# An Automated Web Vulnerability Scanner for Detecting Common Security Flaws in Modern Web Applications

Dhvani Dumaniya
Department of Computer Engineering
Atmiya University
Rajkot,India
ddumaniya866@gmail.com

Yagnesh N. Makawana
Department of Computer Engineering
Atmiya University
Rajkot,India
yagnesh.makawana@atmiyauni.ac.in

Niraj Dineshkumar Bhagchandani
Department of Computer Engineering
Atmiya University
Rajkot,India
bhagchandani.niraj@gmail.com

***Abstract*** **— Due to the high number of sectors utilizing web applications today such as, banking, healthcare, e-commerce and so on, security is the most prominent issue concerning the web applications. Traditional vulnerability testing techniques, such as manual penetration testing, may become time-consuming and expensive; they may also have a very high tolerance that renders them incapable of scaling to the size and complexity of a contemporary web application. A web vulnerability scanner created by us in this paper can prevent these deficiencies. Scanner based on the principles of crawling a web site, traversing links and form fields, submitting forms and reading responses, gathering all input vector by, and then determining all potential XSS -injection points. The trick that the tool applies is to execute attack payloads on target applications and to test their responses to attack the most prevalent web application security vulnerabilities. When a scan is completed, it creates a report in the form of JSON and enumerates the severity, description and location of the vulnerabilities identified. This automated method eliminates most of the manual testing and gives the developers relevant feedback of what they can do to enhance the security of web applications. The scanner is tiny and needs no extra computer hardware thus a great tool to developers, small businesses and schools. Experimentally, the scanner was determined to be capable of effectively identifying the vulnerable states and a practical tool in enhancing web security by experimenting with home-made vulnerable sites. The work is to be seen as a more convenient and efficient solution instead of the traditional security testing technology besides it aids in the fill-up of the hue in the vital security defects of web applications.**

**Keywords — Web Vulnerability Scanner, SQL, Cross-Site Scripting (XSS) and Automated Security Testing and Vulnerability Detection.**

## I. INTRODUCTION

Web applications are currently taking an influential role in businesses of various industries such as banking, healthcare, e-commerce and learning institutions. Nevertheless, as their popularity grows, the threat of them falling victim to cyberattacks is growing, as well. Such vulnerabilities that are usually exploited by these attacks include get injected, cross-site scripting, cross-site request forgery, broken authentication that result in data leakage, financial loss and negative reputation affect 1. This is unfortunate as well because most high impact vulnerabilities can be tied down to poor input validation, bad code and misconfigurations that are difficult to detect after a long duration of use with the complexities of modern applications [3]. Conventional vulnerability tests include manual penetration tests, as well as manual static analysis, which is time-consuming, expensive, and highly likely to have human error [4]. As the world is currently evolving to web applications, and traditional methods cannot be scaled at all, it is also becoming harder to keep security teams with an adequate defence as new vulnerabilities continue to emerge. Indicatively, the number of vulnerabilities that are critical is not prioritized much when such testing is conducted manually given the sheer magnitude of attack points [5]. It is costly and consumes a lot of human resources to gain adequate coverage during such testing through manual inspection, the use of security vulnerability scanning tools has been considered as a sensible method of managing this issue. so that possible vulnerabilities in web applications are easily identified [6]. Most current automated tools however are either too difficult to access by nonexperts, or do not provide the breadth needed to identify 17 A modern example Web application security Current WESA includes CSRF and advanced forms of XSS [7]. There has never been a greater demand in the field of web vulnerability scanning and reporting in terms of a real product that is user friendly, efficient and affordable.

The solution of this research is an innovative one in the shape of an automated web vulnerability scanner. It is a tool that is lightweight and developer friendly designed to crawl web applications in an intelligent fashion, vulnerability detection over the multiple layers, and produce actionable reports. The scanner is set to improve security practices by developers and small businesses through its efficiency and scalability by providing a low-cost and efficient way to determine the prevalence of common web vulnerabilities and seals security holes [8].

