**Exploring Web Application Vulnerabilities: Penetration Testing and OWASP Risk Analysis**

Nur Athirah Hanim Bikri1, a), Sumendra Yogarayan1, 2, b), Pa Pa Min1, 2, c), Thinesh Ganesan1, 3, d)

*1Faculty of Information Science and Technology, Multimedia University (MMU), 75450 Melaka, Malaysia*

*2Centre for Intelligent Cloud Computing, COE for Advanced Cloud, Multimedia University, 75450 Melaka, Malaysia*

*3Centre for Advanced Analytic, COE for Artificial Intelligence, Multimedia University, 75450 Melaka, Malaysia*

*b) Corresponding author: sumendra@mmu.edu.my*

*a) hanimnurathirah@gmail.com*

*c) papamin@mmu.edu.my*

*d) thinesh.ganesan@mmu.edu.my*

**Abstract.** Web applications have become a basic requirement in this digitalization era. As the internet usage grows and the number of web applications with it, everyone requires a certain degree of security to keep information secure and avoid cyber-attacks. Data leakage or unauthorized access are the potential problems triggered by any security vulnerability in a web application. Thus, penetration tests have emerged as an accepted technique to reduce vulnerabilities of web applications. Vulnerability Assessment and Penetration Testing (VAPT) is a security testing procedure that is utilised to validate and minimize the probable vulnerabilities of the web application systems. The project will specifically look at vulnerabilities that exist in web applications which shall be used to demonstrate the risk that could arise as a result of vulnerabilities. In the penetration testing procedure, pentester will identify security measure vulnerabilities and perform simulated attacks to determine the capacity of web applications in resisting possible threats. The outcomes of this project will prove the effective exploitation of the vulnerabilities discovered in web applications.

# INTRODUCTION

Web applications, leveraging knowledge such as HTML and JavaScript, facilitate a variety of activities, from online transactions to social networking [1]. They are convenient, but they are exposed to threats, like Cross-Site Scripting (XSS), SQL Injection, and Denial of Service (DoS) [2] [3] [4]. Through security vulnerabilities, these attacks steal user data, damage trust and disrupt services. There are now more than 1.98 billion web applications worldwide, and the market is growing by more than 6 percent each year, primarily due to the use of applications in e-commerce, fintech, and digital healthcare [5] [6]. Automated tools have increased the efficiency of identifying common vulnerabilities; however, they tend to miss logical flaws and as such, manual penetration testing must be incorporated to augment the gaps [7] [8]. Critical functions which depend on web applications include online banking, and HIPAA-covered healthcare systems in the United States highlight the importance of ensuring adequate security controls to reduce risk [9]. One of the examples of the XSS attack was posted through National Vulnerability Database (NVD) in 2024 of Health Care Hospital Management. It was identified that the web portal had several stored XSS vulnerabilities [10].

Vulnerability assessment is an important process to identify, categorize, and prioritize potential weaknesses in a system's security infrastructure [12]. Weaknesses, in most cases caused by software bugs, poor authentication scheme, or system design, present exploitable opportunities to the attackers [13]. Such tests that apply both automated and manual techniques play an important role in identifying and resolving security vulnerabilities before their exploitation [14]. Nevertheless, with the growing use of web applications as the foundation of various online services, they have also become the preferred target of malicious parties, resulting in major risks, including unauthorized disclosure, integrity loss, and denial of service [15] [16]. Although vulnerability detection tools have improved, there are still loopholes in the thorough protection of web applications concerning the changing threats. This study aligns with OWASP Top 10 framework, specifically addressing vulnerabilities in categories such as A01: Broken Access Control, A03: Injection, and A05: Security Misconfiguration [17] [18].

Penetration testing is a proactive method of investigating and fixing security vulnerabilities by method of simulated attacks [19] [20]. It helps organizations to identify vulnerabilities as though they were an attacker, either internally or externally and offers practical guidance to fix the security posture of an organization [21]. Although automated penetration testing tools offer scalability, manual testing is still important to resolve complex vulnerability and business logic issues [22] [23]. Past studies in the field of web application security have used penetration testing approach to discover vulnerabilities and have proved numerous techniques of attacks and tools. As an example, [24] used the Burp Suite to perform attacks against web applications, which is also limited to a specific tool, which may limit the extent of the testing. On the same note, [25] discussed the important vulnerabilities like Cross-Site Scripting (XSS) and SQL Injection and did not provide an in-depth look at the mitigation techniques. The authors adopted ISSAF and OWASP frameworks of penetration testing [26] and pointed out SQL Injection and brute force attacks. They were however limited to these frameworks, and it was suggested that future research involves more complex tools such as Maltego and SQLMap.

Despite significant contributions, several common issues are still existent, such as the dependence on particular tools (e.g., Metasploit or Skipfish) and theoretical discussions instead of practice. Such as [27] presented the techniques of web jacking but in a conceptual form only. Moreover, [28] and [29] emphasized the possible drawback of relying on single tools only, which may fail to cover other types of vulnerabilities. Such shortcomings explain why it is necessary to have a more unified and automated mechanism of web application testing that would encompass the use of various tools and frameworks, along with machine learning-based solutions to deal with scale and new threats.

