National College of Ireland

A Novel Web Application security vulnerability scanning tool.

MSc Research Project
Industry Internship

Abhay Singh
Student ID: X21212341

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Abhay Sureshkumar Singh
Student ID:	X21212341
Programme:	Masters in Cybersecurity **Year:** 2022-2023
Module:	Industry Internship
Supervisor:	Vikas Sahni
Submission Due Date:	04-09-2023
Project Title:	A novel Web Application security vulnerability scanning tool.
Word Count:	7203 (including references) **Page Count:** 22

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

Signature: Abhay Singh

Date: 04-09-2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

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

A Novel Web Application security vulnerability scanning tool.

Abhay Singh
X21212341

Abstract

Web Application usage has been increasing day-by-day as organizations provide variety of services based on people's daily life requirements. Securing these web applications and network infrastructure has become a crucial task. There are various vulnerability scanners available in different programming languages with multiple functionalities with an ability to handle specific vulnerabilities, but they are unnecessarily complex in nature for developers as well as end-users. This study aims to fill this gap by investigating the benefits, challenges, and best practices associated with developing a Flask-based vulnerability scanner. By utilizing Flask's light weight and flexibility, the scanner offers modularity, extension, and easy connection with Flask-based apps.

With the Flask framework in mind, this research work presents a state-of-the-art vulnerability scanning tool for online applications that increases threat assessment coverage and mitigation tactics. The developed tool hides the complexity of underlying API calls by combining open-source tools detection method and running a comprehensive rule-based system, data format conversion, and optimized workflow. A user-friendly, non-technical executive summary is created from the tool's output, assisting in better risk understanding and remediation techniques.

***Keywords:* - Flask framework, Web Vulnerability Scanner, threats, Issues, Nessus, Skipfish, rules, vulnerability detection**

1 Introduction

1.1 Background and Rationale

The development of the internet, the World Wide Web, and web applications has a significant influence on every aspect of professional and interpersonal human life. Because the web application is the initial face of all enterprises worldwide, it was always expanding quickly and becoming more sophisticated. The fact that a web application has vulnerabilities is owing to the fact that it was created with less programming expertise and without being tested for flaws.(Ablahd, 2023). With Broken Access Control being the first among the OWASP top 10 vulnerabilities, is termed as a vulnerability to a web application and many others like it.

There are nearly endless attack possibilities, even if there are many factors to consider when assessing a vulnerability scanner by comparing the vulnerability scanning coverage, precision, recall, and time complexities.(Tung *et al.*, 2013) Previous studies have suggested a variety of evaluation matrices due to the large diversity of security risks. While insufficient procedures sometimes result in inaccurate detection by the tools, a thorough scan frequently generates many vulnerability alarms.(Tung *et al.*, 2013).

A web application is always based on what framework it is built upon. Considering Python as the web application programming language and the area of research in this paper, it offers few web development frameworks. Due to its feature like, readable syntax, minimal

boilerplate, high-level data structures to avoid complexity, and extensive standard library support for file handling, networking, and many other functionalities, it supports programmers with reduced work of logic building and design implementation making it their best choice to opt for this language.

Framework	Description
Django	High-level framework for rapid development with built-in features and an admin panel.
Flask	Lightweight micro-framework offering flexibility and essential tools for web applications. A Flask extension framework is used to provide support for functionalities that are not covered by the micro-framework itself.
CherryPy	Object-oriented framework that maps Python objects to HTTP requests, suitable for small apps.
Sanic	Asynchronous framework inspired by Flask, designed for efficient handling of concurrent workloads.

Table 1: Web Development frameworks for Python

While vulnerability scanners exist in the market, there is a lack of research and development specifically tailored to the Flask framework. Many existing scanners are built on different frameworks, making integration with Flask-based applications cumbersome and complex. By exploring the Flask framework for vulnerability scanning, this research aims to bridge the gap and provide insights into building a scanner that aligns seamlessly with Flask's principles and practices.

Flask is renowned for its simplicity, flexibility, and modularity, which are highly advantageous when developing web applications. Leveraging these qualities for vulnerability scanning can lead to a scanner that is lightweight, customizable, and easily adaptable to diverse projects. This research topic allows for exploration on how Flask's strengths can be harnessed to create a specialized vulnerability scanner.

1.2 Aim

How to develop an effective web application vulnerability scanner tool?

1.3 Objectives

- Creating an effective web vulnerability scanner using Flask's minimalist architecture and extensible features to scan networks and applications for potential vulnerabilities.
- Building a scanner to provide a flexible scanning strategy by permitting users to dynamically load and manage vulnerability scanning rules.
- Integrate smoothly with Flask's routing, to provide a set of API endpoints, allowing external systems or tools to interact with its capabilities.

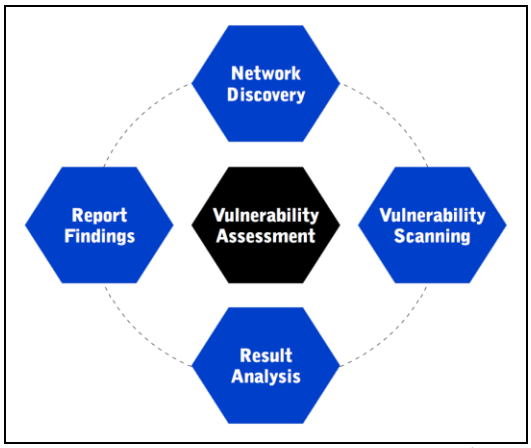


Fig.1 Vulnerability Assessment Cycle.[1]

2 Related Work

Web vulnerability scanning involves the systematic identification and assessment of security vulnerabilities in web applications. Various approaches have been proposed, including static analysis, dynamic analysis, and hybrid techniques. The literature extensively discusses the principles, challenges, and methodologies related to web vulnerability scanning.

The paper (R *et al.*, 2023) discusses the Axiom technique used in research study to provide a unique framework for automated web application security screening and information collecting. By automating asset discovery, filtering targets based on parameters like open ports, and utilizing the Axiom methodology for quicker and distributed scanning, the framework helps find and fix vulnerabilities in web applications, helpful for bug bounties and penetration testing. In addition to showcasing the framework's potential to improve online application security efficiency and scalability, the research assesses its performance, constraints, and obstacles. The study also examines the logic behind OWASP vulnerabilities and provides a comparative review of current automated security screening solutions.

The paper (Ablahd, 2023) discusses detection of website vulnerabilities using Python with multiple frameworks brought in focus. The author mentioned multiple frameworks like Django, Flask, AST, CFG while using it for building a vulnerability scanner tool called SCANSCX. The tool is divided into two sections - non-full stack micro-web framework chosen as Flask and a full-stack web development framework selected was Django. The paper only uses Flask framework for completion of the web application and does not make any use of security modules provided by flask extension. It can be utilized as a part of research in the proposed tool development completely based on Flask.

The study discussed in the article (Laksmiati, 2023) focuses on vulnerability assessment on WordPress-based websites to find potential security holes that might be exploited by hackers. Network-based scanning tools like Nmap and WPScan are used in the study to find and examine website vulnerabilities. The findings point to a number of vulnerabilities that

[1] https://transform-mpi.com/training-tentang-security-vulnerability-assessment/

need to be addressed and mitigated right once. To reduce the danger of cyberattacks, it is important to regularly undertake vulnerability assessments on WordPress websites.

The research article by (Chen et al. (2020)) provides a powerful web vulnerability scanner made to handle the security issues brought on by web applications. The scanner is designed to collect data and check out vulnerability detection thoroughly, improving online security assurance. The suggested scanner in this work blends information gathering with vulnerability detection and assists in identifying typical online vulnerabilities like SQL injection and XSS. The structure, module designs, and functionalities of the scanner are described in the article.

