ACIG
APPLIED
CYBERSECURITY
& INTERNET
GOVERNANCE

NASK

Vulnerabilities of Web Applications: Good Practices and New Trends

Mateusz Nawrocki | Cracow University of Technology, Poland | ORCID: 0009-0007-5370-3497

Joanna Kołodziej | NASK – National Research Institute, Poland | ORCID: 0000-0002-5181-8713

Received: 11.10.2024

Accepted: 20.12.2024

Published: 25.12.2024

Cite this article as:
M. Nawrocki, J. Kołodziej, "Vulnerabilities of web applications: Good practices and new trends," ACIG, vol. 3, no. 2, 2024, pp. 122–143. DOI: 10.60097/ACIG/199521

Corresponding author:
Mateusz Nawrocki, Cracow University of Technology, Poland; E-mail: mateusz.nawrocki@pk.edu.pl
0009-0007-5370-3497

Abstract

Web application security remains a critical challenge in mitigating vulnerabilities that expose sensitive data and systems to cyberattacks. This paper addresses the recent trends in the vulnerability of web applications to cyberattacks. It explores implementing and evaluating security mechanisms in web services guided by the Open Web Application Security Project's (OWASP) Top 10 framework. The OWASP analyser – a test application prepared to simulate the broken access control, Structured Query Language (SQL) Injection, and cross-site scripting (XSS) attacks – was executed in three realistic scenarios: web applications without any protection mechanism, essential safeguards, and advanced measures. The experimental results demonstrate the effectiveness of layered security strategies and highlight the best practices, such as role-based access control, secure cryptographic methods, and comprehensive logging. The analysis highlights the need to embed security throughout Web applications' implementation and use cycle. While advanced measures, such as encryption and real-time monitoring, increase resilience to sophisticated attacks, even basic practices can provide significant application protection if applied consistently.

Keywords

cybersecurity, vulnerability, web application, OWASP TOP 10

1. Introduction

Web applications have become deeply embedded in various fields and aspects of functioning in the IT era. It isn't easy to consider modern e-commerce management systems, communications, entertainment, and banking and financial services without sophisticated, intelligent, and responsive web services, recently supported by artificial intelligence (AI). The Internet was recognized in the early 1990s as the sixth primary mass medium in civilisation's development. Thus, web applications have become the foundation for developing modern digital technologies. However, the ubiquity of digital applications makes them prime targets for cyberattacks. Security gaps and any vulnerability to external manipulation are exploited to steal data, users' identities, and, finally, to obtain specific financial benefits [1].

Security in IT refers primarily to ensuring the stability and resilience of various applications, systems, and data against unauthorised attacks that may result in illegal access to these resources. Over the past few years, this issue has been a frequent topic of commercial reports prepared for various institutions, from global agencies and government structures to scientific publications of interest mainly to the academic community. An example of such a publication is the work of Al-Ibrahim and Al-Deen [1], which describes the principal vulnerabilities of educational and research-related websites and services and methods to counteract the poisoning of content published there. The authors point out differences like these threats depending on the ownership structure of the university or school and the education profile. An example of a publication aimed at e-commerce environments is the report prepared by Thuraisingham et al. [2], which – in addition to threats and the most common attacks – describes tools to support methods of controlling access to the infrastructure and resources of a given company as well as secure systems for managing workflow in a company or an organisation.

Within the rapidly evolving IT sector, threats are also undergoing continuous transformation. As the amount of sensitive data available online increases, so does the need for tools to protect it. Effective data protection systems for online systems should work based on the following basic principles:

- Separation of databases from applications (installation on different servers)
- Encryption of data files and backups
- Widely used firewalls and other methods to limit access to sensitive data.

However, using even the strictest procedures and the most perfect tool does not provide a 100% guarantee of protecting data and resources from unauthorised access and use. For example, when identifying and analysing threats such as phishing, it is essential to remember that these attacks often take advantage of human naiveté and inattention by enabling unauthorised access to sensitive data [3]. Today, user behaviour and preferences are the weakest links in the security chain [4].

The Open Web Application Security Project (OWASP) initiative [5] has played a fundamental role in identifying and developing prevention methods for Web application security's highly complex thematic horizon. OWASP is not only a project but a global community that makes efforts to improve web application security. This community's main activities are identifying and compiling complex taxonomies, ranking threats, and developing strategies and guidelines for mitigating and eliminating security vulnerabilities in web applications. Every 3 years OWASP publishes Top 10 reports on the most critical vulnerabilities affecting the security of web applications, highlighting areas that require focused attention and the implementation of appropriate protective mechanisms [6].

The research presented in this paper aims to briefly analyse web application vulnerabilities and evolving trends in developing strategies for securing these applications by comparing and analysing the last two editions of the OWASP Top 10 reports from 2017 and 2021. Understanding the nature and sources of web application vulnerabilities to attack and manipulation is paramount in an era where web applications are integral parts of our digital lives.

Based on the latest OWASP 2021 report, an OWASP analyser (OA) tool was developed and installed to illustrate web application vulnerabilities to specific attacks. OA is a hybrid web application combining features frequently encountered in social media platforms, e-commerce websites, and content management systems (CMS). It comprehensively tests various vulnerabilities and security defences in a single web environment. Through deploying multiple layers of security, ranging from basic defence mechanisms (or lack thereof) to approaches from OWASP Top 10 recommendations, OA monitors exploitation of the specific vulnerabilities and assesses which defence mechanisms best mitigate or prevent possible attack scenarios.

The experiments conducted using AO were aimed at identifying critical vulnerabilities of web applications based on the guidelines of the 2021 edition of the OWASP Top 10 report. Three application variants were implemented – from a version with no protection through an iteration containing basic security measures to a robust configuration using advanced defences. Penetration tests were conducted using popular security tools, such as Burp Suite [7], simulating Structured Query Language (SQL) Injection, and XSS and phishing attacks. The effectiveness of the defence methods used in the tests was evaluated, as was the difficulty level in bypassing each security measure. The experimental analysis concludes with recommendations and insights to raise awareness among developers and end users of the most prevalent cyber security threats.

