Volume: 13 Issue: IV Month of publication: April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69308

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 13 Issue IV Apr 2025- Available at www.ijraset.com*

# Webshield: A Comprehensive Web Vulnerability Detection and Recommendation Tool

KalaiSelvi B[1], Pushpa K[2], Priyadharshini S[3], Saranya devi K[4]
[1]*Associate Professor, Department of Computer Science and Engineering, Mahendra Engineering College, Mallasamudram, Tamil Nadu, India*
[2, 3, 4]*UG Student, Department of Cyber Security, Mahendra Engineering College, Mallasamudram, Tamil Nadu, India*

***Abstract: The Webshield is a lightweight yet robust web vulnerability detection tool built to help users identify security flaws in web applications with minimal effort. Designed to be beginner-friendly, it allows anyone to scan a website simply by entering its URL, after which it automatically checks for threats like SQL Injection, Cross-Site Scripting (XSS), Directory Traversal, Command Injection, and missing HTTP security headers. What sets it apart is its ability to mimic real-world attack scenarios by injecting crafted payloads and analysing how the server responds. Any discovered vulnerabilities are clearly displayed using a color-coded output system along with concise, actionable suggestions for remediation. The tool's Python-based architecture makes it easy to customize, extend, or integrate into development pipelines, supporting both educational and professional use cases. It also generates detailed summary reports that highlight all findings, helping users fix problems and track improvements over time. Webshield not only detects risks but also promotes secure coding practices and raises awareness about common security oversights. Whether you're a student, developer, or ethical hacker, this scanner provides a practical way to enhance web application security.***
***Keywords: Web security, Vulnerability Scanner, SQL Injection, XSS Protection, Cybersecurity, Website Protection.***

## I. INTRODUCTION

You would be hearing about a new website going online or getting hosted almost every other second. These websites today offer an enormous range of online services—everything from banking, shopping, education, and communication to entertainment and beyond. As the Internet has grown in size and sophistication, it has also become more complicated and increasingly vulnerable. Websites are no longer just static pages for displaying information—they now handle sensitive end-user data, conduct financial transactions, manage personal identification information, and support crucial business processes. This transformation has resulted in the exponential growth of web-based cyber threats. With the steady rise in these cyberattacks, it's become clear that many developers and organizations are falling short when it comes to implementing basic security practices.

A significant number of websites, particularly those built by individuals or teams lacking a strong security background, are vulnerable to widespread and dangerous attacks such as SQL Injection, Cross-Site Python-based vulnerability scanner created with the goal of identifying as many common web application vulnerabilities as possible. It's not designed to replace advanced, enterprise-level vulnerability scanners. Instead, it aims to give users - whether developers, students, or ethical hackers - a simple and effective way to detect potential security flaws and address them before they can be exploited. This scanner conducts a series of automated tests on a target URL to uncover threats like SQL Injection, XSS, command injection, directory traversal issues, and the absence or misconfiguration of important HTTP security headers. The tool works by injecting crafted payloads into URL parameters and analysing the responses returned by the server. For example, during a SQL Injection test, the scanner might add an input like "' OR '1'='1" into the URL, then check for typical SQL error messages or observe whether the server behaves in unexpected ways. Similarly, to detect XSS vulnerabilities, the tool might insert a harmless JavaScript payload, and if that code gets reflected back into the page response, it's a sign that the application is vulnerable to XSS.

One standout feature of the Webshield is its ability to scan and assess the presence of HTTP security headers. These headers are critical in defending against attacks like clickjacking, MIME sniffing, and reflected XSS. The scanner specifically looks for headers like Content-Security-Policy, X-Frame-Options, Strict-Transport-Security, X-XSS-Protection, and X-Content-TypeOptions. If any of these are missing, the scanner flags them and provides a clear explanation of why they matter, helping users understand how these headers contribute to the security of their applications. But the tool doesn't stop at detection-it also gives users practical, actionable recommendations for remediation. It doesn't just tell you something's wrong; it explains how to fix it.

