CHRIST
COLLEGE OF ENGINEERING
IRINJALAKUDA
Affiliated to KTU | Approved by AICTE | Managed by CMI Fathers | NBA Accredited

Software Requirements Specifications

Reconify - Web Security Assessment Platform

BESTIN BABU	CCE21CS030
JERRY JOHN	CCE21CS043
RUSHALIN RENNY	CCE21CS057

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

2021 - 2025

CHRIST
COLLEGE OF ENGINEERING
IRINJALAKUDA
Affiliated to KTU | Approved by AICTE | Managed by CMI Fathers | NBA Accredited

C E R T I F I C A T E

This is to certify that the Software Requirements Specification for

Reconify - Web Security Assessment Platform

is a bonafide SRS document of the CSD 415 Project Phase I work done by
BESTIN BABU (CCE21CS030)
JERRY JOHN (CCE21CS043)
RUSHALIN RENNY (CCE21CS057)
under my supervision and guidance, in partial fulfillment of the requirements for the award of the Bachelor Degree of Engineering in the Branch of Computer Science and Engineering from APJ Abdul Kalam Technological University

Guide : **Ms. Minnu Mootheden**

Head of the Department: **Dr. Vince Paul**

External Examiner:

Place: Irinjalakuda
Date:

Office Seal

Abstract

This project aims to develop an advanced security platform that automates critical aspects of web application vulnerability assessments. By leveraging both passive and active reconnaissance techniques, the platform enhances the efficiency and accuracy of vulnerability detection. Key features include Automated URL Discovery, which identifies web application URLs from various sources, ensuring comprehensive coverage of potential entry points, and JavaScript File Analysis, which inspects JavaScript files for vulnerabilities like insecure API calls and Cross-Site Scripting (XSS).

The platform also offers Customizable Scanning Options, allowing users to tailor scans based on specific requirements, such as scan depth and target domains. Real-Time Progress Notifications via Telegram keep users informed throughout the scanning process, enabling faster responses to potential threats.

Using a Multi-Layered Approach, the platform combines passive data gathering with active system probing, providing a more thorough security assessment. After each scan, it generates detailed reports, highlighting vulnerabilities and offering remediation strategies.

By automating the initial stages of vulnerability identification, the platform reduces manual effort, allowing security professionals to focus on analysis and remediation. This project helps organizations improve the security of their web applications by providing actionable insights and streamlining the overall assessment process.

Contents

Chapter 1
Introduction

1.1 Purpose

The system automates the discovery of web application URLs and analyzes JavaScript files for vulnerabilities using both passive and active reconnaissance techniques. It employs tools like Waybackurls and GAU to collect historical and publicly available URLs, ensuring hidden entry points are detected. JavaScript file analysis, through tools like JSFinder and a custom module, identifies vulnerabilities in sensitive areas such as API endpoints and authentication details. The combination of passive (data collection) and active (scanning) methods offers a thorough security assessment. With real-time progress updates via Telegram and customizable scanning options, the system streamlines vulnerability detection and enhances overall web security.

1.2 Intended Audience and Reading Suggestions

This document is intended for security professionals, system administrators, and developers focused on web application security. Security professionals should understand the system's use of passive and active reconnaissance techniques for comprehensive vulnerability identification, while system administrators will benefit from features like automated URL discovery, JavaScript file analysis, and real-time notifications. Developers can use the system to identify vulnerabilities in JavaScript files during development, and stakeholders are encouraged to review customizable scanning options and real-time updates to maximize security benefits.

1.2.1 Security Professionals

This document is intended for security analysts and experts tasked with identifying and mitigating web application vulnerabilities. The system's automated tools streamline assessments, enabling professionals to focus on higher-level analysis and remediation, with expertise in both passive and active reconnaissance techniques crucial for maximizing its effectiveness.

1.2.2 System Administrators

System administrators responsible for securing web infrastructure will find this document valuable for configuring and integrating the security tool into their systems. Understanding the tool's automated URL discovery, JavaScript file analysis, and real-time notifications is essential for ensuring comprehensive security coverage and timely responses to potential threats.

1.2.3 Developers

Web developers will find this document helpful in ensuring their applications are secure from flaws. By using the system's JavaScript file analysis, developers can detect insecure API calls, exposed endpoints, and data leaks early in the development process, allowing for proactive vulnerability identification before deployment.

1.2.4 Stakeholders

Stakeholders should focus on sections covering:

1. Passive and Active Reconnaissance Techniques: Learn about automated URL discovery and how both passive and active methods complement each other for thorough scanning.
2. JavaScript File Analysis: Understand how the system analyzes JavaScript files for security flaws and sensitive data exposure.
3. Customizable Scanning Options: Explore how the system allows for tailored scanning configurations to meet specific organizational needs.
4. Real-Time Progress Notifications: Learn how the system keeps users informed of scan progress through notifications, ensuring continuous monitoring and swift response.

1.3 Project Scope

The project focuses on developing a security tool that automates key aspects of web application vulnerability assessment, including URL discovery, JavaScript file analysis, and real-time progress updates via Telegram. The primary objective is to enhance the efficiency and accuracy of vulnerability scanning by offering customizable scanning options and comprehensive reporting features.

