International Journal of Computational Research in Science and Technology
E-ISSN: 3108-0871, Volume 2 Issue 1
Jan-Feb 2026, PP 14-21

Pentraguard - Web Vulnerability Scaner and Security Analyzer

Mrs. S. Revathi[1], Sabarinathan S[2], Sai Vigneshwaran A[3], Soundararajan T[4]

[1]Assistant Professor Department of Computer Science and Engineering Kongunadu College of Engineering and Technology Tamilnadu, India.

[2,3,4]Department of Computer Science and Engineering Kongunadu College of Engineering and Technology Tamilnadu, India

Abstract - PentraGuard is a Python-based ethical web vulnerability scanner designed as a Dynamic Application Security Testing (DAST) tool to evaluate the security posture of real-time web applications. It automates systematic testing to uncover common vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), insecure HTTP headers, and sensitive data exposure, aligning with the OWASP Top 10 security risks. The system functions in two core modes: Discovery Mode and Active Scan Mode. In Discovery Mode, the scanner safely analyzes and maps the application structure by identifying web pages, links, and input forms without using harmful payloads. Active Scan Mode performs controlled security assessments using predefined attack patterns to detect vulnerabilities. To enforce ethical usage, PentraGuard includes a domain ownership verification feature that restricts scanning to authorized or owned sites. The tool produces comprehensive vulnerability reports containing technical evidence, impact analysis, and remediation recommendations, enabling users to effectively mitigate identified risks.

Keywords - web vulnerability scanner, DAST, SQL Injection, OWASP Top 10, Discovery Mode , Active Scan Mode.

I. Introduction

Web applications have become fundamental components of modern digital ecosystems, enabling organizations to deliver services, manage data, and facilitate user interactions across the internet. However, the rapid growth of web-based systems has been accompanied by an escalation in cyber threats, particularly attacks that exploit vulnerabilities inherent in poorly secured applications. As a result, organizations require robust and automated security solutions to continuously assess and strengthen the resilience of their online platforms. PentraGuard is developed as a Python-based ethical web vulnerability scanning system designed to meet this critical need through the principles of Dynamic Application Security Testing (DAST). By focusing on the security posture of live web applications, PentraGuard enables real-time evaluation of potential weaknesses before malicious actors can exploit them.

The system targets widely known and high-impact vulnerabilities—including SQL Injection (SQLi), Cross-Site Scripting (XSS), insecure HTTP headers, and sensitive data exposure—based on selected categories from the OWASP Top 10 framework. These vulnerabilities represent significant risks to data confidentiality, integrity, and availability, making their early detection essential for maintaining secure digital infrastructures. PentraGuard operates using two complementary modes: Discovery Mode and Active Scan Mode. Discovery Mode performs a non-intrusive analysis by mapping web application structures, identifying pages, links, and input forms without deploying harmful payloads. In contrast, Active Scan Mode applies controlled attack patterns to evaluate actual security weaknesses, ensuring a comprehensive vulnerability assessment process.

To promote responsible and ethical usage, PentraGuard integrates a domain ownership verification mechanism, ensuring that scanning activities are restricted exclusively to authorized or

self-owned websites. This safeguards against misuse and aligns the system with legal and ethical standards of cybersecurity practice. The tool further enhances security management by generating detailed vulnerability reports that include evidence, impact assessments, and practical remediation strategies. Through its automated capabilities and ethical safeguards, PentraGuard serves as an effective and educational platform for improving the security posture of modern web applications.

II. Related Works

Bertoglio and Zorzo [18] systemically reviewed 54 primary studies using quality criteria to selected papers to determine reliability and credibility. The criteria grouped papers as 'Good', ' Very good ', and 'Excellent'. The study identified scanners used for penetration testing and their characteristics. Based on their analysis, 13 scanners were identified as the most cited ones. It also identified frameworks, methodologies, and security, testing models. Additionally, it analyzed the relationship between scanners and models besides some challenges of penetration testing. The researchers further identified process efficiency and effectiveness as critical challenges besides the vulnerability assessment process. Also, they noted that challenges in the analysis model and security scanners influence the security of scanners. Our research method is more extensive than the above study. We analyzed 320 studies in our work, and a total of 30 scanners were collected and identified in the paper. All of the returned scanners ' characteristics were provided based on what was indicated in the research papers and information available on the scanners' websites. Similarly, a survey study by Mirjalili et al. explored the applications of web penetration testing and its models and highlighted the comparison between web vulnerability scanners.