For instance, if an XSS vulnerability is detected, the scanner might suggest sanitizing user inputs, using output encoding, and implementing a robust Content-Security-Policy. If any headers are missing, it also provides the exact syntax you should add to your server configuration. This turns the tool into not just a scanner, but also an educational aid, helping users learn while they secure their applications.

One of the most important strengths of the Webshield lies in its flexibility and ease of extensibility. Since it's built using Python, the code is fully open, readable, and modifiable. Users can build on top of it, add new scanning modules, integrate it into Dev SecOps pipelines, or customize it to suit their environment. This makes it an excellent fit for both classroom and professional settings. Cybersecurity students can use it to understand how real-world vulnerabilities are detected. Developers can use it as a first-level security check during development. Ethical hackers can even use it as the foundation for more advanced scanning tools or projects. To further improve usability, the scanner includes a feature that allows users to generate detailed summary reports after each scan. These reports provide a comprehensive breakdown of every test conducted—whether it passed or failed, what payloads were used, and what remediation steps are recommended. These reports are incredibly helpful for developers, as they give a clear picture of where issues lie and how to fix them. From a documentation and compliance standpoint, these reports are also valuable, as they help teams keep track of security improvements over time and ensure they're meeting required standards and regulations.

The Webshield manages to strike a solid balance between being powerful and easy to use. It's designed to be approachable for users at all levels of technical skill. There are no complex configurations needed to get started. As long as the user has a working Python environment and an internet connection, the tool can be run directly from the terminal. For those who prefer visual tools, the scanner can also be adapted into a graphical user interface or web dashboard, giving it more versatility and wider reach. Importantly, the tool was developed with a strong emphasis on ethical use. Webshield is strictly intended for scanning websites that the user owns or has explicit permission to test. Running scans on unauthorized websites can be illegal and is considered unethical. As cybersecurity threats grow more sophisticated, the consequences of unapproved testing can be serious. Therefore, this tool promotes transparency, legal compliance, and responsible usage. Practicing ethical cybersecurity is not just a legal necessity-it's a moral one too, especially in today's interconnected digital world. The Webshield was created in response to a growing demand for efficient, beginner-friendly, and practical web vulnerability tools. It sits in the middle ground between enterprise-level platforms and manual code reviews-offering an automated, yet understandable, method for scanning websites for security issues. It's packed with features that serve both experienced developers and curious learners alike. As the tool continues to grow and develop-potentially adding more advanced capabilities like CMS vulnerability detection, brute force login analysis, or AI-based behavioural pattern recognition-it will continue to bridge the gap between awareness and action in the field of web security. At its core, Webshield is more than just a tool-it's a step toward building safer digital spaces through education and proactive Défense. With help from community feedback, ongoing updates, and shared learning, it has the potential to make a meaningful impact on how developers and organizations approach web security. Whether you're a hobbyist, a student, a cybersecurity enthusiast, or a developer responsible for protecting user data, Webshield provides an accessible and empowering way to identify and fix security issues before they can be exploited.

