# Software Proposal Document for Web Vulnerability Scanner

Ahmed Hossam, Youssef Sultan, Mostafa Saleh and Youssef Tamer.
Supervised by: Dr. Mohamed El Emam

April 8, 2024

Table 1: Document version history

| Version | Date | Reason for Change |
|---|---|---|
| 1.0 | 28-Dec-2023 | SRS First version's specifications are defined. |
| 1.1 | 30-Dec-2023 | Scope defined. |
| 1.3 | 05-JAN-2024 | Operational Scenarios added. |
| 1.5 | 05-JAN-2024 | Functions added. |
| 1.7 | 07-JAN-2024 | Data Design updated. |
| 1.8 | 09-JAN-2024 | Appendices updated. |

**GitHub:** https://github.com/AhmedHossam9/Web-Vulnerability-Scanner


Scan to access the project's repository on GitHub

# Contents

**Abstract**

Web applications play a crucial role in facilitating the exchange of both services and data. However, as individuals increasingly rely on these different web applications to handle their variety of tasks, the security risks associated to these applications continue to grow consistently. Considering the fact that web applications are inherently connected to the internet, protecting and ensuring one's private information is secure becomes a matter of utmost importance. Different kinds of cyberattacks pose a significant threat to both the integrity and confidentiality of any kind of information processed or stored. Our project is motivated by shortcomings observed in existing security tools. The lack of free open source scanners that detect RCE vulnerabilities. The existing scanners offer limited customization options, making them difficult to adapt to specific needs. Most of the free available web scanners are extremely limited and offer few features while keeping the important ones behind paywalls. Our project aims to create an enhanced version of web security scanners that's easily accessible and free of charge.

# 1 Introduction

## 1.1 Purpose of this document

The main purpose of this document is to highlight essential features, functionality and the constraints of our Vulnerability Scanner project. This document demonstrates the core fundamentals requisite in the system development process, which includes the functional and non-functional requirements It ensures that all project stakeholders, including developers, testers, and end users, have a clear grasp of the goals and parameters of the work by a comprehensive guide. As we also supply the upcoming project users with a full manual that contains a description regarding each stage with its output.

## 1.2 Scope of this document

This document discovers and illustrates the functional and non-functional conditions, system overview, system compass, database schema, the class illustration. The main compass is to show the rudiments of our design so that contributors who are formerly part or might involve could understand these ways we are furnishing for the web scanner. It describes clear figure of the design system.

## 1.3 Business Context

The Vulnerability Scanner project emerges as a crucial tool in the field of cybersecurity in an era dominated by evolving cyber threats. Companies that operate in the connected digital world are more vulnerable to attacks by hostile actors looking to take advantage of holes in systems and networks. This software system offers a sophisticated way to spot and fix possible vulnerabilities because it is thoughtfully made to maneuver through this terrain. The Vulnerability Scanner makes a substantial contribution to enterprises' proactive defense posture. Through comprehensive evaluation of networks, systems, and applications, it enables enterprises to remain ahead of possible cyber-attacks. This strengthens the digital infrastructure and complies with legal requirements and industry best practices.The project is essentially more than just a software tool; it represents

a strategic asset for companies negotiating the tricky terrain of cybersecurity. The Vulnerability Scanner plays a vital role in achieving the overall objective of protecting sensitive data and preserving the integrity of digital assets by strengthening resilience, adhering to compliance standards, and encouraging a proactive security approach.

# 2 Similar Systems

## 2.1 Academic

### 2.1.1 Evaluation of web vulnerability scanners

[1] In this paper, two open source vulnerability scanners for web applications are evaluated, OWASP Zed At-tack Proxy (OWASP ZAP) and Skipfish. The authors explain the importance of web application security and how these scanners can help. They also provide a method of evaluation and experimental results. The PDF file is structured into five sections: introduction, basic concepts of web vulnerability scanners and web application vulnerabilites, experimental environment, experimental results and analysis, and conclusion.

NUMBER OF VULNERABILITIES OF THE EVALUATED SCANNERS DETECTED IN WAVSEP

| | OWASP ZAP | Skipfish |
|---|---|---|
| High Vulnerabilites | 0 | 0 |
| Medium Vulnerabilities | 34 | 11 |
| Low Vulnerabilities | 2535 | 14 |
| Informational Vulnerabilities | 625 | 353 |
| SUMMARY | 3194 | 378 |

Figure 1: Results

### 2.1.2 Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark

[2] In this paper, Mburano et al. in this study assesses the effectiveness of open- source web vulnerability scanners by employing the OWASP standard. A comparison of two scanners, ZAP and Arachni, demonstrated that neither is constantly superior in detecting web vulnerabilities. In four vulnerability orders, the OWASPstandard proved to be more demanding than the WAVSEP standard. The study recommends exercising the OWASP standard as the primary evaluation tool for these four orders, with the WAVSEP standard acting as a reciprocal tool to enhance the evaluation process

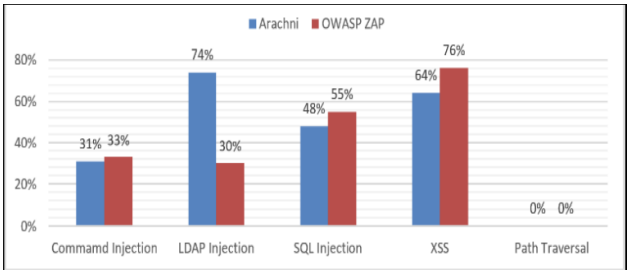



Figure 2: Graph

### 2.1.3 W-VST: A Testbed for Evaluating Web Vulnerability Scanner

[3] In this paper, Yuan-Hsin Tung et al. discusses the evaluation of web vulnerability scanners and proposes a cost-effective approach to evaluating them. The advanced confusion matrix is introduced as a tool to estimate the performance of web vulnerability scanners. The file also discusses related work and presents experimental results.

| No. | Vulnerability | Quantity | Percentage |
|---|---|---|---|
| 1 | Cross-Site Scripting:Reflected | 1389 | 43% |
| 2 | Cross-Site Scripting:Stored | 754 | 23% |
| 3 | SQL Injection | 354 | 11% |
| 4 | Privacy Violation | 273 | 8% |
| 5 | Path Manipulation | 182 | 6% |
| 6 | Unreleased Resource: Database | 97 | 3% |
| 7 | Null Dereference | 58 | 2% |
| 8 | Password Management: Hardcoded | 35 | 1% |
| 9 | Command Injection | 28 | 1% |
| 10 | HTTP Response Splitting | 21 | 1% |
| Summary | | 3226 | 99% |

Figure 3: Analysis

## 2.2 Business Applications

1. Tenable[4] :Tenable Vulnerability Management is an online vulnerability management platform that helps groups identify, choose to focus on, and address vulnerabilities through their entire IT system. It offers thorough asset visibility across cloud, containerized, and on-premises environments. In order to offer a consolidated perspective of an organization's total security posture, Tenable VM also integrates with a large number of other security tools.