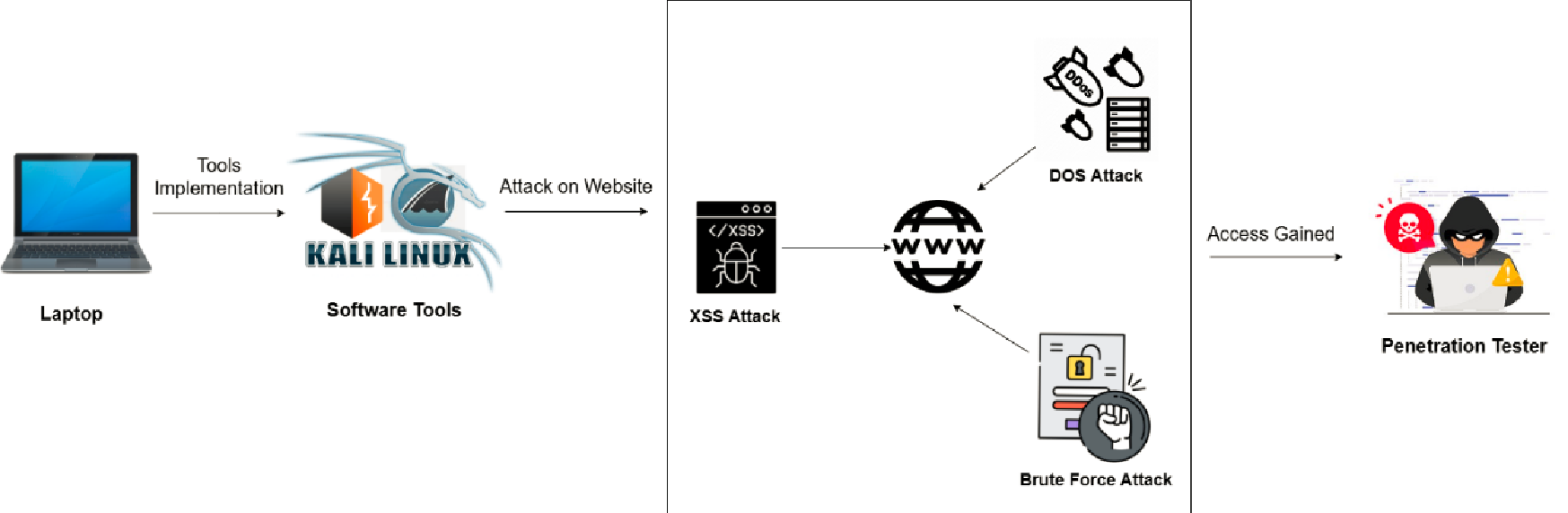
# METHODOLOGY

## Proposed Work

The research gaps are filled through the use of our proposed combination of a set of tools that include Burp Suite, Hydra, Wireshark, Slowloris, and Nikto in the web application vulnerability. Using these tools, we can offer a practical framework integrating automatic scanning, network traffic analysis, authentication testing and resource-targeted attacks therefore overcoming the limitations of dependency on tools, absence of practical execution and scope limitations.

This hybrid methodology balances the speed and coverage of automated tools with the depth of manual testing. Automated scanners like Nikto or Hydra detect common misconfigurations and brute-force points, while manual techniques enabled through Burp Suite’s Intercept/Intruder features, uncover business logic flaws and contextual weaknesses often missed by scanners. Tool selection was guided by specific strengths where Burp Suite was chosen over OWASP ZAP due to its broader plugin ecosystem, intercept interface, and payload manipulation features, particularly suited for XSS and input-based attacks. Hydra was used for high-speed brute force testing due to its native parallelism support. Nikto, while basic, served as a lightweight and fast inspection tool.

As illustrated in the workflow in Figure 1, the penetration testing process is carried out via a set of software tools integrated on the Kali Linux platform to attack web applications. In this process, a penetration tester will boot a laptop and apply a number of tools that are in the Kali Linux, including the Burp Suite, Hydra, Wireshark, Slowloris, and Nikto. They are used to launch certain attacks such as Cross-Site Scripting (XSS), Denial of Service (DoS) and brute-force attacks on vulnerabilities in the web application. These attacks are carried out in a manner that allows the tester to make use of any possible security vulnerabilities and gain unauthorized access to the website.



