# PROJECT REVIEW I

**SYSTEM STUDY AND PROPOSED WORK**

# for

**Automated Web Reconnaissance Tool (AWRT)**

# Approved Prepared By

Beulah Christalin Latha Sidharth - URK20IS1007

Jijo Ashik - URK20IS1012

Logesh - URK20IS1066

Akshay - URK20IS1073

**

<<Date>>

# TABLE OF CONTENTS

#### Contents Page No

1. **INTRODUCTION**
   1. Overview of the Project 3
   2. Document Conventions 3
   3. Motivation and Background 3
   4. Problem Statement 4
   5. Objective of the Project 5
   6. Relevance and Need of the Project in the Present Context 6
   7. Project Scope 7
   8. References 8

#### LITERATURE REVIEW

* 1. Introduction 8
  2. Review of the Existing Systems 8
     1. Existing System 1 [Technologies used, Features, Drawbacks] 9
     2. Existing System 2 [Technologies used, Features, Drawbacks] 10
     3. Existing System 3 [Technologies used, Features, Drawbacks] 12
     4. Existing System 4 [Technologies used, Features, Drawbacks] 14
     5. Existing System 5 [Technologies used, Features, Drawbacks] 16
  3. Summary of Drawbacks of the Existing Systems/Research Gaps 20
  4. Proposed Approach 20
  5. Unique Features of the Proposed System 21
  6. Utility Value of the Proposed System 22
  7. Scalability and Environmental Sustainability 22

#### SYSTEM REQUIREMENTS

* 1. Introduction 23
  2. Users of the System 23
  3. Functional Requirements 24
     1. Functional Requirement 1 etc. 24
  4. Non-Functional Requirements 33
     1. Design and Implementation Constraints 35
  5. Hardware and Software Requirements 36
  6. External Interface Requirements 37
     1. User Interfaces 38
     2. Hardware Interfaces 38
     3. Software Interfaces 39
     4. Communication Interfaces 40
  7. Feasibility Analysis of the Requirements 40

1. **USER INTERFACE DESIGN**
   1. Screen Element Requirement Analysis
   2. Screen Interfaces

1. ****Appendices** 
   1. Context Diagram 42
   2. Data Flow Diagram 43
   3. Use Case Diagram 44

**

1. **INTRODUCTION**

### Overview of the Project

An Automated Web Reconnaissance Tool is a software application that automates the process of web reconnaissance, which is the act of gathering information about a target website. The goal of this tool is to streamline the process of identifying potential vulnerabilities and misconfigurations in a website's infrastructure and applications.

The tool typically consists of several modules or components that work together to perform various tasks, such as network scanning, web server scanning, application scanning, and vulnerability identification. The tool may also include features such as automation of repetitive tasks, reporting and alerting, and integration with other security tools.

The benefits of an Automated Web Reconnaissance Tool are that it can save time and effort in identifying vulnerabilities and misconfigurations, as well as increase the accuracy and thoroughness of the reconnaissance process. It can also be used to proactively identify issues before they can be exploited by attackers.

However, it is important to note that the tool should be used ethically and with the consent of the website owner. Using the tool to scan websites without permission is illegal and can result in legal consequences.

### Document Conventions

● The headings are used in Times New Roman

● Size of each word: 12

● Heading size: 14

● Word fonts are used in Times New Roman

● Highlighted word is used in Times New Roman

● Size of a highlighted word is: 14

### Motivation & Background

**Motivation:**

The motivation for developing an Automated Web Reconnaissance Tool is to streamline the process of identifying potential vulnerabilities and misconfigurations in a website's infrastructure and applications. In today's digital age, the number of websites and web applications has increased dramatically, making it challenging for security professionals to identify and address potential security issues. An Automated Web Reconnaissance Tool can save time and effort in identifying these issues, as well as increase the accuracy and thoroughness of the reconnaissance process.

**

**Background**:

Web reconnaissance is the process of gathering information about a target website in order to identify potential vulnerabilities and misconfigurations. Traditionally, this process has been performed manually by security professionals, which can be time-consuming and prone to human error. In recent years, there has been a growing interest in developing automated tools that can assist in the reconnaissance process.

The development of an Automated Web Reconnaissance Tool requires a strong understanding of web technologies and security vulnerabilities. The tool typically consists of several modules or components that work together to perform various tasks, such as network scanning, web server scanning, application scanning, and vulnerability identification. The tool may also include features such as automation of repetitive tasks, reporting and alerting, and integration with other security tools.

### Problem Statement

The problem statement in an Automated Web Reconnaissance Tool is to automate the process of web reconnaissance, which is the act of gathering information about a target website, and identify potential vulnerabilities and misconfigurations in a website's infrastructure and applications. The traditional manual process of web reconnaissance is time-consuming, prone to human error, and can be ineffective in identifying potential security issues.

Furthermore, as the number of websites and web applications continues to grow, the need for automated tools that can assist in the reconnaissance process becomes increasingly important. The development of an Automated Web Reconnaissance Tool addresses this need and can save time and effort in identifying potential security issues, increase the accuracy and thoroughness of the reconnaissance process, and proactively identify issues before they can be exploited by attackers.

However, it is important to note that the use of such a tool must be ethical and within the bounds of the law. Using the tool to scan websites without permission is illegal and can result in legal consequences. Therefore, the tool must be designed to ensure that it is used ethically and with the consent of the website owner.

### Objective of the Project

The objective of the Automated Web Reconnaissance Tool project is to automate the process of web reconnaissance and increase the accuracy and thoroughness of identifying potential vulnerabilities and misconfigurations in a website's infrastructure and applications.

This is achieved through the development of a tool that can perform the following objectives:

1)Automate the process of gathering information about a target website, including network and web server scanning, application scanning, and vulnerability identification.

2)Save time and effort in identifying potential security issues, increase the accuracy of the reconnaissance process, and proactively identify issues before they can be exploited by attackers.

3)Provide a comprehensive report of the identified vulnerabilities and misconfigurations in a website's infrastructure and applications.

4)Allow for customization and configuration of the tool to meet the specific needs of the user.

5)Ensure the tool is used ethically and with the consent of the website owner.

### Project Scope

Web Server Scanning: The tool should be able to scan a website's infrastructure to identify open ports, running services, and operating systems. Web Application Scanning: The tool should be able to scan the website's applications to identify vulnerabilities in the code and configurations, such as SQL injection, cross-site scripting (XSS), and insecure authentication mechanisms.

* Vulnerability Identification: The tool should be able to identify potential vulnerabilities and misconfigurations in a website's infrastructure and applications based on industry best practices and known vulnerabilities.
* Reporting: The tool should provide a comprehensive report of the identified vulnerabilities and misconfigurations, including severity levels and recommendations for remediation.
* Customization: The tool should allow for customization and configuration to meet the specific needs of the user, such as choosing the types of scans to perform and defining the scope of the scan.
* Ethics and Legal Compliance: The tool should be designed to ensure ethical and legal compliance, including obtaining consent from website owners before performing scans and ensuring compliance with data protection laws.

### References

* "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws," by Dafydd Stuttard and Marcus Pinto, is a comprehensive guide to web application security that covers reconnaissance, vulnerability identification, and exploitation.
* "Web Application Security: A Beginner's Guide," by Bryan Sullivan and Vincent Liu, provides an introduction to web application security, including reconnaissance and vulnerability scanning.
* "Black Hat Python: Python Programming for Hackers and Pentesters," by Justin Seitz, includes a chapter on web reconnaissance that covers techniques for discovering web applications and identifying vulnerabilities.
* "Mastering Modern Web Penetration Testing," by Prakhar Prasad, provides a practical guide to modern web penetration testing, including web reconnaissance, vulnerability identification, and exploitation.
* "Web Application Vulnerabilities: Detect, Exploit, Prevent," by Steven Palmer, provides an overview of web application vulnerabilities and includes a section on reconnaissance.
* "The Art of Reconnaissance: Information Gathering Techniques," by Carlos A. Perez, provides an in-depth look at information gathering techniques, including web reconnaissance.
* "Automated Web Application Security Testing: Simplified and Effective Approach," by Prasad J Pandit, provides an overview of automated web application security testing, including reconnaissance and vulnerability scanning.