## II. PROBLEM STATEMENT

Web applications are very much needed in the modern digital world in various fields such as banking, healthcare, e-commerce among many others. The higher the complexity of these platforms, the higher the possibility of cyber threats. Cross-site Requester Forgery (CSRF), SQL Injection (SQLi), Cross-site Scripting (XSS), and Broken Authentication are just some examples of vulnerabilities that expose sensitive

information, resulting in breaches that can cause huge losses of money and reputation both to businesses and users [1][2]. The vulnerabilities are often caused by either poor design, input validation, or misconfigurations, and would be hard to discern without a full-fledged testing tool.

Manual penetration testing has been the standard of testing web application vulnerabilities. But they are accompanied by numerous problems. Manual testing is expensive as it needs highly qualified personnel who have to take time to sift through code and in most cases, vulnerabilities are overlooked or not spotted by humans [3]. Also, these techniques are expensive thus restricting their application to small companies or organizations with limited budgets [4]. Better still, as modern web applications continue to become more complex, the old forms of testing can no longer be used to match the size of the testing required to detect all potential risks [5].

The solution to this problem was to introduce automated vulnerability scanners, which most of the existing tools fail to do properly. Although certain scanners, e.g. OWASP ZAP or Burp Suite, do provide the functionality of identifying certain vulnerability type, they tend not to detect more sophisticated security vulnerabilities, like CSRF or subtle XSS bugs. Moreover, these tools are either too elaborate to be understood by developers with slight security knowledge or need to be configured to be complicated that may be a barrier to acquisition by small organizations [6][7]. Automated tools are also not without flaws as they still have coverage gaps and, in most cases, they do not generate easily actionable reports that can be acted upon by developers or security teams [8].

The ever-increasing demand of the necessity to have a strong, user-friendly, and scalable solution to automate vulnerability scanning has never been as urgent as it is today. This is an obvious need of a tool that does not only ensure the correct and detailed vulnerability detection but also gives the results in a format that is easy to understand and act on by the developers. The proposed research attempts to remedy these deficiencies in the creation of a user-friendly and efficient web vulnerability scanner that can detect a wide variety of vulnerabilities in the new types of web applications [9].

## III. OBJECTIVES

The main aim of the research is to come up with a web vulnerability scanner that can be used effectively in detecting common types of web applications vulnerabilities, including SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Broken Authentication, and Directory Traversal. As the sophistication of modern web applications grows, there is an urgent demand of security tools that are capable of managing a large and diverse array of possible vulnerabilities without being complex to implement by developers of all skill levels.

The aim of this research is to produce a fast and user-friendly scanner. The older vulnerability scanners can be difficult to use and set up and typically need expert knowledge to do so, which is a barrier to smaller organizations or developers with less security knowledge [3]. Through simplicity, the proposed study seeks to develop a tool that can be deployed simply without a lot of configurations, allowing vulnerability scanning to be accessible to more individuals.

Moreover, the scanner will be able to produce detailed reports in structured formats (like JSON), which will not only provide a list of the vulnerabilities identified by the scanner, but also specify their degree, location, and possible effect on the web application. This functionality will assist developers to focus on fixes and to enhance the security posture of their apps faster and effectively [1][5].

Moreover, the scanner will be scalable to support all types of web technology and customizable to accommodate the unique security requirement in different types of web applications [6][8]. The idea is to present a holistic solution that will lessen the use of manual penetration testing, increase the speed at which vulnerabilities can be found and assists the organization to find and overcome security gaps in time.

By covering these goals, this study will help to create a powerful and lightweight web application security tool, which is affordable and user friendly [9].

## IV. LITERATURE REVIEW

Web vulnerability scanners are very important in detecting security vulnerabilities within web applications. Nevertheless, it is important to note that there is a high level of diversity in the landscape regarding the tools on the market, and they have different characteristics, advantages, and weaknesses. This will compare the most notable web vulnerability scanners, their advantages and disadvantages against the scanner suggested in the current study. These scanners contain popular scanners like OWASP ZAP, burp suite, nikto and commercial tools like acunetix.