The remainder of the paper is organised as follows. Section 2 outlines the main tenets of the OWASP Top 10 reports. The last two editions of these reports were compared, and a simple comparative analysis highlighted changes in the threat landscape and trends in the development of effective web application security tools. Section 3 describes the architectural model concepts, functions, and relationships between the main components of the OA application. The experimental analysis and results obtained are described at length in Section 4. Section 5 lists the most important web application security guidelines. The work concludes with Section 6.

2. OWASP Top 10 – A Review of 2017 and 2021 Editions

OWASP is a global non-profit organisation that brings together security experts and developers striving to improve the security of web applications. OWASP reports have become a roadmap for the focused community. They set trends in the security market for modern intelligent web services.

The first OWASP Top 10 report was released in 2003, with subsequent updates following approximately every 3 years: 2004, 2007, 2010, 2013, and 2017. The latest revision was published in 2021, suggesting that a new edition may soon be on the horizon. Each release reflects shifts in the threat landscape, incorporating new vulnerabilities and attack vectors that emerge alongside evolving technologies [8].

This section presents a brief comparative analysis of the last two editions of the OWASP Top 10 reports: the 2017 edition and the 2021 edition. The analysis focuses primarily on the evolution of web application vulnerabilities over just 3 years. Comparing these two OWASP reports shows how crucial the OWASP community is in shaping web application security practices.

2.1. Review of the 2021 Edition of OWASP Top 10

Below is a short review of the vulnerabilities in the 2021 edition of the OWASP report, which outlines the changes compared to the 2017 edition, along with the key threats and recommended defences.

1. **A01 – 2021: Broken access control.** It occurs when applications fail to enforce access restrictions properly and give unauthorised users access to the data. This vulnerability is the most serious web application security risk (5th position in the 2017 edition).
2. **A02 – 2021: Cryptographic failures.** It results from insufficient or improper use of cryptography, such as storing passwords in plain text or using outdated algorithms. Previously known as A03 – 2017: Sensitive Data Exposure, it often leads to sensitive data exposure.
3. **A03 – 2021: Injection.** It involves unvalidated user input reaching an interpreter (e.g. databases), leading to the execution of unintended commands. Defensive measures include parameterised queries and thorough validation of all inputs. The 2021 edition contains XSS.
4. **A04 – 2021: Insecure design.** A new category that highlights shortcomings in the early design stages, such as neglecting threat modelling or risk assessment.
5. **A05 – 2021: Security misconfiguration.** This covers a wide range of misconfigurations, like leaving default credentials unchanged, enabling debug modes in production, or inactive unnecessary features. It was A06 in the 2017 edition.
6. **A06 – 2021: Vulnerable and outdated components.** It addresses risks of running unsupported or outdated software components, libraries, and frameworks. Regular updates and dependency checks are critical for mitigation.
7. **A07 – 2021: Identification and authentication failures – formerly known as Broken Authentication.** This focuses on weak passwords, insufficient multi-factor authentication (MFA), or poor session management. Enforcing strong password policies and secure session handling is crucial.
8. **A08 – 2021: Software and data integrity failures.** This emphasises maintaining the integrity of application code and data, such as verifying software updates via digital signatures and using secure serialisation – a new category in 2021.
9. **A09 – 2021: Security logging and monitoring failures.** This category reflects the importance of logging and monitoring security events. Without proper logs or alerting mechanisms, attacks may go unnoticed for long periods.

10. **A10 – 2021: Server-side request forgery (SSRF).** It allows attackers to manipulate server-side requests, potentially accessing internal systems or sensitive data that would otherwise be restricted.

2.2. Comparison of the 2017 and 2021 Editions

The results of a short comparative analysis of both OWASP Top 10 editions are presented in Fig. 1.

The 2017 report outlined 10 critical threats: injection, broken authentication, and security misconfiguration. The main changes in the 2021 report can be defined as follows:

1. **Three new categories**
   a. *Insecure design* emphasises addressing security considerations during the initial design phase.
   b. *Software and data integrity failures* focuses on code and data integrity issues, including secure software updates and serialisation methods.
   c. *C. server-side request forgery (SSRF)* highlights vulnerabilities allowing attackers to manipulate server-side requests to access internal resources.
2. **Renamed and merged categories**
   Particular vulnerabilities were combined or renamed to reflect better current cybersecurity challenges (e.g. *broken authentication* became *identification and authentication failures*).
2. **Greater emphasis on secure design**
   The 2021 edition underlines the need to integrate security throughout the entire application lifecycle, rather than viewing it solely as an implementation concern.

A comparative analysis of the 2021 and 2017 versions of the OWASP Top 10 reveals several observations about changing trends in web application security vulnerabilities. These observations have substantial implications for security practitioners, developers, and organisations looking to strengthen their web application security measures. One finding is the continued presence of three threats that emerged in 2017. This means that injection, broken access control and cryptographic failure threats are still relevant and must be prioritised consistently [8]. The 2021 report updates the descriptions and scope of some vulnerabilities to account for the dynamics of changes in the development of modern web application architectural models.

