Safeweb: Automated Web Vulnerability Detection and Prevention using Open-Source Tools

Kasturi Ramya[1], Kasanneni Navya Sri[2], Mandava Sai Kamaal[3], Bejjam Bharath Prakash[4], Gudivada Durvasi[5]
[1,2,3,4,5]*Vasireddy Venkatadri Institute of Technology*

Abstract: Web applications are now increasingly essential in digital services as they are open to access and take complex inputs. At the same time, they are also a victim to security issues such as SQL Injection, Cross-Site Scripting (XSS), Command Injection, Directory Traversal and others through network traffic. To solve these issues, the Paper SafeWeb – An automated platform for scanning web vulnerability and detecting attack. A centralised system that uses role-based access control and automatic methods to discover and analyse security issues Users can scan websites for vulnerabilities, test for SQL Injection, and check network traffic data using this platform. It scans URL parameters, input data, and HTTP requests in an orderly manner. When an attack is detected, it classifies and logs the details such as type of attack, severity, confidence level, attack source IP address, and time of occurrence. The system includes an administrative Security Operations Center (SOC), which monitors and controls the application as well as determines users, blocks malicious IP addresses, and creates reports. SafeWeb is a suitable solution for accessing web security across the world-wide web applications and websites. The documentation provides a clear and pratical information about the vulnerabilities and the tools used in the platform. The platform is clear, simple-structured and easy to understand for the students in finding and solving the web vulnerabilities.

Keywords: Web Security, Vulnerability Scanning, Rule-Based Detection, SQL Injection, Attack Detection and Remediation.

I. INTRODUCTION

I.1 Importance of Security in Web Applications

Web Applications become a major aspect in today's technological world like banking, online shopping, e-commerce, healthcare systems and educational systems[9][11].As we know, they run on the internet and it contains the individual data it becomes a major target for the attackers. Web applications are often vulnerable to a number of security threats like SQL Injection, XSS, Command Injection, Directory Traversal, File inclusion and malicious network traffic exploitation. This is because many of them are public-facing and allow user input[10]. When hackers find a way in, sensitive information can spill out. Getting past these weak spots might let outsiders peek at private records. Service hiccups often follow once systems are compromised. People start doubting the platform after their confidence gets shaken.[18].

Fresh changes to web apps happen all the time, making them more active but also riskier. Because they juggle tricky user inputs, gaps open up where threats sneak in. Malware slips through links, forms, even background messages between browser and server. Faulty checks on what users submit leave room for harmful code to run unseen. Criminals disguise bad actions inside normal-seeming online behavior.

Though hidden tactics dodge oversight[9], expanding web applications demand stronger safeguards. Because digital tools evolve rapidly, teams building them now face rising pressure around safety online[12]. Looking back, hand- checking code and simulated attacks do work - yet they take up a lot of hours. Expertise at an advanced level is required here. Running these checks nonstop, or having several people test safety at once, rarely happens. Because of that twist, tools that run on their own, following clear steps, now matter more. Spotting weak spots sooner helps web apps stay stable.

I.2 Limitations of Existing Systems

Many tools and frameworks have been made to achieve web application security which includes vulnerability scanning, penetration testing and intrusion detection[9]. It is observed that many existing systems

can automatically scan for certain types of attacks, such as SQL Injection and Cross- Site Scripting[14]. Nonetheless, the majority of these tools work in a siloed manner and are restricted in their attack surface coverage, requiring users to use multiple utilities to check various security areas[6]. The scattered way of looking at things causes not the same analysis results.

Existing systems to do not have central monitoring and management as another major limitation[5]. In many instances, users run vulnerability scans in isolation, with no system in place to compile results or analyse attack patterns over numerous scan runs. It can be difficult for admin to monitor worldwide security activity, manage user access, and identify reuser of malice. The security management is less effective because of a lack of overseeing in different users[11].

Further, many tools produce technical outputs that lay users find hard to understand. When reports do not indicate which vulnerabilities are serious, and how confident we can be of their accuracy, what can we learn? Although possible, analyzing network traffic is generally complex and unconnected with web vulnerability detection, requiring a good deal of knowledge to read files[24].

