UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI
DEPARTMENT OF INFORMATION COMMUNICATION TECHNOLOGY

BACHELOR THESIS

By:

Trần Đức Tuấn - BI12-467
Cyber Security

Title:

Web Application Reconnaissance Flow Analysis

Supervisor: Nguyễn Minh Hương - ICT Lab

Hanoi, 2024

Table of Contents

Abstract.

My project aims to separate, research, and analyze a standard reconnaissance workflow. I focus on researching protocol and service, storing information on a web application, and analyzing the output value of each stage in the workflow. Those support the idea that automatically the whole reconnaissance process.

I developed a tool with the purpose of an automated reconnaissance process. The tool is designed as a set of modules corresponding with its information-gathering technique such as domain discovery, endpoint fuzzing, collect DNS records. The tool should also be developed with the idea that it is automatic for the whole reconnaissance process to optimize the target's data-collecting process.

I used Python3 to develop the tool. However, for my tool to run correctly, I need to research and analyze the reconnaissance process, which includes techniques such as OSINT, fuzzing, and scanning. I would like to clarify the idea, workflow, and protocols for each method.

For future improvement of the tool, modules that I did not develop would be replaced by self-developed ones. Besides, I wish the modules could run in parallel to optimize time.

I. Introduction.

1. Problem Statement.

In the field of offensive security, reconnaissance is a crucial initial phase in identifying potential vulnerabilities within a target web application however, the process of executing reconnaissance can take much time and effort. Pen-testers often rely on multiple tools and techniques to gather necessary information, leading to inefficiencies, redundancies, and a fragmented workflow. Moreover, the increasing complexity of web applications creates additional challenges. Traditional reconnaissance tools may need to fully address these complexities, resulting in incomplete data collection. This gap can lead to missed opportunities for identifying critical security vulnerabilities [1]. While most people in the field of offensive security are performing manual reconnaissance, there is an automated tool named reNgine which is designed to supercharge the recon process for security pros, pen-testers, and bug bounty hunters [2]. But the drawback of reNgine is depending on integrated tools such as AMASS, subfinder, nuclei, … [3].

Given these challenges, I plan to develop a user-friendly automated tool that efficiently gathers information from various sources to streamline the reconnaissance process. By automating both passive and active reconnaissance, the tool can enhance the accuracy and speed of the information-gathering process, allowing for more effective vulnerability identification and mitigation.

2. Proposed Solution.

To deal with the challenges of manual reconnaissance in cybersecurity, this project proposes the development of an automated reconnaissance tool that streamlines both passive and active reconnaissance processes which are detailed research and analysis to clarify protocols and workflows. The tool automates the critical stages of both passive and active reconnaissance, following a structured workflow that mirrors the methodologies used by penetration testers. This includes gathering publicly available information (OSINT) during passive reconnaissance and conducting targeted probes during active reconnaissance. The tool will be designed with a module structure, allowing further customize and handling of a wide range of targets from small websites to complex web applications. Pen-testers could choose between using the free version of gathering data's APIs or providing product keys in a config file. To support further analysis and decision-making, the tool will generate raw data on the reconnaissance findings. These data include insights into the web application's infrastructure, potential vulnerabilities, and other critical information.

3. Project Objective.

The primary objective of this project is to develop a recon tool to equip recon modules which are mentioned in the research part. My purpose is to actualize all the proof of concept of techniques and protocols in the research part in a usable tool. This tool would use the domain name of a web application in the tool's automated recon process and return raw data of the web application.

The secondary objective is to clarify protocols and techniques that are used to search for structure and information related to the input target domain. In each technology or protocol or some kind of researched information, I would like to determine the workflows of technologies and protocols and mention critical data of web application target what can archive from those techniques or worthy for the next stage of gathering data.

4. Project Scope.

My project involves breaking down the entire reconnaissance process into individual phases and exploring the search and analysis techniques, protocols, and workflows associated with each phase. It also provides an introduction to specific tools designed for each phase and includes the development of a new reconnaissance tool. However, the project does not focus on customizing an open-source reconnaissance tool for automation, defining red team techniques in a custom reconnaissance flow, or investigating and analyzing other phases of the red team workflow.

II. Background.

1. Web Application.

A web application runs on a web server and is accessed through a web browser over a network, such as the Internet or an intranet. Unlike traditional desktop applications installed locally on a user's computer, web applications are accessed via URLs and can be used from any device with an internet connection [4].

2. Web Application Domain.

A web application's domain refers to the unique name that identifies a website on the internet. It's the address that users type into their web browsers to access a particular website. The web domain is part of the URL (Uniform Resource Locator), which directs visitors to the specific site [6]. In my research scenario, a web domain is the primary identifier used to explore and analyze the various aspects of a target's online presence.

3. Offensive Security - Red Teaming & Penetration Testing.

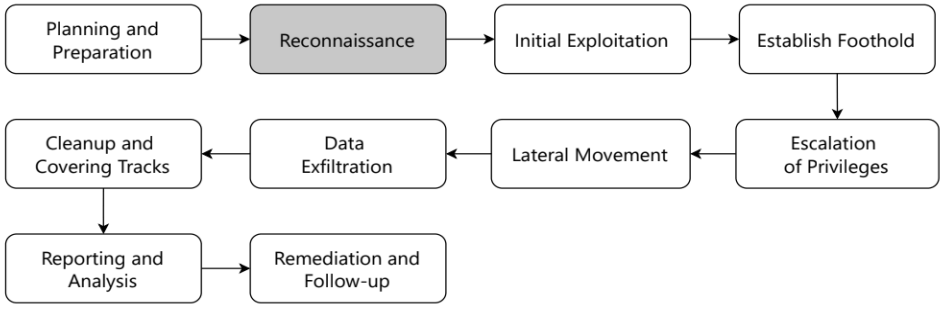

Figure 1: Red team workflow diagram.

Figure 1 presents the process of red teaming which is a structured process used to simulate an attack on an organization to identify vulnerabilities and improve security [7]. Red teaming focuses on deep attacks on the organization's server such as taking control server and attacking related infrastructure and servers while pen-testing which presents is initial access of the whole red team process involves identifying vulnerabilities, exploiting them to determine their impact, and providing actionable recommendations to mitigate the discovered weaknesses. Pentest which is the initial exploitation step typically includes a comprehensive examination of various aspects of the target environment, such as network infrastructure, web applications, and internal systems.

4. Reconnaissance.

Its regular name is Recon for short. It is the process of gathering information about a target system, network, or organization before attempting an attack or security assessment. The Reconnaissance process involves collecting extensive information about the potential targets, their vulnerabilities, and possible attack vectors. In this report, I focused on web reconnaissance, demonstrated analysis workflows, and clarified the analysis of the mentioned tools in the reconnaissance process.

5. Web Application Overall Structure

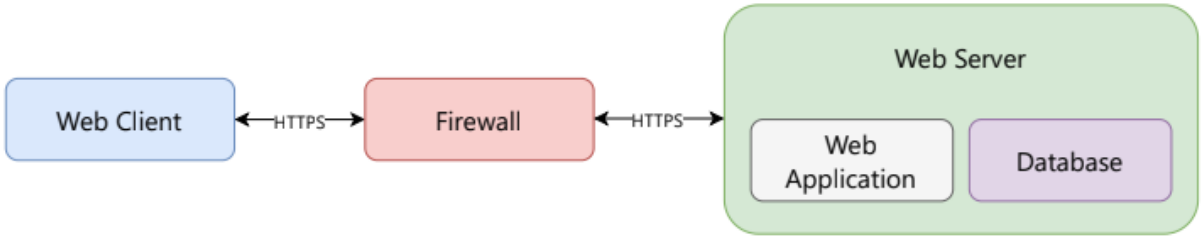

Figure 3: Web Application Overall Structure.

This diagram includes the main parts of the web server such as the client, server, applications, databases, and firewall. The web client is the user's device, and has role is to render and display data received from the server. A firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules. On the hardware side, a web server is a computer that stores web server software and a website's component files. A web server connects to the Internet and supports physical data interchange with other devices connected to the web. On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an HTTP server. An

HTTP server is software that understands URLs and HTTP. An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device. Web Application is an application run on a corresponding web engine that is installed inside a web server, resolves incoming requests from the client, executes logic, and responds to the client. The database is a service that plays the role store and providing data for resolving logic processes in web applications.

III. Methodology.

1. Web Domain Reconnaissance Workflow.

I would like to separate the recon workflow into stages by techniques before digging into the analysis. A recon process is regularly divided into two main stages, which are passive recon and active recon. While passive recon involves gathering information without directly interacting with the target, the other one involves interacting with the target system. This can be more detectable by the target but often captures more detailed information.

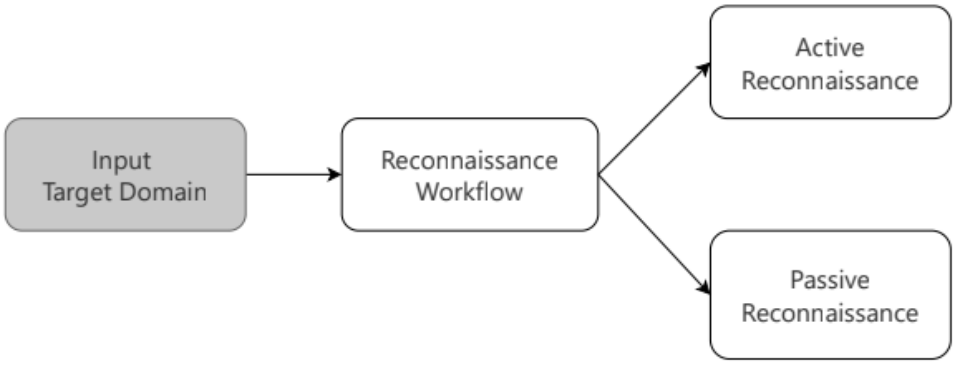

Figure 2: Reconnaissance workflow diagram.

In the passive recon stage, I would like to demonstrate analyzing workflows and tactics for searching for information related to the target. This stage also has another name and is open-source intelligence (OSINT). This technique focuses on searching for critical data of targets in several public databases, leaked databases, protocols, and sensitive credentials.

In the active recon stage, pen testers directly interact with the web application to extract information based on requests and responses from the server. This process

involves using protocols to transmit data. It provides pen-testers with an overview of the web application's structure, including endpoints, subdomains, and parameters. At the end of the workflow, pen-testers scan for potential vulnerabilities in the target using various vulnerability scanning templates.

