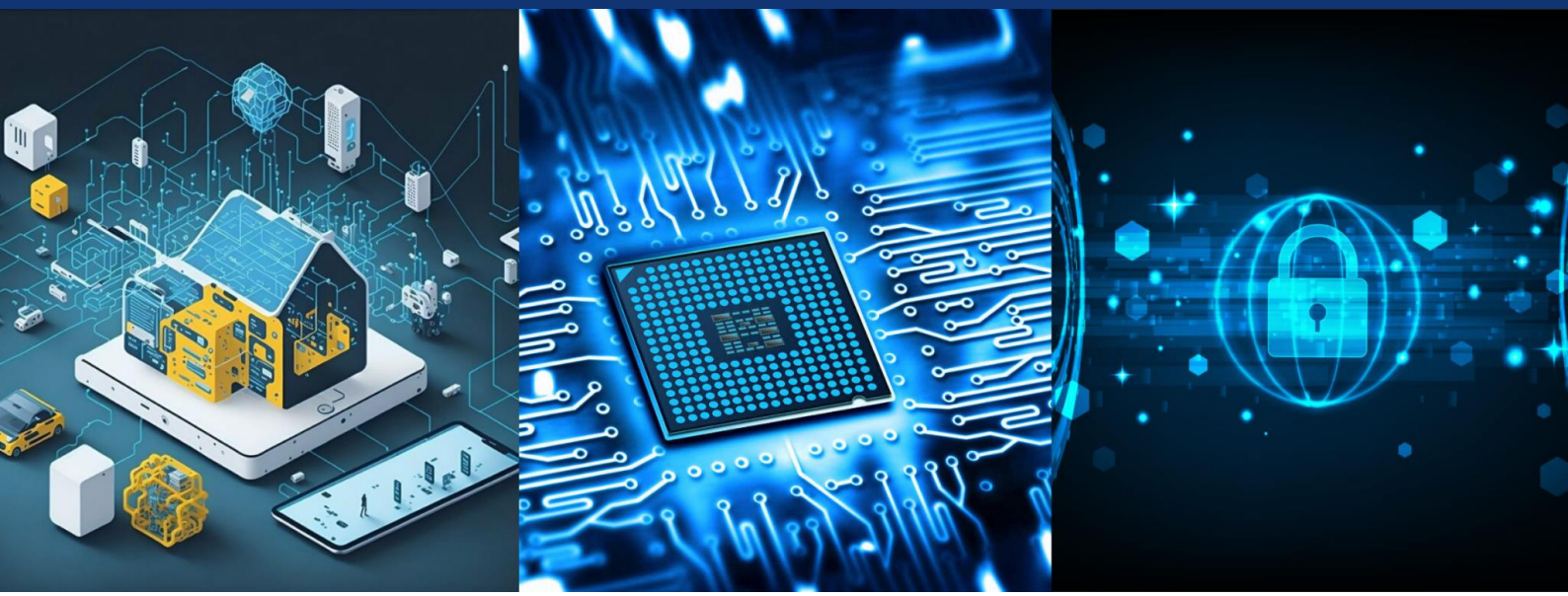
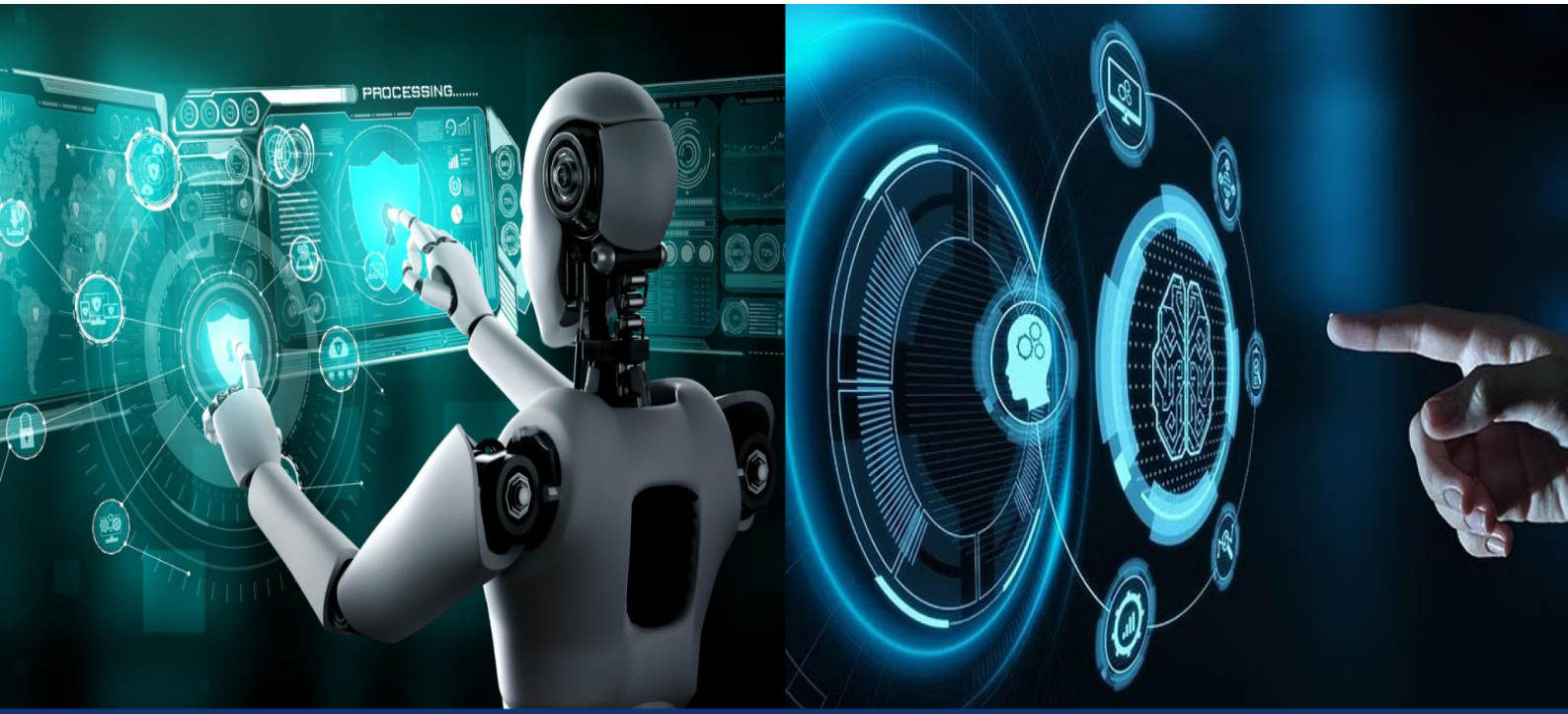
ISSN(O): 2320-9801
ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.771 **Volume 13, Issue 4, April 2025**