Two new classes of vulnerabilities featured in the 2021 edition; “Software and Data Integrity Failures” and “Unsecured Design,”

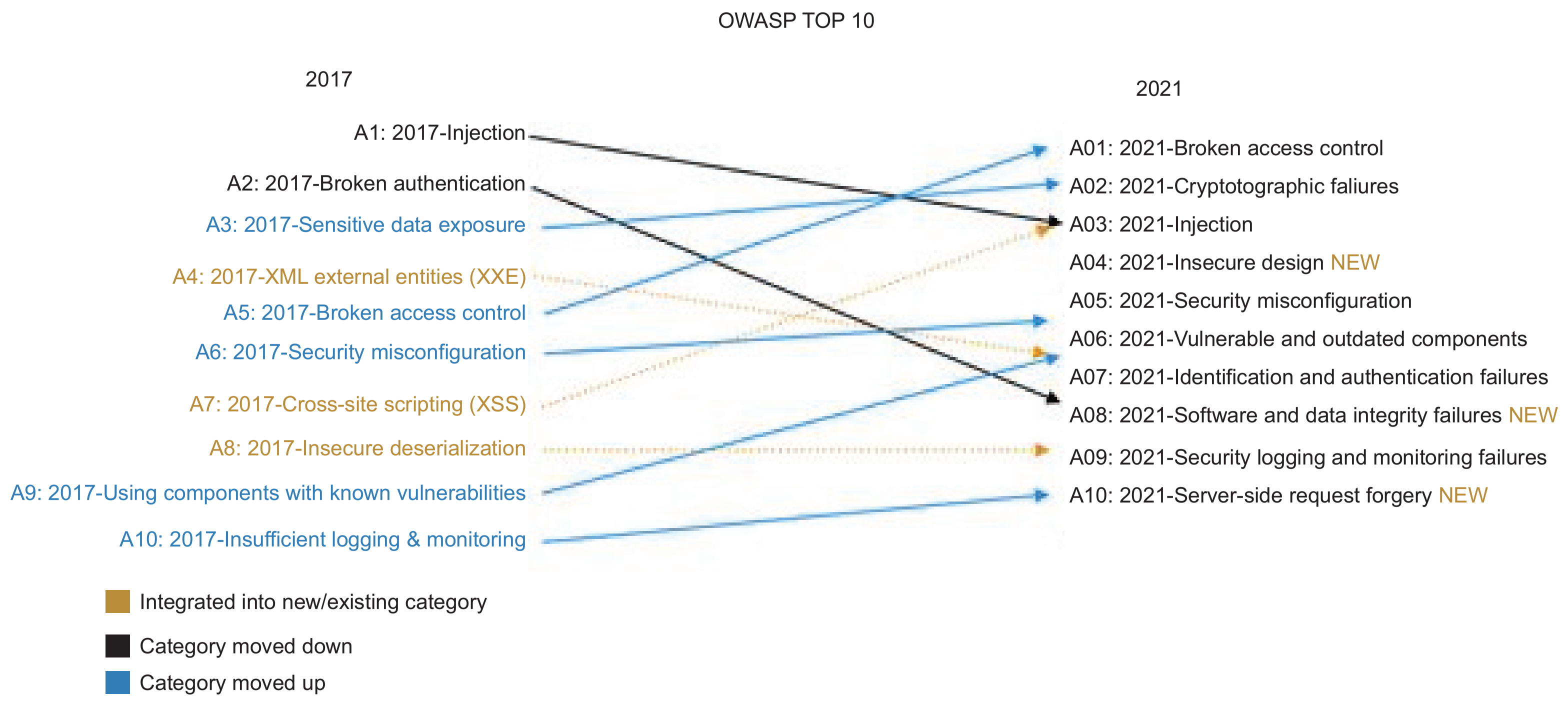
OWASP TOP 10
2017
2021
A1: 2017-Injection
A2: 2017-Broken authentication
A3: 2017-Sensitive data exposure
A4: 2017-XML external entities (XXE)
A5: 2017-Broken access control
A6: 2017-Security misconfiguration
A7: 2017-Cross-site scripting (XSS)
A8: 2017-Insecure deserialization
A9: 2017-Using components with known vulnerabilities
A10: 2017-Insufficient logging & monitoring
A01: 2021-Broken access control
A02: 2021-Cryptotographic faliures
A03: 2021-Injection
A04: 2021-Insecure design NEW
A05: 2021-Security misconfiguration
A06: 2021-Vulnerable and outdated components
A07: 2021-Identification and authentication failures
A08: 2021-Software and data integrity failures NEW
A09: 2021-Security logging and monitoring failures
A10: 2021-Server-side request forgery NEW
Integrated into new/existing category
Category moved down
Category moved up

Figure 1. Comparison of vulnerability rankings in the 2017 and 2021 editions of OWASP Top 10 reports (based on the source [8]).

emphasise the importance of addressing security issues as early as a web application's design and implementation stage. The growing prevalence of data integrity in modern applications was also highlighted.

The simple comparative analysis conducted in this chapter has confirmed the dynamics of the development of Internet applications along with new threats. Many researchers refer to this phenomenon as a dynamic threat landscape. Security strategies need to evolve at a pace similar to the development of application development tools, and employers and organisations need to adopt flexible and adaptive strategies for web application security, staying abreast of the latest trends and vulnerabilities. Such adaptability is critical to effectively countering new and emerging threats.

3. OWASP Analyser

The original OA application was inspired by illustrating specific threat scenarios and the significant vulnerabilities of web applications to selected attacks, such as injection or XSS. The application is designed to demonstrate the consequences of these attacks in specific examples – this could be data theft or unauthorised access to an account. With the use of OA, it becomes possible to demonstrate robust defence mechanisms and raise awareness among users and developers about the critical importance of adequate security measures throughout the software lifecycle.