The survey reviewed previous literature on pen test methods and scanners and divided it into three categories. The first category examined and compared various methods and scanners. The second one proposed a new method or scanner for detecting vulnerabilities in web applications. The third category involved proposing a proper testing environment for executing web penetration testing. Moreover, the paper observed a correlation between 13 open-source and seven commercial scanners. It also noted that there are two core factors to judge the effectiveness and efficiency of the scanners. First is the "Structural Design" which deals with the GUI (Graphical User Interface), user ease, customization, and performance. The other key decision factor is the "Supported Features and Functionalities", which incorporate crawling techniques (automatic/manual), analysis techniques, auditing, and logging along with the generation of user reports.

The researchers found that some of the reviewed scanners had technical problems such as the inability to detect some types of attacks, such as stored SQLi and stored XSS attacks. Also, some scanners did not support new technologies and were incapable of detecting vulnerabilities attributed to application logic flows. In our survey, we identified 21 free/open source and 9 commercial scanners. In addition, our research covers additional aspects such as the developer that created the scanner, the technology utilized to design it, and the scanner's operating platform (e.g. Windows, Mac OS X or Linux). We also looked at the scanner's user interface, whether it was GUI or CLI. Furthermore, we included the availability of documents, such as the user manual and installation guide. Another study conducted by Kyriakos et al. reviewed existing literature on web vulnerability scanners. The researchers delved deep into fundamental open-source scanners and databases. They examined the web vulnerability of fundamental open-source scanners and databases by comparing them based on configuration, functionality, and support. The study also examined the scanners by comparing their accuracy of identifying vulnerability, errors in a web application, and their frequency. Moreover, it evaluated the functionality of the scanners based on categorization, vulnerability coverage, risk assessment inference, and counter-measuring. Besides, the researchers determined configuration using architecture, operation system support, level of usage, required resources, modularity, and access control mode. The researchers concluded that complete benchmarking of vulnerabilities, scanning strategy and workflow is essential to support the execution of the scanners. In comparison to this

study, our analysis is more complete because it covers the most prevalent commercial and open-source scanners, whereas this study solely focused on open-source scanners. Additionally, we examined and discussed common web vulnerability scanner features, whereas this study only addressed three: settings, functionality, and support. Furthermore, the performance of the researched scanners in finding vulnerabilities in web applications was not included in this study. However, we reviewed in-depth the findings of evaluation studies undertaken on these scanners, as well as the knowledge gap in this domain.

Kumar and Sheth conducted a review on the Zero-day vulnerabilities and the web application scanners that are used to detect these vulnerabilities in web services. The study explained different techniques used to detect and prevent zero-day vulnerability based on statistical-based methods, behaviour and signature-based methods, and hybrid techniques. The primary objective of each technique is to recognize the exploits' existence, eliminate them in real-time, and minimize the damage induced by the attack. One significant challenge is to ensure that the victim's machine threshold delay for analysis and quarantine is not exceeded. However, in some cases, this can cause undermining of the affected system. The researchers concluded that Zero-day attacks could misuse obscure vulnerabilities due to the absence or lack of antivirus, patches, and intrusion-detection signatures. To combat zero-day attacks, updating the system can disclose patches for most of the unknown vulnerabilities that were not detected during the system's development. In addition to that, the researchers suggested a robust framework designed to help the penetration tester detect and prevent zero-day vulnerabilities and remote code execution. Our research is thorough, and it includes information on all vulnerabilities identified by the Open Web Application Security Project (OWASP), including the OWASP Top Ten - 2010, the OWASP Top Ten - 2013, and the OWASP Top Ten - 2017. Furthermore, we looked into both the commercial and open- source scanners for detecting these security laws.

Seng et al. conducted another survey on the available methodologies used to assess web vulnerability scanners regarding test coverage, attack coverage, and vulnerability detection rate. It also highlighted the OWASP Top Ten vulnerabilities in web applications and the popular test-beds used to evaluate the web vulnerability scanners. In this study, the authors investigated some popular web vulnerability scanners, including Acunetix Web Vulnerability Scanner, BurpSuite, N-Sparker, Wapiti, W3af, Vega, Arachni, and Owasp Zap. Nevertheless, the paper could not answer some of the research questions aimed at quantifying the quality of web application security scanners. For instance, the suitable number of testbeds used to benchmark a web application security scanner remained unknown. It only showed that the number of testbeds used to benchmark web vulnerability scanners ranged from zero to thousands. Besides, the researchers did not specify measurement metrics used in describing the test coverage of web application vulnerability scanners, attack coverage, and vulnerability detection rate.