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Evaluation for Vulnerability Scanning in Web Applications

Prof. Smita Chunamari, Atharva Kharate, Shantanu Kesarakar, Vishwaja Pilankar, Kaivalya Umale

Assistant Professor, Dept. of Computer Engineering, A. C. Patil College of Engineering, Navi Mumbai, Maharashtra, India

UG Student, Dept. of Artificial Intelligence and Data Science, A. C. Patil College of Engineering, Navi Mumbai, Maharashtra, India

ABSTRACT: As the internet continues to evolve, the security of web applications has become a paramount concerns for businesses and individuals alike. The ”Website Vulnerability Scanner” project is designed to address this challenge by developing a comprehensive tool that automat-can detect, analyze, and report potential security vulnerabilities on websites. Thisscanner focuses on identifying a wide range of common web application vulnerabilities,including but not limited to SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), broken authentication, insecure file uploads, and security mis-configurations.The scanner leverages both active and passive scanning techniques to detect weaknesses by simulating potential attacks and analyzing a website’s response. It integrates with popular vulnerability databases such as OWASP and CVE, ensuring that the tool stays to date with the latest threats. The tool generates detailed reports with identified vulnerabilities, risk levels, and actionable recommendations for mitigation, aiding developers to strengthen the web security.This project is developed using Python, incorporating libraries such as ‘requests‘, ‘BeautifulSoup‘, and ‘nmap‘ for network scanning, and is built with modularity and scalability in mind. The tool aims to be user-friendly, making it accessible to both web developers and security analysts. By automating the vulnerability detection process, the project contributes to the overall enhancement of Web application security in a proactive manner.

KEYWORDS: Benchmarks; Software Vulnerability; Vulnerability Detection,Software Engineering, Information Security

I. INTRODUCTION

With the rapid expansion of the internet and the increasing reliance on web applications for both personal and professional activities, web security has become more critical than ever. As web applications store and process sensitive data, they are prime targets for cyber-attacks, which can lead to data breaches, financial loss, and reputational damage. One of the most effective ways to safeguard web applications is by identifying and addressing vulnerabilities before they can be exploited by malicious actors[4, 14].

The ”Website Vulnerability Scanner” project aims to tackle this issue by providing an automated tool that scans web applications for potential security weaknesses. These vulnerabilities can include well-known threats such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and broken authentication, which are among the top risks identified by organizations like OWASP (Open Web Application Security Project)[5].

This scanner automates the process of testing web applications for vulnerabilities by simulating various attack vectors. It performs both active and passive scans to detect potential weaknesses and uses current vulnerability databases to ensure accurate and up-to-date results[6, 10]. The tool generates detailed reports highlighting the discovered vulnerabilities, their severity levels, and recommendations for mitigation, helping developers and security teams take proactive measures.

Developed using Python, the tool incorporates open-source libraries and frameworks, offering flexibility and scalability. By providing a user-friendly interface and comprehensive scanning capabilities, this project not only enhances the security posture of web applications but also contributes to creating a safer digital ecosystem[13, 2].

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. MOTIVATION

With the growing reliance on web applications for everyday services, web security has become a critical concern. Cyber-attacks exploiting vulnerabilities such as SQL injection, cross-site scripting (XSS), and broken authentication pose significant risks to sensitive data and systems. Manual security testing is often time-consuming and inefficient, making automated solutions essential for timely detection of threats[9].

The motivation behind the ”Website Vulnerability Scanner” is to provide an efficient, automated tool that helps developers and administrators identify and fix vulnerabilities before they can be exploited. By simplifying the security assessment process and offering actionable remediation advice, this project aims to enhance web application security and make proactive protection more accessible to all.

III. PROBLEM STATEMENT

The ”Website Vulnerability Scanner” project addresses this challenge by developing an automated tool that scans web applications for security vulnerabilities, offering a scalable and consistent method for detecting and reporting potential risks. This tool aims to assist web developers and administrators in securing their applications by providing timely identification of weaknesses and recommendations for remediation.

IV. LITERATURE REVIEW

Although there are a vast number of Security testing applications available in the market, it’s hard to configure these applications to run on the environments as well as won’t be covering all of the three main perspectives that have identified during the research. Static code analysis, dependency security check, vulnerability check. The project provides them as a package for the organizations to make their day to day work easy.