2. Passive Reconnaissance.

OSINT is an abbreviation for Open-Source Intelligence. It is the process of collecting, analyzing, and utilizing information that can be gathered from the internet. The expected output of the process is critical data related to the target, such as IP address, domain information, employees' email, sensitive information, hidden endpoints, documents, and social network information.

2.1 OSINT Definition and Workflow

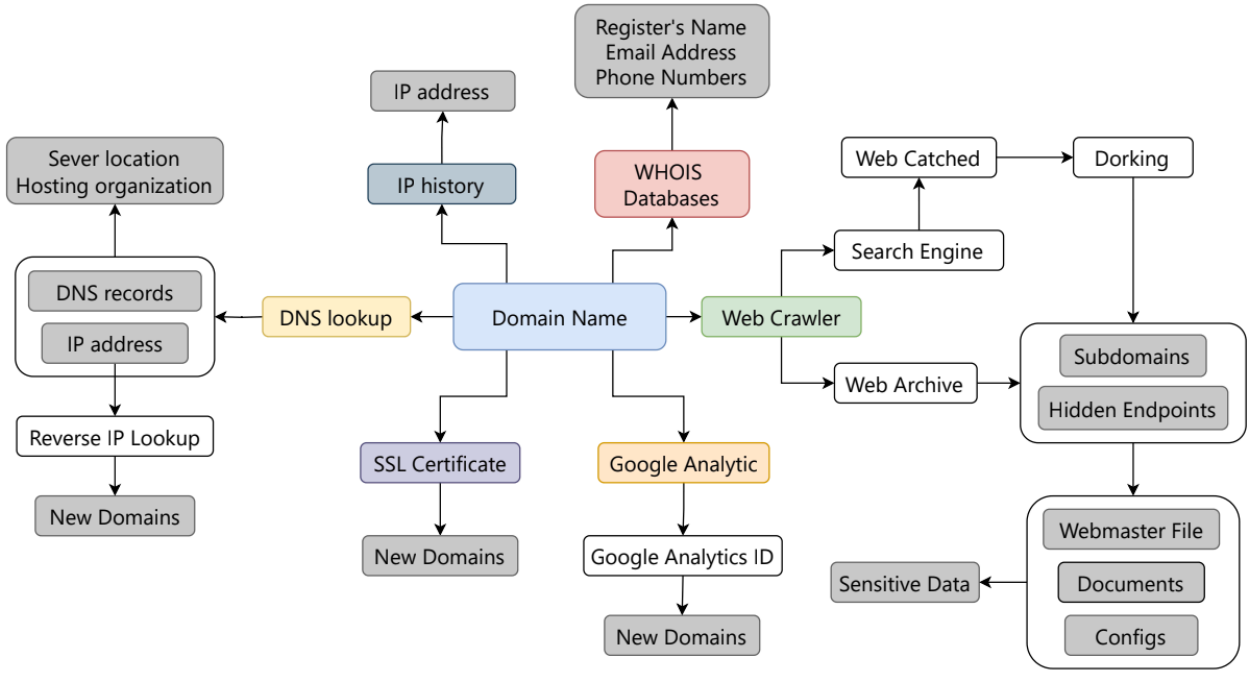

Figure 4: OSINT workflow diagram.

This workflow is customized from the OSINT domain workflow of IntelTechniques[5]. The workflow starts at the blue center box meaning that this process requires a live web domain. Above the main colored box around the domain box are five resources or techniques for gathering, extracting & analyzing data: Crawler, DNS lookup, WHOIS database, SSL Certificate, and Google Analytics. Crawler is the foundation of both search engines and third-party web archives. While the former uses a crawler to optimize searching performance and provide a searching operator, the latter provides

services for data tracking. Using Google search engine operator (Google Dorking) to optimize searching performance and collect the most qualified data. This technique also collects sensitive information (credentials, config,...) that is embedded in web content, hidden paths, subdomains, and server hardware information. The process continues with the query to the WHOIS database by payload is the target's domain name to gather the domain register's name email and phone number,… SSL certificates enable websites to use HTTPS, which is more secure than HTTP. An SSL certificate is a data file hosted in a website's origin server. Thus, there is a possibility of discovering the domain and subdomain with an SSL certificate while that certificate covers both the main domain and the subdomain. The host target IP address can be collected in the DNS query process's DNS syntax response. From the collected IP address, pen-testers can track data related to the host location and hosting organization or get new domains hosted in the same hardware by reverse IP lookup. Once you have the Google Analytics ID from one domain, pen-testers can search the web for other domains using the same ID cause the same entity manages it.

2.2 Web Crawler.

As mentioned in Figure 4, a web crawler is an automated program that systematically browses the web to collect, retrieve, and index information from websites. Here is a general workflow of a web crawler.

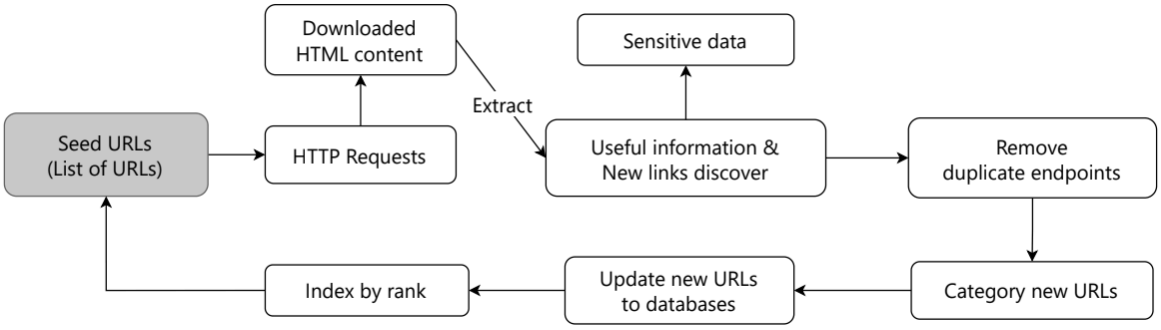

Figure 5: Crawler process diagram.

The web crawler prepares a list of URLs (Seed URLs) as input for the crawl process. The selection of seed URLs depends on the goals of the web crawler. For each endpoint in the seed list, the crawler uses HTTP requests to fetch the HTML content of the web pages and parses it based on HTML tags. Then, the crawler extracts useful information, most of which are external links pointing to new websites. Sometimes, the crawler can detect sensitive data in the HTML content. In the view of Googlebot, for searching purposes, it prioritizes extracting content in header tags and images. After

finishing the downloading process of the seed list, the crawler removes duplicates, irrelevant links, or URLs that already exist in the seed list. URLs are prioritized based on the crawler's algorithms, considering factors like page importance and update frequency while scheduling to avoid server overload. Unlike some specific-purpose spider bots, crawlers that serve search engines like Google or Bing index URLs based on their rank (number of backlinks) and categories to enhance performance. Backlinks also known as inbound links or incoming links, are links from one website to a page on another website. It is an external link placed on other sites that point to your website. Last, the crawler updates new-found URLs into the seed list to initialize new crawl processes.

The web crawler plays an important role in the search engine process. Indeed, using search engines to find information means searching for data from cached versions of websites that match the search's form. A cached page is web content that has been saved by a search engine on its servers or by a user's browser. Sometimes, internet users can not access search results in the browser because it is cached data of the page at crawled time but not real-time data.

For the OSINT process, the crawler provides an overall view of the target's web structure, especially hidden endpoints, expired domains, or internal documents. Opposite to search engines' crawlers, third-party crawlers not only store web content but also store and index elder versions of web cache. This difference helps reveal potential vulnerabilities of the target web application. I consider a scenario where a web developer has to shut down a service on the web application that points to an expired domain. But the developer just deletes a service's DNS record so that a basic user is unable to access the service, while the attacker may find the IP of that service and, of course, old services always contain vulnerabilities.

2.3 Search Engine Dorking.

While using Google searching, Google interprets the search query by splitting it into individual words or terms and searches for pages that are relevant to any or all of those terms. Google uses its ranking algorithms to show the most relevant results based on factors such as relevance, popularity, user behavior, and content quality. Google might also consider synonyms, related terms, and variations of the words in the query to provide a comprehensive set of results. Google tries to understand the context and intent behind the search query to offer the most useful results.

Because regular searching is easy to use and collects many search results by inputting search terms, this feature can cause a critical weak point in low accuracy. To avoid is disadvantage, attackers suggest using a more advanced searching technique. Google Dorking, also known as Google hacking, involves using advanced search

operators to find specific information. It involves using basic search operators to find information that is publicly accessible but might not be immediately obvious. It's often used to find particular files, information, or directories on a website. Dorking is a combination of using fixed terms placed in double or single quotes and Google search operators. While regular searching can split search terms to collect the most possible result, advanced search with quotes makes Google search engine algorithms respect full-string search to collect accurate results.

2.4 Hidden Endpoints.

Hidden endpoints are web application URLs or paths that exist but are not documented or publicly advertised. These endpoints are not intended for general use and may serve various specific purposes, such as use in internal of companies or organizations, testing new features, and serving superusers. Indeed, hidden endpoints are latent risks of being attacked by entities outside the organization. Attackers would like to focus on the endpoints that bring value, the web application's information as endpoints are internal documents or web config files.

Attackers have many ways to collect hidden endpoint archives, search engine Dorking, and webmaster files. For web archives, as I mentioned in the above part, Wayback's crawler downloads web content automatically but, the important thing is the crawler indexes the URLs. This structures the web accidentally so the attacker can collect all the URLs that are collected by the crawler.

Besides URLs collected from web archives, attackers also can gather hidden paths in webmaster files which optimize the way search engine crawlers interact with their websites. There are two webmaster files we should pay attention to: robots.txt and sitemap.xml. robots.txt is a file that is placed at the document root of a web application and provides directives to web crawlers about which pages or sections of a site should not be crawled or indexed. sitemap.xml is a file that contains a list of URLs on a website, along with metadata about each URL.

Technique search engine Dorking does not directly gather URLs as both methods above. This technique is mentioned above with the ability to search endpoints that are hard to find, low rank, and can not be found by regular searching. At the time when web programming templates were created, It marked an explosive growth of web applications' services. But, while all web developers focus on creating services that make people's daily lives more convenient, developers usually are not concerned about existing security problems. To ensure the development progress of the web service and reduce the time for the project's newbie to understand the code, developers rarely change the names of various file types and folder names, especially in documents and configs folders.