III. Proposed Method

System Overview:
The proposed system, PentraGuard, is a Python-based ethical web vulnerability scanning platform designed to evaluate the security posture of live web applications using Dynamic Application Security Testing (DAST) principles. The system integrates two core operational modes to balance safety and accuracy. Discovery Mode performs non-intrusive mapping of the target application by identifying accessible URLs, internal links, input forms, and server configurations without executing harmful payloads.

Active Scan Mode triggers controlled vulnerability tests based on predefined attack patterns to detect SQL Injection (SQLi), Cross-Site Scripting (XSS), insecure HTTP headers, and sensitive data exposure, aligning with the OWASP Top 10 standards. To enforce responsible use, PentraGuard includes a domain ownership verification process, ensuring scans are executed only on authorized or user-owned web assets. The system produces structured vulnerability reports with evidence, severity levels, impact analysis, and remediation steps, helping developers and administrators strengthen overall application security.

International Journal of Statistics and Mathematical Research Trends
E-ISSN: 3108-0871, Volume 2 Issue 1
Jan-Feb 2026, PP 14-21

System Architecture

The architecture of PentraGuard is designed to support ethical, automated, and modular web vulnerability assessment. It is structured into four major components: the User Interface Layer, the Verification Module, the Scanning Engine, and the Reporting Module. The User Interface Layer provides options for entering target URLs, selecting scan modes, and managing scan configurations. Before any testing begins, the Domain Ownership Verification Module validates authorization through methods such as token placement, email verification, or DNS-based confirmation to ensure ethical usage.

The central component, the Scanning Engine, operates in two modes. In Discovery Mode, the crawler navigates the website, collecting URLs, forms, scripts, HTTP headers, and structural metadata. It stores this information in a mapping database that guides further analysis. In Active Scan Mode, the engine injects predefined payloads to detect SQLi, XSS, insecure headers, and other OWASP-aligned vulnerabilities. The engine includes submodules such as the Request Handler, Payload Manager, Response Analyzer, and Risk Evaluator, which collaboratively process requests, execute tests, analyze responses, and classify security risks.

Finally, the Reporting Module compiles detected vulnerabilities into detailed reports containing technical evidence, severity ratings, and remediation suggestions. This architecture ensures scalability, accuracy, and controlled execution of security assessments.

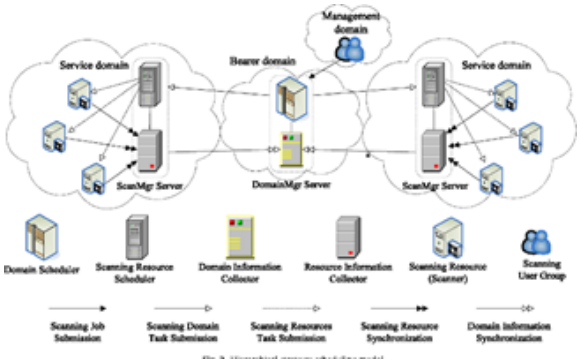

Fig.1. System architecture

Project Description

The first phase involves analyzing all functional and non-functional requirements needed to build PentraGuard. This includes identifying the types of vulnerabilities to be detected, selecting scan modes, defining reporting features, and establishing user authorization controls. Ethical usage is prioritized through the inclusion of a domain ownership verification mechanism to prevent unauthorized scanning. Technology choices, API constraints, compliance with OWASP guidelines, and performance expectations are also finalized during this stage.
The second phase focuses on designing the complete system architecture. This includes planning the User Interface Layer, Domain Verification Module, Scanning Engine, and Reporting Module. Flow diagrams, data mapping, and the interactions between components are defined. Both Discovery and Active Scan workflows are outlined, along with the integration between the crawler, payload executor, and response analyzer, ensuring modularity, scalability, and secure operation.