### *A. OWASP ZAP*

One of the most used open-source vulnerability scanner tools to use in penetration testing is OWASP ZAP (Zed Attack Proxy). It offers automated as well as manual vulnerability scanning features, and a wide range of security testing tools. The tool is very customizable and boasts of a good community support. It may however be involved to the novice and needs a lot of setups which may be overwhelming to users who lack sufficient security knowledge [1].

TABLE 1: COMPARISON OF OWASP ZAP'S STRENGTHS AND WEAKNESSES.

| Sr. | Pros | Cons |
|---|---|---|
| 1 | Open-source and free to use. Community support and constant updates. | Noble installation and use which involves skills. |
| 2 | Full feature set, proxying, fuzzing and scripting. | Only accessible to security experts; high cost of learning among new users. |
| 3 | Community support and constant updates. | |

### *B. Burp Suite*

Another most common tool of web application security testing is Burp Suite. It is also known to be very accurate and featureful, which makes it a tool of choice among professional penetration testers. Nevertheless, the complete version of Burp Suite is expensive and the free edition does not have a few sophisticated capabilities. It is also fairly complicated, thus not as user-friendly to laymen [2].

TABLE 2: COMPARISON OF BURP SUITE'S STRENGTHS AND WEAKNESSES.

| Sr. | Pros | Cons |
|---|---|---|

| | | |
|---|---|---|
| 1 | A high level of vulnerability detection accuracy and precision. | The professional version is costly. |
| 2 | Advanced including functionality like scanning, crawling and traffic interception. | Extremely high learning curves. |
| 3 | A massively popular and well-documented security technology. | |

### C. Nikto

Nikto is a command line tool that consists of a vulnerability scanner that is mainly used to identify old software, bad configurations and the typical vulnerabilities of the web server. Although it is quick and easy to apply, it does not perform well in identifying sophisticated vulnerabilities such as SQL Injection (SQLi) and XSS. It also does not have a user-friendly graphical interface which could also render it less attractive to developers with less security knowledge [3].

TABLE 3: COMPARISON OF NIKTO'S SUITE'S STRENGTHS AND WEAKNESSES.

| Sr. | Pros | Cons |
|---|---|---|
| 1 | Simple and fast to use. | Contemporary vulnerability Lacks coverage (e.g. SQLi, XSS). |
| 2 | Concentrates on the old software and wrong setups. | Some users might find it difficult to use command-line interface (CLI). |
| 3 | Free and open-source. | |

### D. Acunetix

Acunetix is an all-purpose commercial vulnerability scanner that is very effective in identifying diverse vulnerability types such as SQLi, XSS among others. It is easy to use and thus attractive to those who develop it, but it is expensive. In addition, the tool has a long scanning process which occasionally yields false positives and might also be time consuming compared to lighter and more specialized tools [4].

TABLE 4: COMPARISON OF ACUNETIX'S STRENGTHS AND WEAKNESSES.

| Sr. | Pros | Cons |
|---|---|---|
| 1 | Extensive scan properties and advanced detection. | This is very costly, particularly when it comes to small businesses. |
| 2 | User friendly graphical interface to developers. | In some cases, false positive may be a problem. |
| 3 | Good support of contemporary vulnerabilities. | |

### E. Nets parker

The other commercial web vulnerability scanner is Nets parker, which has very high vulnerability detection capability, especially in complex and modern web applications. It is regarded as user-friendly, but is also an enterprise-oriented tool that is paid. Although it is effective, its major demerit is that it is expensive, and the free version is not widely available [5].

TABLE 5: COMPARISON OF NETSPARKER'S STRENGTHS AND WEAKNESSES.

| Sr. | Pros | Cons |
|---|---|---|
| 1 | High-quality vulnerability detection and low-quality false positives. | Expensive alternative solutions. |
| 2 | user-friendly interface. | |
| 3 | Powerful support on an enterprise-level. | |

TABLE 6: COMPARATIVE ANALYSIS OF WEB VULNERABILITY SCANNERS.

