

Enhanced web application security through advanced penetration testing techniques

Dolantina HYKA

Mediterranean University of Albania, Tirana, Albania
dolantina.hyka@umsh.edu.al

Festim KODRA

Mediterranean University of Albania, Tirana, Albania
festim.kodra@umsh.edu.al

Daniel ÇIKA

Mediterranean University of Albania, Tirana, Albania
daniel.cika@umsh.edu.al

Abstract

The paper considers how penetration testing can be conducted in a structured way to identify critical vulnerabilities in web applications and to enable an ongoing security improvement process. Literature background: The paper is built on the basis of the OWASP Top 10 and the practices of conducting a penetration test, but positioned in the context of modern SCRD cybersecurity research, which focuses on practical risk-based approaches to cybersecurity. Approach: The methodology includes the literature review in the sphere of web application security and practical assessment of a certain web application. The procedure of testing includes such steps as reconnaissance, scanning, verification, exploitation, impact assessment, reporting, and recommendations for mitigation of identified problems. The testing is conducted with the use of such tools as Burp Suite, Nmap and manually. Results: The assessment and analytical framework concentrate on SQL injection, cross-site scripting, cross-site request forgery, vulnerabilities of plugins and components, zero-day vulnerabilities, and security issues associated with open-source software. It is found that automatic scanning is not enough for comprehensive security assessment and manual verification is needed to assess exploitability and possible impact. Implications: The findings provide organizations, developers, and security departments with recommendations about secure coding, continuous testing, timely updates of components, cryptographic security, training employees and other processes of managing vulnerabilities. Value: The paper provides an analytical framework for testing and mitigating web application vulnerabilities by associating them with testing steps, security controls, and organizational implications. It also stresses the importance of continuous improvement of security, rather than one-time testing. As the evidence base for the paper does not have reproducible numbers of identified vulnerabilities and information about the target architecture, the manuscript provides the qualitative evidence only.

Keywords: Ethical Hacking; OWASP Top 10; SQL Injection; XSS; Vulnerability Management

1. Introduction

Web applications play a crucial role in digital services and are integral to many business processes and customer interactions. The multiplicity of functionalities, exposure to the Internet, dependency on databases, authentication, third-party components, and client-side technologies result in a considerable, ever-growing attack surface, where vulnerabilities may have implications for confidentiality, integrity, availability, and trust. The original study focuses on addressing this challenge by analyzing penetration testing as a practical solution for identifying vulnerabilities and deriving control measures from the technical findings.

The research is based on the lack of connection between awareness about the most common vulnerabilities in web applications and the information required for evaluating the effectiveness of the controls implemented in a certain application. The presence of a list of vulnerabilities per se does not prove the safety of an application. The Web Security Testing Guide by OWASP highlights the importance of consistency, reproducibility, rigour, documentation, and linking of the detected vulnerabilities with consequences and measures [1], [2]. This makes it possible to apply a more methodologically sound approach to the practical nature of the current study.

Recently published papers in the SCRD security community focus on practical assessment of cybersecurity, risk management and consequences of compromising data [3], [4]. The current paper continues the same line, providing an extension to the practical approach for assessing web application security through connecting vulnerability-oriented review with penetration testing. The contribution of the current study is not in a novel vulnerability detection algorithm. Instead, it presents a systematic framework for performing manual and automated testing, analysis of its results, and derivation of the control measures.

It should be noted that the scale of the problem under discussion is evident. According to Positive Technologies' 2019 assessment of production web applications, 82% of the detected vulnerabilities in tested systems came from application source code, not from incorrect configuration of servers [12]. Moreover, Verizon's 2025 Data Breach Investigations Report includes Basic Web Application Attacks among the patterns of incidents detected across the population of breaches analyzed [17]. Although these statistics do not relate to the application considered in the source manuscript since the application does not include vulnerability counts in its record, it helps to see that the problem discussed in the current paper is an ongoing, well-documented issue.

1.1. Research question and objectives

This research will be conducted based on the research question below: "How can penetration testing be used effectively for detecting and resolving web application vulnerabilities and how can security be continuously improved?"

