WEB APPLICATION VULNERABILITY SCANNER

[1] A.LAKSHMAIAH, [2] N.SANTHOSH, [3] M.KARTHIK KUMAR YADAV, [4] Y.DURGA PRASAD,[5] SK.ANWAR PASHA
[1]Assistant Professor, Department of CS, Sri Indu College Of Engineering & Technology, Hyderabad.
[2,3,4,5] U.G. Scholar, Department of CS, Sri Indu College Of Engineering & Technology, Hyderabad.

ABSTRACT

As web applications continue to play a vital role in modern digital systems, they are increasingly exposed to security threats such as SQL Injection, Cross-Site Scripting (XSS), and server misconfigurations. These vulnerabilities can lead to serious consequences for organizations, making proactive security assessment essential. To address this challenge, automated scanning solutions are necessary to identify weaknesses before they can be exploited. This paper presents the development of a website vulnerability scanning system implemented using Python. The proposed system incorporates Google OAuth2 for secure user authentication and supports multiple user roles to manage access and functionality effectively. It is designed to automatically scan user-submitted websites for potential security issues. By utilizing Python libraries such as requests, BeautifulSoup, and scapy, the system is able to detect common vulnerabilities and provide comprehensive reports for administrators. The evaluation of the system demonstrates its capability to efficiently and accurately identify security flaws in web applications. By automating the vulnerability detection process, the proposed solution enhances the overall security posture of web systems and assists organizations in maintaining safer and more reliable online platforms.

Keywords: Website Security, Vulnerability Scanner, Python, SQL Injection, Cross-Site Scripting, Web Application Security.

I. INTRODUCTION

Web applications are widely used for e-commerce, banking, healthcare, and other domains. As their usage increases, so does the risk of cyberattacks. Reports indicate that more than 60% of organizations experience web-based vulnerabilities annually. Traditional manual testing approaches are time-consuming and error-prone. Hence, there is a demand for automated vulnerability scanners capable of detecting threats before exploitation. This paper proposes a Website Vulnerability Scanner System implemented using Python. The scanner validates user-submitted websites, performs vulnerability detection, and reports issues. The system is designed to be lightweight, modular, and user-friendly for both administrators and end-users.

II. METHODOLOGY

System Design

The system is designed as a modular, role-based website vulnerability scanning architecture that integrates multiple coordinated components to ensure secure and efficient operation. It begins with an Authentication Module that uses Google OAuth2 to verify user identity and assign roles such as user or administrator. Once authenticated, users interact with the Website Submission Module to enter URLs, which are validated and sanitized before entering the scan queue. The Scan Management Module orchestrates scanning operations by managing task scheduling and real-time progress tracking. The Vulnerability Detection Module forms the core of the system, executing checks for SQL Injection, Cross-Site Scripting, insecure headers, SSL/TLS issues, and open ports using Python libraries like Requests, BeautifulSoup, and Scapy. Detected issues are stored in the database and compiled into structured reports through the Report Generation Module, enabling administrators and users to review results through a lightweight and intuitive interface built using Python-based backend frameworks.

Design Considerations

Several key design considerations guided the development of the system to ensure security, scalability, and usability. First, modularity was prioritized so that each component—authentication, scanning, detection, and reporting—operates independently, allowing easy maintenance and future upgrades such as ML-based analysis. Security considerations were central, particularly in handling user input, implementing OAuth2

authentication, and preventing unauthorized access to scanning results. Performance optimization was also essential, leading to the use of lightweight Python libraries to minimize resource consumption while maintaining accurate detection capabilities.The system's design additionally considers compatibility and deployment ease by supporting simple database setups like SQLite/MySQL and ensuring platform flexibility through Python frameworks. Lastly, user experience was emphasized by enabling clear scan progress visibility, generating comprehensive reports, and ensuring smooth access for both administrators and end-users with varying technical expertise.