Attackers usually take advantage of those default name paths to design vulnerability dorking to search web applications that do not configure well enough to hide those sensitive endpoints.

2.5 Sensitive Data, Server's Config, and Documents.

Indeed, sensitive data is the goal of the reconnaissance process. Pen-testers are internet users who focus on searching for information that ordinary users rarely find and access - this is sensitive data. Based on the reconnaissance workflow in Figure 4, sensitive data is narrowed in data that is stored by web crawlers. It may be hidden data of the target application that is embedded in web content such as API key, employees, or superuser credentials, …

Web application configuration refers to the process of setting up and managing the settings and parameters that control the behavior, security, and functionality of a web application. This includes configuring server settings, database connections, environment variables, and application settings. In other words, a server's config can be the server's setting or instructions for initializing the server or even the server's secret key. If web crawlers provide the pen-testers overview of the web's structure, the config file may contain information about services of the applications application, such as subdomain, virtual host, folder permissions, communication protocol between the application's services, and cryptography secret key.

Documents may stand for images and texts, but the pen-testers usually focus on types of internal documents that bring back rich information such as salary sheets or employee sheets and documents that are created on a web server. Visible data from those documents can be an employee's email, real name, real name, or various kinds of sensitive data. Indeed, attackers regularly demonstrate invisible data inside those files - metadata. Metadata is information that is stored within a file and used to provide context or descriptions about that file. So, metadata of files that are created on a web server is very valuable because it contains critical information such as opera system information, file's creator software version, username of author, embed email, and file path. Each gathered information above opens an attack arrow to the web server system: the attacker can search for CVE based on a version of an opera system or software, phishing email attack, and reveal the web structure on the file path.

2.6 WHOIS.

WHOIS is a public database that stores the information collected when someone registers a domain name or updates their DNS settings. Every domain name that’s been registered belongs to someone, and by default, that registration information is public.

WHOIS is a way of storing that information and making it available for the public to search. The information collected during the domain registration process includes your: Name, Address, Phone Number, and Email Address. As a drawback, it doesn’t display all of the registration information for every domain name, like .com and .net can store more data than that domain. Some domains don't require policy so it's always displayed.

Additionally, WHOIS records also show ownership of subdomains. Subdomains, typically controlled by the primary domain owner, are not normally included in WHOIS records because they are part of a larger domain and need their registration information. Ownership and control of subdomains are typically managed through the primary domain owner's DNS (Domain Name System) settings.

2.7 IP Address Lookup.

An Internet Protocol (IP) address is the unique identifying number assigned to every device connected to the Internet. An IP address definition is a numeric label assigned to devices that use the internet to communicate. In reconnaissance workflow, pen-testers can achieve the target IP address by using a DNS query. Here is a general workflow of DNS record query, this flow is applied for both command and internet browsers.

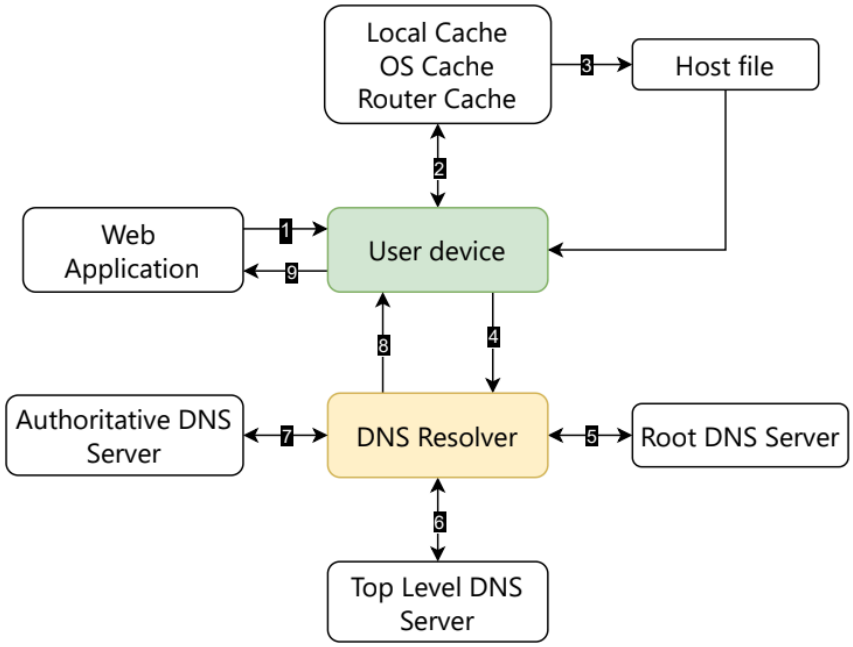

Figure 6: DNS workflow diagram

First, the process requires a web domain to initialize the workflow, then check that domain if it exists in the local cache, OS cache, and router cache. The local cache, also

known as the browser cache, is maintained by web browsers. When a user visits a website, the browser stores the DNS resolution results temporarily. The OS cache, or system cache, is maintained by the operating system and is used to store DNS query results system-wide. When an application needs to resolve a domain name, the OS checks its cache. If a recent entry is found, it uses the cached IP address. The router cache is maintained by the router or network gateway device that connects a local network to the internet. When a device on the network makes a DNS query, the router checks its cache. Last, check the host file, a static text file used by the operating system to map host names to IP addresses. Users changing the content of this file to map the domain with IP may not resolve in public DNS.

Next, the client requests the nearest DNS resolver provided by the ISP (Internet Service Provider), responsible for communicating with several other DNS servers to hunt down an IP address. Because DNS resolvers are supplied by different organizations, they are different systems, so they can have their policy and cache. So, the DNS resolver's cache is also looked up for domain information. Then, the resolver queries to DNS servers in order queries are root DNS server, Top Level DNS server, and Authoritative DNS server to collect the IP address of the provided domain.

When there is a request to complete a domain name into an IP address, the client will send the request to the nearest DNS resolver. The resolver server queries a DNS root server, which is the top level in the DNS hierarchy. When the root server receives a query for the IP address for google.com Google, for example, the root server is not going to know what the IP address is. The DNS root server manages all top-level domains so the root server will direct the resolver to the TLD or top-level domain server for the .com domain. So the resolver will now ask the TLD server for an IP address for the domain based on instructions in response's root DNS server.

The top-level domain server stores address information for top-level domains such as .com, .net, .org, and so on. TLD servers store numbers of DNS records, which specify the authoritative DNS servers for each level domain under the TLD. This particular TLD server manages the .com domain which google.com is a part of. So when a TLD server receives a query for the IP address for google.com, the TLD server is not going to know what IP addresses for google.com but they store information about the authoritative DNS servers that are responsible for providing the IP addresses for these domain names. So the TLD will direct the resolver to authoritative name servers. So once again the resolver will now ask the authoritative name server for the IP address for google.com. Authoritative name servers or servers are responsible for knowing everything about the domain which includes the IP address.

Authoritative DNS is the system that keeps official records corresponding to domain names such as IP addresses. Domain names are the human-readable names of IP addresses that direct applications such as browsers to websites such as google.com. The authoritative server responds with a DNS record that includes the IP address and the DNS resolver server caches the IP address and sends it to the browser. So now the client or browser gets the domain's IP address.

There is another way to capture target IP addresses. As I mentioned earlier in the **Web Crawler** part, third-party crawlers store web content for extracting data purposes and they also store the older version of web content or the IPs that are used to point to domains of target web application.

2.8 Google Analytics ID.

Google Analytics is used to track website performance and collect visitor insights. It can help organizations determine top sources of user traffic, and gauge the success of their marketing activities and campaigns. Every Google Analytics website has a unique tracking ID (a UA or GA ID). This ID is included in the website's HTML or JavaScript code and can often be found by simply viewing the page source or intercepting network traffic. Sometimes, the same Google Analytics ID is used across multiple websites. By finding other sites using the same ID, pen-testers can identify related domains or subdomains. If a Google Analytics ID is configured to track multiple domains, it might indicate that those domains are related or controlled by the same entity.

2.9 Domain Discovery.

According to Figure 4, although domain discovery is the technique mentioned first in **Web Crawler** and also appears in **IP Adress** Lookup and **Google Analytics ID**, I would like to spend this part centralizing techniques of discovering domains such as extract from third-party web crawlers, google dorking, reverse IP lookup and domain discover with SLL certificates.

"There's nothing you can't find; it's just that you haven't found it yet." Google Dorking provides a set of search operators to help users optimize their search. This means users can search subdomains of the web application by combining and using the Google searching operators Smart and Creative.

Although using web crawlers similar to search engines, third-party services usually do not provide searching services, they provide extracting data services instead. Pen-testers can reveal web target applications based on endpoints that are extracted from crawlers' data. Indeed, those endpoints include domain and subdomain so pen-testers can collect domain and subdomain by handling and filtering that data. This involved

developing a technique for reverse IP lookup and domain enumeration using query crawlers' services.

In modern life, most web applications require and need a secure and private connection to communicate with other devices or services in the internet space, and HTTPS protocol is created. HTTPS uses an encryption protocol to encrypt communications. The protocol is called Transport Layer Security (TLS), although formerly it was known as Secure Sockets Layer (SSL). This protocol uses an SSL certificate which is stored in a web server to create a secure connection or searchable database of certificate transparency logs. Certificate Transparency is an Internet security standard and open-source framework for monitoring and auditing digital certificates. When web administration wants to register SSL certificates, the administrator sends a CSR (Certificate Signing Request) which contains the domain name and other information like organization name, location, and public key to a trusted root Certificate Authority (CA). If domain owners can secure multiple domains and sub-domains with one SSL certificate, they would use the Subject Alternative Name (SAN) certificate to cover all managed domains with a single SSL certificate. This encrypts processing management is the idea of technique domain discover with certificates. This technique usually has high accuracy because all the subdomain that are extracted from the certificate is added by the SSL register or the domain owners.

4. Active Reconnaissance.

Opposite to passive surveillance, the active process directly interacts with the target web application to collect useful information instead of searching in legal public sources. Meanwhile, they still have the same purpose, the active reconnaissance process continues gathering and extracting value data to detect weak points and reveal vulnerabilities. Besides some familiar fields like IP, domain & subdomain, and endpoints, this process also collects data such as source code, databases, and file storage.