Objectives include: (a) examination of common web application vulnerabilities, focusing especially on the OWASP top 10; (b) conducting a penetration test process for a web application; (c) discussion of vulnerability types such as SQL injection and cross site scripting; (d) linkage of vulnerabilities to the methods used in overcoming them; and (e) creation of a checklist framework.

In intelligent and adaptive communities, the security of web applications has much to do with continuity and accessibility of public services. The vulnerabilities in municipal websites, e-governance systems, urban IoT interfaces, or other forms of digital public services can put users' information at risk or cause problems in the continuity of these services or even their credibility. Such problems can also influence the transport infrastructure and mobility services, which will prevent residents from accessing the labor

market or services provided in the territory. It is therefore clear that such risks are important in relation to resilient communities since the continuity of these services could contribute towards inclusive development in the region.

2. Related work and conceptual framework

The literature identified in the original manuscript acknowledges web application security as an ongoing problem and OWASP Top 10 as the popular document for conveying common application-security risks. The re-framing makes use of the OWASP Top 10:2021 taxonomy where appropriate, while also separating the OWASP Top 10 from an overall penetration-testing approach. As OWASP itself states, the WSTG should be used for technical testing and not considered a comprehensive testing checklist [2], [5], [8].

Penetration testing is important since it provides for more than simple identification and can prove that a potential vulnerability is actually exploitable. WSTG defines security testing as an active process of evaluating application controls and emphasizes the need to document testing efforts, impact, and proposed solutions [2]. Similarly, the original manuscript utilizes manual testing together with tools like Burp Suite and Nmap and structures the practical process into reconnaissance, scanning, exploitation, post-exploitation, reporting, and remediation.

Previous literature has discussed issues related to security and privacy within the context of the Internet of Things (IoT). They have also discussed the potential use of blockchain and machine learning in solving such problems [6]. Previous research on the topic of web application security has also used automatic tools and manual pen-testing to study the TLS/SSL setup, key management, and usage of deprecated cryptographic algorithms [7]. Both these researches have provided an important basis for the framework of this paper.

Work done on automated phishing site detection has explored the relationship between user perceptions of trust in security software [18]. It has relevance to the current study through analogy, where automated scans can guide further investigation, but manual verification must be done to see if a weakness is exploitable.

The research into explainable recommendation systems has investigated how design of explanation can help users calibrate their trust in recommendation [19]. By analogy, penetration-test report must present rationale for each conclusion in order to enable administrator to evaluate recommended remedy.

The conceptual framework used in this research paper thus consists of four interdependent layers: identification of threat, validation of it on the technical level, assessment of the potential impact and remediation/continuous improvement. This helps to overcome a major drawback of many descriptive studies of security issues – separation of vulnerability description from actual evidence and practical recommendations. A vulnerability is defined not only as a technical issue, but as a condition that may impact the organization's security objectives and business continuity.

2.1. Vulnerability classes considered

SQL injection is taken into account since improper input validation would enable attackers' data to manipulate database queries, leading to confidentiality and integrity issues. XSS attack is also considered due to unsafe handling of user-provided content that could result in executing malicious scripts in the victim's browser. CSRF attacks are also included due to the possibility of making unintended actions performed by authenticated users due to lack of request validation control mechanisms.

Vulnerable and outdated components and plugins are also taken into account, especially third-party software that has some vulnerabilities. A zero-day vulnerability is a unique type of attack since the newly discovered or publicized vulnerability does not have an available fix yet. Components based on open source are not necessarily insecure. However, it is essential to manage dependencies, monitor them, configure, and update. The list of components was maintained from the original manuscript.

2.2. Cryptographic protection as a complementary control

Vulnerability testing takes into account how the application processes the requests and inputs, but the confidentiality of the data while it is being transferred and stored relies on the cryptographic techniques chosen by developers independently of the OWASP Top 10 vulnerabilities categories. The research by Hyka and Baxhaku on the development of diagnostic cryptographic examples with the help of the software Maple is a good example of how cryptographic primitives immune to simple cryptanalysis can be constructed and examined [9]. Another study conducted by the same researchers identifies the main cryptographic problems, including key management issues and obsolete cipher configurations, found in ICT services in developing countries, and claims that most often, those problems are organizational rather than mathematical in nature [9]. When combined with another study of data protection practices in public and private administration [10], such research can contribute to viewing cryptographic security, as well as input validation and access control, as the regular elements of penetration testing, not the additional issue to be considered afterwards.