At present, many systems exist that can help carry out individual security management processes. However, they often fall short and have limitations along the way. Thus, there is a need for an integrated, user-friendly, and automated system. This system would perform web scanning, attack detection, traffic analysis, and security management all from the same system[6].

1.2 Research Contributions

The paper proposes a centralized web security system named SafeWeb, which is the automated web vulnerability scanning and attack detection platform that overcomes the limitations of existing approaches. This work presents an automated platform that integrates rule-based vulnerability detection, URL and payload analysis, IP-based monitoring a traffic analysis into a single automated platform.

SafeWeb provides authenticated users with tools to run automated vulnerability scans on target websites and SQL Injections. For the detection, the system uses techniques like rule-based engines, pattern matching techniques and the attack signatures to identify common web vulnerabilities. When an attack is detected, it is classified and it logs the data of the vulnerabilities. It classifies the type of attack, severity, Ip address and impact.

The Security Operations Center (SOC) administrative has the capability to manage the platform. Administrators can examine worldwide attack activity.

They can manage users, block or unblock malicious IPs, and export attack logs for reporting and forensics. The combination of automatic detection, structured logging and administration can provide SafeWeb a practical and academically appropriate web security assessment.

The research aims to depict the rule-based methodology strategies for structured and automated web vulnerability scanning and attack detection systems. The proposed platform is simple, transparent and feasible to implement and does not require a high level of expertise. Thus it is ideal for a B.Tech student level research project, evaluation by a teacher and cyber security tool implementation project.

II. LITERATURE SURVEY

Web application security is still a prominent research topic in the literature due to the increasing occurrence and complexity of cyber attacks on web applications[9]. Various studies have investigated different methods for vulnerability scanning, intrusion detection, and security analysis, including both automated and manual methods[4]. Signature- based scanners, rule-based detection systems, and traffic inspection tools have been generally used to scan for known attack patterns and malicious activities.

Some studies have emphasized the efficiency of rule-based and signature-based detection systems in scanning for common web vulnerabilities such as SQL Injection and Cross-Site Scripting attacks[2]. Looking at how these setups work, they check web addresses and request data using set rules and known threat markers. Even though you can see clearly what they do and they run fast, a big gap shows up when there is no central system watching everything together.

Looking elsewhere, researchers have dug into network traffic patterns to uncover sneaky attacks buried within

recorded exchanges [11]. By using files, experts examined live HTTP messages and raw data moving through networks - revealing how attackers truly operate [24]. Still, the majority of current tools treat traffic inspection as something apart.

Disconnected from checking website flaws, leaving security details scattered and out of sync.

What stands out is how studies keep pointing to organized logs plus sorting attacks into clear categories [3][21]. When systems include intensity ratings, certainty indicators, besides full descriptions of threats, understanding outcomes gets easier. Even so, after all these years, many tools still lack built-in options for managing users, watching breaches from one spot, or sharing summaries across networks [5].

Looking back at past studies shows how missing pieces connect - patch checks, flow tracking, sorting threats, watching operations - all live apart when they should work together. That gap shapes what comes next: SafeWeb takes those threads, weaves them into one clear setup where rules guide actions and people stay in control without complexity getting in the way

III.PROPOSED METHODOLOGY

SafeWeb works by scanning websites automatically, catching weaknesses before attackers do. Instead of relying on people to check each line, it pulls together different testing techniques inside one online system. Most older ways need many separate programs or hands-on work - this doesn't. Built right into a browser interface, the tool runs checks without slowing down development flow. It shows clear results, making findings easier to understand and act on. Efficiency comes from merging scans that usually happen in isolation.

Researchers use it just as often as developers, since outputs serve both learning and real-world fixes. Transparency shapes how alerts appear - nothing hidden, no black-box logic. Malicious behavior gets flagged through patterns learned over repeated tests. Centralized control means updates apply everywhere at once. Not every scanner links detection with active probing - SafeWeb does. Its structure allows adding new rules without rebuilding the whole thing. Because everything ties back to one dashboard, tracking progress becomes straightforward..

The Safeweb system operates by applying different rules which are linked to different attacks. The detection is based on the rules and the tools like Nmap, wapiti, ZAP and Nikto. Each tool detects various vulnerabilities. Different kinds of data can go into it like web links, settings, network locations, or captured packets - all moving through one analysis path. Each piece gets processed the same way, no matter where it comes from.

