iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78795

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Automated Web Vulnerability Detection and Intelligent Query Interface

Anshu V. Bind[1], Prasad R. Dawane[2], Abhay A. Dubey[3], Saurabh R. Singh[4], Dr. S. Riyazoodin[5]
Department of Computer Engineering, Theem College of Engineering, University of Mumbai, India

Abstract: ***Vulnerability Detection in Web Applications is a simple yet powerful tool designed to automatically detect security problems in web applications. In today's digital world, many websites and apps are targeted by hackers. It is very important for developers and organizations to find and fix these issues before attackers take advantage. Vulnerability Scanner helps in doing this quickly and efficiently. This tool is built using popular and reliable technologies such as Django, HTML, CSS, JavaScript, MySQL, and Docker. It provides an easy-to-use interface where users can start scanning their websites without needing deep technical knowledge. Vulnerability Scanner checks websites for weaknesses like open ports, outdated software, exposed data, and more. These checks help in identifying risks that can be fixed early. One of the main features of this project is its integration with Sn1per, a powerful tool used by cybersecurity professionals for ethical hacking and penetration testing. This allows Vulnerability Scanner to find a wide range of security threats and vulnerabilities. After scanning, our project generates detailed reports that include the problems found, their possible risks, and suggestions for how to f ix them. This helps users make smart decisions about improving their website's safety. This means it can run on different operating systems and setups easily. Organizations can deploy this project in various environments without any difficulty. This makes it suitable for both small startups and large companies. This project automates this process, so developers and security teams can focus on solving the problems instead of spending time finding them. This also helps in keeping the security process fast and reliable. The tool's reporting system provides export options in various formats such as PDF, CSV, and JSON. In conclusion, Vulnerability Scanner is a useful and efficient tool for managing web security.***

Keywords: OWASP, Web Vulnerability, SQLI, XSS, Django, Vulnerability Scanner, JSON, Docker.

I. INTRODUCTION

The evolution of the internet has transformed the way individuals and organizations interact, transact, and share information. Web applications are at the core of this digital transformation, providing platforms for online banking, shopping, learning, healthcare management, government services, and entertainment. As their usage expands, the reliance on web applications to store and process personal, financial, and organizational data has also increased significantly. While these applications have improved efficiency and convenience, they have simultaneously introduced new security challenges. Cyberattacks on web applications are becoming more frequent and sophisticated, exploiting weaknesses that often go unnoticed during development or deployment stages. Such weaknesses, known as vulnerabilities, may include flaws in coding, misconfigurations, improper validation of user inputs, outdated software components, and insufficient security controls. The exploitation of vulnerabilities can have devastating effects. Attackers may inject malicious code, bypass authentication systems, steal sensitive information, disrupt business operations, or deface websites. Real-world incidents have shown that even a minor flaw in a web application can lead to massive financial losses, legal consequences, and irreversible damage to an organization's reputation. Therefore, the detection and prevention of vulnerabilities are critical aspects of maintaining the security and reliability of web systems. Traditional methods of manual testing are often time-consuming, error-prone, and incapable of keeping pace with the complexity and scale of modern applications. Our project, titled "Vulnerability Detection in Web Applications," aims to provide an effective solution to this growing problem. The system is designed to perform automated scanning of web applications to detect potential vulnerabilities before they can be exploited by malicious actors. By integrating multiple open-source security tools and leveraging technologies such as Python, PostgreSQL, and web servers, the system ensures accuracy and efficiency in vulnerability identification. The approach involves taking a web application URL as input, performing systematic scanning, analyzing the detected issues, and generating detailed reports with insights and recommendations. This helps developers and organizations identify weaknesses early in the software lifecycle and implement corrective measures to strengthen application security.

©IJRASET: All Rights are Reserved

II. PROCEDURE FOR PAPER SUBMISSION

The development of the proposed Automated Web Vulnerability Detection and Intelligent Query Interface project was carried out in multiple structured phases to ensure systematic implementation and evaluation.

It begins with Phase 1: Project Planning, during which requirement analysis, literature survey, and project specification were completed in January 2026. This phase helped in identifying the limitations of existing vulnerability scanners and defining the objectives, scope, and technical requirements of the proposed system. The system architecture and technology stack were also finalized during this stage to establish a strong foundation for development.

Following this, Phase 2: Core System Development focused on building the fundamental infrastructure of the system during late January and early February 2026. This phase included backend development using Python frameworks, database setup using PostgreSQL, and development of the basic user interface. The main objective of this stage was to develop the scanning engine, URL crawler, and database connectivity required for vulnerability detection.

In Phase 3: Vulnerability Detection Module Development, which was carried out during February 2026, the focus shifted toward implementing the core security features of the system. During this phase, modules for detecting SQL Injection, Cross-Site Scripting (XSS), CSRF vulnerabilities, and security misconfigurations were developed. Feature extraction and parameter fuzzing techniques were also integrated to improve detection capability.

