Journal of Applied Technology and Innovation
Vol. **09,** Issue **2**, 2025**,** pp. **49-54**
e-ISSN:2600-7304-ISSN
DOI: https://doi.org/10.65136/jati.v9i2.29

Web Application Common Vulnerabilities and Preventions.

Julia Juremi [1*], Ali Allouche [1], Kazi Farhan Ishraq [1]

[1] Forensics & Cybersecurity Research Centre, Asia Pacific University of Technology & Innovation Kuala Lumpur, Malaysia
*Corresponding Author: julia.juremi@mail.apu.edu.my

Abstract
World Wide Web also known as www has been used by almost everybody in the world for education, entertainment, commerce and many more. As World Wide Web expands more it has become vulnerable for cyber-attacks from cybercriminals across the world. This paper aims to explore the most common web application attacks and vulnerabilities, what is the cause of it, what affect it might have, and how to prevent it from happening by using best practices and tools. In this paper OWASP top 10 and modern security frameworks will be focused on to understand how to secure web applications against threats. Various tools and best practices will also be discussed to mitigate these risks and protect sensitive data.

Keywords: *Web Application Vulnerabilities, Cross-Site Scripting, Web Security, Injection, Access Control, Secure Design, Owasp Top 10, Penetration Testing.*

1. Introduction

The World Wide Web has developed with breath-taking pace, changing the face of worldwide infrastructure to make it a sole utility for education, entertainment, marketing, and critical communication. Unfortunately, it creates new problems, since with a rise in website usage, there is a direct rise in vulnerability. As Kaur et al. (2022) might suggest, it is no longer a choice to be made, but it is a basic necessity to focus on web security. Since there is no concrete defensive mechanism, organizations can face immense dangers, along with possible hacking of critical information like shipping address, credit card number, and user password. With losses going beyond what can be imagined, a hacking theft may cause a catastrophe in financial as well as reputation terms to any firm, no matter what size, as evident in major hacking campaigns, which came with major incidents like SolarWinds, analyzed in a study by Bhatt & Patwa (2021).

In order to address this increasingly hostile environment, however, the Open Web Application Security Project (OWASP) plays an crucial role in providing an annual list of necessary security considerations for web development that continues to gain consensus among industry experts (OWASP Foundation, 2021a; 2021b; 2021c). Though necessary for defensive preparedness, however, studies published in the past few years suggest that it is no longer enough to solely depend upon technology as an antidote for web security concerns since the human factor continues to contribute to this threat to a large extent (Charalambous & Stavrou, 2025). The increasing adoption of DevSecOps, which seeks to incorporate security processes and procedures directly within the developmental environment of an application, underscores the need to address this issue in a pro-active manner for detecting vulnerabilities (Rida & El Hore, 2024).

The aim of this current review is to explore commonly found web application attacks and security vulnerabilities. This study reviews reasons behind the security weakness and the potential operational consequences, as well as methods for preventing them. In this context, this current study soughtt to provide a comprehensive overview based on the OWASP Top 10 and current research articles in the field of how to protect web applications against current threats.

2. Methods

This study employs a narrative literature review methodology to critically synthesize the evolving landscape of web application through the integration of industry applications and the contemporary scholarly findings. According to Snyder (2019), it is imperative that one find ways to connect disparate fields so as to match theoretical notions of vulnerability to practical notions of protection and vice versa. Rather than restricting the analysis to a rigid quantitative set, this review prioritizes high-impact technical reports and comparative studies in the bid to understand the dynamic nature of contemporary web threats. The basis of this model lies in the security risk classifications curated by the OWASP Foundation (2021a).

The selection of literature was conducted through thematic filtering, targeting peer-reviewed articles and conference publications published between 2020 and 2025. This period allows for an assessment of recent architectural shifts, such as CI/CD pipelines and DevSecOps architectures. The evaluation has organised varying pieces of literature into two well-structured themes: "Vulnerability Mechanics" and "Mitigation Strategies." The selection of themes is similar to that conducted by authors of interest, specifically Kaur et al. (2022), allowing for a direct examination of how well current solutions can handle identified threats. The application of static vulnerability assessment and dynamic testing approaches provides a well-rounded and current assessment of web application protection.

3. Findings

Expanding upon the foundation of the established narrative review framework (Snyder, 2019), this section unveils the amalgamated discoveries concerning pivotal web application vulnerabilities. The ensuing examination meticulously outlines particular attack pathways, such as injection mishaps and access control weaknesses, while assessing the efficacy of contemporary remediation tactics and security approaches.

