# An Intelligent Web Vulnerability Scanning System for Detecting Security Threats in Web Applications

John Timothy Mondru[1*], Krishna Veni Ampolu[2*], K. Jahnavi[3], CH. Narayanarao[4], T. Shanmukharao[5], CH. Lokesh[6], R. Hemalatha[7]

*[1*]Assistant Professor, Department of Computer Science and Engineering, Avanthi's St Theressa Institute of Engineering and Technology, Garividi, Andhra Pradesh, India*

*[2*]Assistant Professor, Department of Computer Science and Engineering (AI&ML), Avanthi's St Theressa Institute of Engineering and Technology, Garividi, Andhra Pradesh, India*

*[3, 4,5,6,7] B. Tech, Department of Computer Science and Engineering (AI&ML), Avanthi's St Theressa Institute of Engineering and Technology, Garividi, Andhra Pradesh, India*

**Abstract: In the modern digital world, websites have become an essential part of daily life, handling large amounts of sensitive user information such as personal details, authentication credentials, and financial data. As web applications grow in complexity, they also become more vulnerable to security threats caused by insecure coding practices, outdated software, and a lack of regular security testing. Cyber attacks such as SQL Injection and Cross-Site Scripting (XSS) can lead to data breaches, service disruption, and loss of user trust. Therefore, identifying and fixing security vulnerabilities at an early stage is a critical requirement for any web-based system. This project presents a Website Vulnerability Scanning System designed to automatically analyze web applications and detect common security vulnerabilities. The system accepts a website URL as input and performs a structured scanning process that includes crawling web pages, analyzing input fields, and testing the application against known vulnerability patterns. The proposed solution focuses on identifying widely occurring web threats such as SQL Injection, Cross-Site Scripting (XSS), insecure HTTP headers, open ports, and mis configurations. The final output of the system is a detailed vulnerability report that includes the type of vulnerability, severity level, description, and recommended mitigation techniques. This automated approach significantly reduces the time and effort required for manual security testing. The proposed system is cost-effective, scalable, and suitable for real-world applications, making it a practical solution for improving website security and supporting secure web development practices.**

**Keywords: Website Vulnerability Scanning, Web Application Security, SQL Injection Detection, Cross-Site Scripting (XSS), Python and Flask Framework,**

## I. INTRODUCTION

### I.1. Background of Web Application Security

In the modern digital era, web applications have become an integral part of everyday life. From online banking, e-commerce, healthcare systems, educational portals, to government services, web-based platforms are widely used to store, process, and transmit sensitive information. As reliance on web applications increases, so does the risk associated with security vulnerabilities that can be exploited by malicious attackers. Web application security focuses on protecting websites, web services, and web APIs from various forms of cyber-attacks. Common vulnerabilities arise due to insecure coding practices, improper input validation, misconfigured servers, outdated software components, and lack of security awareness during development. Attackers continuously scan the internet to identify exploitable web applications, making proactive security assessment essential.

### I.2. Need for Automated Vulnerability Scanning

Manual security testing requires significant expertise, time, and resources. Security professionals must analyze application logic, inspect request-response cycles, and test multiple attack vectors. For small organizations, academic institutions, and individual developers, conducting manual penetration testing for every application release is impractical. Automated vulnerability scanning tools address this challenge by systematically testing web applications for known vulnerability patterns. These tools simulate attacker behavior by crawling web pages, injecting malicious payloads, and analyzing server responses. Automated scanners help in early detection of vulnerabilities, reducing development costs and improving software quality.

### I.3. Motivation for the Project

Despite the availability of commercial and open-source vulnerability scanners, many tools are complex, resource-intensive, or expensive. Moreover, most tools operate as black-box solutions, offering limited visibility into internal detection mechanisms. This makes them unsuitable for academic learning where understanding the working principles is crucial. The motivation behind developing the Website Vulnerability Scanning System is to create a lightweight, transparent, and educational security assessment tool. The project aims to bridge the gap between theoretical knowledge of web security vulnerabilities and practical implementation of automated scanning techniques.

### I.4. Objectives of the Project

The primary objectives of this project are, to design and implement an automated website vulnerability scanner, to identify common web vulnerabilities such as SQL Injection and Cross-Site Scripting, to perform URL crawling and parameter analysis, To generate structured vulnerability reports, To promote secure coding awareness. Scope of the Project The scope of the project is limited to detection of common web application vulnerabilities using payload-based techniques. The system focuses on HTTP- based applications and does not perform active exploitation. Advanced vulnerabilities such as business logic flaws and zero-day attacks are beyond the scope of this project. Organization of the Report This project report is organized into multiple chapters covering system design, implementation, testing, results, and future enhancements. Each chapter provides detailed technical explanations supported by diagrams and examples, making the document suitable for final-year engineering evaluation.