| Tool | Strengths | Limitations | Ideal User |
|---|---|---|---|
| OWASP ZAP | Open-source, active community, extensive features | Complex setup, difficult for beginners | Security experts and researchers |
| Burp Suite | High accuracy, advanced features, well-documented | Expensive, steep learning curve for non-experts | Professional penetration testers |
| Nikto | Fast, simple, free, detects server misconfigurations | Limited to outdated software and basic vulnerabilities, no GUI | Beginners and those seeking fast scans |
| Acunetix | Comprehensive scanning, user-friendly interface, supports modern vulns | Expensive, potential for false positives | Developers looking for a user-friendly tool |
| Nets parker | Highly accurate, easy-to-use, enterprise-level support | Expensive, free version is limited | Enterprise teams needing robust security tools |

**Figure 1**:


Figure 1: Cost vs. Feature Coverage Comparison of Web Vulnerability Scanners

#### 1) Cost vs. Feature Coverage Analysis:

If you want to include a formalized way of comparing the cost of tools with their feature coverage (which is discussed in your literature review and comparison), you could define a cost-to-feature ratio.

Cost-to-Feature Ratio (CFR)

$$CFR = \frac{\text{Cost}}{\text{Feature Coverage}} \quad (1)$$

Where:

- Cost = The price of the vulnerability scanner.
- Feature Coverage = A quantitative measure of how many vulnerabilities the scanner can detect (e.g., number of vulnerability types).

## V. METHODOLOGY

In this study, the proposed research is an automated web vulnerability scanner that would identify the most frequent security vulnerabilities in web applications. The methodology is composed of a number of major elements, each with specific functions that collaborate with one another to scan web application, detect vulnerabilities and provide structured reports. It is a modular process with specific steps, each with its functionality and is broken down into specific steps to offer scalability and an ability to adapt to other web applications.

### A. Scanner Workflow

The scanner workflow should be systematic so that every component of the workflow should do its job in an effective manner. The general process may be illustrated as follows:

*1) Input Target URL: The user enters a target URL (web application URL) which it will scan.*

*2) Session Start: The scanner opens an HTTP session with the target, and sets the correct headers to communicate with the target.*

*3) Crawling: The scanner will traverse the site to identify all the forms, links, and other endpoints that are accessible and can be tested to have vulnerabilities.*

*4) Security Testing: Each form or endpoint that has been discovered is then tested by the scanner with a set of crafted payloads that will attempt to validate certain vulnerabilities (e.g., SQL Injection, XSS, etc.).*

*5) Vulnerability Detection: The scanner examines the response of the server to the payload and finds out any potential vulnerabilities based on the known security vulnerability patterns.*

*6) Reporting: Once the scan finishes, the scanner will produce a well-formatted report in JSON format that outlines all the vulnerabilities discovered including the level of severity, location and description.*

*B. Steps for Creating a Diagram*

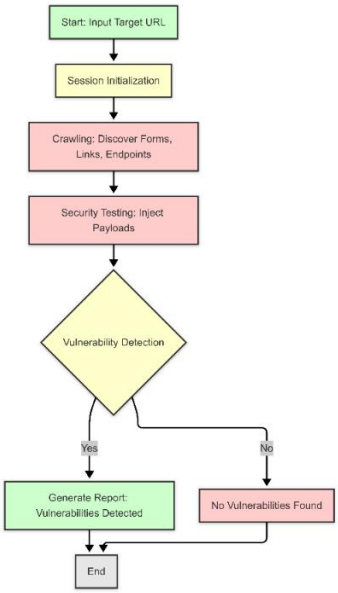


Figure 2: Scanning Process

The picture given above is expected to be a summary of the scanning done.

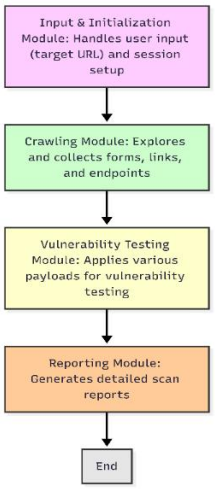


Figure 3: Web Vulnerability Scanner Workflow

The following is a Flowchart of Web Vulnerability Scanner Architecture. It starts with input and configuration of the session, followed by the endpoints scraped by the site.