## II. WORKING PROCESS

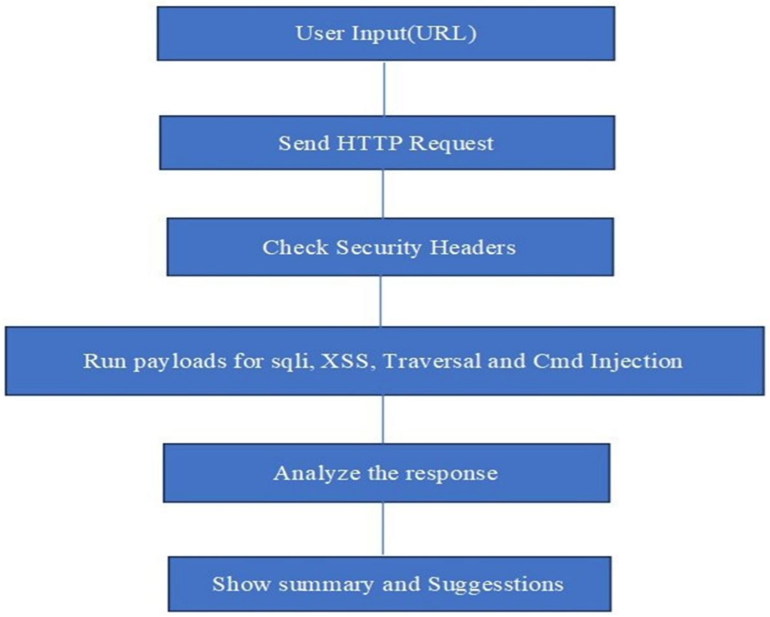
Due to limited technical knowledge or lack of awareness, many people-especially small business owners and independent developers-often overlook website security. This is exactly where Webshield steps in. It's a simple yet powerful tool designed to detect common website vulnerabilities before attackers can exploit them. But how does it really work? Let's break it down in a way that anyone, regardless of technical skill, can easily follow. To begin using Webshield, the user simply enters the URL of the website they wish to scan. This URL acts as the starting point for all the checks the scanner will perform. As soon as the scanner receives the input, it begins by analysing the HTTP security headers of the site. These headers are tiny bits of metadata sent from the server to the browser, and they play a critical role in protecting users from certain types of attacks. Once the header inspection is complete, the scanner proceeds to look for some of the most commonly exploited vulnerabilities in websites. One of the first issues it checks for is SQL Injection (SQLi). This type of attack occurs when a website unknowingly allows malicious users to tamper with its database queries. To test for this, the scanner sends crafted input like ' OR 1=1 -- into URL parameters and watches how the server behaves. If it notices an unusual error message or altered output, it may indicate that the site is open to SQL injection. After checking for SQLi, the scanner moves on to testing for Cross-Site Scripting (XSS). XSS attacks happen when a malicious user injects harmful scripts into a website, which then get executed in another user's browser. To detect this vulnerability, Webshield Scanner tries inserting harmless JavaScript snippets like <script>alert('XSS')</script> into input fields or URL parameters. If the scanner finds that the script is reflected back in the response page, that means the website is not properly filtering input, making it vulnerable to XSS. The next scan focuses on Directory Traversal vulnerabilities.

This issue arises when an attacker can manipulate file paths to access restricted parts of a server. For example, trying inputs like ../../etc/passwd could let an attacker read sensitive system files if the server is not properly secured. The scanner tests for these patterns and looks for signs that unauthorized data has been exposed, which would be a clear indicator of this vulnerability. Following that, Webshield runs a check for Command Injection vulnerabilities. This threat allows attackers to run system-level commands on the server by passing specially crafted inputs. Inputs like ; ls or | who am i are used by the scanner to test how the system responds. If it detects output that seems to be the result of a real system command, the scanner flags it as a potential command injection flaw. As the scan continues, Webshield logs all the findings in real time. It compiles the results into a detailed report that shows which tests passed and which ones exposed potential security risks. But what truly makes Webshield stand out is that it doesn't just stop at detection. It also provides clear, step-by-step recommendations on how to fix each issue it discovers. For example, if SQL Injection is detected, the tool may advise using prepared statements or parameterized queries and even show snippets of correct code. At the end of the scanning process, the tool generates a summary report for the user. This report outlines the results of each test-such as SQLi, XSS, directory traversal, and others-along with a brief explanation of the vulnerability, the specific input that triggered it, and a suggested fix. This is especially helpful for developers who may not be security experts but want to ensure that their websites are protected. One of the unique advantages of Webshield is that it's built using Python, a language known for being easy to read and widely supported. This makes it an ideal project for beginners, hobbyists, and professionals alike. Anyone with a basic understanding of Python can modify the tool, extend its features, or even integrate it into their automated development workflow-like in CI/CD pipelines. Since the tool runs in a simple terminal environment and doesn't require complex setup, it's accessible to students, freelance developers, and small teams. Another powerful aspect of Webshield is its vulnerability insights module. Instead of simply identifying a problem, this feature educates users about why the issue matters and how to prevent it in future development. It acts almost like a guide or mentor. For instance, when XSS is detected, the scanner might advise developers to validate user input, escape HTML content, and implement strong Content Security Policies (CSP). These suggestions don't just patch current vulnerabilities-they help users build long-term security awareness.