The (Tang and Zhou, 2021) research presents a cutting-edge web vulnerability scanner that makes use of an intelligent crawler strategy. This approach focuses on examining URL characteristics that are crucial to the scanner's functionality. In light of these traits, the scanner effectively eliminates pointless URLs using attribute tags. The research introduces a unique four-fold bloom filter that takes into account both geographical and temporal complexity to improve the URL deduplication process. By drastically reducing the amount of storage space needed and the amount of time needed for URL deduplication, this method offers substantial advantages that will encourage wider usage.

The research paper (Odion *et al.*, 2023) introduces the "Vulnerability Scanner (VulScan)," a system designed to identify six categories of web application attacks. It addresses the growing challenges of web security by actively scanning components like links, forms, and headers. The paper extensively discusses the technical workings of the scanner, employing tools like Selenium WebDriver and requests library for vulnerability detection modules. It emphasizes both active and passive operations for different vulnerabilities, such as XSS, CSRF, DoS, and more. Furthermore, the reporting module is highlighted, showcasing the importance of clear vulnerability communication. The paper presents a comprehensive approach to web security, detailing methodologies, technical implementation, and a wider coverage of vulnerabilities.

The study (Sagar *et al.*, 2018) compares the effectiveness of three web vulnerability scanners (OWASP ZAP, Skipfish, and w3af) against the top 10 web application risks listed by OWASP. In the experiment, susceptible programs like DVWA are put to the test, and the outputs of the scanners are examined for flaws like SQL injection, Cross-Site Scripting (XSS), Broken Authentication, and others. The capacity to accommodate input vectors, audit features, and vulnerability identification is compared for each scanner in the article. It offers information on the many aspects of scanning, including as crawling, attacker modules, and analysis tools.

The assessment of vulnerability scanners for online applications is covered in (Tung *et al.*, 2013) research article, with a particular emphasis on the problem of repeated vulnerability alarms. By expanding the confusion matrix to account for genuine duplication (TD) and false duplication (FD) of vulnerabilities, the authors provide a method for cost-effective assessment. They provide a methodology that takes the price of checking duplicate vulnerabilities into account and use this method to construct the Web Vulnerability Scanner Testbed (W-VST). The paper examines the improved precision and F-measure formulae and

stresses the significance of taking false positives and redundancy into account when evaluating scanners.

The (Zukran and Siraj, 2021) research investigates how well open-source automated vulnerability scanners perform in identifying SQL Injection and Cross-Site Scripting (XSS) flaws in web applications. The authors use a technique to assess scanner detection coverage and accuracy rates. They test OWASP ZAP and Skipfish, two well-known open-source scanners, on weak web apps. The importance of accuracy and coverage in vulnerability identification is discussed in the study, which also gives conclusions based on vulnerability categories and risk levels.

To comprehend the characteristics and efficacy of web application vulnerability scanners (WVS), the study (Alazmi and De Leon, 2022) focuses on a systematic assessment of WVS. It thoroughly examines 90 research publications and discovers 30 Web Application Vulnerability Scanners (WVS), only 12 of which include quantitative evaluations of their efficacy. In particular, SQL Injection (SQLi) and Cross-Site Scripting (XSS) are discussed as OWASP Top Ten vulnerabilities that these scanners may identify. The results show differences in reported detection rates between several evaluation studies. The study emphasizes the requirement for more thorough testing and assessment of WVSs in order to handle the changing web application security scenario.

The research (Farrell, 2022) examines how Content Management Systems (CMS) like WordPress can automate threat assessment coverage. By developing an automated WordPress vulnerability screening tool, it seeks to close a crucial gap in the detection and remediation of problems. The article analyses the efficiency of current security analysis tools (such as Black Kite, Security Scorecard, BitSight, and WPScan) in lowering costs and broadening threat coverage. The approach uses secondary case-study analysis to examine the capabilities, data sources, detection techniques, and degree of work needed for threat assessment for each instrument.

The development of a web application vulnerability scanning system for the power industry is covered in the (Wang *et al.*, 2019) research paper. The fundamentals of the system are covered, with a focus on web crawler-based passive and active scanning techniques. The document explores many vulnerability types, including XSS and SQL injection, and details how to find them. It also describes the system's essential elements, such as the web crawler and detection modules, and emphasizes the tool's adaptable plug-in nature.

The (Jain *et al.*, 2023) research paper introduces a web application vulnerability scanning system for network security, emphasizing vulnerability analysis to detect hidden weaknesses. It focuses on revealing implicit discrepancies that can harm the network, including vulnerabilities like Cross-site scripting (XSS), Cross-site request forgery (CSRF), DDoS, and SQL injection. It evaluates different scanners, and proposes an innovative prototype called "JARVIS" that offers scanner agnostic capabilities. The paper concludes with results showing how the system identifies vulnerabilities and improves website security.

The (Zhang, Hu and Huo, 2021) research paper presents a browser-based Cross Site Request Forgery (CSRF) detection model, addressing the serious threat of CSRF attacks in web applications. The model examines HTTP requests and web page content to spot possible CSRF attacks, offering defense against attackers that take advantage of browser cached

cookies and implicit authentication. The procedure of a CSRF assault is discussed in the article, along with server-based and client-based CSRF protection strategies, and a browser-based detection model using Google Chrome is suggested. The model design includes a CSRF handler module, which notifies users of suspected CSRF sites and gives them the option of continuing or not, as well as HTTP request analysis and content analysis.

To address software vulnerabilities and highlight the necessity of security measures to avert assaults, the (Kharat and Chawan, 2022) research study presents a Vulnerability Management System (VMS). A vulnerability database, data processing platform, and vulnerability scanner are all part of the suggested VMS. It describes a four-step procedure that includes locating vulnerabilities, assessing how serious they are, implementing the necessary fixes, and producing thorough vulnerability reports. In order to find vulnerabilities and rank their severity, the system performs binary and source code analysis.

An integrated framework for software vulnerability identification, analysis, mitigation, and management based on autonomous computing is proposed in (Kumar and Sharma, 2017) research study. The framework integrates autonomic computing's self-configuration, self-healing, self-prevention, and self-optimization features. With the goal of lowering security risks, threats, and financial loss while improving software security, the framework's self-managing features provide cross-cutting security improvements, automatic scanning, and intelligent vulnerability evaluation.

The (Susanto, Rizko and Purbohadi, 2020) research paper focuses on conducting vulnerability assessment to enhance website security, with a specific focus on WordPress-based sites. The study detects vulnerabilities using network-based scanners like Nmap and WPScan. WordPress websites are vulnerable to hacker attacks due of their popularity, according to the article. The process covers requirement analysis, design, testing, implementation, and outcome analysis. The results show open ports, older software versions, and WordPress feature, configuration, and theme vulnerabilities. The study emphasizes regular vulnerability assessments and proactive risk mitigation for enterprises and users.

The (Xu *et al.*, 2022) research study develops a web vulnerability identification analyzer for security-conscious enterprises as well as individuals. The study discusses sub-domain scanning, application fingerprint recognition, web crawling, and vulnerability verification. The technology detects SQL injection and XSS in target webpages using fuzzing and PoC verification. The paper describes the analyzer's architecture, components, and workflow, showing how it effectively finds and confirms online vulnerabilities. The proposed technique reduces penetration testing time and improves online security by correcting vulnerabilities quickly.