1.3.1 Automated URL Discovery

The tool automates the discovery of URLs from various sources like historical archives and publicly available data, ensuring comprehensive coverage of all potential entry points.

1.3.2 JavaScript File Analysis

The tool scans JavaScript files for vulnerabilities such as insecure API calls and sensitive data exposure, using tools like JSFinder and custom analysis modules.

1.3.3 Real-Time Progress Updates via Telegram

Real-time notifications via Telegram allow users to monitor the scan's progress without needing to remain logged into the system.

1.3.4 Customizable Scanning Configurations

Users can customize scan parameters, including target domains and scan depth, ensuring flexibility to meet diverse security needs.

1.3.5 Comprehensive Reporting

The tool generates detailed reports highlighting vulnerabilities and providing remediation suggestions, which are easily shareable in PDF format for security analysis.

1.4 Block diagram

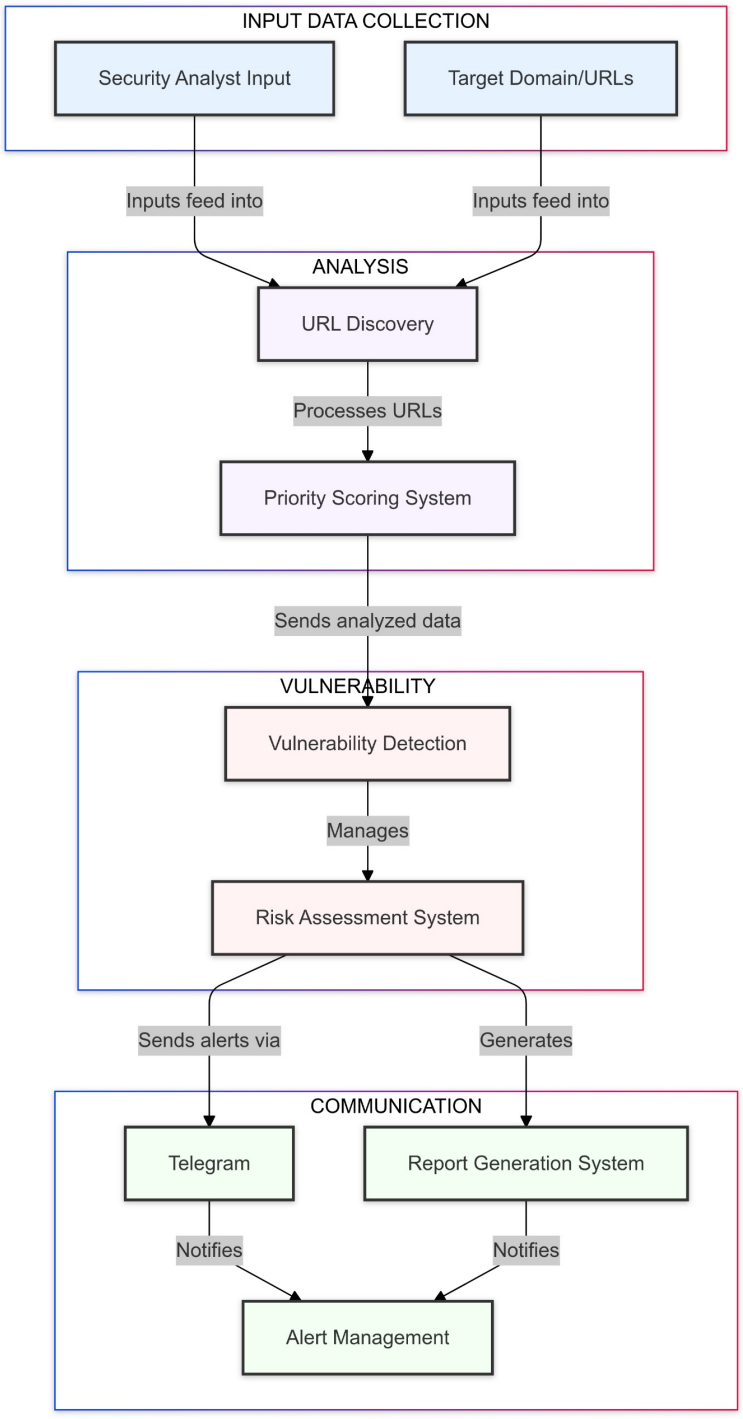

Figure 1.1: Block diagram

Chapter 2
Literature Review

2.1 Critique of the Review

The development of security assessment systems has been a vital area of research, with notable advancements in vulnerability detection, penetration testing, and automated analysis. Several studies have focused on implementing intelligent systems that utilize real-time data and advanced algorithms to improve the accuracy and efficiency of security assessments for target applications.