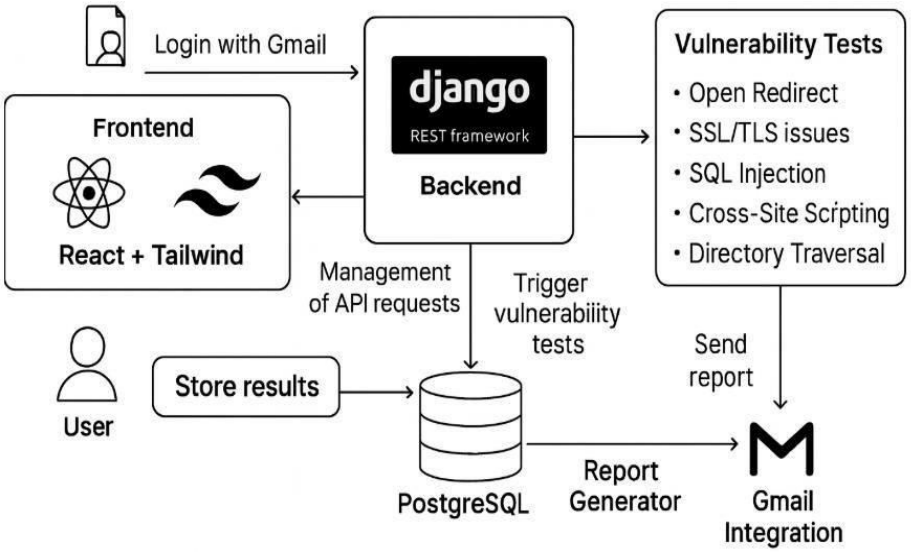

Fig 1:

III. MODELING AND ANALYSIS

Vulnerability Detection Test Cases

Test Case ID	Input	Expected Output	Actual Output	Status
TC-VS-01	Login page SQLi payload	Detect SQL Injection	SQLi detected	Pass
TC-VS-02	XSS script in search param	Detect XSS	XSS reflected	Pass
TC-VS-03	Header inspection	Missing headers flagged	Headers flagged	Pass
TC-VS-04	Admin directory access	Detect directory listing	Listing found	Pass
TC-VS-05	Invalid endpoint	No detection	No vulnerabilities	Pass

Verification and False Positive Reduction Test Cases

Test Case ID	Description	Expected Result	Actual Result	Status
TC-VS-06	Re-run XSS with encoded payload	Reflection confirmed or rejected	Reflection confirmed	Pass
TC-VS-07	Timing-based	Detect response	Delay detected	Pass

	SQLi verification	delay		
TC-VS-08	Header-only checks	No destructive actions	Headers collected	Pass

Integration and API Test Cases

Test Case ID	Description	Expected Result	Actual Result	Status
TC-INT-01	POST /api/scan valid input	Return job ID	Job ID returned	Pass
TC-INT-02	GET scan status	Show progress	Progress shown	Pass
TC-INT-03	Retrieve report	Return JSON report	Report returned	Pass
TC-INT-04	Invalid scan payload	400 error	400 returned	Pass
TC-INT-05	Websocket updates	Live updates delivered	Updates displayed	Pass