The (Verma, 2023) research paper investigates Insecure Deserialization vulnerabilities and their exploitation across multiple programming languages. It discusses how serialization and deserialization can be exploited by adversaries to compromise systems, delving into the serialization and deserialization processes. This paper explains vulnerability exploitation techniques in PHP, Java, and Python, including magic methods, POP chains, and gadgets. The highlighted prevention techniques include input validation, allowlisting, and encryption. The paper proposes the development of a Python-based automated vulnerability scanner for Insecure Deserialization, delineating modules for port scanning, OS and service detection,

data extraction, and vulnerability verification. It stresses the significance of protecting human-editable data, such as cookies, from exploitation.

The (Parimala, Sangeetha and AndalPriyadharsini, 2018) research paper titled "Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery" addresses the security threat posed by Cross-Site Request Forgery (CSRF) in web applications. It proposes a Python-based automated utility for detecting CSRF vulnerabilities in URLs and local host addresses. This paper examines the historical context of web vulnerabilities, CSRF attacks, various attack methods, and the significance of automated security testing tools. It explains session management vulnerabilities, investigates the CSRF attack mechanism, and provides examples of CSRF attacks. The proposed system includes real-time scanning for CSRF vulnerabilities, token analysis, and Same Origin Policy verification. The implementation incorporates a user-friendly graphical user interface for input and output display.

The study (Saabith, Fareez and Vinothraj, 2019) paper discusses Python's popularity, simplicity, community, web development, and big data and machine learning applications. Python is examined in corporate applications, software development, and language design. Django, Web2Py, TurboGears, Pyramid, Flask, CherryPy, and other Python web development frameworks are also introduced in the article, along with their important features and use cases. The paper introduces Flask as a lightweight micro web framework compatible with Google App Engine, built-in development server, and debugger. It highlights Flask's lightweight nature and applicability for smaller projects in the Python web development framework ecosystem.

The research paper (Ghimire, 2020) contrasts two web applications developed with the Flask and Django frameworks. It concentrates on the Flask-based application's design patterns, requests and routing, blueprints, infrastructure, configurations, security, deployment, error handling, caching, and adaptability. The Flask application employs the recommended directory structure, view decorators, automatic request context handling, blueprints for modularity, and an ORM (Peewee) to handle database interactions. The document discusses security measures, configuration management, Heroku deployment, error handling with Python exceptions, and caching options. The adaptability of Flask is highlighted, as it permits the simple incorporation of libraries and extensions for various aspects of web development, ultimately expediting development while adhering to the MVC design pattern.

The paper (Lokhande *et al.*, 2015) explores web development with Python and Flask efficiently. It highlights Python's simplicity, vast libraries, and error-handling skills. The study emphasizes Flask's micro framework role in simplifying core functionality and facilitating plugin extension. Flask's structure, Jinja2, and Werkzeug WSGI Toolkit are covered. Template inheritance, file organization, and Flask program implementation are covered. The paper also discusses Jinja2 templating's sandbox execution and template inheritance features. The focus is on fast web development, open-source collaboration, and Python and Flask's benefits in constructing attractive and functioning web portals.

The "Web Application Intrusion Detection System for Input Validation Attack" research paper by (Park and Park, 2008) describes a Web Application Intrusion Detection System (WAIDS) based on an Anomaly Intrusion Detection model. This paper discusses the

detection of input validation assaults against web applications, including the challenges posed by such attacks and the limitations of traditional methods such as signature-based detection. The proposed method for detecting anomalous requests includes data collection, keyword extraction, similarity measurement using the Needleman-Wunsch algorithm, and filtering/reporting. Using profiles generated from standard web request data, the study exhibits improved detection rates and reduced false positives. The experimental results demonstrate that the WAIDS approach is effective at detecting input validation attacks.

The (Goethem *et al.*, 2014) research paper "Large-scale Security Analysis of the Web: Challenges and Findings" examines web security on over 22,000 websites from 28 EU nations. It discusses web application vulnerabilities and security measures to prevent typical attacks. The paper provides a security score system that evaluates defenses and vulnerabilities. Security mechanisms (HTTP Strict Transport Security, Secure Cookies, Content Security Policy, etc.), vulnerability detection (Vulnerable Remote JavaScript Inclusion, Mixed-content Inclusion, SSL-stripping Vulnerable Form, etc.), and a Common Weakness Scoring System-based scoring system are covered. The study stresses the importance of external security evaluations and offers web application security tips.

Table 2: Summarized Literature Review Table

Sr No.	**Research Author and Paper details**	**Advantages**	**Disadvantages**
1.	**(R *et al.*, 2023)** Web Application Security Testing Framework using Flask	The distributed and automated Axiom technique increases online application security screening and vulnerability identification.	Certain vulnerabilities or network environments may limit the framework's practicality.
2.	**(Ablahd, 2023)** Using Python to Detect Web application vulnerability	Researchers designing Flask-based security solutions can include SCANSCX vulnerability scanning using the paper's methods.	The paper's absence of Flask extension security modules may limit its Flask app security assessment.
3.	**(Laksmiati, 2023)** Vulnerability Assessment with Network-Based Scanner Method for Improving Website Security	The study finds vulnerabilities using Nmap and WPScan and provides insights into successful vulnerability assessment.	The frequent usage of Nmap and WPScan may reduce vulnerability analysis depth by hindering other evaluation methods. The study does not compare scanning methods.
4.	**(Chen et al. (2020))** An Automatic Vulnerability Scanner for Web Applications	The document describes the scanner's structure, module designs, and functions to enable researchers and practitioners to use it. A powerful web application security scanner, it increases online security assurance and may prevent cyberattacks.	Scalability of the scanner in handling large-scale online applications is not adequately examined, raising concerns about its performance and efficiency in complex systems.
5.	**(Tang and Zhou,**	The novel four-fold bloom filter for	The study illustrates the benefits

	2021) Design and Implementation of High-performance Web Vulnerability Scanner Based on Python Intelligent Crawler	URL deduplication considers geographical and temporal complexity. This solution saves storage space and deduplication time, enabling large-scale scanning.	of the suggested strategy, but it may not address situations where it may be less effective, such as specific websites or target regions.
6.	**(Odion *et al.*, 2023)** VulScan: A Web-Based Vulnerability Multi-Scanner for Web Application	VulScan, a robust vulnerability scanner that detects six web application attacks. This wide coverage enhances vulnerability detection. It describes VulScan's Selenium WebDriver and requests library implementation.	The paper's technical complexity may make adoption harder for less experienced readers or practitioners without an online security background.
7.	**(Sagar *et al.*, 2018)** STUDYING OPEN-SOURCE VULNERABILITY SCANNERS FOR VULNERABILITIES IN WEB APPLICATIONS	The study compares OWASP ZAP, Skipfish, and w3af to the Top 10 web application risks. This comparison shows pros and cons. The study examines scanner input vector flexibility, auditing, and vulnerability identification.	The study's planned test scenarios and susceptible programs may not accurately represent internet vulnerabilities' dynamic nature and scanners' real-world effectiveness.
8.	**(Tung *et al.*, 2013)** A cost-effective approach to evaluating security vulnerability scanner	The paper advises evaluating repeated-warning vulnerability scanners with an enlarged confusion matrix. The Web Vulnerability Scanner Testbed analyzes vulnerability scanner proposals. This controlled environment evaluates scanners.	The paper's concentration on duplicate vulnerability alarms may limit its assessment and overshadow scanner performance and efficacy.
9.	**(Zukran and Siraj, 2021)** Performance Comparison on SQL Injection and XSS Detection using Open-Source Vulnerability Scanners	The study tests open-source vulnerability scanners on poor web apps to apply the findings to real security settings. The study evaluates scanner detection coverage and accuracy for SQL Injection and XSS vulnerabilities.	The study only examines two open-source vulnerability scanners, OWASP ZAP and Skipfish, which may not cover the complete vulnerability scanner environment.
10	**(Alazmi and De Leon, 2022)** A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners	To understand 30 Web Application Vulnerability Scanners (WVS), 90 research publications were analyzed to determine their characteristics and efficacy. Quantitative evaluations in 12 of 30 WVS underline the relevance of empirical data in measuring vulnerability scanner performance and dependability.	The study emphasized quantitative evaluations, although only 12 of 30 WVS included them, limiting its statistical significance. Existing research publications may bias the assessment toward positive results or vulnerability scanners, lowering its comprehensiveness and generalizability.
11	**(Farrell, 2022)**	The study evaluates security	Due to case and data selection,