**FIGURE 1.** Penetration testing workflow

## OWASP Risk Rating Framework

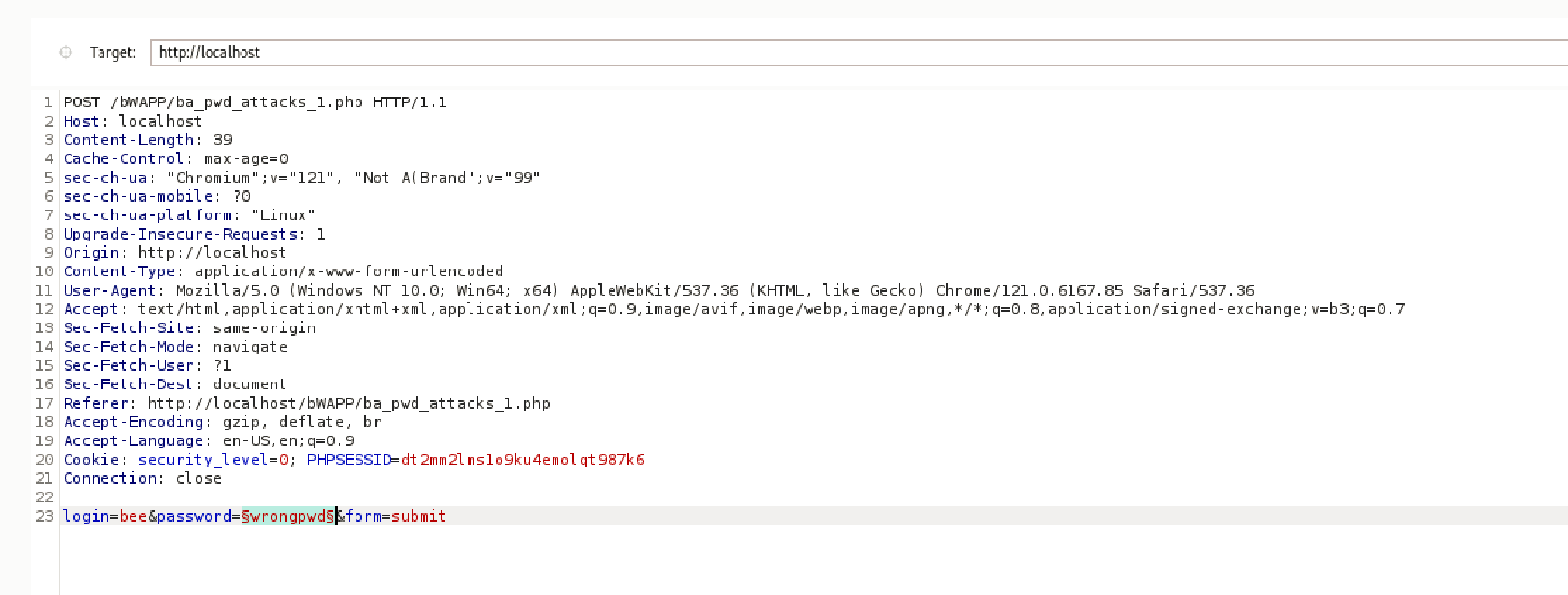
OWASP risk rating framework is a systematic way of rating security threats according to their possible impact and probability. Impact assesses the possible outcome of an exploit and is further split into technical impact, dealing with the immediate effects on systems, such as data breaches or service impairment, and business impact which takes into account secondary implications, such as reputation damage, loss of revenue and regulatory compliance. Likelihood on the other hand measures the likelihood of a vulnerability being exploited based on two sub-factors namely the threat agent factors that measure the skill level, motivation and opportunity of the potential attackers and the vulnerability factors that measure the ease of discovery and exploitation of vulnerability. Organizations can compute an overall risk rating (low, medium, high, or critical) by adding scores of these two categories, as illustrated in Table 1, in order to effectively prioritize the vulnerabilities and concentrate on improving the most serious threats.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE 1.** OWASP risk rating framework | | | | |
|  | **Overall Risk Severity** | | | |
| Impact | HIGH | Medium | High | Critical |
| MEDIUM | Low | Medium | High |
| LOW | Note | Low | Medium |
|  | LOW | MEDIUM | HIGH |
|  | Likelihood | | | |

# TESTING AND RESULT

## Brute Force

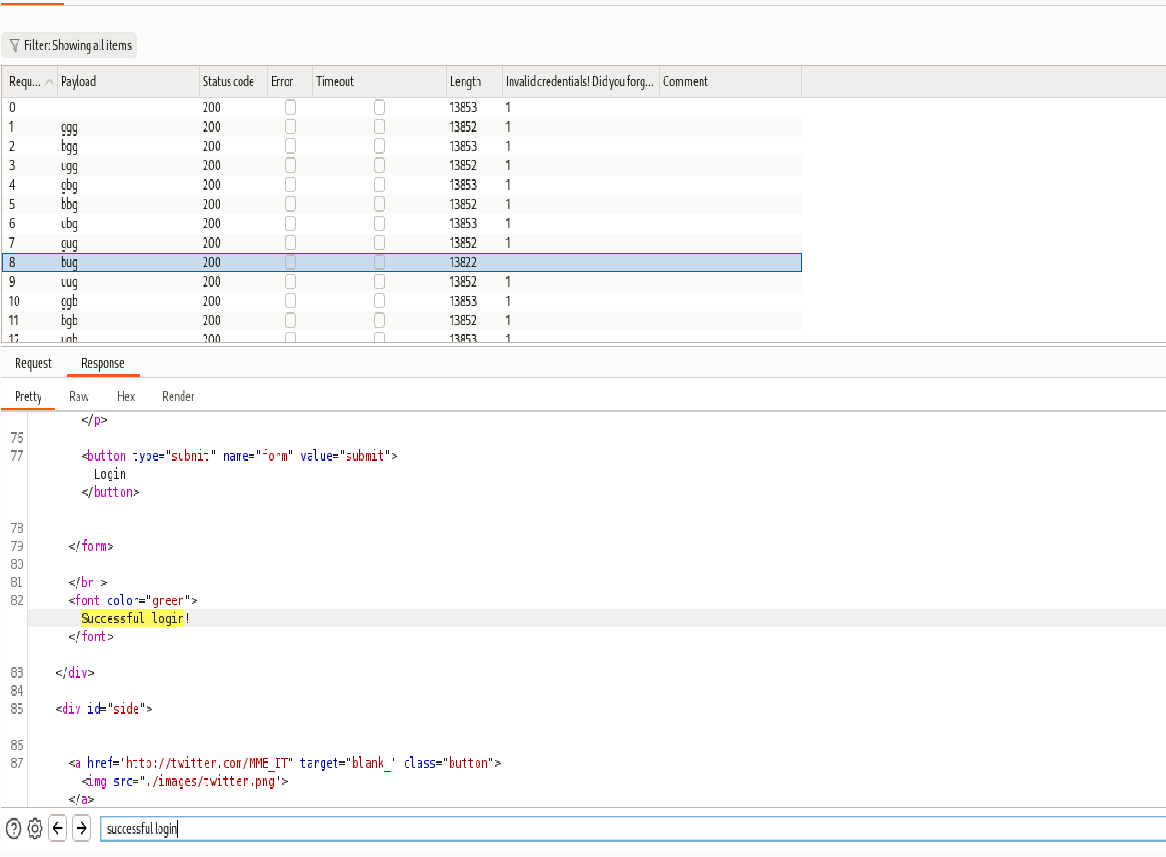
The Brute Force attack was tested using two tools: Burp Suite and Hydra. In Burp Suite, the attack begins by enabling the “Intercept” option to capture HTTP requests from the browser. The intercepted request was then sent to the “Intruder” tab, where the password parameter was selected as the payload position as shown in Figure 2.



