Spencer Harman

9 July 2025

https://github.com/spencerharmann/VulerabilityScanner

Bridging the Gap Between Vulnerability Scanning and Actionable Intelligence

---

Abstract

Just about everything runs on a web app nowadays, so regulating security checks for them has become a critical task. While many scanners exist, they often fall into two categories: they're either too specialized to see the bigger picture, or too complex and resource heavy. This leaves a major gap between simply finding a potential weakness and actually understanding the risk it poses.

This paper introduces a web vulnerability scanner I designed in Python to bridge that gap. My tool uses a hybrid approach by combining simple information gathering with active tests for major and common vulnerabilities like SQL Injections and Cross-Site Scripting.

The key innovation of the scanner is its ability to provide instant context. When the scanner identifies a server banner or a piece of software, it searches the National Vulnerability Database (NVD) to see if the version detected has any Common Vulnerabilities or Exposures (CVEs). Testing on an intentionally vulnerable website that discloses its vulnerabilities showed that this method not only was able to effectively detect flaws but also relate them to thorough and documented threat intelligence, leading to a much more useful and actionable security report.

---

1. **Introduction**

Web applications have become the backbone of communication, commerce, and data exchanges, making their security an extremely important concern. Bad actors continuously try to find exploits in websites to compromise sensitive data, disrupt services, and inflict financial and reputational damage. The Open Web Application Security Project (OWASP) Top 10 provides an up-to-date list with the most crucial security risks including injection flaws, security misconfigurations, broken access control, and the usage of vulnerable and outdated components (OWASP).

The security industry has produced a wide variety of scanning tools to combat these kinds of threats, but most open-source tools (though very powerful) are often specialized, requiring security analysts to basically "chain" together separate tools for reconnaissance, SQLi tests, and XSS detection. On the other hand, commercial suites can be very expensive and don't provide clear insights on their methodologies. Burp Suite, for example, is an extremely powerful tool, but lacks transparency and operates like a "black box". An even more significant issue is the "intelligence gap": a scanner may report the presence of a specific version of software, but it leaves the analyst with a very tedious task of determining its association with publicly known exploits.

This paper addresses these challenges through a custom-built, hybrid web vulnerability scanner. The research aimed to answer the following: 1) How can a lightweight tool integrate passive discovery, active injection, and external threat intelligence? 2) How valuable is the automated enrichment of scan data with real-time CVE information?

The cultivation of this work is a web vulnerability scanner that fuses these different analysis techniques. By correlating discovered server banners with the National Institute of Standards and

Technology’s (NIST) Common Vulnerabilities and Exposure (CVE) Application Interface, the tool bridges the gap between raw data and actual intelligence, allowing developers and security teams to prioritize remediation efforts based on real-world risk (National Institute of Standards and Technology).

---

2. Background Check

An understanding of both the target landscape and the existing tools out there are essential. This scanner targets three of the most common (and dangerous) web application vulnerabilities:

- SQL Injection (SQLi): When an attacker exploits vulnerable input fields that don’t properly validate user input, which can allow them to view or even modify databases.
- Cross-Site Scripting (XSS): When an attacker injects malicious scripts into pages, allowing them to do things like stealing sensitive data, session hijacking, and more.
- Security Misconfigurations: An improper or incomplete implementation of security settings, including exposing sensitive files, enabling directory listings, or using default credentials.

The landscape of web vulnerability scanners is already diverse. Tools like Burp Suite and OWASP ZAP are powerful and widely used interception proxies, which provides immense control for manual testing but requires significant user interaction (Kali). Command-line tools like SQLMap are the “top dog” for SQLi detection, but don't test for any other vulnerabilities. Scanners like Nikto are amazing at finding server misconfigurations but perform limited injection testing. This project finds a middle ground by synthesizing these capabilities into a single, automated workflow, and relates findings to active CVE reports- a feature not included in these other tools.

---

3. System Design

The scanner is a modular Python application. The core of the tool is an asynchronous HTTP engine built using httpx and asynchio libraries, allowing it to handle multiple network requests at the same time. An httpx.AsynchClient is initialized, which allows the user to do an authenticated scan when login credentials are given by automatically handling session cookies. The system is made up of four primary analysis modules.

3.1. Passive Reconnaissance (scanInfoGathering)

The first phase of any scan involves low-impact reconnaissance. This module inspects the target's HTTP responses to understand the site. It grabs headers such as “Server” and “X-Powered-By” headers to identify server and application technologies. It also checks for the presence (and correctness) of security headers such as “Content-Security-Policy” and “Strict-Transport-Security” and checks for “Set-Cookie” attributes that might be exploitable.

3.2. Active Injection (scanSQLi, scanXSS)

These modules probe the application for vulnerabilities.