2.3. Conceptual testing framework

The proposed framework clearly shows how each test relates to the evidence that needs to be collected. Reconnaissance gives the actual attack surface; scanning detects possible weaknesses; manual validation determines if the information gathered was real and significant; exploitation determines the possible impacts of such a vulnerability within the authorized scope of action; and reporting relates technical evidence to remediation priorities. Continuous testing, in turn, gives feedback for the next assessment cycle.

The framework aligns well with OWASP's model of testing, which divides passive knowledge about the application into active security testing, and suggests a proper balance of manual analysis, threat modeling, code review, automated testing, and penetration testing depending on the situation [1], [2]. Thus, the framework is more of a research tool, and not an attempt to prove that one particular method or tool can find all vulnerabilities.

2.4. Standards and methodological frameworks

The OWASP WSTG is not the only methodology-based framework available in the domain of security testing, and comparing it with some other prevalent frameworks makes clear what the methodology presented in this paper borrows from where and what it consciously avoids borrowing from. NIST Special Publication 800-115 is a technology agnostic security testing methodology that focuses on planning, discovery, attack, and reporting tasks and is often cited in compliance-based testing efforts [13]. PTES stands for the Penetration Testing Execution Standard and offers a seven-stage methodology that emphasizes planning activities such as pre-engagement scoping and threat modeling prior to technical testing [20]. MITRE ATT&CK serves to complement both of the above by providing a repository of adversary tactics and techniques applied post-compromise and thereby assists in interpreting what a successful exploitation stage means in terms of the overall threat model of an organization, not just the discovery stage [14]. ISO/IEC 27001 contextualizes security testing in the context of a management system in which testing frequency and remediation responsibilities are associated with a documented information security management system and not as a standalone technical task [16]. Table 1 highlights these relationships.

Table 1. Complementary security-testing frameworks and standards

Framework	Primary focus	Where it complements this study
OWASP WSTG	Web-application-specific technical test cases across authentication, session management, input handling, and business logic	Primary technical reference used in this study [2]
NIST SP 800-115	General information-security testing lifecycle: planning, discovery, attack, and reporting	Engagement structure and reporting discipline [13]
PTES	Seven-phase, engagement-oriented methodology from pre-engagement through reporting	Scoping and threat-modeling emphasis [20]
MITRE ATT&CK	Catalog of adversary tactics and techniques observed after initial access	Interpreting post-exploitation impact and attacker follow-on behavior [14]
ISO/IEC 27001	Requirements for an information security management system (ISMS)	Embedding testing within organizational governance and audit cycles [16]

Source: Authors' synthesis of publicly available framework documentation [2], [13], [14], [16], [20].

None of them rival the OWASP WSTG so much as they approach the same challenge from a different angle. While WSTG continues to be the most exhaustive regarding technical test guidelines specific to web applications; NIST SP 800-115 and PTES provide more comprehensive guidance on the process and reporting aspects; ISO/IEC 27001 represents the standard of how testing fits in the scope of security management responsibilities; and MITRE ATT&CK is the best framework for defining what occurs after the vulnerability exploitation. This paper employs the WSTG as its key technical framework, in keeping with the source manuscript, but presents the overall framework explicitly.

3. Methodology

The research employs a combined literature and practical approach. The literature aspect sets out the conceptual foundation for web application security, penetration testing, typical vulnerability categories, and risk mitigation measures. The practical aspect, as described in the source paper, involves the use of penetration testing methods on a chosen web application through manual testing and security tools. Analysis is carried out in relation to the theoretical framework.

The methodology is purposely designed in such a way that a security assessment can be conceptually replicated without being dependent on a particular piece of software. The tools enable evidence collection, but the research methodology is driven by the testing goals, validation procedures, and risk interpretation. The process is confined to a permitted test on the chosen application; the paper does not encourage testing of unauthorized systems.