## LITERATURE REVIEW

### Introduction

The Automated Web Reconnaissance Tool is a project aimed at developing an automated tool for web reconnaissance, with the goal of improving the efficiency and effectiveness of web application security testing. In order to develop such a tool, it is essential to have a comprehensive understanding of the current state of web reconnaissance, including existing techniques, tools, and best practices.

A literature review is an important component of the research process for the Automated Web Reconnaissance Tool project, as it can provide a detailed overview of the existing research related to web reconnaissance. By reviewing relevant literature, it is possible to identify the most effective approaches for automated web reconnaissance and understand the challenges associated with detecting vulnerabilities through reconnaissance.

Additionally, a literature review can provide insights into the current state of web application security, including common vulnerabilities and attack vectors. This knowledge can inform the development of the Automated Web Reconnaissance Tool, as it can help in designing the tool to address common vulnerabilities and attack vectors.

### Review of the Existing Systems

* 1. **Existing System 1 Nmap:**

Nmap is a powerful network exploration tool that can be used for a variety of purposes, including information gathering and reconnaissance. Some of the key features of Nmap that make it useful for information gathering include:

### Features of Nmap

Nmap is a powerful and widely used tool for network exploration, management, and security auditing. It offers a wide range of features that are useful for automated web reconnaissance, including:

* Port scanning: Nmap can scan a range of ports on a target system, identifying which ports are open, closed, or filtered.
* OS detection: Nmap can identify the operating system running on a target system by analyzing network packets and comparing them to a database of known OS fingerprints.
* Service detection: Nmap can identify the services running on a target system, such as web servers, FTP servers, and email servers.
* Scripting engine: Nmap has a powerful scripting engine that can be used to automate tasks, such as vulnerability scanning and network mapping.

Flexible output: Nmap can output results in a variety of formats, including XML, grepable, and human-readable formats. Stealth scanning: Nmap offers a range of scanning options that can be used to avoid detection by intrusion detection systems and firewalls.

* + 1. **Technology and Algorithms used in Nmap:**

Nmap is a popular and powerful open-source tool used for network exploration, management, and security auditing. It uses a combination of different technologies to perform its tasks, including:

* Port Scanning: Nmap uses various scanning techniques to determine which ports are open and what services are running on them. It can use different scanning methods, such as TCP, UDP, and ICMP, to determine the open ports.
* Operating System Detection: Nmap uses various methods to detect the operating system running on the target system. It can do this by sending packets with specific flags set and analyzing the responses.
* Service and Version Detection: Nmap can identify the services and software versions running on the target system by analyzing the responses received from the target system.
* Scripting Engine: Nmap has a scripting engine that allows users to write and execute scripts that can perform complex tasks, such as vulnerability detection, network mapping, and enumeration.
* Zenmap GUI: Zenmap is a graphical user interface for Nmap that provides a user-friendly interface for scanning and analyzing networks. It allows users to easily configure and run Nmap scans, view the results, and generate reports.

### Drawbacks of Nmap

While Nmap is a powerful tool for information gathering and reconnaissance, it does have some drawbacks that should be considered. Some of these drawbacks include:

* Network congestion: Nmap can generate a lot of network traffic, which can lead to network congestion and slowdowns. This can be especially problematic on large networks or networks with limited bandwidth.
* False positives/negatives: Nmap may produce false positives or false negatives in its scan results, which can lead to inaccurate information or missed vulnerabilities.
* Detection by intrusion detection/prevention systems: Nmap scans can be detected by intrusion detection and prevention systems, which can alert defenders to the presence of an attacker on the network.
* Time-consuming: Nmap scans can be time-consuming, especially on large networks or when using advanced scanning options. This can be a limitation in time-sensitive situations.

Legal considerations: The use of Nmap may be subject to legal restrictions or requirements, depending on the jurisdiction and the intended use of the tool. It is important to be aware of any legal considerations before using Nmap or any other information-gathering tool.

* 1. **Existing System 2 gobuster**

Gobuster is a tool used in automated web reconnaissance to perform directory/file and DNS subdomain brute-forcing. It is designed to be simple and efficient, making use of multi-threading to increase the speed of the scans. Gobuster works by sending HTTP requests to the target website and analyzing the response codes to determine whether a particular directory or file exists. It also checks for the existence of DNS subdomains by brute-forcing the subdomains against a wordlist. Gobuster is written in Go, a programming language known for its speed and efficiency, and is open source software released under the Apache License 2.0.

**2.2.1 Features of Gobuster:**

Fast and efficient: Gobuster is designed to be fast and efficient, allowing users to quickly scan web servers and identify hidden files and directories.

* Multiple modes: Gobuster offers different modes, including directory mode, DNS mode, and VHost mode, allowing users to customize their scans and target specific areas of a web server.
* Customizable wordlists: Gobuster allows users to create and customize their own wordlists, which can be used for directory and file brute-forcing.
* HTTP/S support: Gobuster supports both HTTP and HTTPS protocols, allowing users to scan web servers securely.
* Output formats: Gobuster offers different output formats, including JSON, XML, and plain text, making it easy to integrate with other tools or scripts.
* Multithreaded: Gobuster is multithreaded, allowing users to perform multiple scans simultaneously, which can save time and increase efficiency.
* Proxy support: Gobuster can be configured to use a proxy, allowing users to scan web servers through a proxy server for increased anonymity.
  + 1. **Technology and Algorithms used in Gobuster:**

Gobuster is a tool used for web content discovery and enumeration. It is written in Go programming language and uses several technologies to perform its operations, including:

* Goroutines: Goroutines are lightweight threads used by Gobuster to perform its operations concurrently, making it faster and more efficient.
* Regular expressions: Gobuster uses regular expressions to match patterns in the response body of web pages, allowing it to identify specific types of content, such as directories, files, or subdomains.
* HTTP requests: Gobuster sends HTTP requests to the target website to enumerate its content. It supports several HTTP methods, including GET, HEAD, and POST.
* DNS resolution: Gobuster can perform DNS resolution to identify subdomains of the target website.
* Wordlists: Gobuster uses wordlists to generate a list of possible content paths or subdomains that it can try to discover.

### Drawbacks of Gobuster

While Gobuster is a powerful and useful tool for information gathering, there are some potential drawbacks to using it:

* Limited functionality: While Gobuster is great for directory and file brute-forcing, it does not have the same range of functionality as more comprehensive tools like Nmap. For example, it cannot perform port scanning or detect vulnerabilities.
* False positives: Gobuster relies on brute-forcing to identify directories and files, which can result in false positives. This can be a time-consuming issue, as false positives need to be manually filtered out.
* Resource-intensive: Gobuster can be resource-intensive, especially if multiple threads are used. This can lead to performance issues on slower or less powerful systems.
* Lack of customization: While Gobuster allows users to create and customize their own wordlists, it does not offer the same level of customization as some other tools. For example, it cannot be configured to use specific HTTP methods or headers.

**2.3.1 Existing System 3 amass:**

Amass is an open-source tool used for conducting a reconnaissance process to gather information about a target. The tool has been developed in Go language and is designed to perform various tasks such as network mapping, subdomain enumeration, reverse IP resolution, and many more. The tool is highly efficient and can be used to perform various tasks related to reconnaissance and intelligence gathering.

Amass is a modular tool and supports various data sources like Brute-Forcing, API, and Scraping for finding the target's domain and subdomain names. The tool is highly extensible and can be used with other tools to gather information about the target network. Amass also supports a range of output formats and can generate reports in JSON, CSV, and other formats. Overall, Amass is a highly efficient tool for information gathering and reconnaissance.