3.1 Injection

Injection is one of the top web application vulnerabilities and commonly used by cybercriminals (OWASP Foundation, 2021a). There are types of injections and they are Cross-Site Scripting, SQL injection and External Control File name or Path. These types of attacks happen when an attacker inject a malicious script (e.g. SQL scripts, HTML, Server-Side and JAVASCRIPT) in to the input fields of the web application enabling unauthorized access to databases (Gupta & Gupta, 2015). By accessing the database all the users they are registered and all the information weather it was passwords, transaction information's or credit card numbers will be in the hand of the attacker. According to OWASP averagely 48% of the web applications has been tested and it had an average of 7% exploitation which isn't a low number.

Web application is vulnerable for injection when input fields are not validated or sanitized, dynamic queries or non-parameterized calls are used without context-aware escaping, and Hostile data usage like object-relation mapping or SQL queries to extract sensitive information for database. The best way to prevent such attacks is to check the source code of the web application if its vulnerable to injection or no. Testing all parameters either manually if the website isn't very large or by using an automated tools such Burp Suite (Abualese & Al-Rousan, 2023) and to check for it before the production deployment static, dynamic and interactive application security testing tools could be included in the CI/CD pipelines for detection due to the mechanism of it and how to works.

3.2 Insecure Design

Insecure design has a wide range of vulnerabilities represented as missing or ineffective control design (OWASP Foundation, 2021b). Insecure design is different than other web application vulnerabilities which is insecure implementation, they have different root causes and remediation. A secure design can still have implementation vulnerabilities that may be exploited which could be fixed by either reviewing and updating the source code or by using other methods, but an insecure design is very hard to fix once the web application has been deployed because web application security measures where never implemented at first to defend from curtain attacks and to do so it will need and overhaul and takes time to be done. Secure design is a methodology that constantly checks threats and ensure that the source code is well designed and tested (Konev et al., 2022). Threat modeling should be integrated into refinement session or in web application development stage, looking for changes in access control and other security controls. Understand what the corporate requirements are with consideration of CIA and AAA could help to Secure development lifecycle knowing that will help in selecting a secure design pattern, component library, tooling and threat modeling while developing the website. To prevent such occurrence, while in development lifecycle get help from application security professionals to check whether the website is a secure design or no and taking measure to fix, using threat modeling for access control and critical authentication, integrate security in user story while developing the web application, check connections between frontend and backend for any vulnerabilities that might occur and limit resource consumption by users.

3.3 Broken Access Control

Broken access control is also one of the major web application vulnerabilities that allows attacks to have unauthorized access by bypassing restrictions and having access to the system and sensitive data (Zhong, 2023). This happens when security measures for user control is not done properly or not even founded to begin with. Broken access control occurs when users can have access to admin privileges or restricted URLs without proper authentication. Bypassing these privileges could be done by altering the HTML page, the internal application state, by attacking tools that modify API requests or by poor web application by misconfiguring CORS, accessing API with missing access controls (POST, GET, DELETE) abusing the token session of logging in by using the same one each time without generating a new token for each login (OWASP Foundation, 2021c). According to OWASP average of 48% web applications were tested and had an average of 4% incidence rate. To prevent broken access control a developer needs to implement role-based access control or attribute access control to limit the access of users and let them use what they need only, this will protect the data and have a proper authorization, limiting the use of Cross-Origin Resource Sharing (CORS) usage, disabling web server directory listing and make sure backup files are not present in the web roots, alerts for admins of log access control failures when needed, limit API request rate and control access for automated attacks and using short-lived and new JWT or login token each time a user logout and wishes to login again.

3.4 Software and Data Integrity Failures

Software and data integrity failures happens when vulnerabilities that comes when a system fails to protect the software updates, sensitive data and continuous integration/continuous delivery (CI/CD) pipelines from been altered by someone who got unauthorized access (OWASP Foundation, 2021c). This type of vulnerability occurs when a web application is relying on plugins, libraries or modules used from untrusted sources, repositories and content delivery networks or CI/CD that is managed insecurely, this could lead to unauthorized access, a malicious code, corruption of data or whole control of system in some cases (Rida & El Hore, 2024). This leads to lose of integrity and without its CIA won't be accomplished. According to OWASP an average of 45% that had an average of 7%. One attack was made at a nation-state level that had targeted around 18,000 organization and affected around 100, It was one of the biggest breaches that happened using this vulnerability, its known as SolarWinds malicious update (Bhatt & Patwa, 2021). To prevent from such vulnerability been exposed using digital signatures or similar mechanism to have integrity of software or data are coming from the expected source and they

haven't been altered. Ensuring that libraries and dependencies used to build the web application are coming from trusted repositories and content delivery networks (CND's).