4.1 Active Reconnaissance Workflow.

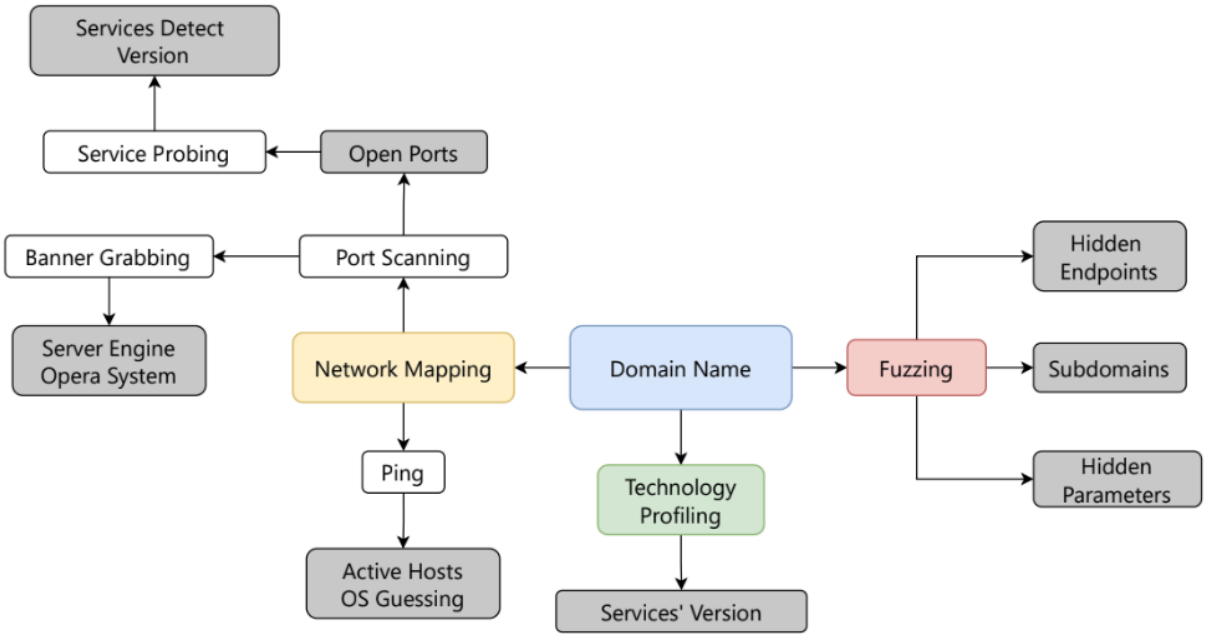

Figure 7: Active Reconnaissance Diagram.

Like OSINT workflow, the active workflow also initializes with a web domain as a starting point as I designed in Figure 7, which mentions using three main techniques: fuzzing, network mapping, and service fingerprinting. Fuzzing is a basic and well-known technique, a world list attack. At the same time, pen-testers test the system by preparing payloads and determining the information, and structure of the web application based on behavior and response. Technology profiling focuses on identifying technologies from the web layer by analyzing HTTP headers, cookies, and HTML content. Last, network mapping is discovering the topology of a network, including the devices connected to it, their IP addresses, and the relationships between them.

4.2 Ping.

According to Figure 7, I would like to start with the yellow branch first - Service Fingerprinting and dig into the network mapping part. The reason I start at this stage first is the attribute of the active reconnaissance process - direct interaction with the web application. So if the target is disabled to access from the internet or offline, this thing should terminate all the steps behind. This technique involves sending a series of ICMP echo requests to a range of IP addresses to determine which hosts are active on the network. This technique can be used to identify the IP addresses of hosts that are alive and responding to network requests and use the Internet Control Message Protocol

(ICMP) to send and receive messages. The package is sent from the source host to the target host over the network using IP. The ICMP Echo Reply packet mirrors the Echo Request, changing the ICMP type to 0 (Echo Reply) and keeping the identifier, sequence number, and payload data intact.

There is a technique that can take advantage of the command ping - guessing target opera systems of target based on TTL (Time-To-Live). Time to live refers to the amount of time or “hops” that a packet is set to exist inside a network before being discarded by a router [8]. Indeed, if the ICMP Echo Reply message does not contain the service’s IP address, this technique could fail if pen-testers depend only on the ICMP Echo. But ICMP operates at the network layer (Layer 3) of the OSI model and it does not operate independently. Hence, it has to use IP (internet protocol), which is the primary protocol at the network layer to communicate within networks then some IP headers are encapsulated in the reply package. Besides, The TTL value varies depending on the version of an operating system and device so this is the key to detecting the target’s OS based on the ping command result. As I mentioned before, this technique usually guesses because there are many cases in which the TTL does not match the standard value so pen-testers have to guess the OS based on that TTL near which point.

4.3 Port Scanning.

A port is a virtual point in the network, managed by a computer's operating system. Each port is assigned a specific process or service, allowing computers to communicate between different kinds of traffic.

Port scanning is the technique to reveal vulnerabilities footprint in a network that pen-testers can use to gain access. Port scanning aims to determine the IP addresses, hosts, ports, and which ports are open and sending or receiving data in a network. Port scanning can also reveal firewalls and other security measures between a server and a user’s device.

In this report, I focus on the technique of Nmap which is a Linux native command and powerful at port scanning. There are six main kinds of port scanning techniques TCP connect, TCP SYN, TCP ACK, TCP NULL, TCP FIN, and TCP XMAX [9]. Each technique involves sending and receiving packages to determine whether that port is open, closed, or filtered. The default mod of Nmap to scanning port is TCP connect which is based on a three-way handshake mechanism.

Transmission Control Protocol (TCP) is a connection-oriented protocol for communications that helps in the exchange of messages between different devices over a network. The Internet Protocol (IP), which establishes the technique for sending data packets between computers, works with TCP. The position of TCP is at the transport

layer of the OSI model. TCP also helps in ensuring that information is transmitted accurately by establishing a virtual connection between the sender and receiver.

Nmap uses a three-way shake to create a TCP connection to the target to determine the status of the port. First, the client initiates the connection by sending a SYN packet to the server. This packet includes an initial sequence number (ISN) that the client chooses randomly. While the target receives the SYN packet, the server responds to it by sending back a SYN-ACK packet. This packet contains the client's sequence number plus one (ISN + 1) and the server's initial sequence number, chosen randomly by the server. In this step of the process, if the server responds to an RST packet, the TCP connection cannot be established because the port is closed or not available. If the client can not receive anything, this means there is a firewall or packet filter in place between the client and server, then the port's status is filtered. To complete the TCP connection, The client responds to the SYN-ACK with an ACK packet which contains the server's sequence number plus one.

4.4 Banner Grabbing.

Banner grabbing is a method used by security teams and hackers to gather information about network computer systems and software type and version by querying open ports. A banner is a piece of information displayed by a host that provides details about the service or system, such as its software version, operating system, and other relevant facts. The text contained in a banner can help identify the software name, version numbers, and operating systems running on network hosts. This information can then be used to discover potential vulnerabilities in the network.

Banner grabbing has two primary methods: active and passive. Active banner grabbing involves directly connecting to a server, while passive banner grabbing gathers information without direct interaction, helping to avoid detection. In the concept of active reconnaissance, I would like to clarify the active banner grabbing. Active banner grabbing involves sending specially crafted packets to a target's remote host and analyzing the responses. This process can uncover key details about the versions of running services and operating systems.

4.5 Service Probing.

Service probing, also called service grabbing, is a crucial step in network reconnaissance. It identifies and fingerprints services running on a target system's open ports. This process involves sending crafted network packets to these ports and analyzing the responses. The target system often returns information in service banners, revealing details like service type, version number, and underlying software.

Nmap, a powerful network scanning tool, is commonly used for service probing due to its effectiveness and reliability. It not only detects open ports but also probes them to identify running services. Nmap uses a comprehensive set of signature databases and advanced fingerprinting techniques, providing detailed service information. This makes it an ideal choice for thorough service analysis.

4.6 Technology Profiling.

Technology profiling refers to identifying and analyzing the technologies used by a target system, application, or organization. This can include web servers, programming languages, content management systems (CMS), frameworks, databases, and other software or hardware components. This stage discovers the specific technologies in use, such as Apache or Nginx for web servers, PHP or Python for server-side scripting, and MySQL or PostgreSQL for databases while adding its version. Narrowed to the scope of web applications reconnaissance, this technique focuses on determining the signature of the in-use server's component in HTTP or HTTPS response. By analyzing the behavior of response and collecting the terms which could match with kinds of technical components. Those terms are signature which only that component or a specific version has.

4.7 Fuzzing.

Fuzzing is a dynamic testing technique used in the active reconnaissance process to deliver unexpected or unsafe data inputs into a target system to find vulnerabilities, weaknesses, or misconfigurations. Fuzzing, as used like a kind of word list attack, is the process of methodically changing certain sections of a URL, form field, or API request with different payloads (random strings, common paths, or special characters, for example) to probe the system for unexpected behaviors. Finding hidden endpoints, input validation problems or exploitable vulnerabilities that static analysis might not be able to find readily is the aim.

Endpoint fuzzing is the clearest example of the word list attack. Endpoint fuzzing is a specialized form of fuzzing that focuses on discovering hidden or undocumented API endpoints within a web application or service. In this process, automated tools send payloads that are self-prepare or customed by pen-testers to an application's API endpoints to trigger unexpected or erroneous behavior. The goal is to identify endpoints that may not be publicly exposed or well-documented but could still be accessed and potentially exploited.

Although domain discovery in the passive reconnaissance process is very effective but still depends on third-party services to archive data of domains and subdomains. In

the concept of an active process, pen-testers are allowed to declare a fuzzing process to reveal hidden subdomains proactively.

There is a higher level of fuzzing. While endpoint and subdomains are based on the web's structure, parameters fuzzing requires analysis of the way that the web applications interact with the parameters. Because an active host will handle the input from a parameter in an existing endpoint the display on the page changes to return the handle data's result.

IV. Implementation.

In the Implementation section, I would like to realize the results of the reconnaissance workflow research and systemize it into an automatic reconnaissance tool, ensuring that the tool could handle tasks ranging from OSINT-based data collection to more intrusive active probing of web applications. This section details how the conceptual workflows were translated into a functional, terminal-based tool designed to automate both passive and active reconnaissance tasks in Figure 4 and Figure 7. This part also presents how the gathered data is collected, extracted, and stored.