3.1. Data collection and analysis

The source manuscript states the use of Burp Suite and Nmap, accompanied by manual testing. The evidence gathered is analyzed through five phases: (1) identification of potential exposed functionality and entry points; (2) identification of potential vulnerabilities; (3) verification of the results; (4) estimation of the security and operational consequences of vulnerabilities; and (5) development of mitigation solutions. The results are compared against the classification of the vulnerabilities that was outlined in the literature review.

The unit of analysis is the vulnerability class and its connection to the application control. For instance, SQL injection is analyzed as connected with input validation and parameters used for database queries; XSS as connected with output encoding and input validation; CSRF as connected with requests verification mechanism; and component vulnerabilities as connected with the dependency and update management of components.

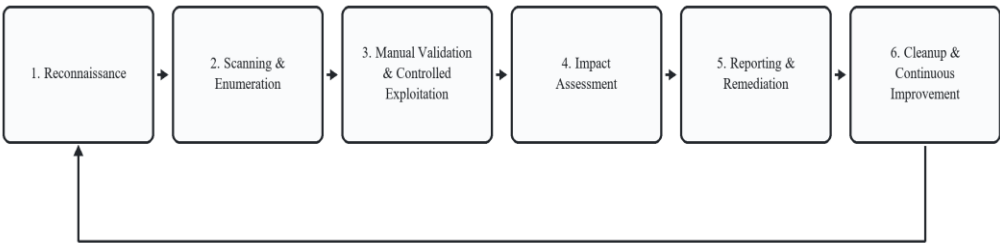


Figure 1. The six-stage penetration-testing lifecycle used in this study
Source: Authors’ own illustration based on the six-stage penetration-testing procedure described in Section 3.2.

3.2. Penetration-testing procedure

The actual process includes the following steps:

- 1) Reconnaissance. The information about the target application and its surface is collected by using relevant discovery methods and tools such as Nmap and Burp Suite.
- 2) Scanning and enumeration. The automated and manual approaches are used to discover the vulnerabilities, the exposed services, the entry points for the application, and the suspicious behavior.
- 3) Validation and controlled exploitation. The found candidates are verified in order to eliminate the false positives and, if it is allowed, demonstrate the possible impact of those vulnerabilities.

To distinguish a scanner alert from a confirmed vulnerability, the tester should manually replay the authorized request, inspect how the input is represented in the response, and compare the raw response with the browser-rendered result. The sequence in figure 2 illustrates a safe triage path for a suspected reflected cross-site scripting (XSS) finding. A safely encoded marker supports classifying that tested candidate as a false positive; an unencoded marker in an executable context indicates a potential finding and requires controlled confirmation within the approved scope.

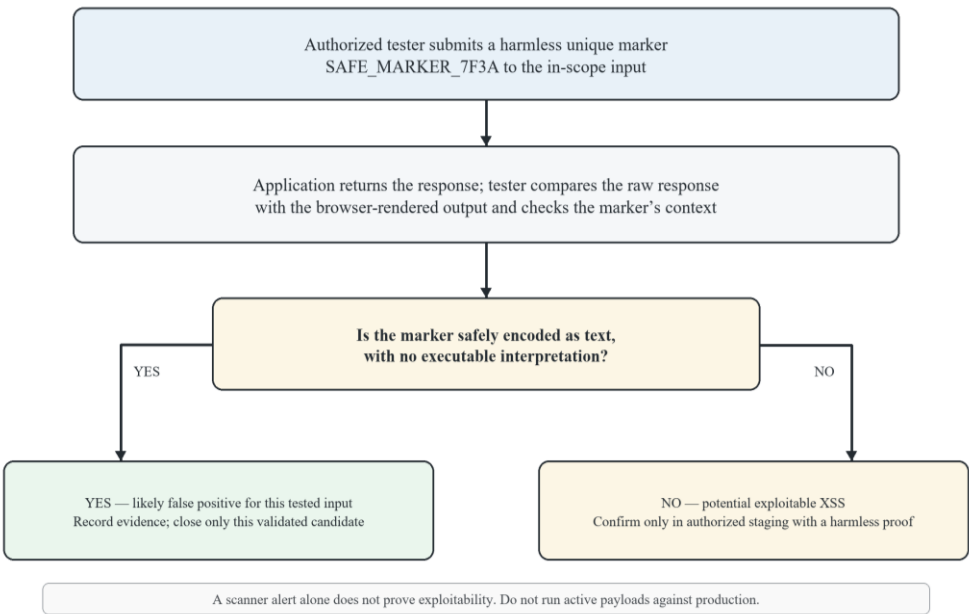


Figure 2. Manual validation of a suspected reflected XSS finding
Source: Authors' illustration based on the validation procedure in Section 3.2 and OWASP Web Security Testing Guide [2].