**2.3.2 Features of Amass:**

Amass is an open-source tool designed for network mapping and reconnaissance that provides several features for information gathering, such as:

* Subdomain Enumeration: Amass can quickly enumerate subdomains of a given domain by querying various data sources such as DNS servers, certificate transparency logs, and web archives.
* Active Scan: It can perform active reconnaissance by probing the discovered subdomains using various techniques such as DNS resolution, HTTP probing, and service scanning.
* Port Scanning: Amass can conduct port scanning to identify open ports and services running on them.
* Metadata Extraction: It can extract metadata such as email addresses, IP addresses, and other network-related information from the identified subdomains.
* Integration: Amass can integrate with other tools such as Nmap and Masscan for a more comprehensive reconnaissance.
* Flexible Configuration: Amass provides a wide range of configuration options to customize the reconnaissance process according to the user's requirements.

**2.3.3 Technology and Algorithms used in amass:**

The Amass tool uses several technologies to perform its information gathering tasks. These technologies include:

* DNS: Amass uses DNS queries to discover subdomains and related domain information.
* Scraping: Amass scrapes web pages to find email addresses, IP addresses, and other information related to domains and subdomains.
* Certificates: Amass uses digital certificates to discover additional subdomains and domain information.
* API: Amass can leverage various APIs to gather additional domain and subdomain information.

Brute Force: Amass can perform brute force attacks to discover subdomains and domain information by guessing domain names.

**2.3.4 Drawbacks of amass:**

Some of the drawbacks of the Amass tool in gathering information are:

* False positives: Amass tool can sometimes generate a lot of false positives, which can be time-consuming to sift through.
* Slow speed: Amass tool can be relatively slow when scanning large domains, and this can slow down the entire reconnaissance process.
* Resource-intensive: Amass tool can be resource-intensive and can require significant computing power and memory, especially when dealing with large domains.
* Limited support: Amass tool has limited support for some operating systems and platforms, which can make it difficult to use in some environments.
* Lack of integration: Amass tool may not be fully integrated with other tools in the reconnaissance toolchain, which can create gaps in the information-gathering process.

**2.4.1Existing System 4 Nikto**

Nikto is an open-source web server scanner that performs comprehensive tests against web servers for various items, including over 6700 potentially dangerous files or CGIs, versions on over 1250 servers, and version-specific issues on over 270 servers. It is designed to scan webservers and provide a report of potential vulnerabilities that may exist on the server. Nikto is a command-line tool that can perform a wide range of checks on web servers, including checks for outdated software, default configuration issues, and potential vulnerabilities in third-party software. It is a popular tool for penetration testing and security auditing.

**2.4.2 Features of Nikto**

Nikto is a popular web server scanner used to find vulnerabilities and potential security holes in web servers. Some of the key features of the Nikto tool in information gathering are:

* Comprehensive scanning: Nikto scans web servers comprehensively, including the web server, software running on it, outdated software, and other known vulnerabilities.
* Cross-platform: The tool is written in Perl, making it compatible with multiple operating systems.
* Multiple scans: The tool provides options for multiple scans, including full scans and specific scans for plugins, CGI scripts, and other files.
* Updated database: Nikto has an updated database of more than 6700+ known vulnerabilities and weaknesses, which allows the tool to identify potential security holes.
* SSL support: Nikto provides SSL support, which allows it to scan HTTPS websites.
* False-positive reduction: The tool provides options to reduce false positives, which can help save time and resources.
* Report generation: Nikto provides detailed reports in HTML, CSV, and XML formats, which can be used for further analysis and remediation.
* Open-source: Nikto is an open-source tool, which means it can be easily customized to suit specific requirements.

**2.4.3 Technology and Algorithms used**

Nikto is written in Perl and uses various open-source Perl modules to perform its scans. It uses a combination of manual tests and heuristics to identify potential vulnerabilities in web servers and applications. Nikto can also make use of external plugins and scripts for additional functionality. In terms of technology, Nikto relies on HTTP and HTTPS protocols to communicate with web servers and exchange data during its scans.

**2.4.4 Drawbacks of Nikto**

Here are some potential drawbacks of using Nikto tool for information gathering:

* False positives: Nikto may sometimes generate false positives, which can lead to unnecessary investigation and wasted time.
* Limited scope: Nikto is mainly designed to scan web servers and web applications, so it may not be suitable for detecting vulnerabilities in other types of systems or protocols.
* Limited configuration options: Nikto provides some basic configuration options, but it may not be as customizable as some other tools.
* No exploitation capabilities: Nikto is primarily a vulnerability scanner, and it does not have built-in capabilities for exploiting or attacking vulnerabilities that it finds.
* No GUI: Nikto is a command-line tool, which may be less user-friendly for some users who prefer graphical user interfaces.

**2.5.1 Existing System 5 Wapiti:**

Wapiti is an open-source web application vulnerability scanner that is designed to detect various security vulnerabilities in web applications. It performs black-box testing, meaning it does not have access to the source code or architecture of the application being tested. Instead, it sends requests to the web application and analyzes the responses to detect vulnerabilities.

Wapiti can be used to scan for a range of vulnerabilities, including SQL injection, cross-site scripting (XSS), file inclusion, command injection, and many others. It also supports various authentication mechanisms and can be used to scan both authenticated and unauthenticated web applications.

In addition to vulnerability scanning, Wapiti can also be used to perform various other tests such as crawling the application, detecting backup files, and detecting web application firewalls (WAFs). Wapiti is written in Python and uses various libraries and tools such as Requests, YAML, It is a command-line tool and can be run on various operating systems including Windows, Linux, and macOS.

**2.5.2 Features of Wapiti:**

Wapiti is a command-line tool used to audit the security of web applications. Some of its key features include:

* Cross-platform support: Wapiti can run on various operating systems, including Linux, macOS, and Windows.
* Customizable scanning: Users can customize their scans by specifying different HTTP methods, payloads, and injection points.
* File upload testing: Wapiti can check for vulnerabilities in file upload functionality, including file type and size validation.

Authentication support: Wapiti can handle different types of authentications, including basic authentication, forms-based authentication, and cookies-based authentication.

* Reports: The tool generates HTML reports that detail the vulnerabilities found during the scan, including the severity of each vulnerability and recommendations for remediation.
* Integration with other tools: Wapiti can be integrated with other tools, such as Burp Suite and OWASP ZAP, to enhance the overall security testing process.
* Open-source: Wapiti is an open-source tool, which means that its source code is publicly available and can be modified by users to suit their specific needs.

**2.5.3 Technology and Algorithms used in Wapiti:**

Wapiti is a command-line tool written in Python and uses various technologies for its functionality, including:

* Regular expressions: Wapiti uses regular expressions to search for patterns in the HTML code of the web application.
* HTTP requests: The tool sends HTTP requests to the web application to test for vulnerabilities.
* Python libraries: Wapiti uses several Python libraries such as BeautifulSoup and Requests to parse and analyze the HTML code.
* Multi-threading: Wapiti can use multiple threads to scan web applications faster.
* Database support: Wapiti supports SQLite, MySQL, and PostgreSQL databases to store scan results.



**2.5.4 Drawbacks of Wapiti:**

* Limited support for JavaScript-based applications: Wapiti has limited support for scanning JavaScript-based applications, which can be a disadvantage when testing modern web applications.
* False positives: Like other vulnerability scanners, Wapiti can produce false positives, which can lead to wasted time and effort in investigating non-existent vulnerabilities.
* Limited GUI interface: Wapiti has a limited GUI interface, which can be challenging for beginners who are not familiar with command-line interfaces.

### Overview of the Technologies used

Web Crawling/Scraping Technologies such as Python's Beautiful Soup and Scrapy, and Ruby's Nokogiri are commonly used for web crawling and scraping. These technologies allow the extraction of specific information from web pages, which can be useful for information gathering.