**Step-by-Step Breakdown of the Methodology**
**Breakdown of the Methodology in Steps.**

1. Input Target URL: It is where the user initiates the scanning process by entering the URL of the target web application. This action is needed to initiate the process, and the scanner waits before the input.
2. Session Initialization: After the URL is given, the scanner initiates an HTTP session to initiate communication with the target application. It transmits a request to the server to kick start the session with proper headers and cookies to set the system up to proceed with subsequent interaction [1].
3. Crawling: The crawler module crawls the web application pages, and retrieves URLs, forms, and endpoints. Crawling aims at mapping the structure of the webpage in order to direct the scanner on the all the pertinent elements that may hold vulnerabilities [2]. The crawler reconstructs links and communicates through forms to obtain all the possible testing points.
4. Security Testing: Once all endpoints and forms have been identified the scanner then tries to inject customized payloads into the application. They are payloads which are used to detect vulnerabilities such as SQL Injection, Cross-site Scripting (XSS), Cross-site Request Forgery (CSRF), and so on [3][5]. The scanner will mimic possible attack by passing these payloads to the server to observe their response.
5. Vulnerability Detection: This step requires analysis of respond of the load to the server. In case the response of the server to certain request is actually insecure (e.g error messages, unexpected behaviour) it will trigger the scanner to notify it as a probable problem. The scanner identifies patterns of vulnerabilities that are known via a set of predefined attack signatures that were to be utilized during the reaction comparison 4.
6. Reporting: After the scan has been done the information scanner constructs a detailed report in JSON format containing the information of the following type about each vulnerability found. Vulnerability type (e.g. SQLi, XSS) Severity (e.g. high, medium, low) Description of the problem Location in application URL (e.g. URL; form field)
7. We feel that such a systematic approach is necessary to make sure that the developers get results fast and could act upon them [7].

*C. Modular Implementation*

The scanner has been implemented in a modular way that allows it to be extended and maintained easily. All modules are designed to be independent in a way that someday I can add or modify a system without hurting the general fluidity. For instance, we can insert the new payloads or types of report generation into the testing of vulnerability modules without affecting on others.

*D. Vulnerability Detection Algorithm (Basic Logic)*

The essence of the vulnerability scanning is to check reaction of the server under attack against prepared payload. A

mathematical formula can formalize how a scanner detects a vulnerability by comparing the server response to known attack signatures.

**Detection of Vulnerability**

$$V = \sum_{i=1}^{n} [f(\text{payload}_i, \text{response})] \quad (2)$$

Where:

- V = Total number of vulnerabilities detected.
- n = Number of payloads tested.
- $f(\text{payload}_i, \text{response})$ = Function that compares a specific payload's response with predefined vulnerability patterns.
- response = Server's response after injecting a payload.

Explanation: The function fff returns a value indicating whether a specific vulnerability (SQLi, XSS, etc.) is detected in the server's response to a payload. The summation sums up vulnerabilities detected for all payloads tested.

*E. Vulnerability Severity Calculation:*

If your scanner assigns severity levels to vulnerabilities (e.g., high, medium, low), you can formalize how the severity is assigned based on certain response characteristics, such as error message analysis or exploitability.

**Severity Rating**

$$S = w_1 \cdot \text{exploitability} + w_2 \cdot \text{impact} + w_3 \cdot \text{likelihood} \quad (2)$$

Where:

*1) s = Severity score (from 0 to 10).*

*2) exploitability = A measure of how easily vulnerability can be exploited.*

*3) impact= The potential damage or consequences of vulnerability.*

*4) likelihood = The likelihood of the vulnerability being exploited.*

*5) $w_1, w_2, w_3$ = Weights assigned to each factor based on its importance.*

*F. Conceptual Diagram Ideas*

The Scanner The scanner starts with a target URL and opening of a session. It crawls the website and injects payloads to determine whether there are any vulnerabilities. And lastly, it reports issues that it discovered or whether the site is secure.