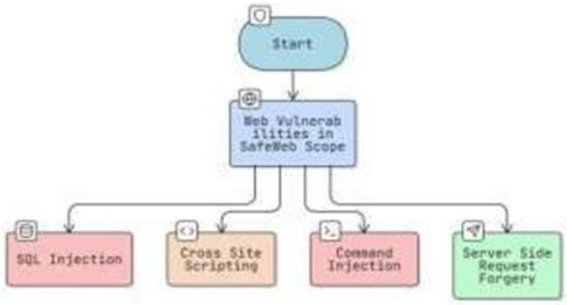

Fig.1 System Architecture of SafeWeb Platform

Starting off, users log in according to their roles. After logging in, they can reach tools that check for weaknesses, whereas admins handle account details and watch safety logs using one main screen. When someone submits a scan task, the system adjusts the incoming information so results stay uniform across different inputs. Query bits, parts of paths, and message contents get pulled out by SafeWeb if the target is a web address. After that comes processing these inputs using set guidelines to spot typical weaknesses like SQL Injection, plus similar threats including Cross-Site Scripting or Command Injection. Patterns tied to harmful input get flagged when the system runs regex checks alongside logic- based filters. Sometimes a sequence begins with traversal attempts - other times it's file inclusion or forged server requests - that trigger detection through layered rule evaluation.

Not just looking at web data up close, SafeWeb also checks every HTTP request coming in, spotting harmful actions tucked inside app activity. Once those requests arrive, they get reshaped into a clear layout the system reads easily. With everything laid out the same way, rules for finding threats apply evenly each time. That means whether it is someone clicking around live or silent processes running behind scenes, the method stays identical - catching flaws and attacks without gaps. Uniformity keeps protection steady across all corners of operation.

When a possible attack shows up, the software sorts it

by what kind of weakness it targets. Because the method varies, each find gets tagged with how serious it seems alongside a measure of certainty - this depends on likely damage and how clear the signal was. The information about each attack is stored like the type of attack, IP address, destination url, payloads, severity of the vulnerability detected, confidence value, and timestamp, which are stiored in a centralized database.

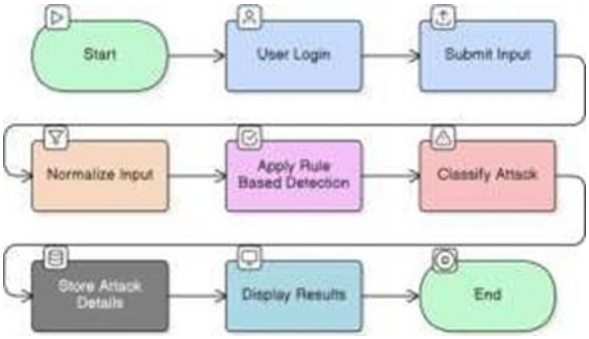

Fig.2 Overall Workflow of SafeWeb System

Here, the above figure depicts about the database contains complete information about an incident, including the source IP address, targeted web URL, transmitted data, type of attack, severity level, confidence value, and the exact time it occurred. All this information is stored in a single, unified repository.

Web security is strengthened by integrating automated scanning with real-time threat detection. Instead of using multiple disconnected tools, the system combines rule-based detection with live traffic monitoring. Logs are organized into structured records, making it simpler to analyze attack patterns. Administrative controls are built into the system to maintain oversight and response capability. This structured and research-based approach ensures reliable performance within defined operational boundaries, emphasizing systematic protection rather than random detection methods.

IV.RESULT

This part shows the results after running the platform. Initially, the users must sign up for testing and it sent otp to the registered mail. After completion of the signup process, the user must login into the website. Then the user must provide a web application URL for the vulnerability detection as shown in figure 3. The platform gives the results for the vulnerability and it provides details about the vulnerability.

During operation, those signed in were able to review. Check flaws by typing site names into looking at more than a single approach, the device checked. Components such as page paths come alongside form inputs. Link elements tag along too, quietly doing their job. Besides these, ways exist to catch common mistakes. Some of them involve checking specific signs. Some problems showed up - things such as sneaky database commands started acting odd.