Phase 4: Analysis and Reporting Module, conducted during March 2026, concentrated on developing the dashboard and report generation features. This phase involved implementing visualization tools, integrating machine learning techniques for false positive reduction, and developing automated report generation in multiple formats. The objective was to provide clear insights, severity classification, and remediation suggestions for detected vulnerabilities.

Finally, Phase 5: Testing and Deployment was completed during late March and April 2026. This phase involved system testing, performance evaluation, bug fixing, and deployment preparation. Various test cases were executed to evaluate detection accuracy, system stability, and scan performance. The project concluded with the final deployment of the system in April 2026, marking the successful completion of the project.

III. EXPERIMENTAL SETUP

The proposed Automated Web Vulnerability Detection System was implemented and tested in a controlled experimental environment to evaluate its detection capability and system performance. The system was developed using Python as the core programming language with Django and FastAPI frameworks for backend development. PostgreSQL was used for database management, while HTML, CSS, and JavaScript were used for frontend development. Docker was used for containerization to ensure scalability and portability of the application.

The experiments were performed on a system with a minimum Intel i3/i5 processor, 8 GB RAM, and stable internet connectivity. The system accepts a target website URL as input along with scan configuration parameters such as crawl depth (3–5 levels), number of threads (5–10), timeout settings (20–30 seconds), and scan type (full scan or quick scan). The scanner was configured to detect common vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), CSRF, security misconfigurations, and exposed headers.

The experimental workflow included data collection from target websites, preprocessing of URLs and parameters, feature extraction, vulnerability detection using rule-based and AI-assisted techniques, and automated report generation. The performance of the system was evaluated using metrics such as accuracy, precision, recall, F1-score, scan completion rate, and false positive rate.

The results show that the proposed system is capable of detecting major web vulnerabilities with good accuracy while maintaining low false positives. The automated scanning and reporting features reduce manual effort and provide actionable remediation suggestions. This demonstrates that the system can be effectively used for automated security testing of web applications.

IV. METHODOLOGY

The proposed system follows a structured methodology for automated detection of vulnerabilities in web applications. The methodology consists of multiple stages including data collection, crawling, vulnerability detection, analysis, and report generation.

The process begins with target input, where the user provides the web application URL and scan configuration parameters. The system then performs web crawling to discover webpages, forms, parameters, and hidden endpoints. This helps in creating a complete attack surface map of the target application.

After crawling, the system performs feature extraction, where important elements such as input fields, request parameters, headers, and scripts are identified. These features are then passed to the vulnerability detection modules.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The vulnerability detection phase uses rule-based scanning and automated testing techniques such as parameter fuzzing, signature matching, and template-based scanning to identify common vulnerabilities like SQL Injection, Cross-Site Scripting (XSS), CSRF, and security misconfigurations. The system also integrates machine learning techniques to reduce false positives and improve detection accuracy.

Next, the analysis phase evaluates detected vulnerabilities based on severity level, risk score, and exploitability. The system classifies vulnerabilities into categories such as low, medium, and high risk to help users prioritize remediation.

Finally, the report generation phase produces detailed vulnerability reports containing vulnerability type, affected location, severity level, and recommended mitigation steps. The reports can be exported in multiple formats such as PDF, CSV, and JSON for further analysis. This methodology ensures systematic detection, analysis, and reporting of vulnerabilities while reducing manual effort through automation and intelligent analysis.

V. MOTIVATION

With the rapid growth of web applications and digital services, cybersecurity has become a critical concern for organizations and developers. Web applications frequently suffer from vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), CSRF, and security misconfigurations, which can lead to data breaches and financial losses. Traditional vulnerability detection methods often require significant manual effort, expert knowledge, and time, making them inefficient for continuous security assessment.

This motivated us to develop an automated web vulnerability detection system that simplifies the scanning process while improving detection accuracy and reducing false positives. The project also aims to integrate intelligent analysis techniques to enhance detection capability and provide meaningful insights through structured reports. Additionally, there is a need for cost-effective and easy-to-use security tools that can be used by developers during the development phase itself.

Therefore, the main motivation behind this work is to design a scalable, automated, and intelligent vulnerability detection system that helps in early identification of security flaws, reduces manual testing efforts, and improves the overall security posture of web applications.

VI. SYSTEM ARCHITECTURE

The system architecture of the proposed Automated Web Vulnerability Detection and Intelligent Query Interface is designed using a modular and scalable approach to ensure efficient vulnerability detection, analysis, and reporting. The architecture mainly consists of four major layers: User Interface Layer, Backend Processing Layer, Scanning and Analysis Layer, and Infrastructure Layer. These components work together to provide automated and intelligent vulnerability assessment of web applications.

1) User Interface Layer: The User Interface (UI) acts as the entry point of the system where users interact with the application. The interface is developed using HTML, CSS, and JavaScript to provide a simple and user-friendly environment. Through this interface, users can submit target URLs, configure scan parameters, start vulnerability scans, and view scan results. The UI also provides features such as scan history, report download options, and vulnerability visualization dashboards. This layer ensures ease of use so that even users with limited cybersecurity knowledge can perform vulnerability assessments efficiently.