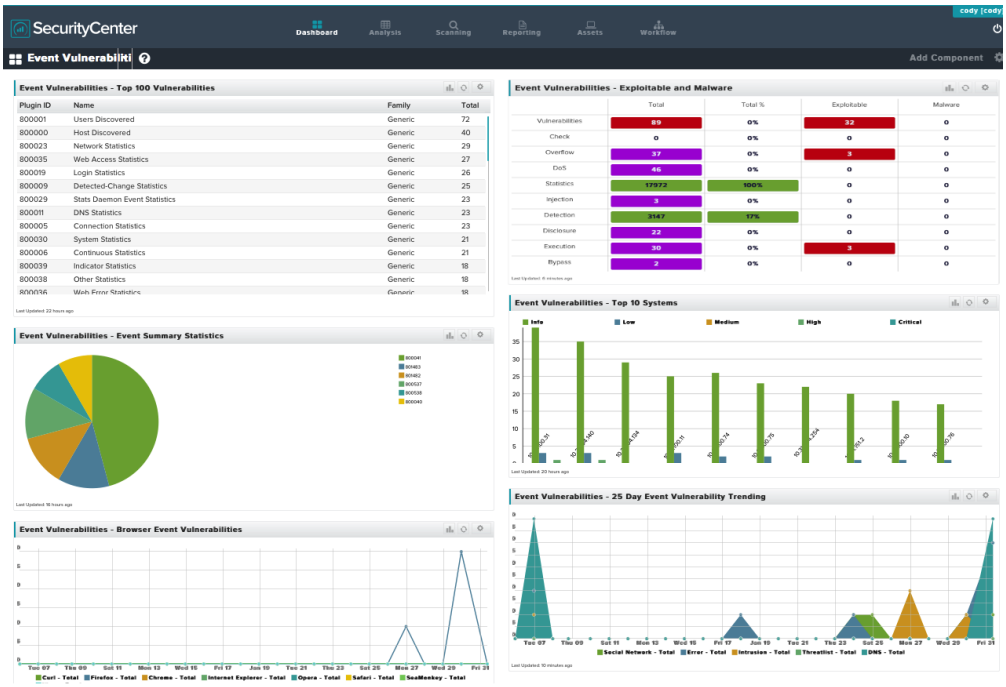



Figure 4: Tenable

2. Veracode[5] :Veracode is a cybersecurity company that expertizes in furnishing operation security results. Their platform is designed to make companies identify and address vulnerabilities in software operations. Veracode offers a comprehensive system for operation security by employing both static and dynamic analysis to assess law for implicit security excrescences. The platform's support for multiple programming languages and flawless integration into the software development life cycle enable early discovery and resolution of security issues. To help businesses cover their apps and forfend off cyberattacks, Veracode offers an automated security vulnerability discovery and form result that's both scalable and programmable.

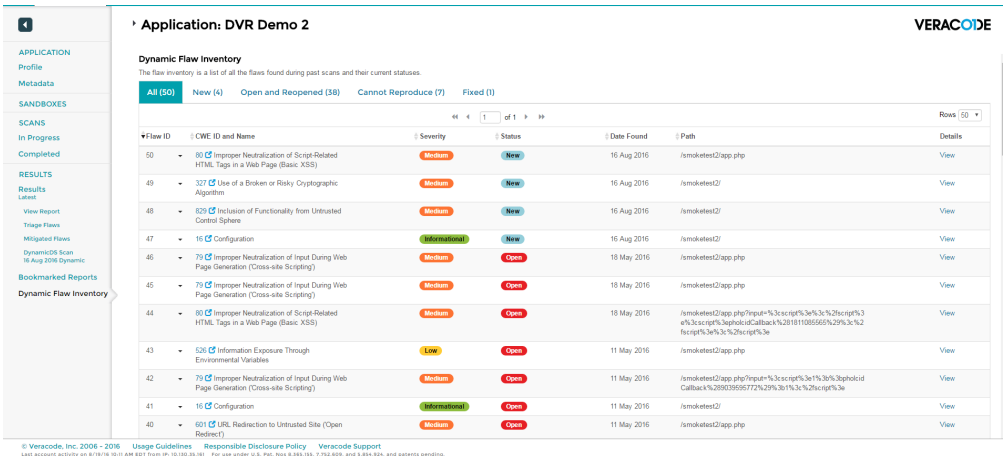



Figure 5: Veracode

# 3 System Description

## 3.1 Problem Statement

In today's modern digital world, web applications have been playing a vital role when it comes to facilitating the online exchange of both data and services. However, as individuals increasingly rely on these different web applications to handle their variety of tasks, the security risks asscoiated to these applications continue to grow consistently. Considering the fact that web applications are inherently connected to the internet, protecting and ensuring one's private information is secure becomes a matter of utmost importance. Different kinds of cyberattacks pose a significant threat to both the integrity and confidentiality of any kind of information processed or stored by the web applications.

Our inspiration for working on this project stemmed from a thorough examination of different existing security tools and scanners that were specifically designed for websites. Through this analysis, many major and common issues were found. Firstly, a large percentage of the existing scanners lack scanning for the RCE (Remote Code Execution) vulnerability. Secondly, it was found that these scanners offer limited customization options, making them difficult to adapt to specific needs. Finally, most of the free available web scanners are extremely limited and offer few features while keeping the important ones behind paywalls.