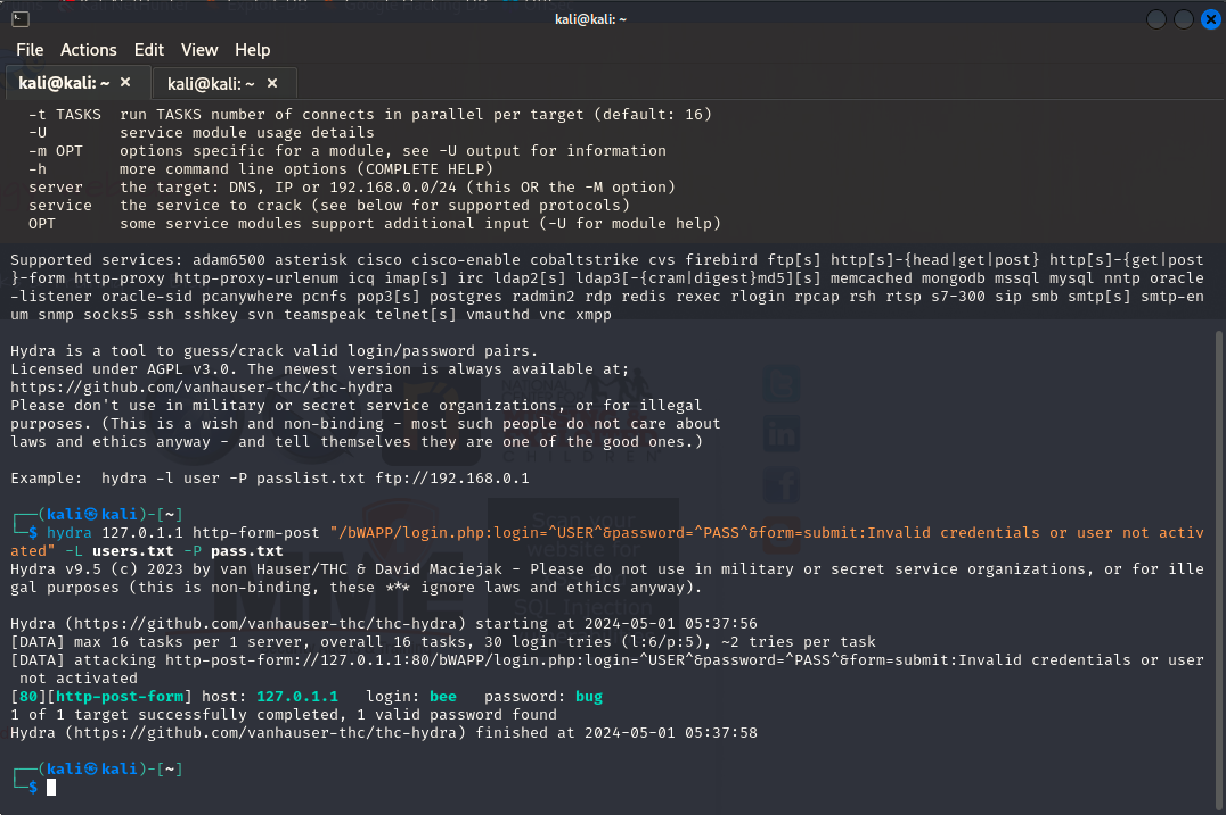
**FIGURE 2.** Burp suite intruder setup

A brute-force payload was configured using a specific character set, with the settings defined to generate all possible combinations of the characters *g, b, u*, with the length of 3. Grep settings were applied to identify responses indicating login success or failure. The results revealed that the correct password was “bug”, as this was the only credential that bypassed the error message and displayed the “Successful login” response as shown in Figure 3.

Similarly, Hydra was used to perform the brute force attack by preparing username and password lists in separate files. The attack was executed with a command specified the target IP, the login form path, and the expected error message. The Hydra tool successfully identified the username as “bee” and the password as “bug”, granting access to the bWapp application as shown in Figure 4.