2) Backend Processing Layer: The backend layer is responsible for handling system logic, request processing, and communication between different modules. This layer is implemented using Python frameworks such as Django and FastAPI to ensure high performance and scalability. The backend manages user requests, processes scan configurations, controls scanning workflows, and stores results in the PostgreSQL database. It also handles authentication, API communication, and report generation processes.

3) Scanning and Analysis Layer: This is the core component of the system where the actual vulnerability detection takes place. This layer consists of multiple modules working together:

- Web Crawler: This module scans the target website and discovers URLs, forms, parameters, and hidden endpoints. It helps in identifying the complete attack surface of the web application.
- Vulnerability Scanner: This module performs automated testing using predefined rules and templates to detect vulnerabilities such as SQL Injection, XSS, CSRF, and security misconfigurations.
- Feature Extraction Module: This component extracts important request and response features such as parameters, headers, scripts, and input fields required for vulnerability analysis.
- Machine Learning Module: The ML module helps reduce false positives by analyzing detected vulnerabilities and filtering incorrect results. It improves detection accuracy through intelligent classification techniques.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

- Fuzzing Module: This module sends modified inputs to test application behavior and identify hidden vulnerabilities.

4) Data Storage Layer: The data storage layer manages all system data including scan configurations, vulnerability results, reports, and user information. PostgreSQL database is used for structured data storage, while evidence data such as logs and scan outputs can be stored in file storage systems. This layer ensures proper data management, retrieval, and historical scan tracking.

5) Reporting and Visualization Layer: After vulnerability detection, the system generates detailed reports containing vulnerability type, severity level, affected components, and remediation suggestions. Reports can be exported in multiple formats such as PDF, CSV, and JSON. Visualization tools such as dashboards and charts help users understand security risks easily and take corrective actions quickly.

6) Infrastructure and Deployment Layer: The system is deployed using Docker containers to ensure portability and scalability. Containerization allows the system to run in different environments without dependency issues. Monitoring tools can also be integrated to track system performance and scanning activities.

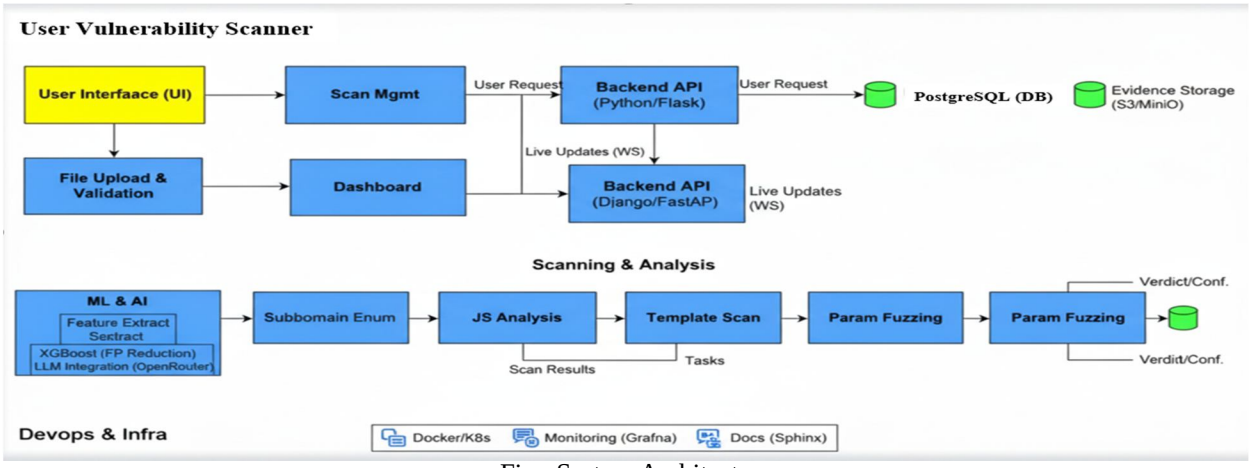


Fig. System Architecture

VII. RESULTS AND DISCUSSIONS

The experimental results demonstrate that the proposed system successfully detects common web vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and security misconfigurations with good accuracy and low false positive rates. The automated scanning approach reduced manual effort and improved detection efficiency. The generated reports provided clear vulnerability details and remediation suggestions, making the system practical for real-world web security assessment. The results confirm that the proposed system is effective, scalable, and suitable for automated vulnerability detection.

Figure I - The dashboard provides a centralized view of the vulnerability scanning system, showing key details such as total scans, security score, vulnerability severity, and recent scan activities. It also provides quick access to features like scan launch, CVE lookup, AI assistant, and report generation, helping users easily monitor and manage web security assessments.

