BOHR International Journal of Smart Computing and Information Technology
2026, Vol. 7, No. 1, pp. 29–40
DOI: 10.54646/bijscit.2026.53
www.bohrpub.com

RESEARCH

Web application security using top 10 OWASP

Sabitha Banu[1*], H. Shanmatha[1], Mehdi Gheisari[2,3,4,5], Zhou Pingmei[4], Hossein Akhtari[6], Sajad Dashti[7] and Seyed Kazem Gheblezadeh Meimouneh[8]

[1]Department of Computer Science with Cybersecurity, PSGR Krishnammal College for Women, Coimbatore, India
[2]Institute of Artificial Intelligence, Shaoxing University, Zhejiang, China
[3]Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Science, Chennai, India
[4]Department of R&D, Shenzhen BKD Co LTD, Shenzhen, China
[5]Department of Computer Engineering, Shi.C., Islamic Azad University, Shiraz, Iran
[6]Department of Electrical Engineering, Kazeroon Branch, Islamic Azad University, Fars, Iran
[7]Islamic Azad University, Bandargaz Branch, Bandar Abbas, Iran
[8]Department of Computer Science, Meybod, Islamic Azad University, Meybod, Iran

*Correspondence:
Sabitha Banu,
sabithabanu@psgrkcw.ac.in

Received: 04 May 2026; **Accepted:** 28 June 2026; **Published:** 31 July 2026

Web applications have become an integral part of everyday life, enabling services such as online banking, e-commerce, education, and communication. As their adoption continues to increase, so does the risk of cyberattacks targeting security weaknesses within these applications. Many of these vulnerabilities arise from insecure coding practices, improper configurations, and inadequate security controls. To address these challenges, the Open Web Application Security Project (OWASP) Top 10 serves as a widely accepted framework for identifying and mitigating common web application security risks. This study investigates web application vulnerabilities based on the OWASP Top 10 framework through a practical security assessment approach. Various security tools, including Burp Suite, OWASP ZAP, Threat Dragon, Hydra, Trivy, and Splunk, were utilized to perform threat modeling, vulnerability assessment, authentication testing, dependency analysis, and security monitoring. Testing was conducted in a controlled environment to evaluate the effectiveness of these tools in identifying security weaknesses. The assessment revealed several significant vulnerabilities, including Broken Access Control (IDOR), Cryptographic Failures, HTML Injection, Insecure Design, Identification and Authentication Failures, and Security Logging and Monitoring Failures. The findings demonstrate how these weaknesses can compromise application security and expose systems to potential attacks. Appropriate mitigation measures were also identified to reduce associated risks. The study concludes that web application security requires continuous assessment and proactive security practices throughout the software development lifecycle. Adopting OWASP guidelines and implementing effective security controls can significantly enhance the protection and resilience of modern web applications.

Keywords: web application security, OWASP top 10, vulnerability assessment, penetration testing, burp suite, OWASP ZAP, threat modeling, security monitoring

Introduction

Web applications are widely used in everyday activities such as online banking, e-commerce, education, and social networking. As these applications handle large amounts of sensitive information, ensuring their security has become increasingly important. However, vulnerabilities caused by insecure coding practices, weak authentication mechanisms, improper input validation, and misconfigurations can expose applications to cyberattacks.

The Open Web Application Security Project (OWASP) Top 10 provides a globally recognized framework that

identifies the most critical security risks affecting web applications. It serves as a valuable guideline for developers and security professionals to understand, detect, and mitigate common vulnerabilities.

This study focuses on evaluating web application security using the OWASP Top 10 framework. Practical testing was conducted in a vulnerable web application environment to identify and analyze common security issues, including Broken Access Control, Cryptographic Failures, HTML Injection, Insecure Design, Authentication Failures, and Security Logging and Monitoring Failures. The findings highlight the importance of secure development practices and effective security testing in protecting modern web applications.

Literature survey

Widyawati et al. (1) presented a study titled "Web Security Vulnerability Analysis and Mitigation Based on OWASP Top 10," published in the Journal of Artificial Intelligence and Engineering Applications (JAIEA). This work provides a detailed analysis of common web application vulnerabilities based on the OWASP Top 10 framework and discusses effective mitigation strategies with a focus on modern web applications.

Li and Li (2) proposed a paper entitled "Evolution of Application Security based on OWASP Top 10 and CWE/SANS Top 25 with Predictions for the 2025 OWASP Top 10," presented at the ICICT 2025 Conference. The study analyzes the evolution of application security threats over time and predicts future OWASP risks by comparing OWASP Top 10 vulnerabilities with CWE/SANS Top 25 attack patterns.

Patil et al. (3) published a review paper titled "A Review of the OWASP Top 10 Web Application Security Risks and Best Practices" in the ICCUBEA 2023 Conference. This work offers a comprehensive review of OWASP Top 10 security risks and highlights best practice mitigation techniques that can be adopted to ensure secure web application development.

Rohmaniah et al. (4) presented the study "Enhancing Website Security Using VAPT Based on OWASP Top Ten," published in the Journal of Applied Informatics and Computing. This research demonstrates how vulnerability assessment and penetration testing (VAPT) techniques based on OWASP Top 10 can improve the resilience and security posture of modern web applications.

Qadir et al. (5) proposed "Comparative Evaluation of Approaches & Tools for Security Testing of Web Applications," published in PeerJ Computer Science. The paper compares different security testing tools and methodologies aligned with OWASP, highlighting their effectiveness in detecting vulnerabilities across diverse web applications.

Fredj et al. (6) conducted a study titled "An OWASP Top Ten Driven Survey on Web Application Protection Methods," available on TechRxiv. This work surveys various protection mechanisms adopted to secure web applications from OWASP Top 10 vulnerabilities, providing an overview of preventive strategies and best practices.