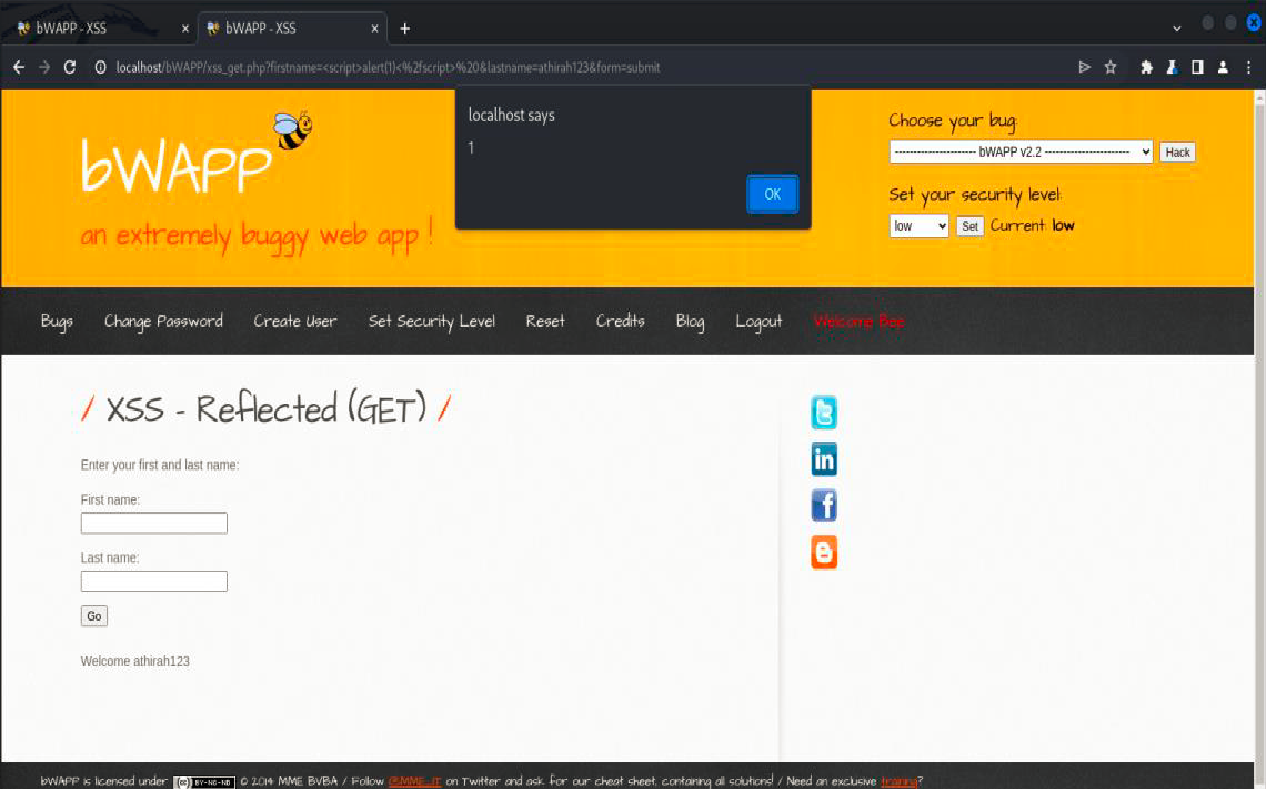
**FIGURE 3.** Login response identified



**FIGURE 4.** Hydra execution output

## Cross-Site Scripting (XSS)

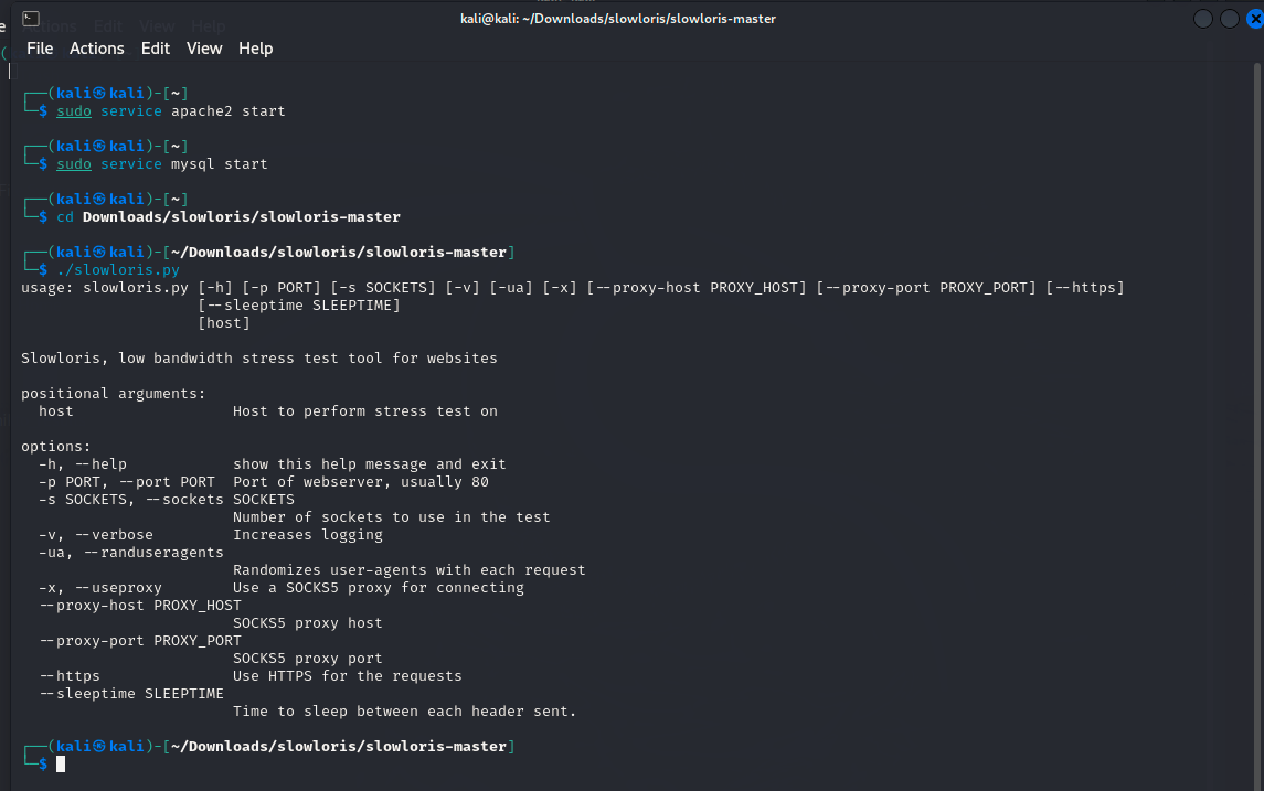
A reflected XSS vulnerability was tested using Burp Suite. Malicious script payloads were injected into input fields on a vulnerable web form and forwarded to Burp Suite’s “Intruder” tab. Payloads sourced from GitHub included *"<script>alert('XSS')</script>"* and various payload were also analysed such as script triggering alert boxes/extracting sensitive user information. The application executed injected scripts in the browser, confirming its susceptibility to reflected XSS attacks. One payload exposed the user’s credentials, while another displayed the injected variable like firstname as shown in Figure 5.