	Abstraction and automation of WordPress vulnerability scanning	analysis technologies' efficiency to help practitioners choose threat assessment tools based on their capabilities, data sources, detection methodologies, and resource needs.	secondary case-study analysis may bias and limit research generalizability.
12	**(Wang *et al.*, 2019)** Research on Web Application Security Vulnerability Scanning Technology	Passive and active scanning, vulnerability categories, and plug-in adaptation are covered in the study, showing a comprehensive web application security methodology.	The research should evaluate the scanning system's efficacy, performance, and real-world applicability.
13	**(Jain *et al.*, 2023)** Web Scanner: An Innovative Prototype for Checking Web Vulnerability	Web application vulnerability assessment finds explicit and implicit vulnerabilities, including advanced threats like DDoS, XSS, CSRF, and SQL injection, according to the research.	Adapting the system to different network setups and sectors may require more research and validation than the paper provides.
14	**(Zhang, Hu and Huo, 2021)** A Browser-based Cross Site Request Forgery Detection Model	The study introduces a browser based CSRF detection approach that examines HTTP requests and web page content to improve web application security by addressing a specific vulnerability.	Practical tests or case studies showing the detection model's efficacy and adaptability across online apps and settings would help the study.
15	**(Kharat and Chawan, 2022)** Vulnerability Management System	The study proposes a Vulnerability Management System (VMS) with a vulnerability database, data processing platform, and vulnerability scanner to handle software vulnerabilities holistically.	Real-world applications or case studies could validate the Vulnerability Management System in software settings. Integration of binary and source code analysis may present complexity and resource difficulties that must be addressed for practical usability and efficiency.
16	**(Kumar and Sharma, 2017)** An integrated framework for software vulnerability detection, analysis, and mitigation: an autonomic system	Self-managing features like automatic scanning and intelligent vulnerability evaluation can improve vulnerability management efficiency and save manual effort.	An integrated framework based on autonomous computing may be difficult to create, integrate, and integrate with current software environments.
17	**(Susanto, Rizko and Purbohadi, 2020)** Security Assessment Using Nessus Tool to Determine Security Gaps on the Repository Web Application in Educational	The paper tackles a real-world issue by focusing on vulnerability assessment for popular WordPress-based websites, providing practical insights into detecting and addressing security vulnerabilities.	The study focuses on WordPress-based websites, but it may benefit from exploring and addressing a wider range of vulnerability assessment methodologies beyond network-based scanners to present a more complete picture.

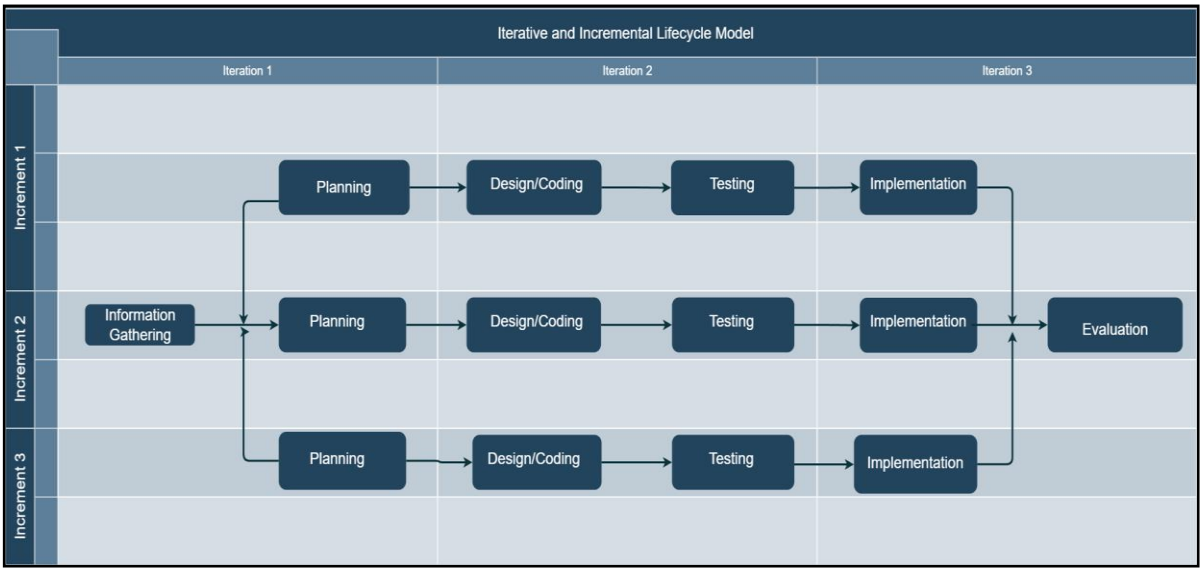
	Institutions		
18	**(Xu *et al.*, 2022)** Web Vulnerability Detection Analyzer Based on Python:	The technology's focus on fuzzing, PoC verification, and reduced penetration testing time shows its ability to swiftly uncover and fix flaws, improving online security.	The study focuses on SQL injection and XSS vulnerabilities, limiting its relevance to other web issues enterprises may encounter.
19	**(Verma, 2023)** Insecure Deserialization Detection in Python	The study examines Insecure Deserialization vulnerabilities in PHP, Java, and Python, revealing common attack methods and preventive solutions.	Due to the intricacy of Insecure Deserialization vulnerabilities, the document may not cover all exploitation scenarios or preventative techniques.
20	**(Parimala, Sangeetha and AndalPriyadharsini, 2018)** Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery	A Python-based automated application that efficiently detects Cross-Site Request Forgery (CSRF) vulnerabilities is proposed in the paper as a realistic solution.	The research suggests an automated utility; however it may benefit from a comparison with existing CSRF detection tools to demonstrate its efficiency and effectiveness.
21	**(Saabith, Fareez and Vinothraj, 2019)** Python current trend applications-an overview	The article emphasises Flask as a lightweight micro web framework to demonstrate its interoperability, built-in features, and applicability for smaller applications, giving readers practical insights into its possibilities.	The paper offers numerous Python web frameworks, but a comparative examination of their merits and disadvantages could help readers choose one for specific tasks.
22	**(Ghimire, 2020)** Comparative study on Python web frameworks: Flask and Django	The article covers Flask-based web application development's design patterns, security, deployment, error handling, and adaptability, giving readers a complete grasp.	The study compares Flask to Django, although project complexity and requirements may limit its applicability. The study may lose out on a holistic view of both frameworks' strengths and drawbacks.
23	**(Lokhande *et al.*, 2015)** Efficient way of web development using python and flask	New web developers can benefit from Python and Flask's simplicity, broad libraries, fast development, and open-source collaboration, according to the research.	The article introduces Python and Flask for web development, however experienced developers may want more technical details or advanced subjects.
24	**(Park and Park, 2008)** Web Application Intrusion Detection System for Input Validation Attack	An anomaly-based intrusion detection model and the Needleman-Wunsch algorithm for similarity assessment demonstrate the paper's new approach to web application assault detection beyond signature-based methods.	While effective for input validation assaults, the paper's concentration on a single sort of attack may limit its applicability to other web vulnerabilities and security concerns.
25	**(Goethem *et al.*, 2014)** Large-Scale Security Analysis of the Web: Challenges	The suggested security score system, security mechanism coverage, vulnerability detection, and Common Weakness Scoring	The research focuses on EU websites; hence its results and recommendations may not apply to non-EU websites, limiting its

	and Findings	System-based scoring provide web developers and security professionals with realistic direction on web security assessment and enhancement.	worldwide influence.