* Port Scanning: Nmap is a popular tool for port scanning, which involves scanning a network for open ports on different hosts. Nmap uses various techniques such as SYN scan and UDP scan to determine which ports are open and which services are running on those ports.
* Subdomain Enumeration: Tools like Amass, Sublist3r, and Recon-ng use various techniques such as brute-forcing, search engines, and certificate transparency logs to discover subdomains associated with a given domain.
* Vulnerability Scanning: Tools like Nikto and Wapiti can scan web applications for vulnerabilities such as SQL injection, cross-site scripting (XSS), and file inclusion vulnerabilities.

OSINT: Open-source intelligence (OSINT) tools like Maltego, theHarvester, and Shodan can be used to gather publicly available information about an organization or target.



Social Engineering: Tools like SET (Social-Engineer Toolkit) and BeEF (Browser Exploitation Framework) can be used to carry out social engineering attacks and gather information from users through phishing and other techniques.

### Analysis of the Algorithms

Algorithms used in tools for information gathering can vary depending on the specific tool and its purpose. However, some common algorithms that may be used include:

* Brute force: this algorithm involves trying every possible combination of characters until a match is found. It is often used for password cracking.
* Dictionary attack: this algorithm involves trying a list of common words and phrases as possible passwords. It is also used for password cracking.
* Fuzzing: this algorithm involves sending many random inputs to a system to try to identify vulnerabilities or unexpected behavior.
* Pattern matching: this algorithm involves searching for specific patterns or regular expressions in data to identify relevant information.
* Heuristics: this algorithm involves using a set of rules or guidelines to identify potentially relevant information or patterns.
* Machine learning: this algorithm involves training a computer to recognize patterns in data and make predictions or classifications based on that information. It can be used for tasks such as identifying anomalies or predicting future behavior.

### Drawbacks of the Existing Systems/Research Gaps

Based on the review of the existing systems in information gathering, the following drawbacks and research gaps can be identified:

* Limited functionality: Some of the existing tools such as Nmap and Dirb have limited functionality in terms of their ability to identify vulnerabilities and potential threats in web applications.
* False positives: Many tools such as Nikto and Wapiti may generate false positives, which could lead to wasting time and resources on investigating non-existent vulnerabilities.
* Lack of automation: Some tools such as TheHarvester require significant manual intervention to be effective, which can be time-consuming and inefficient.
* Limited scope: Existing tools may not be effective in identifying vulnerabilities in all types of web applications or may have limitations in terms of the types of vulnerabilities they can identify.
* Research gaps: There is a need for more research on developing tools that can identify vulnerabilities in modern web applications, such as those using JavaScript frameworks, APIs, and microservices.

### Proposed Approach

The proposed approach for information gathering using the Automated Web Reconnaissance Tool will involve several steps:

* Subdomain enumeration: The first step will involve using the Amass tool to find subdomains of the target website. This will help identify any hidden or forgotten subdomains that may contain sensitive information.
* Port scanning: Once the subdomains are identified, the next step will involve using the Nmap tool to scan for open ports. This will help identify any potential vulnerabilities that can be exploited.
* Web application scanning: The next step will involve using the Nikto and Wapiti tools to scan the web application for vulnerabilities. This will include identifying outdated software, configuration issues, and other security flaws that can be exploited by attackers.
* Directory and file enumeration: Finally, the Gobuster and Dirb tools will be used to enumerate directories and files on the web application. This will help identify any hidden or sensitive files that may have been overlooked in the initial reconnaissance*.*
  1. **Unique Features of the Proposed System:**

The proposed system using automated web reconnaissance tool has several unique features, including:

* Integration of multiple information gathering tools: The system integrates various information gathering tools such as Nmap, the Harvester, Gobuster, Amass, and Nikto, which enables it to gather information from multiple sources.
* Automation: The system automates the information gathering process, eliminating the need for manual intervention. This saves time and increases efficiency.
* Customizable: The system is highly customizable, allowing users to specify the target website, the type of information to be gathered, and the tools to be used.
* User-friendly: The system has a simple and intuitive interface, which makes it easy to use, even for users with limited technical skills.
* Comprehensive reporting: The system generates a comprehensive report of the information gathered, including vulnerabilities, subdomains, open ports, and other relevant information.
* Continuous scanning: The system supports continuous scanning, which allows users to keep track of changes to the target website and to identify new vulnerabilities as they arise.



* 1. **Utility Value of the Proposed System**

The utility value of the proposed system in automated information gathering tool is high as it provides a comprehensive and efficient approach to gather information from websites. The automated web reconnaissance tool combines the features of different existing tools and technologies to provide a more robust and accurate information gathering process. It can save a considerable amount of time and effort for security professionals, researchers, and ethical hackers who rely on information gathering to identify potential vulnerabilities and threats in web applications. The automated system can also help to reduce the risk of manual errors and ensure the consistency and reliability of the results. Overall, the proposed system can enhance the effectiveness and efficiency of the information gathering process for web applications.

* 1. **Scalability and Environmental Sustainability**

The scalability of the automated web information gathering tool will depend on various factors such as the size of the target website, the number of users accessing the tool, and the computing resources available. It is essential to design the tool in a scalable manner so that it can handle large websites and a high number of users. In terms of environmental sustainability, the tool should be designed to use computing resources efficiently to minimize energy consumption and reduce the carbon footprint. It is also important to use eco-friendly computing resources and adopt sustainable practices in the development and deployment of the tool.



1. **SYSTEM FEATURES**

**3.1 Introduction**

The system requirements for any software tool are essential to ensure its proper functioning and smooth execution. The same is true for the Automated Web Reconnaissance Tool, which is designed to automate the information gathering process for web applications.

The system requirements include hardware and software specifications that are necessary to run the tool on a computer or server. These requirements determine the compatibility of the tool with the user's system and ensure that the tool can perform optimally.

The system requirements for the Automated Web Reconnaissance Tool are determined based on factors such as the size and complexity of the target website, the number of concurrent scans, and the type of scanning techniques used. The following section outlines the minimum and recommended system requirements for the tool.

* 1. **Users of the System: -**

The main use of automated web reconnaissance tools is to gather information about a target web application or website in an automated and efficient manner. This information can then be used to identify vulnerabilities and potential security risks, and ultimately improve the security posture of the target. Some specific use cases of automated web reconnaissance tools include:

* Penetration testing: Automated web reconnaissance tools are commonly used by security professionals and penetration testers to identify potential vulnerabilities and attack vectors in web applications.
* Vulnerability assessment: These tools can be used to conduct automated scans and tests of web applications to identify known vulnerabilities and weaknesses.
* Security research: Researchers and academics may use automated web reconnaissance tools to collect data and analyze web application behavior for research purposes.
* Website management: Website administrators can use these tools to perform scans and tests of their own websites to identify potential issues and improve overall website security.
* Competitive intelligence: Automated web reconnaissance tools can also be used by businesses and organizations to gather information on competitors' websites, marketing strategies, and overall online presence.



* 1. **Functional Requirements:**

The functional requirements of an automated web reconnaissance tool can vary depending on the specific goals and features of the tool. However, some common functional requirements that can be included in the tool are:

* Information gathering: The tool should be able to perform information gathering on the target website(s) such as identifying the IP address, finding subdomains, scanning for open ports, and identifying web technologies and server software.
* Vulnerability scanning: The tool should be able to scan the target website(s) for vulnerabilities in web applications and web servers, including known security issues, misconfigurations, and weak authentication mechanisms.
* Content discovery: The tool should be able to discover the content of the website, including hidden and restricted pages, directories, files, and APIs.
* Reporting and output: The tool should be able to provide detailed and actionable reports on the findings of the information gathering, vulnerability scanning, and content discovery. The output should be presented in a clear and concise format, including recommendations for remediation.
* Customization: The tool should be flexible enough to allow for customization and configuration based on the specific needs of the user or project.
* Automation: The tool should be able to automate the process of web reconnaissance to save time and effort.
* Integration: The tool should be able to integrate with other security tools and systems, including vulnerability management and patching tools.
* User management: The tool should provide user management functionality to ensure that only authorized users can access and use the tool.
* Security and privacy: The tool should follow best security and privacy practices, including protecting sensitive data and not compromising the security of the target website(s) during reconnaissance.