There are already some applications on the market with similar functionality. One example is Damn Vulnerable Web Application (DVWA) [9]. DVWA is a web application designed as a tool for learning and testing various security techniques, such as SQL Injection and XSS, in a controlled environment. It is particularly useful for beginners in penetration testing and vulnerability analysis. Another example is OWASP Juice Shop [10], which is used to simulate real-world security vulnerabilities. OWASP Juice Shop is a modern web application aimed at learning and testing skills in cybersecurity. The project supports development in application security by providing scenarios aligned with the OWASP Top 10. Another example is bWAPP [11], which is particularly useful for practicing and understanding over 100 web vulnerabilities, including those outlined in the OWASP Top 10, in a safe and controlled environment, making it an excellent tool for security enthusiasts, developers, and students to enhance their knowledge of web application security. Against this background, OA stands out for its flexibility and easy adaptability to different threat landscapes, and meets the guidelines in the latest OWASP top 10 reports.

3.1. OWASP Analyser Architectural Model

Figure 2 presents a high-level diagram of the OA architectural model. This model consists of three main interconnected modules: a client-side interface module, a Python-based server, and an SQLite database.

3.1.1. Client Module

The client module was implemented using typical web technologies, including HTML, CSS, and TypeScript, combined with the angular and angular material frameworks. These tools were

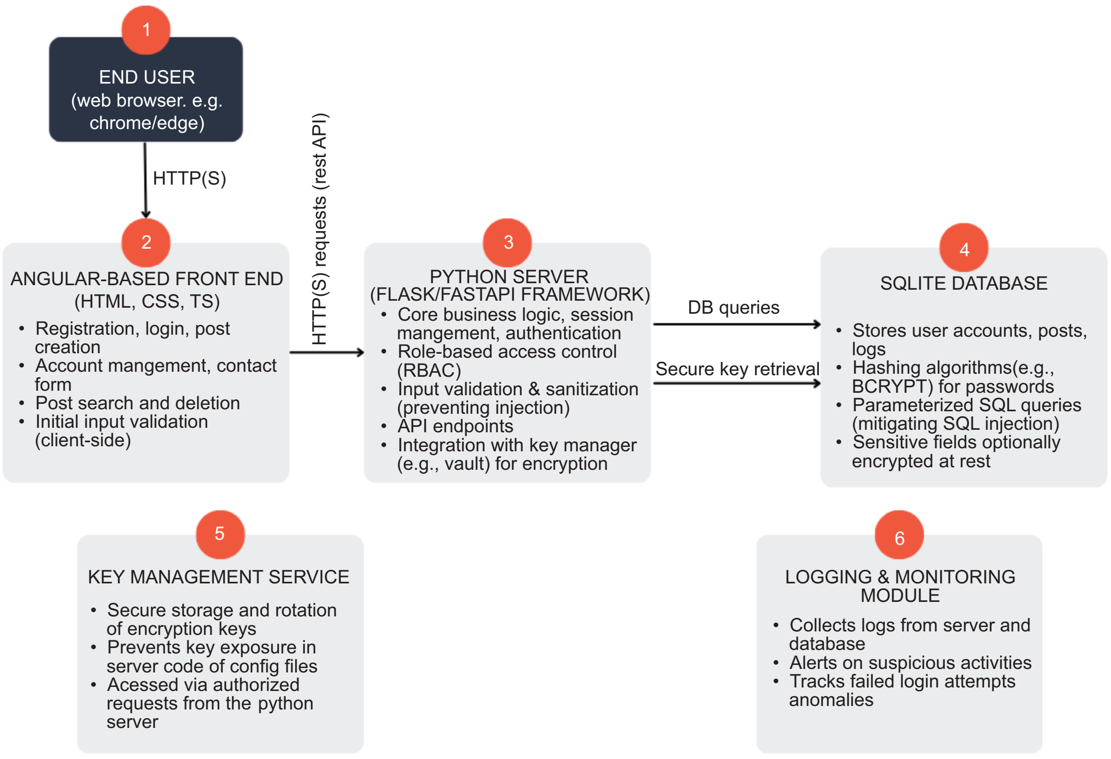
1
END USER (web browser. e.g. chrome/edge)
HTTP(S)
2
ANGULAR-BASED FRONT END (HTML, CSS, TS)
• Registration, login, post creation
• Account mangement, contact form
• Post search and deletion
• Initial input validation (client-side)
HTTP(S) requests (rest API)
3
PYTHON SERVER (FLASK/FASTAPI FRAMEWORK)
• Core business logic, session mangement, authentication
• Role-based access control (RBAC)
• Input validation & sanitization (preventing injection)
• API endpoints
• Integration with key manager (e.g., vault) for encryption
DB queries
Secure key retrieval
4
SQLITE DATABASE
• Stores user accounts, posts, logs
• Hashing algorithms(e.g., BCRYPT) for passwords
• Parameterized SQL queries (mitigating SQL injection)
• Sensitive fields optionally encrypted at rest
5
KEY MANAGEMENT SERVICE
• Secure storage and rotation of encryption keys
• Prevents key exposure in server code of config files
• Acessed via authorized requests from the python server
6
LOGGING & MONITORING MODULE
• Collects logs from server and database
• Alerts on suspicious activities
• Tracks failed login attempts anomalies

Figure 2. Architectural model of OWASP Analyzer. A high-level overview of six interconnected modules: client-side front end, Python server, SQLite database, key management, and logging & monitoring for secure operations.

used to develop a user-friendly responsive interface while maintaining flexibility for continuous improvements in security mechanisms. For example, TypeScript's strong typing system has facilitated accurate input validation and helped to minimise security vulnerabilities due to improper data handling. A concrete example is the login method, which uses strict type-checking to prevent malicious or invalid data from being processed, thus reducing the likelihood of injection attacks.

The client module contains the most commonly attacked functions, such as user registration, login, post creation, and account management. Initially, these functions were left unprotected to accurately simulate threats, such as XSS and cross-site request forgery (CSRF). This created a baseline environment, which was then used to test the effectiveness of the implemented security tools.