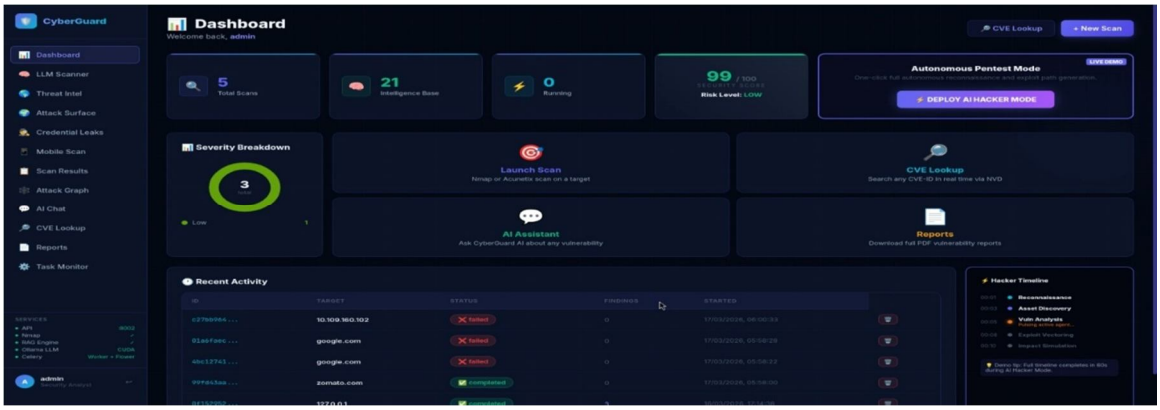


Figure II - The Launch Scan interface allows users to configure and start vulnerability scans by entering a target IP or URL and selecting scanning engines such as Nmap or Acunetix. It also provides a real-time scan log to monitor progress and results, making the scanning process simple and efficient for security assessment.

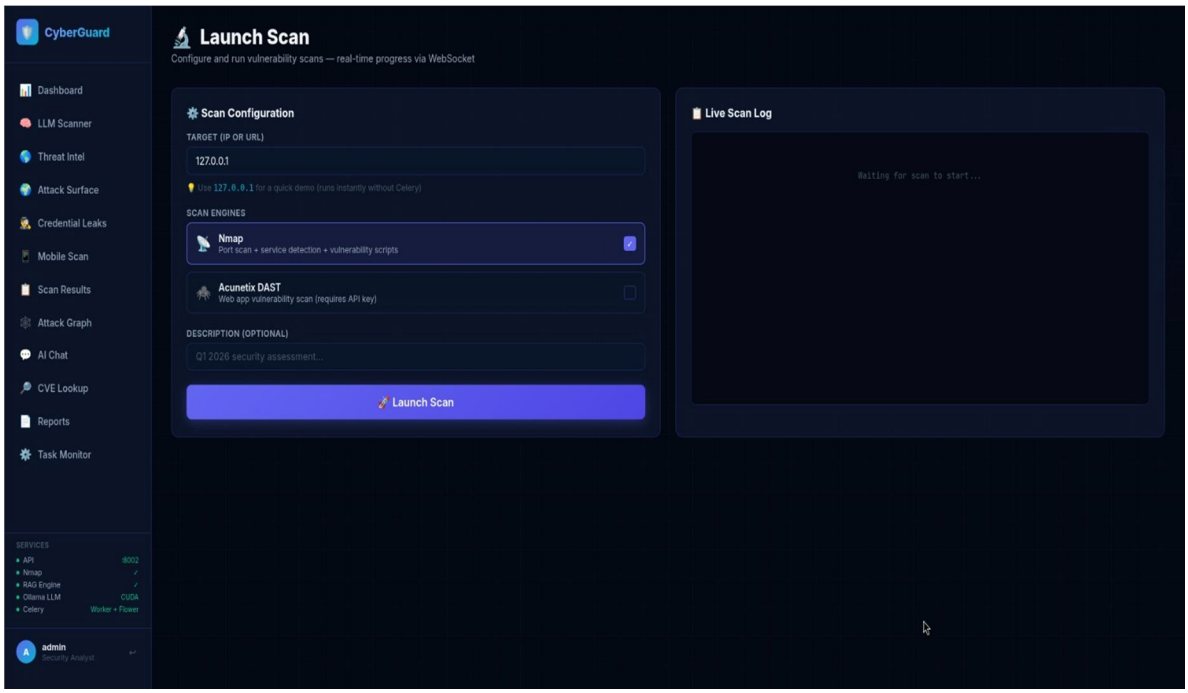


Figure III - The CVE Lookup interface allows users to search for known vulnerabilities using a CVE ID. It displays important details such as the CVSS security score, vulnerability description, publication details, and reference links, helping users quickly understand the severity and impact of specific security vulnerabilities.