## 3.2 System Overview

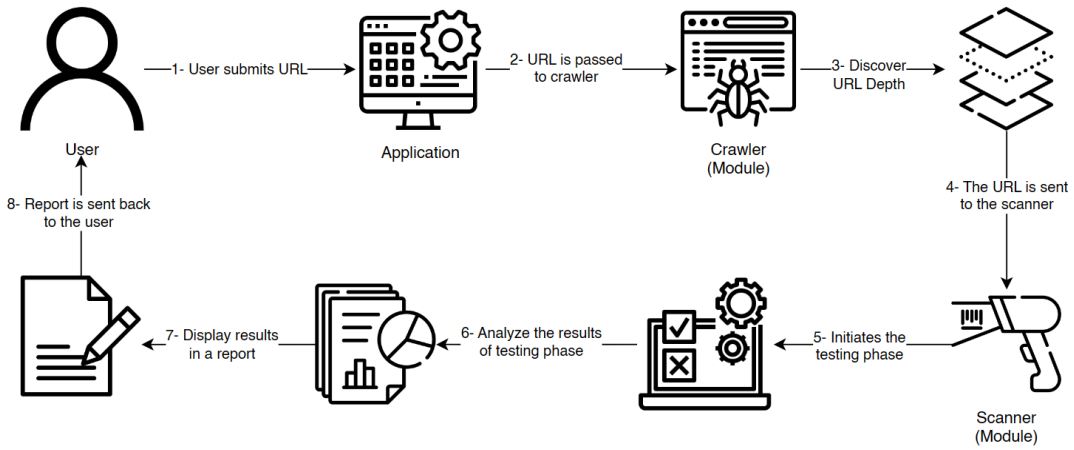



Figure 6: Project Overview

'Figure 1' shows the overview of the proposed project. It contains the following steps:

- User Input: submits a URL to the server. The server then checks if the URL is valid. If it is not, the user is asked to resubmit a valid URL. If the URL is valid, the server sends a score report back to the user.
- Crawling: The crawler discovers the depth of the URL submitted by the user.

- Scanner: The testing phase is initiated by the scanner to check if there are any detected vulnerabilities in the submitted URL.
  - The scanner includes the methods that are being used to during the testing phase , It generates various of JS Payloads that are being used to detect possible vulnerabilities, Using both POST and GET methods. As it parses the found forms in the site with those chosen methods and sends requests within using the generated pay-loads.
- Analyzing phase: The results of the testing phase is analyzed for the report that will be submitted to the user.
- Post-Analyzing phase: The score report is saved in a text file and then checked and sorted according to the discovered vulnerabilities before being sent out to the user.
- Methods:
  - Crawling and Mapping: Utilizes advanced crawling techniques to map the structure of web applications, ensuring comprehensive coverage during vulnerability assessments:
    - The Crawl function is a main of component of the crawler module is in charge of starting crawling the site in sequence to find links and examine the web app. As it triggers method getLinks to explore and get links from the base URL then it repeats the same process up to a defined depth.
  - Automated Scanning: Conducts automated scans of web applications to detect vulnerabilities such as SQL injection, cross-site scripting (XSS), Port scanning & RCE.
  - Real-time Reporting: Generates detailed and real-time reports highlighting identified vulnerabilities, their severity levels, and recommended remediation steps.
- Output:
  - The server gives the website a security score based on the vulnerabilities that it has found.

## 3.3 System Scope

The scope of the project focuses on the development and implementation of an advanced web security tool. This tool is designed specifically to enhance the overall security of web applications. The project addresses common issues found in existing security scanners, with a specific focus on addressing the following challenges:

**Port Scanning Feature:**

- It is a fundamental cybersecurity practice that maps network structures, identifies potential vulnerabilities, and aids in intrusion detection.
- It enhances security by evaluating firewall configurations, troubleshooting network issues, and providing a comprehensive asset inventory, contributing to a proactive and robust network defense strategy.

**RCE Scanning:**

- It enhances cybersecurity by proactively identifying and mitigating vulnerabilities that allow attackers to execute arbitrary code, preventing unauthorized access and potential malicious activities.
- It improves incident response, ensures regulatory compliance, and contributes to continuous monitoring and improvement of system security.

**Advanced Security Features:**

- Integrate automated scanning features to be able to detect and identify vulnerabilities.
- Use advanced crawling techniques that provide comprehensive coverage during vulnerability assessments.

**User-Friendly Interface:**

- Create a user-friendly interface to accommodate a diverse user base.
- Ensure there's accessibility for both security professionals and users with varying levels of expertise.

**Real-time Reporting:**

- Implement a real-time reporting feature that generates detailed reports without delay.
- Highlight all of the detected vulnerabilities and mention their severity levels then finally recommend actions for users to take.

**Comprehensive Database:**

- We aim to establish and maintain a database of vulnerabilities to serve as a foundation to help with accurate threat identification.
- It will be guaranteed that the database is expansive and will cover a wide range of potential security threats.

In summary, our project's scope is multi-faceted. We are aiming to create a powerful, user-friendly, advanced, and adaptive web vulnerability scanner that effectively enhances overall web application security.

## 3.4 System Context

As shown in figure 7, the user submits a link to the website he wishes to scan for vulnerabilities. The user will be given a choice to do a full vulnerability scan or a customized one for specific threats. A detailed report about the found vulnerabilities and their threat level will be generated after the scan is finished and later sent to the user to review.