Many of the assessed research studies lack temporal relevance due to the rapid change of technology and web security policies. Some older methods may not adequately reflect online security or the latest vulnerabilities and mitigation solutions since technology and threats change. This constraint proves to the necessity for current research to address web security issues. This research's application improves current vulnerability detection and provides a variety of threat detection to cover all conceivable web application issues.

3 Research Framework

This project develops a web application vulnerability detection system using an Iterative and Incremental SDLC approach. Due to its versatility, the Iterative and Incremental Approach is used to modify and improve the system through repeated planning, design/coding, testing, and implementation cycles. A thorough Requirement Gathering and Evaluation step establishes the project's scope, objectives, and vulnerabilities, laying the framework for succeeding iterations. Each iteration cycles planning, design/coding, testing, and implementation. Each cycle improves system security by fixing vulnerabilities found during Information Gathering. This method keeps the detection system updated on new threats and exploits. User feedback also improves system performance in subsequent generations. Continuous testing, integration, automated security scanning, and code reviews improve the whole solution's quality and security. This study technique uses the Iterative and Incremental Approach, Information Gathering, and Evaluation to create a resilient and flexible web



application vulnerability detection system for emerging security concerns.

Figure 2. Iterative and Incremental Model

4 Design Specification

The web application executes a vulnerability scan based on the entered target URL or IP address. The preceding section attempts to provide a summary of the web application's flow sequence. As soon as the shell script is executed to launch the application, the start_workers() module and the Redis server are initialized to initiate a perpetually running web server database on the backend. The Workers module initiates the Attacker, Scanner, and Scheduler functions. These are the three most essential features of this web application that are crucial for detecting vulnerabilities.
The Attacker component launches attacks against specified IPs and ports based on the configuration provided. It applies principles to scanned data and operates concurrently using threads.

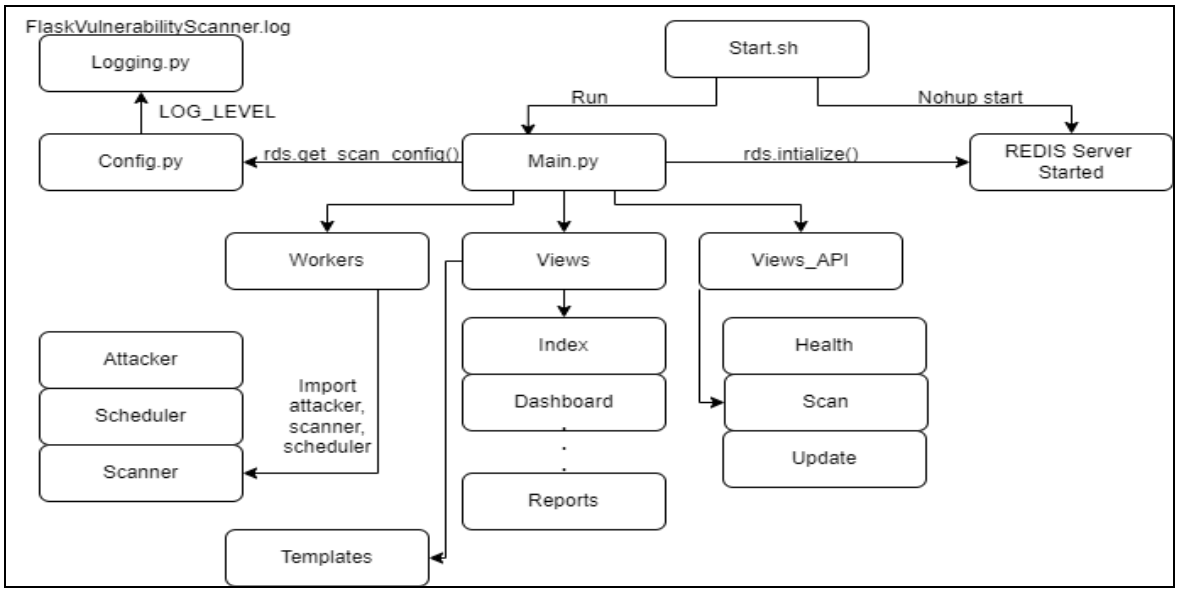


Fig.3 Flow Diagram of Proposed Scanner

The Scheduler component orchestrates the total operation. Based on the supplied configuration, it manages the scheduling of network scans and the execution of post-assessment actions. It ensures that the scanning and evaluation process is governed and adheres to the predetermined frequency.
The Scanner module conducts network searches to identify assets and open ports. It schedules IP addresses and domains for scanning, performs scans, and maintains scan results in Redis.
The Manager module verifies that the specified role exists. It checks for vulnerabilities and threats based on the supplied IP address or URL by executing all types of rules specified in the rules directory and executing all desired rules. The Scheduler retrieves IP Addresses utilizing Python functions. This IP address is retrieved from Redis data by the Scanner function to determine if a scan is to be performed. Using the logger module, all error messages and successful transactions are logged, and the scanned data is then stored in the Redis database in the form of results.

5 Implementation

5.1 Specification about added functionalities

Flask Application (Main.py) Security functionalities:

Configuration and Secret Key:
The configuration of the application is altered with certain values, including the SESSION_COOKIE_SAMESITE setting. Moreover, a secret key is generated and assigned

as the application's secret key. This confidential key is essential for secure data transmission and session management.

Security Headers:
This is an after-request hook that appends several security-related response attributes. These identifiers help defend the application against a variety of attacks. These header values are obtained from the config module.

Context Processors:
Context processors are pre-rendered functions that can inject variables into the template context for all views. There are three context processors in this code:
status() provides the current status of the scanning procedure based on Redis database information. show_ frequency() provides the configuration for the scan frequency. show_vuln_count() returns the number of vulnerabilities that have been stored in the Redis database.

5.2 Detection of types of Vulnerabilities

The Manager.py code defines functions for a role-based system's rule management and loading. These functions search specific directories for Python modules (rules) and dynamically install them. This type of dynamic rule loading is prevalent in applications that support flexible rule systems, such as security tools and policy engines. Using mod.Rule(), a Rule class instance is created (assuming the class already exists within the module). The loaded rule instance is added to the loaded_rules dictionary using the module name as the key.
This function returns the loaded_rules dictionary containing instances of loaded rules. In the diagram above, it specifies different types of rules created based on four categories introduced in this application that are brute force (Credentials rule), Configuration (Misconfiguration rule), CVEs (NVD/NIST Vulnerabilities), Vulnerabilities (Known flaws), Services (Vulnerable Ports).

5.3 Proposed Scanner

Initiation: User provides scan configurations. register.py is responsible for initiating the scan by parsing and validating the configurations using the SchemaParser class in parser.py.
Configuration Parsing: parser.py parses and sets up the scan configurations.
Rule Management: manager.py dynamically loads the necessary rule modules based on the specified role.
Scanning: port_scanner.py identifies open ports on the target systems. Vulnerability detection rules are then applied based on the open ports.
Result Analysis: triage.py processes the raw scan results, identifying and classifying vulnerabilities.
Reporting: reports.py generates scan reports in various formats like CSV, HTML, and TXT.
Utilities: Throughout the process, several utility functions from utils.py are utilized. logging.py handles logging, ensuring that events, actions, and errors are recorded. Redis (managed by redis.py) is used for temporarily storing scan results, managing scan tasks, or maintaining session data.
Security: security.py manages aspects related to secure communication, data encryption, or user authentication.
Background Processes: workers.py manages resource-intensive tasks in the background.