1. Shar & Tan (2013) examined the use of static analysis techniques for detecting web application vulnerabilities, focusing on common issues like cross-site scripting (XSS) and SQL injection. The study proposed an improved static analysis tool that reduces false positives and improves detection accuracy. Their findings emphasized the importance of integrating static analysis into the software development lifecycle to mitigate security risks early in development. **?**

2. Johns & Winter (2006) presented "RequestRodeo," a client-side protection mechanism against session riding attacks. Their research focused on the vulnerabilities posed by unauthorized access to user sessions, highlighting the potential for attackers to hijack sessions and manipulate user actions. The study proposed a framework that utilizes client-side validation techniques to ensure that requests made by users are legitimate. By implementing this solution, the authors demonstrated a significant reduction in the risk of session riding attacks, emphasizing the importance of enhancing client-side security measures in web applications. **?**

3. Nanda, Goyal, & Sharma (2017) provided a broad survey of the current state of web application security, highlighting the increasing complexity of web ap-

plications and the corresponding rise in security challenges. They review the most prevalent threats, such as SQL injection and cross-site scripting (XSS), and explore various defense mechanisms. The paper also discusses the importance of adhering to security best practices during the development process to minimize vulnerabilities and improve the overall resilience of web applications. **?**

4. Almohri, Chellappa, & Sun (2017) introduced a dynamic black-box testing framework for web vulnerability assessment, designed to identify security flaws without requiring access to the application's source code. This framework automates the process of discovering vulnerabilities, such as input validation issues, in real-time, which is particularly useful for assessing the security of web applications with little upfront knowledge. The study showcases the effectiveness of this method in identifying common security issues, thereby offering a flexible and scalable approach to vulnerability testing. **?**

5. Huang, Tsai, & Chen (2004) proposed a comprehensive testing framework for assessing the security of web applications, focusing on systematically identifying vulnerabilities. Their method involves structured testing processes that simulate real-world attacks, allowing developers to identify weak points in the application's design or implementation. By providing a clear testing framework, the authors aim to enhance the ability of developers to proactively address security concerns, improving the robustness of web applications against attacks. **?**

6. Doupe, Cova, & Vigna (2010) conducted a detailed analysis of black-box web vulnerability scanners, exploring why these automated tools often fall short in detecting complex vulnerabilities. The study reveals significant gaps in the capabilities of these scanners, particularly when dealing with dynamic content and modern web architectures. The authors argue that current scanners lack the sophistication to fully explore complex application logic, and they suggest improvements in scanner design to bridge this gap.**?**

7. Antunes & Vieira (2011) compared two common security testing methods: penetration testing and static code analysis. The research highlights the strengths of penetration testing in simulating real-world attacks to identify vulnerabilities, particularly in identifying SQL injection points. However, static code analysis excels at examining source code for security flaws without executing the code. The authors conclude that both approaches have complementary strengths and should be used together for a more comprehensive detection of vulnerabilities in web services. **?**

8. Bau & Mitchell (2011) provided a security evaluation of popular web appli-

cation frameworks, focusing on the common vulnerabilities that arise from their use. The study identifies structural flaws in the frameworks themselves, which could be exploited by attackers, and calls for the implementation of better security practices in their design. The authors suggest that developers should not rely solely on the security features provided by these frameworks but should also conduct independent assessments to ensure robust application security. **?**

9. Xie & Aiken (2006) focused on detecting security vulnerabilities in scripting languages through static analysis, which involves examining the code for security flaws without executing it. This approach helps in early identification of issues such as input validation errors, reducing the risk of attacks on web applications. By detecting these vulnerabilities early in the development process, the study aims to improve the security of applications before they are deployed, thus minimizing potential exploitation risks. **?**

10. Fonseca, Vieira, & Madeira (2009) conducted an empirical study to test and compare several web vulnerability scanning tools, specifically targeting their ability to detect SQL injection and cross-site scripting (XSS) attacks. Their research evaluates the accuracy and efficiency of different scanning tools, revealing that while some tools perform well in identifying these vulnerabilities, others miss critical issues. The authors provide recommendations for improving scanning tools to enhance their effectiveness in protecting web applications from these types of attacks. **?**

2.2 Summary

The research papers collectively focus on enhancing the security of web applications, cloud environments, and network systems through various methods such as intrusion detection, vulnerability scanning, and penetration testing. Key findings include the importance of feature selection in anomaly-based IDS, the role of dynamic and static analysis in detecting vulnerabilities, and the need for improved web application frameworks and testing methods to address complex security challenges. These studies emphasize hybrid approaches and automated tools to increase the accuracy, efficiency, and comprehensiveness of security assessments across different environments.

Chapter 3
Requirements

3.1 Functional Requirements

The platform operates on a robust and scalable architecture, designed to optimize performance and flexibility while ensuring security and reliability. The architecture includes:

3.1.1 User Authentication and Authorization

The system's User Authentication and Authorization is designed to ensure secure access for all users. It allows users to register and log in using encrypted credentials, ensuring sensitive data remains protected. Role-based access control (RBAC) is implemented, ensuring users have appropriate permissions based on their roles, whether they are security analysts, developers, or administrators.

3.1.2 Project Management

Users can create, update, and delete projects easily. Each project stores essential details, including the target domains, scan configurations, and scan status. This ensures that users have an organized view of their work, with the ability to manage multiple projects simultaneously and track progress across different assessments.

3.1.3 Scan Management

The platform supports multiple types of scans, such as URL discovery and JavaScript analysis. Users can initiate, pause, or stop scans as needed, giving them full control over the scanning process. This functionality is crucial for efficiently managing

time and resources, especially when scanning large web applications with varying complexities.