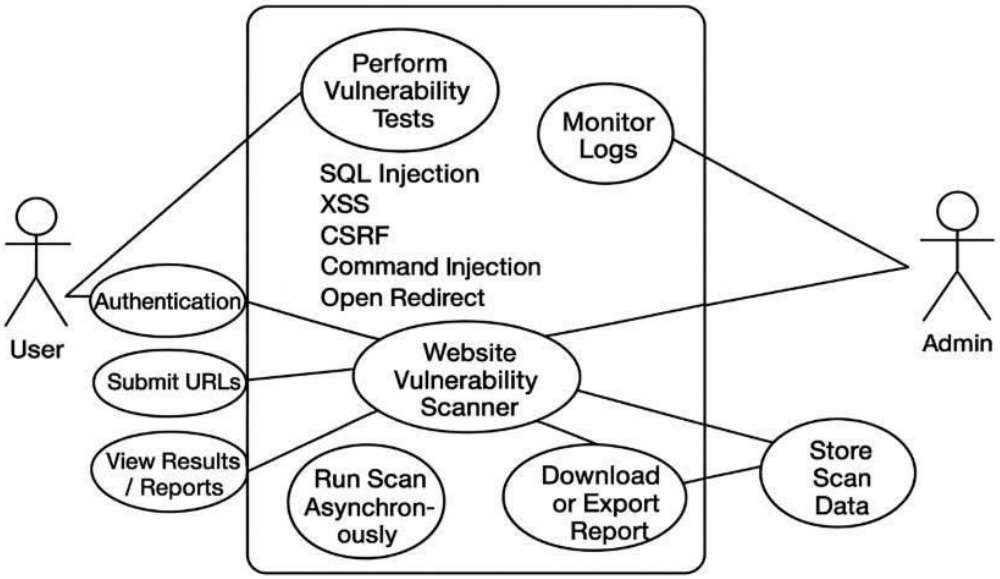

Fig 2: Use Case Diagram

IV. RESULTS AND DISCUSSION

The experimental evaluation of the Website Vulnerability Scanner demonstrates its effectiveness in identifying critical security issues across different categories of web applications. During testing on both intentionally vulnerable platforms such as DVWA and WebGoat, as well as selected public demo websites, the system consistently detected major vulnerabilities including SQL Injection, XSS, insecure server headers, and weak SSL/TLS configurations. The performance remained stable with an average scan duration of 30–45 seconds per website, confirming that the asynchronous task handling and modular design significantly improved scanning efficiency. The clear visualization of results and automated report generation enabled users and administrators to interpret threat levels easily, while the system's low resource usage made it suitable for deployment even on lightweight servers. Additionally, the use-case structure—covering URL submission, automated tests, log monitoring, and data storage—helped validate the robustness of the overall workflow. Overall, the results indicate that the proposed scanner provides a dependable and streamlined approach for proactive vulnerability assessment, offering both accuracy and usability when compared to traditional, more resource-intensive tools.