Kumar and Rani (7) published "Implementation and Analysis of Web Application Security Measures using OWASP Guidelines" in the ICMACC 2022 Conference. The study focuses on the practical implementation of OWASP security measures and analyzes their performance, offering insights into effective mitigation strategies for real-world web applications.

Nawrocki and Kołodziej (8) conducted a study titled "Vulnerabilities of Web Applications: Good Practices and New Trends," published in Applied Cybersecurity & Internet Governance. This work discusses modern web vulnerabilities and emerging cybersecurity trends with OWASP alignment.

Nedeljković et al. (9) conducted a study titled "Use of OWASP Top 10 in Web Application Security," published in ITEMA 2020. This work explains the importance of OWASP guidelines in strengthening web application security.

Lala et al. (10) conducted a study titled "Secure Web Development using OWASP Guidelines," published in ICICCS 2021. This work focuses on secure coding techniques and development practices using OWASP principles.

Methodology

The experiments and security assessments were conducted from November 2025 to March 2026 in a controlled laboratory environment (11–27).

Test environment

The security assessment was conducted in a controlled laboratory environment using Kali Linux 2025.4 as the primary operating system. The tools used during the assessment included Burp Suite Community Edition, OWASP ZAP 2.16, Hydra 9.5, OWASP Threat Dragon 2.x, Splunk Enterprise 9.x, and Trivy 0.61. The target applications selected for testing were Damn Vulnerable Web Application (DVWA), Buggy Web Application (bWAPP), and OWASP Mutillidae. These applications were chosen because they intentionally contain vulnerabilities representing different categories of the OWASP Top 10, enabling comprehensive security testing and analysis.

The methodology adopted in this research follows a systematic and tool-driven approach to assess the security of web applications in alignment with the OWASP Top 10 vulnerabilities. Three intentionally vulnerable web applications, namely DVWA, bWAPP, and Mutillidae, were selected as the target environments because each application

demonstrates different categories of OWASP Top 10 vulnerabilities. Using multiple applications enabled broader coverage of attack scenarios than relying on a single testing platform. DVWA was primarily used for authentication and access control testing, bWAPP for business logic and parameter tampering assessments, and Mutillidae for evaluating injection-related vulnerabilities. Initially, threat modeling was performed using OWASP Threat Dragon to identify potential attack surfaces, trust boundaries, and high-risk components within the application architecture. Based on the identified threats, dynamic application security testing was carried out using Burp Suite and OWASP ZAP to detect vulnerabilities such as injection flaws, cross-site scripting, insecure authentication mechanisms, and security misconfigurations. To strengthen the assessment, Trivy was used to scan application dependencies and configurations for known vulnerabilities, providing insight into risks arising from outdated or insecure components. Authentication security was further evaluated using Hydra through controlled brute-force testing to analyze the resilience of login mechanisms. In addition to vulnerability detection, Splunk was deployed to collect and analyze logs generated during testing, enabling real-time monitoring of security events and suspicious activities. The combined use of these tools allowed for comprehensive vulnerability identification, correlation of security findings, and validation of attack scenarios, thereby ensuring a practical and effective evaluation of web application security. Furthermore, the findings obtained from each tool were cross-verified to reduce false positives and ensure the accuracy of the identified vulnerabilities.

False positive validation

To improve the accuracy of the vulnerability assessment, all findings generated by security tools were manually verified before inclusion in the final results. Vulnerabilities identified by OWASP ZAP were cross-checked using Burp Suite and additional manual testing. Findings that could not be reproduced or validated during testing were classified as false positives and excluded from the final analysis. This validation process helped ensure the accuracy, reliability, and consistency of the reported security findings.

Results or finding

Broken access control ticket price (IDOR)

Refer to **Figures 1–4** for additional information.

Recommended fix

Use Authorizations in the Server-side code for all requests to check the authorization of users before using any resources. Always use the principle of least privilege and review the access control policy periodically.

Cryptographic failure

Refer to **Figures 5** and **6** for additional information.

Recommended fix

Use HTTPS with TLS 1.2 or higher for all communications, encrypt sensitive data at rest, and avoid storing passwords in plain text. Strong cryptographic algorithms should be used for data protection.

HTML injection

Refer to **Figures 7–10** for additional information.

Recommended fix

Implement input validation and output sanitization, use parameterized queries or prepared statements, and avoid dynamically constructing structured query language (SQL) queries using user-supplied data.

Insecure design

Refer to **Figures 11** and **12** for additional information.

Recommended fix

Apply secure design principles, perform threat modeling during development, and conduct regular security reviews to identify and mitigate risks before deployment.

Security misconfiguration

Refer to **Figures 13** and **14** for additional information.

Recommended fix

Implement secure configuration settings for servers, applications, databases, and network components. Remove unnecessary services, default accounts, and unused features, regularly apply security patches and updates, enforce proper access controls, and conduct periodic configuration reviews and vulnerability assessments to identify and remediate misconfigurations.

Identification and authentication failure

Refer to **Figures 15** and **16** for additional information.

