International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 11 | Nov 2026 www.irjet.net p-ISSN: 2395-0072

Scanner4U: A Web-Based System for Preliminary Vulnerability Assessment, Security Scoring and Automated Audit Reporting

Suraj Mengade[1], Aditya Bande[2], Samadhan Rathod [3], PROF. R.S.Agarkar[4]

[1,2,3] (Students, Department of Computer Engineering), International Centre of Excellence in Engineering & Management

[4] (Professor, Dept. of Computer Engineering), International Centre Of Excellence in Engineering & Management

---

Abstract - *Web applications are widely used for delivering modern digital services, making their security an important concern. However, developers and students often lack accessible tools for performing preliminary security assessments and understanding the identified risks. This paper presents Scanner4U, a web-based system for preliminary vulnerability assessment, security scoring, and automated audit reporting. The proposed system performs website crawling and passive security checks across six categories: security headers, transport security, cookies, exposed files, forms and authentication, and information disclosure. Identified findings are classified according to severity and used to calculate an application-defined security score and corresponding grade. The system also provides scan history, comparison of consecutive assessments, and downloadable PDF reports containing findings and remediation recommendations. Experimental evaluation was conducted on three web targets. For one target, the system discovered 2,112 pages, analysed a sample of 30 pages, and completed the assessment in 3.3 seconds, identifying seven security findings and assigning a score of 75/100. Repeated scans produced consistent results. However, testing also revealed that insufficient crawl coverage could produce misleading security scores. Therefore, Scanner4U is intended as a preliminary security assessment tool and does not replace comprehensive vulnerability assessment or penetration testing.*

Index Terms - Web application security, vulnerability assessment, vulnerability scanning, security analysis, risk classification, security scoring, automated security reporting.

I. INTRODUCTION

Web applications now carry education, commerce, banking and communication, so a single weakness can expose user data or disrupt a service. Many weaknesses are not deep logic flaws but visible configuration gaps: missing HTTP security headers, weak cookie attributes, exposed files, or verbose error and version information. Professional testing finds these and more, but it is costly, and small teams and student developers rarely have access to it before deployment. Automated scanners lower the barrier, but their output is often a long, technical list that is hard to prioritise or share with non-specialists. The practical problem addressed here is therefore twofold: performing a safe, repeatable first-pass check, and presenting the outcome in a form that supports decisions, such as what is most serious, where it occurs, how to fix it, and whether things improved since the last scan.

This paper presents Scanner4U, a web-based preliminary vulnerability assessment and reporting system. Its contributions are: (i) a unified workflow from URL input to crawl, passive checks, severity classification and score; (ii) a per-category score breakdown that makes the grade explainable; (iii) scan history with fixed/still-open/new comparison between consecutive scans; (iv) automated PDF audit reports; and (v) an honest evaluation, including a failure case, of what such a tool can and cannot claim.

II. RELATED WORK

The OWASP Top 10 is the most widely used awareness document for critical web application risks [1], and the OWASP Web Security Testing Guide gives detailed test procedures [11]. NIST SP 800-115 describes planning, executing and reporting security tests, and stresses that results must be interpreted in the context of scope and limits [2]. Vulnerability identification and severity are standardised through the CVE programme [3] and the Common Vulnerability Scoring System [4].

On the tooling side, OWASP ZAP offers active and passive web scanning for security professionals [12], Nikto checks servers for dangerous files and outdated components [13], and the MDN HTTP Observatory grades a site's security headers and related settings [14]. These tools are powerful but typically present either raw technical output or a header-only grade. Scanner4U targets the gap between them: it combines several passive check categories with an explainable score, scan-to-scan comparison and a shareable report, aimed at developers and learners rather than specialist testers. Its score is application-defined and is not a CVSS score.

III. SYSTEM ARCHITECTURE

Scanner4U follows a layered design (Fig. 1). The presentation layer is a React single-page application. The scan engine fetches the target, reads robots.txt and sitemaps, follows internal links and runs the check modules. The analysis layer normalises findings and assigns severity; the scoring layer converts them into a 0–100 score and letter grade; and the reporting layer renders the dashboard, shareable report and PDF. Supabase provides authentication (Google or e-mail sign-in) and persistent scan history.