- 4) Impact assessment. The successful and credible attacks are examined for the confidentiality, integrity, availability, impact on users and organizations.
- 5) Reporting and remediation. The vulnerabilities are documented along with their causes, impacts and mitigation recommendations.
- 6) Cleanup and process improvement. The test artifacts are cleaned up if needed, and the findings are used in the future test cycle.

3.3. Methodological limitations

Notably, there are certain pieces of information which cannot be found in the given case study in order to make this empirical report fully reproducible, such as architectural data on the target application, the period of time spent on testing, the versions of the tools used, the amount of the endpoints tested, the full request and response data, and the number of the confirmed findings. As a result, this revised case study will not provide any quantitative findings or CVSS scores and technical data related to the target application without proper evidential support in the source material.

Such limitation does not reduce the value of the conceptual and qualitative analysis provided; however, it limits the extent to which the conclusions can be drawn regarding the security level of the tested application.

4. Findings

Practical assessment provided by the source manuscript suggests the set of vulnerability classes that can have material impact on web application security. Findings presented in the source manuscript support the paper's key argument about significant consequences of seemingly minor application weaknesses that may provide attacker with access to sensitive data, authenticated functionality or control over the application itself.

Firstly, SQL injections and XSS deserve extra attention since they result from improper processing of attacker-supplied input and may influence data confidentiality, integrity, and application user sessions. For their detection, beyond running a generic vulnerability scan, specific input points have to be identified and tested, as well as their behavior analyzed within application context according to OWASP's recommendations for avoiding injections [2].

Secondly, CSRF illustrates the importance of request-forgery protection, since the manuscript provides the use of CSRF tokens as a mitigation for such attack, which is consistent with OWASP's prevention guidance for this class of threats [8]. In broader perspective, request-forgery protection should be analyzed in relation to the use of application authentication, session management, cookies and general workflow implementation.

Thirdly, plugin and component vulnerabilities demonstrate that application security partially depends on software supply chain and maintenance processes. As such, timely updates, dependency list, vulnerability monitoring and removing unused components become part of security controls framework rather than merely operational practices.

Finally, zero-day and open source vulnerabilities emphasize the importance of defense-in-depth strategy in case of absent patch. When a vulnerability cannot be fixed immediately, organization should rely on various controls such as segmentation, monitoring, restrictive configurations, compensatory controls and readiness to react to the incident.

4.1. Vulnerability-to-control mapping

Table 2 summarizes the relationship between the principal vulnerability classes discussed in the manuscript, the testing focus, and the corresponding mitigation direction.

Table 2. Vulnerability-to-control mapping

Vulnerability class	Testing focus	Primary control	Potential impact
SQL injection	Input points; query behavior; manual validation	Parameterized queries; input validation; least privilege	Unauthorized data access or modification
Cross-site scripting (XSS)	Reflected/stored input handling; output behavior	Context-aware output encoding; input handling; CSP where appropriate	Session or data compromise; malicious content execution
CSRF	State-changing requests; authentication context	Anti-CSRF tokens; appropriate cookie controls; origin checks where applicable	Unauthorized actions performed in a user's context
Vulnerable plugins/components	Component inventory; version and known-risk review	Patch/update management; dependency monitoring; removal of unnecessary components	Known exploit paths; service compromise
Zero-day exposure	Configuration and attack-surface review	Defense in depth; monitoring; compensating controls; rapid response	Potential compromise before patch availability
Open-source component risk	Dependency and configuration assessment	SBOM/dependency governance; updates; secure configuration	Inherited vulnerabilities and supply-chain exposure

Source: Authors' synthesis based on the vulnerability classes and mitigation directions reported in the study and OWASP testing guidance [1], [2].