Webshield is lightweight and fast. It doesn't need high-end hardware or huge storage space. It simply runs the checks, analyses the responses, and produces a clear and concise report. Whether you're a university student working on a web project, a freelance web designer managing client sites, or a small business owner maintaining your company's online presence, this tool allows you to take control of your website's security with ease. In essence, Webshield operates as a dependable virtual assistant, helping you perform essential security audits on your website without requiring deep cybersecurity expertise.

All you need to do is input a URL, and the tool takes care of the rest-scanning for missing headers, injecting test payloads, analysing server responses, and finally presenting everything in an easy-to-understand report. Thanks to its intuitive design, practical tips, and straightforward usability, it becomes an essential tool for anyone aiming to build safer, more secure websites.

### *A. Flowchart*


User Input(URL)
Send HTTP Request
Check Security Headers
Run payloads for sqli, XSS, Traversal and Cmd Injection
Analyze the response
Show summary and Suggesstions

*1) User Input / URL Submission*

The process starts when the user inputs the URL of the target website they want to scan. This URL acts as the central point from which all vulnerability tests will be carried out. Before the scan begins, the tool performs input validation to make sure the URL is in the correct format and doesn't pose any immediate risk. This step ensures that users don't mistakenly scan an invalid link or unauthorized site.

Key Points:

- User submits a target web application's URL.
- Input is validated to ensure it's correctly formatted and safe.
- Acts as the entry point for all further security tests.

*2) HTTP Request & Response Handling*

After the URL is accepted and validated, the scanner sends an HTTP GET request to the server using the Python requests library. The server's response, including headers and the page content, is captured and stored for further analysis. This allows the scanner to observe how the web server reacts to standard and malicious inputs.

Key Points:

- HTTP GET request is sent to the URL.
- Response headers and body are captured.
- Data is stored for further vulnerability checks.

*3) Security Header Analysis*

Security headers play a vital role in protecting websites from known attacks like clickjacking, MIME-type sniffing, and XSS.

The scanner checks whether key HTTP headers are present in the response. These include Content-Security-Policy, StrictTransport-Security, X-Frame-Options, X-Content-Type-Options, and X-XSS-Protection. If any of these headers are missing or misconfigured, the tool logs them and indicates their severity level.

Key Points:

Scanner checks for critical security headers:

- Content-Security-Policy
- Strict-Transport-Security
- X-Frame-Options
- X-Content-Type-Options X-XSS-
- Protection

Missing headers are flagged and rated (High/Medium).

Results are added to the summary report.

*4) Payload-based Vulnerability Testing*

This is the core part of the scan. Webshield uses crafted payloads to test how the web application handles potentially malicious input. The scanner sends these payloads through URLs and input fields and then observes how the server responds. Based on that, it determines whether vulnerabilities exist.

Key Points:

SQL Injection (SQLi) Testing:

- Payloads like ' OR 1=1 -- and " OR "1"="1" are sent.
- If SQL errors or odd behaviours are detected, SQLi is flagged.

Cross-Site Scripting (XSS) Testing:

- Script tags like <script>alert('XSS')</script> are injected.
- If the script is reflected unescaped, it confirms XSS vulnerability.

Directory Traversal Testing:

- Paths like ../../etc/passwd or ..\..\windows\win.ini are
- used. If protected files are accessed or exposed, it's flagged.

Command Injection Testing:

- Payloads such as ; ls, | who am i are injected.
- Unexpected output or command execution reveals vulnerability.

*5) Vulnerability Detection & Suggestion Module*

When a vulnerability is found, Webshield doesn't just alert the user—it also provides clear recommendations on how to fix the issue. This makes it useful not only for security testing but also as an educational tool. These suggestions are tied to best practices in secure coding and web development.

Key Points:

Vulnerabilities are logged with specific triggering payloads.

Recommendations are provided for each vulnerability type:

- XSS  Sanitize input, use CSP, encode output.
- SQLi  Use parameterized queries, ORM frameworks.