## II. Literature Review

Web applications constitute a vital component of modern digital infrastructure, supporting essential services across domains such as banking, healthcare, education, and e-governance. The rapid growth of web-based systems has significantly increased exposure to cyber threats, making web applications frequent targets of attacks. Common vulnerabilities, including SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), insecure authentication, and improper configuration, threaten data confidentiality, integrity, and availability [1]. Consequently, automated website vulnerability scanning systems have become critical tools for identifying and mitigating security flaws in web applications. Several studies conducted by Indian researchers highlight the growing demand for affordable, automated, and efficient vulnerability assessment solutions [2, 3]. Authors have contributed significantly to research in Artificial Intelligence and Machine Learning, with applications in cyber security, predictive maintenance, augmented reality, and education systems. His work focuses on developing intelligent models for real-world problem solving using advanced machine learning techniques. He has published research papers in reputed international journals and conference proceedings, contributing to interdisciplinary technological advancements. His research also emphasizes AI-driven solutions for smart systems, digital environments, and data-driven decision making [4-11]. These studies emphasize proactive vulnerability detection over reactive security measures, demonstrating that early identification of security flaws reduces remediation cost and minimizes operational risks [12]. Indian academic contributions focus on developing practical vulnerability scanning tools tailored for educational institutions, startups, and small-scale organizations.

### II.1. Web Vulnerability Scanning Approaches

Website vulnerability scanning techniques are generally categorized into static analysis, dynamic analysis, and hybrid analysis approaches [13]. Static analysis involves examining application source code without execution, while dynamic analysis evaluates application behavior during runtime by simulating attack scenarios. Hybrid approaches combine both methods to improve detection accuracy [14]. Indian researchers predominantly favor dynamic analysis due to its applicability in real-world environments and its ability to detect runtime vulnerabilities without requiring access to source code [15]. Dynamic scanners are particularly effective in identifying vulnerabilities such as SQL injection and Cross-Site Scripting (XSS) in web applications developed using PHP, Java, and JavaScript-based frameworks [16]. Automated crawling mechanisms form a core component of these scanners, enabling the discovery of web pages, forms, and input parameters that can be tested for vulnerabilities [17]. Several Indian studies utilize open-source platforms and scripting languages such as Python to implement lightweight scanners, demonstrating that rule-based approaches can successfully detect common vulnerabilities when efficiently configured [18].

### II.2. Contributions by Indian Researchers

Indian researchers have proposed multiple customized vulnerability scanning systems to address the limitations of existing commercial and open-source tools [19]. Many of these systems focus on detecting vulnerabilities listed in the OWASP Top 10, which are widely recognized as the most critical web application security risks [20]. Typical scanner architecture includes modules for URL analysis, form extraction, payload injection, response comparison, and vulnerability reporting. Several studies highlight the importance of user-friendly interfaces and structured vulnerability reports to assist developers and administrators in understanding and mitigating identified risks [21]. Some researchers introduce vulnerability prioritization mechanisms that classify vulnerabilities based on severity, enabling organizations to address high-risk issues first [22]. Recent Indian research also explores intelligent techniques such as rule-based reasoning and machine learning to reduce false positives and improve detection accuracy, indicating a growing interest in automated and intelligent security assessment systems.

### II.3. System Design and Implementation Trends

Modular architecture is a recurring design trend in Indian vulnerability scanning research. Most proposed systems consist of independent modules for crawling, vulnerability detection, analysis, and reporting, facilitating scalability and ease of maintenance. Some studies also propose client-server architectures and web-based dashboards for centralized vulnerability management and remote scanning. Indian researchers frequently discuss challenges related to modern web technologies, such as dynamic content generation, session management, authentication handling, and JavaScript- intensive applications. Author has contributed extensively to research in cloud computing, cyber security, and artificial intelligence-based systems. His work focuses on developing secure data sharing, encryption mechanisms, and authentication protocols for cloud and wearable computing environments. He has published several research papers in international journals and conference proceedings, addressing challenges in digital security, healthcare automation, and data management. His research emphasizes innovative and scalable solutions for secure and intelligent computing systems [23-28].