- SQL Injection: The scanSQLi function tests error based, Boolean blind based, and time-based SQL injections. It first tries error-based injections by attempting to trigger database errors using common payloads into GET parameters and POST form fields and checks the response body for indications. It checks for Boolean based blind injections by comparing the content length of responses to true and false conditions. If the “true” condition response length is similar to the baseline but the “false” condition is significantly different, a Boolean based SQL injection is inferred. It also checks for time-based injections, measuring response delays after injection SLEEP() or pg_sleep() commands. If the response time differs significantly, a time-based SQL injection is inferred.

- Cross-Site Scripting: the scanXSS module parses the target URL to find parameters and injects various XSS payloads. It inspects the HTML response to determine if the payload was reflected, which indicates a potential vulnerability.

3.3. Intelligence Correlation

This module is the scanner’s key innovation. When the program discovers a software version banner from the reconnaissance module, it triggers the following actions:

1. Extraction: The software name and version are extracted (e.g. “nginx”, “1.21.0”).
2. CPE Construction: A Common Platform Enumeration (CPE) virtualMatchString is constructed (e.g. cpe:2.3:a:nginx:nginx:1.21.0).
3. API Query: A request is made to the National Vulnerability Database API with the CPE string.
4. Reporting: The JSON response is parsed to extract all associated CVE’s, their descriptions, and their CVSS severity scores. This is then added to the final report.

---

4. Experimental Evaluation

To validate the program's effectiveness, a series of tests were conducted on http://testphp.vulnweb.com: an example PHP application that is intentionally made to be vulnerable to web attacks. It has built-in vulnerabilities such as Cross-Site Scripting, SQL Injections, security misconfigurations, and an outdated/vulnerable version of PHP.

Table 1: Summary of test results from http://testphp.vulnweb.com:

Vulnerability Class	Scanner used	Detected?	Details
SQL Injection	scanSQL	Y	Detected an error-based SQL injection with payload: "'".
Reflected XSS	scanXSS	Y	Detected a POST XSS injection with 8 different payloads
Information Gathering	scanInfoGathering	Y	Found a PHP version of 5.6.40 and nginx version of 1.19.0
Security Misconfigurations	scanSecurityMisconfiguration	Y	CSP header missing, admin/ directory listing is enabled, security headers are missing.
CVE Correlations	scanNVD	Y	Found 17 correlating CVEs to PHP 5.6.40.

These results confirm the scanner's effectiveness at identifying security misconfigurations, SQL injections, Cross-Site Scripting, security headers, and relating findings to real CVE reports. The CVE correlation with the PHP 5.6.40 finding demonstrates the value of being a hybrid tool, as it automatically turned a simple information disclosure into an actionable security alert.

---

5. Discussion

The experimental evaluation validates the original thesis of this project: that integrating active scanning with external threat intelligence provides a more thorough and complete security assessment. The scanner's ability to not only provide a thorough report of flaws on the website but to also immediately provide context about its real-world vulnerability (CVE) is an advantage it has over almost all other existing tools.

5.1. Limitations

Every tool has its limitations– the current implementation of the scanner does not include a crawler and can only scan URLs given by command line input. This means that the scanner can't find new links, pages, or API endpoints on the same domain, and could potentially cause it to miss vulnerabilities on pages like /admin.php unless the user specifically points the scanner at them. The user would have to run the scanner again on that new page. The scanner does not render JavaScript either, making it blind to vulnerabilities that exist within client-side code such as DOM-Based XSS. Furthermore, the XSS and SQLi payloads are from a static, fixed list. They are well known and can be blocked by Web Application Firewalls.

5.2. Future Work

The current architecture of the scanner allows for easy modularity. The scanNVD module, for example, was an extension added on top of an already functional vulnerability scanner. The next step for this scanner is to add a recursive web crawler to automate discovery and scanning of all reachable pages of an application. Following that, new modules could be created to detect other OWASP Top 10 vulnerabilities such as Stored Cross-Site Scripting, Server-Side Request Forgery, Insecure Deserialization, or Broken Access Control. An LLM trained on HTML responses could also be utilized using PyTorch to create more intricate payloads and exploit methods, further improving the scanner's robustness.

---

6. Conclusion

This paper has presented the design, implementation, and testing of a hybrid web vulnerability scanner. By combining multiple detection methodologies, the tool successfully identifies a range of critical web vulnerabilities. Its paramount function is the intelligence correlation with those vulnerabilities, fusing its findings with CVE reports from the National Vulnerability Database. The conjunction of active probing and external intelligence allows the scanner to produce more contextual, valuable, and actionable reports, demonstrating a powerful example for the next generation of automated security tools.

---

7. References

Kali. “Burpsuite.” *Kali Linux*, 20 May 2025, https://www.kali.org/tools/burpsuite/. Accessed 11 July 2025.

National Institute of Standards and Technology. “Vulnerability APIs.” *NVD*, 20 September 2022, https://nvd.nist.gov/developers/vulnerabilities. Accessed 11 July 2025.

OWASP. “OWASP Top Ten.” *OWASP Foundation*, September 2024, https://owasp.org/www-project-top-ten/. Accessed 11 July 2025.