Promotes secure coding practices.

## III. RESULTS AND DISCUSSION

The Webshield underwent rigorous testing on several platforms, including purposely vulnerable web applications like DVWA (Damn Vulnerable Web App), Web Goat, and custom-built sandbox environments. Through these tests, the scanner consistently demonstrated its capability to identify a variety of critical web application vulnerabilities. These included SQL Injection (SQLi), Cross-Site Scripting (XSS), Directory Traversal, and Command Injection, as well as the absence of vital HTTP security headers such as Content-Security-Policy, X-Frame-Options, Strict-Transport-Security, and others.

Each scan performed by the tool was accompanied by a comprehensive summary report, clearly outlining the presence or absence of each vulnerability, along with the specific payloads used, diagnostic descriptions, and actionable mitigation advice. For instance, when the scanner injected classic SQLi payloads like ' OR 1=1 --, it successfully identified applications that returned error messages or behaved abnormally, signalling a potential SQLi vulnerability. Similarly, for XSS testing, JavaScript payloads like <script>alert('XSS')</script> were used, and if these scripts were reflected back unfiltered in the server response, the application was flagged as XSS-vulnerable.

Directory traversal flaws were exposed by navigating to sensitive files such as /etc/passwd or windows/win.ini, and command injection vulnerabilities were identified when payloads like ; ls or | who am i triggered unexpected system-level outputs. Beyond detection, the scanner's built-in suggestion module added immense value by guiding users on how to fix each detected issuesuch as using parameterized queries for SQLi, applying input sanitization for XSS, or configuring proper access controls for sensitive directories.

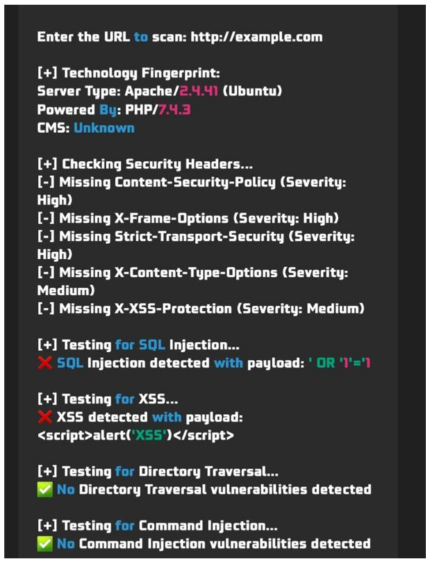
Another noteworthy outcome was the effectiveness of the HTTP header analysis module. Many modern web attacks rely on exploiting weak or missing security headers, and Webshield accurately highlighted their absence. For example, when headers like X-Content-Type-Options or X-XSS-Protection were missing, the scanner flagged them along with severity levels (e.g., High, Medium) and explained their significance, making it easy even for non-security professionals to grasp their importance. The accessibility of the tool is also a major achievement.

Developed using Python, the scanner is lightweight, cross-platform, and does not require any complex installation process or advanced system configurations. It can be run from the terminal with a single command, and its readable codebase allows for easy modifications, enabling users to add new payloads, tests, or integrations into CI/CD pipelines. This makes it an excellent fit for developers, students, ethical hackers, and QA engineers alike. The tool's performance was consistent across different testing environments, and its scan completion times were quick-usually taking just seconds to minutes, depending on the complexity of the site.

From a usability standpoint, the scanner's straightforward interface, simple input format, and clean report generation make it a practical solution for both real-world and educational use cases.