3.1.2. Server Module

Developed in Python, the server module handles basic tasks such as processing client requests, authenticating users, and interacting with the database. Security testing of the base OA versions revealed several security vulnerabilities, especially endpoints, without proper input validation and authorisation checks. For example, the/admin endpoint initially allowed access to unauthenticated users due to inadequate session token management.

In response, secure session management and robust backend access controls were implemented. Session tokens were dynamically generated using cryptographically secure random values and securely stored to prevent unauthorised reuse or tampering. Moreover, role-based access control restricted privileged operations, such as modifying user data, to authorised users only, reducing the likelihood of access controls being broken.

3.1.3. Database Layer

SQLite, a lightweight relational database, stores user credentials, posts, and other critical OA application data. Early iterations of the application faced significant risks, such as storing passwords in plain text. These issues were mitigated by adopting strong password hashing algorithms (such as bcrypt [12]) and ensuring that sensitive data remained encrypted during transmission and storage.

Additional safeguards included parameterised queries to protect against SQL Injection attacks. All user inputs were sanitised and processed with prepared instructions, preventing malicious actors from altering query logic or compromising database integrity.

3.1.4. Integrated Security Measures

Improvements in security mechanisms are methodically implemented and tested across all application layers. On the front end, input validation and client-side filters helped mitigate XSS attempts and invalid requests. On the server side, stricter authentication mechanisms, enforced API speed limits, and centralised logging were used to detect anomalies. The database encrypted critical fields, credentials were stored using secure hashing, and periodic audits were implemented to identify potential component vulnerabilities.

4. Experimental Analysis

The experimental analysis presented in this section shows the vulnerability of web applications according to the OWASP Top 10 2021 report. In the experiments, the designed and implemented OA application has the character of a social network. Social networks offer extensive configurations and options, which are easy to use for attackers. Nowadays, in the era of social media saturation, many people worldwide have several accounts on different sites. The application client is, therefore, simple and intuitive. OA in this implementation has the following functionalities:

1. login
2. Registering a new account with the application
3. Adding posts to the global board after completing the following information:
   - Title
   - Category
   - Location
   - Date of the event (if you create a post with an event)
   - Description
4. Deleting posts from the global board
5. Changing account settings – changing user name
6. Sending a contact form to the owner of web application
7. Searching for posts against criteria:
   - Title
   - Category
   - Location

- Start date
- End date

On the server module, addresses have been prepared waiting to send the appropriate request from the client module, which is checked and redirected to the database to perform the following operations:

1. Checking whether the logged-in user exists in the database, and whether his data is correct to allow logging into the application
2. Adding a new user to the database
3. Adding posts
4. Deleting posts
5. Editing posts
6. Log out
7. Checking authorisation
8. Deletion of authentication token data follows the logout process of the user; this provides additional security against unauthorised access
9. Checking whether a given user has authorisation to perform particular actions
10. Additional validation of data sent by the client application, should an attack be attempted
11. Encryption of data so that it does not leak during attacks
12. Providing information on the operation performed, whether it was successful, and what errors were intercepted
13. Sending a contact message to the application owner
14. Editing data regarding a particular account, error handling, for example, what if incorrect data is given,
15. Unblocking CORS for the particular address to which requests are made and those from which they are received,

The tests were conducted in the following three scenarios of OA configuration:

1. Basic configuration (without security measures)
   In the initial phase, the application was tested without any security mechanisms. This identifies the main problems and vulnerabilities of the application related to lack of input validation, unsecured password storage, and unauthorised access to administrative resources. The test results in this scenario served as a baseline for subsequent testing phases.
2. Configuration with basic security measures
   The second phase introduced minimal security measures, such as input validation, access restrictions to the administration

panel, and parameterised queries. These changes aimed to mitigate common threats, including SQL Injection and XSS attacks.

3. Configuration with advanced security
   Comprehensive security mechanisms were implemented in the final phase of experimental analysis. These included role-based access control (RBAC), encryption of sensitive data, and automated session management. In addition, monitoring and logging mechanisms were integrated to reduce response time to potential attacks.

4.1. Results

The tests identified key weaknesses in application security and evaluated the effectiveness of various protection methods. The results of the experiments were interpreted in terms of the following five criteria in line with the OWASP Top 10 2021 report: flaws in access control mechanisms, vulnerability to SQL Injection attacks, vulnerability to XSS attacks, vulnerability to cryptographic errors and behaviour when logging, and monitoring tools are introduced.

4.1.1. Unauthorised Access Control

Initial tests showed that an unauthenticated user could access the administration panel through URL manipulation, for example, by appending/admin to the page address. This allowed unauthorised users to view sensitive data and potentially make system-level changes. For the basic configuration, all 30 attempts to access the admin panel without valid credentials succeeded, underscoring the severity of the vulnerability.

After implementing session management and enforcing server-side validation through the validation of authorisation tokens and user role verification, the vulnerability was successfully neutralised. In repeated tests in the new configuration, none of the 30 unauthorised access attempts failed, translating into a 100% success rate in blocking unauthorised logins.

Figure 3 shows the results of this experiment. It clearly shows the importance of combining session token validation with robust backend controls. Even if attackers try to manipulate URLs or inject forged session tokens, the server's RBAC mechanisms verify the authenticity of each request and privilege level. As a result, only legitimate and authenticated users with appropriate privileges are granted access to the/admin route, reducing the risk of data breaches or system misuse.

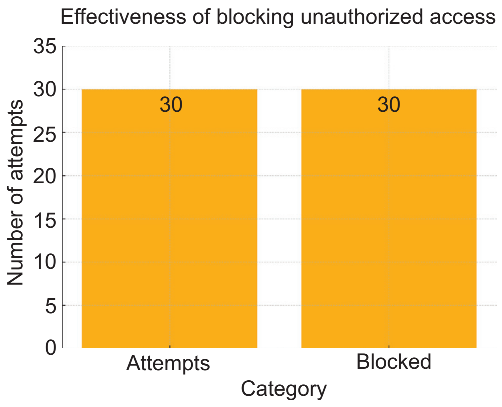

Figure 3. Effectiveness of blocking unauthorised access. Results demonstrate the significance of robust backend controls and session token validation in preventing unauthorised attempts.

