National College of Ireland

Consolidating reconnaissance tools and techniques for penetration testing under a single platform

MSc Research Project
Cyber Security

Mayuresh Gadekar
Student ID: x21179280

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mayuresh Rajesh Gadekar
Student ID:	x21179280
Programme:	MSc Cyber Security 2022-23 **Year:**
Module:	MSc Research Project/Internship
Supervisor:	Imran Khan
Submission Due Date:	18th September 2023
Project Title:	Consolidating reconnaissance tools and techniques for penetration testing under a single platform
Word Count:	5658 **Page Count** 19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Mayuresh Gadekar

Date: 15th September 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Consolidating reconnaissance tools and techniques for penetration testing under a single platform

Mayuresh Gadekar
x21179280
Master of Science in Cyber Security
National college of ireland

Abstract

The reconnaissance stage of penetration testing could be greatly enhanced by integrating several reconnaissance tools and techniques into a single platform. This may also result in a target system's potential vulnerabilities being identified with more accuracy and efficiency. Penetration testers can concentrate on other areas of the testing process and potentially find more complicated security flaws by automating and expediting the reconnaissance process. The major top-level domain of the organization is used as the input for the recon framework in this study, which then does the recon and displays the findings in the relevant directories.

Keywords: Reconnaissance, Penetration testing, vulnerability

1 Introduction

Penetration testing is a crucial practice in cybersecurity, serving as a proactive measure to identify and mitigate security vulnerabilities within a company's online resources. A fundamental stage of this process is reconnaissance, where the tester collects valuable information about the target system before initiating the security assessment. Reconnaissance techniques can be broadly categorized into active and passive approaches. While active reconnaissance involves direct interaction with the target system, passive reconnaissance avoids such interactions and relies on publicly available information. The research focuses on methods of passive reconnaissance, which offer the advantage of being stealthier and reducing the risk of detection during the information-gathering phase. The use of various reconnaissance tools and techniques, such as port scanner, Google Dorking, subdomain enumeration, content discovery, vulnerability scanning, GitHub Dorking, and JavaScript enumeration, is common in this stage. However, employing multiple tools individually can be cumbersome, time-consuming, and may lead to inefficiencies in the reconnaissance process.

The motivation behind this research stems from the need to streamline the reconnaissance phase of penetration testing by consolidating multiple tools and techniques under a single platform. This would alleviate the challenges posed by using disparate tools and enhance the overall efficiency and effectiveness of the reconnaissance process.

Hypothesis:

- The integration of various passive reconnaissance techniques into a single platform will significantly improve the efficiency and effectiveness of the information-gathering phase during penetration testing.

- The unified approach to passive reconnaissance will reduce the risk of detection during the reconnaissance process, making it stealthier compared to employing multiple tools individually.
- By using the proposed integrated platform, the identification of potential vulnerabilities in target systems will be more accurate and comprehensive compared to traditional, tool-centric reconnaissance methods.

Research Question: To what extent would integrating multiple pentest tools and techniques into a single platform improve the process of reconnaissance in penetration testing?

The paper addresses the challenge of employing various reconnaissance tools individually by proposing a unified approach that integrates multiple techniques, including port scanning, Google Dorking, subdomain enumeration, Vulnerability scanning, content discovery, GitHub Dorking, and JavaScript enumeration. This integrated platform offers a more streamlined and efficient reconnaissance process, enhancing the overall effectiveness of penetration testing by saving time, reducing complexity, and improving the accuracy of identifying potential vulnerabilities in target systems.

The Structure of the Document:
In the introduction section, the paper introduces the subject matter, providing a motivational context for its study supported by pertinent citations. The research questions, objectives, and potential hypotheses are outlined, along with a concise overview of the contribution this work offers to scientific literature. This section concludes by providing an outline of the report's structure. The subsequent literature review presents an analysis of existing reconnaissance tools, identifying their limitations and gaps in their consolidation. The methodology section elaborates on the proposed Recon Framework, including its description, implementation particulars, and design considerations. Moving to the evaluation stage, the paper outlines real-world testing scenarios and data collection methodologies, followed by a performance comparison with conventional reconnaissance methods. The discussion section critically examines findings and underscores the advantages of the Recon Framework, offering insights into the integration of tools and techniques for effective reconnaissance. The conclusion section summarizes the research's contributions and implications, also delving into potential future directions and areas for improvement.

2 Related Work

2.1 Various reconnaissance methods and the importance of the reconnaissance phase

There is a lack of an accessible tool that can help in the reconnaissance stage of such penetration testing, claims Ahana Roy (Roy, et al., 2017). They created a tool that collects a company's digital footprint, which is helpful during the information-gathering stage of a penetration test. In (Becker, et al., 2017), Kristian Beckers provides details of a survey conducted on technologies available in 2017 for social engineering and intelligence gathering. It gives a summary of their features and capabilities. Usman Ali Dar analyzes numerous recon techniques that an attacker or hacker could use to obtain information on the target in (Dar & Iqbal, 2018). It picks the techniques that gather the most information about the target online while remaining anonymous.