* + 1. **System Feature 1 in Nmap:**

Some of the key features of Nmap include:

* Host discovery: Nmap can identify live hosts on a network and the IP addresses they use.
* Port scanning: Nmap can scan the target network to identify the open ports and services running on each port.
* Service detection: Nmap can detect the service running on each open port and gather information about it.
* Operating system detection: Nmap can determine the operating system of the target system based on various characteristics of the network responses.
* Scriptable interaction: Nmap supports scripting using Lua programming language, allowing users to write their own scripts for various network scanning tasks.
* Flexible output: Nmap can generate output in various formats, including XML, HTML, grepable, and others, making it easy to parse and analyze the results.
* Extensibility: Nmap can be extended using various plugins and scripts, allowing users to add custom functionality to the tool.
* Stealth scanning: Nmap supports various stealth scanning techniques to avoid detection by firewalls and intrusion detection systems.
  + 1. **Description and Priority**

In nmap, the description is a brief explanation of the purpose or functionality of a particular scan or command. It provides the user with a clear idea of what the scan or command is intended to accomplish.

Priority in nmap refers to the order in which the various scans or commands are executed. Some scans may take longer to complete, while others may be more critical to the overall objective of the information gathering process. Prioritizing scans based on their importance and potential impact can help optimize the scanning process and reduce the time required to gather the desired information.



* + 1. **Stimulus/Response Sequences**

Stimulus/Response Sequences in Nmap can be described as follows:

Stimulus: The user enters a command to scan a target network or host using Nmap.

Response: Nmap executes the scan and provides a report of the results, which may include information such as open ports, running services, operating system details, and more.

Stimulus: The user specifies different scanning options or parameters, such as timing options or scan types.

Response: Nmap adjusts the scan based on the specified options and provides a report with the relevant information.

Stimulus: Nmap encounters a firewall or other network security measure that is blocking some or all of its scans.

Response: Nmap may attempt to bypass the security measure by using advanced scanning techniques, such as decoy scanning or fragment scanning, in order to obtain as much information as possible about the target network or host.

Stimulus: Nmap encounters a target that is not responsive or is experiencing network issues.

Response: Nmap may retry the scan several times, adjust the timing and packet rate, or use other methods to try and establish a connection with the target in order to obtain information about it. If the target is still not responsive, Nmap may mark it as "down" and move on to the next target.



* + 1. **System Feature 2 Gobuster:**

Some of the system features of Gobuster tool in information gathering are:

* Fast and multi-threaded: Gobuster can perform its task quickly and efficiently by utilizing multiple threads to send HTTP requests.
* Modular design: Gobuster has a modular design that allows for the use of custom modules to extend its functionality.
* Directory/file brute forcing: Gobuster can be used for both directory and file brute forcing by providing a wordlist of possible directory and file names.
* Custom headers and cookies: Gobuster allow for the use of custom headers and cookies to simulate different user agents and sessions.

Recursive brute forcing: Gobuster can perform recursive brute forcing by following links found in HTML pages and continuing to scan the website for additional directories and files.

* + 1. **Description and Priority in Gobuster:**

As an open-source tool, Gobuster is not typically accompanied by a detailed requirements specification document that includes a description and priority of features. However, some of the main features and priorities for Gobuster may include:

* Recursive directory and file enumeration: This feature allows Gobuster to recursively search through a web application's directory structure to identify hidden directories or files that are not easily discoverable. This can be a critical component of a comprehensive vulnerability assessment and penetration testing strategy, so this feature likely has a high priority.
* Support for different wordlists: Gobuster allows users to specify different wordlists to use for brute-force directory/file discovery. This feature allows users to customize their testing approach and adjust their wordlist based on the specific application being tested. This feature may be considered moderately high priority.
* Multi-threading support: Gobuster's support for multi-threading allows it to quickly and efficiently scan large applications with many directories and files. This feature likely has a high priority, as speed and efficiency are critical in the context of vulnerability assessments and penetration testing.
* Authentication support: Gobuster can be configured to use different authentication methods, including HTTP basic authentication and session cookies. This feature is important for testing web applications that require authentication, and may be considered a high priority.
* Integration with other tools: Gobuster can be integrated with other tools such as Burp Suite and OWASP ZAP to provide a more comprehensive testing approach. This feature may be considered moderately high priority, as many security professionals prefer to use multiple tools in their testing approach.
  + 1. **Stimulus/Response Sequences in Gobuster:**

Stimulus/Response sequences in Gobuster information gathering tool can be described as follows:

**Stimulus:**

User inputs the domain name or IP address of the target website along with the selected wordlist and options.

User initiates the scan by running the Gobuster tool.

**Response:**

Gobuster tool sends requests to the web server using various HTTP methods and checks the response code and body.

If the response code indicates that the file or directory exists, it is reported to the user.

If the response code indicates an error, the tool proceeds to the next request.

The results of the scan are displayed to the user in a structured format, indicating the status code and the corresponding URL.



* + 1. **System Feature 3 amass:**

Some of the system features of Amass tool in information gathering are:

Cross-platform compatibility: Amass tool can run on multiple operating systems including Windows, Linux, and Mac OS X.

* Integration with other tools: Amass can be integrated with other popular tools such as Nmap and Burp Suite to enhance the effectiveness of information gathering.
* Wide range of data sources: Amass has the ability to gather data from multiple sources including DNS, web archives, search engines, and more.
* Customizable configuration: Amass allows users to configure and customize the tool according to their specific needs, including options for setting timeouts, ports, and other parameters.
* Extensive documentation: Amass has extensive documentation and a user-friendly interface that makes it easy for users to understand and use the tool effectively.
  + 1. **Description and Priority in amass:**

Amass is a powerful information gathering tool that helps in enumerating subdomains, IP addresses, and ports of a target domain. It uses a number of techniques to gather this information including scraping websites, brute-forcing DNS nameservers, querying Certificate Transparency (CT) logs, and more.

The priority of Amass is to provide comprehensive information gathering results in a fast and efficient manner. It has the ability to gather large amounts of data on a target and can be customized to suit the needs of the user.

Overall, the main goal of Amass is to provide a detailed map of the target's online presence, including all associated domains, IP addresses, and other related information.



* + 1. **Stimulus/Response Sequences in amass:**

Stimulus: User inputs a target domain name or IP address.

Response: Amass starts enumerating subdomains, IP addresses, and other network entities associated with the target.

Stimulus: User provides a list of subdomains or IP addresses.

Response: Amass performs additional passive and active reconnaissance to find associated network entities.

Stimulus: User specifies a wordlist or dictionary to brute-force subdomains and directories.

Response: Amass performs brute-force enumeration to identify new subdomains, directories, and files.

Stimulus: User sets up Amass to integrate with other tools and services, such as DNS resolvers, web APIs, or databases.

Response: Amass leverages these resources to enhance its reconnaissance capabilities and produce more accurate results.

**3.4.1 System Feature 4 Nikto:**

Comprehensive scanning: Nikto tool can perform comprehensive scans on web servers, covering more than 6700 potentially dangerous files and CGIs.

* SSL support: The tool is capable of performing scans on SSL-enabled websites and can identify SSL misconfigurations.
* CGI vulnerability checks: Nikto can identify common CGI vulnerabilities such as arbitrary file retrieval, SQL injection, and cross-site scripting.
* Multiple output formats: The tool provides multiple output formats, including HTML, CSV, and NBE, which can be used for easy analysis and reporting.



* Plugin support: Nikto can be extended with plugins to add new features and capabilities to the tool.
* Proxy support: The tool can be configured to use a proxy for scanning, which can be useful in certain situations.
* OS detection: Nikto can detect the underlying operating system of the target server, which can be useful in further identifying potential vulnerabilities.
* Authentication support: The tool can support authentication to access password-protected pages or perform scans as an authenticated user.

**3.4.2 Description and Priority in Nikto:**

Nikto is an open-source web server scanner that can perform comprehensive tests on web servers for known vulnerabilities and misconfigurations.