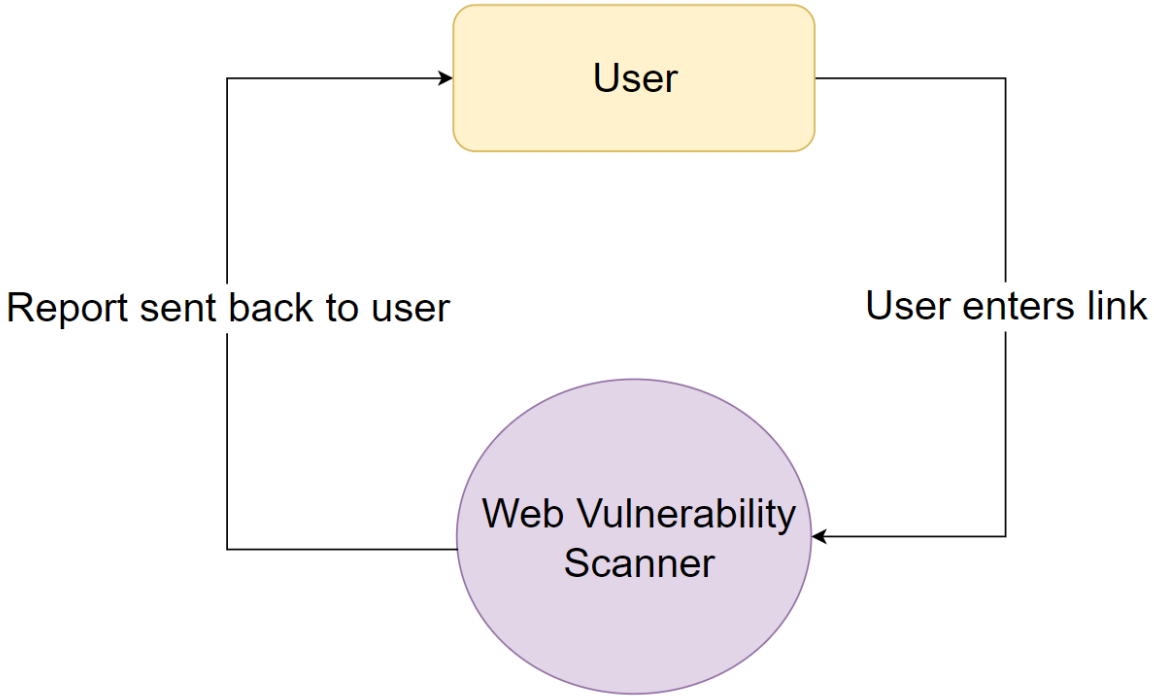



Figure 7: System context

## 3.5 Objectives

- Develop an advanced web vulnerability scanner to effectively identify and address website vulnerabilities, ensuring heightened security for businesses and individuals in safeguarding sensitive information.
- Create an intuitive and user-friendly interface, simplifying the configuration and initiation of scans for seamless user experience.
- Utilize datasets from reputable sources, including Kaggle, NIST, and NVD, alongside custom-created datasets, to enhance the accuracy and comprehensiveness of the web vulnerability scanner.

- Release the web vulnerability scanner as an open-source project, fostering community collaboration, usage, and development.
- Implement automated scanning features that efficiently detect and identify a broad spectrum of vulnerabilities, including SQL injection, cross-site scripting (XSS), port scanning, and remote code execution (RCE), contributing to a proactive defense strategy.
- Establish and maintain a comprehensive database of vulnerabilities, covering a wide range of potential security threats. This foundation will enhance accurate threat identification and support ongoing improvements in system security.
- Integrate a real-time reporting feature into the web vulnerability scanner, generating detailed reports promptly. These reports will highlight detected vulnerabilities, their severity levels, and provide actionable recommendations for users to address identified security issues effectively.

## 3.6 User Characteristics

- The web security tool is designed for a diverse user base that includes cyber-security experts, system administrators, developers, and users with limited cyber-security knowledge. It's user-friendly for those unfamiliar with advanced features, and compliance officers can use it to ensure adherence to regulations. Auditors can rely on the scanner's detailed reports for risk analysis. The design aims to cater to users with different levels of expertise and responsibilities, ensuring effective use across various roles within an organization.

# 4 Functional Requirements

## 4.1 System Functions

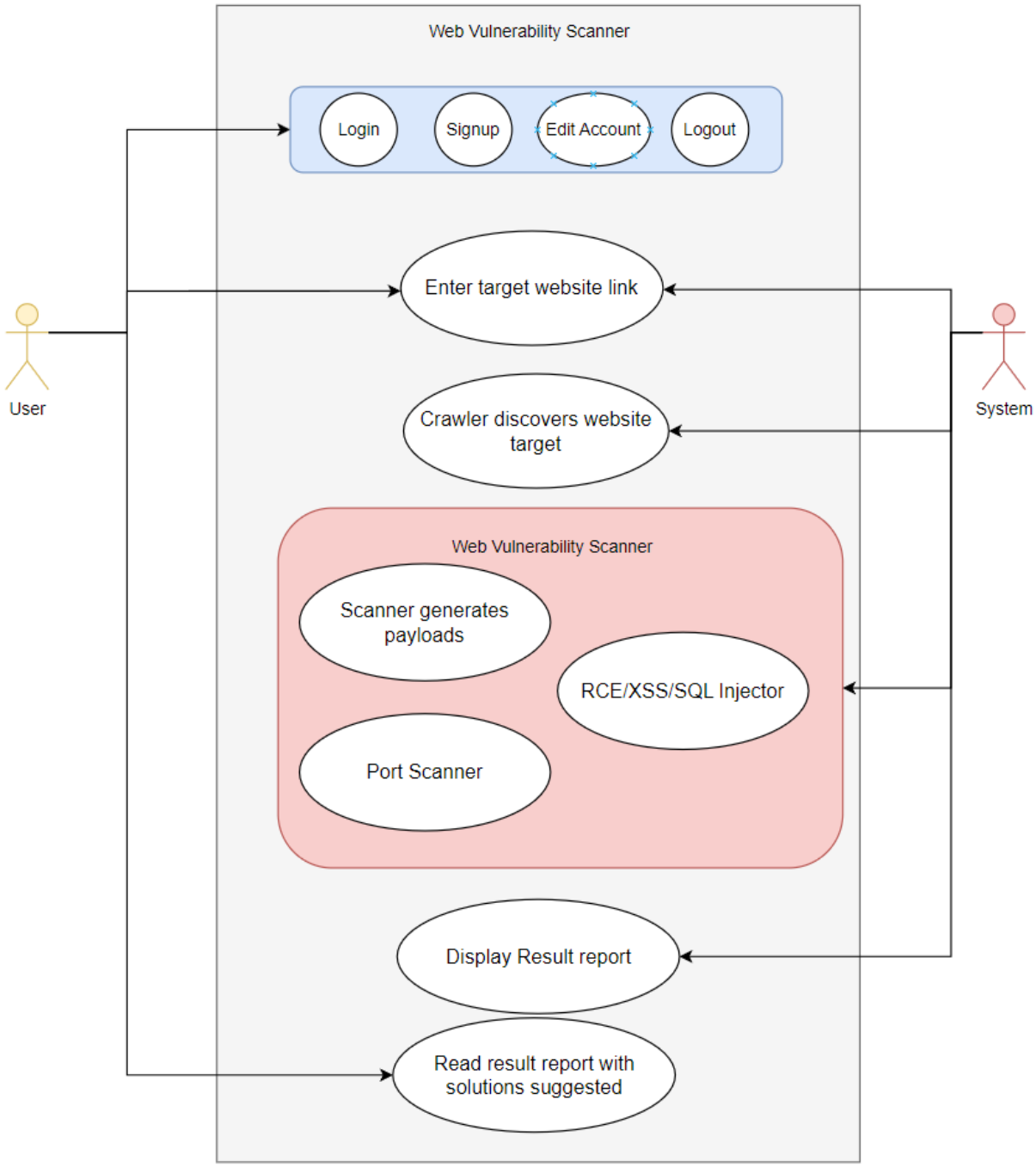



Figure 8: Use Case Diagram

- **ID01** The user shall be able to sign up to create a new account.