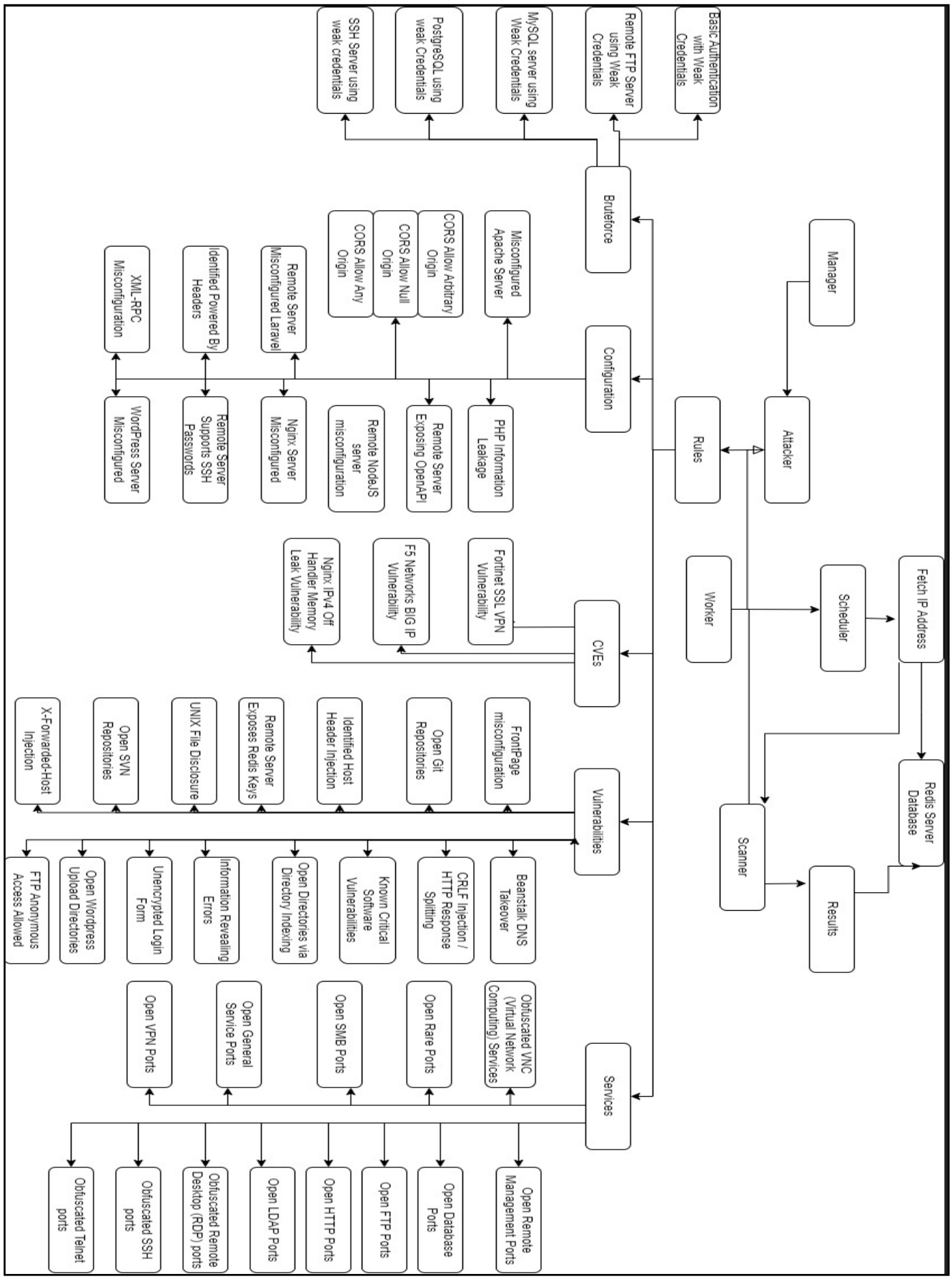


Fig.4 System Architecture of Proposed Scanner.

6 Evaluation

In this evaluation, an investigation of the domain-specific proficiency of a web application scanner in comparison to other Open-Source Web Vulnerability Scanners available online was carried out. From pinpointing vulnerabilities with accuracy to assessing its performance and reporting prowess, some diverse capabilities were observed. This evaluation sheds light on its potential strengths and areas for improvement, providing valuable insights for the development of secure web applications.

6.1 Experiment / Case Study 1

Nessus Vulnerability Scanner **vs** Flask Vulnerability Scanner:

GUI of both the tools are shown below along with the report generated from their scanning.

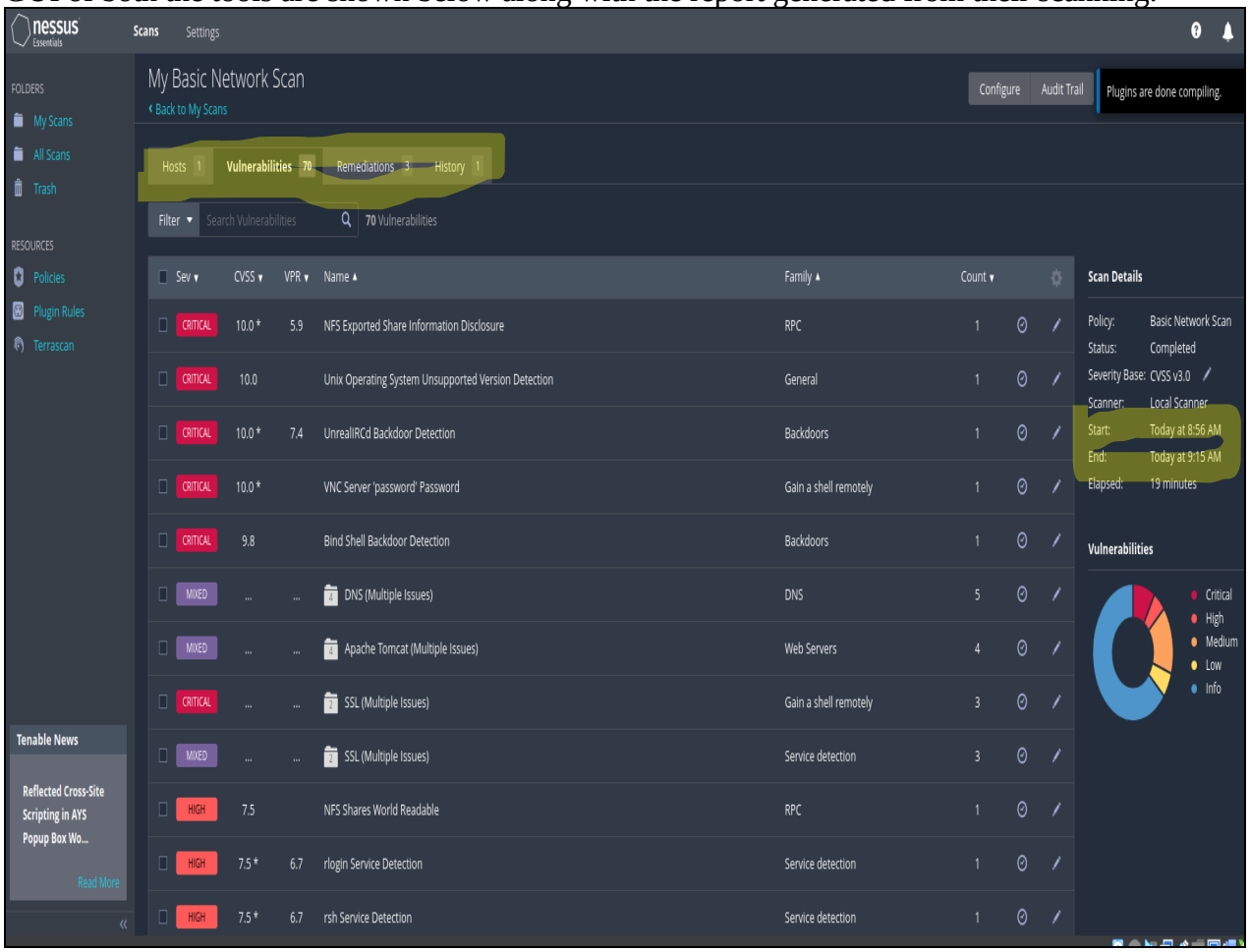


Fig.5 Nessus Scanning Dashboard Displaying Vulnerabilities & other details.