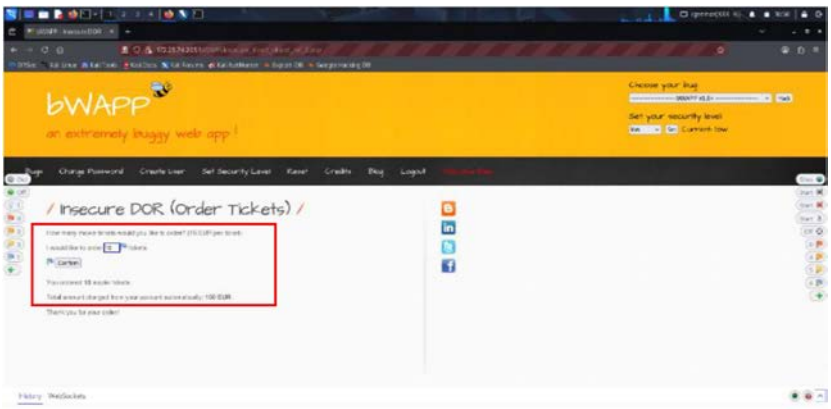

FIGURE 1 | Ordered 10 Tickets in bWAPP website at 150 EUR.

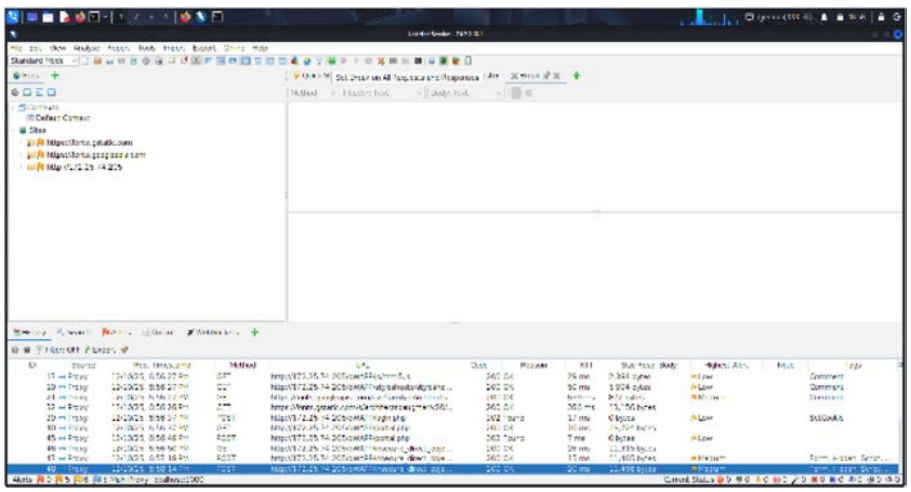

FIGURE 2 | Captured the request using OWASP ZAP.

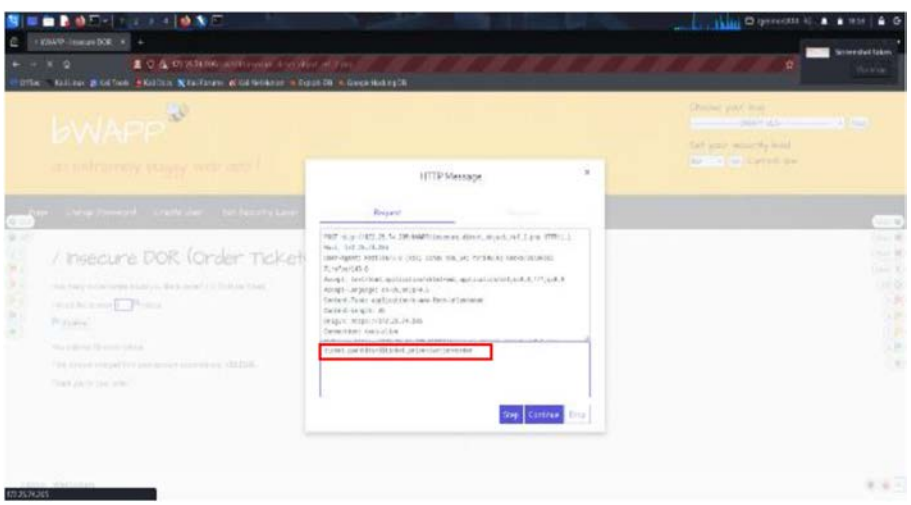

FIGURE 3 | Modified the captured request of ticket price from 15 EUR to 1 EUR.

Recommended fix

Implement strong authentication mechanisms by enforcing complex password policies, enabling multi-factor authentication (MFA), and applying account lockout controls to prevent brute-force attacks. Use secure password storage techniques such as hashing with strong algorithms, manage user sessions securely, and regularly monitor authentication logs for suspicious activities.

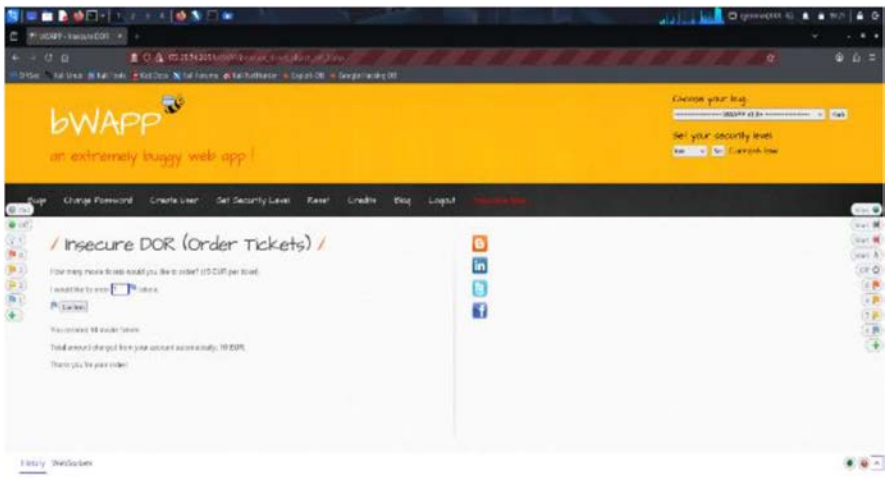

FIGURE 4 | The price of 15 tickets has been tampered from 150 EUR to 10 EUR.

