

Security Analysis of Selected Web Applications Using Vulnerability Scanners

Mariusz Choroś*, Marta Dziuba-Kozieł

Department of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract

Web applications are among the most frequently targeted systems in today's cyber-threat landscape, and vulnerabilities such as SQL injection, cross-site scripting and various configuration errors have dominated the OWASP Top 10 list for years. This article examines the security of a web application using modern vulnerability scanning tools and penetration testing. The study was conducted in two stages: in the first stage, the deliberately vulnerable OWASP Juice Shop application was used, with three scanners: OWASP ZAP, Burp Suite Community Edition and Nuclei applied to it. In the second stage, the tools were used to assess the security of a proprietary web application. All tests were conducted in a local, controlled environment and utilised Docker containerisation for the deliberately vulnerable application. The results of the proprietary application assessment revealed no logical vulnerabilities from the OWASP Top 10 category. The tools did not confirm the presence of SQL Injection, XSS or Broken Access Control. The identified issues related solely to the configuration of HTTP security headers. The study confirms that none of the scanners used can detect the full spectrum of vulnerabilities on their own, and that a combination of automated, semi-automated and template-based scanning provides the most comprehensive assessment of web application security.

Keywords: web application security; vulnerability scanner; OWASP ZAP; Burp Suite; Nuclei; penetration testing

*Corresponding author

Email address: s95376@pollub.edu.pl (M. Choroś)

Published under Creative Common License (CC BY 4.0 Int.)

1. Introduction

Web application security is one of the key challenges facing modern IT, as evidenced by the fact that attacks on web applications widespread availability and the processing of sensitive account for nearly 40% of all data breaches, and the number of reported security incidents is constantly rising [1]. Due to their user data, web applications remain the primary target of attackers, who exploit well-known classes of vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Broken Access Control and Security Misconfiguration, as defined in the OWASP Top 10 [2].

This article discusses the issue of web application security testing using DAST (Dynamic Application Security Testing) tools with different scanning methodologies. The study was conducted in two stages. In the first stage, three tools with different scanning methodologies: OWASP ZAP, Burp Suite Community Edition and Nuclei were applied to the deliberately vulnerable OWASP Juice Shop application, an environment designed to test the effectiveness and performance of security testing tools. In the second stage, all three tools were used to assess the security of a proprietary web application, utilising the distinct methodological approaches of each. The results obtained were compared with statistics on the prevalence of vulnerabilities described in the existing scientific literature, enabling a comprehensive assessment of the security level of the tested application.

DAST (Dynamic Application Security Testing) tools represent one of the fundamental approaches to the automated security verification of live applications. Vulnerability scanners detect vulnerabilities by dynam-

cally testing applications from the outside, without access to their source code. Each of these tools uses a different scanning methodology, which results in differences in the effectiveness of vulnerability detection.

2. Related work

The topic of application security testing has become quite popular in the scientific literature, a trend driven by the rapidly evolving landscape of cyber threats and the development of tools and solutions specifically designed for the cybersecurity industry.

Article [3] presents a systematic literature review covering ninety scientific papers and identifying thirty different WVS (Web Application Vulnerability Scanners) tools. It was shown that only twelve of these had any quantitative assessment of effectiveness, and the research focused almost exclusively on two types of vulnerabilities from the OWASP Top 10 list: SQL Injection and XSS. The authors explicitly identified this narrow scope of testing as a key limitation of the current state of research.

Another important point regarding research was presented in [4]. Twelve tools were compared in the context of assessing web applications for vulnerabilities, highlighting the problem of false positives, which requires significant involvement on the part of the tester and constitutes one of the limitations of automated tools. A conclusion widely accepted in application testing is that no single scanner is capable of detecting the full spectrum of vulnerabilities and should therefore be used in combination with other scanners and manual verification of results.

During the analysis, a study was identified that examined a slightly different segment of applications but employed a similar methodology [5]. The analysis focused on popular CMS platforms and utilised automated scanners to identify vulnerabilities and flaws requiring a more in-depth analysis of system behaviour and business logic.

Another comprehensive analysis of a real-world web application was published by a Thai university [6] and provides a wealth of quantitative and qualitative data for research purposes. It was noted that both tools detect partially different vulnerabilities and classify them differently within the OWASP Top 10, which confirms the need to use them in combination, as in the case of study [3]. A limitation of this study, however, is the lack of a third tool employing a different scanning method.

Similar conclusions emerge from a comparative analysis of tools in [7], which demonstrated clear differences in the number of vulnerabilities detected between tools. However, this study was limited exclusively to test environments based on deliberately vulnerable laboratory applications, and did not include applications written under real production conditions.

According to the Edgescan report [8], which analyses real-world web applications, over 33% of the vulnerabilities detected were classified as critical or high. SQL injection has remained the most common critical vulnerability in web applications since 2022, accounting for 19.47% of all high- and critical-severity vulnerabilities. These figures serve as a benchmark for assessing the security level of a proprietary application against real-world production applications.

All the studies analysed point to the necessity of using a combination of different scanners, despite research into various types of web applications with differing levels of security, as well as variations in the selection of scanning tools.