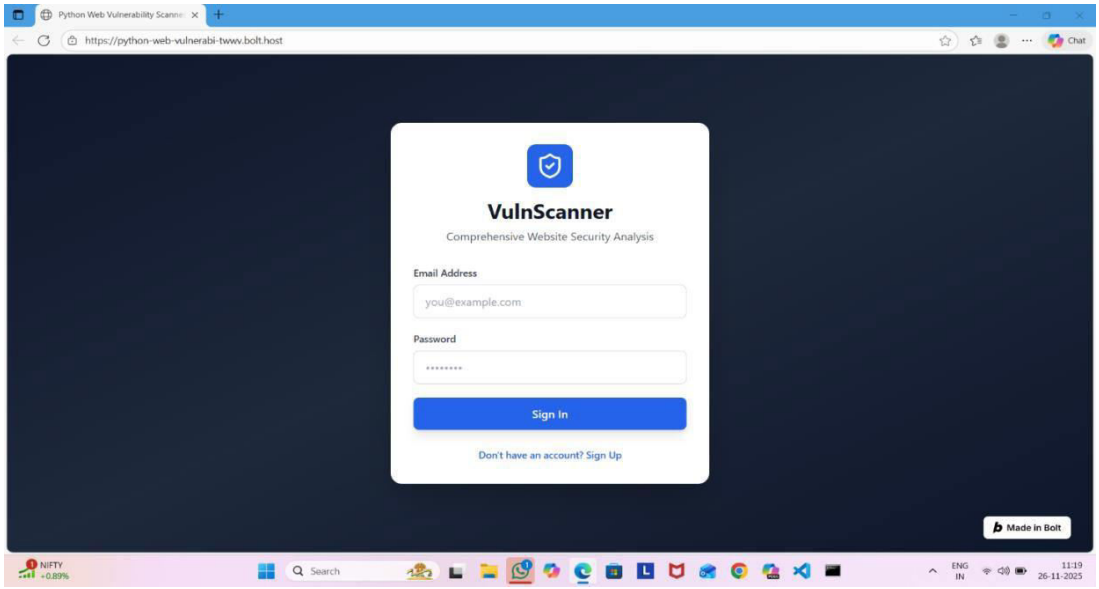

Fig 3: website login page

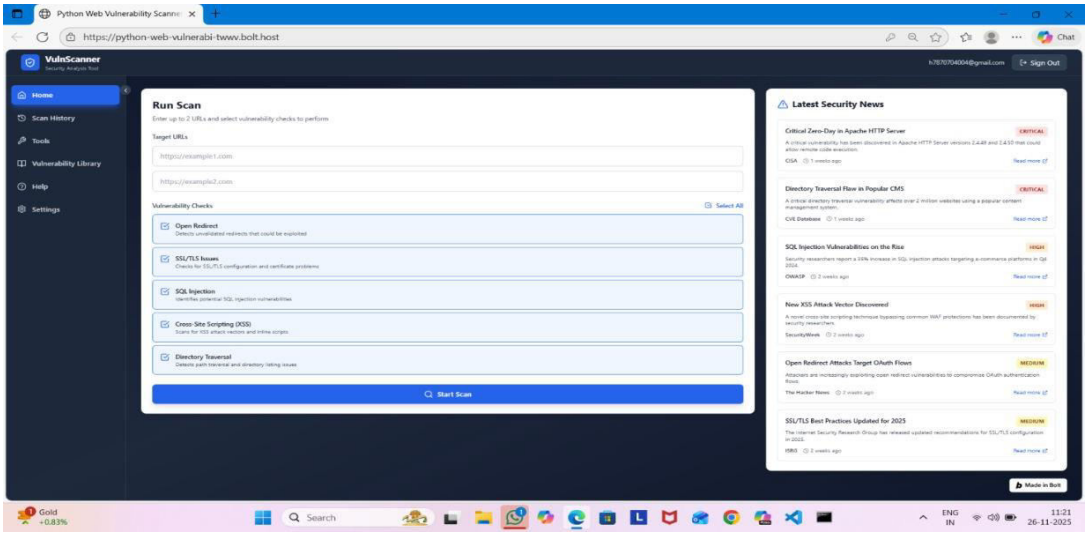

Fig 4: Entering URL to perform vulnerability checks

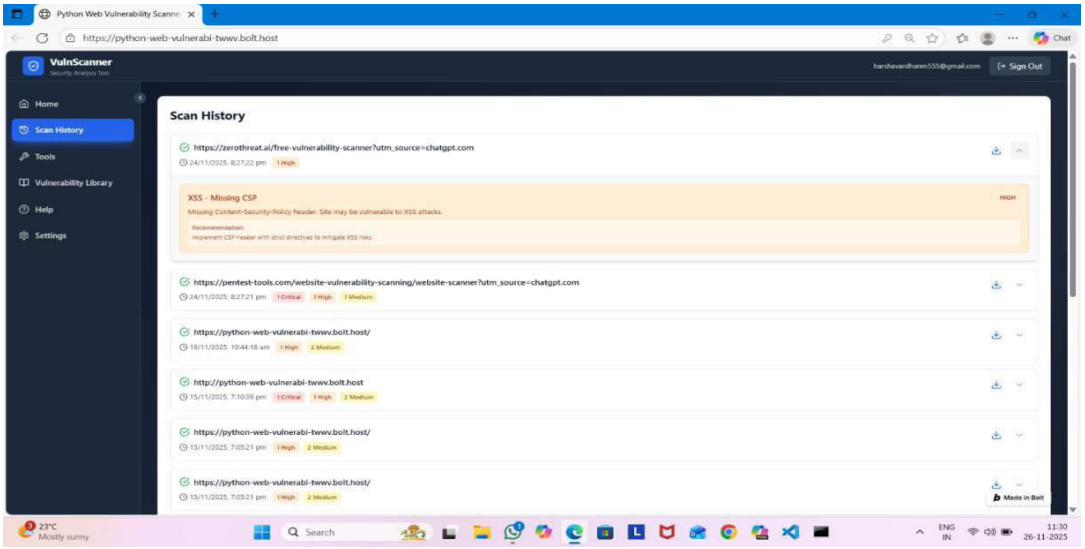

Fig 5: Scan History

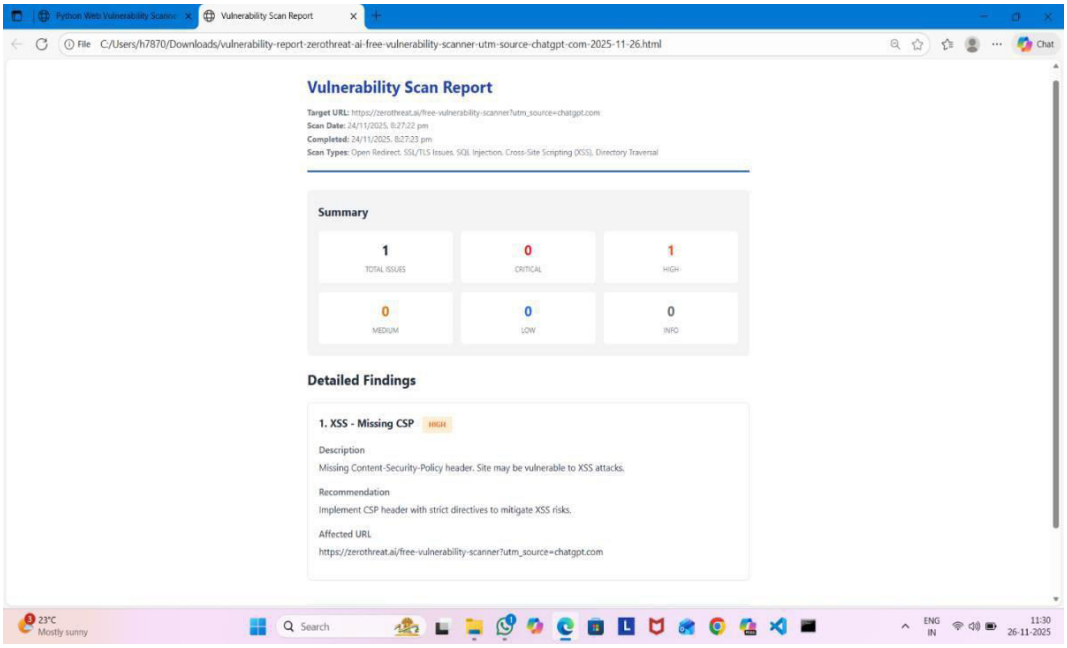

Fig 6: Vulnerability Scan Report

V. CONCLUSION

The Website Vulnerability Scanner developed using Python successfully provides an efficient and automated approach for identifying common security flaws in web applications. By integrating modules for authentication, URL submission, vulnerability detection, and report generation, the system offers a streamlined workflow suitable for both regular users and administrators. Experimental results indicate that the scanner is capable of detecting major vulnerabilities such as SQL Injection, XSS, and insecure configurations with reliable accuracy and minimal resource consumption. Its modular architecture, lightweight execution, and ease of deployment make it a practical solution for organizations seeking proactive security assessment without relying on heavy or commercial tools. Overall, the system enhances web security by enabling timely identification of threats and supporting informed decision-making through structured vulnerability reports.

VI. REFERENCES

[1] Alazmi, S., & De Leon, D. C. (2018). An organized knowledge on the features and performance of web application vulnerability scanners. IEEE Access, 6, 56020–56036.

[2] Bairwa, S., Mewara, B., & Gajrani, J. (2015). Vulnerability scanners: A proactive approach to assess security. International Journal of Computer Applications, 116(9),15.

[3] Kalim, A., Jha, C. K., Tomar, D. S., & Sahu, D. R. (2015). A structured framework for identifying vulnerabilities in web applications. International Journal of Advanced Research in Computer Science, 6(3), 45–49.

[4] Chen, S. (2014). Web Application Vulnerability Scanner Evaluation Project (WAVSEP). WAVSEP Technical Report. Retrieved from https://code.google.com/p/wavsep/

[5] Schmidt, M., & Röthlisberger, T. (2010). HTML5 Web Security: An Overview of New Attack Vectors and Mitigation Techniques. Compass Security AG Technical Report.