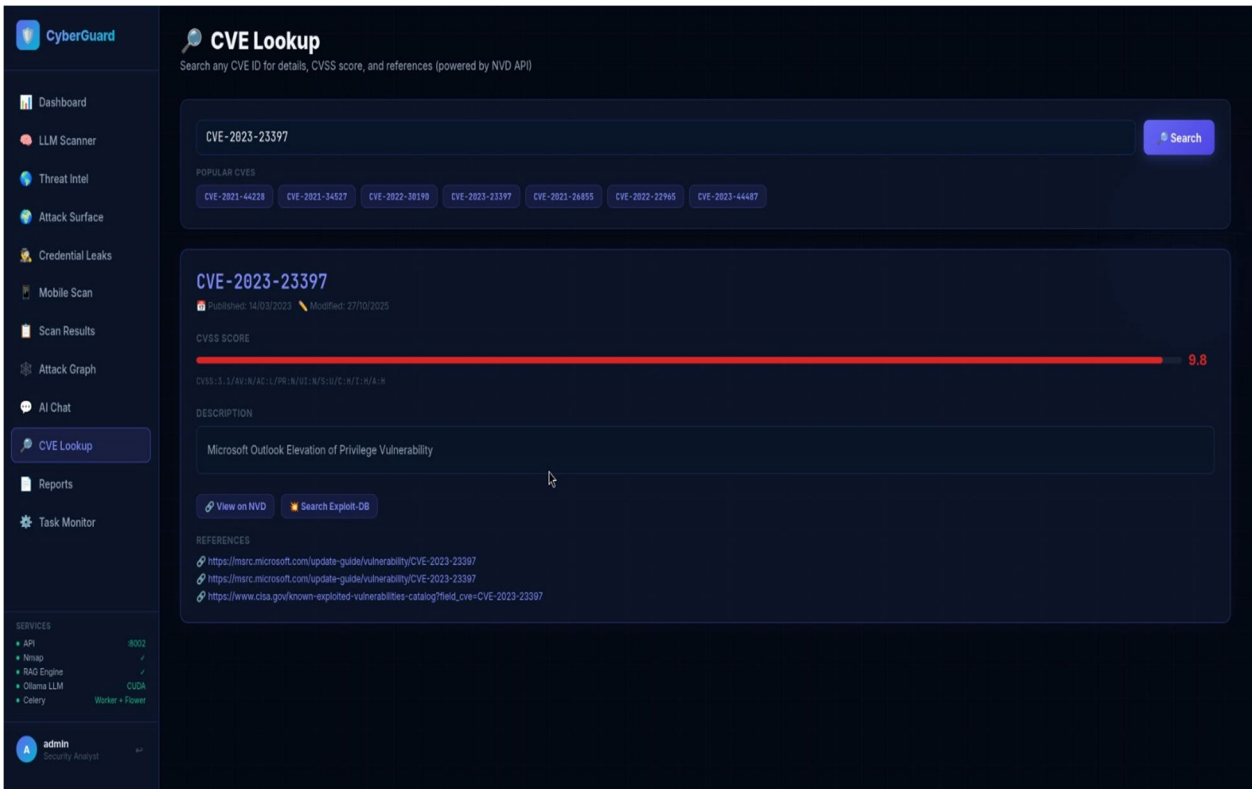


Figure IV - The CyberGuard AI Chat interface provides an intelligent assistant that helps users understand vulnerabilities, CVEs, and attack methods. It allows users to ask security-related questions and receive explanations along with remediation suggestions, making vulnerability analysis easier and more interactive.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