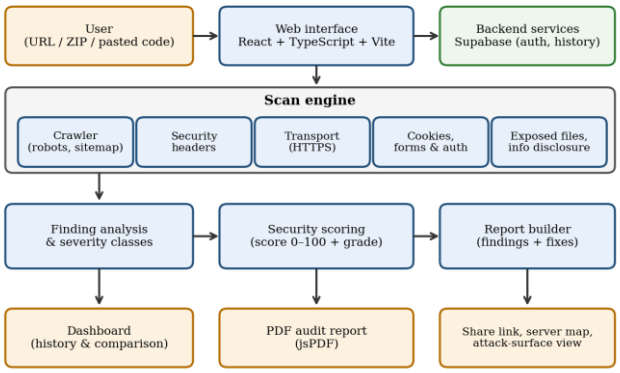


Fig. 1. High-level architecture of Scanner4U.

TABLE I. MAIN MODULES AND RESPONSIBILITIES

Module	Responsibility
Interface	URL / ZIP / pasted-code input, live scan view, report, dashboard
Crawler	Fetch robots.txt, discover sitemap and internal links, count pages
Check modules	Six passive categories (Section IV-B)
Analysis	Severity assignment, grouping, recommendations
Scoring	Overall score, grade and per-category percentage
Reporting	On-screen report, PDF export, share link, history comparison

IV. METHODOLOGY

A. Input and Crawling

The user selects one of three input modes: a website URL, an uploaded project archive (.zip), or pasted code. For a URL, the scanner fetches the target and its robots.txt, discovers sitemaps and follows internal links to enumerate public pages. Because a full crawl of a large site is expensive, a bounded sample of pages is analysed in depth while the total number of discovered pages is still reported (for example 30 of 2,112). Login and sign-up pages are identified and counted separately. The interface requires the operator to confirm they own the target or have explicit permission to test it.

B. Check Categories

Each scanned page and response is examined in six categories: (1) security headers, such as Content-Security-Policy, clickjacking protection and Referrer-Policy; (2) transport security, i.e. HTTPS use, redirects and mixed content; (3) cookies and their protective attributes; (4) exposed files that should not be publicly reachable; (5) forms and authentication, including CSRF-related indicators on login and sign-up forms; and (6) information disclosure through verbose headers or content. All checks are passive: they observe responses and do not attempt exploitation.

C. Severity Classification

Findings are labelled High, Medium, Low or Informational. High and Medium findings are weaknesses with meaningful security implications that should be prioritised; Low findings are hardening opportunities; Informational items are observations that may need review but are not confirmed vulnerabilities. Each finding carries a title, category, affected location, risk description and recommended fix.

D. Scoring and Grading

The system computes an overall score from 0 to 100 from the findings and their severities and maps it to a letter grade (for example 94 maps to A and 75 to B in our trials). A per-category percentage is shown next to the number of issues in that category, so a user can see which area lowers the grade. The score summarises only what was checked; it is not a proof of security.

E. Reporting and Comparison

Results are stored per target. When a target is scanned again, the dashboard compares the new result with the previous one and reports how many findings were fixed, remain open or are new. The report can be exported as a PDF containing target, timestamp, score, summary, findings, locations and recommendations, or shared through a link. Additional context panels show server location derived from the public IP address, an attack-surface view of entry points and third-party code, and a heuristic estimate of whether the site shows signs of AI-assisted construction.

V. IMPLEMENTATION

The front end is written in TypeScript with React and built with Vite [5], [6], [8]. PDF audit reports are generated client-side with jsPDF [7]. Supabase [9] supplies

authentication and the database for saved scans, and Vitest [10] is used for automated tests of the analysis and scoring functions. Figs. 2 and 3 show the entry screen and the live scan view, which streams a log of crawl and check stages and displays pages scanned and findings so far.



Fig. 2. Home screen with input modes (website URL, project ZIP, pasted code).

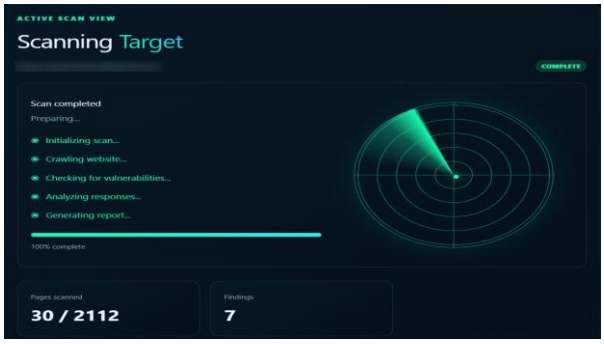


Fig. 3. Live scan view with progress, pages scanned, findings and scan log.