Dr. Arun Kumar claims in (Kumar & Arora, 2018) that web apps are becoming increasingly common and a top target for security assaults. This is because of the increasingly frequent usage of them for intricate services. Even though several techniques have been developed to safeguard online applications and stop attacks against them, very little effort has been made to contrast these techniques and develop a more thorough understanding of web application security research. An in-depth examination of attacks against key parameters is provided in this study. Susanne Young (Young, 2014) discusses how third-party data breaches cost substantially more per lost record than local data breaches do.

Arjun Guha et al. (Guha, et al., n.d.) state that to perform static analysis, JavaScript files must be gathered, made readable, and security flaws must be found. Mayur Parmar uses a search string with complex parameters in (Parmar, 2019) to unearth concealed data. It assists with tasks like network mapping, port scanning, data gathering, and bug bounty programs. When looking for weaknesses, Rizdqi Akbar Ramadan et al. (Ramada & Maland, 2019) stress the value of doing thorough reconnaissance. Techniques for subdomain enumeration are also described here. In (Mundalik, 2015), Suraj S. Mundalik et al. examine information collecting strategies and the attack simulation implementation that is carried out specifically with Kali Linux OS utilizing a variety of penetration testing approaches.

2.2 Vulnerabilities that can be exploited due to reconnaissance.

The Domain Name System has been a crucial component of both web services and cyber security, according to S.M. Zia Ur Rashid in (Rashid, 2019). The security and performance of the nameserver's domain names are solely their responsibility. However, because sample security and DNS misconfiguration, there may be a potential for external services to take control of the subdomain.

According to Andres Ojamaa (Ojamma & Duuna, 2012), it is simple to detect logs and issues in server-side programs, but it is more difficult when using client-side web apps. The use of Angular, React, etc. in the client-side app's front end points the way for security flaws.

Jaspher Kathrine et al.'s article (Kathrine, 2020) describes the patterns that are essential to subdomain enumeration. Since developers frequently utilize patterns like QA, dev, staging, API, and UAT when naming subdomains, enumerating subdomains can be accomplished simply by brute forcing these patterns.

Comparison and contrast:

Survey Paper	Publication Year	Key Problem	Subdomain Enumeration	Port Scanning (Nmap)	Content Discovery	OWASP top 10 Vulnerabilities (Nuclei templates)	JavaScript Enumeration	Github Dorking	Google Dorking
Ahana Royetal	2017	Presents a tool which gather the footprint of a corporation, helpful for information gathering phase during a penetration test	Y	Y	N	N	N	N	Y
Kristian Beckers	2017	Describes the details of a survey done on tools in 2017 which are there for social engineering and intelligence gathering	Y	Y	N	N	Y	N	N
Usman Ali Dar	2018	Explores different kinds of recon techniques that are used by an attacker or hacker to collect information regarding the target	Y	Y	N	N	N	N	N
Dr Arun Kumar	2018	explains that as Web Applications are increasingly used for complex services, they become a popular and great target for security attacks.	Y	Y	Y	N	N	N	N
Susanne Young	2014	Discusses how third-party data breaches cost significantly more per lost record than local data breaches.	N	N	N	N	N	N	N
Arjun Guha	2015	Describes that in static analysis, we need to identify and gather JavaScript files	N	N	N	N	Y	N	N
Mayur Parmar	2019	Uses search string which uses advanced options to find the hidden information.	N	N	N	N	N	N	Y
Rizdqi Akbar Ramadan et.al	2020	explains the importance of performing ample reconnaissance	Y	Y	Y	Y	N	N	N
Suraj S. Mundalik et.al	2015	Explores information gathering techniques and the attack simulation implementation that is done particularly with Kali Linux OS	Y	Y	Y	Y	N	N	N
S.M. Zia Ur Rashid	2019	Describes that the Domain name system has been an essential part of cyber security and an essential part of the web service used	Y	N	Y	Y	N	N	N
Andres Ojamaa	2012	Describes that is easy to find log and bugs in server-side applications	N	Y	Y	Y	N	N	N
Jaspher Kathrine et.al	2020	Tells us about the patterns which play a vital role in Subdomain Enumeration.	Y	N	N	N	N	N	N

Figure 1: Compare and Contrast

2.3 Research Niche

The use of reconnaissance techniques in cybersecurity stands out as a study area after evaluating the chosen publications. The papers specifically discuss numerous information gathering techniques like subdomain enumeration, Google hacking, and javascript enumeration, among others. The discovery of potential attack vectors and weaknesses that attackers can take advantage of requires the use of these approaches.

The assessment of the literature emphasizes the major contributions made by earlier work in the realm of cybersecurity. The articles present various strategies for reconnaissance, the importance of the reconnaissance stage in the attack lifecycle, and potential exploitable flaws. Understanding how to apply these strategies to identify and stop assaults is still lacking, though.

3 Research Methodology

3.1 Research Method

The Recon Framework, a fully functional reconnaissance framework created in shell script. Port scanner, Google dorking, subdomain enumeration, vulnerability scanning, content discovery, Github dorking, and JavaScript enumeration are all part of the Recon Framework. By combining several penetration testing tools and techniques into a single platform, the solution seeks to enhance the process of reconnaissance.