**Description:**

Nikto aims to scan web servers for over 6700 potential vulnerabilities, including server and software misconfigurations, known vulnerabilities in web servers, and common configuration errors.

The tool is designed to be flexible, with support for multiple scanning modes and the ability to fine-tune the scans to specific needs.

**Priority:**

The priority of Nikto is to identify vulnerabilities and misconfigurations in web servers quickly and efficiently.

The tool is designed to be user-friendly and accessible to both novice and experienced users.

Nikto also prioritizes accuracy in its results, avoiding false positives while providing comprehensive reports on potential issues.

**3.4.3 Stimulus/Response Sequences in nikto:**

**Stimulus:**

User provides the target URL or IP address

User specifies options and preferences

User initiates scan

**Response:**

Nikto connects to the target web server

Nikto sends HTTP requests to the target and analyzes the responses

Nikto reports any discovered vulnerabilities, misconfigurations, or potential security issues.

**3.5.1 System Feature 5 wapiti:**

Wapiti is an open-source web application vulnerability scanner that is designed to scan the web application for potential vulnerabilities. Some of the system features of Wapiti include:

* Multi-threaded scanner: Wapiti uses multi-threading to scan the target web application, which enables it to perform fast and efficient scans.
* Customizable plugins: Wapiti comes with a wide range of plugins that can be customized to scan for specific vulnerabilities, such as SQL injection and cross-site scripting (XSS) attacks.
* Fingerprinting: Wapiti can identify the web server technology, scripting language, and database used by the target web application, which helps to identify potential vulnerabilities.
* Authentication support: Wapiti can authenticate to the web application using different methods, such as form-based authentication, HTTP Basic Authentication, and Cookie-based authentication.
* Reporting: Wapiti provides detailed reports on the vulnerabilities found during the scan, including the severity level and recommendations for remediation. The reports can be generated in various formats, such as HTML, JSON, and XML.

**3.5.2 Description and Priority in wapiti:**

Wapiti is an open-source web application vulnerability scanner used for identifying security vulnerabilities in web applications. It is designed to perform black-box testing of web applications and can detect several types of vulnerabilities, including SQL injection, file inclusion, command execution, and more.

The priority of Wapiti is to provide a comprehensive and accurate assessment of web application security vulnerabilities. It aims to provide a simple, easy-to-use tool for security professionals and web application developers to identify and fix vulnerabilities before they can be exploited by attackers.

Wapiti's description includes the following features:

* Cross-platform: Wapiti is written in Python and can run on any platform that supports Python.
* Multi-threaded: Wapiti can use multiple threads to scan web applications quickly and efficiently.
* Plugin-based: Wapiti can be extended through plugins, which can be used to add new scanning capabilities or customize existing ones.
* Easy-to-use: Wapiti is designed to be simple and easy to use, even for those without extensive security knowledge.
* Comprehensive reporting: Wapiti provides detailed reports of the vulnerabilities it discovers, including the location of the vulnerability and steps to reproduce it.

Overall, Wapiti is a powerful and flexible tool for web application vulnerability scanning, with a focus on simplicity and ease of use.

**3.5.2 Stimulus/Response Sequences in wapiti:**

The Stimulus/Response Sequences in Wapiti tool for information gathering can be summarized as follows:

Stimulus: The user inputs the target URL and selects the desired scan options.

Response: Wapiti sends requests to the target URL, analyzes the responses, and reports any vulnerabilities found.



Stimulus: The user modifies the scan options to include or exclude certain types of vulnerabilities or tests.

Response: Wapiti adjusts its scanning behavior accordingly and provides updated results.

Stimulus: Wapiti encounters a difficult-to-scan page or a page that requires authentication.

Response: Wapiti can prompt the user for credentials or adjust its scan behavior to bypass certain authentication methods.

Stimulus: Wapiti discovers a vulnerability that requires further investigation or exploitation.

Response: Wapiti can provide information or options for further testing or exploitation of the vulnerability.

Stimulus: The user terminates the scan or Wapiti encounters an error.

Response: Wapiti provides a report of the vulnerabilities found or the reason for the error.

* 1. **Non-Functional Requirements:**

Non-functional requirements are the criteria that define how well the system operates and the attributes that the system must possess, but do not directly relate to its functionality. Here are some non-functional requirements for the automated web reconnaissance tool:

* Performance: The tool must be able to handle large-scale web applications and produce results in a reasonable amount of time.
* Reliability: The tool must be dependable and produce consistent and accurate results.
* Security: The tool should not have any vulnerabilities that can be exploited by hackers and should have proper measures in place to protect the user’s data.
* Scalability: The tool should be able to handle increased workloads and data volumes as the user base grows.
* Usability: The tool should be user-friendly and easy to use for both novice and advanced users.
* Compatibility: The tool should be compatible with a wide range of operating systems and web technologies.
* Maintainability: The tool should be easy to maintain, with clear and concise documentation and well-organized code.
* Portability: The tool should be easily portable to different environments and platforms.

**3.4.1 Design and Implementation Constraints:**

Design and implementation constraints refer to limitations or conditions that restrict the design and implementation process of a system. Some of the design and implementation constraints that may apply to an automated web reconnaissance tool include:

* Compatibility: The tool should be designed to work with multiple operating systems and web browsers to ensure compatibility with various environments.
* Scalability: The tool should be scalable to handle a large amount of data, and the design should accommodate potential growth in data and functionality.
* Usability: The tool should be user-friendly and easy to use, even for non-technical users.
* Security: The tool should be designed with security in mind, such as ensuring the confidentiality, integrity, and availability of data.
* Performance: The tool should be designed to provide fast and efficient performance, even when handling large amounts of data.
* Portability: The tool should be designed to work on different platforms, including desktops, laptops, and mobile devices.
* Maintainability: The tool should be designed to be easy to maintain, update, and enhance over time.
* Accessibility: The tool should be designed to be accessible to users with disabilities, such as those who rely on screen readers or other assistive technologies.
* Legal and regulatory constraints: The tool should be designed to comply with legal and regulatory requirements, such as data protection laws, intellectual property laws, and privacy regulations.
* Resource constraints: The tool should be designed to operate within the constraints of available resources, such as processing power, memory, and storage capacity.
  1. **Hardware and Software Requirements:**

The hardware and software requirements for an automated web reconnaissance tool may vary depending on the specific tool being used. However, some general requirements that may apply to many tools are:

**Hardware requirements:**

A computer with a processor of at least 2 GHz or faster

At least 4 GB of RAM

A hard disk drive with at least 10 GB of free space

A reliable internet connection

**Software requirements:**

A modern operating system, such as Windows 10, macOS, or Linux

Python programming language and relevant libraries for tool development

Third-party tools and libraries for specific functionalities, such as network scanning or web crawling

A web server to host the tool and web application frameworks to develop the user interface

* 1. **External Interface Requirements:**

External interface requirements refer to the interaction between the automated web reconnaissance tool and external entities such as users, hardware devices, and software applications. The external interface requirements for the tool include:

* User Interface: The tool must have a user-friendly interface that is easy to use and understand. It should provide options for inputting user requirements and displaying the output results.
* Database Interface: The tool should be able to interact with the database to store and retrieve information related to the web reconnaissance activities.
* Network Interface: The tool must be able to interact with network protocols to gather information related to the target websites.
* Operating System Interface: The tool should be able to work with different operating systems such as Windows, Linux, and Mac OS.
* Output Interface: The tool must be able to generate reports and outputs in different formats such as text, HTML, and CSV.
* API Interface: The tool should be able to interact with different API interfaces to gather information related to the target websites.
* Browser Interface: The tool should be able to interact with different browsers to gather information related to the target websites.
* Email Interface: The tool should be able to interact with email servers to gather information related to the target websites.

**3.6.1 User Interfaces:**

The user interface of the automated web reconnaissance tool can vary depending on the specific tool or tools being used. However, in general, the user interface should be designed to be user-friendly and intuitive, with clear and easy-to-understand menus, buttons, and icons. The user interface should also provide real-time feedback to the user as the tool is running, indicating progress, errors, and any other relevant information.

