnerabilities found	**Pre-requisite: TC-EXP-001** 1. Navigate to the Report tab 2. Expand the tree table (each risk level)	Positive: Query vulnerabilities found from the database	Risk count banner is accurately displayed and the tree table widget is populated with vulnerability entries	Same as expected result	Pass	High
TC-REP-002	View command execution log from report tab entry	1. Navigate to Report tab 2. Double-click on a risk entry 3. View command execution log and explanation	Positive: Query existing log data	Log window displays: command, tool, output, and AI-generated explanation	Same as expected result	Pass	High
TC-REP-003	Invalid or corrupted entry handling	1. Navigate to Report tab 2. Double-click corrupted or unlinked log entry	Negative: Corrupted or missing reference	Error message: “Log entry unavailable or corrupted”	Same as expected result	Pass	Medium

Table C.11: Test Cases for Command Execution Log Viewer Module.

Module:	Command Execution Log Viewer						
Description:	Validate display, format, and AI explanation						
Objective:	Ensure clean presentation and functionality of the log + AI section						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-CEL-001	Show command, tool used, and raw output	1. Open command log 2. Verify UI sections: Tool Name, Tool Category, Command executed, Console Output	Positive: Sample command executed	All fields are populated properly; Output is scrollable, readable	Same as expected result	Pass	High
TC-CEL-002	Generate AI explanation	1. Open a log with AI support 2. Click "Ask AI" and wait for response 3. Review explanation	Positive: Utilise the risk detail found (PoC), tool category and tool name into a pre-build prompt	Loading is indicated, explanation appears contextualized with potential impact, fix guidance, and reference links	Same as expected result	Pass	High
TC-CEL-004	Resize and close modal window	1. Try resizing the modal 2. Click "Close" button	N/A	Window behaves normally; No crash or UI break	Same as expected result	Pass	Low
TC-CEL-005	AI Explanation error handling	1. Simulate AI error (e.g., no context, or LLM offline) 2. Try generating explanation	Negative: No explanation generated	Message: “AI explanation failed to load. Please try again later.”	Same as expected result	Pass	Medium
TC-CEL-006	Prompt when AI API key is missing	1. Launch Command Execution Viewer 2. Ensure no key is stored in keyring 3. Click “Generate Explanation”	No API key	Prompt appears asking user to input API key or open settings via cog icon	Same as expected result	Pass	High

TC-CEL-007	Update AI API key from settings icon	1. Click on settings (cog) icon 2. Input a valid OpenAI API key 3. Save key 4. Retry “Generate Explanation”	New valid key	Key is securely saved using "keyring"; AI explanation is regenerated without further prompts	Same as expected result	Pass	High
TC-CEL-008	Enter invalid API key	1. Enter a clearly malformed or fake key	Invalid key	App displays error message like “Invalid API key format” or “Authentication failed”	Same as expected result	Pass	Medium
TC-CEL-009	Auto-load stored API key from keyring	1. Ensure a valid API key is stored in keyring 2. Open Command Execution Viewer 3. Click “Generate Explanation”	Valid stored key	AI explanation loads directly without user interaction	Same as expected result	Pass	High
TC-CEL-010	Remove AI API key via settings	1. Click settings icon 2. Click “Remove API Key” button 3. Confirm action 4. Attempt “Generate Explanation” again	N/A	Key is deleted from keyring; next attempt prompts user for a new key	Same as expected result	Pass	High
TC-CEL-011	Cancel key prompt	1. Trigger API key prompt 2. Click “Cancel” without entering anything	N/A	Action is cancelled; AI explanation is skipped with a warning “No API key available”	Same as expected result	Pass	Low

Table C.12: Test Cases for Log Tab Module.

Module:	Log Tab						
Description:	Verify system activities and verbosity functionality						
Objective:	Ensure clean presentation and verbosity control.						
Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Status**	**Severity**
TC-LOG-001	Show the system activities	1. Navigate to the Log tab.	Positive: Navigate around the system and perform some actions to trigger log	System activity is logged accurately and informative.	Same as expected result	Pass	Medium
TC-LOG-002	Change the verbosity level	1. Navigate to the Log tab. 2. Select different log verbosity	Positive: Select different verbosity	The log table will reload and show the corresponding log based on verbosity (1-Error, 2-Error and Warning, 3-Error, Warning and Info)	Same as expected result	Pass	Low

Appendix D: User Acceptance Test Questionnaires

Table D.1: Summary of User Acceptance Testing Data Collected via Likert Scale.

ID	Questionnaires	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Configuration Phase						
Q1	The process of creating a new hunt with a custom name and log level was straightforward.	0	0	0	0	15
Q2	I found it easy to define the testing scope (domains, IPs, ports, etc.).	0	0	1	1	13
Q3	Selecting tools from the configuration interface was intuitive and informative.	0	0	0	0	15
Q4	Managing session-related settings (cookies) was simple and clear.	0	2	1	6	6
Q5	Managing user agent strings was simple and clear.	0	0	0	2	13
Reconnaissance Phase						
Q6	Starting reconnaissance and viewing tool-specific outputs was easy to understand.	0	0	1	0	14
Q7	The proxy configuration option was helpful for monitoring requests.	0	0	0	0	15
Q8	The information collected during reconnaissance was relevant and well-presented.	0	0	0	2	13
Q9	The tool provided adequate feedback and status updates during recon tasks.	0	1	1	0	13
Findings Management						
Q10	Filtering and managing findings from reconnaissance was easy and flexible.	0	0	0	3	12
Q11	I was able to exclude findings from the exploitation phase with minimal effort.	0	1	0	0	14
Exploitation Phase						
Q12	The process of starting exploitation and executing tools was smooth and reliable.	0	0	1	0	14
Q13	Mapping findings to CWE data was useful and worked as expected.	0	0	0	0	15
Q14	Viewing vulnerability details in the Exploit tab was informative and accessible.	0	0	2	3	10
Q15	The system clearly categorized risks (High, Medium, Low, Info) for better understanding.	0	0	0	0	15
Command Execution Log (AI Integration)						
Q16	Double-clicking findings to view detailed execution logs was intuitive.	0	0	0	0	15
Q17	The command execution log provided meaningful technical information.	0	0	1	1	13
Q18	The AI-generated explanation helped me understand the vulnerability better.	0	0	0	0	15

Q19	The system prompted me appropriately when my AI API key was missing or invalid.	0	0	0	2	13
Q20	Managing (add/update/delete) the AI API key was simple and user-friendly.	0	0	0	0	15
Reporting and Summary						
Q21	The report view was clear and summarized all findings effectively.	0	0	1	5	9
Q22	Overall, the interface helped me stay organized during the assessment workflow.	0	0	0	0	15
Overall Satisfaction						
Q23	WABUHA made web app reconnaissance and exploitation easier to understand.	0	0	1	2	12
Q24	I would recommend WABUHA to beginners interested in penetration testing.	0	0	0	1	14