To achieve this goal, the following methodology is adopted:

Literature Review: The study process begins with a thorough review of the tools and methods for existing reconnaissance in the literature. To find relevant research studies, publications, and tools, this involved examining academic databases, trade shows, and other pertinent sources. The study concentrated on outlining the advantages and disadvantages of the currently available tools and methodologies as well as the state of the research in the area. The creation of the test scenarios was made possible by this phase.

Test Scenario: Several test scenarios were used to assess the efficacy and efficiency of various reconnaissance tools and methods based on the findings of the literature review. The scenarios required testing a variety of online apps. The scenarios included various attack paths and techniques that attackers frequently employ to gain unauthorized access to systems, such as the discovery of web-based vulnerabilities. Testing was done on vulnerable website like **http://testfire.net and http://testphp.vulnweb.com.**

Design and implement the Recon Framework using the test scenarios and research objectives: The design and implementation of the Recon Framework will be guided by the test scenarios and research goals.

Experiment: Through a controlled experiment, the researcher assessed the Recon Framework by using it to conduct reconnaissance on vulnerable websites like http://testfire.net/ and http://testphp.vulnweb.com/. To make sure that the results were reliable and consistent, the experiment was carried out in a controlled setting.

The analysis included measures such as time taken to complete the reconnaissance process, time taken by each integrated tool to complete its operation, number of vulnerabilities identified, number of hidden directories found, number of secrets found.

Research Findings documentation, Limitations, and Future Directions:
The research findings are compiled and presented in the final report, which also covered the study's limits, an overview of the literature review, a description of the Recon Framework, the experiment's outcomes, an analysis of the data, and possible areas for further study. To make the report understandable to both technical and non-technical audiences, it was written in a clear and straightforward manner.

3.2 Research Resources

Research resources are the tools, procedures, and data used in conducting research.
Shell Scripting: Shell scripting is the process of writing programs or scripts in a shell language to automate administrative tasks or perform various system functions.

Vulnerable web applications: To assist security professionals in training and learning how to discover, exploit, and patch security flaws, web applications that are vulnerable to attack are purposely developed with known security flaws. These programs can be used to test these products, develop security policies, and train security personnel. Two examples of weak online programs are the websites for Acunetix Art and Altoro Mutual Bank (http://testfire.net/ and http://testphp.vulnweb.com/, respectively).

Assetfinder: It is a robust domain reconnaissance tool designed to identify and gather potential. domains and subdomains associated with a specified target domain. This tool assists security professionals and researchers in comprehensively mapping the digital footprint of a target. Assetfinder's ability to locate subdomains contributes swiftly and efficiently to the initial stages of reconnaissance in penetration testing, enabling a more thorough assessment of an organization's online presence. For example, if testfire.net is a domain then altoro.testfire.net is the subdomain. (hudson, n.d.)

Httprobe: It is a powerful tool used in the field of cybersecurity and penetration testing to assess the availability of HTTP and HTTPS servers associated with a list of domains. This tool takes a provided list of domains as input and performs a probing process to identify which of these domains have active and functioning HTTP and HTTPS servers. This functionality is crucial for reconnaissance, as it helps security practitioners identify live web servers that can be potential targets for further analysis and assessment. For example, httprobe adds http:// or https:// in prefix of the domain and see whether its live or not. (hudson, n.d.)

nDorker: The technique in the realm of cybersecurity that aids in efficiently locating potentially sensitive information online. Dorking involves the use of specialized search queries, known as "dorks," to uncover hidden or confidential data that may not be accessible through standard search methods. nDorker takes this concept to the next level by automating Dorking procedures, enabling faster and more comprehensive data collection.

One of nDorker's prominent functionalities is its ability to perform **Google Dorking**. Google dorks are meticulously crafted search queries that delve deep into Google's search engine to reveal information that may not be easily discoverable using regular search queries. This feature allows cybersecurity professionals to uncover potential vulnerabilities, leaked data, and other sensitive information that might pose security risks to organizations. For example, **site:.altoro.testfire.net db_password =** is an advance search query.

Furthermore, nDorker extends its capabilities to **GitHub Dorking**. Just as with Google dorks, GitHub dorks are specialized search queries intended to pinpoint valuable information stored within GitHub repositories. This can include source code, configuration files, and even sensitive credentials inadvertently exposed in code repositories. nDorker streamlines the process of GitHub dorking, making it faster and more efficient for security practitioners to identify potential security gaps within code repositories. For example, **"altoro.testfire.net" filename:config** is an advance query for github.

Additionally, nDorker offers **Shodan dorking** capabilities. Shodan is a unique search engine that focuses on identifying various internet-connected devices, systems, and services. By utilizing specific search queries, Shodan dorking helps uncover vulnerable or improperly configured devices and services that could be exploited by malicious actors. nDorker's integration of Shodan dorking simplifies the discovery of potential targets for penetration testing and vulnerability assessment. For example, **hostname:altoro.testfire.net** is an advance search query for shodan. (Khatiwada, n.d.)

Paramspider: Identifies parameters using the entered domain's web archives. also locates parameters from subdomains. supports the exclusion of URLs with extensions. efficiently and

properly saves the output result. For example, http://testphp.vulnweb.com/listproducts.php?artist=%27 artist is the parameter for the URL. (Batham, n.d.)