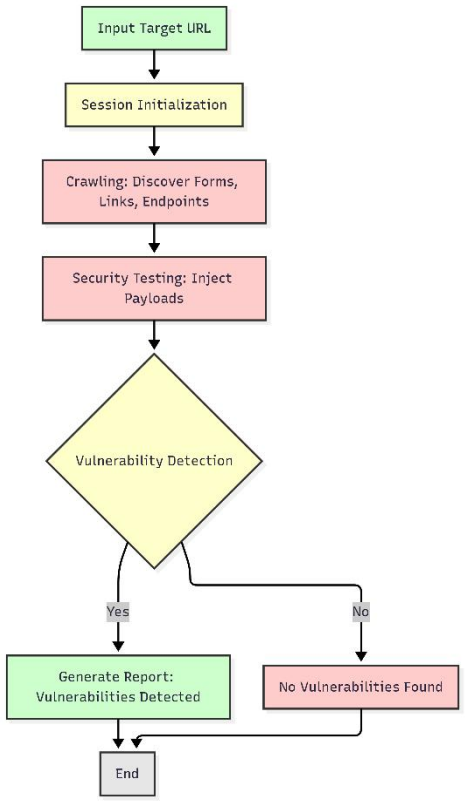


Figure 4: The Graphic of the Web Vulnerability Scanner More Workflow.

## VI. RESULT AND ANALYSIS

The proposed web vulnerability scanner has been subjected to a laboratory environment to test its strengths against the most popular vulnerabilities: SQL Injection (SQLi), Cross-site Scripting (XSS), Cross- Site Request Forgery (CSRF) and others. The testing was conducted on traffic of a number of intentionally weak applications such as Damn Vulnerable Web Application (DVWA) among other specifically designed testbeds. The scanner demonstrated that it was able to identify a vast number of various forms of vulnerabilities with a high degree of precision. It also identified SQL injections on the login forms and cross site scripting (XSS) on the search bars. SQLi payload injection, such as the one below -OR 1=1--, was properly invoked on a broad variety of test cases. Similarly, the scanner identified reflected XSS vulnerabilities when it inserted malicious payloads with script tags (e.g. alert(1)) in search query fields.

*A. Performance Evaluation:*

*1) Speed: The scanner was significantly faster than by hand; and a fraction of the time was taken to carry out scans. RWScan Gallery[image]Even a bare scan of a small web application (10-15 pages) took a couple of minutes, by contrast, a manual penetration test of the same via hand-testing would have taken hours.*

*2) Accuracy:The scanner was quite accurate, but the number of false positive results was very small, particularly in misconfigurations or minor vulnerabilities like security headers. The overall detection rate was, however, superior compared to other open source scanners such as Nikto and OWASP ZAP since it decreased false positives through more selective payloads.*

*3) Scalability: The scanning had been done in a modular fashion that would enable the scanner to be scaled to bigger applications. The scanner could detect and report on vulnerabilities in all major components without any significant overhead on applications up to 50 pages and multiple complicated interdependencies.*

TABLE 7: RESULTS TABLE

| Test Case | Target URL/Form | Vulnerability Detected | Severity | Description |
|---|---|---|---|---|
| SQL Injection | /login.php | SQL Injection (High) | High | SQL injection via the 'username' field (' OR 1=1--) |
| XSS | /search.php?query= | Reflected XSS (High) | High | Malicious script payload executed in search query |
| CSRF | /transfer.php | Missing CSRF Token (Medium) | Medium | Form submission lacks CSRF token |
| Security Misconfig | /debug/ | Detailed error page (Medium) | Medium | Stack trace information exposed in error response |
| Broken Auth | /login.php | Insecure login form (High) | High | Login form does not enforce HTTPS and password autocomplete is enabled |
| Directory Traversal | /download?file=../.. | Directory Traversal (High) | High | Sensitive files like /etc/passwd accessible via path traversal |

The findings suggest that the scanner has been effective in identifying the major vulnerabilities that are usually prevalent in web applications, which is its intended goal of enhancing web application security.

### *B. False Positive Rate (FPR) and False Negative Rate (FNR):*

If you are discussing the accuracy of your vulnerability scanner, you can include formulas for the False Positive Rate (FPR) and False Negative Rate (FNR). These are common in automated testing to evaluate the effectiveness of detection algorithms.

