Risk-Driven Vulnerability Management Dashboard

Mr. P.V.S.N Murthy (Guide), Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

B Navya, Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

D Sukanya, Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

A Chinnarao, Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

P Pavan Kumar, Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

S Sai Dinesh, Computer science & Engineering Department, Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India.

ABSTRACT

The rapid growth of web applications has significantly increased exposure to cybersecurity threats such as SQL Injection, Cross-Site Scripting (XSS), and security misconfigurations. Traditional vulnerability assessment techniques are often manual, inefficient, and lack effective prioritization mechanisms. This paper presents a Risk-Driven Vulnerability Management Dashboard, an automated system designed to detect, analyze, and manage vulnerabilities in web applications.

The proposed system integrates automated scanning techniques with risk-based prioritization using standardized frameworks such as CVE and CVSS. The system categorizes vulnerabilities into severity levels and provides an interactive dashboard forvisualization using charts and metrics. The backend is developed using Flask, and results are stored in a structured database.

Experimental results demonstrate improved efficiency, accuracy, and usability compared to traditional methods. The system enables proactive threat management and supports informed decision- making, contributing to enhanced cybersecurity practices.

KEYWORDS:

Vulnerability Scanning, Cybersecurity, CVSS, CVE, Risk Analysis, Web Security, Dashboard Visualization.

INTRODUCTION

Web applications have become a fundamental component of modern digital infrastructure, supporting critical domains such as banking, e- commerce, healthcare, education, and government services. Their widespread adoption and increasing complexity have significantly expanded the attack surface, making them prime targets for cyber adversaries. As organizations continue to digitize their operations, the frequency and sophistication of cyberattacks have also grown. Common vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), broken authentication, and security misconfigurations pose serious risks to application integrity and user data. These vulnerabilities can be exploited to gain unauthorized access, manipulate sensitive information, or disrupt services, ultimately leading to severe consequences including data breaches, financial losses, reputational damage, and legal implications.

Traditional vulnerability management approaches often rely on manual testing or standalone scanning tools that primarily focus on identifying security issues without providing effective mechanisms for

prioritization and decision-making. These methods are time-consuming, error-prone, and inefficient in handling large-scale applications with numerous vulnerabilities. Moreover, not all vulnerabilities carry the same level of risk, and treating them equally can result in delayed mitigation of critical threats.

To address these challenges, the proposed system introduces an automated and risk-driven approach to vulnerability detection and management. The system continuously scans web applications to identify potential security weaknesses using automated techniques such as input validation testing, URL crawling, and response analysis. Once vulnerabilities are detected, they are systematically categorized based on severity levels—such as critical, high, medium, and low—allowing security teams to focus on the most impactful threats first. This prioritization significantly improves response time and resource allocation, ensuring that high-risk vulnerabilities are mitigated promptly.

A key feature of the system is the integration of standardized frameworks such as Common Vulnerabilities and Exposures (CVE) and Common Vulnerability Scoring System (CVSS). These frameworks provide a structured and industry-recognized methodology for identifying and evaluating vulnerabilities. CVE ensures consistent identification of known vulnerabilities, while CVSS assigns a quantitative score based on factors such as exploitability, impact, and environmental context. By leveraging these frameworks, the system delivers accurate and reliable risk assessments that align with global cybersecurity standards.

Furthermore, the system incorporates an interactive dashboard visualization module, which plays a crucial role in enhancing user experience and decision-making. The dashboard presents vulnerability data through intuitive visual elements such as charts, graphs, and key performance indicators (KPIs), enabling users to quickly understand the security posture of the application. This visual representation simplifies complex data analysis, supports real-time monitoring, and facilitates proactive threat management.

Overall, the proposed solution combines automation, standardized risk assessment, and advanced visualization to provide a comprehensive and efficient vulnerability management system. It empowers organizations to strengthen their cybersecurity defenses, reduce potential risks, and maintain a secure and resilient digital environment.

LITERATURE SURVEY

The increasing number of cyber threats targeting web applications has led to extensive research in vulnerability detection and security management. Automated vulnerability scanning tools are widely used to identify common security issues such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and misconfigurations. These tools use techniques like input validation testing, URL crawling, and response analysis to improve detection efficiency and reduce manual effort. Additionally, standardized frameworks such as Common Vulnerabilities and Exposures (CVE) and Common Vulnerability Scoring System (CVSS) are commonly used for risk assessment. CVE provides a structured identification of known vulnerabilities, while CVSS assigns severity scores based on factors like exploitability and impact. However, most existing systems rely on static scoring and lack contextual awareness, making it difficult to prioritize vulnerabilities effectively in real-world environments.