FIGURE 5 | Entering username and password, capturing the request in Burp Suite.

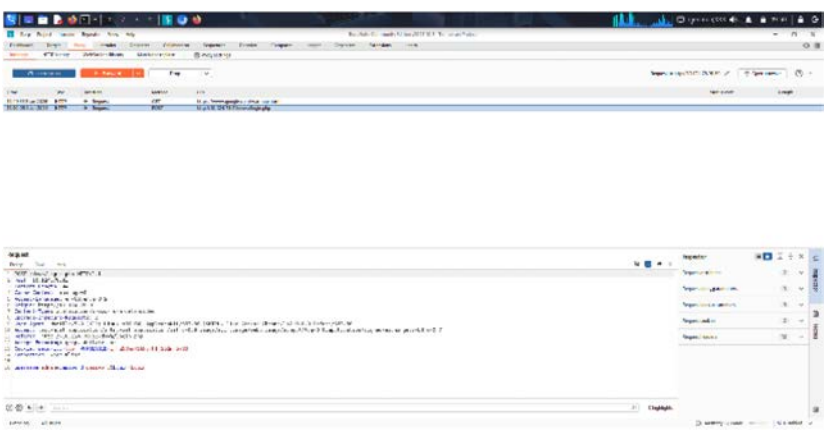

FIGURE 6 | The username and password are in plain text.

Cross-site request forgery (CSRF)

Refer to **Figures 17** and **18** for additional information.

Recommended fix

Implement anti-cross-site request forgery (CSRF) tokens for all state-changing requests and validate them on the server side. Configure cookies with the Same Site attribute (Strict or Lax) to prevent unauthorized cross-site requests. Require re-authentication or MFA for sensitive actions and validate the Origin and Referrer headers where applicable. Conduct regular security testing to ensure CSRF protections remain effective.

Security logging and monitoring failures

Refer to **Figures 19** and **20** for additional information.

FIGURE 7 | Accessing the Mutillidae website and viewing browser info.

FIGURE 8 | User agent contains browser information.

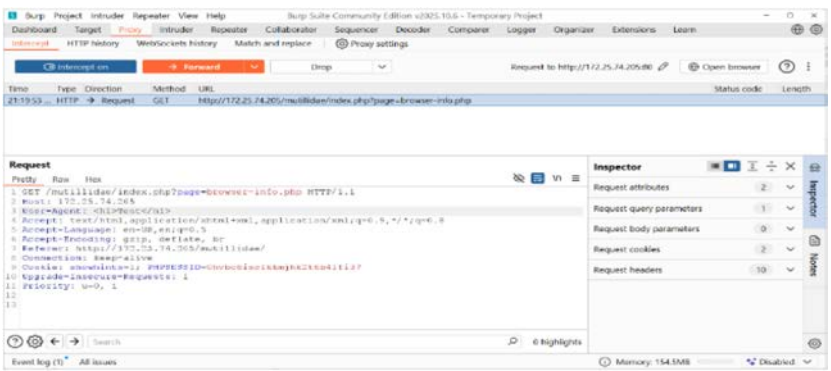

FIGURE 9 | Changing user-agent details by performing HTML injection using Burp Suite.

FIGURE 10 | The user agent has been modified as test.

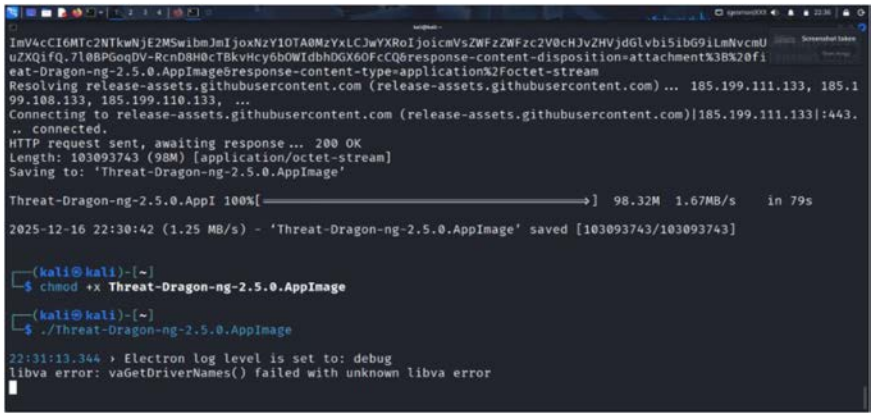

FIGURE 11 | Creating a threat model in OWASP Threat Dragon.

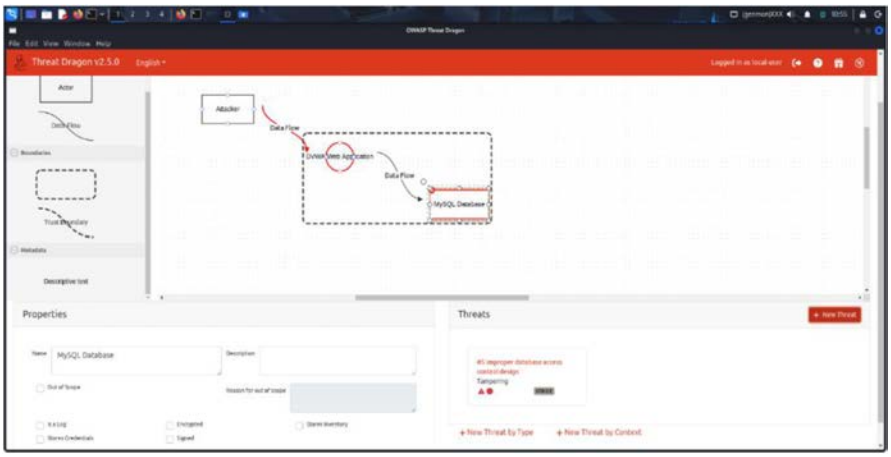

FIGURE 12 | Identifying and analyzing security threats.