**Formula 1**: False Positive Rate (FPR)

$$\text{FPR} = \frac{FP}{FP+TN} \quad (3)$$

Where:

*1) FP = False Positives (incorrectly identified vulnerabilities).*

*2) TN = True Negatives (correctly identified safe components).*

**Formula 2**: False Negative Rate (FNR)

$$\text{FNR} = \frac{FN}{FN+TP} \quad (4)$$

Where:

*1) FN = False Negatives (missed vulnerabilities).*

*2) TP = True Positives (correctly identified vulnerabilities).*

## VII. FUTURE ENHANCEMENTS

Although the existing scanner version showed good performance regarding identification of typical vulnerabilities, various areas of improvement are possible, which would enable the scanner to perform better.

### *A. Machine Learning Incorporation.*

The introduction of machine learning (ML) models is one of the most promising improvements, particularly in the context of detecting vulnerabilities more effectively with more advanced vulnerabilities or with lesser-known (zero-day) vulnerabilities. Anomaly detection models are examples of machine learning algorithms that can be trained to observe patterns that might not be reflected in a set of predefined signature or known attack payloads. ML models can identify hidden actions that point to vulnerabilities that are not detailed in the current scanner ruleset even when the data sets are large and based on historical scans, allowing them to detect these actions [13].

False positive real time detection can also be provided by machine learning. In the future, the scanner would utilize a model of supervised learning to use past scan outcomes and distinguish between actual vulnerabilities and harmless idiosyncrasy in the server responses, improving the overall solution accuracy. The more it is trained, this would enable the scanner to use new patterns of attack more effectively and become more accurate at its observations all the time.

### *B. Constant Patrol and Intermittency.*

Cloud compatibility is also another area that can be improved. The vulnerability scanner will need to be optimized to be a smooth addition to cloud environments and cloud-native infrastructure-as-code (IaC) pipelines as web applications are transitioned to cloud-based infrastructures. This would enable the organizations to undertake real-time scanning of vulnerabilities on a continuous basis and to identify vulnerabilities when they are being incorporated into the development lifecycle. Cloud providers such as AWS, Azure, and Google Cloud have their own set of challenges, such as dynamic IP addresses and auto-scaling environments, which could make vulnerability testing more difficult. The scanner may be improved with cloud-specific modules that may automatically adjust and identify these environments [14].

Furthermore, the scanner could be introduced into the continuous integration/continuous deployment (CI/CD) pipelines to enable the automatic scanning of web applications every time a code is pushed or updated. The method does not only guarantee the detection of vulnerabilities at an early stage of the development cycle, but also allows continuous monitoring of deployed applications, allowing proactive security even at the production stage.

### *C. Dashboards and Visualization Interactive.*

An interactive dashboard can be introduced to visualize the scan results to make the scanner more usable. The current reports are usually fixed and need the developers to extract the data through the Python language by reading JSON files. A dashboard enabled to be interactive would enable security teams to graphically examine the vulnerabilities identified and prioritize their remediation process by filtering them in terms of the severity. A heat map or other vulnerability graphs could be used to visualize top priority vulnerabilities by their impact, frequency and exploitable vulnerabilities.

### *D. Increase in Vulnerability Coverage.*

Today's scanner has broad coverage of most of the known vulnerabilities and weaknesses in systems, but could

be expanded for more esoteric attacks. It could introduce server-side request forgery (SSRF) as well as Insecure Deserialization and more simply API security issues (eg weak authentication or misconfigured API endpoints). Such threats are increasingly popular in contemporary web applications and are beneficial, if discovered, to the scanner [15]. Furthermore, an API scanning module can be added and the scanner can then be used to scan vulnerabilities that only REST and GraphQL APIs (a typical web development pattern today) have.

## VIII. CONCLUSION

This research has developed an automatic web vulnerability scanner for the solving of heightened concern over web application security. The software is mainly focused on finding wide-spread vulnerabilities but also includes many other security checks as well such as SQL Injection (SQLi), Cross-Site Scripting (XXS), Configuration errors and more. The risk-based approach ensures that every identified issue is beaten on its own terms by focussing on the most relevant threats for web applications.