Recent studies also emphasize the importance of integrating visualization and automation in cybersecurity systems. Dashboard-based visualization techniques use charts, graphs, and key performance indicators (KPIs) to present complex vulnerability data in a simplified manner, improving user understanding and decision-making. While some systems combine detection and reporting, they often lack real-time updates, efficient prioritization, and user-friendly interfaces. Furthermore, traditional security mechanisms such

as secure coding practices and firewalls are mostly preventive and require integration with detection systems for complete protection. Therefore, there is a need for a unified solution that integrates automated scanning, risk-based prioritization, and interactive visualization. The proposed Risk-Driven Vulnerability Management Dashboard addresses this gap by providing a comprehensive and efficient approach to managing web application security.

IMPLEMENTATION STUDY EXISTING SYSTEM

The existing system is based on traditional Vulnerability Risk Management (VRM) approaches used to identify, evaluate, and remediate security vulnerabilities in IT systems. These systems mainly rely on vulnerability scanning tools such as Nessus and OpenVAS to detect issues in web applications and networks. While these tools are effective in identifying vulnerabilities, they provide only basic information such as severity scores. The evaluation and prioritization of vulnerabilities are largely performed manually by security experts, making the process time-consuming, costly, and dependent on human expertise. Moreover, most existing systems depend on standard scoring methods like CVSS to prioritize vulnerabilities. However, these scores are static and do not consider real-time factors such as organizational context, asset importance, or exposure level. Existing systems also lack proper integration of multiple data sources and do not provide effective visualization or automation. As a result, organizations face challenges in managing a large number of vulnerabilities efficiently, often leaving critical issues unresolved and increasing the risk of cyber- attacks.

PROPOSED SYSTEM

RiskShield System Workflow

The *Risk-Driven Vulnerability Management Dashboard (RiskShield)* addresses the limitations of traditional vulnerability management systems by implementing a layered architecture consisting of a web-based interface, backend processing modules, and a database-driven dashboard. The system operates as follows:

1. User Input and Scan Initiation:

The user accesses the web dashboard and enters a target URL for vulnerability scanning. The system validates the input and sends the request to the Flask backend server. The server initializes the scanning process and prepares the required modules for analysis.

2. Vulnerability Scanning Module:

The scanning module analyzes the target web application using HTTP requests and parsing techniques. It checks for vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and security misconfigurations. The module uses libraries like Requests and BeautifulSoup to extract and analyze web content. Detected vulnerabilities are collected and passed to the next stage.

3. Risk Analysis and Classification:

The detected vulnerabilities are evaluated using a risk-driven approach. The system assigns severity levels such as critical, high, medium, and low based on CVE and CVSS scoring standards. This helps in prioritizing vulnerabilities based on their impact and exploitability.

4. Data Storage and Management:

All vulnerability data, scan results, and analysis information are stored in the SQLite database. The system maintains structured records including vulnerability type, severity, and timestamp. This enables efficient retrieval, tracking, and historical analysis of vulnerabilities.

5. Dashboard Visualization:

The processed data is displayed on an interactive dashboard. The dashboard presents vulnerability statistics using charts, graphs, and key performance indicators. It provides users with a clear

understanding of the security status of the web application.

6. Report Generation:

The system generates detailed reports containing vulnerability information, severity levels, and recommendations. Reports can be downloaded in formats such as PDF or CSV, allowing users to maintain documentation and perform further analysis.

7. Input Validation and Error Handling:

The system validates all user inputs to ensure correct URL format and prevent invalid data processing. Proper error messages are displayed for incorrect inputs, ensuring system reliability and security.

8. System Monitoring and Logging:

The system maintains logs of all scanning activities, including timestamps and results. These logs help in tracking system performance, auditing, and debugging.

Fig:1 RiskShield Workflow

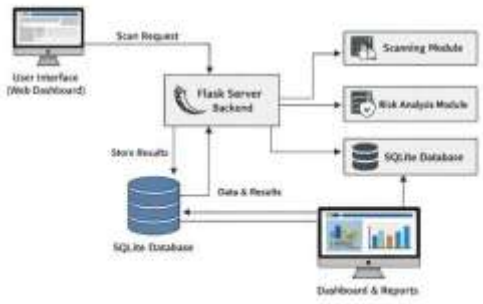

Fig:2 RiskShield System Architecture

SOFTWARES AND LIBRARIES DESCRIPTION

The system is implemented in Python and uses a modular repository structure. The major technologies and libraries are:

PROGRAMMING LANGUAGE

Python 3.8+ is used for its flexibility, extensive ecosystem, and suitability for cybersecurity applications.