1. Overview of System Architecture.

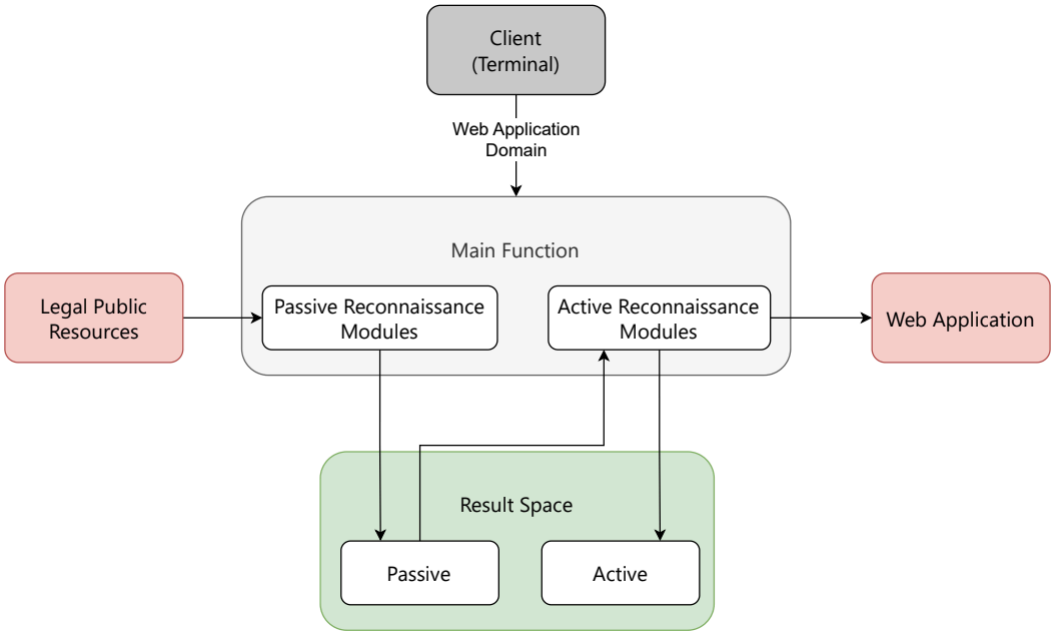

Figure 8: System Architecture.

All of the reconnaissance methods that are discussed in the Methodology session are centralized in the module structure to create the tool. Modular architecture was used in the tool's implementation to provide scalability and flexibility. Since each module, passive and active reconnaissance was created separately, adding new features and updating them was simple. The design reduces the requirement for custom-built solutions when well-established alternatives are available by utilizing existing libraries and APIs to expedite development and guarantee dependability. This method shortens the tool's development time and guarantees that it will be simple to maintain and expand in the future. All of the reconnaissance results from the entire procedure are kept in a folder called the result space. Result space is a folder that is generated from the reconnaissance process of the automated tool. For each target domain, the result space is named by the day that running the script and is continued with the target domain name. In the main result space, there are two main folders active and passive, which are responsible for saving data returned from the corresponding modules. Two distinct folders hold the collection of outcomes from the two major modules.

According to the ANNEX part, the tool is developed with a terminal-based structure with results such as process tracking and time-consuming are displayed in the terminal. In contrast, data as a result of the reconnaissance process is stored by raw data type in the result space.

2. Passive Reconnaissance.

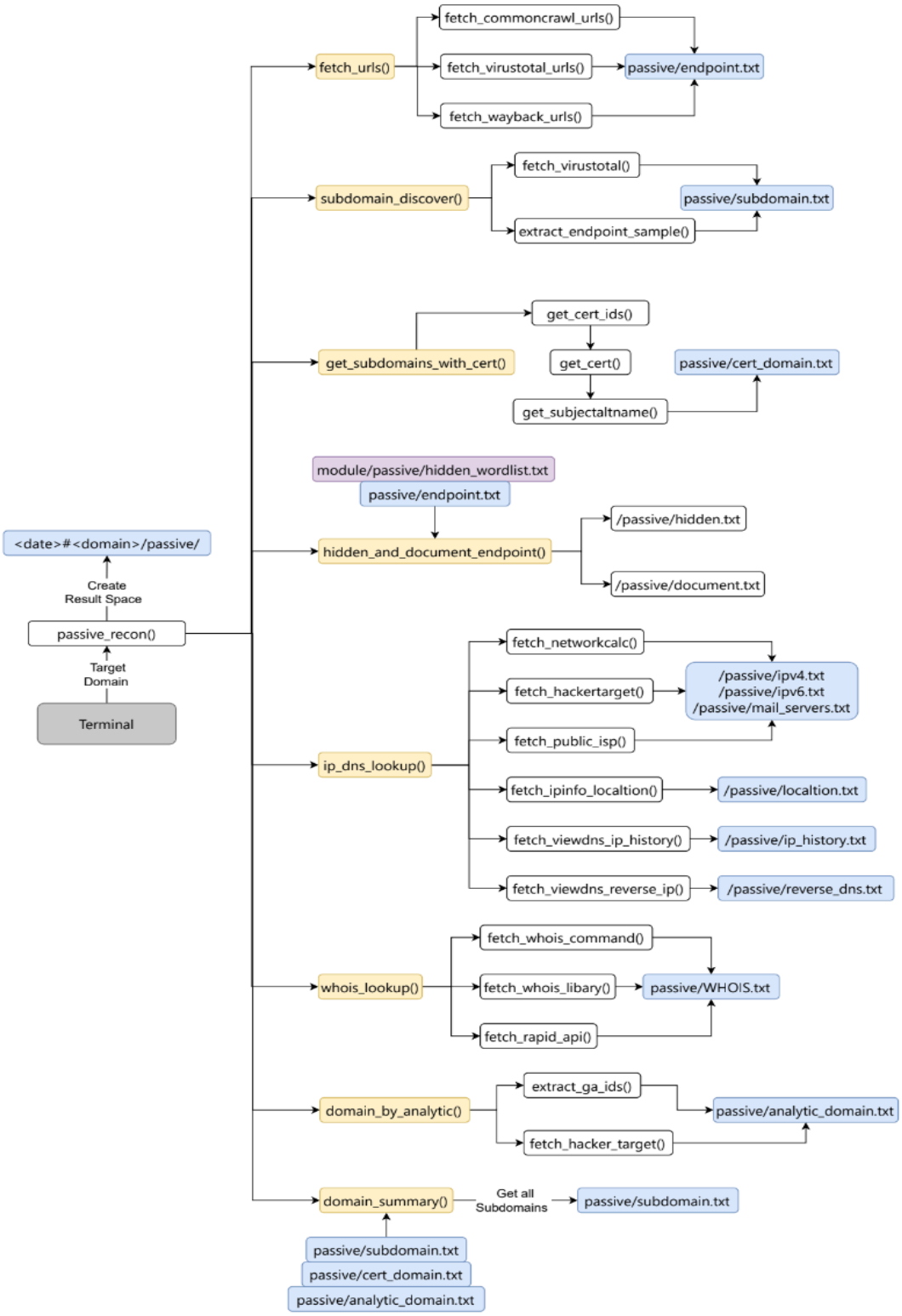

Figure 9: Passive Reconnaissance Function Structure and Data Sequence.

The passive reconnaissance module was implemented using a combination of third-party APIs and custom scripts which involved directly fetching data from public sources or extracting data ready to exist in the resulting space to gather information without actively interacting with the target. Figure 9 represents the passive reconnaissance module cell in Figure 8. Figure 9 is depicted as sub-cells linked together by arrows to represent the overall structure of the passive reconnaissance module, the structure and operation of the functions, and the distribution and storage of the acquired data.

2.1 Web Crawler & Dorking.

The Web Crawler was designed to systematically explore and retrieve information from the target web application, focusing on identifying publicly accessible resources, links, and data. Rather than building a custom web crawler from scratch, which would require significant development time and effort, the decision was made to utilize several existing, well-known third-party crawlers' APIs to provide data for the reconnaissance process.

The Wayback Machine is a powerful tool for accessing historical snapshots of web pages which can reveal previously exposed endpoints, resources, and other valuable information that might no longer be available on the current live site. Its primary strength lies in its ability to capture and preserve entire web pages, including images, videos, JavaScript, and CSS files, which contribute to the visual and functional integrity of the site.

In addition to the Wayback Machine, the Common Crawl API was integrated to provide access to large-scale web archives. Since Common Crawl is tailored for developers, it’s easier to integrate into automated tools that need to process and analyze web data at scale. The Wayback Machine might not capture every single page or may have gaps in its historical coverage for certain sites. Common Crawl can sometimes fill these gaps by providing additional snapshots or by capturing pages that the Wayback Machine might have missed.

In addition to using a third-party crawler, I implemented a Google Dorking form to help filter crawler data based on the target domain. This form allows the tool to narrow the results and focus only on URLs and resources relevant to the analyzed domain. By filtering the data this way, the tool can more effectively identify relevant endpoints and resources.

The combination of both Wayback Machine and Common Crawl gives me an impressive breadth and depth of information, which is key to enhancing the

discoverability of web application endpoints. The list of combined endpoints is stored in a text file so that the following modules can be used.

2.2 Hidden Endpoints, Server's Config, and Documents.

Although it has been mentioned in the Methodology section as two individual stages extracting sensitive information in page content consumes too much effort. So in this stage, I aim to list out all endpoints that potentially embed critical data.

The hidden endpoints component is designed to identify endpoints within the target web application that are not immediately visible or accessible through standard navigation paths. After collecting the list of endpoints from the crawler in the previous stage, the next step is to filter these endpoints using a predefined wordlist. This wordlist includes keywords that are often associated with sensitive or hidden resources and it is compared with the discovered URLs to identify matches. Once the filtering process is complete, the tool generates a refined list of endpoints into a text file. This thing not only clarifies the set of results but also supports the following process.

2.3 WHOIS.

The WHOIS component is responsible for retrieving and analyzing domain registration information to provide insights into the ownership and background of the target domain. Because WHOIS services on the internet all equip captcha to prevent bots and automate purposes, so I suggest using three ways to get WHOIS data: Linux native command - whois, whois python3 library, and WHOIS API. The priority level of using these tools corresponds to the way I list them. The reason is I prioritize the solution which is performed in native using native command and library but whois command provides raw data and richer data than the python library. The whois command is a standard utility available on Unix-like systems that allows users to query WHOIS databases from the command line. The Python library is a lightweight tool that allows for the programmatic retrieval of WHOIS information directly from WHOIS servers. The WhoisXML API provides a robust and reliable way to access detailed WHOIS information for domain names. This API is used to query a centralized database of domain registration records, offering data such as registrant details, registration dates, and domain status.