Qsreplace: Accept URLs on stdin, replace every value in the query string with one that has been supplied by the user, and only output one set of query string parameters per host and path. For example, by executing this command ***python3 paramspider.py -d http://testfire.net/ \| qsreplace'"><script>confirm(1)</script>'*** the URL *http://testfire.net/util/serverStatusCheckService.jsp?**HostName=%22%3E%3Cscript%3Econfirm%281%29%3C%2Fscript%3E.*** The value to parameter **HostName** is automatically provided by Qsreplace. (Hudson, n.d.)

ffuf: Typical directory finding, virtual host discovery (without DNS records), and GET and POST parameter fuzzing are all made possible by the fast web fuzzer implemented in Go. For example, https://altoro.testfire.net/**aux**. Now **aux** is the hidden directory found in altoro.testfire.net subdomain. (Hoikkala, n.d.)

SecretFinder: It is a Python script written to find sensitive data in JavaScript files, such as apikeys, accesstokens, authorizations, JWT tokens, etc. It accomplishes this by combining a sizable regular expression with the help of jsbeautifier for Python. Four smaller regular expressions make up the entire regular expression. These oversee searching and locating anything in JS files. (mallock, n.d.)

Nmap: Cybersecurity professionals regularly use Nmap, a powerful network exploration and security auditing tool, to locate hosts and services on a computer network. For example,

PORT STATE SERVICE VERSION

80/tcp open http Apache Tomcat/Coyote JSP engine 1.1

|_http-server-header: Apache-Coyote/1.1

|_http-title: Altoro Mutual

Depicts that there is port 80 open on the Altoro mutual website hosted server. (nmap.org, n.d.)

Virtual environments: Virtual environment that will be used in study is Virtual Box. Attacking virtual machine called Kali Linux was used in the with the help of VirtualBox Manager. (VirtualBox, n.d.)

Test website: The recon framework's functionality is assessed using test websites. To assess the effectiveness of the recon framework, vulnerable web applications like http://testfire.net/ and http://testphp.vulnweb.com/ are employed. The efficiency of various security measures can be evaluated using known vulnerabilities in these test programs.

4 Design Specification

The script called **recon.sh** utilizes several tools such as **`assetfinder`, `httprobe`, `nDorker.py`, `SecretFinder.py`, `paramspider.py`, and `nmap`.**

Before proceeding with the script, there are specific pre-requisites and requirements that need to be fulfilled to ensure its successful execution. These prerequisites encompass the necessary environment and tools that enable the script to function effectively. Here is an elaboration of the key prerequisites:

Bash Environment: The script is designed to run in a Bash environment. It leverages the capabilities of the Bash shell for executing commands, handling variables, and managing the flow of the reconnaissance process. I used **kali Linux** for the Bash environment.

Pre-requisites:
Assetfinder: This tool is integral for subdomain discovery. It assists in locating subdomains associated with the target domain.

Httprobe: It is employed for probing live domains. It confirms the availability of third-level domains by verifying their responsiveness to HTTP requests.

nDorker.py: nDorker serves as the tool for Google, GitHub, and Shodan dorking. It conducts specialized searches to uncover potential information leaks and vulnerabilities. SecretFinder.py: SecretFinder.py is essential for identifying secrets within JavaScript files. It performs a thorough scan for sensitive information that might be embedded in the JavaScript code.

Paramspider.py: This tool plays a critical role in identifying OWASP Top 10 vulnerabilities, including XSS, LFI, SSRF, SQL Injection, and Command Injection. It systematically analyzes parameters for vulnerabilities.

Nmap: It is utilized for scanning open ports. It helps in identifying active services and potential entry points for exploitation.

Ffuf: It is a versatile tool used for directory enumeration. It helps in uncovering hidden directories and files that might be vulnerable to unauthorized access.

Qsreplace: It is employed in the context of parameter-based vulnerabilities. It replaces specific query string values for testing various inputs to identify vulnerabilities. Collectively, these tools contribute to a comprehensive reconnaissance process that involves subdomain discovery, live domain probing, vulnerability identification, and directory enumeration. By ensuring that these tools are installed and accessible within your environment, the script can be effectively executed to enhance security assessment and risk mitigation strategies for the target domain.

The **recon.sh** script follows a well-defined workflow to conduct comprehensive reconnaissance on a target domain, utilizing a combination of specialized tools and techniques. The step-by-step process ensures a thorough assessment of potential vulnerabilities and information leaks. Here's an expanded description of each workflow stage:

User Input: The reconnaissance process begins with the user providing a domain as a command-line argument. This domain serves as the focal point for the entire analysis, guiding the script towards gathering relevant information.

Subdomain Gathering and Sorting: The script initiates by utilizing assetfinder, a subdomain discovery tool. It collects a range of subdomains related to the provided domain. These subdomains are then sorted into third-level domains, which helps in organizing and categorizing the gathered information more effectively.

Probing for Live Domains: To distinguish active and responsive domains, the third-level subdomains undergo probing using httprobe. This step confirms the viability of each domain, ensuring that only live and accessible hosts are further considered in the analysis.

Dorking for Information Leaks: The script employs nDorker.py to conduct specialized searches on the live subdomains. It performs Google, GitHub, and Shodan dorking, seeking out any potential information leaks that might expose sensitive data. This phase enhances the understanding of potential vulnerabilities in the target domain.