4.1.2. SQL Injection

In the initial phase of the application's vulnerability to SQL Injection attacks, crafted queries (e.g. 'OR '1'='1) were injected, allowing attackers to retrieve sensitive data and unauthorised access to critical information without valid credentials. To address this problem, parameterised queries and additional input validation mechanisms were introduced, effectively sanitising user input before passing it to the database. As a result, all 25 SQL Injection attempts conducted in the follow-up tests were successfully blocked. In addition, logging functions were improved to track suspicious queries, thus enabling faster incident response and anomaly detection. The results of these experiments are presented in Fig. 4.

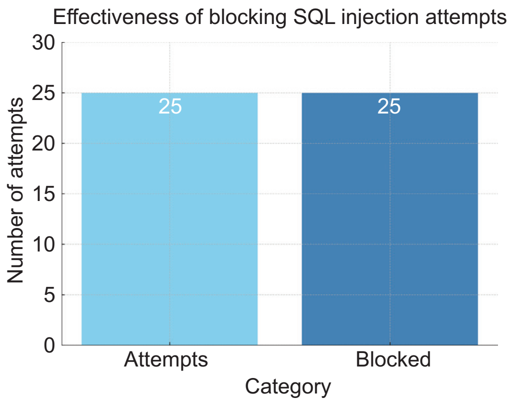

Figure 4. Effectiveness of blocking SQL Injection attempts. It highlights the reliability of security mechanisms in detecting and preventing all SQL Injection attempts.

4.1.3. Cross-Site Scripting

Cross-site scripting attacks primarily targeted a post-creation form, in which malicious scripts were injected to activate user accounts in browsers without their knowledge. In a basic configuration, these scripts are executed without any restrictions, posing a serious threat to data confidentiality and integrity.

Once input sanitisation and content escaping techniques were implemented, each of the 25 recorded XSS attack attempts was successfully neutralised (see Fig. 5). The security measures demonstrate that user-generated content is properly filtered and rendered as a plain text, rather than processed as executable code. Defence mechanisms are activated on the client side (to provide immediate feedback and prevent basic exploits) and on the server side (to validate and sanitise incoming data against more advanced payloads).

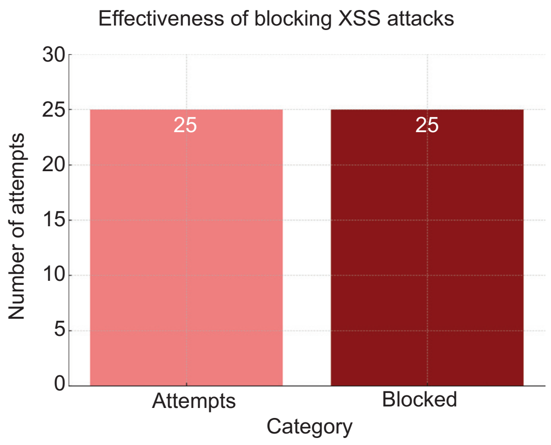

Figure 5. Effectiveness of blocking XSS attacks. It demonstrates the robustness of implemented security measures in successfully preventing all XSS attack attempts.

4.1.4. Cryptographic failures

Initially, the application stored passwords in plain text, which posed a serious security risk. The plain text credentials could be immediately exploited if an attacker gains access to the database. To address this vulnerability, the bcrypt hash algorithm was integrated, introducing a computational cost that makes brute-force attempts much more difficult.

In subsequent tests, brute-forcing passwords protected by bcrypt took more than 10 hours of continuous computation on standard hardware when configured with 10 rounds of hashing, as shown in Fig. 6. This marked improvement illustrates the effectiveness of a robust hash function in protecting sensitive data. Salting and appropriate cryptographic parameters further reduced the likelihood of successful password cracking, ensuring that user credentials remain secure even during a partial database breach. The application strengthened its overall security status by adopting standard cryptographic practices and protected users from unauthorised account access.

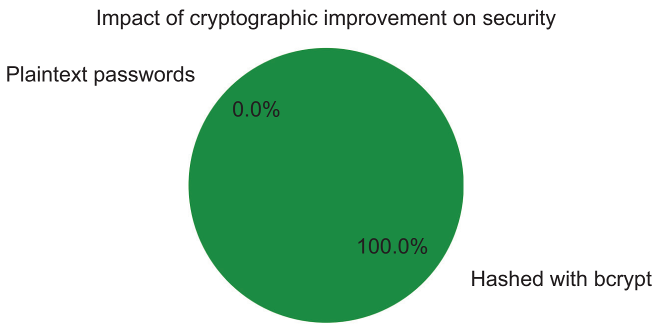

Figure 6. Impact of cryptographic improvements on security. This highlights complete transition from plain text passwords to secure hashing with bcrypt, ensuring enhanced data protection.

4.1.5. Login and Monitoring

The application failed to log login attempts or suspicious activity without activating logging and monitoring mechanisms, making detecting attacks or investigating incidents difficult. In the final configuration, event logging and real-time monitoring were implemented to capture critical security events, such as failed login attempts, SQL Injection attempts, and unusual user behaviour.

In most scenarios, these measures reduced response times from several hours (up to 240 minutes) to less than 15 minutes, enabling rapid intervention to stop threats. The results of the experiments are shown in Fig. 7, indicating the importance of continuous monitoring in implementing modern security strategies. By proactively analysing logs, setting up automatic alerts, and reviewing anomaly reports, organisations can respond quickly to potential breaches, thereby minimising damage and preserving the integrity of user data.