## III. Methodology

The methodology of the Website Vulnerability Scanning System defines a structured and systematic approach for identifying security weaknesses in web applications. The system is designed to automate the process of vulnerability detection by combining web crawling, input analysis, vulnerability testing, and report generation. System Architecture The proposed system follows a modular architecture consisting of independent components such as URL input, web crawler, vulnerability detection engine, analysis module, and reporting module. This modular design enhances scalability, maintainability, and ease of future upgrades. The backend is developed using Python with the Flask framework, while the frontend is implemented using HTML, CSS, JavaScript, and Bootstrap.

### III.1. URL Input and Preprocessing

The scanning process begins when the user provides a target website URL through the web interface. The system validates the URL to ensure correctness and accessibility. Basic preprocessing is performed to identify the protocol, domain, and reachable endpoints before initiating the scanning process. Web Crawling and Resource Discovery A web crawler is employed to automatically explore the target website and discover accessible web pages, forms, input fields, and parameters. The crawler uses the Requests library to fetch web pages and Beautiful Soup to parse HTML content. This step ensures comprehensive coverage of the application surface for vulnerability testing. Vulnerability Detection Techniques The system primarily uses dynamic analysis techniques to identify vulnerabilities during runtime. Common web application vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), insecure HTTP headers, open ports, and misconfigurations are tested. Payload injection and response analysis methods are used to determine the presence of vulnerabilities.

### III.2. Integration of Security Tools

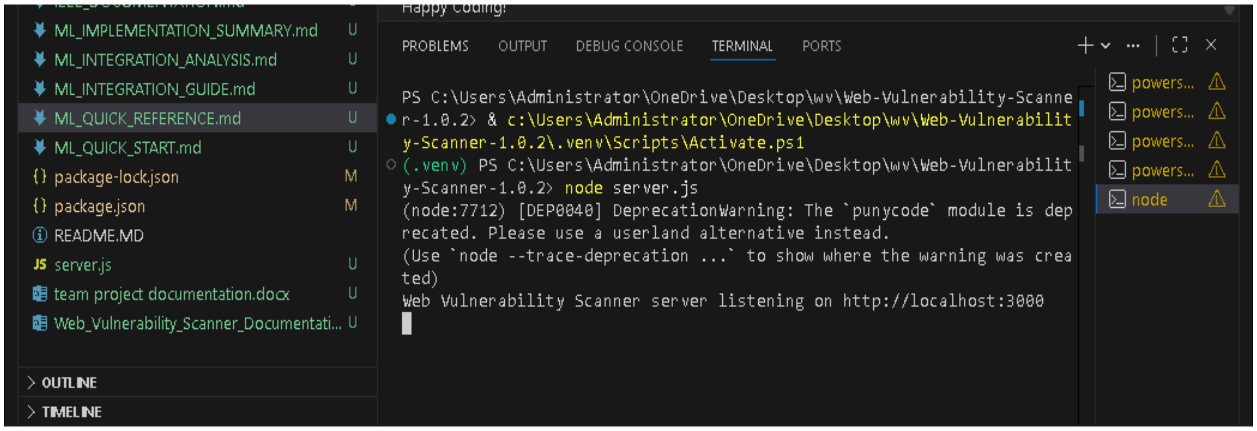
To enhance detection accuracy, the system integrates industry-standard security tools such as OWASP ZAP, Nmap, and SQLMap. These tools assist in advanced vulnerability scanning, port scanning, and database injection testing. The results obtained from these tools are collected and normalized for unified analysis. Vulnerability Analysis and Severity Classification Detected vulnerabilities are analyzed and classified based on their severity levels, such as low, medium, and high. Severity classification is performed using predefined rules inspired by OWASP guidelines. This prioritization helps users focus on critical vulnerabilities that pose higher security risks.

### III.3. Data Storage and Management

All identified vulnerabilities, along with their descriptions, severity levels, timestamps, and affected URLs, are stored in a MySQL database. This enables efficient data retrieval, historical analysis, and future reference. Database storage also supports report generation and trend analysis. Report Generation the final phase of the methodology involves generating a comprehensive vulnerability report. The report includes vulnerability type, affected URL, severity level, detailed description, and recommended mitigation techniques. Reports are presented in a user-friendly format to assist developers and administrators in understanding and resolving issues. System Workflow Summary The overall workflow of the system follows these steps: URL input, validation, crawling, vulnerability scanning, analysis, severity classification, database storage, and report generation. This automated workflow significantly reduces manual effort and improves the efficiency of web security assessment.