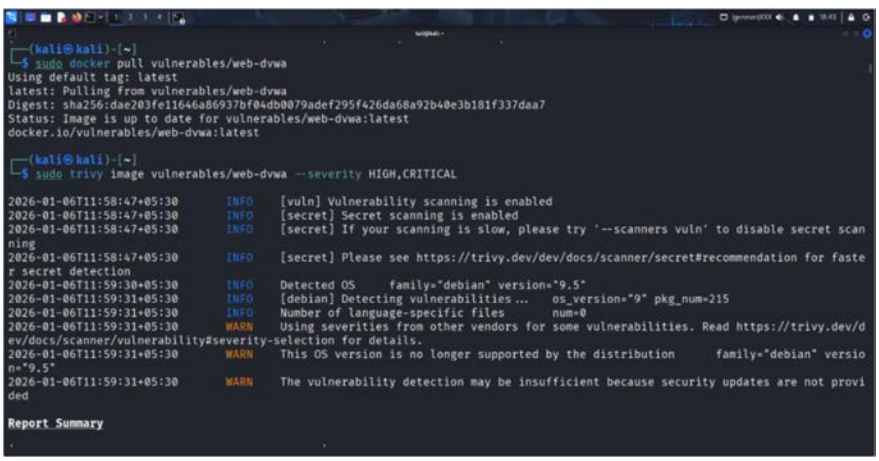

FIGURE 13 | Creating a threat model in OWASP Threat Dragon.

Query used

```
index=apache_logs
  | stats count by clientip
  | where count > 10
```

Observation

The query identified IP addresses generating a high number of requests within a short period. Such activity may indicate brute-force attacks or abnormal user behavior. By monitoring these events in real time, Splunk helps improve

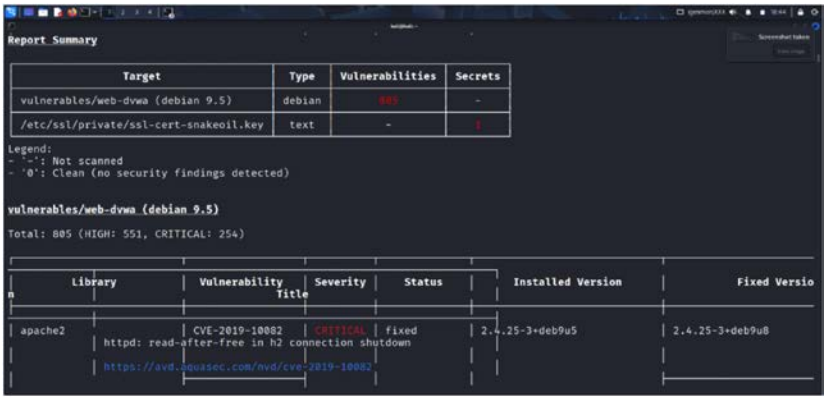

FIGURE 14 | Trivy vulnerability report summary.

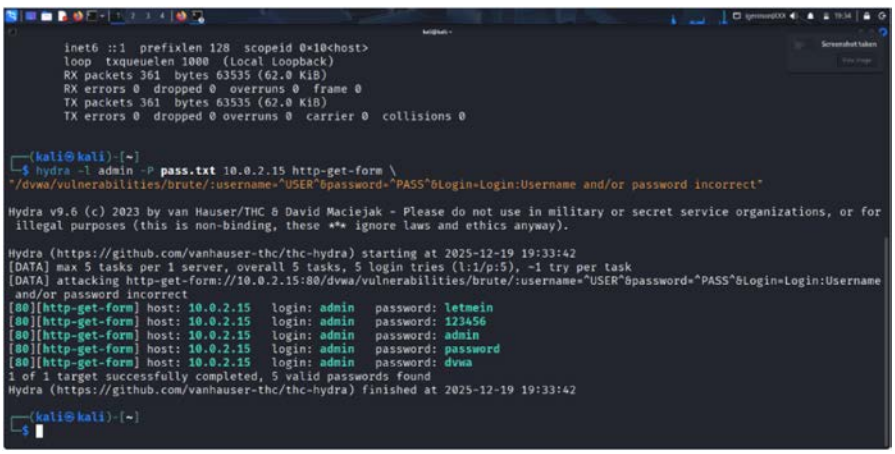

FIGURE 15 | Performing a brute force attack using Hydra.

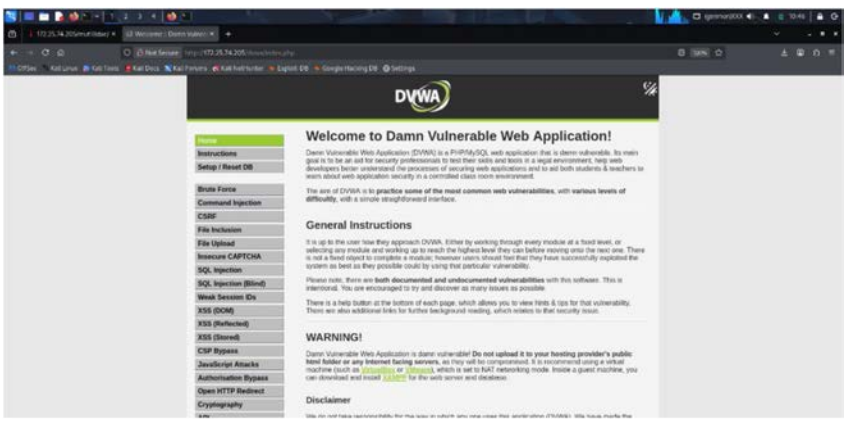

FIGURE 16 | Successful login using cracked credentials.

visibility into potential security threats and supports faster incident response.