**FIGURE 5.** Script injected output

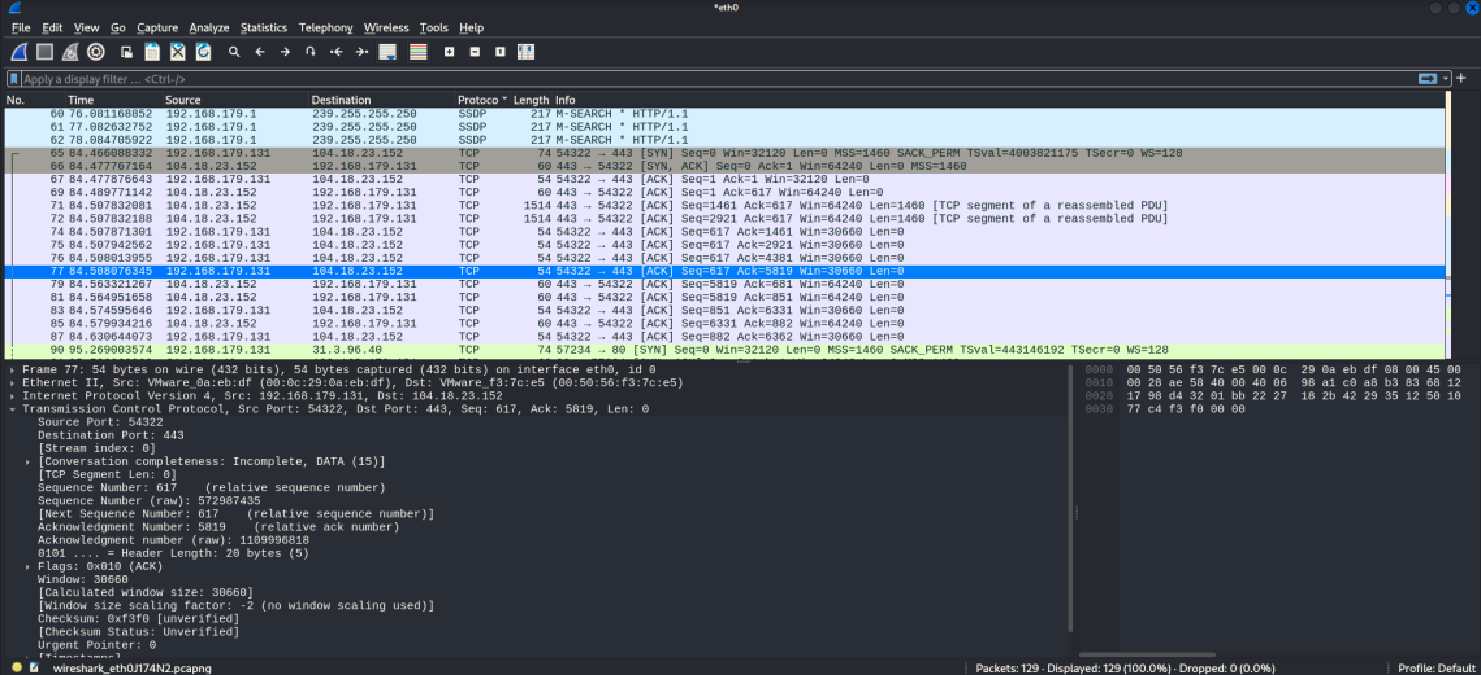
## Denial-of-Service (DoS)

The Denial-of-Service (DoS) attack was tested using Slowloris and Wireshark. The attack was executed by running the Slowloris python script with the target IP and port specified. Slowloris sends partial HTTP requests to keep server sockets open, gradually exhausting available resources as shown in Figure 6.



**FIGURE 6.** Slowloris execution command

During testing, the target server did not enforce connection-level rate limiting, idle timeouts, or maximum concurrent connection thresholds. These defenses typically close slow or malformed connections after a set time or block excessive connections from a single IP. Their absence allowed Slowloris to send partial HTTP headers and keep connections open indefinitely, progressively exhausting server resources. As the attack continued, legitimate users experienced delays or were unable to access the web page due to the server’s inability to process new requests. Network traffic captured via Wireshark revealed a pattern of ACK packets with no response, consistent with a socket exhaustion attack as shown in Figure 7. Once the execution was stopped, the server resumed normal operations, and the web page became accessible again.