2.4 IP Address Lookup.

The IP Address Lookup component focuses on gathering information related to the IP addresses associated with the target domain. This stage focuses on using the DNS resolver’s result effectively by extracting the embedded data in the response DNS record

such as IPv4 of A record, IPv6 of AAAA record, mail server of MX record (Mail Exchange), and extracting location based on IP range. I suggest two ways to achieve those data using the DNS resolver library of Python or using third-party APIs. The library is designed to facilitate DNS lookups within a programmatic environment, allowing you to resolve domain names into the DNS record data by requesting to Google ISP or Cloudflare ISP with the corresponding records., I also requested several APIs to enrich the data sets of DNS records and fill some fields that are unable to be asked by public ISPs.

2.5 Google Analytics.

The Google Analytics component focuses on extracting and analyzing Google Analytics IDs (UA-IDs) from the target domain's web content. I specific the home page and then download its HTML content. After the web content is downloaded, the tool extracts the ID embedded in the script tag of the HTML content. The API service already downloads the HTML content in the crawling process and then extracts the Google Analytics ID for searching for the same entity's purpose.

2.6 Domain Discovery.

The Domain Discovery component aims to identify and enumerate domains and subdomains associated with the target or hosted on the same IP address. The domain discover module has the most diverse search methods of all the modules. First, I extract the domain from the endpoints list that was already downloaded from the crawlers' API in the previous stage. The form of endpoints is a full path structure so it contains all possible domains and subdomains, after extracting I indexed and removed all duplicate results. Another way the tool is used is to extract domains from the SSL certificates. By querying SSL certificate transparency logs, the tool can extract this information. In this process, the tool searches the list of certificate IDs in the certificate transparency logs with the web domain and then downloads all the certificates based on those IDs. After finishing the download process, the tool decodes the SSL certificates based on the algorithms mentioned in the transparency logs database. Those new decoded domains are updated in the previous domain set.

3. Active Reconnaissance.

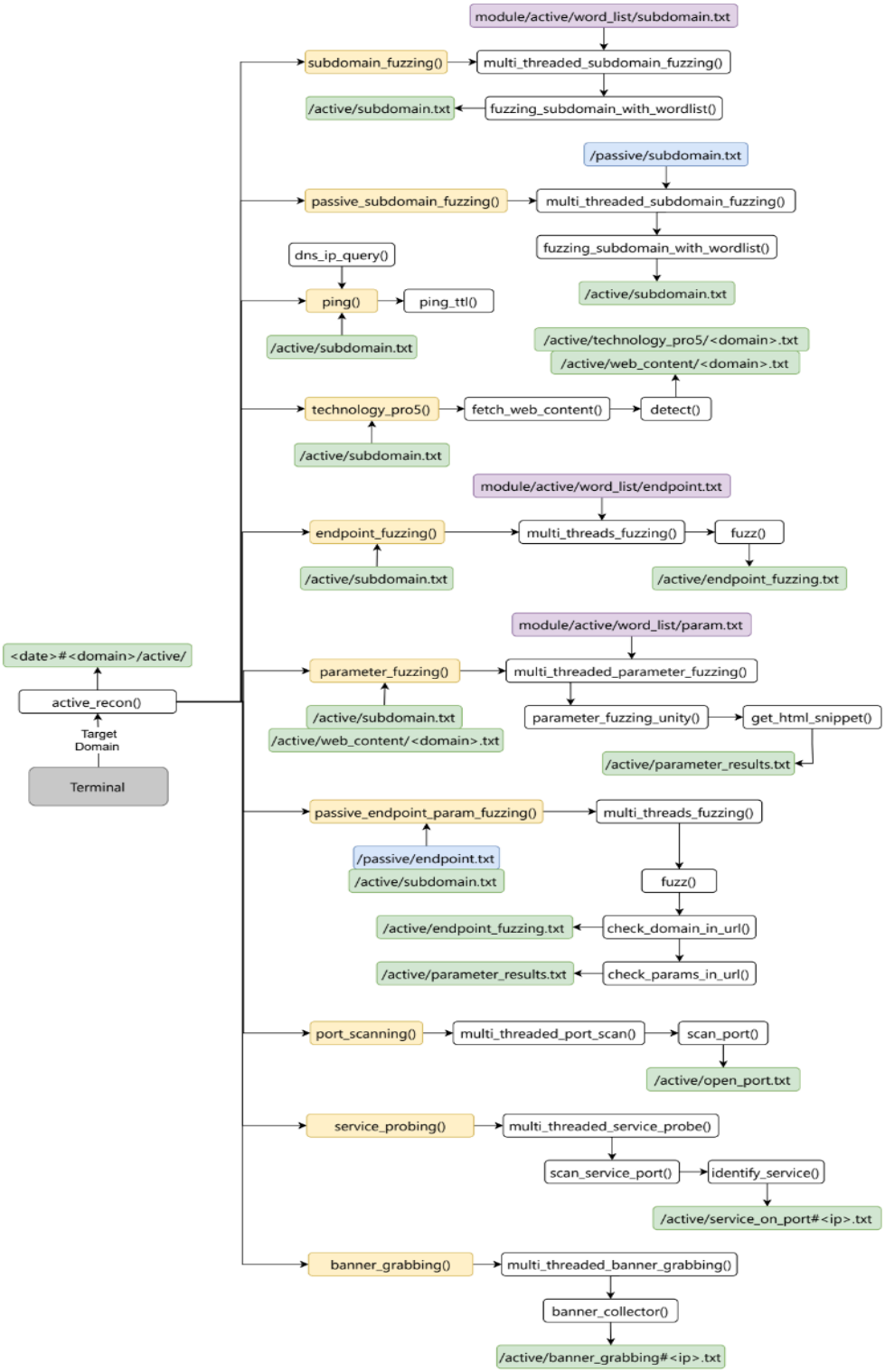

Figure 10: Active Reconnaissance Function Structure and Data Sequence.

The active reconnaissance module was implemented using fuzzing and scanning tactics which involved directly delivering the customed payloads corresponding to that module. Besides the effort of building customed payloads, I also use several Linux native commands to prevent missing results caused by errors and enrich data in the result space. In some Modules, I consider test data ready to exist in the result space which is created in the passive reconnaissance process to verify data accuracy. The active reconnaissance module cell in Figure 8 is seen in Figure 10. The overall structure of the passive reconnaissance module, the organization and functioning of the functions, and the distribution and archiving of the collected data are all represented by the sub-cells connected by arrows in Figure 10.

3.1 Ping.

The ping sweeping process begins by resolving the target's domain to an IP address using the DNS resolver Python library. This step ensures that the IP address is collected even if users do not perform the process of passive reconnaissance before. Then, the tool uses Linux native commands, such as ping, to identify whether the target service is down or not. I suggest using the ping command instead of building a script to send ICMP requests from scratch because the user Python in the Linux system requires root permission to send raw ICMP packages while the command is manually installed and trusted by users.

The tool sends ICMP echo requests across the range and analyzes the responses to determine which hosts are alive. Additionally, the tool leverages the Time-to-Live (TTL) values returned in the ICMP responses to infer the operating system of the active hosts, providing further insight into the target network's composition.

3.2 Fuzzing.

The fuzzing module in the tool is designed to discover hidden endpoints, subdomains, and vulnerable parameters within the target web application. The fuzzing process is powered by wordlists downloaded from the **SecLists** repository, which serve as payloads for the different types of fuzzing operations. The tool uses the wordlists to brute-force potential endpoints, subdomains, and parameters by appending each wordlist entry to the base URL and making HTTP requests then the results are written to a file in the result space.

In subdomain fuzzing, the process is performed by prefixing each wordlist entry to the main domain and attempting to resolve the DNS entries. If the user runs the tool for passive reconnaissance with the same domain, this module also tests all subdomains collected in the passive process to enrich subdomain results in the active process.

In endpoint fuzzing, the tool adds each word from the wordlist to the base URL and sends HTTP requests to find hidden pages or files. It looks for valid responses (status 200) and saves successful results. The difference is that the automated tool not only fuzzes the root domain but also the subdomains that are fuzzed in the subdomain module. Finally, this module fuzzes based on the available endpoints from passive reconnaissance, the endpoints being tested are forced to contain domains that have been authenticated from the previous module.

With parameters fuzzing, I also declare a fuzzing process with parameter word lists. The payload which is loaded to this parameter is fixed to 1. The payload can be anything because if that parameter exists, it will handle the input and then may change the content on the page. This process also occurs in the endpoint fuzzing module, fuzzing in all confirmed endpoints and then extracting parameters from endpoints that are listed in a passive process that is confirmed by endpoint fuzzing modules.

3.3 Technology Profiling.

The technology profiling process begins by downloading the HTML content of the target web page. This content is then analyzed using a data file provided by Wappalyzer, which contains regular expressions and specific terms designed to detect the footprint of various technology components within the HTML. The tool parses the HTML content, applying these regex patterns to identify the technologies used by the web application, such as content management systems, JavaScript frameworks, analytics tools, and more. The tool's unique feature is that it performs profiling technology on the root domain and all verified subdomains.

3.4 Port Scanning, Service Probing, and Banner Grabbing.

Port scanning, service probing, and banner grabbing have been combined into a single process for more efficient monitoring. The tool uses Python's socket library to create TCP connections to specified target ports, handling all three tasks in a streamlined workflow. This approach eliminates redundant operations, making the scanning process faster and more efficient.

The process starts with port scanning, where the tool attempts to connect to a series of specified ports to check if they are open, closed, or filtered. Using multithreading, multiple ports are scanned simultaneously. Once an open port is identified, the tool moves to the next stage without delay.

Service probing will only be performed on open ports. Using the same TCP connection, the tool will send specific requests to the service running on the port. By

combining port scanning and service probing in one connection, the tool reduces unnecessary operations and speeds up the process.

After the service probe, the tool continues with the banner collection, still using the same TCP connection. The tool requests and retrieves service banners, which provide information about the software and services running on the port. The collected banners are appended to the port information in the data store. This method ensures no additional connections are made, maintaining efficiency throughout the scan. The banner is part of the version data if the pen-tester manually scans the port with Nmap.

V. Results.

This section outlines the outcomes of applying the developed reconnaissance tool in real-world testing scenarios. After executing the reconnaissance process using manual and automated methods, the tool is expected to retrieve various critical information about the target system. The results will be compared against the expected outcomes as a manual survey to evaluate the tool's accuracy, and efficiency. Additionally, tool testing will be demonstrated through terminal-based outputs and results from multiple test cases. In this section, I would like to survey three random domains for manual techniques and use the automated tool to compare results, time, and performance.