The third phase emphasizes implementation and integration. All core components are developed using Python, including the web crawler for Discovery Mode and the predefined payload-based testing for Active Scan Mode. The Domain Verification Module is integrated to ensure authorized usage. Submodules such as the request handler, scan core, payload manager, and analyzer are connected, and unit tests are conducted to validate each component.
The final phase includes comprehensive testing, reporting, and deployment. The system is tested in controlled environments to confirm accuracy, security, and ethical compliance

while minimizing false positives. Once validated, detailed vulnerability reports are generated with severity ratings, evidence, and mitigation recommendations. Deployment is then carried out in a secure environment, supported by documentation and user guidelines, with provisions for ongoing updates and maintenance aligned with OWASP standards.

Overall Working Flow of the Proposed System:
The overall working flow of the proposed PentraGuard system begins with the user entering the target website URL and selecting the desired scanning mode. Before any scanning activity is initiated, the system performs domain ownership verification using methods such as DNS validation, token-based verification, or email confirmation to ensure that the user is authorized to assess the selected web application.

Once authorization is confirmed, the system proceeds to Discovery Mode, where the crawler safely navigates through the target site, gathering structural information such as links, pages, forms, scripts, and HTTP headers without executing harmful inputs. This information is stored in a mapping database and forms the basis for deeper analysis.

After the discovery phase, the user may initiate Active Scan Mode, during which the system applies predefined attack payloads to detect common vulnerabilities, including SQL Injection, Cross-Site Scripting, insecure headers, and sensitive data exposure. The request handler sends crafted inputs, and the response analyzer evaluates returned data for unusual patterns, errors, or security weaknesses aligned with OWASP standards.

Once scanning is complete, the system compiles results into a detailed vulnerability report containing evidence, severity levels, impact explanations, and remediation guidelines. The final output helps users understand risks and implement corrective actions to strengthen web application security.

Performance Evaluation
The performance of PentraGuard was assessed to measure its effectiveness, efficiency, accuracy, and ethical compliance as a Dynamic Application Security Testing (DAST) tool. This evaluation examined the tool's ability to accurately identify vulnerabilities, its operational impact on target web applications, and its suitability for real-time security testing.

PentraGuard exhibits strong vulnerability detection capabilities, successfully identifying common web security issues such as SQL Injection (SQLi), Cross-Site Scripting (XSS), insecure HTTP headers, and exposure of sensitive information. By adhering to the OWASP Top 10 security standards, the scanner maintains relevant and comprehensive coverage of critical risks. The use of controlled and predefined payloads in Active Scan Mode helps reduce false positives while ensuring dependable detection results.

Regarding execution efficiency, Discovery Mode performs passive analysis to map application components such as pages, links, and input forms without affecting normal application functionality. This non-intrusive approach enables accurate target identification for further testing. Active Scan Mode is designed to achieve an effective balance between scan thoroughness and execution time, allowing detailed assessments without placing excessive load on server resources.

In terms of system resource usage, PentraGuard operates with moderate CPU and memory requirements, making it suitable for deployment on standard computing environments. Network traffic is carefully regulated to prevent unintended service disruptions during scanning.The inclusion of an ethical control mechanism, particularly domain ownership verification, ensures that scans are limited to authorized targets, promoting responsible use.

Furthermore, the detailed vulnerability reports—containing technical evidence, impact analysis, and remediation recommendations—support efficient mitigation efforts.Overall, the

evaluation indicates that PentraGuard is an efficient, reliable, and ethically sound DAST tool for securing modern web applications.

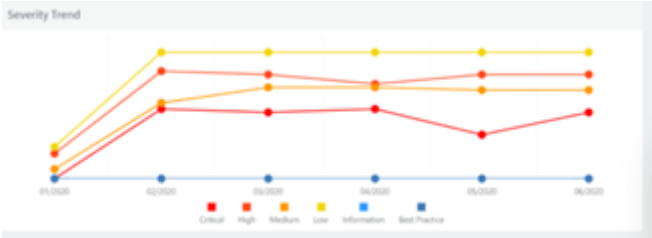

Fig.6. Performance analysis

Survey Goals:
There is currently no guidance in properly selecting the right tools to identify application vulnerabilities. Such a selection is usually performed via a manual process that can involve searching the web to find the right tool, inspecting the tool features and possibly getting in contact with the tool provider to find the most suitable pricing model matching the application risk requirements.

A structured approach was adopted for the tool search and selection process, consisting of two primary phases: (a) identifying and exploring available tools, and (b) choosing the most suitable tools from the identified options based on predefined inclusion criteria.
The search followed a dual-approach strategy. In both approaches, a carefully selected set of highly relevant keywords was used. This keyword set was slightly adjusted based on the specific artefact being examined, whether a database or a tool.