The literature review provided an insight into the subject under investigation and helped to understand how to conduct an analysis in accordance with best practice, as well as the key aspects of the process of researching web application vulnerabilities. Conclusions were drawn which shaped the approach to conducting the research through the use of three scanners and the analysis of two different applications.

3. Materials and methods

3.1. Tested applications

The research was conducted using a proprietary web application developed for the purposes of an engineering thesis entitled „The System Supporting the Organization of Skydiving”. The server-side of the application was built using the Xampp package, a MySQL database, the Express application framework and the Node.js runtime environment. The user interface and the graphical part of the application were created using the React framework.

The application includes the basic features typical of web-based systems: login, registration and data entry forms, an administration panel, and authentication and authorisation mechanisms. It also includes the modules and features necessary for use by customers, employees and administrators.

The security analysis of this application serves as an example of testing software developed under real-world conditions, without any particular focus on security and without the introduction of any deliberate vulnerabilities. Its vulnerabilities are not known in advance, which makes it a highly reliable subject for testing.

The second subject of the research is the OWASP Juice Shop - a deliberately vulnerable application developed by the OWASP organisation, which serves as a learning environment for ethical hacking and security tool testing. The application simulates an online shop containing a multitude of vulnerabilities corresponding to the OWASP Top 10 categories, including SQL Injection, Cross-Site Scripting, Broken Access Control and many others. Some of its functionality is hidden from testers and requires deeper exploration and testing of its capabilities. It has been specifically chosen as a control application to verify the capabilities of the tools used and to investigate what types of vulnerabilities it possesses in relation to the proprietary application. The OWASP Juice Shop application is available as a Docker image on Docker Hub at [bkimminich/juice-shop](https://hub.docker.com/r/bkimminich/juice-shop) and in the project's official repository on GitHub [9].

3.2 Tools and configuration

The selection of the scanners listed below was not random but was based directly on a review of the literature. Each tool represents a slightly different approach to application testing, and their combination was intended to ensure a comprehensive assessment from various angles.

OWASP ZAP

OWASP Zed Attack Proxy, also known as OWASP ZAP for short, is one of the most popular vulnerability scanners used during penetration testing of web applications. It is an open-source tool available on Windows, Linux and macOS. The scanner's core functionalities include, amongst others, a local proxy for intercepting and modifying HTTP requests, a website exploration tool, and an automated vulnerability scanner. In the research conducted, ZAP was run in active scanning mode, performing a full audit of both applications. The tool was selected due to its widespread use, open licence and proven effectiveness. It enables the automatic detection of various types of vulnerabilities, making it one of the key tools in application security testing [10].

Burp Suite Community Edition

Burp Suite is one of the leading tools for penetration testing of web applications, developed by PortSwigger. The research utilised the free Community Edition, which does not include a built-in automated scanner. The tool was used as an HTTP proxy, enabling the interception and modification of communication between the browser and the server. Semi-automated tests were carried out using the Repeater module, which allows for the manual preparation and repetition of HTTP requests with modified parameters set according to the application endpoint under test. The tool was selected to complement automated scanners with the capability for precise manual analysis, which is also the scanner's greatest limitation. The Community Edition does not include the automatic scanning option available in the Professional version.

Nuclei

Nuclei is a modern scanner based on YAML templates, which is run from the operating system's command line. It utilises a database of over ten thousand templates covering vulnerabilities from the OWASP Top 10 and the CVE database. Unlike ZAP and Burp Suite, it operates using templates, which allows it to target specific classes of vulnerabilities. Its limitation is that it detects only vulnerabilities included in the available templates, and unique or non-standard vulnerabilities may be overlooked. The tool was selected due to its modern approach to analysis and its distinct operational architecture compared to the other scanners used.

3.3 Experimental Setup

All tests were carried out on a local workstation running the Windows operating system. The OWASP Juice Shop application was run in a Docker container, whilst the proprietary application ran on a local development server using the XAMPP package. The test subjects were launched in separate test sessions.

The study was conducted in two stages. In the first stage, all three tools were applied to the OWASP Juice Shop application, an environment designed to verify the effectiveness of security tools in the presence of real-world vulnerability classes. OWASP ZAP stood out for its effectiveness and automatic detection of configuration issues, Burp Suite CE enabled the confirmation of logical vulnerabilities through semi-automated testing, and Nuclei effectively identified known CVE patterns based on templates. In the second stage, all three tools were used to assess the security of a proprietary web application, utilising the distinct methodological approaches of each.

The scanners were launched using the default settings, with the exception of Nuclei, for which a specific range of templates was specified for the application under test. The scanners were installed locally on the hard drive and launched as executable applications, with

the exception of Nuclei, which could only be used via the Windows Command Prompt.

In order to ensure comparable testing conditions for both applications, all external factors that could interfere with the results were eliminated: Windows Defender and the system firewall were disabled (it was found that their activation prevented the Nuclei tool from functioning correctly).