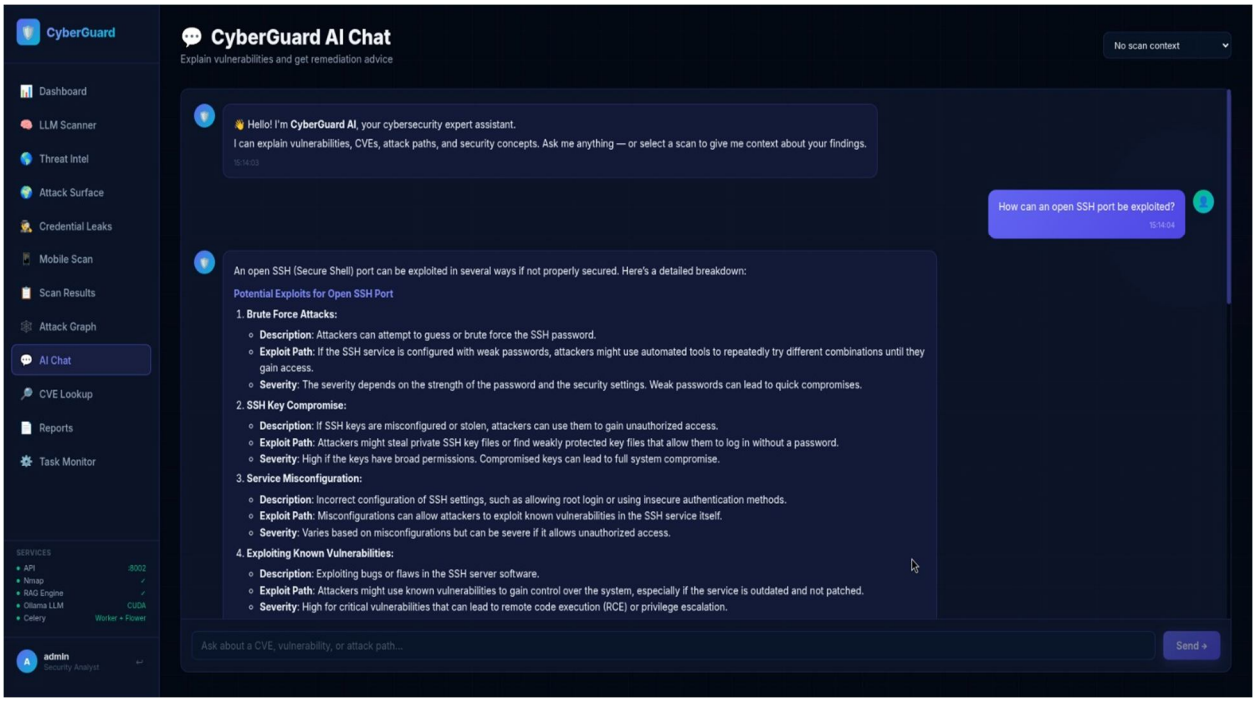


Figure V - The Reports interface displays completed vulnerability scans and allows users to download detailed security reports in PDF format. It shows scan targets, scan IDs, timestamps, and number of findings, helping users review and manage vulnerability assessment results efficiently. It also helps in tracking past scan history and comparing security results over time. This module supports better decision-making by providing organized and downloadable security assessment documentation.

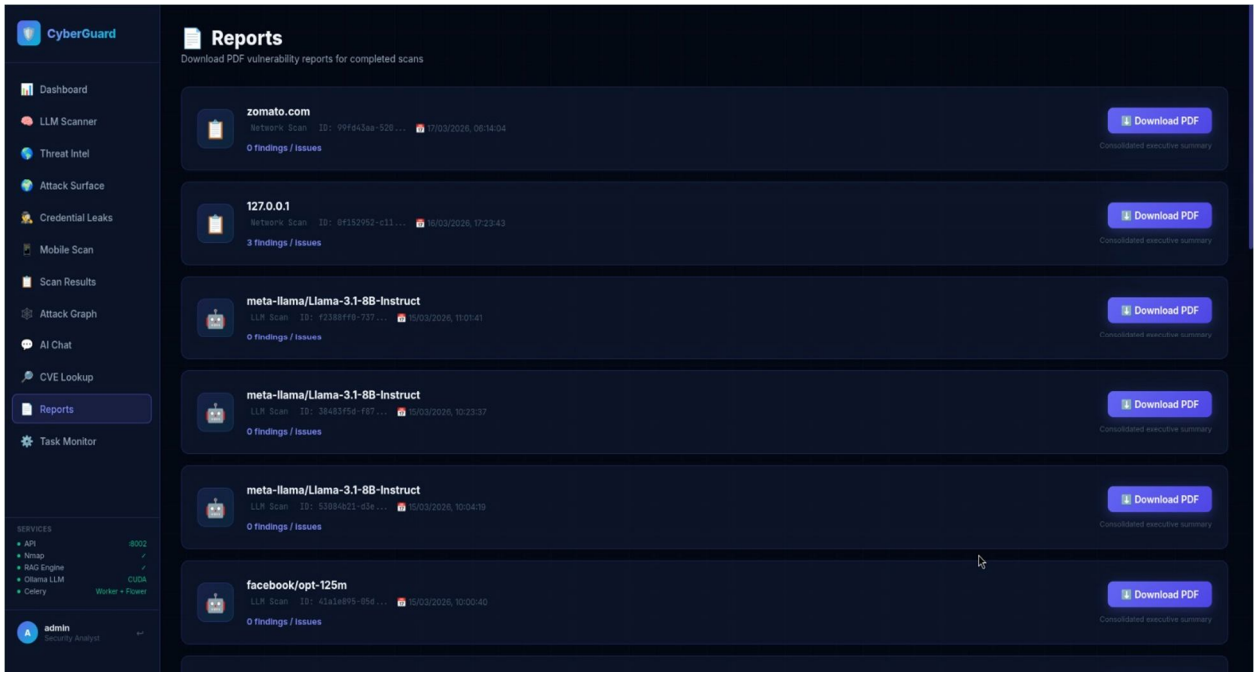
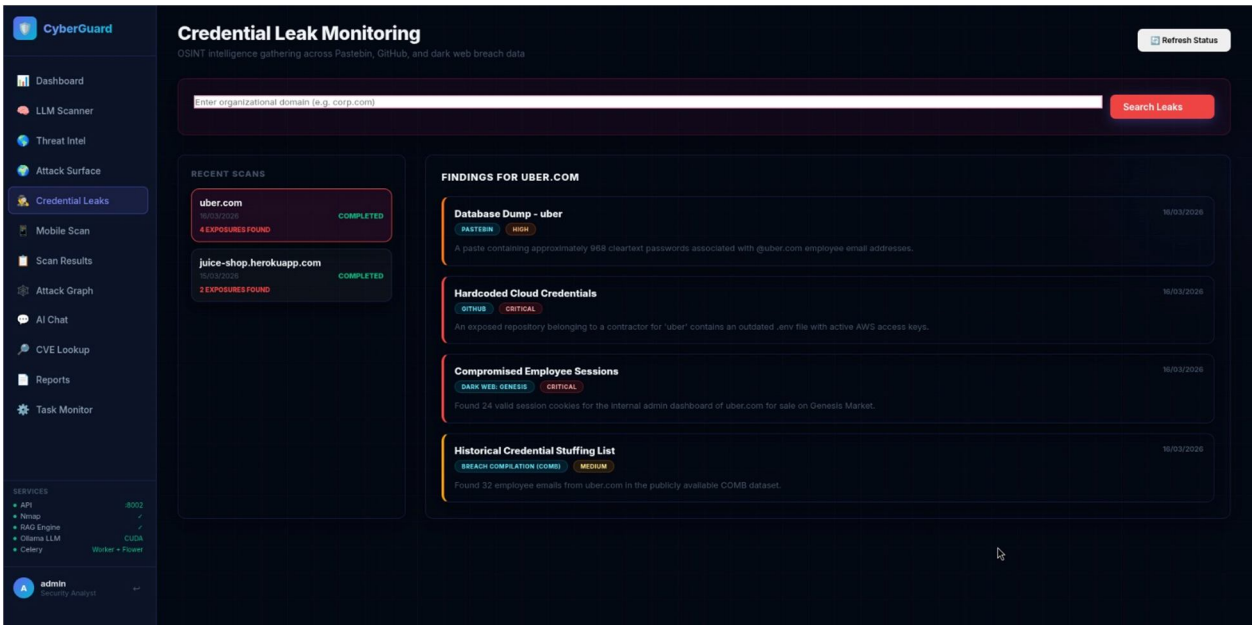