Results and Discussion

The implementation of PentraGuard demonstrated effective performance in identifying common web application vulnerabilities while maintaining ethical and controlled scanning practices. During testing on authorized test web applications, the scanner successfully detected vulnerabilities aligned with the OWASP Top 10, including SQL Injection (SQLi), Cross-Site Scripting (XSS), missing or misconfigured HTTP security headers, and instances of sensitive data exposure. The Discovery Mode efficiently mapped application structures by extracting URLs, input fields, and forms without injecting malicious payloads, ensuring zero impact on application availability and integrity.

In Active Scan Mode, PentraGuard applied predefined and safe attack patterns to identified inputs. The results showed high accuracy in detecting reflected XSS and basic SQL injection flaws, supported by clear technical evidence such as payload reflections and anomalous server responses. The separation of discovery and active scanning reduced false positives and allowed targeted vulnerability assessment, improving overall scan reliability. Additionally, the domain ownership verification mechanism successfully restricted scans to authorized targets, reinforcing ethical usage and preventing misuse.

From a usability perspective, the generated vulnerability reports were comprehensive and well-structured, including vulnerability descriptions, severity levels, proof-of-concept evidence, potential impact, and actionable remediation recommendations. This makes the tool suitable not only for security professionals but also for developers and students seeking to understand web security flaws.

However, the current implementation focuses primarily on common and low-to-medium complexity vulnerabilities. Advanced issues such as logic flaws, authentication bypass, and complex injection techniques remain outside the present scope. Overall, the results indicate

that PentraGuard is a reliable and ethical DAST tool for foundational web security assessments, offering a balanced approach between automation, safety, and practical security insights.

IV. Conclusion

PentraGuard successfully demonstrates the effectiveness of an ethical, automated approach to web application security testing through its Dynamic Application Security Testing (DAST) framework. By combining structured application discovery with controlled active scanning, the tool provides a comprehensive assessment of real-time web applications without disrupting normal operations. Its ability to identify critical vulnerabilities such as SQL Injection, Cross-Site Scripting, insecure HTTP headers, and sensitive data exposure ensures strong alignment with the OWASP Top 10 security risks, making it highly relevant for modern web security needs.

The separation of functionality into Discovery Mode and Active Scan Mode enhances both safety and accuracy, allowing applications to be thoroughly mapped before any attack patterns are executed. Additionally, the inclusion of domain ownership verification reinforces responsible and ethical usage, preventing misuse of the scanner on unauthorized targets. The detailed vulnerability reports generated by PentraGuard, which include technical evidence, impact evaluation, and remediation guidance, empower developers and security teams to take informed corrective actions.

Overall, PentraGuard serves as a practical and reliable security assessment tool that bridges the gap between automated vulnerability detection and ethical cybersecurity practices. It not only strengthens the security posture of web applications but also promotes secure development and deployment by enabling early identification and mitigation of potential threats.accuracy. Additional extensions such as dashboard-based visualization, automated reporting, and integration with enterprise energy management systems can further increase industrial applicability. Overall, the proposed system offers a scalable, intelligent, and software-driven solution for industrial carbon footprint prediction and sustainability-oriented decision support.

References

1. A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna, ‘‘Enemy of the state: A state-aware-black-box web vulnerability scanner,’’ in Presented at the 21st USENIX Secur. Symp. (USENIX Secur.), Aug. 2024
2. M. Agarwal and A. Singh, Metasploit Penetration Testing Cookbook. Birmingham, U.K.: Packt, 2023.
3. A. Akbulut, ‘‘VinJect: Toolkit for penetration testing and vulnerability scanning,’’ Düzce Üniversitesi Bilim ve Teknoloji Dergisi, vol. 6, no. 4, pp. 779–790, Apr. 2022. [Online]. Available: https://dergipark.org.tr/en/ download/article-file/517130
4. M. S. Aliero and I. Ghani, ‘‘A component based SQL injection vulnerability detection tool,’’ in Proc. 9th Malaysian Softw. Eng. Conf. (MySEC), Dec. 2021, pp. 224–229, doi: 10.1109/MySEC.2015.7475225.
5. M. S. Aliero, I. Ghani, K. N. Qureshi, and M. F. Rohani, ‘‘An algorithm for detecting SQL injection vulnerability using black-box testing,’’ J. Ambient Intell. Humanized Comput., vol. 11, no. 1, pp. 249–266, Jan. 2020, doi: 10.1007/s12652-019-01235-z.
6. M. Alsaleh, N. Alomar, M. Alshreef, A. Alarifi, and A. Al-Salman, ‘‘Performance-based comparative assessment of open source web vulnerability scanners,’’ Secur. Commun. Netw., vol. 2017, May 2017, Art. no. 6158107.
7. N. Antunes and M. Vieira, ‘‘Comparing the effectiveness of penetration testing and static code analysis on the detection of SQL injection vulnerabilities in web services,’’ in Proc. 15th IEEE Pacific Rim Int. Symp. Dependable Comput., Nov. 2009, pp. 301–306, doi: 10.1109/PRDC.2009.54.