A Comprehensive Study on Web Vulnerability Scanners: Existing Research and Proposed Features by *Azzedine Boudriga, Mouna Karray, and Slaheddine Fhima*
1. Introduction
With the rapid growth of web applications, ensuring their security has become a crucial challenge. Cyber attackers constantly exploit vulnerabilities in web applications to steal sensitive information, manipulate data, or disrupt services. Web vulnerability scanners play a critical role in detecting and mitigating these security threats. This paper provides an in-depth analysis of existing web vulnerability scanners, summarizing key findings from recent research and proposing a set of features for an advanced web vulnerability scanner tailored for improved efficiency and accuracy.
Research on Web Application Security Vulnerability Scanning Technology by *Jie Wu, Zhiqiang Wei, Xiuxian Zhao, and Wei L*[15]

A study conducted by Bin Wang et al. highlights the significance of automated vulnerability scanning in the power industry, where web applications have replaced client applications. The paper outlines two primary phases in vulnerability scanning: crawling and scanning. It discusses the importance of web crawlers in gathering comprehensive website structures and emphasizes the role of automated payload submission to detect security flaws such as SQL injection and XSS vulnerabilities. The research also points out the limitations of commercial scanners, such as high costs and excessive server load, which can impact normal business operations.
Development Processes of Vulnerability Detection Systems Jorge Reyes by*Jorge Reyes, Walter Fuertes, and Mayra Macas*[8]

They categorize detection approaches into software vulnerabilities and network/IoT vulnerabilities, with a focus on using machine learning techniques for vulnerability prediction. The study underscores the lack of a standardized methodology for developing VDS and recommends iterative methodologies for continuous monitoring. Additionally, it highlights the importance of integrating artificial intelligence and collaborative security measures into modern scanning tools.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

An Automatic Vulnerability Scanner for Web Applications by*Haibo Chen, Junzuo Chen, Jinfu Chen, Shang Yin, Yiming Wu, and Jiaping Xu* [7]
Their system, built on the AWVS framework, improves scanning efficiency by gathering and storing relevant information before initiating targeted scans. The study benchmarks the scanner against OWASP ZAP, demonstrating superior detection accuracy for SQL injection, XSS, and framework vulnerabilities. The authors stress the need for scanners to expand their detection scope dynamically based on observed attack patterns.
Comparison of Existing Scanners

Several commercial and open-source vulnerability scanners are widely used, including IBM
AppScan, Nessus, Acunetix, and OWASP ZAP. These tools differ in detection accuracy, scalability, and usability. The primary limitations identified include:

- High computational cost leading to server crashes.
- Limited support for dynamic applications with complex user interactions.
- Inadequate AI-driven vulnerability detection mechanisms.

To address these shortcomings, an enhanced scanner with a combination of static and dynamic analysis, AI integration, and better adaptability is needed[15, 8, 7].

Web Security Challenges
Web applications are frequently targeted by cyber attackers due to their accessibility and the sensitive data they handle. Common security threats include:
SQL Injection (SQLi) – Attackers manipulate database queries to gain unauthorized access to data.
Cross-Site Scripting (XSS) – Malicious scripts are injected into web pages to compromise user security.
Command Injection – Attackers execute arbitrary commands on the server, potentially compromising the entire system.
Despite advancements in web security frameworks, attackers continue to exploit vulnerabilities due to poor coding practices, outdated security patches, and weak authentication mechanisms. Traditional vulnerability assessment techniques are often manual, time-consuming, and prone to human error, making automated vulnerability scanning an essential component of modern cybersecurity strategies [9, 11].

Evolution of Vulnerability Scanning
Early vulnerability scanners were signature-based, relying on predefined attack patterns to detect threats. These scanners had limitations in identifying zero-day vulnerabilities and required constant updates to remain effective. The evolution of artificial intelligence and machine learning has significantly enhanced vulnerability detection methodologies by enabling anomaly detection, predictive analytics, and adaptive learning mechanisms[12].
The proposed system builds upon these advancements by integrating intelligent web crawling, automated payload injection, and machine learning-based classification of vulnerabilities. By leveraging a Random Forest Classifier, the scanner improves accuracy in predicting vulnerability types, thereby reducing false positives and enhancing security response mechanisms[16].