Figure VI - The Credential Leak Monitoring interface helps identify exposed credentials related to a target domain by collecting OSINT data from sources like GitHub, Pastebin, and dark web breach datasets. It displays detected leaks, their severity levels, and descriptions to help security teams take preventive actions. This module improves security awareness by detecting compromised accounts and sensitive data exposures.



VIII. CONCLUSION

In conclusion, the development of this web application vulnerability detection system has successfully established a robust framework for identifying security weaknesses in web applications. The project's architecture, encompassing a modular scanning engine, a comprehensive vulnerability database, and a user-friendly interface, provides a solid foundation for automated security assessment. The integration of automated crawling, intelligent vulnerability detection modules, and AI-assisted analysis further strengthens the capability of the system to detect security flaws efficiently. The experimental evaluation, guided by carefully defined input parameters and performance metrics, demonstrated the system's ability to effectively detect common web vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), and security misconfigurations while providing actionable insights through detailed reporting. The automated workflow also reduces manual testing effort, improves detection speed, and enhances the reliability of security assessment processes. Although the system shows promising results in accuracy and efficiency, the dynamic nature of web security necessitates continuous improvement to address emerging threats and zero-day vulnerabilities. Future enhancements can include real-time vulnerability monitoring, integration with CI/CD pipelines for DevSecOps practices, advanced machine learning models for behavioral analysis, and support for cloud and API security testing. The outlined future scope highlights a clear roadmap for enhancing the system's capabilities, including deeper integration of AI for more intelligent detection and false positive reduction, expanded coverage for modern API and client-side vulnerabilities, and a transition towards a scalable, fault-tolerant architecture. Additionally, improvements in automated remediation suggestions and risk-based prioritization can further enhance the practical usability of the system. Ultimately, this project contributes to strengthening web application security by providing a systematic, scalable, and evolving tool for developers and security professionals.