3.1.4 Discovered URLs Management

The system stores and manages all URLs discovered during scans. Each URL is given a status (scanned or unscanned) and a priority score based on its potential vulnerability. This helps security analysts focus their attention on the most critical areas, streamlining the vulnerability assessment process.

3.1.5 Vulnerability Assessment

Based on the discovered URLs, the system performs assessments to identify security risks, categorizing vulnerabilities based on severity. The results of each assessment are stored and linked to the relevant URLs, ensuring clear traceability from discovery to mitigation.

3.1.6 Reporting Capabilities

Users can generate detailed summaries of their scan results. Reports are customizable and can be downloaded in multiple formats, such as PDF or JSON. These reports include key information on vulnerabilities, severity levels, and recommendations, making them essential for communicating findings to stakeholders or integrating into other systems.

3.1.7 Notifications System

Users stay informed throughout the scanning process with real-time notifications via email or Telegram, keeping them updated on scan progress, detected vulnerabilities, and critical system alerts. This functionality improves user response times, allowing them to act quickly in the event of a security breach.

3.1.8 Dashboard and User Interface

The platform provides users with a visually intuitive way to monitor project status, scan progress, and results. The user-friendly interface ensures that both technical and non-technical users can navigate the platform easily, requiring minimal training to operate.

3.1.9 Hardware Requirements

3.1.9.1 Processor

1. Minimum: Dual-core processor (2.0 GHz)

3.1.9.2 Memory (RAM)

1. Minimum: 4 GB RAM

3.1.9.3 Storage

1. Minimum: 50 GB of available hard drive space

3.1.9.4 Network

1. Minimum: 1 Gbps network interface

3.1.10 Software Requirements

3.1.10.1 Operating System

1. Minimum: Linux (Ubuntu 20.04 or later, CentOS 7 or later)

3.1.10.2 Backend Software

1. Minimum: Python 3.7 or later

3.1.10.3 Frontend Software

1. Minimum:
   (a) Node.js (12.x or later)
   (b) React.js (16.x or later)

3.1.10.4 Database

1. Minimum: PostgreSQL (version 12 or later) or MongoDB (version 4.2 or later)

3.1.10.5 Development Tools

1. Minimum:
   (a) IDE or Code Editor

(b) Version Control System (Git) (2.25 or later)

3.1.10.6 Containerization

1. Minimum: Docker (20.10 or later)

3.1.11 Software Tools

The software tools that are being used in the development are mentioned below:

1. Waybackurls
2. Gau
3. Gospider
4. JSFinder
5. Custom
6. JavaScript Analysis Module
7. urldedupe
8. Custom Parsers
9. Regex Patterns
10. Tools for Header Check
11. Directory Scanning Tools
12. HTML Report Generator
13. Python PDF Library

3.2 Nonfunctional Requirements

3.2.1 Performance

The platform is optimized for high performance, enabling multiple users to access the system simultaneously without lag. Scans are designed to run efficiently, completing quickly even for complex or large applications.

3.2.2 Scalability

Designed with scalability in mind, the platform supports horizontal scaling to accommodate growing numbers of users, projects, and scans. Additional resources can be allocated as needed to maintain stable performance.

3.2.3 Security

Security is prioritized with robust measures in place. All user data is encrypted both in storage and transit, utilizing HTTPS for web traffic and hashing for passwords. The system is fortified against vulnerabilities like SQL injection and Cross-Site Scripting (XSS).

3.2.4 Usability

An intuitive, user-friendly interface ensures easy navigation, minimizing the learning curve. With a clear layout and accessible functionalities, the platform enables users to complete tasks efficiently, focusing on usability to avoid unnecessary complexity.

3.2.5 Reliability

Reliability is essential, with data integrity and availability consistently maintained. Scheduled backups prevent data loss, while a 99.5% uptime target ensures minimal downtime, providing users uninterrupted access to essential functionalities.

3.2.6 Maintainability

The system's modular, well-organized codebase adheres to best practices, simplifying maintenance and updates. New features and bug fixes are easily integrated with minimal impact, supported by comprehensive documentation to streamline future maintenance.

3.2.7 Compatibility

Compatible with major browsers, including Chrome, Firefox, and Safari, as well as mobile devices, the platform ensures flexible, accessible use across different environments and devices.

By meeting these functional and non-functional requirements, the platform delivers a robust, scalable, and secure solution for web vulnerability assessments, addressing both technical and organizational demands.