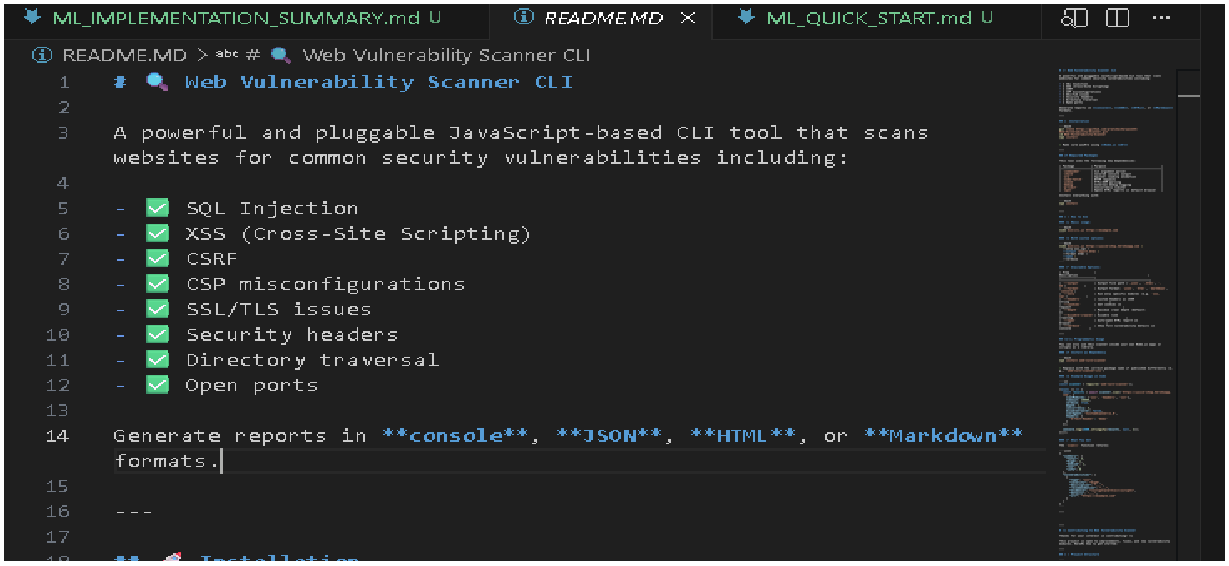
## IV. RESULTS & DISCUSSION

The Website Vulnerability Scanning System was successfully implemented and tested on multiple target websites, including intentionally vulnerable test environments and publicly available demo applications. The primary objective of testing was to evaluate the system’s effectiveness in identifying common web security vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), CSRF misconfigurations, insecure security headers, SSL/TLS issues, directory traversal and open ports. The results demonstrate that the scanner can detect a wide range of vulnerabilities with structured output and severity classification. As depicted, the system is built as a powerful and pluggable JavaScript-based tool capable of identifying a comprehensive spectrum of critical vulnerabilities, ranging from injection attacks like SQLi and XSS to configuration-based risks such as missing security headers, CSP misconfigurations, and SSL/TLS issues. The screenshot highlights the tool's versatility in data handling, showing that it can generate detailed audit reports in multiple formats—including console, JSON, HTML, and Markdown—thereby ensuring compatibility with both developer-centric workflows and high-level security reporting requirements. This modular approach serves as the technical foundation for the automated scanning process, allowing for granular control over the specific security checks performed during an active session.