4.2. Summary of the Experiments

Security tests have demonstrated the key role of proactive measures in mitigating vulnerabilities commonly found in web

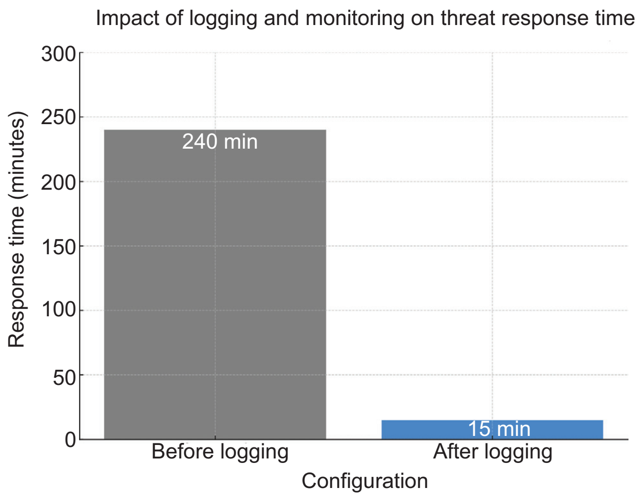

Figure 7. Impact of logging and monitoring on threat response time. It demonstrates a significant reduction in response time from 240 minutes to 15 minutes after implementing logging and monitoring mechanisms.

applications. Even basic security measures, such as input validation, session management, and parameterised queries, can significantly reduce the exposure of web services to threats, such as SQL Injection, XSS, or unauthorised access to data and the application itself.

The implemented advanced security mechanisms justified the concept of a modular architectural model for modern web applications. Role-based access control and robust session management have effectively neutralised access control security vulnerabilities in modular architectures. Implementing advanced cryptographic methods, such as the crypt hash method, has ensured sensitive information's integrity and data confidentiality. Logging and monitoring systems enabled rapid detection and response to suspicious activity, minimising potential damage from brute force and injection attacks.

The experimental analysis conducted led to the following guidelines for web application security:

- *Secure Application Life Cycle* (SDLC): Security mechanisms must be integrated at every implementation and application life cycle stage, from design to deployment. Identifying potential threats through threat modelling and secure coding practices can prevent vulnerabilities from becoming web services.

- *Data validation and sanitisation*: These are fundamental to preventing injection attacks and ensuring data integrity. To ensure maximum effectiveness, input data validation should be enforced on both client and server sides.
- *Cryptographic standards*: Storing sensitive data, such as passwords, in plain text is a serious risk. Standard hash algorithms, such as bcrypt, and encrypting sensitive fields are essential to protect user information.
- *Access control policies*: Implementing detailed access control, including RBAC, allows users to access only the resources required for their roles. Backend validation should complement client-side controls to prevent circumvention.
- *Regular updates and dependency management*: Outdated software components can introduce vulnerabilities that can be exploited. Automated tools, such as dependency scanners, should be used to identify and regularly update unprotected libraries.
- *Comprehensive logging and monitoring*: Effective logging practices enable early detection of malicious activity. Monitoring tools should include real-time alerts for critical events, such as repeated logging failures.
- *Comprehensive logging and monitoring*: Effective logging practices enable early detection of malicious activity. Monitoring tools should include real-time alerts for critical events, such as repeated login failures or injection attempts, enabling rapid intervention.

5. Challenges and Future Trends

Despite marked improvements in vulnerability mitigation, especially in layered and modular architectures, organisations must constantly adapt to the rapidly evolving threat landscape. As technology advances, new types (vectors) of attacks and methodologies are emerging, requiring constant vigilance and innovation. Based on the theoretical and experiential analysis of the threats and vulnerabilities of modern web applications conducted in this work, the list of trends and key issues shaping the dynamics of broad changes in web application security, presented later in this section, has been defined.

5.1. Evolving Cyber Threats

Cybercriminals are rapidly refining their techniques, leveraging automation, social engineering, and AI-driven strategies to launch more sophisticated attacks. Traditional defences, such as basic firewalls or signature-based intrusion detection, may be insufficient against complex threats that adapt in real-time. This calls for

Table 1. Challenges and trends in web application security.

Challenge/trend	Description
Evolving Cyber Threats	Sophisticated attacks leveraging automation, AI, and real-time adaptation; requires anomaly detection and automated responses.
Integration with DevSecOps	Embedding security in CI/CD pipelines to identify vulnerabilities early and improve resilience.
Microservices and Containerisation	Granular controls and container isolation are needed to secure dependencies and configurations in distributed environments.
Zero-Trust Architecture	Continuous verification and dynamic access policies to minimise trust and lateral movement risks.
AI and Machine Learning (ML) for Defence	Using advanced machine learning for threat detection and response while addressing adversarial risks.
Regulatory Compliance and Data Privacy	Compliance with data privacy regulations to avoid fines and reputational damage.
Continuous Security Education	Training stakeholders to recognise threats and apply secure coding practices effectively.

advanced solutions that detect subtle anomalies, integrate threat intelligence, and automatically orchestrate responses to contain breaches before they escalate.

5.2. Integration with DevSecOps

The transition from traditional software development lifecycles (SDLC) to more agile and continuous delivery models has highlighted the need for DevSecOps [13], embedding security at every stage of development. Organisations can identify and remediate vulnerabilities earlier by automating security scans, code reviews, and penetration tests as part of the CI/CD pipeline. This approach reduces the likelihood of security issues persisting into production while ensuring faster release cycles and more resilient applications.

5.3. Microservices and Containerisation

Modern applications often adopt microservices architecture and containerisation (e.g. Docker, Kubernetes) for scalability and maintainability. However, each microservice and container introduces its own dependencies, configurations, and potential vulnerabilities. Securing these distributed environments requires granular access controls, robust container isolation, and regular updates of container images to prevent exploited or outdated components from compromising the entire system.