3.3 Supporting Design Diagrams

3.3.1 Use Case Diagram

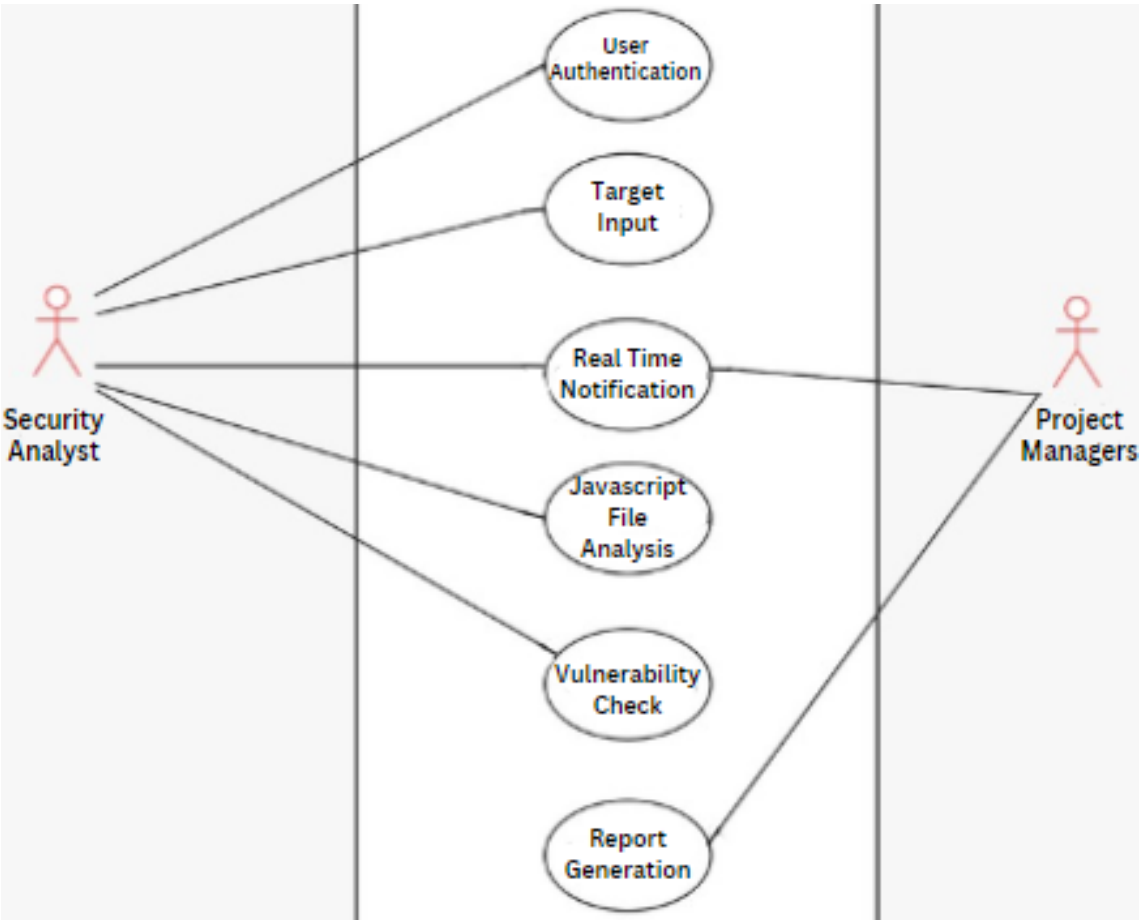

Figure 3.1: Use Case Diagram

Use Case Diagram: This diagram will depict the main actors (Security Analysts, Developers, and System Components) and their interactions with the system, such as initiating scans, analyzing vulnerabilities, and generating reports. It will also include real-time notifications, customizable scanning options, and user access management.