When changing the source code or any configuration has been done a review process is needed to minimize the chance of having a malicious code or configuration that could be found in the software pipeline after updating. Having a proper configuration and access control for the CI/CD to ensure integrity in the code while in build and deploy processes. And making sure data without a digital signature or any kind of encryption is not sent to untrusted clients with having some kind of integrity to check whether the data was altered. Using software supply chain security tool such as OWASP Dependency Check or OWASP CycloneDx can verify that components used to build the web application doesn't contain the known vulnerabilities.

3.5 Methodologies and Techniques

After addressing some of the web application vulnerabilities and how it works, what causes it and how to prevent it. In this methodology and techniques, it is pertinent to consider more in-depth ways on how to take several steps that can help in building and securing a web application from cybercriminals attacks by talking about security measures and tools that can help with detecting the vulnerabilities.

3.5.1 Threat Modeling

Threat modeling process should be one of the first security measure to be taken while building a web application, it works by analyzing the system architecture and identifying potential attack vectors, entry points for attacks and how data flow throughout the web application (Konev et al., 2022). Threat modeling process aims to exploit vulnerabilities that attacks might use to hack into the web application such as injection, broken access control and software and data integrity failures. Tools for Threat Modeling:

- Microsoft Threat modeling tool.
- OWASP Threat Dragon.

3.5.2 Interactive Application Security Testing (IAST)

IAST combines both Static Application Security Testing and Dynamic Application Security Testing (SAST & DAST), it works by analyzing the source code while being executed giving real-time feedback on what vulnerabilities are found (Kaur et al., 2022). Using IAST is effective to detect vulnerabilities such as software and data integrity failures, injection and access control, but keep in mind that continuous monitoring of the source code after updating and checking again is essential. IAST could also be used while in production environments to detect vulnerabilities dynamically during runtime and its best to do manual penetration testing to ensure comprehensive coverage. Tools for IAST:

- Contrast Security
- Hdiv Security
- Seeker

3.5.3 Manual Penetration Testing

Everything automated is easier to use but that doesn't mean not to do manual penetration testing as its crucial for complex vulnerabilities such as broken access control and business logic flaws that automated tools might miss. Since manual testing takes a lot of time to do focusing on the critical parts is the best way to go, testing parts like authentication mechanisms, session management, user roles and privilege escalation by trying to access the admin privilege from a user privilege (Abualese & Al-Rousan, 2023). Preform fuzz testing to identify to see how the web application reacts to unexpected or invalid inputs to prevent injection. Using automated tools is good most of the time but that doesn't mean not doing manual penetration testing. Tools for Manual Penetration Testing:

- Kali Linux tools (Metasploit, Hydra and Nmap)

- Burp Suite Pro
- OWASP Web Security Testing Guide.

3.5.3 Continuous Monitoring and Logging

Continuous monitoring and logging are essential to detect broken access control and software and data integrity failures. It works by analyzing real-time logs and detecting unusual behaviors such as unauthorized access attempts or suspicious code changes. Implementing Security Information and Event Management (SIEM) techniques to centralize logs and monitor security incidents, enable logging for all critical areas related to user authentication, file access and administrative actions happening on the web application helps in monitoring and taking quick counter measures if anything to happen, there are tools which can detect and alert the security team for suspicious actions to take immediate action (Rida & El Hore, 2024). Tools for Continuous Monitoring and Logging:

- Splunk
- Elastic Stack (ELK)
- Graylog

3.5.4 Security Awareness and Training

Security awareness and training is a must, no matter what security measures was taking one small mistake from an employee who doesn't have security awareness will fall victim for an attack from a cybercriminal (Charalambous & Stavrou, 2025). To overcome such thing all employees including developers and IT staff should be trained on security awareness and what to do and what to not. IT developers should get proper training on how to develop and write a secure code to prevent injection and broken access control. This training session should be conducted on regular basis and providing up-to-date guidance by addressing common and new vulnerabilities which could be referred from OWASP, simulating phishing attacks and social engineering cases on all employees to keep them aware and to avoid future incidents.