5.4. Zero-trust Architecture

A zero-trust model maintains that no user, device, or network segment is inherently trusted. Instead, continuous verification (e.g. using multi-factor authentication, dynamic access policies, and strict segmentation) becomes the standard. As remote work and cloud-based services expand, zero-trust frameworks help ensure that each request is rigorously validated, reducing the attack surface and limiting the lateral movement of adversaries once inside a network.

5.5. Artificial Intelligence and Machine Learning for Defence

While attackers leverage AI and machine learning to automate intrusion efforts, defenders can similarly employ these technologies for anomaly detection, threat intelligence, and real-time correlation of events. Advanced machine learning models can help differentiate legitimate user behaviour from malicious activities, significantly improving incident response. However, the risk of adversarial attacks (where attackers poison or manipulate machine learning models) remains an ongoing challenge that security teams must address.

5.6. Regulatory Compliance and Data Privacy

Growing awareness of data breaches and privacy violations has led to more stringent regulations, such as the General Data Protection Regulation (GDPR) [14] in the European Union and similar laws worldwide. Compliance requirements push organisations to adopt stricter security controls, encrypt sensitive data, and maintain detailed logs. Meeting these standards can be complex, but failure to do so exposes organisations to substantial fines and reputational damage.

5.7. Continuous Security Education

Human factors often represent the weakest link in the security chain. Social engineering, phishing, and credential theft rely on user error or lack of awareness. Regular training and awareness programs are vital for developers and end-users, helping them recognise threats, follow secure coding practices, and respond appropriately to security incidents. Ongoing education ensures that stakeholders can effectively navigate emerging threats and vulnerabilities.

6. Conclusions

This paper highlights the new trends and developments in the security of web applications. A list of the most important vulnerabilities of these applications is published once every 3 years as the OWASP Top 10 report. This report includes a ranking of vulnerabilities indicating the most up-to-date at a given time and the most dangerous threats to users of web applications at all levels of their use. Section 2 presented a simple comparative analysis of the last two editions of the OWASP reports. This analysis showed how the threat landscape has changed in just 3 years and the tremendous need for flexible and responsive tools to prevent attacks and eliminate detected web application vulnerabilities.

Experimental results underscore the need to embed security throughout Web applications' implementation and use cycle. While advanced measures, such as encryption and real-time monitoring, increase resilience to sophisticated attacks, even basic practices can provide significant application protection if applied consistently.

Implementing automated security testing, coupled with ongoing education of developers and users on best practices, is essential to reduce the risk of losing data or sensitive information published online. In addition, regular audits and updates are the cornerstone of maintaining secure systems in an evolving threat landscape.

The future of web application security hinges on proactive integrated approaches that blend automation, zero-trust principles, and continuous monitoring. As organisations continue to embrace cloud computing, containerisation, and microservices architectures, DevSecOps practices have become indispensable, ensuring security measures are embedded at every development and deployment phase. By staying informed about the latest trends and adapting defences accordingly, stakeholders can better protect critical data and systems against the expanding spectrum of cyber threats.

References

[1] M. Al-Ibrahim, Y.S. Al-Deen, "The reality of applying security in web applications in academia," *International Journal of Advanced Computer Sciences and Applications*, vol. 5, no. 10, pp. 7–16, 2014. doi: 10.14569/IJACSA.2014.051002.

[2] B. Thuraisingham, C. Clifton, A. Gupta, E. Bertino, E. Ferrari (2002). Directions for web and e-commerce applications security [Online]. Available: http://dx.doi.org/10.2139/ssrn.333682 [Accessed: Dec 20, 2024].

[3] F. Salahdine, Z. El Mrabet, N. Kaabouch, "Phishing attacks detection a machine learning-based approach," *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference* (UEMCON), New York, NY, 2021, pp. 0250–0255.

[4] Gartner, "What is Cybersecurity? Trends, Strategies, and Insights." [Online]. Available: https://www.gartner.com/en/topics/cybersecurity [Accessed: Dec. 20, 2024].

[5] OWASP Foundation, "OWASP Foundation, the Open Source Foundation for Application Security." [Online]. Available: https://owasp.org/ [Accessed: Dec. 20, 2024].

[6] O.B. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, A. Derhab, "An OWASP top ten driven survey on web application protection methods," in *Risks and security of internet and systems*, Lecture Notes in Computer Science, J. Garcia-Alfaro, J. Leneutre, N. Cuppens, R. Yaich, Eds. Cham: Springer, 2021, pp. 235–252.

[7] PortSwigger Ltd., "Burp Suite," [Online]. Available: https://portswigger.net/burp [Accessed: Dec. 20, 2024].

[8] OWASP Foundation, "OWASP Top Ten," [Online]. Available: https://owasp.org/www-project-top-ten [Accessed: Dec. 20, 2024].

[9] Ryan Dewhurst, "Damn Vulnerable Web Application (DVWA)," [Online]. Available: https://github.com/digininja/DVWA [Accessed: Dec. 20, 2024].

[10] OWASP Foundation, "OWASP Juice Shop," [Online]. Available: https://owasp.org/www-project-juice-shop/ [Accessed: Dec. 20, 2024].

[11] Malik Mesellem, "bWAPP: a buggy web application," [Online]. Available: https://www.itsecgames.com/ [Accessed: Dec. 20, 2024].

[12] Wikipedia, "bcrypt," [Online]. Available: https://en.wikipedia.org/wiki/Bcrypt [Accessed: Dec. 20, 2024].

[13] IBM Corporation, "What is DevSecOps?" [Online]. Available: https://www.ibm.com/think/topics/devsecops [Accessed: Dec. 20, 2024].

[14] European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council," [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng [Accessed: Dec. 20, 2024].