3.3.2 ER Diagram

Figure 3.2: ER Diagram Level 0

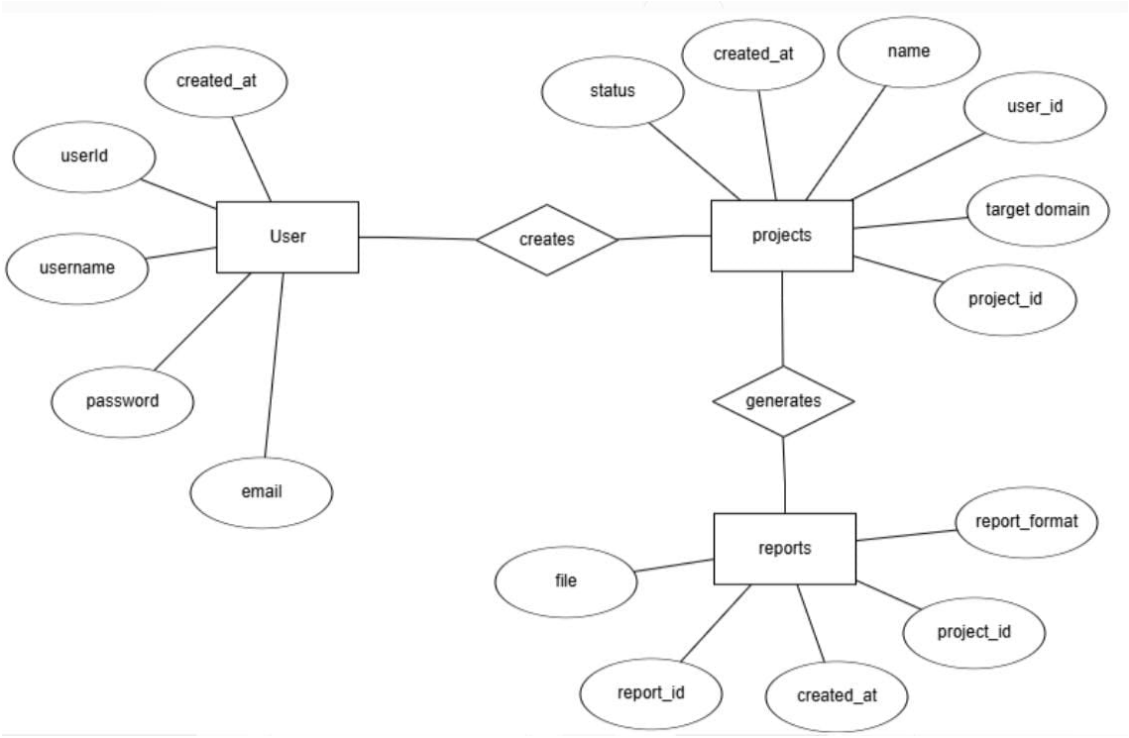

Figure 3.3: ER Diagram Level 1

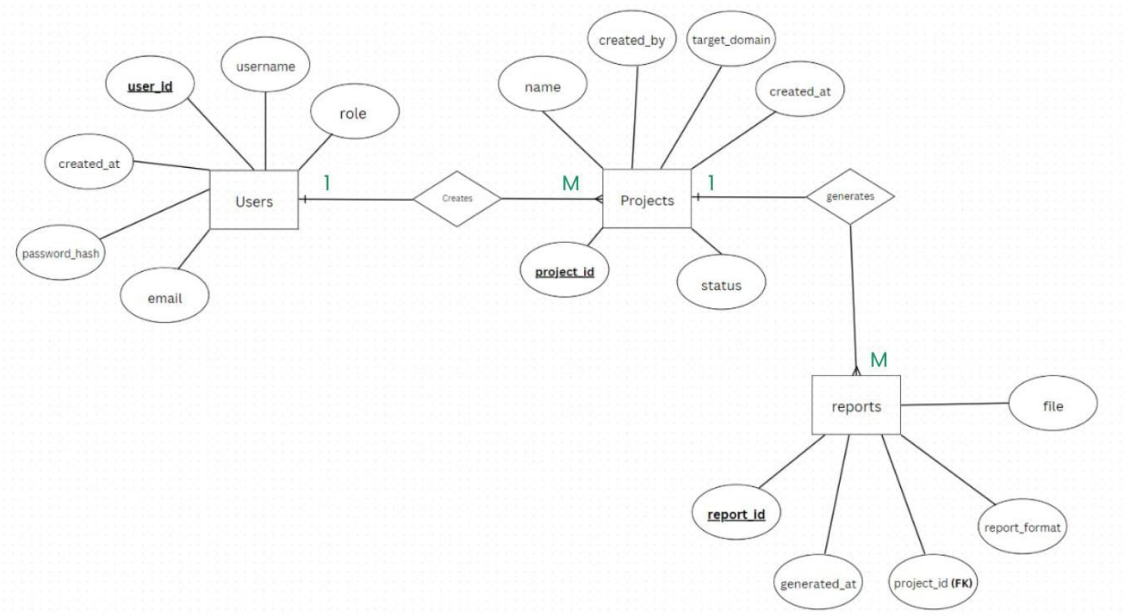

Figure 3.4: ER Diagram Level 2

The complete block diagram will visually represent the end-to-end process flow of the system, from data collection (video feed from Raspberry Pi) to signal control and real-time monitoring. This diagram will include:

3.3.3 Data Flow Diagrams (DFD)

Level 0 DFD (Context Diagram): The Level 0 DFD: The Data Flow Diagram illustrates the flow of data from the Security Analyst and Target Domain to the Security Assessment System, which processes the input and generates Assessment Reports.

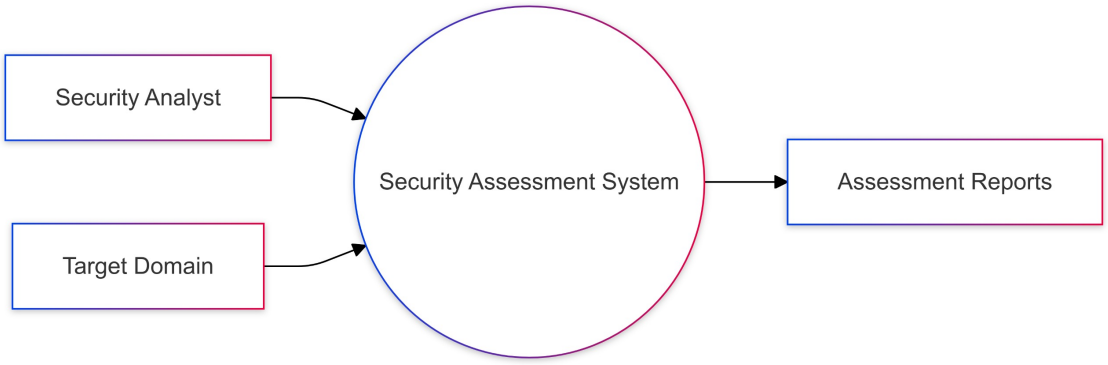

Figure 3.5: Level 0 Data Flow Diagram

Level 1 DFD: The Data Flow Diagram illustrates interactions between the Security Analyst, Target Application, and system components like Authentication, Recon Scanner, Vulnerability Assessment Module, and Report Generation, ultimately producing a Final Assessment Report.