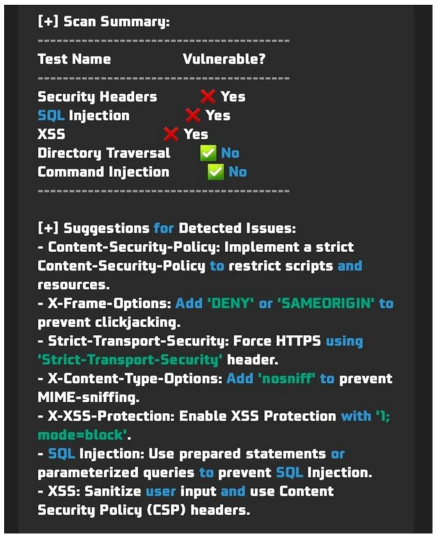
Moreover, the project highlighted the growing gap in secure coding practices, especially among small businesses and individual developers. During testing, even some newly built sites lacked basic protections such as security headers or input validation, exposing them to avoidable risks. This reaffirms the relevance and need for accessible tools like Webshield, which can act as a first line of Défense and awareness. While it doesn't replace enterprise-level scanners like Burp Suite or ZAP in terms of breadth and depth, its ease of use, educational value, and customizable nature make it a strong entry-level solution. Furthermore, the success of this project paves the way for future enhancements.

Potential upgrades include adding CMS-specific checks (e.g., for WordPress), brute force protection analysis, machine learning-based pattern recognition, deeper page crawling capabilities, and integration with vulnerability databases for more accurate risk scoring. In conclusion, the results demonstrate that Webshield is not only technically sound but also impactful in promoting web security best practices. It bridges the gap between awareness and action, empowering users to take control of their web application's security and build a safer digital ecosystem. *A. Sample Output*



```
Enter the URL to scan: http://example.com

[+] Technology Fingerprint:
Server Type: Apache/2.4.41 (Ubuntu)
Powered By: PHP/7.4.3
CMS: Unknown

[+] Checking Security Headers...
[-] Missing Content-Security-Policy (Severity:
High)
[-] Missing X-Frame-Options (Severity: High)
[-] Missing Strict-Transport-Security (Severity:
High)
[-] Missing X-Content-Type-Options (Severity:
Medium)
[-] Missing X-XSS-Protection (Severity: Medium)

[+] Testing for SQL Injection...
❌ SQL Injection detected with payload: ' OR '1'='1

[+] Testing for XSS...
❌ XSS detected with payload:
<script>alert('XSS')</script>

[+] Testing for Directory Traversal...
✅ No Directory Traversal vulnerabilities detected

[+] Testing for Command Injection...
✅ No Command Injection vulnerabilities detected
```

(a)



```
[+] Scan Summary:
----------------------------------------
Test Name              Vulnerable?
----------------------------------------
Security Headers          ❌ Yes
SQL Injection           ❌ Yes
XSS                  ❌ Yes
Directory Traversal       ✅ No
Command Injection          ✅ No
----------------------------------------

[+] Suggestions for Detected Issues:
- Content-Security-Policy: Implement a strict
Content-Security-Policy to restrict scripts and
resources.
- X-Frame-Options: Add 'DENY' or 'SAMEORIGIN' to
prevent clickjacking.
- Strict-Transport-Security: Force HTTPS using
'Strict-Transport-Security' header.
- X-Content-Type-Options: Add 'nosniff' to prevent
MIME-sniffing.
- X-XSS-Protection: Enable XSS Protection with '1;
mode=block'.
- SQL Injection: Use prepared statements or
parameterized queries to prevent SQL Injection.
- XSS: Sanitize user input and use Content
Security Policy (CSP) headers.
```

(b)

Fig 3.1 vulnerability detection and suggestions

Figure 3.1 indicate that Webshield effectively detects vulnerabilities and highlights areas for improvement. The inclusion of remediation suggestions sets it apart from many scanners, as it offers immediate next steps. This makes it particularly useful in educational settings and small dev teams.

## IV. CONCLUSION

The Webshield Scanner has turned out to be a really impactful tool in helping people—especially developers and small teams— spot security issues in their web applications before those issues can be exploited. In today's world, where cyber threats are becoming smarter and more frequent, having something simple yet powerful like Webshield really makes a difference. What makes this tool stand out is how it doesn't just detect problems like SQL Injection, XSS, Command Injection, Directory Traversal, or missing security headers—it also guides users with suggestions on how to fix them. So it's not just a scanner; it's like a digital buddy that checks your site and teaches you better security practices along the way.