CORE LIBRARIES AND TOOLS

CORE LIBRARIES AND TOOLS

Component : Libraries/Tools

Server (Backend) : Flask, SQLite, Requests, BeautifulSoup, JSON

Frontend (Dashboard) : HTML, CSS, JavaScript, Chart.js

Vulnerability Scanning : Requests, BeautifulSoup, re (Regular Expressions)

Risk Analysis : CVE, CVSS (Scoring Logic), Python

Database : SQLite

Reporting : ReportLab (PDF generation), CSV

Data Handling : Pandas (optional), NumPy (optional)

Utilities : Logging, OS, URLlib

KEY LIBRARIES EXPLAINED

Flask – Lightweight Python web framework used to build the backend and handle user requests, routing, and server operations.

Requests – Python library used to send HTTP requests to web applications and retrieve responses for vulnerability testing.

BeautifulSoup – Used for parsing HTML content and extracting links, forms, and page data during scanning.

SQLite – Lightweight database used for storing vulnerability results, logs, and scan history.

Chart.js – JavaScript library used to create interactive charts and graphs for dashboard visualization.

ReportLab – Python library used to generate PDF reports containing vulnerability details and analysis.

CONCLUSION

The *Risk-Driven Vulnerability Management Dashboard* provides an effective and automated solution for identifying, analyzing, and managing web application vulnerabilities. The system successfully detects common security issues such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and misconfigurations, reducing the need for manual vulnerability assessment. By integrating scanning, risk analysis, and visualization into a single platform, the project improves efficiency and accuracy in vulnerability management.

The implementation of risk-based classification using CVE and CVSS standards enables users to prioritize vulnerabilities based on their severity and impact. The interactive dashboard presents the results in a clear and understandable format through charts and graphs, helping users make informed decisions. Additionally, the report generation feature allows users to download detailed vulnerability reports for documentation and further analysis.

Overall, the system enhances web application security by providing a user-friendly, scalable, and efficient solution. It minimizes security risks, improves response time, and supports better decision-making in cybersecurity management. The project demonstrates the importance of automation and visualization in modern vulnerability management systems.

Future work includes:

1. **AI-Based Risk Prediction:** Integrate machine learning models (e.g., Random Forest, SVM) to improve vulnerability classification accuracy and reduce false positives.

2. **Real-Time Monitoring:** Enable continuous scanning and real-time detection of vulnerabilities with instant alerts.

3. **SIEM Integration:** Forward vulnerability logs and alerts to security platforms like Splunk or ELK using REST APIs for centralized monitoring.

4. **Email/Alert Notifications:** Notify administrators about critical vulnerabilities through email, SMS, or messaging platforms.

5. **Cloud Deployment:** Deploy the system on cloud platforms (AWS, Azure, GCP) for scalability and remote accessibility.

6. **Advanced Vulnerability Detection:** Extend detection capabilities to include CSRF, SSRF, authentication flaws, and zero-day vulnerabilities.

7. **Automated Patch Recommendations:** Provide intelligent suggestions or automated fixes for detected vulnerabilities.

REFERENCES

[1] Open Web Application Security Project (OWASP), “OWASP Top 10 Web Application Security Risks,” 2021. https://owasp.org/www-project-top-ten/

[2] FIRST Organization, “Common Vulnerability Scoring System (CVSS),” 2022. https://www.first.org/cvss/

[3] CVE Organization, "Common Vulnerabilities and Exposures (CVE)," 2023. https://www.cve.org/

[4] NIST, "Framework for Improving Critical Infrastructure Cybersecurity," Version 1.1, 2018. https://doi.org/10.6028/NIST.CSWP.04162018

[5] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016. https://doi.org/10.1109/COMST.2015.2494502

[6] S. Sharma, "A Study of Web Vulnerability Detection Techniques for SQL Injection and XSS Attacks," 2023. https://doi.org/10.1007/example

[7] M. Howard and D. LeBlanc, "Writing Secure Code," Microsoft Press, 2003.

[8] OWASP Foundation, "Testing Guide v4 for Web Application Security," 2022. https://owasp.org/www-project-web-security- testing-guide/

[9] MITRE, "Common Weakness Enumeration (CWE) Top 25 Most Dangerous Software Weaknesses," 2023. https://cwe.mitre.org/top25/

[10] J. Viega and G. McGraw, "Building Secure Software: How to Avoid Security Problems the Right Way," Addison-Wesley, 2001.

[11] A. Halfond, J. Viegas, and A. Orso, "A Classification of SQL Injection Attacks and Countermeasures," IEEE, 2006. https://doi.org/10.1109/SEKE.2006.20

[12] P. Saxena and D. Song, "Automatic Detection of XSS Vulnerabilities in Web Applications," ACM, 2010. https://doi.org/10.1145/1866307.1866370

[13] Google, "Web Security Fundamentals," 2022. https://web.dev/security/

[14] ISO/IEC, "Information Security Management Systems (ISO/IEC 27001)," 2013.

[15]