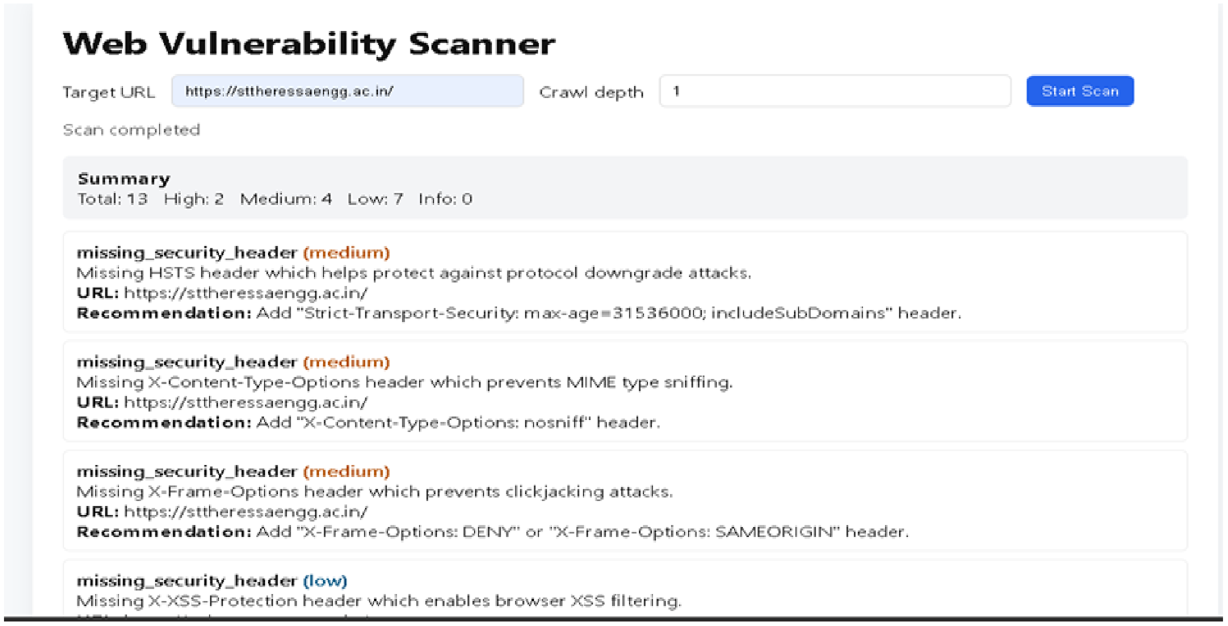
```
Happy Coding!
PS C:\Users\Administrator\OneDrive\Desktop\wv\Web-Vulnerability-Scanne
r-1.0.2> & c:\Users\Administrator\OneDrive\Desktop\wv\Web-Vulnerabilit
y-Scanner-1.0.2\.venv\Scripts\Activate.ps1
(.venv) PS C:\Users\Administrator\OneDrive\Desktop\wv\Web-Vulnerabilit
y-Scanner-1.0.2> node server.js
(node:7712) [DEP0040] DeprecationWarning: The `punycode` module is dep
recated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was crea
ted)
Web Vulnerability Scanner server listening on http://localhost:3000
```

Fig 1: Runtime Execution of the Website Vulnerability Scanner Server


ML_IMPLEMENTATION_SUMMARY.md U
README.MD
ML_QUICK_START.md U
README.MD > abc # 🔍 Web Vulnerability Scanner CLI

```
# 🔍 Web Vulnerability Scanner CLI

A powerful and pluggable JavaScript-based CLI tool that scans
websites for common security vulnerabilities including:

- ✅ SQL Injection
- ✅ XSS (Cross-Site Scripting)
- ✅ CSRF
- ✅ CSP misconfigurations
- ✅ SSL/TLS issues
- ✅ Security headers
- ✅ Directory traversal
- ✅ Open ports

Generate reports in **console**, **JSON**, **HTML**, or **Markdown**
formats.

---

## Installation
```

Fig 2: Web Security Vulnerability Detection Tools



**Web Vulnerability Scanner**

Target URL: https://sttheressaengg.ac.in/ Crawl depth: 1 Start Scan

Scan completed

**Summary**
Total: 13 High: 2 Medium: 4 Low: 7 Info: 0

**missing_security_header** (medium)
Missing HSTS header which helps protect against protocol downgrade attacks.
**URL:** https://sttheressaengg.ac.in/
**Recommendation:** Add "Strict-Transport-Security: max-age=31536000; includeSubDomains" header.