8. N. Antunes and M. Vieira, ‘‘Detecting SQL injection vulnerabilities in web services,’’ in Proc. 4th Latin-American Symp. Dependable Comput., Sep. 2009, pp. 17–24, doi: 10.1109/LADC.2009.21.
9. N. Antunes and M. Vieira, ‘‘Benchmarking vulnerability detection tools for web services,’’ in Proc. IEEE Int. Conf. Web Services, Jul. 2010, pp. 203–210, doi: 10.1109/ICWS.2010.76.
10. N. Antunes and M. Vieira, ‘‘Enhancing penetration testing with attack signatures and interface monitoring for the detection of injection vulnerabilities in web services,’’ in Proc. IEEE Int. Conf. Services Comput., Jul. 2011, pp. 104–111, doi: 10.1109/SCC.2011.67.
11. N. Antunes and M. Vieira, ‘‘Defending against web application vulnerabilities,’’ Computer, vol. 45, no. 2, pp. 66–72, Feb. 2012, doi: 10.1109/MC.2011.259.
12. N. Antunes and M. Vieira, ‘‘Designing vulnerability testing tools for web services: Approach, components, and tools,’’ Int. J. Inf. Secur., vol. 16, no. 4, pp. 435–457, Jun. 2016, doi: 10.1007/s10207-016-0334-0.I. Jolliffe and J. Cadima, “Principal component analysis: A review and recent developments,” Philosophical Transactions of the Royal Society A, vol. 374, no. 2065, pp. 1–16, 2016.
13. L. Auronen, ‘‘Tool-based approach to assessing web application security,’’ Helsinki Univ. Technol., Espoo, Finland, Tech. Rep. T-110.501, 2002, vol. 11, pp. 12–13. [Online]. Available: http://citeseerx.ist.psu.edu/ viewdoc/download? doi=10.1.1.104.893&rep=rep1&type=pdf
14. N. F. Awang and A. A. Manaf, ‘‘Detecting vulnerabilities in web applications using automated black box and manual penetration testing,’’ in Proc. Int. Conf. Secur. Inf. Commun. Netw. Cairo, Egypt: Springer, Sep. 2013, pp. 230–239, doi: 10.1007/978-3-642-40597-6_20.
15. T. Basso, P. C. S. Fernandes, M. Jino, and R. Moraes, ‘‘Analysis of the effect of Java software faults on security vulnerabilities and their detection by commercial web vulnerability scanner tool,’’ in Proc. Int. Conf. Dependable Syst. Netw. Workshops (DSN-W), Jun. 2010, pp. 150–155, doi: 10.1109/DSNW.2010.5542602.
16. J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, ‘‘State of the art: Automated black-box web application vulnerability testing,’’ in Proc. IEEE Symp. Secur. Privacy, May 2010, pp. 332–345, doi: 10.1145/1135777.1135817.
17. E. Bazzoli, C. Criscione, F. Maggi, and S. Zanero, ‘‘XSS PEEKER: Dissecting the XSS exploitation techniques and fuzzing mechanisms of blackbox web application scanners,’’ in Proc. 31st IFIP Int. Inf. Secur. Privacy Conf. (SEC). Ghent, Belgium: Springer, May 2016, pp. 243–258, doi: 10.1007/978-3-319-33630-5_17.
18. D. Dalalana Bertoglio and A. F. Zorzo, ‘‘Overview and open issues on penetration test,’’ J. Brazilian Comput. Soc., vol. 23, no. 1, Dec. 2017, doi: 10.1186/s13173-017-0051-1.