Some common elements that may be included in the user interface of an automated web reconnaissance tool include:

* Input fields: Where the user can enter the target website URL or other relevant information.
* Scan options: Where the user can select which types of scans to run, such as vulnerability scanning, subdomain enumeration, or directory enumeration.
* Progress indicators: Which show the status of the scan as it runs, including any errors or warnings.
* Results display: Where the tool displays the results of the scan, organized in a clear and easy-to-understand format.
* Export options: Where the user can export the results of the scan in a variety of formats, such as PDF or CSV, for further analysis or reporting.

**3.6.2 Hardware Interfaces:**

Hardware interfaces refer to the physical components and devices that the system interacts with. In the case of the automated web reconnaissance tool, the hardware interfaces are minimal as the tool runs on a computer system and does not require any specialized hardware. The tool can be run on a standard desktop or laptop computer with the following minimum hardware requirements:

Processor: Intel Pentium 4 or higher

RAM: 2GB or more

Hard disk space: 100MB or more

Additionally, if the tool is used for scanning large networks or websites, a higher-end system with more processing power and RAM may be required to ensure optimal performance.

**3.6.3 Software Interfaces:**

Software interfaces in the automated web reconnaissance tool refer to the software systems, platforms, and programming languages used to build and integrate the tool. Some of the software interfaces required for the tool may include:

* Operating System: The tool may be designed to work on a particular operating system like Linux, Windows, or MacOS.
* Programming Languages: The tool may be developed using one or more programming languages such as Python, Ruby, or Go.
* Web Frameworks: The tool may use web frameworks like Flask or Django to develop the web application part of the tool.
* Database: The tool may require a database system like MySQL or PostgreSQL to store information and results.
* APIs: The tool may use APIs provided by other services to gather information, such as the Shodan API or Virus Total API.
* Web Browsers: The tool may require the use of web browsers like Firefox or Chrome to render web pages or interact with web-based services.
* Network Scanning Tools: The tool may integrate with other network scanning tools like Nmap or Masscan to gather more comprehensive information about the target.

**3.6.4 Communication Interfaces:**

The communication interfaces in automated web reconnaissance tool refer to the protocols and standards used by the tool to communicate with other systems and networks. Some of the communication interfaces that may be relevant for the tool include:

* HTTP/HTTPS: These are the most common protocols used for web communication, and the tool may need to use them to access web resources.
* DNS: The tool may need to query DNS servers to resolve domain names into IP addresses.
* SMTP: If the tool includes email-based features, it may need to use the SMTP protocol to send emails.
* API: The tool may need to integrate with other systems or services using APIs, which could include RESTful APIs or other web-based APIs.
* CLI: The tool may need to support a command-line interface for integration with other tools or for scripting.
* File Formats: The tool may need to support various file formats for importing or exporting data, such as CSV, XML, or JSON.
* Database: The tool may need to interface with databases to store and retrieve information.

**3.7 Feasibility Analysis of the Requirements:**

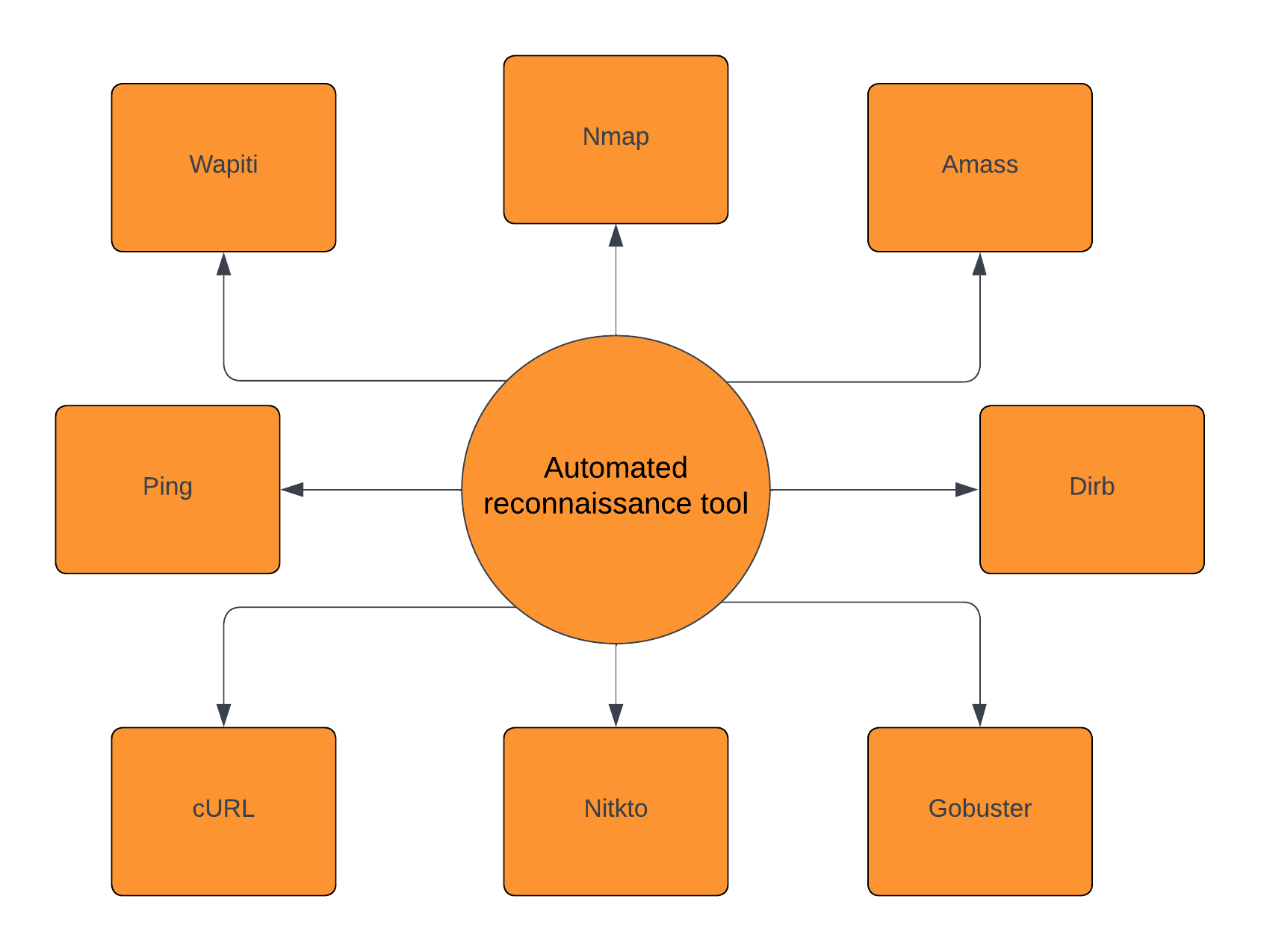
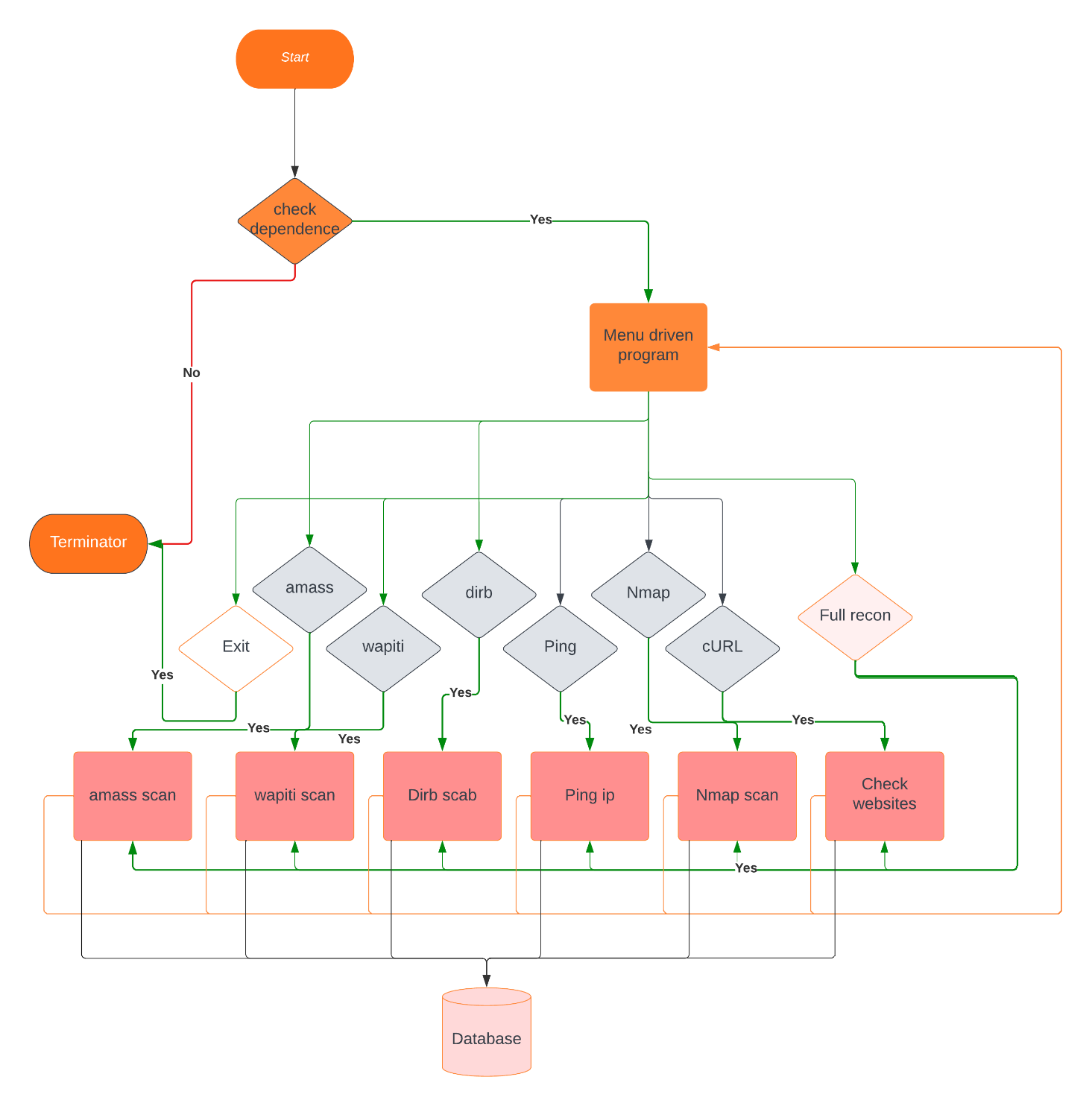
Feasibility analysis is a process of evaluating the proposed system to determine whether it is technically, economically, and operationally feasible. In the case of the automated web reconnaissance tool, the feasibility analysis of the requirements involves evaluating the feasibility of the hardware and software requirements, user interface design, software and communication interfaces, and other factors that could affect the successful implementation of the system.