Secrets Analysis in JavaScript Files: JavaScript files embedded within the live subdomains are meticulously examined using SecretFinder.py. The tool identifies potential secrets that might be hidden within the JavaScript code. This step is crucial for uncovering any sensitive information that could be exploited by attackers.

Identification of Web Vulnerabilities: The script employs the powerful combination of paramspider and qsreplace to detect OWASP based vulnerabilities, including Cross-Site Scripting (XSS), Local File Inclusion (LFI), Server-Side Request Forgery (SSRF), SQL Injection, and Command Injection. Various techniques are applied to each parameter, analyzing responses for vulnerability indicators.

Open Port Scanning: The script utilizes nmap to conduct comprehensive port scanning on the live subdomains. This step reveals the active services and open ports associated with each domain. The results help in identifying potential points of entry and attack vectors.

Directory Enumeration: To uncover hidden directories and files susceptible to unauthorized access, the script performs directory enumeration using ffuf. By systematically testing different directory paths, it helps in understanding the website's structure and possible areas of exploitation.

By meticulously executing each of these steps, the script ensures a thorough and methodical reconnaissance process. This comprehensive analysis aids in identifying vulnerabilities, potential information leaks, and other security concerns associated with the target domain.

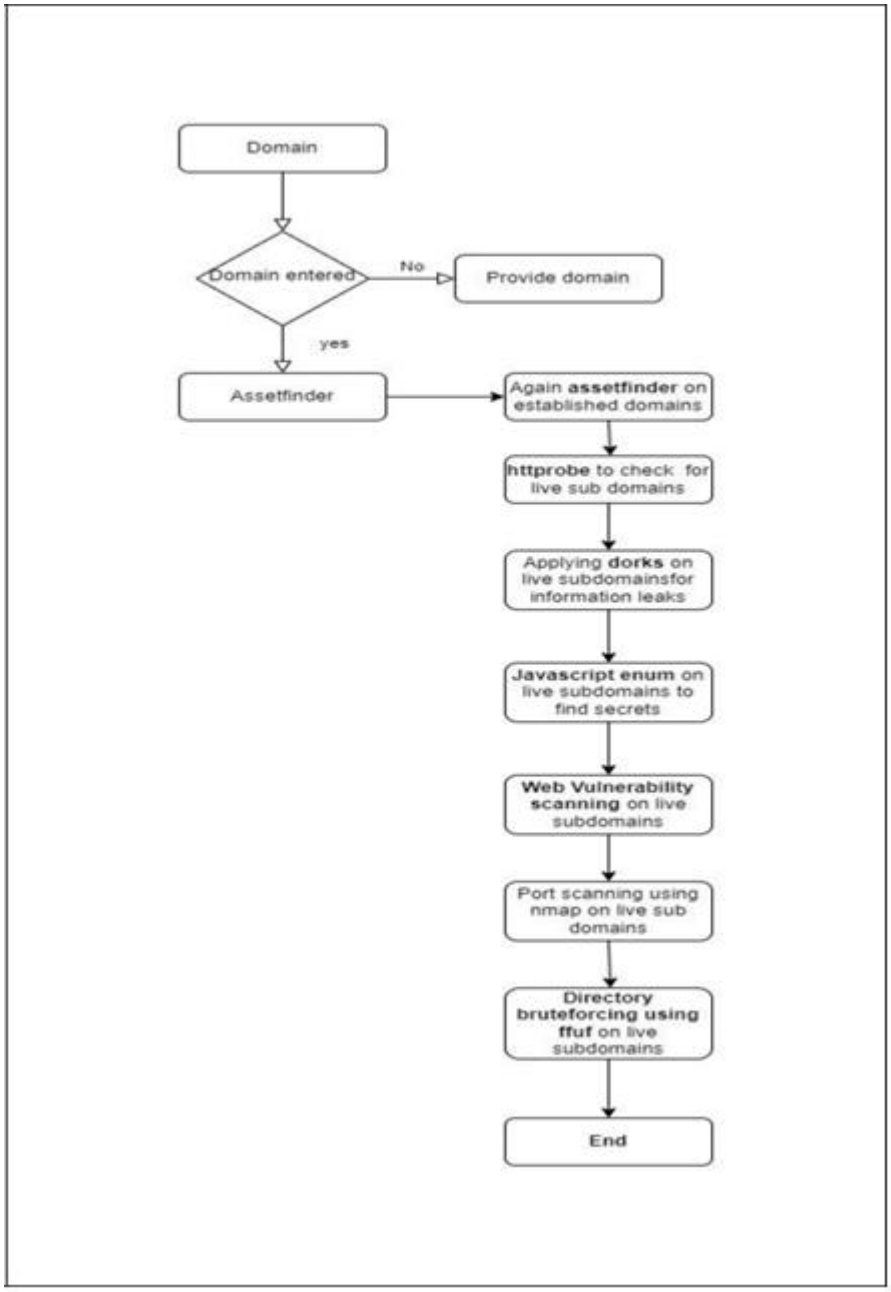

Figure 2: Workflow

Output of the tool: The script generates various directories containing the results of subdomain discovery, live host probing, dorking, vulnerability identification, open port scanning, and directory enumeration. These results provide insights into potential security vulnerabilities and information leaks associated with the target domain.