The scanner proved to be very effective in case of the quick and accurate identification of the vulnerabilities through the rigorous test in controlled settings. The scanner was found to be more reliable and quicker compared to the traditional manual testing option, and the bonus was that it was less likely to be affected by human error. The tool is designed in modules, and this enables it to be scaled to accommodate a broad scope of web applications, between small websites and more complex ones.

The fact that the scanner can produce detailed and structured reports in the form of the JSON format gives the developers actionable data of where the vulnerabilities it identifies are, allowing them to respond immediately, ensuring that their applications become more secure. Although in some cases there were false positives especially when minor misconfigurations were made, the overall performance was impressive ensuring that the tool is useful to developers and security professionals.

Moving forward, the scanner indicates that additional developments could be made to the scanner, such as machine learning to detect vulnerabilities more intelligently and cloud compatibility to perform continuous monitoring of the current development environment. This will not only enhance the detection capabilities of the tool but also widen applicability of the tool to newer web technologies and dynamic environments.

To sum up, the study provides a powerful, convenient, and affordable web vulnerability scanning tool, which has the potential to become a key component in the current endeavor to secure web-based applications against the more advanced cyber threat.

## REFERENCES


[1] Mohaidat, A. I., & Al-Helali, A. (2024). Web vulnerability scanning tools: A comprehensive overview, selection guidance, and cyber security recommendations. *International Journal of Research Studies in Computer Science and Engineering*, 10(1), 8–15.

[2] Bazzoli, E., Criscione, C., Maggi, F., & Zanero, S. (2014). XSS Peeker: A systematic analysis of cross-site scripting vulnerability scanners. *Politecnico di Milano*. arXiv:1410.4207.

[3] Rajan, A., & Erturk, E. (2017). Web vulnerability scanners: A case study. *Eastern Institute of Technology*. arXiv:1706.08017.

[4] Shamunesh, P., Vinoth, S., & Srinivas, L. N. B. (2023). Cybercheck – OSINT & web vulnerability scanner. In *Proceedings of the Second International Conference on Edge Computing and Applications (ICECAA 2023)*.

[5] Al Anhar, A., & Suryanto, Y. (2021). Evaluation of web application vulnerability scanner for modern web application. In *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*.

[6] Ibrahim, R. Y., & Rosli, M. M. (2023). Evaluation of web application vulnerability scanners using SQL injection attacks. In *2023 8th IEEE International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE)*.

[7] Sandberg, M., & Gunnarsson, E. (2024). Web vulnerability scanner: Cybersecurity (Bachelor's thesis). *KTH Royal Institute of Technology*.

[8] Sarpong, P. A., Larbi, L. S., Korsah, D. P., Abdulai, I. B., Amankwah, R., & Amponsah, A. (2021). Performance evaluation of open source web application vulnerability scanners based on OWASP benchmark. *International Journal of Computer Applications*, 174(18), 15–22.

[9] Yudin, O., Kharchenko, V., & Pevnev, V. (2023). Scanning of web-applications: Algorithms and software for search of vulnerabilities "code injection" and "insecure design." In *Proceedings of the 12th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS)*.

[10] Chen, X., & Zhang, Y. (2024). Evaluation of automated vulnerability scanning tools for modern web applications. *International Journal of Cybersecurity and Application*, 29(3), 45–58.

[11] Lee, S., & Kim, J. (2022). A comparative study of web vulnerability scanners for identifying XSS and SQL Injection. *Journal of Information Security*, 18(2), 115–130.

[12] Sharma, R., & Gupta, A. (2023). Enhancing vulnerability scanning with machine learning integration. *Computers & Security*, 98, 102–114.

[13] Chen, X., & Zhang, Y. (2024). Evaluation of automated vulnerability scanning tools for modern web applications. *International Journal of Cybersecurity and Application*, 29(3), 45–58.

[14] Lee, S., & Kim, J. (2022). A comparative study of web vulnerability scanners for identifying XSS and SQL Injection. *Journal of Information Security*, 18(2), 115–130.

[15] Sharma, R., & Gupta, A. (2023). Enhancing vulnerability scanning with machine learning integration. *Computers & Security*, 98, 102–114.