The time of scan initiated and ended has been highlighted in the image above. The vulnerabilities detected are different for both the tools since the plugins or the rules used for detection are detection but when comparing the functionalities, it states similar results by considering the factors like processing time, categorization of threats, and detection rate.

Overview

TIMESTAMP	2023-08-25 18:57:24	CRITICAL	2
ID	A7CF23EA	HIGH	10
NAME	Default	MEDIUM	7
ENGINEER	QuickScan Test	LOW	2
SOURCE IP	127.0.0.1	INFO	1

Vulnerabilities

TITLE	This rule checks for Open Directories via Directory Indexing
FINDINGS	Remote Server has Directory Indexing Enabled
ADDRESS	10.0.2.4
PORT	80
DETAILS	Identified an Open Directory at http://10.0.2.4/test/
RULE_ID	VLN_Z013
MITIGATION	Disable Directory Indexing on the server. Directory Indexing can allow access to files on the server to untrusted sources.
TITLE	This rule checks if FTP Server allows Anonymous Access
FINDINGS	FTP Anonymous Access Allowed
ADDRESS	10.0.2.4
PORT	21
DETAILS	FTP with Anonymous Access Enabled

Fig.6 Report of Metasploitable detected vulnerabilities using proposed scanner.

#	Target	Domain	Port	Severity	Result
1.	10.0.2.4	N/A	80	Critical	Remote Server has Directory Indexing Enabled
2.	10.0.2.4	N/A	21	Critical	FTP Anonymous Access Allowed
3.	10.0.2.4	N/A	3306	High	Remote Server Exposes Database Port(s)
4.	10.0.2.4	N/A	22	High	Remote Server Exposes Administration Port(s)
5.	10.0.2.4	N/A	139	High	Remote Server Exposes Administration Port(s)
6.	10.0.2.4	N/A	445	High	Remote Server Exposes SMB Port(s)
7.	10.0.2.4	N/A	80	High	Critical Vulnerabilities Found
8.	10.0.2.4	N/A	5432	High	Remote Server Exposes Database Port(s)
9.	10.0.2.4	N/A	111	High	Exposed Service Port
10.	10.0.2.4	N/A	21	High	Remote Server Exposes FTP Port(s)
11.	10.0.2.4	N/A	5900	High	Remote Server Exposes Administration Port(s)
12.	10.0.2.4	N/A	23	High	Remote Server Exposes Administration Port(s)
13.	10.0.2.4	N/A	514	Medium	Remote Server Exposes Rare Port(s)
14.	10.0.2.4	N/A	513	Medium	Remote Server Exposes Rare Port(s)
15.	10.0.2.4	N/A	6000	Medium	Remote Server Exposes Rare Port(s)
21.	10.0.2.4	N/A	80	Low	Remote Server Exposes HTTP Port(s)
22.	10.0.2.4	N/A	80	Informational	Identified Powered By Headers

Fig.7 Dashboard Display of Vulnerabilities Detected in Flask Vulnerability Scanner.

6.2 Experiment / Case Study 2

Skipfish Vulnerability Scanner **vs** Flask Vulnerability Scanner:

The comparison of both the tools were performed using a single known target machine as Metasploitable VM (Installed on Oracle Virtualbox, configured with NAT Network).

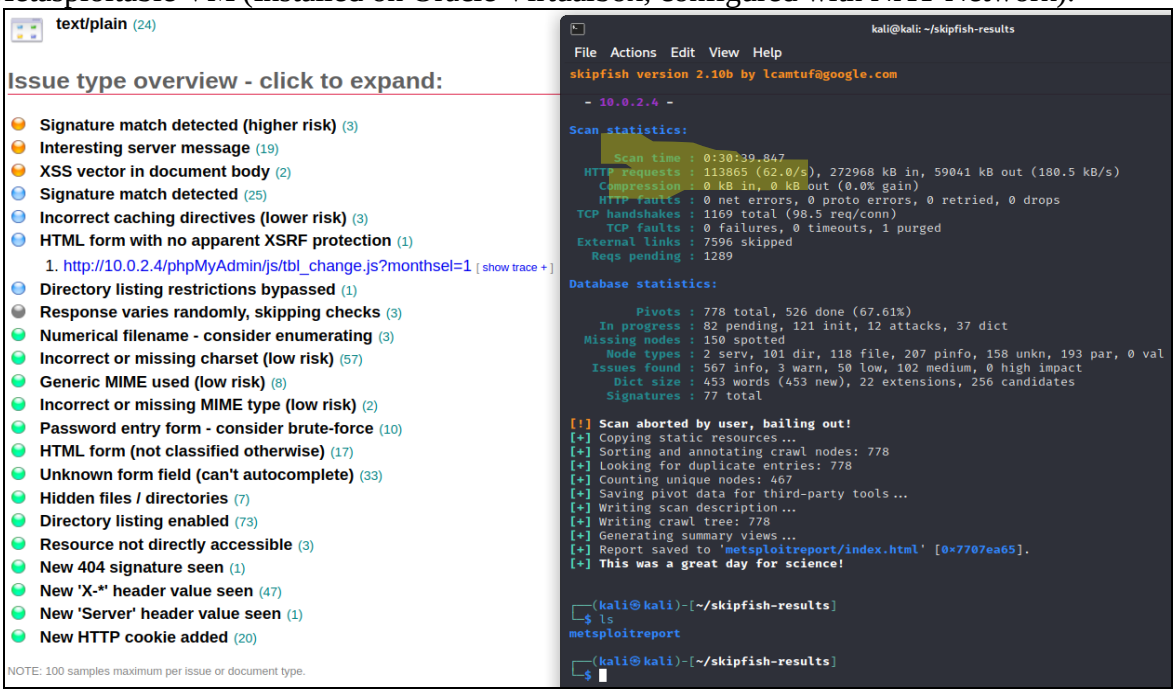


Fig.8 Skipfish Result, and report display along with other details highlighted for comparison.

6.3 Discussion

The proposed scanner boasts remarkable accuracy and offers the added advantage of minimizing scan time, rendering it exceptionally efficient for both bug bounty hunting and organizational maintenance endeavors. While there are numerous other tools available that excel in vulnerability scanning, what sets this scanner apart is its exclusive reliance on the Flask framework. This unique choice underscores a commitment to streamlined simplicity and user-friendliness.

In contrast, when assessing tools like Nessus and skipfish in the evaluation, it becomes apparent that although they exhibit an impressive scope of threat detection, they also come burdened with an extensive array of rules and plugins designed for this purpose. This complexity can be daunting for individuals at the outset of their web development journey, particularly those who lack extensive experience. In this regard, the Flask framework shines as an approachable option, characterized by its ease of comprehension and flexibility for users of varying skill levels. The user-friendly nature of Flask not only facilitates understanding but also encourages customization, allowing developers to tailor solutions that align precisely with their unique needs.