- **ID02** The user shall be able to login to their account.
- **ID03** The user shall be able to edit their account.
- **ID04** The user shall be able to log out of their account.
- **ID05** The user shall be able to enter the targeted website's link.
- **ID06** The system shall be able to perform a crawling phase to discover the URL's base and depth of the target.
- **ID07** The system shall be able to generate various types of payloads for the desired vulnerabilities.
- **ID08** The system shall be able to initiate the testing (injection) phase on to detect vulnerabilities.
- **ID09** The system shall be able to perform port scanning on the targeted website.
- **ID10** The system shall be able to generate and display a detailed report of scan results.
- **ID11** The user shall be able to see the detailed report with threat levels and suggested solutions.

## 4.2 Detailed Functional Specification

Table 2: User Input

| Name | User input |
|---|---|
| Code | ID05 |
| Priority | Extreme |
| Critical | 10/10 |
| Description | The user shall be able to enter the targeted website's link. |
| Input | The user must enter a website's link. |
| Output | Either website is valid or not. |
| Pre-condition | Client must login/sign-up to Web Scanner portal. |
| Post-condition | The submitted website is stored in the system's database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

Table 3: Crawling

| Name | Crawling |
|---|---|
| Code | ID06 |
| Priority | High |
| Critical | 8/10 |
| Description | The system shall be able to perform a crawling phase to discover the URL's base and depth of the target. It spawns multi-processing processes to analyze each link. |
| Input | The user's input which is a targeted website. |
| Output | The URL's base, depth, proxy, headers, and cookies. Links from web pages are discovered, and multi-processing processes are spawned to analyze each link using a core. |
| Pre-condition | Client should be logged in with an existing account. |
| Post-condition | Discovered base and links are stored in the database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

Table 4: Payloads generation

| Name | Payloads generation |
|---|---|
| Code | ID07 |
| Priority | Extreme |
| Critical | 10/10 |
| Description | The system shall be able to generate various types of payloads for the desired vulner- abilities. |
| Input | The chosen vulnerabilities by the user. |
| Output | Payloads generated upon the input. |
| Pre-condition | Client should be logged in with an existing account. |
| Post-condition | Payloads are passed to the system's functions and stored in the database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

Table 5: Test Phase and injecting

| | |
|---|---|
| Name | Test Phase and injecting |
| Code | ID08 |
| Priority | Extreme |
| Critical | 10/10 |
| Description | The system shall be able to initiate the testing (injection) phase on to detect vulnera- bilities. |
| Input | Generated payloads are inherited within the parsed fields. |
| Output | The scanner lists a vulnerable point through GET, POST, and form methods. |
| Pre-condition | Client should be logged in with an existing account. |
| Post-condition | Results are being logged with a threat-level and stored to database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

Table 6: Port Scanning

| | |
|---|---|
| Name | Port Scanning |
| Code | ID09 |
| Priority | Extreme |
| Critical | 10/10 |
| Description | The system shall be able to perform port scanning on the targeted website. |
| Input | Targeted website's link is entered by the user. |
| Output | Opened ports within the targeted website. |
| Pre-condition | Client should be logged in with an existing account. |
| Post-condition | Targt and opened ports are being logged and stored in the database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

Table 7: Result Report

| | |
|---|---|
| Name | Result report |
| Code | ID11 |
| Priority | Extreme |
| Critical | 7/10 |
| Description | The user shall be able to see the detailed report with threat levels and suggested solu- tions. |
| Input | None |
| Output | Detailed scan result report with suggested solutions. |
| Pre-condition | Client should be logged in with an existing account. |
| Post-condition | Report result are being logged and stored in the database. |
| Dependency | Client must input a website/target and have an account on the scanner's portal. |
| Risk | None |

# 5 Design Constraints

## 5.1 Standards Compliance

- **Security Standards:** The tool should adhere to industry-standard security practices, ensuring that it follows protocols and guidelines for secure coding, data protection, and secure communication.

- **Web Application Security Standards:** Compliance with established web application security standards, such as OWASP (Open Web Application Security Project), can be crucial. Adhering to OWASP guidelines helps in addressing common vulnerabilities and enhancing the overall security posture.

- **Crawling and Scanning Standards:** Crawling and Scanning Standards: The automated scanning and crawling methods should adhere to recognized standards for vulnerability assessment to ensure the reliability and accuracy of the results. This might include following guidelines from security organizations and standards bodies.

- **User Interface Accessibility Standards:** The user-friendly interface should comply with accessibility standards, ensuring that it is accessible to users with varying levels of expertise. This might involve adhering to accessibility guidelines like the Web Content Accessibility Guidelines (WCAG).

- **Data Protection and Privacy Standards:** Given the sensitivity of the data involved in security scanning, compliance with data protection and privacy standards, such as GDPR

(General Data Protection Regulation), is essential. This includes ensuring secure storage, processing, and transmission of user data.

- **Reporting Standards:** Real-time reporting should meet certain standards for clarity and comprehensiveness. Adhering to established reporting standards ensures that the information presented to users is structured, informative, and actionable.

- **Testing Standards:** Rigorous testing should align with industry testing standards, ensuring comprehensive coverage across diverse web environments. This might involve following testing frameworks and methodologies.

## 5.2 Hardware Limitations

- **Processing Power:** Automated scanning and crawling processes, especially in real-time, can be computationally intensive. The tool's efficiency may be constrained by the processing power of the server or hardware it runs on.

- **Memory (RAM):** Large-scale crawling and scanning activities may require significant memory resources to store and process data efficiently. Inadequate RAM could impact the tool's performance and responsiveness.

- **Storage Capacity:** Storing and managing a comprehensive vulnerability database, along with reports, may demand substantial storage capacity. The tool's effectiveness could be limited by insufficient storage space.

- **Network Bandwidth:** Real-time reporting and communication with users depend on network bandwidth. Low bandwidth may lead to delays in generating and transmitting reports, affecting the tool's responsiveness.

- **Concurrency and Parallelism:** The ability to handle multiple concurrent scans and user requests may be constrained by the hardware's capacity for parallel processing. Inadequate support for concurrency could impact the tool's scalability.

- **Graphics Processing Unit (GPU):** Certain security algorithms, especially those involving complex data processing, may benefit from GPU acceleration. However, hardware limitations in GPU availability or compatibility could impact the tool's performance.