VI. EXPERIMENTAL RESULTS

A. Setup

Scanner4U was run between 26 and 29 October 2026 on three web targets: Target A, a public educational website; and Targets B and C, test deployments hosted behind development tunnels. Scans used the standard configuration without credentials. On Target A, the scanner discovered 2,112 pages, sampled 30 of them in depth and finished in 3.3 s. Target C scored 94 (grade A). Findings for Targets A and B are summarised in Table II.

B. Findings and Scores

TABLE II. SCAN RESULTS ON THE EVALUATED TARGETS

Metric	Target A	Target B
High	0	0
Medium	2	2
Low	3	3
Informational	2	1
Total findings	7	6
Score / grade	75 (B)	75 (B)

Scores and counts are taken directly from the application's dashboard.

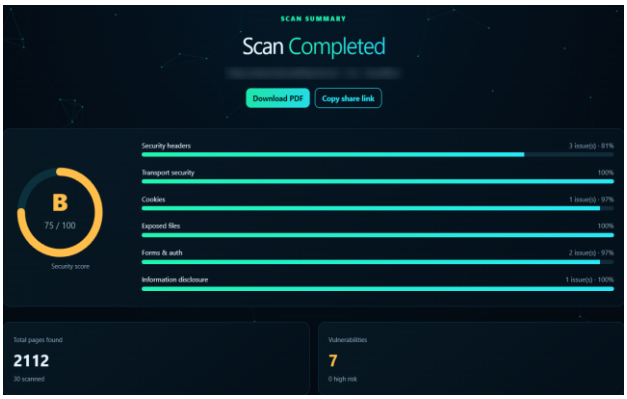


Fig. 4. Report summary for Target A: grade B (75/100) with category breakdown.

On Target A, all 7 findings fall in four categories: security headers (3 issues, 81%), forms and authentication (2 issues, 97%), cookies (1 issue, 97%) and information disclosure (1 issue); transport security and exposed files were clean (Fig. 5). The recurring problems were a missing Content-Security-Policy, missing clickjacking protection and a missing Referrer-Policy, all of which are inexpensive to fix by adding response headers. No high-risk finding was reported.

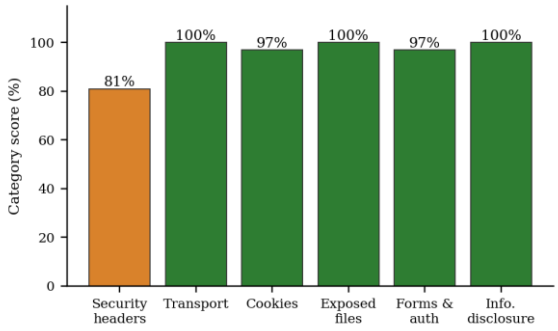


Fig. 5. Category scores for Target A.

C. Repeatability and History

Target A was scanned on 27 and 29 October 2026. Both scans produced grade B (75/100), and the dashboard reported 0 fixed, 7 still open and 0 new findings (Fig. 6). This agreement indicates that the checks are deterministic for an unchanged site, which is a prerequisite for using scan-to-scan comparison to verify remediation.



Fig. 6. Dashboard comparing the latest scan with the previous scan.

D. Context Panels

For Target A the report also resolved the server's public IP address to a location, hosting organisation and autonomous system, and displayed the result on a world map, alongside the heuristic AI-construction estimate (30%, "some AI-tool signs", based on utility-class patterns, marketing wording and placeholder text) (Fig. 7). These panels give reviewers context but are informational and are not part of the security score.



Fig. 7. Server location and AI-construction estimate panels (IP address obscured).

VII. DISCUSSION AND LIMITATIONS

Sampling coverage. Only 30 of 2,112 discovered pages (about 1.4%) were analysed on Target A. Findings that are site-wide, such as missing headers, are well captured by sampling, but page-specific weaknesses may be missed.

False sense of security. When a heavily bot-protected public service was scanned, the crawler obtained one page and scanned none, yet the report showed A+ (100/100) (Fig. 8). The score therefore reflected the absence of evidence rather than evidence of absence. We recommend coverage-aware grading: if pages scanned is zero or below a threshold, display "insufficient coverage" instead of a grade, and flag blocked requests as a finding.