V. SYSTEM ARCHITECTURE

Architecture Overview
The proposed scanner will be designed using a microservices architecture to support modularity and scalability. Key components include:

1. Web Crawler – Collects target URLs and parameters.
2. Detection Engine – Executes scanning methodologies.
3. Machine Learning Module – Enhances zero-day vulnerability detection.
4. Database – Stores identified vulnerabilities and historical data.
5. Reporting Interface – Presents insights in an interactive dashboard.
6. Automated Update System – Ensures the scanner stays up-to-date with new threats.
7. Threat Intelligence Module – Continuously updates known exploits and attack vectors[15,8, 7].

Based on the findings from existing studies, the following key features are proposed for

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

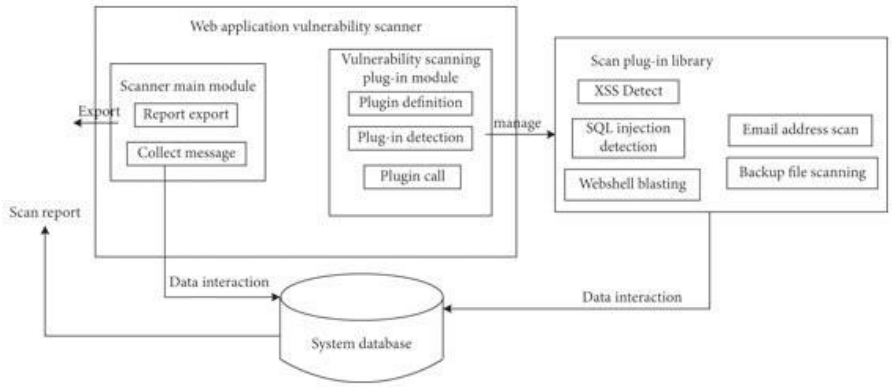


Figure 1: Flowchart

an advanced web vulnerability scanner:

1. User Interface (Web/CLI):

- Provides an interface (Web or Command-Line) for the user to input the target website URL.
- Displays the analysis status and allows downloading of vulnerability reports.[15, 8, 7]

2. Web Crawler

• Automatically explores the target website to discover all reachable links, forms, and query parameters. • Extracts HTML DOM structure to find dynamic input fields [15, 8, 7].

3. Payload Injector (Fuzzing Module)

- Injects a wide range of crafted malicious payloads (e.g., for SQL Injection, XSS) into de-tected input points.
- Handles both GET and POST methods with custom headers and cookies[15, 8, 7].

4. Threat Dictionary

- Contains a predefined list of payloads for different types of attacks such as XSS, SQLi,Command Injection, etc.
- Updated regularly based on OWASP top 10 and known CVEs[15, 8, 7].

5. Response Collector

- Captures HTTP responses returned by the web server after payload injection.
- Stores critical response metadata: status codes, content length, headers, and error messages.

6. Feature Extractor

• Processes each response to extract technical features for ML prediction: HTTP status code, Response time, Presence of known error patterns (e.g., SQL errors, HTML tags), Reflected payloads in the response body, Structural content changes pre- and post-injection[15, 8, 7].

7. ML Classifier (Random Forest Model)

- Trained on labeled response data to identify various attack types.
- Predicts vulnerability category (e.g., SQLi, XSS, or No Vulnerability).

8. Report Generator

• Compiles analysis results into a human-readable report in HTML/CSV format.

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VI. METHODOLOGY

Below is the methodology section for your Website Vulnerability Scanner and Prediction System in detail, written in a research paper format. This section explains the stepbystep approach to designing and implementing the system[3].

1. Problem Definition:

Identify the limitations of traditional scanners and the need for ML-based vulnerability prediction.

Problem Identification

Web applications are vulnerable to attacks such as SQL Injection (SQLi), CrossSite Scripting (XSS), and Command Injection.

Traditional vulnerability scanners lack the ability to classify vulnerabilities accurately or provide actionable insights[1, 10].