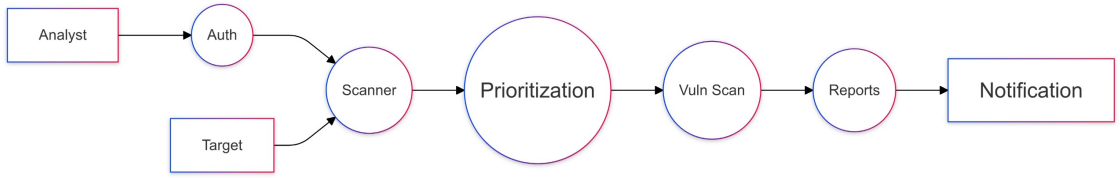

Figure 3.6: Level 1 Data Flow Diagram

Level 2 DFD: The Data Flow Diagram shows how the Security Analyst interacts with system components like Authentication, Recon Scanner, and Vulnerability Scanner, leading to the generation of a Final Assessment Report through steps such as URL Discovery, JavaScript Analysis, and Report Generation, with notifications sent through the Notification System.

Figure 3.7: Level 2 Data Flow Diagram

TIMELINE DIAGRAM

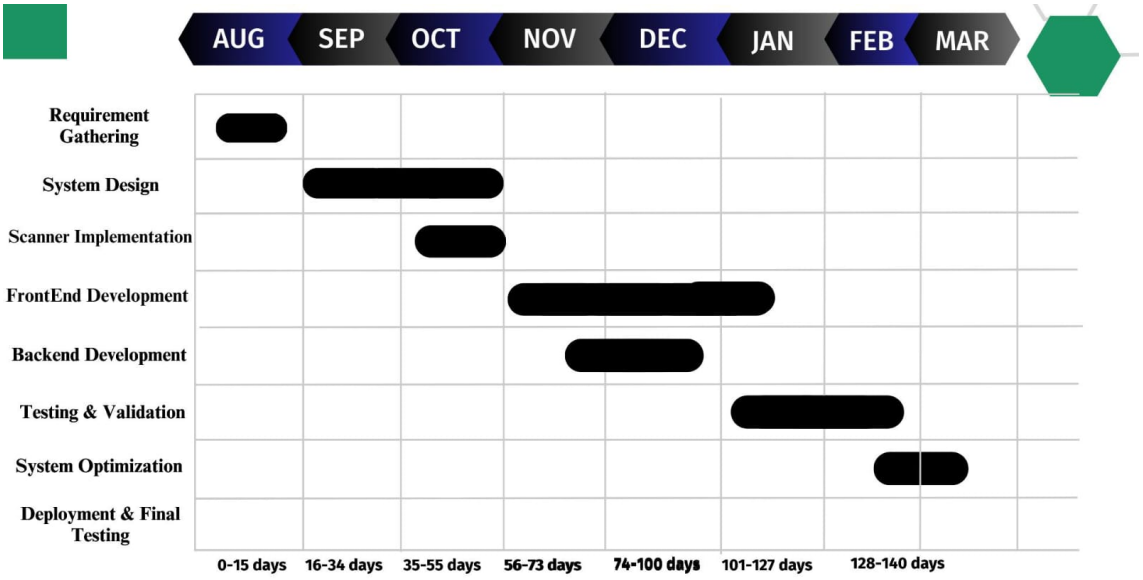

Figure 3.8: Timeline diagram

Chapter 4
Conclusion

The development of this automated vulnerability assessment tool for web applications represents a crucial advancement in cybersecurity. By integrating automated URL discovery, JavaScript file analysis, and customizable scanning options, the system significantly enhances the efficiency and effectiveness of vulnerability assessments. The tool utilizes advanced reconnaissance techniques, including both passive and active methods, to uncover potential entry points and vulnerabilities that could be exploited by attackers.

A standout feature is the automated URL discovery capability, which leverages tools like Waybackurls and Gau to systematically identify all potential access points to an application. Coupled with JavaScript file analysis, the system addresses critical security gaps, helping developers and security professionals detect issues early in the development cycle. Real-time notifications via Telegram keep users informed of scanning progress, enabling immediate responses to emerging threats.

The architecture of the system is designed for scalability and adaptability, utilizing a Python-based backend and a React-based frontend for an interactive user experience. Additionally, the implementation of a rule-based dynamic URL priority algorithm allows for intelligent management of scanning processes, ensuring that resources are efficiently allocated to the most critical areas.

In summary, this project is a significant contribution to the field of web application security. By automating essential tasks and providing comprehensive reporting, it empowers organizations to proactively identify and remediate vulnerabilities, thereby enhancing their overall security posture. As cyber threats continue to evolve, the need for innovative solutions like this one will remain paramount.

Chapter 5
References

5.1 Bibliography

1. Incremental Security for Cyber-Physical Systems Panda, A., Baird, A., Pinisetty, S., Roop, P. (2023). Develop approaches for incrementally enforcing security in cyber-physical systems, focusing on adaptability. *IEEE Access*, 11, 18475–18498. https://doi.org/10.1109/ACCESS.2023.3246121.