1. Expected Result From Manual Process.

Manual or traditional reconnaissance is a data collection process with the object considered a web application. This process will be performed in the same order and steps as automated reconnaissance as mentioned in the methodology section. This marks a standard for evaluating the automated process. In case study one, I want to focus on the practice, testing, and measurement of the results and time of the manual reconnaissance process in both passive and active aspects, but the tools and techniques that I used in this process are also mentioned. In case study two, it is the measurement and evaluation of the results of both passive and active reconnaissance of the automated tool, and the comparison of the results between manual reconnaissance and the reconnaissance support tool.

1.1 Passive Reconnaissance.

	Domain 1		Domain 2		Domain 3		Average	
	Result	Time	Result	Time	Result	Time	Result	Time
Endpoints & Documents	419	20m	532	16m	33	12m	328	16m
Domains	31	6m	156	6m	58	3m	81	5m
IP Address	1	3s	1	3s	2	8s	1	4.7s
WHOIS	1	3	1	4	1	2	1	3s

Table 1: Summary table of results of Passive Reconnaissance data of Manual Process.

The passive reconnaissance phase aims to collect important data through a variety of methods: web crawling, Google Dorking, WHOIS queries, IP address lookups, and domain discovery. Table 1 is designed with columns and rows, while rows are present for techniques and phases in the passive reconnaissance, columns are present with the main column being a list of target web applications, and an average column and 2 sub-columns present the number of captured results and consumed time for the whole phase. In the result columns, the first three rows present the number of endpoints that are revealed as hidden endpoints and documents, the number of discovered subdomains, and the number of IP addresses while the last row presents the possibility of collecting WHOIS record (it would be 0 if no WHOIS record found). In time columns, while the first two rows present the time cost for those two techniques in minutes, the remainder present time-consuming in seconds.9

For domain enumeration, I used a tool called **Subfinder** to search for subdomains associated with a given domain, illustrating passive domain discovery. For IP lookups, I used the **nslookup** command, a network administration tool used to query DNS servers to retrieve mappings between domain names and IP addresses as well as other DNS records. For tracking the geographic location of IP addresses, I used the web-based tool nslookup.io, which provides a convenient interface for IP analysis.

WHOIS record collection was done manually using web-based tools. Typically, WHOIS records are searched via Whois.com. However, since all three targets identified in this project have domains with the **.vn** TLD, I chose to use the regional resource www.vnnic.vn/whois-information to ensure accurate and region-specific data.

1.2 Active reconnaissance.

	Domain 1		Domain 2		Domain 3		Average	
	Result	Time	Result	Time	Result	Time	Result	Time
Ping	Alive	1s	Alive	1s	Alive	1s	Alive	1s
Subdomains Fuzzing	5	310s	33	292s	2	194s	13	265.3s
Endpoints Fuzzing	10361	501s	195839	655s	65	38s	68755	398s
Parameters Fuzzing	164	875s	124	742s	32	266s	106	627.6s
Technology Profiling	16	0.71s	3	0.71s	18	0.72s	12	0.71s
Port Scanning	306	1h33m	671	39m	579	1h04m	518	65.3m

Table 2: Summary table of results of Active Reconnaissance data of Manual Process.

In the active reconnaissance phase, the expected results include detailed data collection, which can be achieved through direct interaction with the target system using techniques such as pinging, port scanning, banner grabbing, service probing, technology profiling, and fuzzing. Table 2 represents the data in the same way as Table 1. The different points are all result columns that present the number of captured data such as the number of subdomains, number of endpoints, number of parameters, number of detected technology, and the number of open ports that are detected with a service.

The ping procedure confirms which hosts are alive and reachable on the network by sending an ICMP response request. The expected result is a list of live hosts along with their associated IP addresses, along with Time to Live (TTL) value analysis to potentially infer the host operating system. This step ensures that further reconnaissance only focuses on systems that are alive and reachable.

There are three main techniques in the fuzzing phase: endpoint fuzzing, parameter fuzzing, and subdomain fuzzing. For all three fuzzing techniques, I use the FFUF tool to brute force the necessary information about the three phases. The order is subdomain, endpoint then parameter because I need to identify the subdomains first and then find the paths and parameters of those subdomains.

The technology profile identifies the underlying technologies and frameworks used by the target system. At this stage, I use **Wappalyzer** to analyze and collect the technologies that the web application is using. This includes web servers (e.g. Apache, Nginx), content management systems (CMS), programming languages (e.g. PHP, Python), and frameworks (e.g. Django, Laravel). Some technologies also come with version information. This technique is applied across all subdomains to optimize the information.

Port scanning occurs simultaneously on all subdomains, checking open ports and gathering information about services that are likely to run on those ports.

2. Automated Tool Testing.

This section presents a comprehensive evaluation of the reconnaissance tool by applying it in different scenarios. The tool will be tested in both passive and active reconnaissance modes, and the results will be compared with known data or other tools to measure its effectiveness.

2.1 Test Case 1: Passive Reconnaissance.

In the passive reconnaissance stage, this tool performs better in many ways but in some aspects, it has to ignore the time factor to collect as much information as possible. The result set of this stage is indexed into files, and each type of data the tool receives is saved in separate data files.

	Domain 1		Domain 2		Domain 3		Average	
	Result	Time	Result	Time	Result	Time	Result	Time
Endpoints	96893	49.50s	758057	200.5s	759101	373.12s	538683	207.0s
Hidden Endpoints	180	49.83s	1204	202.2s	14144	376.4s	5182	209.5s
Documents	6556	49.83s	7749	202.2s	95	376.4s	4800	209.5s
Domains	23	142.1s	735	154.2s	102	151.41s	286	149.9s
IP Address	1	2.69s	1	8.30s	2	4.36s	1	4.36s
WHOIS	1	1.33s	1	1.34s	1	1.69s	1	1.45s

Table 3: Summary table of results of Passive Reconnaissance data of Automated Tool.

The passive reconnaissance phase of the automated tool contains several modules: endpoints fuzzing, parameters fuzzing, subdomains fuzzing, ping, technology profile, and port scanning. Table 3 follows the same structure as the two tables mentioned before, especially Table 1. The row and column structure of Table 3 is almost the same as Table 1 as Table 3 has replaced the first row of Table 1 with 3 rows to contain the values of endpoints, hidden endpoints, and documents. In the result columns, the corresponding rows still show the number of results obtained, only the row of WHOIS record shows the ability to collect this type of record. The time columns show the time that each process takes and are shown in seconds.

The **Average** column in the table provides the average values for both the results and the execution time across the three domains. This column serves as a general summary to evaluate the overall performance of the passive reconnaissance tool across different domain types. This data will be compared with the same data in Table 1 to compare the efficiency in results and time of both methods.

For Hidden Endpoints and Documents, the automated process significantly outperformed the manual process. In the automated process, an average of 5,182 hidden endpoints and 4,800 documents were found in approximately 209.51 seconds. In contrast, the manual process only identified an average of 328 hidden endpoints and documents in 16 minutes. This shows that the automated method is faster and more efficient in finding results.

In Domain Discovery, the automated process also collected more results with an average of 286 domains found in 149.94 seconds, while the manual process discovered an average of 81 domains in 5 minutes. Although the manual method showed a smaller difference in time, the automated process still proved to be more efficient, finding more domains in less time.

For IP and WHOIS Lookups, both processes yielded the same results, each finding one IP address and one WHOIS record for each domain. However, the manual process took an average of 4.7 seconds to discover the IP and 3 seconds to look up the WHOIS, slightly longer than the automated process (4.36 seconds for the IP address and 1.45 seconds for the WHOIS).

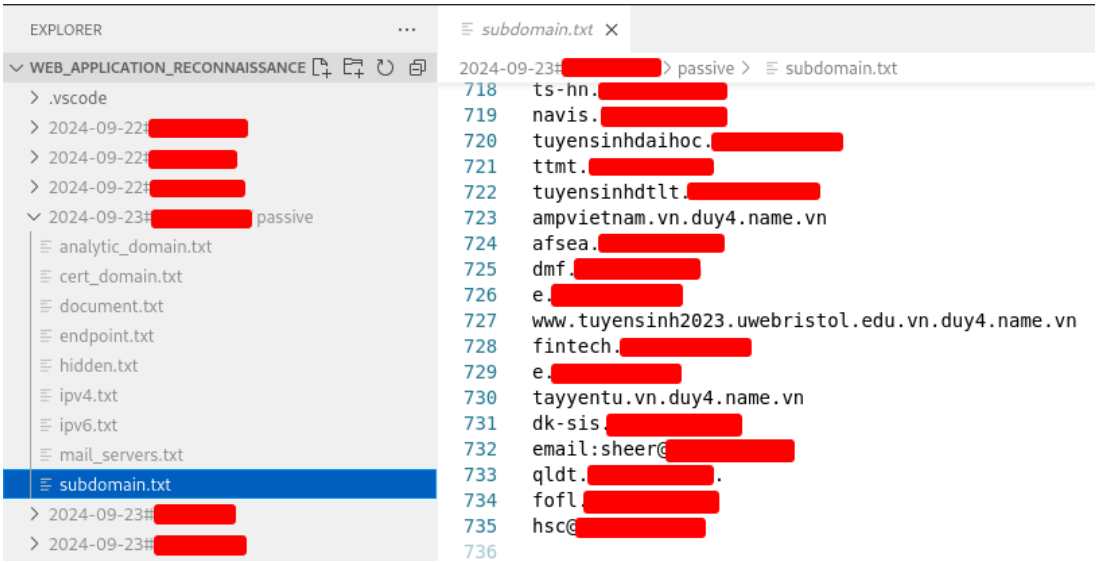

Figure 11: Example of Result Space of Automated Tool Passive Reconnaissance.

Figure 11 shows the result space for passive reconnaissance on the second domain. As shown in Figure 8, the result space has two parts: active and passive. In Figure 11, the right half shows the results of domain discovery, with all passive reconnaissance techniques producing results stored in text files.