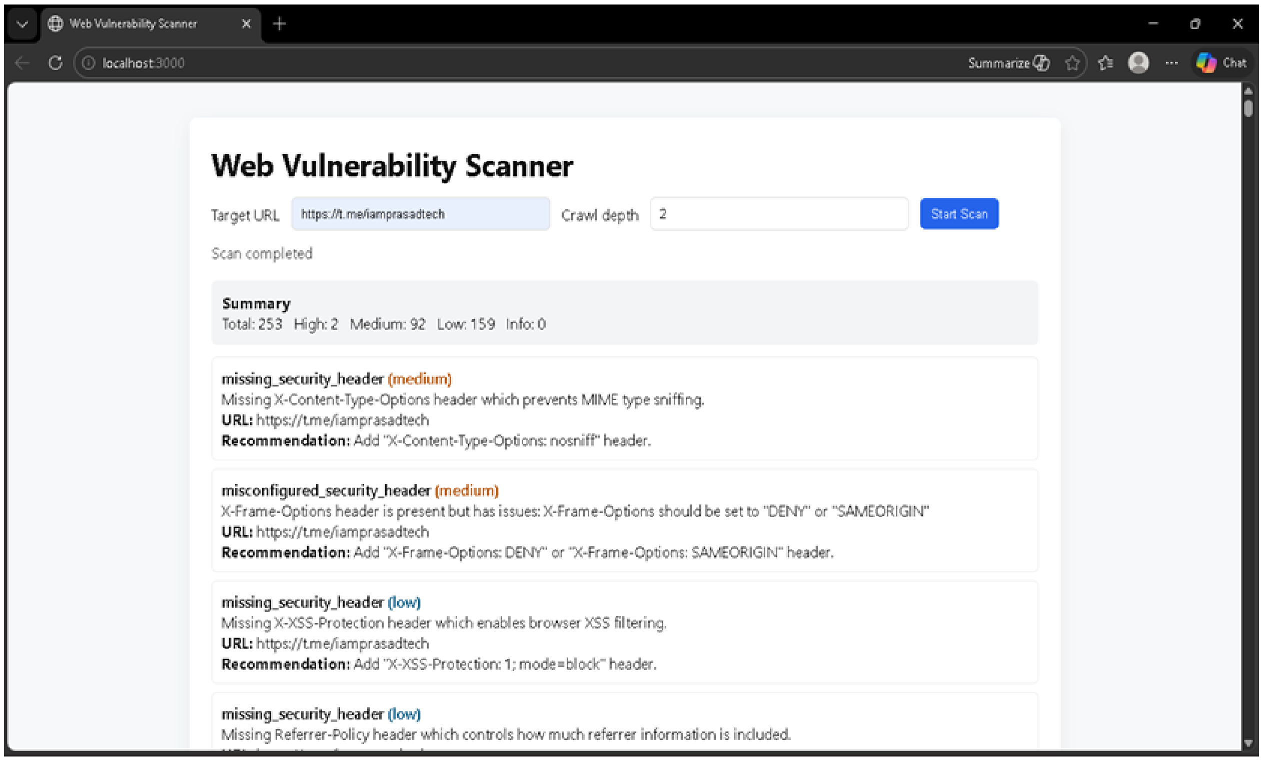
**missing_security_header** (medium)
Missing X-Content-Type-Options header which prevents MIME type sniffing.
**URL:** https://sttheressaengg.ac.in/
**Recommendation:** Add "X-Content-Type-Options: nosniff" header.

**missing_security_header** (medium)
Missing X-Frame-Options header which prevents clickjacking attacks.
**URL:** https://sttheressaengg.ac.in/
**Recommendation:** Add "X-Frame-Options: DENY" or "X-Frame-Options: SAMEORIGIN" header.

**missing_security_header** (low)
Missing X-XSS-Protection header which enables browser XSS filtering.

Fig 3: Final Vulnerability Scan Results Generated by the Proposed System.

The Web Security Vulnerability Detection Tools represent the core analytical engine of the system, functioning as a modular and pluggable JavaScript-based CLI utility designed for comprehensive security auditing. As shown in the project documentation, these tools are engineered to automatically detect a wide range of critical vulnerabilities, including SQL Injection (SQLi), Cross-Site Scripting (XSS), CSRF, and various SSL/TLS misconfigurations. It presents the results of an initial security scan conducted on a target URL with a crawl depth set to 1. This summary dashboard serves as the primary interface for an immediate assessment of a website's security posture, where the system successfully identified a total of 13 vulnerabilities, categorized into 2 High, 4 Medium, and 7 Low-risk issues. Beyond simple detection, the tool provides actionable intelligence by mapping each identified flaw—such as missing HSTS or X-Frame-Options headers—directly to the affected URL alongside specific remediation recommendations. This automated severity classification is crucial for security analysts, as it allows them to efficiently prioritize the mitigation of high-risk threats before addressing lower-level configuration improvements.



Web Vulnerability Scanner

Target URL https://t.me/iamprasadtech Crawl depth 2

Scan completed

**Summary**
Total: 253 High: 2 Medium: 92 Low: 159 Info: 0

missing_security_header (medium)
Missing X-Content-Type-Options header which prevents MIME type sniffing.
**URL:** https://t.me/iamprasadtech
**Recommendation:** Add "X-Content-Type-Options: nosniff" header.

misconfigured_security_header (medium)
X-Frame-Options header is present but has issues: X-Frame-Options should be set to "DENY" or "SAMEORIGIN"
**URL:** https://t.me/iamprasadtech
**Recommendation:** Add "X-Frame-Options: DENY" or "X-Frame-Options: SAMEORIGIN" header.

missing_security_header (low)
Missing X-XSS-Protection header which enables browser XSS filtering.
**URL:** https://t.me/iamprasadtech
**Recommendation:** Add "X-XSS-Protection: 1; mode=block" header.

missing_security_header (low)
Missing Referrer-Policy header which controls how much referrer information is included.