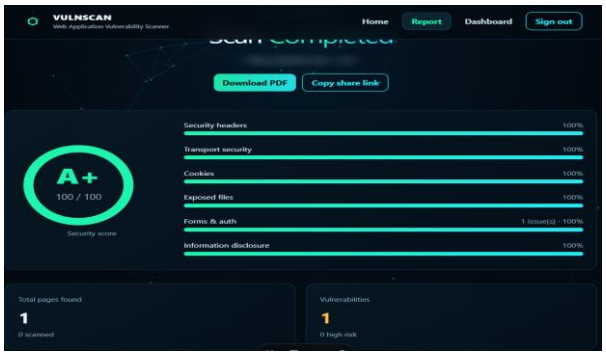


Fig. 8. Failure case: A+ (100/100) with one page found and none scanned.

Scope of checks. The checks are passive and cover configuration-level issues. Injection, broken access control and business-logic flaws are outside current coverage. False positives and false negatives are possible, and the score is application-defined rather than CVSS-based, so it should not be compared with other tools' scores.

Responsible use. Scanning must be restricted to systems the operator owns or is authorised to test. Scanner4U should be treated as a preliminary aid that complements, and does not replace, manual penetration testing [2], [11].

VIII. CONCLUSION AND FUTURE WORK

Scanner4U turns a preliminary website security check into an explainable, repeatable and shareable assessment: a crawl, six passive check categories, severity classification, a category-level score, history comparison and PDF reporting. Trials showed that it can assess a 2,112-page site in seconds, report actionable header and cookie issues, and give identical results on repeated scans. A failure case showed that coverage must be part of the score. Future work includes coverage-aware grading, mapping findings to OWASP Top 10 categories, finding verification to reduce false positives, authenticated scanning, larger evaluation on deliberately vulnerable test applications with known ground truth, and reporting detection rates against other tools.

ACKNOWLEDGMENT

The authors thank their guide, Prof.R.S.Agarkar, the Head of Department and the faculty of the Department of Computer Science and Engineering for their guidance and support.

REFERENCES

1) OWASP. (2025, July). OWASP Top Ten. Owasp.Org; OWASP. https://owasp.org/www-project-top-ten/

2) Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (2008, September 30). Technical guide to

information security testing and assessment. Csrc.Nist.Gov. https://csrc.nist.gov/pubs/sp/800/115/final

3) CVE. (2025). CVE-website. Www.Cve.Org. https://www.cve.org/

4) Common vulnerability scoring system SIG. (n.d.). FIRST - Forum of Incident Response and Security Teams. Retrieved October 31, 2026, from https://www.first.org/cvss

5) Meta Open Source. (2025). React. React.Dev. https://react.dev/

6) Microsoft, "TypeScript Documentation." [Online]. Available: https://www.typescriptlang.org/docs/

7) Parallax, "jsPDF: Client-side JavaScript PDF generation." [Online]. Available: https://github.com/parallax/jsPDF

8) VoidZero Inc., & Vite contributors. (n.d.). Vite | next generation frontend tooling. Vitejs. Retrieved October 31, 2026, from https://vite.dev

9) Supabase. (2025). Supabase docs. Supabase.Com. https://supabase.com/docs

10) Patak, V. A. F. H. O. A. P. J. S. (2026, June 2). Vitest. Vitest.Dev. https://vitest.dev/

11) OWASP. (2020, December 3). OWASP web security testing guide. Owasp.Org. https://owasp.org/www-project-web-security-testing-guide/

12) OWASP. (2020). Owasp Zap. Zaproxy.Org. https://www.zaproxy.org/

13) C. Sullo and D. Lodge, "Nikto web server scanner." [Online]. Available: https://github.com/sullo/nikto

14) Mozilla, "MDN HTTP Observatory." [Online]. Available: https://developer.mozilla.org/en-US/observatory

BIOGRAPHIES

1) PROF R. S. AGARKAR

FACULTY OF DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING INTERNATIONAL CENTER OF EXCELLENCE IN ENGGINERING & MANAGEMENT (ENGINEERING

2) SURAJ MENGADE

Pursuing (CSE) ICEEM INTERNATIONAL CENTER OF EXCELLENCE IN ENGGINERING & MANAGEMENT (ENGINEERING

3) ADITYA BANDE

Pursuing (CSE) ICEEM INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING & MANAGEMENT (ENGINEERING

4) SAMADHAN RATHOD

Pursuing (CSE) ICEEM INTERNATIONAL CENTER OF EXCELLENCE IN ENGINEERING & MANAGEMENT (ENGINEERING