4.2. The vulnerability landscape in context

However, Positive Technologies' analysis of production web applications revealed cross-site scripting in a majority of tested applications and pointed out that the vast majority of identified issues stem from the application source code rather than infrastructure misconfiguration, which is in line with the results described in the source manuscript [12]. Injection vulnerabilities, cross-site scripting, and broken access control have been included in the CWE Top 25 Most Dangerous Software Weaknesses published by MITRE in several of the recent years [15]. These external statistics, independently audited, confirm the vulnerability types analyzed in the paper without replacing the application-specific statistics to be provided as part of a potential future empirical extension.

4.3. An illustrative case: Yahoo data breaches

The key idea of the paper, that even small weaknesses can combine into something serious, can be best exemplified by the case of Yahoo data breaches revealed in 2016–2017. At first, Yahoo reported about a 2014 breach caused by a state-sponsored attacker, according to which 500 million Yahoo accounts were affected; then it turned out that there had been a 2013 breach affecting 1 billion accounts, and eventually in October 2017 the company announced that both breaches impacted all three billion existing Yahoo accounts [11]. The initial breach vector was a spear-phishing attack that allowed an intruder to enter Yahoo's system where he/she proceeded to use credentials and session cookie forgery to expand his/her privileges. Apart from the damage to reputation, Yahoo's finances took tangible hits: Verizon cut its purchase price for Yahoo business by \$350 million during the acquisition, and Yahoo settled for a \$117.5 million class-action lawsuit and also had to pay a \$35 million SEC fine for under-reporting of the breaches [11]. Table 3 shows the numbers involved.

Table 3. Yahoo data breach — scale and cost summary

Item	Figure
Accounts ultimately confirmed affected	≈ 3 billion (all Yahoo accounts, per the October 2017 revised disclosure)
Initial access vector	Spear-phishing email against a Yahoo employee
Purchase-price reduction (Verizon acquisition, 2017)	\$350 million
Class-action settlement (2019)	\$117.5 million
SEC fine for delayed disclosure (2018)	\$35 million

Source: Authors' synthesis based on publicly reported breach and litigation figures [11].

This case demonstrates the position taken by this paper in two ways. One, the original vulnerability exploited was not the result of using a complex zero-day; rather, the vulnerability involved a social engineering vector that targets employees, exactly the kind of weakness that a properly scoped and recon-based penetration test would expose. Second, the fact that there were several revisions of the number of compromised accounts demonstrates the importance of reporting and continual improvement, as discussed in Section 3.2. An organization that does not know the scope of the problem is going to have difficulty solving it.

5. Discussion

The results confirm the research question in that penetration testing becomes most effective when it is regarded as a structured validation exercise rather than simply as a tool-enabled bug hunting exercise. Automated tools may help to discover issues, but their analysis by humans is needed to conclude if an issue is exploitable, if it is a false positive or if it produces a business/operational impact.

Additionally, the paper provides an explanation about the relation between OWASP Top 10 and penetration testing. It is valuable for communicating risks and prioritizing them but it is not a full test methodology. It is only the WSTG that provides a wide range of tests

and focuses on information gathering, authentication, authorization, input validation, business logic, cryptography, client-side security and reporting [2]. This helps to strengthen scientific position of the paper and to avoid overstating results because of using one risk taxonomy.

Another conclusion relates to organizational practices. A finding might have little use if it cannot result in a remediation decision. Hence, the revised framework connects the finding with control, impact and continuous improvement. It is in line with latest SCRD papers that regard cybersecurity as a problem of technical protection, risk management and organization [3], [4].

5.1. Contribution and practical implications

The first major contribution is a succinct model which brings together four separate concepts: vulnerability categories, testing operations, security measures, and organizational implications. For developers, this model helps secure coding and component management. For security professionals, it provides a standardized foundation for planning and reporting on the assessment. For management, it allows the technical problems to be translated into operational risks and mitigation priorities.

The model also facilitates ongoing testing. Applications evolve as a result of new functionality, plugins, libraries, configuration changes, and upgrades. Thus, a one-off vulnerability assessment is quickly rendered obsolete. Repeated penetration testing, coupled with automated scanning and secure development practices, is much more in line with the dynamic nature of an application attack surface.