2. Cyber-Attacks in Power Systems Tatipatri, N., Arun, S. L. (2024). Review and analyze the impacts of cyber-attacks on power systems, along with detection and defense mechanisms. *IEEE Access*, 12, 18147–18167. https://doi.org/10.1109/ACCESS.2024.3361039.

3. Monitoring and Defense of Industrial CPS Jiang, Y., Wu, S., Ma, R., Liu, M., Luo, H., Kaynak, O. (2023). Explore methods for monitoring and defending industrial cyber-physical systems from attacks, with an emphasis on control-based solutions. *IEEE Transactions on Industrial Cyber-Physical Systems*, 1, 192–207. https://doi.org/10.1109/TICPS.2023.3317237.

4. Detection of Social Semantic Attacks Almousa, M., Anwar, M. (2023). Investigate methods for detecting social semantic attacks using character-aware language models, focusing on URL-based threats. *IEEE Access*, 11, 10654–10663. https://doi.org/10.1109/ACCESS.2023.3241121.

5.2 Webliography

1. Automated Removal of XSS Vulnerabilities Shar, L. K., Tan, H. B. K. (2013). Explore automated approaches for removing cross-site scripting vulnerabilities in web applications. Focus on methods to identify and neutralize XSS issues.

Information and Software Technology, 55(5), 950–962. https://doi.org/10.1016/j.infsof.2012.12.001.

2. RequestRodeo for Session Riding Protection Johns, M., Winter, J. (2006). Investigate client-side solutions like RequestRodeo for protecting against session riding attacks. Evaluate effectiveness in enhancing session security. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, 1324–1329. https://doi.org/10.1145/1141277.1141581.

3. Survey on Web Application Security Nanda, A., Goyal, V., Sharma, A. (2017). Analyze the state-of-the-art advancements in web application security, focusing on common vulnerabilities and solutions. *International Journal of Information Technology and Computer Science*, 9(1), 68–75. https://doi.org/10.5815/ijitcs.2017.01.07.

4. Dynamic Black-Box Web Vulnerability Testing Almohri, H. M., Chellappa, N., Sun, K. (2017). Study dynamic black-box testing techniques to detect vulnerabilities in web applications, emphasizing accuracy and adaptability. In *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, 379–396. https://doi.org/10.1109/SP.2017.25.

5. Framework for Web Security Assessment Huang, Y.-W., Tsai, C.-H., Chen, D. T. (2004). Develop and evaluate frameworks to assess the security of web applications. Emphasize thorough testing for reliability. *Computer Networks*, 44(4), 593–607. https://doi.org/10.1016/j.comnet.2004.02.005.

6. Analysis of Web Vulnerability Scanners Doupe, A., Cova, M., Vigna, G. (2010). Examine the effectiveness of web vulnerability scanners in identifying security weaknesses, with an emphasis on accuracy and reliability. In *Proceedings of DIMVA*, 111–131. https://doi.org/10.1007/978-3-642-14215-4_7.

7. Comparing Penetration Testing and Static Code Analysis
Antunes, N., Vieira, M. (2011). Compare penetration testing with static code analysis for detecting SQL injection vulnerabilities in web services. Evaluate detection efficacy. In *Proceedings of DSN*. https://doi.org/10.1109/DSN.2011.5958233.

8. Security Evaluation of Web Application Frameworks Bau, J., Mitchell, J. C. (2011). Assess the security capabilities of various web application frameworks, focusing on strengths and potential vulnerabilities. In *Proceedings of IEEE SP*. https://doi.org/10.1109/SP.2011.22.

9. Static Detection of Scripting Vulnerabilities Xie, Y., Aiken, A. (2006). In-

vestigate methods for static detection of vulnerabilities in scripting languages, focusing on accuracy in identifying security flaws. In *15th USENIX Security Symposium*.
https://www.usenix.org/conference/15th-usenix-security-symposium.

10. Comparison of Web Vulnerability Scanning Tools Fonseca, J., Vieira, M., Madeira, H. (2009). Compare various tools for identifying SQL injection and XSS vulnerabilities in web applications, evaluating effectiveness and efficiency. In *Proceedings of PRDC*, 365–372. https://doi.org/10.1109/PRDC.2009.23.

Chapter A
Appendix A: Glossary

1. **API**: Application Programming Interface. A set of protocols and tools that allows different software applications to communicate with each other, enabling integration and functionality sharing.
2. **URL**: Uniform Resource Locator. A reference or address used to access resources on the internet, such as web pages, images, and files.
3. **Vulnerability Assessment**: A systematic examination of a system, application, or network to identify security weaknesses that could be exploited by attackers, allowing organizations to address potential risks.
4. **Data Encryption**: The process of converting information into a coded format to prevent unauthorized access, ensuring that sensitive data remains confidential and secure during transmission and storage.
5. **Session Management**: The process of securely managing user sessions during their interaction with a system, including user authentication and maintaining the state of user interactions to enhance security and user experience.
6. **User Interface (UI)**: The means by which a user interacts with a computer system, encompassing the display screens, pages, and visual elements that facilitate user engagement with the application.
7. **Real-Time Updates**: Notifications or information provided to the user instantly as events occur, ensuring timely communication and awareness of important changes or alerts.