- **Load Balancing:** Distributing incoming requests efficiently across multiple servers may be constrained by hardware limitations. Load balancing mechanisms help address this, but hardware capabilities play a role in the overall effectiveness.

- **Redundancy and Failover:** Ensuring high availability and minimizing downtime may require redundancy and failover mechanisms. Hardware limitations could affect the tool's ability to implement these features effectively.

## 5.3 Other Constraints as appropriate

- **Time Constraints:** Project timelines may impose constraints on development, testing, and deployment. Balancing a timely release with thorough testing is crucial.

- **Budgetary Constraints:** The availability of financial resources may impact the choice of technologies, the size of the development team, and the overall scope of the project.

- **Legal and Regulatory Constraints:** Compliance with legal frameworks and industry regulations, such as data protection laws and privacy regulations, may impose constraints on how the tool processes and handles user data.

- **User Acceptance Constraints:** User acceptance testing and feedback may reveal constraints related to the tool's usability and whether it effectively meets user expectations and requirements.

- **Scalability Constraints:** Anticipating the tool's growth and ensuring it can scale to accommodate increased user loads and data volumes may impose constraints on the chosen architecture and technologies.

- **Ethical Constraints:** Ethical considerations, such as avoiding misuse of the tool for unauthorized activities, must be addressed. Implementing ethical practices and preventing unintended consequences are constraints that should be taken into account.

- **Documentation Constraints:** Comprehensive documentation is essential for users and developers. Constraints related to the thoroughness and clarity of documentation can impact the tool's usability and maintainability.

- **User Training Constraints:** Ensuring that users, including security professionals and those with varying levels of expertise, can effectively use the tool may pose challenges related to training and educational materials.

# 6 Non-functional Requirements

- **Security:** The system should protect users by encrypting their password. Also, it should make sure of confidentiality and integrity of critical data such as scan results and user information.

- **Reliability:** The system should have a great level of availability so that it would be accessible to users at any time and should operate even in the presence of errors.

- **Performance:** The system should be fast in responding to user requests especially in scanning and giving a detailed report which will contain each vulnerability and its severity level.

- **Portability:** Since the system is web based, it should be responsive so that it would be used on different devices such as mobile, PC or tablet.

- **Maintainability:** The system should be flexible , so that it would be easy to be modified on or updated to.

- **Usability:** The system should have a user-friendly interface that would be easy to use.

# 7 Data Design

- **Dataset:** The Common Vulnerabilities and Exposures (CVE) is a comprehensive repository managed by the MITRE Corporation, serving as a standardized catalog for identifying and naming vulnerabilities in software and hardware systems. Each vulnerability is assigned a unique identifier, known as a CVE ID, facilitating tracking and referencing across diverse security tools and databases. The dataset provides structured information, including detailed descriptions, affected software or hardware, severity levels, and potential solutions. Adhering to a common format ensures consistency in CVE entries, employing standardized terms and definitions for improved communication within the cybersecurity community. This collaborative effort involves contributions from security researchers, vendors, and organizations, ensuring the database remains current with the latest vulnerabilities. Moreover, the integration of CVE with various security tools enables automated tracking and identification of vulnerabilities in different systems. Temporal information, such as the discovery and disclosure dates, adds a crucial dimension to understanding the lifecycle of vulnerabilities within the CVE dataset.

| Vulnerability Name | Vulnerability Description | Severity Level | Mechanism |
|---|---|---|---|
| SQL Injection | Allows attackers to execute malicious SQL queries, accessing, modifying, or deleting data in a database. | High | Exploits improper input handling, enabling injection of SQL commands through forms, URLs, or other inputs. |
| Cross-Site Scripting (XSS) | Enables attackers to inject malicious scripts into web pages, compromising user data or hijacking sessions. | Medium | Occurs due to inadequate input validation, allowing insertion of scripts into web pages viewed by other users. |
| Remote Code Execution (RCE) | Allows attackers to execute arbitrary code on a target system, potentially leading to complete compromise. | Critical | Typically results from insecure deserialization, input validation issues, or vulnerabilities in APIs. |
| Server-Side Request Forgery (SSRF) | Permits attackers to make requests from the server to other resources, potentially accessing internal systems. | High | Exploits server-side functionality to access resources, often through improper handling of user-supplied URLs. |
| Cross-Site Request Forgery (CSRF) | Forces users to execute unwanted actions on a web application in which they're authenticated. | Medium | Exploits the trust a site has in a user's browser by executing unauthorized actions via forged requests. |
| Insecure Deserialization | Allows attackers to manipulate serialized objects to execute arbitrary code, leading to various attacks. | High | Occurs when untrusted data is deserialized without proper validation, enabling code execution. |
| Information Disclosure | Leaks sensitive information like system details, configurations, or user data, aiding further attacks. | Low | Arises due to improper security configurations or errors in applications leading to data exposure. |

Figure 9: Database containing popular vulnerabilities inherited from CVE dataset

- **Database:** The system employs a versatile database structure, featuring a Database class with key-value storage. It securely manages user credentials, vulnerability data, and scan results. The database ensures efficient data retrieval and storage, supporting crucial functionalities such as user authentication, vulnerability tracking, and historical scan record storage.

# 8 Preliminary Object-Oriented Domain Analysis


# 9 Operational Scenarios

- Primary Actor: User
- Preconditions
  - The user has access to the web security tool with his registered account.
  - The user has a valid URL that they want to assess for potential vulnerabilities.
- **Scenario 1: Full Vulnerability scan**
- User Submits URL
  - The user logs into the web security tool and navigates to the scan section.

  - The user inputs the target URL of their web application for comprehensive vulnerability assessment.

- URL Validation
  - The system validates the submitted URL to ensure it is in the correct format and is accessible.

- Crawling and Mapping
  - The system initiates a crawling process to discover the depth and structure of the submitted URL.
  - Advanced crawling techniques are employed to map the entire web application, ensuring comprehensive coverage.

- Automated Scanning
  - Multiprocessing processes are spawned to analyze each discovered link systematically.
  - The automated scanning feature is activated, conducting scans for vulnerabilities such as SQL injection, cross-site scripting (XSS), port scanning, and Remote Code Execution (RCE).
  - Payloads are generated and injected into the website to identify potential vulnerabilities.

- Results Analysis
  - The system analyzes the results of the automated scanning, compiling a detailed report on the detected vulnerabilities, their types, and severity levels.

- Real-time Reporting
  - The web security tool generates a real-time, detailed report for the user, highlighting all identified vulnerabilities and their severity levels.
  - The report includes suggested security measures to mitigate the detected vulnerabilities.