IX. ACKNOWLEDGEMENT

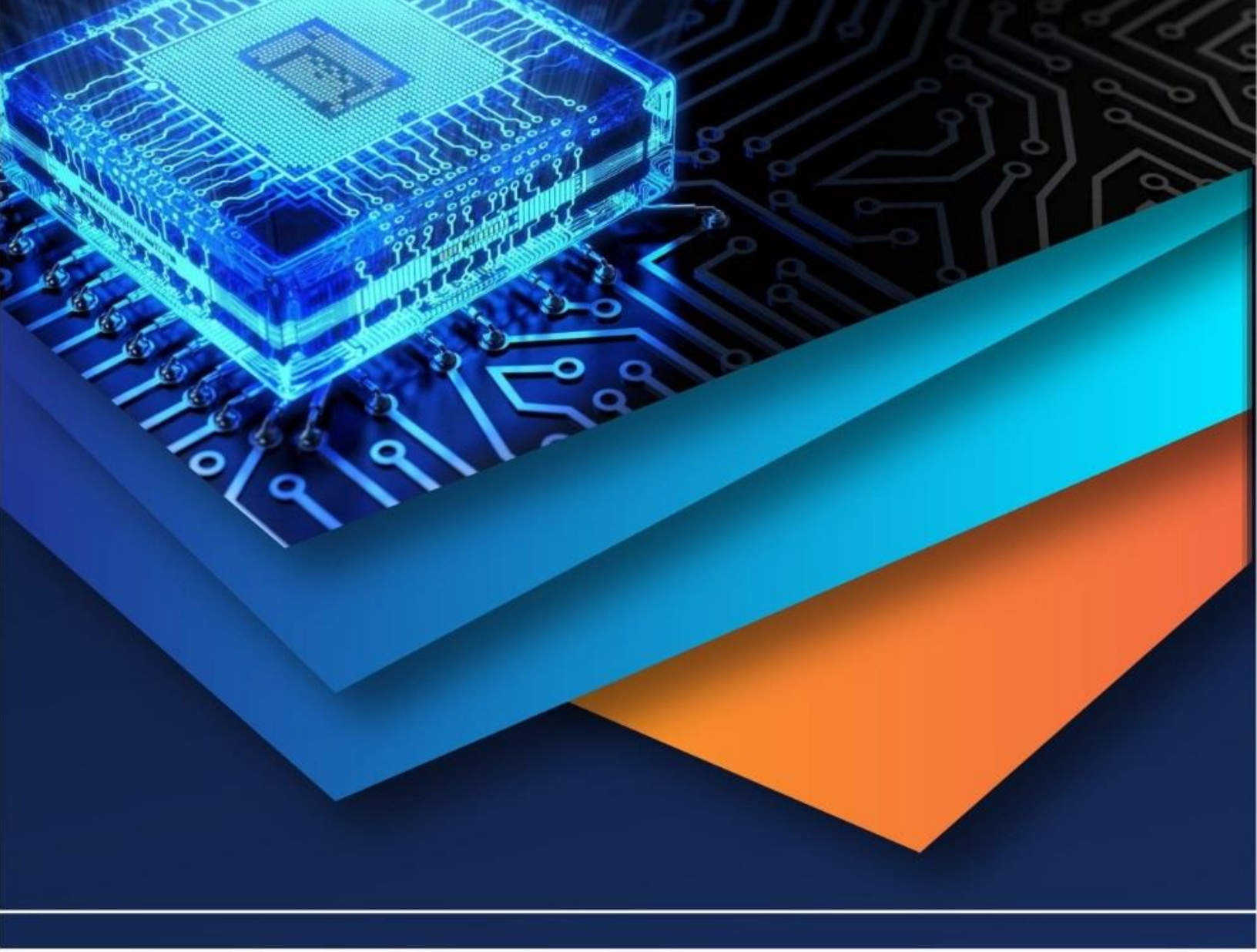
First and foremost, we thank God Almighty for blessing us immensely and empowering us at times of difficulty like a beacon of light. Without His divine intervention we wouldn't have accomplished this project without any hindrance. We are also grateful to the Management of Theem College of Engineering for their kind support. Moreover, we thank our beloved Principal Dr. Riyazoddin Siddiqui, our Director, Dr. N. K. Rana for their constant encouragement and valuable advice throughout the course. We are profoundly indebted to Prof. Raees Ahmad, Head of the Department of Computer Engineering and Prof. Jagruti More, Project Coordinator for helping us technically and giving valuable advice and suggestions from time to time. They are always our source of inspiration. Also, we would like to take this opportunity to express our profound thanks to our guide Dr. S. Riyazoodin, Assistant Professor, Computer Engineering for his valuable advice and whole hearted cooperation without which this project would not have seen the light of day. We express our sincere gratitude to all Teaching/Non-Teaching staff members of Computer Engineering department for their co-operation and support during this project.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

REFERENCES

[1] A. I. M. D. A. Al-Helali, "Web vulnerability scanning tools," International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), vol 10, no.1, 2024, pp. 8-15.
[2] S. K. N. M. Preeti Devadiga, Shruti Varankar, "Ai-based web vulnerability scanner: A comprehensive review," SSRN, Sept 2024.
[3] J. G. Sheetal Bairwa, Bhawna Mewara, "Vulnerability scanners: A proactive approach to assess web application security," IJCSA, Vol.4, No.1, March 2014.
[4] J. S. A. S. K. P. A. P. Rathod, S.K. Jagtap, "An automatic vulnerability scanner for web applications with firewall techniques," JETIR, Vol.9, No.8, Aug 2022.
[5] T. A. K. S. Rabaya Sultana Mim, Abdus Satter, "Automated software vulnerability detection using codebert and convolutional neural network," ENASE, Feb 2024.
[6] H. S. S. K. S. P. Aniket Maurya, Atharva Sail, "Webguard: A web vulnerability scanner for web applications," IJRTI, Vol.10, No.4, April 2025.
[7] E. P. C. Ajah Ifeyinwa, Agu Sunday, "Network vulnerability analysis," IJC, Volume 34, No1, pp 129-139.
[8] P. D. I. Samruddhi S. Khedkar, "Automated penetration testing," IJIRID, Vol 3, Issue 5, October 2024.
[9] D. S. K. S. H. P. R. Punyaben Patel, Reddyvari Reddy, "Enhancing web application security: (web vulnerability scanner)," IJERT, Volume 13, Issue 03 March 2024.
[10] P. Morge, "Automated web application vulnerability scanner," IRJMETS, Volume 07, Issue 02 February 2025.
[11] V. K. V. K. N. K. B. Nishika Reddy, Akarsh Kumar Trivedi, "Web vulnerability scanner– poc bomber," IJNRD, Volume 9, Issue 5 May 2024.
[12] E. A. Farah Abu-Dabaseh, "Automated penetration testing: An overview," CS IT CSCP, pp 121-129, 2018.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)