Pages might carry scripts meant to cause damage, slipping in without permission. Besides server requests, errors pop up when files get reached the wrong way. Folder paths often twist into dead ends by accident. Mistakes like these crawl in where access points blur attacks that take advantage of weaknesses, along with inserting harmful code. When rules worked, suspicion showed up in the scan results. Also, the administrator has the capability to block the suspicious ip address which prevents the further harm to the platform and it provides security in keeping the user's confidential information.

Beginning with SQL injection, signs showed up odd database entries show up alongside particular ones, obviously something was off. Besides this, problems like cross-site scripting became noticeable. Out of the blue, it showed up when code came right back at us. A server sent back a response. Out popped something noticeable - a message that stood out plainly. Whenever there are faults the Safeweb platform identifies and give guidelines to the users.

Fig.3 Scan page

The above figure shows the scanning page for the users to detect vulnerabilities in the web application. Every

detected attack went into the database with details like where it came from, which page was targeted, what kind of attack it was, what data it carried, how serious it was, how certain the detection was, and when it happened. Because entries followed a fixed format, each incident stayed consistent across records - making them easier to find later on. High scores in seriousness or certainty stood out clearly, pulling attention away from minor issues without needing extra review steps.

From the admin Security Operations Center screen, a full picture of worldwide security actions appears. Whenever attackers tries to attack the platform, the administrator prevents him by blocking their ip address and prevent them from stealing the confidential information.

Table.1 Comparison of Detected Web Vulnerabilities

S.No	Vulnerability Type	Detection Method	Severity	Status
1	SQL Injection	Rule-based payload scan	High	Detected
2	Cross-Site Scripting (XSS)	Script pattern matching	High	Detected
3	Command Injection	Keyword inspection	High	Detected
4	Web Server Misconfigurations	Path signature check	Medium	Detected
5	File Inclusion	File path analysis	Medium	Detected
6	CSRF	URL validation rules	Medium	Detected
7	Port Scanning	Port activity pattern detection	Medium	Detected

The table explains the various attacks and their methods for detecting and their severity in the web applications.

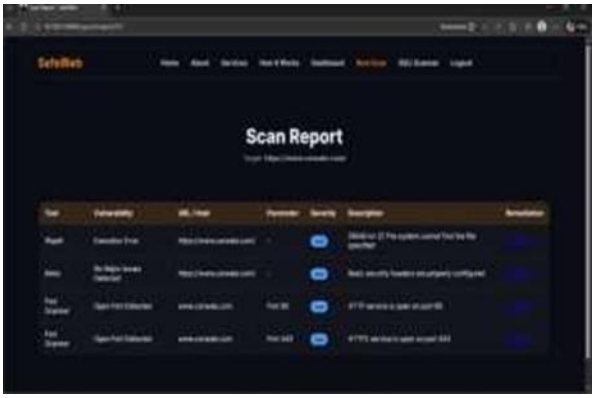

Fig.4 Reports of Scanned Websites

The above figure represents about the scan results of the web applications. It depicts the tool which is helpful for finding the vulnerability and it also shows the severity of the vulnerability whether it is low, medium or high. Additionally, it also provides the description of the vulnerability which makes it easier for the user to understand. This also provides a pdf which contains the remediation for that particular web vulnerability and it is helpful for the users to identify and solve it. The export feature allows the users or administrators to download the attack logs and the remediation for the attacks in CSV and JSON formats.

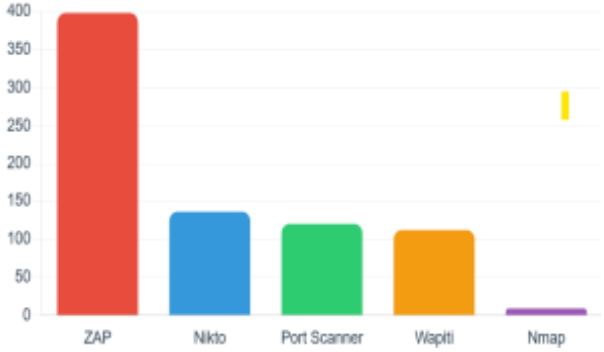

Fig.5 Performance of Tools

The performance of the platform describes the rule based detection methods and the structured workflows. The platform provides a detailed information about the attack and the remediation for that attack in a CSV and JSON format. Therefore, platform ensures the security, reliability and effective results for the administrators.