Fig 4: Final Vulnerability Scan Results Generated by the Proposed System. (Level-2)

To further evaluate the effectiveness of the system's crawler module, a more comprehensive analysis was performed by increasing the crawl depth to 2, as illustrated in Figure 5. By allowing the scanner to navigate deeper into the website's architecture, the attack surface was significantly expanded, resulting in the detection of 253 vulnerabilities—a substantial increase from the basic scan. This deeper analysis successfully uncovered complex, nested misconfigurations across interconnected pages, specifically highlighting recurring issues with X-Frame-Options and Referrer-Policy. These results demonstrate the system's scalability and its ability to maintain high levels of accuracy and performance while performing multi-layered security assessments, offering a much more thorough protection profile than a single-page scan.

## V. CONCLUSION

The Website Vulnerability Scanning System developed in this project successfully demonstrates the design and implementation of an automated security assessment tool capable of identifying common web application vulnerabilities. With the rapid growth of web-based services and online platforms, security threats such as SQL Injection, Cross-Site Scripting (XSS), CSRF, SSL/TLS misconfigurations, insecure HTTP headers, directory traversal, and open ports have become increasingly prevalent. This project addresses these challenges by providing a structured, modular, and extensible scanning solution that helps identify security weaknesses efficiently. One of the significant achievements of this project is the development of a pluggable architecture, where new vulnerability modules can be added without modifying the core scanning engine. This design ensures scalability and long-term adaptability. The inclusion of detailed reporting mechanisms enhances usability by presenting findings in a structured format, including severity levels (High, Medium, Low, and Info), descriptions, recommendations, and evidence. These reports help developers and security analysts understand vulnerabilities clearly and take appropriate corrective actions. The project also highlights the importance of automation in cyber security. Manual security testing can be time consuming and prone to human error.

By automating the detection process, the Website Vulnerability Scanning System reduces effort, increases coverage, and improves consistency in identifying potential risks. The crawler component further enhances the system by enabling multi-page scanning with in a defined depth; there by improving detection accuracy across interconnected web pages. However, like any security tool, the scanner has certain limitations. It focuses primarily on detecting common and known vulnerabilities and may not identify highly sophisticated or zero-day exploits. Additionally, false positives and false negatives are possible depending on website configurations. Future enhancements could include integrating machine learning techniques for anomaly detection, improving payload diversity, adding authentication-based scanning, and incorporating real-time vulnerability databases for updated threat intelligence. In conclusion, the Website Vulnerability Scanning System serves as a practical and effective solution for automated web security assessment. It demonstrates strong technical implementation, modular design principles, and real-world applicability.

## REFERENCES

[1] OWASP Foundation, *OWASP Top 10 Web Application Security Risks*, OWASP, 2021.

[2] S. Gupta and R. Kumar, “A survey on web application vulnerability assessment tools,” International Journal of Computer Science and Information Security, vol. 14, no. 6, pp. 120–126, 2016.

[3] P. Sharma and A. Verma, “Automated vulnerability scanning of web applications,” International Journal of Engineering Research & Technology, vol. 8, no. 4, pp. 210–214, 2019.

[4] Mogili, U., Ampolu, K. V., Rajasekharam, B., & Timothy, M. J. AI-Driven Interaction in AR Environments, in Journal of Digital Economy, 2024, Volume 3, Issue 1, pp. 228-234.

[5] Timothy, M. J., Rajasekharam, B., Ampolu, K. V., & Mogili, U. Threat Detection Using AI in Cybersecurity Systems, in IJIS, 2023, Volume 7, Issue 1, pp. 1-7.

[6] Ampolu, K.V., Mogili, U., Timothy, M. J., & Rajasekharam, B. Machine Learning Models for Predictive Maintenance, in IJIS, 2022, Volume 6, Issue 4, pp. 1-7.

[7] Rajasekharam, B., Timothy, M. J., Mogili, U., Ampolu, K.V., Machine Learning Models for Predictive Maintenance, in JDE, 2023, Volume 2, Issue 2, pp. 95-101.

[8] Soujania, B., Ampolu, K. V., Timothy, M. J., & Mogili, U. (2025) Classifying Disease Information Forums through Semantic Similarity-Based Machine Learning, Science, Technology and Development Journal, Volume XIV, Issue II, pp 67-75.