One of the best parts during the development and testing process was realizing how beginner-friendly the tool actually is. You don't need to be a cybersecurity expert to use it. Anyone with basic Python knowledge can run the tool by entering a URL, and within seconds, they get results about possible threats. The scanner's report is easy to read and highlights the problems clearly along with recommendations. This makes it super useful not only for professional developers but also for students, ethical hackers, or even small business owners who maintain their own websites.
We also added smart touches like auto-suggestions for fixing issues and a neat summary report at the end of each scan, so you don't miss anything important. It even flags missing security headers and explains what they do, which is something that's often overlooked but plays a big role in protecting web users. Over time, this project showed us how a simple scanner could actually make a big difference, not just by finding flaws, but by raising awareness and encouraging better security habits in day-to-day coding.In short, Webshield Scanner isn't just another project—it's a tool made with real intention. It bridges the gap between complex security scanners and complete beginners who just want to protect their websites. It's quick to use, gives meaningful results, and most importantly, it's growing with room for more features like CMS vulnerability checks and smarter analysis. By putting this tool out there, we hope it becomes part of the bigger movement toward building a safer, stronger, and more responsible internet.

## V. CONFLICT OF INTEREST

Our goal is to enhance website security by identifying vulnerabilities at an early stage, offering practical fixes, and encouraging forward-thinking cybersecurity habits to help build a safer digital space for everyone.

## REFERENCES

[1] S. Bairwa, B. Mewara, and J. Gajrani, Vulnerability Scanners—A Proactive Approach to Assess Web Application Security, arXiv preprint arXiv:1403.6955, [2014].
[2] O. Ehichoya and C. C. Nnaemeka, Evaluation of Static Analysis on Web Applications, arXiv preprint arXiv:2212.12308, [2022].
[3] U.-S. Potti, H.-S. Huang, H.-T. Chen, and H.-M. Sun, Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an Example, arXiv preprint arXiv:2501.05907, [2025]
[4] S. B., S. N. R. K., T. J., and S. S., A Comparative Analysis of Vulnerability Management Tools: Evaluating Nessus, Acunetix, and Nikto for Risk Based Security Solutions,arXiv preprint arXiv:2411.19123, [2024].
[5] Application Security, Wikipedia: The Free Encyclopedia, [2023].
[6] OWASP, Wikipedia: The Free Encyclopedia, [2023].
[7] K. Zetter, Hacker Lexicon: SQL Injections, an Everyday Hacker's Favorite Attack, Wired Magazine, [2016].
[8] Wired Staff, XSS Vulnerabilities, Raw SQL Top List of Common Programming Errors, Wired Magazine, [2010].
[9] Wired Staff, 8 Out of 10 Software Apps Fail Security Test, Wired Magazine, [2011].
[10] OWASP Foundation, OWASP Top Ten, OWASP Project, [2021].
[11] MITRE Corporation, Common Weakness Enumeration (CWE), MITRE Technical Reference, [2022].
[12] National Institute of Standards and Technology (NIST), NIST SP 800-53: Security and Privacy Controls for Information Systems and Organizations,NIST Special Publication, Rev. 5, [2020].
[13] International Organization for Standardization, ISO/IEC 27001: Information Security Management Systems, ISO Standard, [2022].
[14] OWASP Foundation, Web Application Security Testing Cheat Sheet, OWASP Cheat Sheet Series, [2020].
[15] OWASP Foundation, SQL Injection Prevention Cheat Sheet, OWASP Cheat Sheet Series, [2021].
[16] OWASP Foundation, Cross Site Scripting Prevention Cheat Sheet, OWASP Cheat Sheet Series, [2021].
[17] Web Application Security Consortium (WASC), WASC Threat Classification,WASC Standards, [2020].
[18] Institute for Security and Open Methodologies (ISECOM), The Open Source Security Testing Methodology Manual (OSSTMM), ISECOM Documentation, [2022].
[19] Penetration Testing Execution Standard (PTES), PTES Technical Guidelines, PTES Documentation, [2021].
[20] Synopsys, Building Security In Maturity Model (BSIMM), BSIMM Technical Report, [2023].