**FIGURE 7.** DOS packet trace

## Security Weakness Analysis

Nikto was used to scan the application for security misconfigurations. The scan identified missing anti-clickjacking headers, wildcard crossdomain.xml entries, ETag exposure, and access to /server-status as shown in Figure 8. While Nikto tool were useful for early scans, it is prone to false positives. For instance, flagged directory listings or misclassified headers were verified manually to confirm actual exploitability. This underscores the need for manual validation of automated scan results.



**FIGURE 8.** Nikto vulnerability report

Using the OWASP risk rating framework, we evaluate the vulnerabilities identified during the testing phase based on their likelihood and impact. Each identified vulnerability is assessed in terms of its likelihood of being exploited and the potential technical and business impacts. By assigning scores to these categories, an overall risk rating is calculated as shown in Table 2 that shows the evaluation of the vulnerabilities found during the tests.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE 2.** Summary of OWASP risk rating | | | | |
| **Test Type** | **Likelihood** | **Technical Impact** | **Business Impact** | **Overall Risk Rating** |
| Brute Force | High | High | Moderate | High |
| Cross-Site Scripting (XSS) | Moderate | High | High | Hight |
| Denial-of-Service (DoS) | High | High | Moderate to High | High |
| Security Weakness | Moderate | Moderate to High | Moderate | Medium to High |

The likelihood of a brute force attack is high due to the ease of exploiting weak authentication mechanisms using tools like Hydra and Burp Suite. The technical impact is also high since unauthorized access can compromise sensitive data or lead to account misuse. However, the business impact is moderate, depending on the sensitivity of the data involved. The overall risk rating for brute force attacks is high. Besides, the likelihood of an XSS attack is moderate as it requires injecting malicious scripts into web pages, which may require some expertise. The technical impact is high because successful execution can lead to session hijacking, credential stealing, or unauthorized actions on behalf of the user. The business impact is also high as it can damage user trust and reputation. Thus, the overall risk rating for XSS is high.

The likelihood of a DoS attack using tools like Slowloris is high since it is relatively straightforward to execute. The technical impact is high as it disrupts the availability of the target web application, leading to service downtime. The business impact is moderate to high, depending on how critical the service is to the organization. The overall risk rating for DoS is high. On the other hand, vulnerabilities such as lack of anti-clickjacking protection and exposed server information are moderately likely to be exploited. Their technical impact ranges from moderate to high, as they can facilitate more sophisticated attacks. The business impact is moderate, as these issues can erode trust if exploited. Thus, the overall risk rating is medium to high, depending on the specific vulnerability.

# CONCLUSION AND FUTURE WORKS

The study carried out was effective to identify, test and assess the vulnerability of web applications through the tools Burp Suite, Hydra, Slowloris and Nikto. The study helped to present a well-organized plan of vulnerability classification according to the likelihood of exploitation and potential impact by correlating the results with the OWASP risk rating model. These findings reinstate the need for consistent, systematic testing against prevalent and advanced threats. Automated tools increased the speed of detection; however, manual testing was still relevant to logic-based vulnerability interpretation and scan results validation. This two-way approach promises a better risk profile. Future works would focus on extending the study towards mobile applications, APIs, and cloud-based systems. In addition, adversarial AI-based testing is also promising direction, which proposes dynamic payloads generation capabilities and anomaly detection. However, reproducibility, explainability of the model, and overfitting to certain patterns are some of the difficulties which need to be resolved prior to operational use.

# Acknowledgments

I would like to express my gratitude to the reviewers and all those who indirectly supported this project. Their constructive feedback and the resources available made the successful completion of this work possible.