- Report Storage
  - The generated report is saved in a database for future reference, creating a historical record of the full vulnerability scan results.

- User Notification
  - The user is promptly notified that the full vulnerability scan is complete and that the detailed report is available for review.

- **Scenario 2: Customized Scan for Specific Threats**

- User Submits URL
  - The user logs into the web security tool and navigates to the scan section.
  - The user inputs the target URL of their web application for comprehensive vulnerability assessment.
- URL Validation
  - The system validates the submitted URL to ensure it is in the correct format and is accessible.
- Threat Selection
  - The user specifies the specific threats they want to focus on during the scan.
  - Available threat options may include SQL injection, cross-site scripting (XSS), or other predefined vulnerabilities.
- Initiation of Customized Scan
  - Upon configuration completion, the user initiates the customized scan process.
  - The system activates the scanning algorithms tailored to the selected threats.
- Real-time Progress Monitoring
  - The system analyzes the scan results, focusing specifically on the selected threats.
  - Severity levels of detected vulnerabilities are assessed based on predefined criteria.
- Results Analysis
  - The system analyzes the results of the automated scanning, compiling a detailed report on the detected vulnerabilities, their types, and severity levels.
- Real-time Reporting
  - The web security tool generates a real-time, detailed report for the user, highlighting all identified vulnerabilities and their severity levels.
  - The report includes suggested security measures to mitigate the detected vulnerabilities.
- Report Storage
  - The generated report is saved in a database for future reference, creating a historical record of the full vulnerability scan results.
- User Notification
  - The user is promptly notified that the full vulnerability scan is complete and that the detailed report is available for review.

# 10 Project Plan

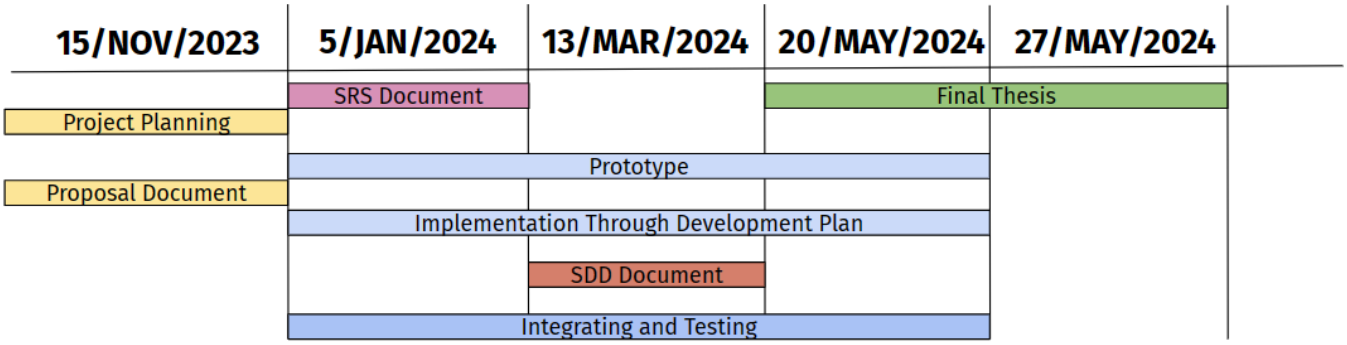



Figure 10: Time Plan

# 11 Appendices

## 11.1 Definitions, Acronyms, Abbreviations

- NIST: National Institute of Standards and Technology
- RCE: Remote code execution.
- XSS: Cross-site scripting.
- SQL: Structured query language.
- NVD: National Vulnerability Database.
- CVE: Common Vulnerabilities and Exposures.
  - Definitions:
    - SQL: structured programming language that is used to handle relational databases and execute various functionalities on the data
    - RCE: unauthorized intruder runs a script or code on victim's system remotely from other location, This kind of attack occurs when the attacker can injects or exploit code exists and accepted by the system to be triggered as legitimate (script/command).

## 11.2 Supportive Documents

- Dataset.
  The Common Vulnerabilities and Exposures (CVE) is a comprehensive catalog of known vulnerabilities and exposures in software and hardware systems. It is maintained by the

MITRE Corporation. The primary purpose of CVE is to provide a standardized way of identifying and naming vulnerabilities.

Key points about the CVE dataset:

- Identification and Naming: Each vulnerability in the CVE database is assigned a unique identifier, commonly referred to as a CVE ID. This ID is used to track and reference specific vulnerabilities across various security tools and databases.
- Structured Information: The dataset includes detailed information about each vulnerability, such as a description, the affected software or hardware, severity level, and possible solutions or mitigations.
- Common Format: CVE entries adhere to a common format to ensure consistency. This includes standardized terms and definitions to facilitate better understanding and communication within the cybersecurity community.
- Community Collaboration: CVE is a collaborative effort that involves contributions from security researchers, vendors, and organizations. This community-driven approach helps keep the database up-to-date with the latest vulnerabilities.
- Integration with Security Tools: Many security tools and platforms integrate with the CVE database, enabling automated tracking and identification of vulnerabilities in various systems.
- Temporal Information: CVE entries may include temporal information, such as the date the vulnerability was discovered, disclosed, and the date when the entry was last updated. This temporal data is crucial for understanding the lifecycle of vulnerabilities.