5 Implementation

In the final stage of the implementation, the bash script called recon.sh is executed to perform reconnaissance on a target domain. This script automates various tasks, such as subdomain discovery, probing for live hosts, dorking, identifying vulnerabilities, port scanning, and directory enumeration.

Execution Flow and Outputs:

The bash script **recon.sh** orchestrates the reconnaissance process in a systematic manner, performing multiple tasks to gather information, identify vulnerabilities, and assess potential attack vectors. The script's execution flow can be summarized as follows:

First, the Main output directory is created **recon_output**. Which will have all the corresponding directories regarding the ouput of the other tools.

Subdomain Discovery and Compilation: The script employs the assetfinder tool to efficiently discover subdomains associated with the target domain. The discovered subdomains are then filtered to extract third-level domains, which are subsequently compiled and stored in **recon_output/thirdlevels** directory.

Probing for Live Domains: The **httprobe** tool is utilized to probe and identify active third-level domains from the compiled list. The results are saved as both original and sorted lists in the **probed hosts text file**.

Dorking and Vulnerability Identification: The script leverages nDorker to perform Google dorking, GitHub dorking, and Shodan dorking, aiming to identify information leaks in the probed subdomains. The results are categorized and stored based on the type of dorking performed in **recon_output/Dorks** directory.

JavaScript Secrets Identification: The script employs **SecretFinder** to analyze JavaScript files present in the probed subdomains. It identifies potential secrets hidden in the code and records the findings in dedicated output **recon_output/Javascript_Enum** directory.

OWASP Top 10 Vulnerabilities: The script utilizes paramspider, qsreplace, and a series of curl commands to identify vulnerabilities aligned with the OWASP Top 10 list. The reconnaissance focuses on vulnerabilities such as XSS, LFI, SSRF, and SQL Injection, recording the findings in **recon_output/OWASP_Top_10** directory.

Port Scanning: The script employs nmap to perform comprehensive port scanning on the probed domains. The results include detailed information about open ports and services running on the target domains. The results are stored in **recon_output/scans** directory.

Directory Enumeration: ffuf is used to enumerate directories on the probed subdomains. The script uses a common wordlist for directory enumeration and saves the results for further analysis in **recon_output/Dirb** directory.

Tools and Languages Used:

- **Bash:** The main scripting language used for implementing the reconnaissance tasks.
- **assetfinder:** Used for subdomain discovery.
- **httprobe:** Employed to probe for active domains.
- **nDorker:** Utilized for Google dorking, GitHub dorking, and Shodan dorking.
- **SecretFinder:** Applied for identifying secrets in JavaScript files.
- **paramspider and qsreplace:** Employed for vulnerability identification.

- **curl:** Used for sending HTTP requests to web servers.
- **nmap:** Utilized for port scanning.
- **ffuf:** Applied for directory enumeration.

6 Evaluation

In this evaluation, we will analyze the results and main findings of the reconnaissance script **recon.sh** provided. The script aims to automate various tasks involved in reconnaissance on a given target domain.

Experiment 1 / Case Study1: Reconnaissance of testphp.vulnweb.com

Phase 1: Subdomain Discovery and Probing

Subdomain Discovery:

The script efficiently used assetfinder to discover subdomains associated with the target domain.
Execution time: Average of 1.062 seconds.
Implication: Swift subdomain discovery is indicative of efficient initial reconnaissance.

Compiling Third-Level Domains:

The script successfully compiled third-level domains from the discovered subdomains.
Execution time: Average of 0.021 seconds.
Implication: Rapid compilation of third-level domains suggests effective data organization.

Gathering Full Third-Level Domains:

The script used assetfinder to gather complete third-level domains.
Execution time: Average of 1.940 seconds.
Implication: Quick gathering of full third-level domains showcases effective data collection.

Probing for Alive Third-Level Domains:
The script performed probing using httprobe to identify active third-level domains.
Execution time: Average of 10.022 seconds.
Implication: The longer execution time might be due to varying response times of probed domains.

Phase 2: Dorking and JavaScript enumeration

Dorking:

The script executed Google, GitHub, and Shodan dorks promptly.

Execution time: 0.243 seconds.
Implication: Efficient execution of dorking to gather publicly available information.

JavaScript Secrets Identification:

The script attempted to find secrets in JavaScript files but did not find any.
Implication: No sensitive information leakage detected in probed subdomains' JavaScript files.

Phase 3: Vulnerability scanning

Vulnerability Identification:

The script successfully identified various vulnerabilities, including 6 XSS, 1 Linux LFI, 1 Windows LFI, 12 SQL Injection, and 5 SSRF vulnerabilities.
Execution time: 1 minute and 23.150 seconds.

Implication: The target domain exhibited multiple vulnerabilities, with SQL Injection being the most significant concern.

Phase 4: Port Scanning and Directory Enumeration

Port Scanning:

The script detected 2 open ports on the target domain.
Execution time: 1 minute and 13.053 seconds.
Implication: Limited open ports indicate a potentially restricted attack surface.

Directory Enumeration:

The script enumerated directories using ffuf.
Execution time: 30.865 seconds.
Implication: The enumerated directories could reveal potential entry points for attackers.