Recommended fix

Organizations should implement centralized log management and configure alerts for unusual activities such as repeated login attempts and excessive requests from a single IP address. Regular log analysis can help detect threats early and strengthen overall security monitoring.

Using components with known vulnerabilities

Refer to **Figure 21** for additional information.

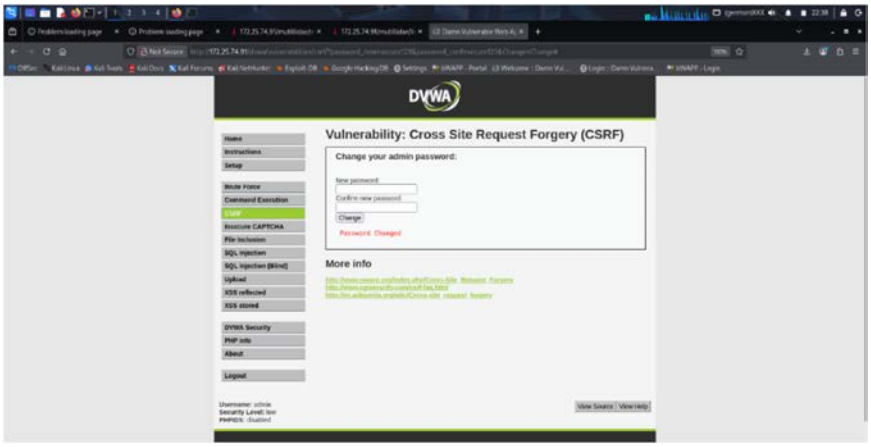

FIGURE 17 | Accessing a cross-site request forgery (CSRF)-vulnerable password change page.

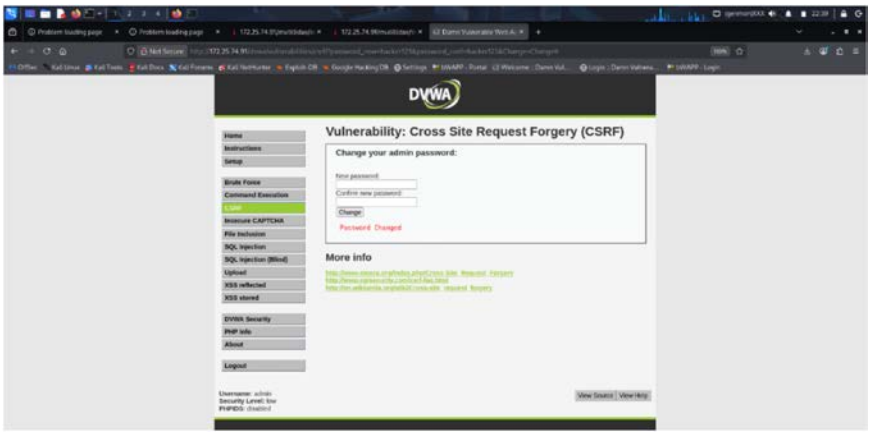

FIGURE 18 | Executing CSRF attack and changing password.

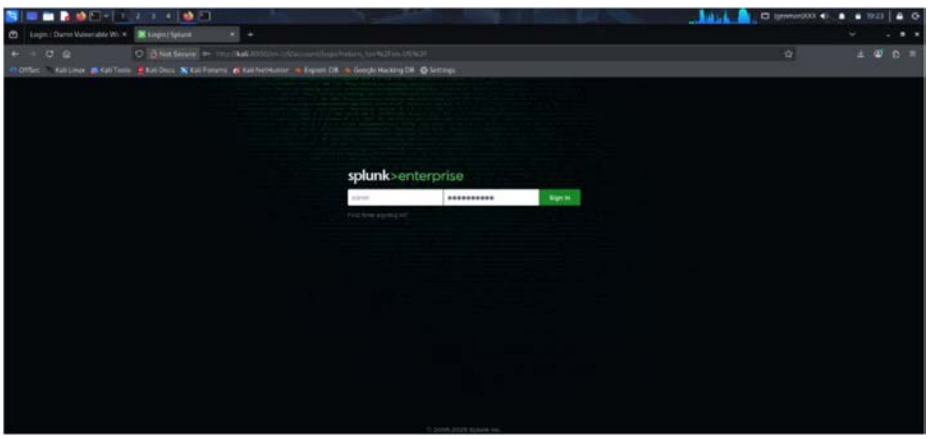

FIGURE 19 | Log collection and forwarding to Splunk.

Recommended fix

Maintain an up-to-date inventory of all software components, libraries, frameworks, and dependencies used in the application. Regularly monitor for security updates and patches, remove unsupported or unnecessary components, and promptly update vulnerable dependencies. Use automated vulnerability scanning tools to identify and remediate known security flaws before deployment.

Unvalidated redirects and forwards

Refer to **Figures 22** and **23** for additional information.