2. Data Collection:

Gather labeled HTTP responses by injecting payloads into test web apps

Web Crawler

Purpose: To explore the website and collect URLs, forms, and input fields[5, 3].

3.Feature Extraction:

Extract technical features (status code, error messages, etc.) from responses.

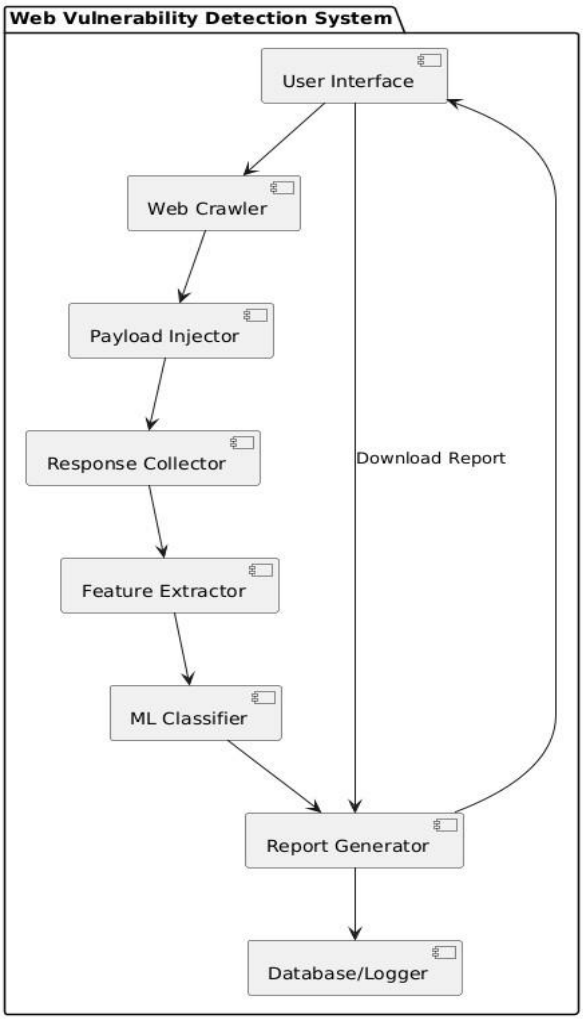


Figure 2: Architecture block diagram

© 2025 IJIRCCE | Volume 13, Issue 4, April 2025| DOI: 10.15680/IJIRCCE.2025.1304207

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

4. Model Training
Train a Random Forest Classifier to classify and predict vulnerability types.
5.Web Crawling and Injection
Crawl the target site, identify inputs, and inject predefined payloads.
6.Vulnerability Prediction
Use the trained model to predict and classify vulnerabilities in responses.

VII. REPORTING AND RECOMMENDATION

Generate a report with findings, risk levels, and remediation steps.
The proposed methodology provides a structured approach to designing and implementing a Website Vulnerability Scanner and Prediction System. By combining web crawling, payload injection, machine learning, and report generation, the system offers an accurate and userfriendly solution for detecting and classifying web application vulnerabilities. Future work includes improving the machine learning model with advanced techniques such as deep learning and expanding the range of detectable vulnerabilities[5, 6, 2, 15].

7.1 Algorithm
Step 1: Input URL
Input:
User provides a website URL to be scanned for vulnerabilities. Target Website URL, Predefined Payload Dictionary and Trained ML Model (Random Forest Classifier) Output: Vulnerability report with predicted categories (e.g., XSS, SQL Injection, etc.)

Step 2: URL Validation
Check: Validate that the URL is correctly formatted (e.g., starts with http:// or https://).
If invalid: Return an error message and stop further processing.

Step 3: Initialize and Set Up Directories
Import libraries (requests, re, BeautifulSoup, joblib, etc.), Load trained Random Forest model and encoders and Initialize headers and configuration

Step 4: Web Crawling
Process: Crawl the website to identify and extract forms, input fields, URLs, and key access points. Identify: Gather information about each form (e.g., action URL, method,input fields) and key directories for further analysis.