# References

1. A. E. Hafez and M. M. Almustafa, “Detecting Security Vulnerabilities in Web Applications: A Proposed System,” *Int. J. Saf. Secur. Eng.* **14**, 627–634 (2024).
2. M. A. I. Mallick and R. Nath, “Navigating the Cyber security Landscape: A Comprehensive Review of Cyber-Attacks, Emerging Trends, and Recent Developments,” *World Sci. News* **190**, 1–69 (2024).
3. P. Tanakas, A. Ilias, and N. Polemi, “A novel system for detecting and preventing SQL injection and cross-site-script,” in *Proc. Int. Conf. Electr. Comput. Energy Technol. (ICECET)* (IEEE, Cape Town, South Africa, 2021), pp. 1–6.
4. A. S. Sthapit and P. Sapkota, “Analysis of web application security management in context of Nepal’s organizations,” M.S. thesis, University of South-Eastern Norway, 2022.
5. R. Aghili, Q. Qin, H. Li, and F. Khomh, “Understanding web application workloads and their applications: Systematic literature review and characterization,” in *Proc. IEEE Int. Conf. Softw. Maint. Evol. (ICSME)* (IEEE, 2024), pp. 474–486.
6. B. Alotaibi, “Cybersecurity Attacks and Detection Methods in Web 3.0 Technology: A Review,” *Sensors* **25**, 342 (2025).
7. Z. Chen, F. Kang, X. Xiong, and H. Shu, “A Survey on Penetration Path Planning in Automated Penetration Testing,” *Appl. Sci.* **14**, 8355 (2024).
8. N. S. Harzevili, A. B. Belle, J. Wang, S. Wang, Z. M. Jiang, and N. Nagappan, “A systematic literature review on automated software vulnerability detection using machine learning,” *ACM Comput. Surv.* **57**, 1–36 (2024).
9. N. Abbasi and D. A. Smith, “Cybersecurity in Healthcare: Securing Patient Health Information (PHI), HIPPA compliance framework and the responsibilities of healthcare providers,” *J. Knowl. Learn. Sci. Technol.* **3**, 278–287 (2024).
10. M. M. Aslam, Y. Z. Rufai, T. Ali, S. Ali, K. Kassim and S. Wasswa, “Introduction to industry’s fourth revolution and its impacts on healthcare,” in *Digital Transformation in Healthcare*, vol. 5 (2024).
11. L. Judijanto, D. Hindarto, and S. I. Wahjono, “Edge of Enterprise Architecture in Addressing Cyber Security Threats and Business Risks,” *Int. J. Softw. Eng. Comput. Sci. (IJSECS)* **3**, 386–396 (2023).
12. M. Walkowski, J. Oko, and S. Sujecki, “Vulnerability management models using a common vulnerability scoring system,” *Appl. Sci.* **11**, 8735 (2021).
13. Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, “A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions,” *Electron.* **12**, 1333 (2023).
14. R. Leszczyna, “Review of cybersecurity assessment methods: Applicability perspective,” *Comput. Secur.* **108**, 102376 (2021).
15. M. A. Almaiah, L. M. Saqr, L. A. Al-Rawwah, L. A. Altellawi, R. Al-Ali and O. Almomani, “Classification of Cybersecurity Threats, Vulnerabilities and Countermeasures in Database Systems,” *Comput. Mater. Continua* **81**, (2024).
16. S. Abdelkader, J. Amissah, S. Kinga, G. Mugerwa, E. Emmanuel, D-E. A. Mansour, M. Bajaj, V. Blazek and L. Prokop, “Securing modern power systems: Implementing comprehensive strategies to enhance resilience and reliability against cyber-attacks,” *Results Eng.* **2024**, 102647.
17. R. Umar, I. Riadi, and M. I. A. Elfatiha, “Security Analysis of Web-based Academic Information System using OWASP Framework,’ *Kinetik: Game Technol. Inf. Syst. Comput. Netw. Comput. Electron. Control* **9**, (2024).
18. L. H. Riberu and A. W. R. Emanuel, “Vulnerability testing and analysis using OWASP Top 10 on academic information system at University XYZ,” in *Proc. Int. Conf. Adisutjipto Aerosp. Electr. Eng. Informatics (ICAAEEI)* (IEEE, 2024), pp. 1–5.
19. D. J. Edwards, “Vulnerability assessment and penetration testing,” in *Mastering Cybersecurity: Strategies, Technologies, and Best Practices*, (Apress, Berkeley, CA, 2024), pp. 371–412.
20. R. Chauhan, R. Singh, R. Bartwal, R. Rawat, and R. Dhanalakshmi, “An investigation of ethical hacking: White-hat hackers,” in *Challenges in Information, Communication and Computing Technology* (2025), pp. 769–774.
21. D. J. Edwards, “Penetration testing,” in *Critical Security Controls for Effective Cyber Defense: A Comprehensive Guide to CIS 18 Controls*, (Apress, Berkeley, CA, 2024), pp. 527–554.
22. A. -D. Tudosi, A. Graur, D. G. Balan, and A. D. Potorac, “Research on Security Weakness Using Penetration Testing in a Distributed Firewall,” *Sensors* **23**, 2683 (2023).
23. A. Fatima, T. A. Khan, T. M. Abdellatif, S. Zulfiqar, M. Asif, W. Safi, “Impact and research challenges of penetrating testing and vulnerability assessment on network threat,” in *Proc. Int. Conf. Bus. Anal. Technol. Secur. (ICBATS)* (IEEE, Dubai, United Arab Emirates, 2023), pp. 1–8.
24. A. Kore, T. Hinduja, A. Sawant, S. Indorkar, S. Wagh, and S. Rankhambe, “Burp Suite extension for script-based attacks for web applications,” in *Proc. Int. Conf. Electron. Commun. Aerosp. Technol. (ICECA)* (IEEE, Coimbatore, India, 2022), pp. 651–657.
25. S. tul Hassan, “Analysis of Vulnerabilities in System by Penetration Testing,” *Pak. J. Sci. Res.* **2**, 22–25 (2022).
26. D. P. Anggraeni, B. P. Zen, and M. Pranata, “Security Analysis On Websites Using The Information System Assessment Framework (Issaf) And Open Web Application Security Version 4 (OWASPv4) Using The Penetration Testing Method,” *J. Pertahanan* **8**, 497–506 (2022).
27. K. Barik, K. Konar, A. Banerjee, S. Das, and A. Abirami, “An exploration of attack patterns and protection approaches using penetration testing,” in *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*, (Springer Nature, Singapore, 2022), pp. 491–503.
28. R. Karayat et al., "Web application penetration testing & patch development using kali linux," in *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 1392-1397, 2022, doi: 10.1109/ICACCS54159.2022.9785232.
29. R. Karayat, M. Jadhav, L. S. Kondaka, and A. Nambiar, “Web application penetration testing & patch development using Kali Linux,” in *Proc. Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, vol. 1 (IEEE, 2022), pp. 1392–1397.
30. M. Tabassum, S. Mohanan, and T. Sharma, “Ethical Hacking and Penetrate Testing using Kali and Metasploit Framework,” *Int. J. Innov. Comput. Sci. Eng.* **2**, 9–22 (2021).