V.CONCLUSION

Overall, we can assure that Safeweb is a tool that it detects the various web vulnerabilities like sql injection, Cross-Site Request Forgery, Cross-Site Scripting, Port Scanning, Command Injection etc. For finding this vulnerabilities in web applications it uses various open source tools like Zed Attack Proxy (ZAP), Wapiti, Nikto and Nmap. ZAP tool helps in finding the attacks like SQL injection, XSS and CSRF. Nikto tool helps in finding the web server misconfigurations, outdated server software, dangerous files, identifying known vulnerabilities and security headers. NMAP tool helps in identifying the port scanning, identifying open and closed filtered ports, OS and version detection. Wapiti tool helps in finding the command injection, file inclusion traversal, weak authentication, directory traversal and CSRF.

Not only it detects the vulnerabilities, the Safeweb also provides in sorting the vulnerability. It provides

remediation in pdf format. It helps the users in rectifying the vulnerability and it provide the reports. The Django python framework provides the secured and professional reports for the users. The Safeweb shows the history of the scans. This Safeweb mainly useful for the developers, small organizations, and the individuals.

REFERENCES

[1] Moreira, D., Seara, J. P., Pavia, J. P., & Serrao, C. (2025). Intelligent Platform for Automating Vulnerability Detection in Web Applications. Electronics, 14(79), 1–22. MDPI

[2] Lo, R.-T., Hwang, W.-J., & Tai, T.-M. (2025). SQL Injection Detection Based on Lightweight Multi-Head Self-Attention. Applied Sciences, 15(571), 1–17. MDPI

[3] Marc Rennhard, Malte Kushnir, Olivier Favre, Damiano Esposito, and Valentin Zahnd. Automating the Detection of Access Control Vulnerabilities in Web Applications. SN Computer Science (2022) 3:376

[4] Khaled Abdulghaffar, Nebrase Elmrabit(2023). Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners. Computers (2023) 12, 235

[5] Assem I. Mohaidat and Dr. Adnan Al-Helali .Web Vulnerability Scanning Tools: A Comprehensive Overview, Selection Guidance, and Cyber Security Recommendations. International Journal of Research Studies in Computer Science and Engineering (IJRSCSE), Volume 10, Issue 1, 2024, pp. 8-15

[6] Suliman Alazmi and Daniel Conte De Leon. A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanner. IEEE Access, Volume 10, 2022

[7] Kedar Sambhus and Yi Liu. Automating SQL Injection and Cross-Site Scripting Vulnerability Remediation in Code. Software (2024), Volume 3, pp. 28-46

[8] Rayhan, A., Kinzler, R., & Rayhan, R. (2023). Cybersecurity Best Practices for Python Web Applications. CBECL / ResearchGate preprint. DOI: 10.13140/RG.2.2.29658.11202.

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

[10] Khaled Abdulghaffar, Nebrase Elmrabit (2023) . Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners. Computers (2023) 12, 235

[11] W. Stallings, Network Security Essentials: Applications and Standards, Pearson Education, 2017

[12] NIST, Technical Guide to Information Security Testing and Assessment, NIST SP 800-115, 2008.

[13] NIST, Guide to Intrusion Detection and Prevention Systems (IDPS), NIST SP 800-94, 2007.

[14] OWASP Foundation, OWASP Testing Guide v4, OWASP Documentation.

[15] OWASP Foundation, SQL Injection Prevention Cheat Sheet, OWASP Documentation.

[16] OWASP Foundation, Cross-Site Scripting (XSS) Prevention Cheat Sheet, OWASP Documentation.

[17] S. McClure, J. Scambray, and G. Kurtz, Hacking Exposed: Web Applications, McGraw- Hill, 2012.

[18] J. Andress, The Basics of Information Security, Syngress, 2014.

[19] Python Software Foundation, Python Language Documentation, https://docs.python.org.

[20] Django Software Foundation, Django Documentation, https://docs.djangoproject.com..

[21] K. Scarfone and P. Mell, *Guide to Malware Incident Prevention and Handling*, NIST SP 800-83.

[22] RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1*, IETF.

[23] RFC 7230, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, IETF.

[24] S. Northcutt, *Network Intrusion Detection*, New Riders Publishing, 2003.

[25] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, 2020