Step 5: Fuzzing / Payload Injection
For each form or parameter: For each payload in dictionary: 1. Inject payload 2. Send
GET/POST request 3. Save HTTP response

Step 6. Feature Extraction
For each response extract features: Status Code, Response Time, Reflected Payloads, Error
Messages, Content Length, Script/Tag count and Construct feature vector

Step 7. ML-Based Prediction
Input feature vector to ML Model, Predict vulnerability category, Get confidence score (if enabled), Log results

Step 8: Virus Scan
Check: If integrated, run a virus scan on the website or downloaded files using an API like VirusTotal to identify any malicious content.

Step 9: Report Generation
Store target URL, vulnerability details, Save in JSON/CSV/HTML format, Add timestamp and payload information.
Step 8: Output and Export
Display results to user, Allow report download, Optionally store in DB for auditing.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Step 10: End Process
Output: Indicate successful completion or show error details if the scanning process encountered issues[8, 1, 9].

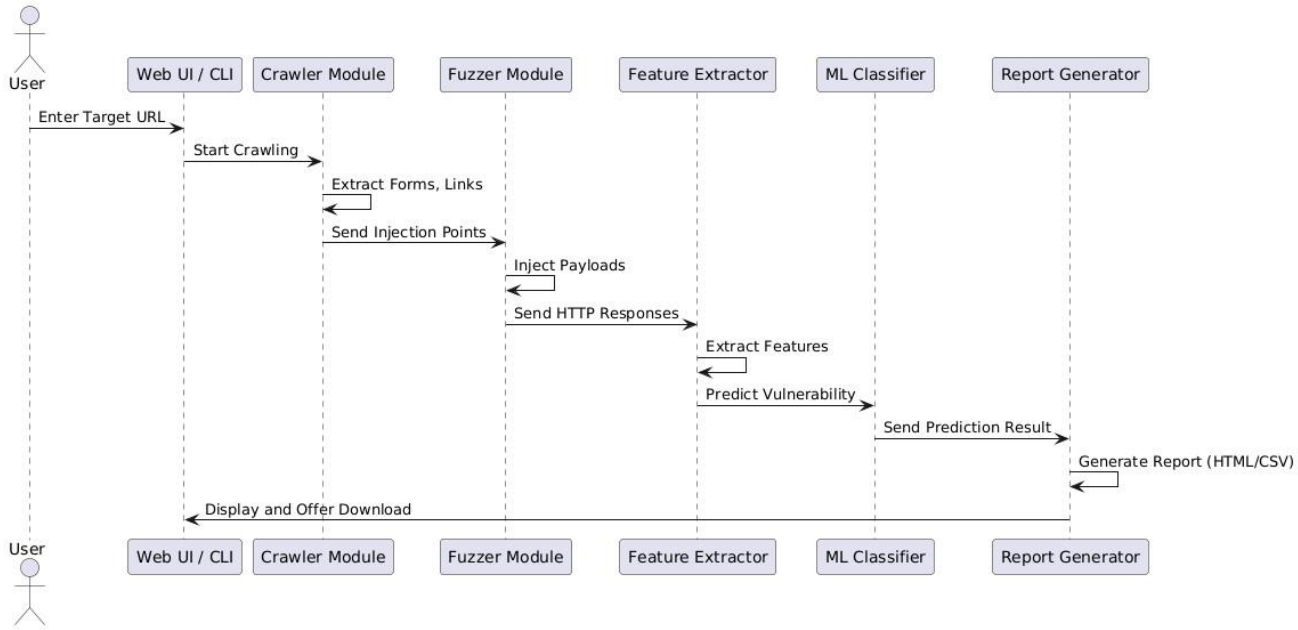


Figure 3: Sequence Diagram

REFERENCES

1. Richard Amankwah, Patrick Kwaku Kudjo, and Samuel Yeboah Antwi. "Evaluation of software vulnerability detection methods and tools: a review". *International Journal of Computer Applications* 169.8 (2017), pp. 22–27.
2. Ross J Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2010.
3. Gupta, P.; Parmar, D.S. Sustainable Data Management and Governance Using AI. World Journal of Advanced Engineering Technology and Sciences 2024, 13, 264–274. [Google Scholar] [CrossRef]
4. Tingli Cheng. "The Application of Web-Based Scientific Computing System in Innovation and Entrepreneurship". *Discrete Dynamics in Nature and Society* 2022 (May 2022). DOI: 10.1155/2022/1453889.
5. Smita R Chunamari and DG Borse. "Robust Framework for Certificateless Authenticated Key Agreement Protocol". *International Conference in Recent Trends in Information Technology and Computer Science (ICRTITCS-2012) Proceedings published in International Journal of Computer Applications (IJCA)(0975–8887)*. Vol. 27. Citeseer. 2012.
6. Smita R Chunamari and DG Borse. "Secure Schematic Model for Verifying Encrypted Image using Invariant Hash Function". *International Journal of Computer Applications* 975 (2013), p. 8887.
7. Jose Fonseca, Marco Vieira, and Henrique Madeira. "Evaluation of web security mechanisms using vulnerability & attack injection". *IEEE Transactions on dependable and secure computing* 11.5 (2013), pp. 440–453.