Overall Evaluation:

The reconnaissance script efficiently identified vulnerabilities and provided insights into potential attack vectors. The results suggest an urgent need to address vulnerabilities such as XSS, LFI, SQL Injection, and SSRF. While open ports are limited, the directory enumeration phase highlighted possible entry points for further investigation. The script's overall execution time was approximately 3 minutes and 31.47 seconds, showcasing reasonable efficiency in reconnaissance.

Experiment 2/ Case Study 2: Reconnaissance of testfire.net

Phase 1: Subdomain Discovery and Probing

Subdomain Discovery:

The script efficiently utilized assetfinder to find subdomains associated with the target domain.
Execution time: Average of 1.125 seconds.
Implication: Swift subdomain discovery suggests an efficient initial reconnaissance phase.

Compiling Third-Level Domains:

The script successfully compiled third-level domains from the discovered subdomains.
Execution time: Average of 0.102 seconds.
Implication: Rapid compilation of third-level domains showcases effective data organization.

Gathering Full Third-Level Domains:

The script gathered complete third-level domains using assetfinder.
Execution time: Average of 5.474 seconds.

Implication: Quick gathering of full third-level domains suggests efficient data collection.

Probing for Alive Third-Level Domains:

The script performed probing with httprobe to identify active third-level domains.
Execution time: Average of 0.84 seconds.
Implication: The longer execution time might be due to varying response times of probed domains.

Phase 2: Dorking and Javascript enumeration

Dorking:

The script executed Google, GitHub, and Shodan dorks promptly.
Execution time: Average of 0.787 seconds.

Implication: Efficient execution of dorking to gather publicly available information.

JavaScript Secrets Identification:

The script attempted to find secrets in JavaScript files but did not find any.
Execution time: Average of 8.494 seconds
Implication: No sensitive information leakage detected in probed subdomains' JavaScript files.

Phase 3: Vulnerability Scanning

Vulnerability Identification:

The script successfully identified vulnerabilities, including 2 XSS and 7 SSRF vulnerabilities in various subdomains.
Execution time: Average of 5 minutes and 27.429 seconds
Implication: The identified vulnerabilities highlight potential security risks within the target domains.

Phase 4: Port Scanning and Directory Enumeration

Port Scanning:

The script detected 4 open ports on each of the probed subdomains.
Execution time: Average of 7 minutes and 60.606 seconds.
Implication: Multiple open ports could indicate a broader attack surface.

Directory Enumeration:

The script enumerated directories using ffuf on each of the probed subdomains.
Execution time: Average of 4 minutes and 47.027 seconds.
Implication: Enumerated directories could reveal potential entry points for attackers.

Overall Evaluation:

The reconnaissance script effectively identified potential vulnerabilities, open ports, and entry points in the target domain and its subdomains. The results underscore the importance of promptly addressing the identified vulnerabilities to enhance the security posture. The script demonstrated efficiency in various stages of reconnaissance, allowing security practitioners to proactively mitigate potential risks. The script's overall execution time was approximately 18 minutes and 0.23 seconds, showcasing reasonable efficiency in reconnaissance.

6.1 Discussion

The comprehensive evaluation of the reconnaissance script's performance through both experiments offers a robust foundation for critical analysis and potential design enhancements. While the script demonstrated commendable efficiency in various reconnaissance phases, there are areas warranting discussion and considerations for improvement.

Efficiency and Effectiveness of the Script:

Both experiments highlighted the script's efficiency in subdomain discovery, third-level domain compilation, and gathering, indicating its capability to initiate reconnaissance promptly. The proactive use of dorking techniques for information gathering from publicly available sources showcased the script's efficacy. Moreover, the identification of vulnerabilities, especially those aligned with the OWASP Top 10 list, emphasizes its potential to assess security risks.

Contextualization within Previous Research:

The findings of both experiments align with the context of previous research on reconnaissance tools and techniques. Existing tools have demonstrated limitations, and the presented script

attempts to address these by integrating diverse methods. The script's strengths, such as subdomain discovery and vulnerability identification, align with the broader recognition of these aspects in the security community. However, there remains room for improvement, particularly in terms of refining the probing approach and expanding the scope of vulnerability assessment beyond the vulnerabilities identified.

7 Conclusion and Future Work

In conclusion, the central research question of this study was: "To what extent would integrating multiple pentest tools and techniques into a single platform improve the process of reconnaissance in penetration testing?" The objectives were to develop and evaluate a reconnaissance script that automates various tasks involved in reconnaissance on target domains. This report has documented the outcomes of two case studies: one involving the reconnaissance of testphp.vulnweb.com and the other concerning testfire.net

The results of both case studies reveal the effectiveness of the developed reconnaissance script in automating and streamlining the reconnaissance process. The script demonstrated proficiency in subdomain discovery, compilation, and probing, as well as in the execution of dorking, vulnerability identification, port scanning, and directory enumeration. Notably, vulnerabilities like XSS, SQL Injection, and SSRF were successfully identified in both target domains, highlighting their susceptibility to potential cyber threats.

Suggestions for Improvement:

Depth of Vulnerability Analysis: For a more comprehensive assessment, delve deeper into the vulnerabilities' impact, possible attack scenarios, and potential consequences.