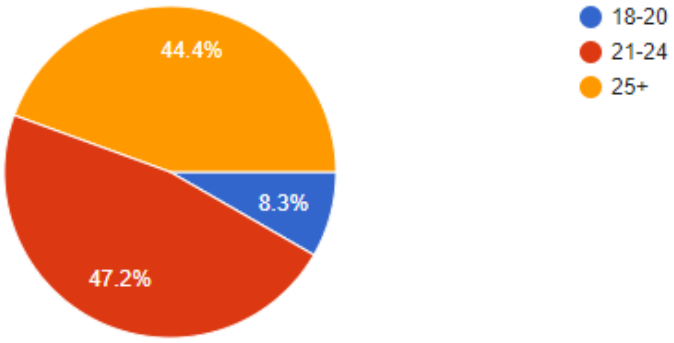
| | | | Threat Level | | | |
|---|---|---|---|---|---|---|
| CVE-2005-2352 | Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') | NONE | MEDIUM | NETWORK | PARTIAL | PARTIAL |
| CVE-2019-8143 | Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') | SINGLE | LOW | NETWORK | NONE | PARTIAL |
| CVE-2019-8142 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8147 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8146 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8145 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8157 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8128 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2019-8232 | Improper Input Validation | SINGLE | MEDIUM | NETWORK | PARTIAL | PARTIAL |
| CVE-2018-5735 | Reachable Assertion | NONE | LOW | NETWORK | PARTIAL | NONE |
| CVE-2010-3674 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | NONE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2018-1074 | Insufficiently Protected Credentials | SINGLE | LOW | NETWORK | NONE | PARTIAL |
| CVE-2018-1062 | Information Exposure | SINGLE | MEDIUM | NETWORK | NONE | PARTIAL |
| CVE-2018-1000095 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | SINGLE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2017-7510 | Information Exposure | SINGLE | LOW | NETWORK | NONE | PARTIAL |
| CVE-2014-7851 | Permissions Privileges and Access Controls | SINGLE | MEDIUM | NETWORK | PARTIAL | PARTIAL |
| CVE-2014-0153 | Information Exposure | NONE | MEDIUM | NETWORK | NONE | PARTIAL |
| CVE-2014-0151 | Cross-Site Request Forgery (CSRF) | NONE | MEDIUM | NETWORK | PARTIAL | PARTIAL |
| CVE-2016-3113 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | NONE | MEDIUM | NETWORK | NONE | NONE |
| CVE-2016-3077 | Improper Restriction of Operations within the Bounds of a Memory Buffer | SINGLE | LOW | NETWORK | PARTIAL | NONE |
| CVE-2019-18659 | Use of a Broken or Risky Cryptographic Algorithm | NONE | LOW | NETWORK | NONE | NONE |

Figure 11: Dataset Sample

- Survey

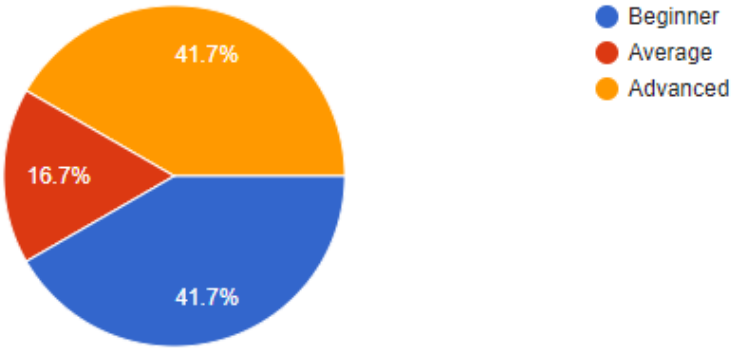
Age

36 responses



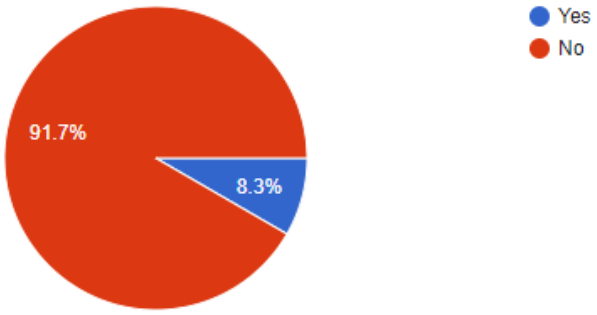

How would you describe your level of expertise in web security?

36 responses



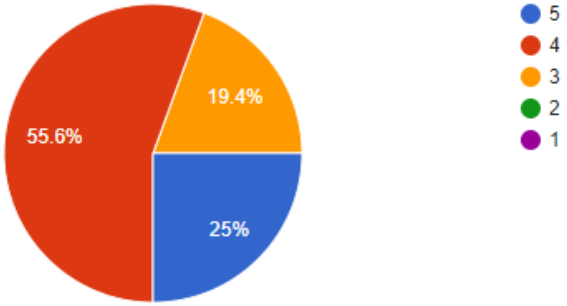
Do you currently use any web vulnerability scanning tools?

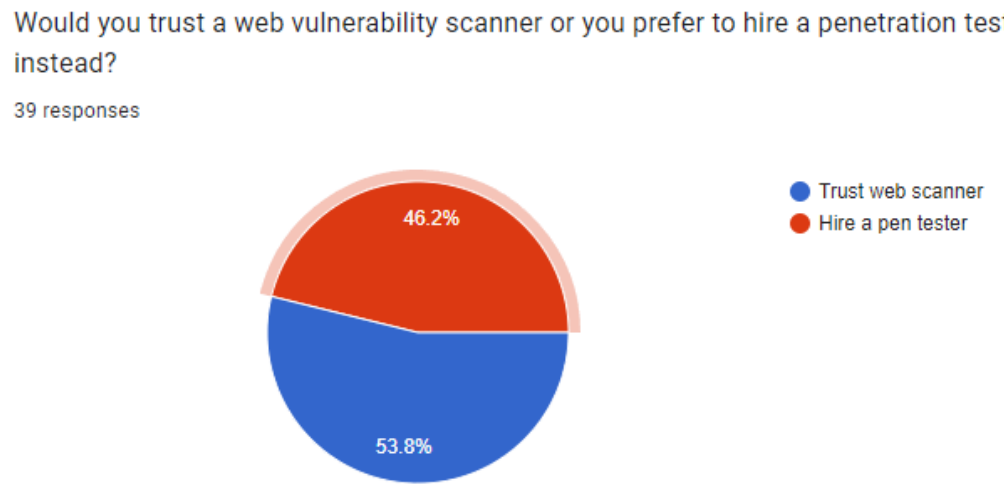
36 responses




Rate the idea of web vulnerability scanner [1=bad , 5=good]

36 responses






# References

[1] Yuma Makino and Vitaly Klyuev. "Evaluation of web vulnerability scanners". In: *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Vol. 1. 2015, pp. 399–402. DOI: 10.1109/IDAACS.2015.7340766.

[2] Balume Mburano and Weisheng Si. "Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark". In: *2018 26th International Conference on Systems Engineering (ICSEng)*. 2018, pp. 1–6. DOI: 10.1109/ICSENG.2018.8638176.

[3] Yuan-Hsin Tung, Shian-Shyong Tseng, Jen-Feng Shih, et al. "W-VST: A Testbed for Evaluating Web Vulnerability Scanner". In: *2014 14th International Conference on Quality Software*. 2014, pp. 228–233. DOI: 10.1109/QSIC.2014.50.

[4] *tenable*. (Accessed: 11/15/2023). URL: https://www.tenable.com/products/tenable-io/.

[5] *veracode*. (Accessed: 11/15/2023). URL: https://www.veracode.com.