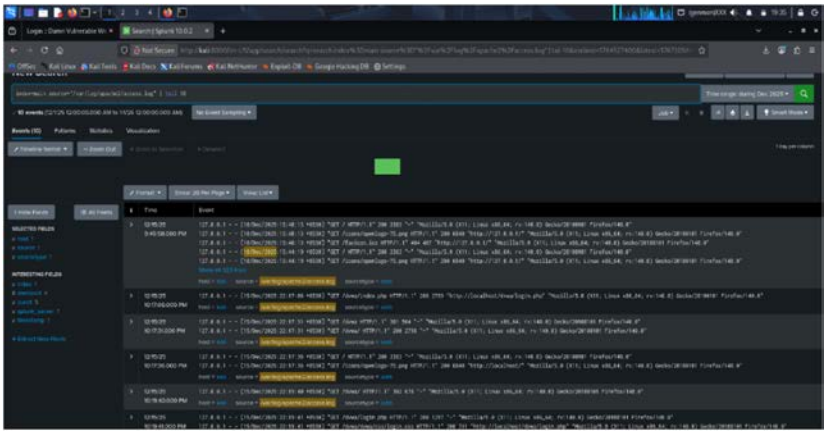

FIGURE 20 | Log analysis and monitoring in the Splunk dashboard.

FIGURE 21 | Exposure of outdated hypertext preprocessor (PHP) version (5.3.2).

FIGURE 22 | Unvalidated redirect demonstration.

Recommended fix

Avoid using user-supplied input directly in redirect or forward destinations. Implement a whitelist of approved uniform resource locators (URLs) and validate all redirect targets before processing. Use indirect references or server-side mappings for redirects and display a warning page when redirecting users to external websites.

Discussion

The security testing performed on DVWA, bWAPP, and Mutillidae showed that web applications can contain several security weaknesses if proper security measures are not followed. The vulnerabilities identified during the assessment matched many categories of the OWASP Top

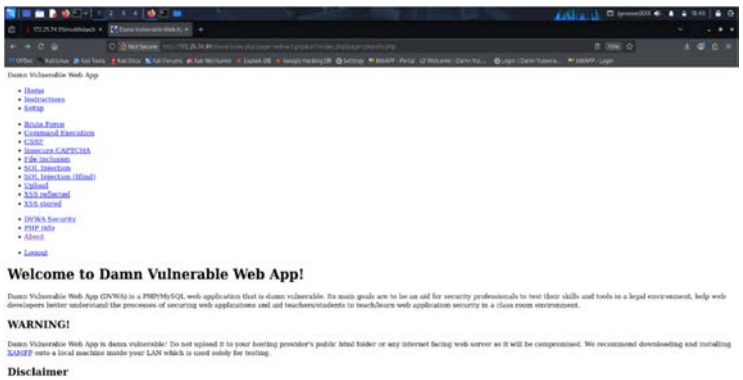

FIGURE 23 | Successful URL manipulation demonstrating A10 – unvalidated redirect vulnerability.

10, proving that these risks are still common in web applications.

The results showed that vulnerabilities such as Broken Access Control, SQL Injection, Security Misconfiguration, CSRF, and Authentication Failures can seriously affect the confidentiality, integrity, and availability of web applications. If these vulnerabilities are exploited by attackers, sensitive information may be exposed or modified without authorization.

The study also demonstrated that many security issues can be prevented by following secure coding practices, implementing proper authentication and authorization controls, validating user inputs, and regularly updating system configurations. Security testing tools helped identify vulnerabilities efficiently and provided valuable insights into application weaknesses.

Overall, the findings emphasize the importance of regularly assessing web applications for security vulnerabilities. Using the OWASP Top 10 as a security guideline helps developers and organizations build more secure applications and reduce the risk of cyberattacks.

Results

The security assessment was conducted on three vulnerable web applications: DVWA, bWAPP, and Mutillidae. Various OWASP Top 10 vulnerabilities were tested and successfully identified using security testing techniques and tools.

The results showed that all three applications contained multiple security weaknesses, including Broken Access Control, Injection, Security Misconfiguration, Identification and Authentication Failures, Insecure Design, CSRF, and Security Logging and Monitoring Failures.

SQL Injection vulnerabilities allowed unauthorized access to database information by manipulating user input fields. Broken Access Control vulnerabilities enabled access to restricted resources without proper authorization. Authentication weaknesses demonstrated the risks associated with weak login mechanisms and poor session management.

Security misconfiguration issues were identified through default settings and improper application configurations. CSRF attacks were successfully performed, showing that unauthorized actions could be executed on behalf of authenticated users. In addition, insufficient logging and monitoring mechanisms made it difficult to detect malicious activities.

The findings indicate that web applications are highly vulnerable when security controls are not properly implemented. The OWASP Top 10 framework provided an effective approach for identifying and understanding these security risks. The results highlight the importance of secure coding practices, regular vulnerability assessments, and proper security configurations to protect web applications from cyber threats.

Conclusion

This study evaluated the security of web applications using the OWASP Top 10 framework. Security testing was conducted on DVWA, bWAPP, and Mutillidae to identify common web application vulnerabilities. The results showed the presence of several security weaknesses, including Broken Access Control, Injection, Security Misconfiguration, Identification and Authentication Failures, CSRF, and Security Logging and Monitoring Failures.

The findings demonstrate that web applications can be vulnerable to various cyberattacks if proper security controls are not implemented. The study highlights the importance of secure coding practices, regular vulnerability assessments, proper authentication mechanisms, and secure system configurations to improve application security.

One limitation of this study is that the testing was performed on intentionally vulnerable applications in a controlled laboratory environment. Therefore, the results may not fully represent the security challenges found in real-world production systems.

Future research can focus on assessing modern web applications, integrating automated security testing tools, and evaluating advanced security mechanisms for detecting and preventing emerging cyber threats. Continuous security assessment and adherence to OWASP guidelines

will help organizations build more secure and resilient web applications.

Author contributions

S.B. conceived and supervised the study. S.H. conducted the experiments, collected data, performed security assessments, and prepared the manuscript. Both authors reviewed and approved the final manuscript.