4. Discussion

The findings from the review of contemporary literature suggests that web application security has progressed from being a static remediation of vulnerability patches to an extremely complicated, dynamic system that warrants perpetual monitoring. The observations accentuate that despite traditional threats such as Injection and Broken Access Control being exceptionally widespread (OWASP Foundation, 2021a; 2021c), the remediation strategies employed to counter them need to be vastly more cohesive than previously considered. A strong recommendation of this research advocates for current observation that despite being highly imperative for quick-point resolution. According to Abualese and Al-Rousan (2023), automated web scanners prove to be highly ineffective when employed alone. Rather, it proves that current armed architecture misalignment warrants highly imperative proactive strategies such as threat modeling (Konev et al., 2022). Moreover, it can be observed that data clearly advocates that highly imperative technological strategies can prove to be utterly ineffective if they lack considerations for human factor involvement in security operations. According to Charalambous and Stavrou (2025), security awareness being imperative to code integrity.

However, this review also has some limitations. This review being a narrative synthesis and not a quantitative systematic review may not focus on some very specific and novel zero-day attacks not yet categorized within the OWASP Top 10 high-impact vulnerabilities. Moreover, the ever-increasing rate of technological developments implies that the efficiency of the parameters for securing the code analyzed by Kaur et al. (2022) may change as attackers develop novel ways to bypass these parameters. Therefore, the results from this review may give a very comprehensive outlook to the current state of threats but will require longer-term research to assess the efficiency of incorporating IAST tools into agile CI/CD pipelines as proposed by Rida and El Hore (2024).

5. Conclusion

As the World Wide Web is expanding and new vulnerabilities and techniques are being discovered or created securing web application from cyber threats will always be a challenge. This paper has discussed some of the most common vulnerabilities found in web application such as injection, insecure design, broken access control and software and data integrity failures, going in-depth on what causes them their impact on web applications and how to prevent them from happening. By taking consideration of best practices like secure coding, threat modeling and automated and manual testing and continuous monitoring ensuring to minimize attacks and protecting sensitive data. In addition, using tools like burp suite, OWASP tools and security guide, Kali Linux and SIEM solutions, it's also crucial to train all employees including IT staff and developers on phishing attacks and how to have a secure web application by integrating security into every phase of the development lifecycle. As technology evolves so will the attackers' ways will do, finding new strategies on how to safeguard web applications from cybercriminals will always be there.

References

Abualese, H., & Al-Rousan, T. (2023). A comparative study of web application security scanners for vulnerability detection. *i-manager's Journal on Software Engineering*, *17*(4), 1-8. https://doi.org/10.26634/jse.17.4.19813

Bhatt, S., & Patwa, N. (2021). Solar Winds Hack: In-Depth Analysis and Countermeasures. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1-6. https://doi.org/10.1109/ICCCNT51525.2021.9579611

Charalambous, M., & Stavrou, E. (2025). Redesigning cybersecurity awareness-raising and training programs: insights from professionals on knowledge, skills and educational practices. *Information & Computer Security*. Advance online publication. https://doi.org/10.1108/ICS-04-2025-0163

Gupta, S., & Gupta, B. B. (2015). Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art. *International Journal of Systems Assurance Engineering and Management*, *8*(S1), 512–530. https://doi.org/10.1007/s13198-015-0376-0

Kaur, S., Gupta, S., & Limbasiya, T. (2022). A comparative study of web application security parameters: Current trends and future directions. *Applied Sciences*, *12*(8), 4077. https://doi.org/10.3390/app12084077

Konev, A., Shelupanov, A., Nabieva, A., & Belov, D. (2022). A survey on threat-modeling techniques: Protected objects and classification of threats. *Symmetry*, *14*(3), 549. https://doi.org/10.3390/sym14030549

OWASP Foundation. (2021a). *A03:2021 - Injection*. OWASP. Retrieved from https://owasp.org/Top10/A03_2021-Injection/

OWASP Foundation. (2021b). *A04:2021 - Insecure Design*. OWASP. Retrieved from https://owasp.org/Top10/A04_2021-Insecure_Design/

OWASP Foundation. (2021c). *A01:2021 - Broken Access Control*. OWASP. Retrieved from https://owasp.org/Top10/A01_2021-Broken_Access_Control/

OWASP Foundation. (2021c). *A08:2021 - Software and Data Integrity Failures*. OWASP. Retrieved from https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

Rida, A., & El Hore, A. (2024). Towards DevSecOps model for multi-tier web applications. *ITM Web of Conferences*, *69*, 04018. https://doi.org/10.1051/itmconf/20246904018

Snyder, H. (2019). Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104, 333-339. https://doi.org/10.1016/j.jbusres.2019.07.039

Zhong, L. (2023). A survey of prevent and detect access control vulnerabilities. *arXiv preprint arXiv:2304.10600*. https://doi.org/10.48550/arXiv.2304.10600