7 Conclusion and Future Work

In conclusion, the proposed scanner's focus on accuracy, efficiency, and its alignment with the Flask framework presents a compelling case for its integration into bug bounty

programs and organizational maintenance routines. This strategic choice not only simplifies vulnerability scanning but also aligns seamlessly with the needs of both novice and experienced developers, all while contributing to the creation of sophisticated and adaptable web applications.

7.1 Future Work

There are very few vulnerability scanners that are completely based on Flask framework making this proposed scanner contribute among them. There is still a lot more vulnerability list to be covered as rules can be created to handle more recent CVEs and Zero Day issues. A better way could be to link the program source code with the NIST/NVD database and automate this process for an AI generated report to notify the SOC team using their communication channels like Teams, Outlook, or Slack. There are other issues like private IP address and URLs that are not covered in this paper and can be utilized as a path to explore different research options.

References

Ablahd, D. (2023) 'Using Python to Detect Web application vulnerability By', 13, pp. 1045–1058.

Alazmi, S. and De Leon, D.C. (2022) 'A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners', *IEEE Access*, 10, pp. 33200–33219. Available at: https://doi.org/10.1109/ACCESS.2022.3161522.

Farrell, S. (2022) *Abstraction and automation of WordPress vulnerability scanning*. National College of Ireland. Available at: https://norma.ncirl.ie/id/eprint/6515.

Ghimire, D. (2020) 'Comparative study on Python web frameworks: Flask and Django'.

Goethem, T. van *et al.* (2014) 'Large-Scale Security Analysis of the Web: Challenges and Findings', in *Trust and Trustworthy Computing*. Available at: https://api.semanticscholar.org/CorpusID:14830723.

Jain, A. *et al.* (2023) 'Web Scanner: An Innovative Prototype for Checking Web Vulnerability', in R. Silhavy, P. Silhavy, and Z. Prokopova (eds) *Software Engineering Application in Systems Design*. Cham: Springer International Publishing (Lecture Notes in Networks and Systems), pp. 680–691. Available at: https://doi.org/10.1007/978-3-031-21435-6_58.

Kharat, P. and Chawan, P. (2022) 'Vulnerability Management System'.

Kumar, M. and Sharma, A. (2017) 'An integrated framework for software vulnerability detection, analysis and mitigation: an autonomic system', *Sādhanā*, 42(9), pp. 1481–1493. Available at: https://doi.org/10.1007/s12046-017-0696-7.

Lokhande, P. *et al.* (2015) 'Efficient way of web development using python and flask'.

Odion, T.O. *et al.* (2023) 'VulScan: A Web-Based Vulnerability Multi-Scanner for Web Application', in *2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG). 2023 International Conference on Science, Engineering and Business for Sustainable Development Goals (SEB-SDG)*, Omu-Aran, Nigeria: IEEE, pp. 1–7. Available at: https://doi.org/10.1109/SEB-SDG57117.2023.10124601.

Parimala, G., Sangeetha, M. and AndalPriyadharsini, R. (2018) 'Efficient Web Vulnerability Detection Tool for Sleeping Giant-Cross Site Request Forgery', *Journal of Physics: Conference Series*, 1000, p. 012125. Available at: https://doi.org/10.1088/1742-6596/1000/1/012125.

Park, Y. and Park, J. (2008) ‘Web Application Intrusion Detection System for Input Validation Attack’, in *2008 Third International Conference on Convergence and Hybrid Information Technology*. *2008 Third International Conference on Convergence and Hybrid Information Technology (ICCIT)*, Busan, Korea: IEEE, pp. 498–504. Available at: https://doi.org/10.1109/ICCIT.2008.338.

R, R. *et al.* (2023) ‘Web Application Security Testing Framework using Flask’, in *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India: IEEE, pp. 1646–1652. Available at: https://doi.org/10.1109/ICAAIC56838.2023.10140422.

Saabith, A., Fareez, M. and Vinothraj, T. (2019) ‘Python current trend applications-an overview’, *International Journal of Advance Engineering and Research Development*, 6(10). Available at: https://d1wqtxts1xzle7.cloudfront.net/61390005/IJAERDV06I108548120191201-27336-1cr1jto-libre.pdf?1575222047=&response-content-disposition=inline%3B+filename%3DPYTHON_CURRENT_TREND_APPLICATIONS_AN_OVE.pdf&Expires=1692664036&Signature=cnLHBL6vY0eM43jcx1~Wt10D-DUKOltALTWN0zt-OSfnvVY86W6CQ6djw0kaK-cbVFQt9SX3jYEeLl9mgeZmu9VnX~eeGN4G7BZp4qAB5twoGVAwK-ITuc3u196n1770r~HH2hXsp2osaU3D7WA4OC1cAI-N2hVzAisvm293xOv1GGurQqQxeZxr8gAxr6GVrlgNXl6TWAHnAFrcmH5dQ1BEUcIhLuM2vioKShGIlRILdMNKhRfh8XOr8jsxNX~3upgYlYT-G7Yr-QroiQfATKuWBT6qDrY8kFuPZaKBp6r9iKGig5Yf6Jj45BUx5V6FvU~WWEKS-TxwQYaAbMCA8g__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

Sagar, D. *et al.* (2018) ‘STUDYING OPEN-SOURCE VULNERABILITY SCANNERS FOR VULNERABILITIES IN WEB APPLICATIONS’, *IIOAB Journal*, 9, pp. 43–49.

Susanto, C.O.N., Rizko, K.N.F. and Purbohadi, D. (2020) ‘Security Assessment Using Nessus Tool to Determine Security Gaps on the Repository Web Application in Educational Institutions’, *Emerging Information Science and Technology*, 1(2). Available at: https://doi.org/10.18196/eist.128.

Tang, J. and Zhou, F. (2021) ‘Design and Implementation of High-performance Web Vulnerability Scanner Based on Python Intelligent Crawler’, in *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*. *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*, Kunming, China: IEEE, pp. 765–769. Available at: https://doi.org/10.1109/CISAI54367.2021.00155.

Tung, Y.-H. *et al.* (2013) ‘A cost-effective approach to evaluating security vulnerability scanner’, in, p. 3.

Verma, A. (2023) *Insecure Deserialization Detection in Python*. Master of Science. San Jose State University. Available at: https://doi.org/10.31979/etd.3yzt-6hxp.

Wang, B. *et al.* (2019) ‘Research on Web Application Security Vulnerability Scanning Technology’, in *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chengdu, China: IEEE, pp. 1524–1528. Available at: https://doi.org/10.1109/IAEAC47372.2019.8997964.

Xu, D. *et al.* (2022) ‘Web Vulnerability Detection Analyzer Based on Python’:, *International Journal of Digital Crime and Forensics*, 14(2), pp. 1–17. Available at: https://doi.org/10.4018/IJDCF.302875.

Zhang, J., Hu, H. and Huo, S. (2021) ‘A Browser-based Cross Site Request Forgery Detection Model’, *Journal of Physics: Conference Series*, 1738(1), p. 012073. Available at: https://doi.org/10.1088/1742-6596/1738/1/012073.

Zukran, B. and Siraj, M.M. (2021) ‘Performance Comparison on SQL Injection and XSS Detection using Open-Source Vulnerability Scanners’, in *2021 International Conference on Data Science and Its Applications (ICoDSA)*. *2021 International Conference on Data Science and Its Applications (ICoDSA)*, Bandung, Indonesia: IEEE, pp. 61–65. Available at: https://doi.org/10.1109/ICoDSA53588.2021.9617484.