8. Mukesh Kumar Gupta, Mahesh Chandra Govil, and Girdhari Singh. "Predicting CrossSite Scripting (XSS) security vulnerabilities in web applications". *2015 12th international joint conference on computer science and software engineering (JCSSE)*. IEEE. 2015, pp. 162–167.
9. Attaluri, V., & Mudunuri, L. N. R. (2025). Generative AI for Creative Learning Content Creation: Project-Based Learning and Art Generation. In Smart Education and Sustainable Learning Environments in Smart Cities (pp. 239-252). IGI Global Scientific Publishing.
10. Zar Chi Su Su Hlaing and Myo Khaing. "A detection and prevention technique on sql injection attacks". *2020 IEEE Conference on Computer Applications (ICCA)*. IEEE. 2020, pp. 1–6.
11. Mamoona Humayun et al. "Cyber security threats and vulnerabilities: a systematic mapping study". *Arabian Journal for Science and Engineering* 45 (2020), pp. 3171– 3189.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 8.771| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

12. Abhishek Kumar et al. "OWASP Vulnerability Scanner". *International Research Journal of Modernization in Engineering Technology and Science* 03.04 (2021), pp. 2142–2148. ISSN: 2582-5208. URL: www.irjmets.com.
13. Carlos Kayembe Nkuba et al. "ZMAD: Lightweight Model-Based Anomaly Detection for the Structured Z-Wave Protocol". *IEEE Access* 11 (2023), pp. 60562–60577.
14. Puneet Panchal et al. "Reduced order modeling of electrical circuits: Simulation and hardware validation". *2017 international conference on computing, communication and automation (iccca)*. IEEE. 2017, pp. 1461–1465.
15. Ali Fathi Ali Sawehli. "Improving Software Security Testing of Software Development Life Cycle (SDLC) for Web-Based Applications By Providing A Quality Vulnerability
16. Assessment System (Web-Vs)". Master's thesis. MA thesis. Asia Pacific University of Technology and Innovation (APU), Dec. 2019. DOI: 10.13140/RG.2.2. 17114.29128. URL: https://www.researchgate.net/publication/338101873.
17. LIMKAH SENG. "Enhancement of automated black-box web application vulnerability assessment algorithms". PhD thesis. Universiti Teknologi Malaysia, 2019.
18. Bin Wang et al. "Research on web application security vulnerability scanning technology". *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE. 2019, pp. 1524–1528.
19. Mikael Willberg. "Web application security testing with owasp top 10 framework" (2019).

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details