Funding

The authors declare that financial support was not received for this work and/or its publication.

Acknowledgments

The authors would like to thank PSGR Krishnammal College for Women for providing the resources and support required to complete this research work.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Syarifudin M, Widyawati L, Asroni O. Web security vulnerability analysis and mitigation based on OWASP TOP 10. *J Artif Intell Eng Appl (JAIEA)*. (2025) 4(3):1829–34.
2. Li Y, Li X. Evolution of application security based on OWASP top 10 and CWE/SANS top 25 with predictions for the 2025 OWASP top 10. *ICICT 2025 Conference*. (2025).
3. Patil S, Rao M, Misal L, Phaldesai D, Shivsharan K. A review of the OWASP top 10 web application security risks and best practices. *ICCUBEA 2023 Conference*. (2023).
4. Rohmaniah D, Ashari WM, Lukman L, Putra AD. Enhancing Website Security Using VAPT Based on OWASP Top Ten. *J Appl Inform Comput*. (2025) 9(2):404–11.
5. Qadir S, Waheed E, Khanum A, Jehan S. Comparative evaluation of approaches tools for security testing of web applications. *PeerJ Comput Sci*. (2025) 11:e2821.
6. Fredj OB, Cheikhrouhou O, Krichen M, Hamam H. An OWASP top ten driven survey on web application protection methods. *Proc. Int. Conf. Risks and Security of Internet and Systems*. Cham: Springer International Publishing (2020).
7. Kumar A, Rani S. Implementation and analysis of web application security measures using OWASP guidelines. *ICMACC 2022 Conference*. (2022).
8. Nawrocki M, Kołodziej J. Vulnerabilities of web applications: Good practices and new trends. *Appl Cybersec Int Gov*. (2024) 3(2):122–43.
9. Nedeljković N, Vugdelija N, Kojić N. Use of OWASP Top 10 in web application security. *Proc. Fourth Int. Scientific Conf. Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture*. (2020).
10. Lala SK, Kumar A. Secure web development using OWASP guidelines. *Proc. 5th Int. Conf. Intelligent Computing and Control Systems (ICICCS)*. Madurai, India: IEEE (2021).
11. Idris M, Syarif I, Winarno I. Web application security education platform based on OWASP API security project. *EMITTER Int J Eng Technol.* (2022) 10(2):246–61.
12. Willberg M. *Web Application Security Testing with OWASP Top 10 Framework*. Turku (Finland): Turku University of Applied Sciences (2019).
13. Helmiawan MA, Firmansyah E, Fadil I, Sofivan Y, Mahardika F, Guntara A. Analysis of web security using open web application security project 10. *Proc. 8th Int. Conf. Cyber and IT Service Management (CITSM)*. Pangkal, Indonesia. Piscataway (NJ): IEEE (2020). 2020 p.
14. Ventura R, Franco DJ, Akram OK. *A Novel VAPT Algorithm: Enhancing Web Application Security Through OWASP Top 10 Optimization*. arXiv Preprint arXiv:2311.10450 (2023).
15. King J. *Android Application Security with OWASP Mobile Top 10*. OWASP (2014).
16. Wen SF, Katt B. A quantitative security evaluation and analysis model for web applications based on OWASP Application Security Verification Standard. *Comput Secur.* (2023) 135:103532.
17. Simplice I, Fidel O, Kennedy GC, Okokpujie K. Enhancing information system security: a vulnerability assessment of a web application using OWASP Top 10 list. *Proc. Int. Conf. Smart Computing and Cyber Security*. Singapore: Springer Nature Singapore (2023).
18. Nilsson D, Åberg H. *HTML5 Web Application Security with OWASP*. Karlskrona (Sweden): Blekinge Institute of Technology, School of Computing (2013).
19. Tudela FM, Higuera JRB, Higuera JB, Montalvo JAS, Argyros MI. On combining static, dynamic and interactive analysis security testing tools to improve OWASP Top Ten security vulnerability detection in web applications. *Appl Sci.* (2020) 10(24):9119.
20. Rafique S, Humayun M, Hamid B, Abbas A, Akhtar I, Iqbal K. Web application security vulnerabilities detection approaches: a systematic mapping study. In: Lee R editor. *Proc. IEEE/ACIS 16th Int. Conf. Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. Piscataway (NJ): IEEE (2015).
21. Silva JCB. *Evaluation of Dynamic Analysis Tools in Detecting OWASP Top 10 Vulnerabilities*. MS Thesis, Universidade de Coimbra (2024).
22. Jakobsson A, Häggström I. *Study of the Techniques Used by OWASP ZAP for Analysis of Vulnerabilities in Web Applications*. Linköping (Sweden): Linköping University, Department of Computer and Information Science (2022).
23. Ksiezopolski B, Mazur K, Miskiewicz M, Rusinek D. Teaching a hands-on CTF-based web application security course. *Electronics* (2022) 11(21):3517.
24. Riadi I, Raharja PA. Vulnerability analysis of E-voting application using Open Web Application Security Project (OWASP) framework. *Int J Adv Comput Sci Appl.* (2019) 10(11):135–43.
25. Idris M, Syarif I, Winarno I. Development of vulnerable web application based on OWASP API security risks. *Proc. International Electronics Symposium (IES)*. Piscataway (NJ): IEEE (2021).
26. Cordella A. *Web Application Penetration Testing: An Analysis of a Corporate Application According to OWASP Guidelines*. Bologna: Alma Mater Studioum University Of Bologna (2018).
27. Goswami S, Krishnan NR, Mukesh, Swarnkar S, Mahajan P. Reducing attack surface of a web application by Open Web Application Security Project compliance. *Def Sci J.* (2012) 62(5):324–30.