Lastly, the alignment with established external frameworks adds the compliance perspective to this model. An organization with the compliance obligations related to ISO/IEC 27001 or similar regulation can implement the testing to remediation chain outlined above without treating it as an independent process: the reconnaissance through remediation stages in section 3.2 correspond to the planning, discovery, attack and reporting activities in NIST SP 800-115 [13]; and the reporting phase can be structured to fulfill the technical and executive reporting requirements set out in PTES [20].

Table 4. Urban IT administrator security checklist

Checklist item	Practical action	Evidence to retain
Define scope and ownership	Identify public-facing applications, APIs, IoT dashboards, and services in scope. Assign an owner for each service and assessment.	Approved scope, service inventory, and named owners
Protect accounts and access	Review authentication, administrator privileges, session handling, and access rights. Remove unnecessary accounts and permissions.	Access review and documented changes
Validate inputs and requests	Test input handling and access controls for injection, cross-site scripting (XSS), and unauthorized actions. Verify CSRF protections where applicable.	Test cases, sanitized evidence, and remediation status

Checklist item	Practical action	Evidence to retain
Maintain components	Inventory frameworks, plugins, libraries, and IoT-facing components. Check for known vulnerabilities and apply updates or compensating controls.	Dependency inventory and update records
Protect sensitive information	Review encryption in transit and at rest where applicable, key handling, and exposure of personal or operational data.	Configuration review and corrective actions
Prepare for service disruption	Confirm that backups, recovery procedures, monitoring, and incident contacts support restoration of essential public services.	Backup and recovery records; incident-response contacts
Verify findings safely	Conduct authorized testing, manually validate scanner findings, assess operational impact, and avoid disrupting live public services.	Test authorization, validation notes, and impact assessment
Track remediation and retest	Assign each finding an owner and priority. Record corrective action and retest after remediation or significant application changes.	Findings register, closure evidence, and retest results

Source: Authors' checklist, synthesized from the testing and mitigation approach described in this study and OWASP guidance [1], [2], [8].

5.2. Validity and future research

The primary validity constraint is the lack of empirical details kept in the source document. The future research would need to address this gap by providing the application architecture, scope of the assessment, testing period, number and types of endpoints tested, tools used and their version numbers, validated vulnerabilities, severity levels, false positives management and remediation verification. Controlled before-and-after assessment would enable the demonstration of effectiveness of particular mitigations.

Another way forward is comparison of purely manual, purely automated, and combined testing methods using the same application and the pre-defined set of test cases. Such methodology would allow measuring the differences in coverage, false positive rates, detection times and effectiveness of remediation. This would significantly improve the empirical part of the paper without altering the core research problem.

6. Conclusion and recommendations

Web application security penetration testing is a crucial tool because it is an efficient method of assessing whether controls work or not and identifying the possible outcomes of vulnerabilities. The modified version of the analysis allows concluding that SQL injection, XSS, CSRF, vulnerable components, zero-day risk, and risks associated with open-source software need to be addressed in various but connected ways.

The study suggests: (1) carrying out regular and scoped penetration testing; (2) using a combination of manual validation and automation tools; (3) adhering to secure coding practices; (4) using timely updating and dependency management controls; (5) securing sensitive data with encryption; (6) request verification with CSRF tokens where it is applicable; (7) enhancing logging, monitoring, and incident response capabilities; (8)

providing constant cybersecurity education; and (9) having an updated vulnerability management checklist.

The main finding is that the security of web applications should be seen as an ongoing and evidence-based process. Penetration testing will be most beneficial if it is reproducible, technically valid, tied to impact, and turned into a remediation action. The modified framework allows achieving the last part while being explicit in its empirical details.