Integration with Other Tools: Integrate complementary tools such as Nikto, or Burp Suite to enhance vulnerability identification.

Comprehensive Performance Metrics: Establish a set of performance metrics considering the scale of target domains and different phases to ensure consistent evaluation.

Key Findings and Implications:

The efficacy of the developed script is evident in its ability to streamline the reconnaissance process, expedite data collection, and identify potential attack vectors. The efficiency demonstrated in subdomain discovery, dorking, and vulnerability identification showcases the script's value in enhancing the penetration testing process.

However, it is important to acknowledge the limitations of this research. The script's efficiency is constrained by varying response times from probed domains, potentially impacting the overall execution time. Additionally, the script's performance might differ on more complex and diverse target domains.

Future Work:

In terms of meaningful future work, rather than simply expanding parameters within the existing script, a follow-up research project could focus on:

- **Enhancing Vulnerability Scanning:** Investigate methods to improve the accuracy of vulnerability detection and classification, potentially integrating machine learning techniques to refine results.

- **Dynamic Response Time Handling:** Develop strategies to optimize the script's response time handling, allowing for more efficient probing and reducing overall execution time.
- **User-Friendly Interface:** Create a user-friendly interface for the script, enabling security practitioners with varying levels of technical expertise to utilize its capabilities effectively.

These proposed directions can lead to more comprehensive and advanced automation of the reconnaissance process, empowering security professionals to conduct thorough assessments and secure digital assets effectively. Additionally, there is potential for the commercialization of the enhanced script as a comprehensive penetration testing tool, providing organizations with a robust solution to assess and fortify their digital infrastructures.

8 References

Batham, D., n.d. *Mining parameters from dark corners of Web Archives.* [Online]
Available at: https://github.com/devanshbatham/ParamSpider

Becker, K., Pape, S., Schaab, P. & Schosser, D., 2017. *Conference: International Conference on Trust and Privacy in Digital Business.* s.l., s.n.

Dar, U. A. & Iqbal, A., 2018. The Silent Art of Reconnaissance: The Other Side of the Hill. *ResearchGate.*

Guha, A., Saftoiu, C. & Krishnamurthi, S., n.d. *The Essence of JavaScript.* s.l., s.n.

Hoikkala, J., n.d. *Fast web fuzzer written in Go.* [Online]
Available at: https://github.com/ffuf/ffuf

hota, S., 2023. *https://www.linkedin.com/pulse/top-10-subdomain-finders-great-website-reconnaissance-samit-hota/.* [Online]
Available at: https://www.linkedin.com/pulse/top-10-subdomain-finders-great-website-reconnaissance-samit-hota/

Hudson, T., n.d. *Accept URLs on stdin, replace all query string values with a user-supplied value.* [Online]
Available at: https://github.com/tomnomnom/qsreplace

hudson, T., n.d. *Find domains and subdomains related to a given domain.* [Online]
Available at: https://github.com/tomnomnom/assetfinder

hudson, T., n.d. *Take a list of domains and probe for working HTTP and HTTPS servers.* [Online]
Available at: https://github.com/tomnomnom/httprobe

kali, n.d. *Eyewitness.* [Online]
Available at: https://www.kali.org/tools/eyewitness/

Kathrine, J., 2020. *Comparative analysis of subdomain enumeration tools and static code analysis.* s.l., ISSN.

Khatiwada, N., n.d. *Automate dorking while doing bug bounty or other stuffs..* [Online]
Available at: https://github.com/nerrorsec/Google-Dorker

Kumar, A. & Arora, S., 2018. A Review on Web Application. *ResearchGate*, Volume 6, p. 4.

mallock, n.d. *SecretFinder - A python script for find sensitive data (apikeys, accesstoken,jwt,..) and search anything on javascript files.* [Online]
Available at: https://github.com/m4ll0k/SecretFinder

Mundalik, S., 2015. Penetration Testing: An Art of Securing the System (Using Kali Linux). *International Journal of Advanced Research in Computer Science and Software Engineering.*

nmap.org, n.d. *Nmap: the Network Mapper - Free Security Scanner.* [Online]
Available at: https://nmap.org/

Ojamma, A. & Duuna, K., 2012. *Assessing the security of Node.js Platform.* s.l., International Conference for Internet Technology and Secured Transaction.

Parmar, M., 2019. Google Dorks -Advance Searching Technique. *ResearchGate.*

Ramada, R. & Maland, R., 2019. *Sudomy: Information Gathering Tools for Subdomain Enumeration and Analysis.* s.l., The 2nd International Conference on Engineering and Applied Science.

Rashid, S. M. Z. U., 2019. *Understanding the Security Threats of Esoteric Subdomain Takeover.* s.l., International Conference on Electrical, Computer and Communication Engineering.

Roy, A., Mejia, L., Helling, P. & Olmsted, A., 2017. *Automation of Cyber-Reconnaissance.* Charleston, IEEE.

tomnomnom, 2022. *qsreplace.* [Online]
Available at: https://github.com/tomnomnom/qsreplace

VirtualBox, n.d. *Welcome to VirtualBox.org!.* [Online]
Available at: https://www.virtualbox.org/

Young, S., 2014. *Using Open Source Reconnaissance Tools for Business Partner Vulnerability Assessment.* s.l., SANS.