2.2 Test Case 2: Active Reconnaissance.

	Domain 1		Domain 2		Domain 3		Average	
	Result	Time	Result	Time	Result	Time	Result	Time
Subdomains Fuzzing	12	67.73s	67	53.8s	15	52.95s	31	58.1s
Endpoints Fuzzing	17481	520s	67663	1236s	361002	4800s	148,75	2,18s
Parameters Fuzzing	9	651.1s	32	1032s	10	4958s	17	2,214
Ping	Alive	2.2s	Alive	4s	Alive	1.18s	Alive	2.46
Technology Profiling	13	6.37s	8	8.5s	8	1.04s	10	5.3s
Port Scanning	122	202s	52	90s	39	78.3s	71	123s

Table 4: Summary table of results of Active Reconnaissance data of Automated Tool.

In the active reconnaissance phase of the automated tool, the expected results include detailed data collection, which can be achieved through direct interaction with the target system using techniques such as pinging, port scanning, banner scraping, service probing, technology profiling, and fuzzing. Table 4 which has the same structure as Table 2, except that all values in the time columns are expressed in seconds shows the data collection process and a summary of this process, in this section, I will also analyze the metrics and compare them with the manual reconnaissance process.

At Ping, both procedures produce identical results for the three domains, with each domain being alive. The time taken to perform this task is always 1 second for both methods.

For Subdomain Fuzzing, the automated process discovered more subdomains across all domains than the manual method. For example, the third identified only 2 subdomains manually but discovered 15 subdomains using the automated method. Similarly, the second domain found 67 subdomains in the automated process, while the manual process discovered only 33 subdomains. The time required to fuzz subdomains was also significantly faster in the automated process, averaging 58.18 seconds, compared to the manual process, which took an average of 265.33 seconds. This highlights the effectiveness of automation in quickly discovering subdomains.

The results for endpoint fuzzing show a clear contrast between the two methods. The automated process identified more endpoints, especially for the third domain, where it found over 361,000 endpoints, compared to just 65 endpoints in the manual process. The time difference was also significant, with the manual process taking 38 seconds for the third domain, while the automated process took over 4800 seconds. This suggests that while manual fuzzing may be faster, it is less comprehensive, potentially missing many endpoints that the automated tool could have caught through its comprehensive approach.

For Parameters Fuzzing, the automated process was again performed more comprehensively. For example, the third domain identified 10 parameters in the automated process, while the manual process only detected 32 parameters, although the time required was longer in the automated method. On average, the automated process took 2.214 seconds compared to the manual process, which averaged 627.67 seconds. This may indicate that the automated tool is more thorough, at the cost of increased time consumption, while the manual process, although faster, is likely to miss certain parameters.

Regarding Technology Profiling, the results are quite similar for both methods, with minor differences in the number of technologies identified. The execution times are almost identical, averaging around 0.71 seconds, which suggests that technology profiling is a relatively simple task and does not benefit significantly from automation.

Port Scanning revealed some interesting differences. The manual process discovered more open ports across all domains, with a notable difference for the first domain, where 306 ports were found manually, compared to only 122 ports in the automated process. However, the automated process was much faster, averaging 123.4 seconds, compared to the manual method, which averaged 65.33 minutes. This suggests that while manual port scanning may reveal more results, automation significantly improves efficiency.

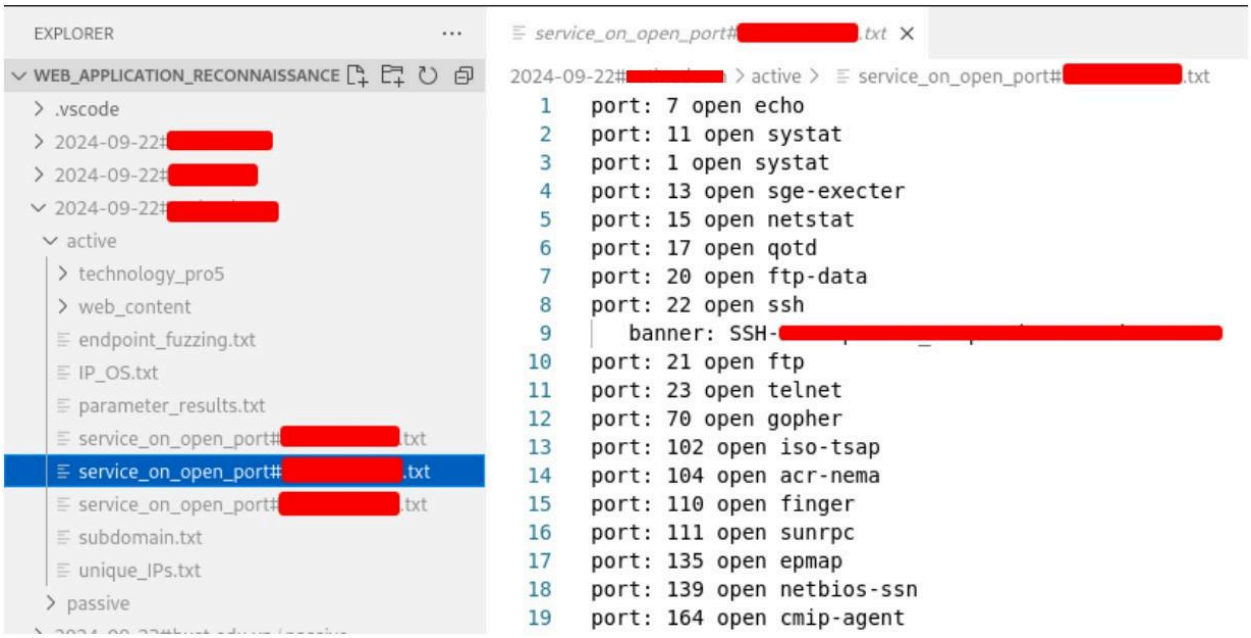

Figure 12: Example of Result Space of Automated Tool Active Reconnaissance.

Figure 12 is an example of the result space when performing active reconnaissance on a specific domain and according to Figure 8, the remaining part of the result space is the active part. As shown in Figure 12, the resulting space of the domain the first domain with the selected part is the result of the active reconnaissance step. In the active process, the techniques all achieve their respective results, which are saved in files in text format. In Figure 12, the right half is the result of the port scanning process.

VI. Conclusion.

This is an automated web application reconnaissance tool that is structured around well-researched reconnaissance modules. This testing is essential in simplifying complex reconnaissance processes, built on both passive and active reconnaissance. This deep dive

includes a study of the core protocols, data structures, and methodologies used in classical penetration testing; broken down into an automated framework. This is what makes this tool so powerful; it simply covers every stage of reconnaissance, right from OSINT-based passive footprinting to active probing and all the steps in between in a well-researched manner that allows for a wealth of information without any false positives on any poor or inefficient methodology.

The analysis is based on the research, definition, and design of the basic stages of the manual reconnaissance process. In the passive reconnaissance phase, the web crawler, WHOIS, DNS, and Google Analytics IDs are used to collect all available information. For 39 active reconnaissance, port scanning and fuzzing are well planned based on the analysis of vulnerabilities, protocols, and service behavior. Both processes are divided into specific modules that can be automated and at the same time provide a level of accuracy that is achievable during manual testing.

The analysis method is very important to create a specific tool for the target type which is a web application. Analyzing the protocols and output data at each stage, the tool has accurately highlighted important information such as hidden endpoints, open ports, and service versions. Furthermore, the fuzzing method for vulnerability detection has been improved by observing common web application behaviors to make the tool more flexible in various scenarios.

The analysis process is not just a task in the tool development process. This makes the tool effective in automating reconnaissance workflows because it accurately represents each technique with its equivalent automation process. Therefore, the tool not only performs reconnaissance automation but also ensures the reliability and comprehensiveness of the collected data. This makes the tool useful for penetration testers and security analysts.

VII. References.

[1] Esra Abdullatif Altulaihan, (March 4, 2023), A Survey on Web Application Penetration Testing. Retrieved from https://www.mdpi.com/2079-9292/12/5/1229

[2] yogeshojha, (Sep 11, 2024), rengine. Retrieved from https://github.com/yogeshojha/rengine?tab=readme-ov-file#workflow

[3] ym500, (July 29, 2022), Feature - Add Support For ARM arch. Retrieved from https://github.com/yogeshojha/rengine/issues/675

[4] Ramotion, (Sep 4, 2024), Web Application vs. Desktop Application: A Comparative Guide. Retrieved from https://www.ramotion.com/blog/web-application-vs-desktop-application

[5] willc, (Jan 24, 2019), OSINT-flowcharts. Retrieved from https://github.com/willc/OSINT-flowcharts
[6] Jessica Scarpati, John Burke, What is a URL (Uniform Resource Locator). Retrieved from https://www.techtarget.com/searchnetworking/definition/URL
[7] Evan Anderson, (July 19, 2023), Red teaming 101: What is red teaming. Retrieved from https://www.ibm.com/think/topics/red-teaming
[8] Imperva, Time To Live (TTL). Retrieved from https://www.imperva.com/learn/performance/time-to-live-ttl/
[9] NMAP.ORG, Nmap Network Scanning, Chapter 15. Nmap Reference Guide, Port Scanning Techniques. Retrieved from https://nmap.org/book/man-port-scanning-techniques.html

ANNEX

1. Terminal-Based Operation.

I used Python3 as the main programming language to develop the automated reconnaissance tool and it is designed to operate exclusively via the terminal, providing a streamlined and efficient interface for users familiar with command-line operations. Indeed, this tool is default developed in Linux Distro then I also use some functions and features that are native to Linux. The tool is invoked using a primary command followed by various options dictating specific reconnaissance tasks. The general syntax is as follows:

```
$ python3 popdom.py [options] [target]
```

Where:

- **python3** tells the system to use the Python 3 interpreter to execute the script.
- **popdom.py** is the name of the script's main function, which is responsible for calling all recon modules.
- **[options]** are optional command-line arguments that you can pass to the script.
- **[target]** is the parameter's value that a be transmitted in the options.

There are some options and their effect.

- **-h, --help:** Display the tool's usage and options' description.
- **-u**: Specify the target domain name, require a domain after as input.
- **-a**: Enable active reconnaissance module.
- **-p**: Enable active reconnaissance module.
- **-ipinfo**: To interact with the **IPinfo** API, mark that the API key stored in the environment file is a free key or premium key, and does not require input.
- **-cert**: Enable function domain to discover with certificates, and does not require input.
- **-cache**: Extension of **-cert** option, mark that the reconnaissance script uses downloaded certificates in storage without downloading new certificates, and does not require input.
- **-threads**: Provide the number of units in fuzzing tactic (default: 100).
- **-port_start**: Define the start port for port scanning (default: 1).
- **-port_end**: Define the end port for port scanning (default: 10000).