References

- [1] OWASP Foundation, “SQL Injection Prevention Cheat Sheet,” OWASP Cheat Sheet Series, 2026. Available: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- [2] OWASP Foundation, “OWASP Web Security Testing Guide,” stable/latest project documentation, 2026. Available: <https://owasp.org/www-project-web-security-testing-guide/>
- [3] C. A. GABRIAN, “Impact Zones: How cybercrime disrupts and shapes the landscape of data security,” International Conference on Machine Intelligence & Security for Smart Cities (TRUST) Proceedings, vol. 1, pp. 59–68, 2024. Available: <https://scrd.eu/index.php/trust/article/view/547>
- [4] G.-L. POPA, “Risk management, protection, and security of personal data in Romania,” International Conference on Machine Intelligence & Security for Smart Cities (TRUST) Proceedings, vol. 1, pp. 69–78, 2024. Available: <https://scrd.eu/index.php/trust/article/view/548>
- [5] E. LEKA, L. LAMANI, A. ALITI, and E. HOXHA, “Web Application Firewall for Detecting and Mitigation of Based DDoS Attacks Using Machine Learning and Blockchain,” TEM Journal, vol. 13, no. 4, pp. 2802–2811, 2024. Available: <https://doi.org/10.18421/TEM134-17>
- [6] E. Leka, E. Hoxha, and G. Rexha, “Security and Privacy Concerns Associated with the Internet of Things (IoT) and the Role of Adapting Blockchain and Machine Learning – A Systematic Literature Review,” in *Proc. 2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, pp. 1690–1696, 2023, doi:10.23919/MIPRO57284.2023.10159869
- [7] D. HYKA, E. VUKA, and M. S. MILOŠEVIĆ, “The future of web security: Advanced testing and cryptographic defense,” WSEAS Trans. Inf. Sci. Appl., vol. 23, pp. 253–261, 2026. Available: <https://doi.org/10.37394/23209.2026.23.19>
- [8] OWASP Foundation, “Cross-Site Request Forgery Prevention Cheat Sheet,” OWASP Cheat Sheet Series, 2026. Available: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
- [9] D. HYKA and A. BAXHAKU, “Some Cryptographic Weakness and the Ways to Avoid Them in ICT Services of Developing Countries,” International Journal of Information Security Science, vol. 3, no. 2, pp. 173–182, 2014.
- [10] D. HYKA, A. HYRA, F. BASHOLLI, B. M. and A. BASHOLLI, “Data Security in Public and Private Administration: Challenges, Trends, and Effective Protection in the Era of Digitalization,” Advanced Engineering Days, vol. 7, pp. 125–127, 2023.
- [11] J. JOHNSON, “Yahoo data breach 2017: The largest publicly disclosed data breach in history,” Statista, 2021.
- [12] Positive Technologies, “Web Application Vulnerabilities and Threats: Statistics for 2019,” 2020. Available: <https://global.ptsecurity.com/en/research/analytics/web-application-vulnerabilities-statistics-2019/>
- [13] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh, “Technical Guide to Information Security Testing and Assessment,” NIST Special Publication 800-115, National Institute of Standards and Technology, 2008. Available: <https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-115.pdf>
- [14] MITRE Corporation, “MITRE ATT&CK,” 2026. Available: <https://attack.mitre.org/>
- [15] MITRE Corporation, “2024 CWE Top 25 Most Dangerous Software Weaknesses,” 2024. Available: https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html
- [16] International Organization for Standardization, “ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection — Information security management systems — Requirements,” 2022. Available: <https://www.iso.org/standard/27001.html>

- [17] Verizon, “2025 Data Breach Investigations Report,” 2025. Available: <https://www.verizon.com/business/resources/reports/2025-dbir-data-breach-investigations-report.pdf>
- [18] Y. Chen, F. M. Zahedi, A. Abbasi, and D. G. Dobolyi, “Trust calibration of automated security IT artifacts: A multi-domain study of phishing-website detection tools,” *Information & Management*, vol. 58, no. 1, Art. no. 103394, 2021, doi: [10.1016/j.im.2020.103394](https://doi.org/10.1016/j.im.2020.103394)
- [19] M. Naiseh, D. Al-Thani, N. Jiang, and R. Ali, “Explainable recommendation: When design meets trust calibration,” *World Wide Web*, vol. 24, pp. 1857–1884, 2021, doi: [10.1007/s11280-021-00916-0](https://doi.org/10.1007/s11280-021-00916-0)
- [20] Penetration Testing Execution Standard (PTES), “The Penetration Testing Execution Standard,” [Online]. Available: <http://pentest-standard.org/>. [Accessed: Oct. 10, 2026].