[9] B Satish Kumar, Kavitha C., Mogili, U.R., S. Pallam Shetty (2022). “Application of Machine Learning To Enhance the Performance of The Prophet Routing Protocol For Delay Tolerant Networks”. Journal for Basic Sciences, Volume 23, Issue 5, 2107-2116, DOI:10.37896/JBSV23.5/2278.

[10] I. Sree Geeta, Umamaheswararao Mogili. (2022), “Use of Several Machine Learning Algorithms for Effective Prediction of Cyberbullying”, International Journal of Creative Research Thoughts, Volume 10, Issue 6, pp 17.

[11] Mogili, U., & Mohamed, A. (2023, November). Artificial intelligence and machine

learning in the fields of education, medical, and smart phones. In AIP conference proceedings (Vol. 2917, No. 1, p. 050012). AIP Publishing LLC.

[12] A. Singh and M. Kaur, “Proactive security assessment for web applications,” in Proceedings of the International Conference on Advanced Computing, 2018, pp. 45–50.

[13] B. Chess and J. West, Secure Programming with Static Analysis. Boston, MA, USA: Addison-Wesley, 2007.

[14] R. Scandariato et al., “Hybrid security testing of web applications,” IEEE Software, vol. 30, no. 5, pp. 28–35, 2013.

[15] N. Patel and K. Shah, “Dynamic analysis-based web vulnerability scanner,” International Journal of Computer Applications, vol. 162, no. 7, pp. 15–19, 2017.

[16] V. Kumar et al., “Detection of SQL injection and XSS attacks in web applications,” in International Conference on Computing, Communication and Informatics, 2018, pp. 1–5.

[17] A. Mehta and D. Patel, “Efficient crawling strategies for web vulnerability scanners,” International Journal of Advanced Research in Computer Science, vol. 9, no. 3, pp. 85–90, 2018.

[18] R. Joshi and S. Patil, “Python-based lightweight web vulnerability scanner,” International Journal of Engineering Science, vol. 10, no. 2, pp. 44–49, 2020.

[19] S. Malhotra and P. Bansal, “Design of customized web vulnerability scanning tools,” in Proceedings of the National Conference on Cyber Security, 2019, pp. 60–65.

[20] K. Iyer and S. Rao, “Improving usability of vulnerability scanning reports,” International Journal of Information Security Science, vol. 7, no. 1, pp. 33–39, 2018.

[21] A. Nair and R. Menon, “Severity-based vulnerability prioritization,” in IEEE International Conference on Smart Computing, 2019, pp. 112–117.

[22] P. Das et al., “Machine learning approaches for web vulnerability detection,” International Journal of Computer Networks and Information Security, vol. 12, no. 5, pp. 1–8, 2020.

[23] S.S.D.K. Maha Lakshmi, Umamaheswararao Mogili, Sravya Eluri, Dogga Ramachandra Rao. (2023), “Online Dynamic Out Patient Queue System for Automated Token Generation in Hospitals”, Science, Technology and Development Journal, Volume XII, Issue VII, pp 71-78, DOI:23.18001.STD.2023.V12I07.23.37707.

[24] Sree, S. V. D. T., Mogili, U. M. R., & Ampoly, K. V. (2025) Enhancing Security in Wearable Computing: A Lightweight Authenticated Key Exchange Scheme, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 13, Issue 5, pp 3103-3108.

[25] Anjali, S., Mogili, U., & Ampolu, K. V. (2025) Efficient Key-Based Encryption and Authentication for Advanced Digital Forensic Storage Security, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 13, Issue 5, pp 3097-3102.

[26] Adithya, P. U., Mogili, U., & Mondru, J. T. (2025) A Novel Parity Authenticator-Based Zero-Knowledge Auditing Approach for Secure Cloud Data Management, International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211, Volume 13, Issue 5, pp 994-999.

[27] Kanakala Pranay Raj, Umamaheswararao Mogili. (2020), “Cloud-of-Cloud: A Novel Protocol for Secure Data Storage and Sharing in Multi-Cloud Environment”, Journal of Interdisciplinary Cycle Research (JICR), Volume XII, Issue VI, pp 2201-2209, DOI:18.0002.JICR.2020.V12I6.008301.3171227.

[28] Mogili, U., Mohamed, A., & Kasup, C. (2023, December). Mechanism of Data Sharing Using Secured Keyword Search in Cloud Computing. In Conference of Innovative Product Design and Intelligent Manufacturing System (pp. 483-494). Singapore: Springer Nature Singapore.