The feasibility analysis of the automated web reconnaissance tool should consider the following:

* Technical feasibility: This involves evaluating whether the proposed system can be developed and implemented using the available technology and resources. It also involves evaluating whether the system can handle large volumes of data and traffic without crashing or slowing down.
* Economic feasibility: This involves evaluating the cost-benefit analysis of the proposed system. It includes the cost of hardware, software, and other resources required for the development and implementation of the system.
* Operational feasibility: This involves evaluating whether the proposed system can be integrated into the existing infrastructure and whether it can be easily used and maintained by the end-users.
* Schedule feasibility: This involves evaluating whether the proposed system can be developed and implemented within the allocated time frame.

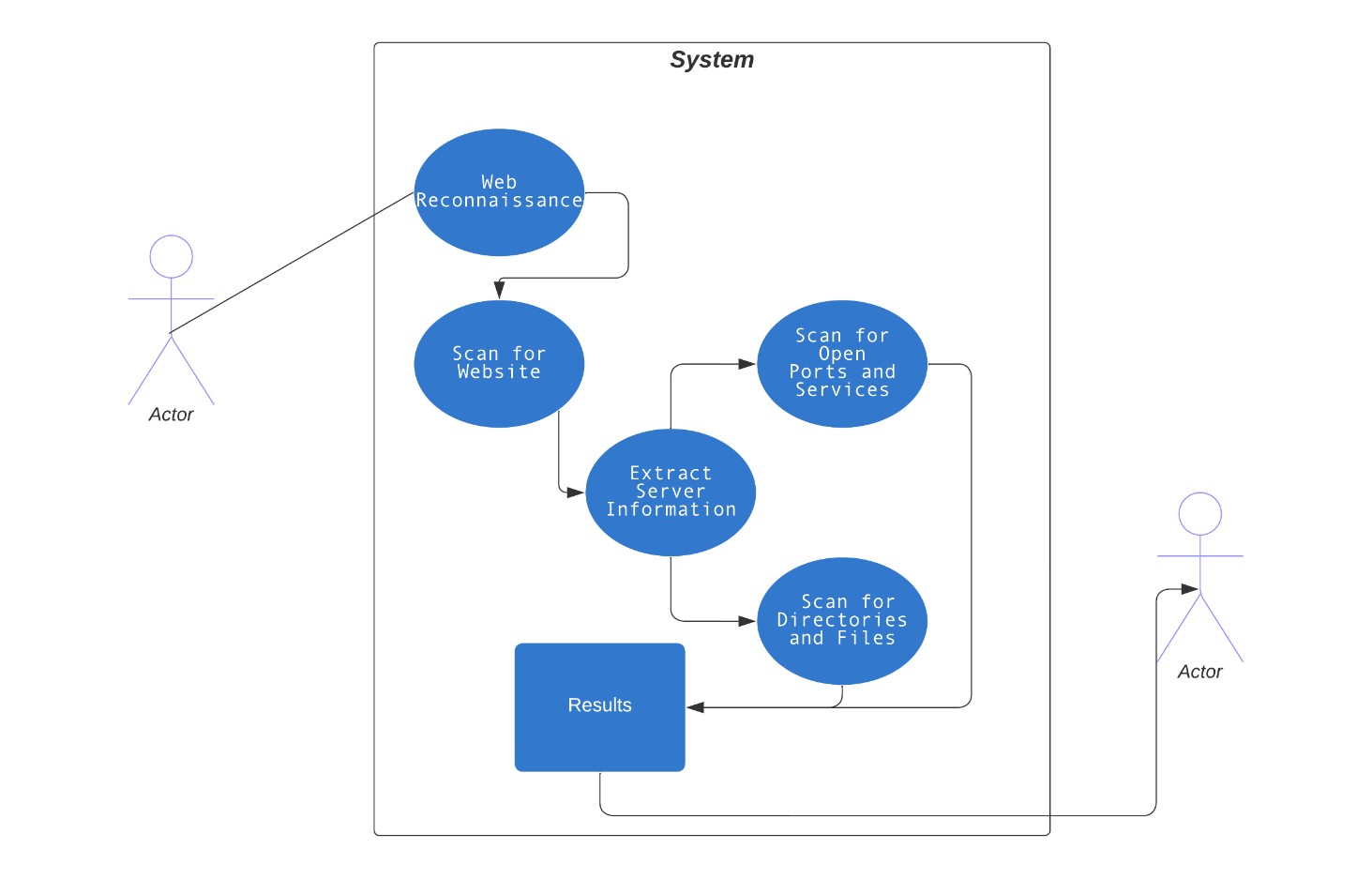
**4. USER INTERFACE DESIGN**

* 1. Screen Element Requirement Analysis
  2. **Screen Interfaces

1. ****Appendices**
   1. **Context Diagram**
   2. ****Data Flow Diagram**

**

* 1. **Use Case Diagram**

****