The automated scan was performed using the 'Attack' function in OWASP ZAP, which automatically indexes application endpoints and runs tests. Burp Suite CE was configured to operate as a proxy and was used exclusively for manual testing on selected entry points. Nuclei, on the other hand, was run with the default set of templates covering the OWASP Top 10 categories and the CVE database, and the results were saved in text files. The reliability of the results was ensured by comparing the results obtained for the author's application with those for the Juice Shop application.

Network traffic during the tests varied depending on the tool used. In the case of OWASP ZAP and Burp Suite CE, the browser was configured to route all HTTP traffic through a local proxy listening on port 8080. The proxy acted as an intermediary for every request sent by the browser to the application, enabling the interception and modification of HTTP communication in real time. In active scanning mode, OWASP ZAP automatically generated and sent requests to the application based on automatically indexed resources. Burp Suite CE, using the Repeater module, allowed the tester to manually craft individual requests and observe the server's responses, enabling analysis of the endpoints they had selected. Nuclei, on the other hand, sent requests directly to the application according to patterns from templates, bypassing the proxy layer. A common feature of all the tools was that they operated without access to the database layer or the application's source code.

3.4 Evaluation Criteria

The security assessment of the tested applications is based on the following criteria:

- The number of alerts detected by each tool
- Types of vulnerabilities detected, classified by category
- Inclusion in the OWASP Top 10
- Classification of detected vulnerabilities by severity level (high, medium, low, informational)
- Ability to manually verify selected vulnerabilities
- Comparison of results between tools

4. Results

4.1. Results of the audit of the OWASP Juice Shop

4.1.1. OWASP ZAP scan

Scanning the OWASP Juice Shop application using the OWASP ZAP tool recorded a total of 431 vulnerability alerts. The vulnerabilities detected spanned all levels of

the application architecture, from the code itself, through configuration, to the management of external dependencies. The most frequently detected alerts were: Timestamp Disclosure – Unix, Cross-Domain JavaScript Source File Inclusion, Cross-Domain Misconfiguration, Content Security Policy (CSP) Header Not Set, and Modern Web Application. Table 1 presents a summary of the number of vulnerabilities detected by severity level.

Table 1: Summary of the number of vulnerabilities detected in the OWASP Juice Shop application, by severity level

Risk level	Number of alerts	Percentage (percentage of a given type of vulnerability)
High	0	0%
Medium	129	29,9%
Low	251	58,2%
Informational	51	11,8%
Total	431	100%

The most commonly detected types of alerts are:

- Timestamp Disclosure – 155 occurrences
- Cross-Domain JavaScript Source File Inclusion – 96 occurrences
- Cross-Domain Misconfiguration – 72 occurrences
- Content Security Policy (CSP) Header Not Set – 57 occurrences
- Modern Web Application (information alert) – 49 occurrences

4.1.2. Burp Suite Community Edition scan

Burp Suite CE was used to perform semi-automated testing of the Juice Shop application. Using the Repeater module, SQL injection vulnerabilities were identified in the login forms, resulting in access to the administration panel without knowledge of the actual credentials. A vulnerability to Reflected XSS was also confirmed by injecting a payload into the search fields. The tests further demonstrated that the combination of SQL injection and the subsequent exploitation of administrator privileges allows for the modification of other users' data, and such a chain of vulnerabilities poses a critical threat to data integrity.

4.1.3. Nuclei scan

A scan of Juice Shop using the Nuclei tool with OWASP Top 10 templates detected vulnerabilities related to application configuration and exposed re-

sources: publicly accessible API documentation and a lack of integrity verification for resources loaded from the CDN. The scanner successfully identified security headers and known CVE vulnerability patterns contained in the template database. The results supplemented the picture of the application's security with vulnerabilities not detected by ZAP.

4.2. Results of the audit of the proprietary application

All three security assessment tools were used to scan the in-house application, in line with their established specialisations: OWASP ZAP for automated scanning, Burp Suite CE for verifying application logic through semi-automated testing, and Nuclei for detecting known CVE vulnerability patterns.

4.2.1. OWASP ZAP scan

Scanning the in-house application using the OWASP ZAP tool recorded a total of 9 alerts. All medium-severity alerts concerned security configurations on the browser and HTTP layer: deficiencies in Content Security Policy (CSP) headers, lack of protection against clickjacking, and incorrect configuration of CORS (Cross-Origin Resource Sharing). Low-severity and informational alerts indicated potential disclosure of information regarding the technologies used and deficiencies in security headers. No high-severity alerts were recorded. However, the absence of serious vulnerabilities such as SQL injection or XSS indicates that the main entry points of the applications are properly secured. Table 2 presents a summary of the number of detected vulnerabilities by threat level.

Table 2: Summary of the number of vulnerabilities detected in proprietary applications by threat level

Risk level	Number of alerts	Percentage (percentage of a given level of vulnerability)
High	0	0%
Medium	4	44,4%
Low	3	33,3%
Informational	2	22,2%
Total	9	100%

4.2.2. Burp Suite Community Edition scan

Burp Suite CE was used in semi-automated mode as an HTTP proxy. The testing involved intercepting requests sent by the browser to the server and modifying them using the Repeater module. The main entry points of the application were tested: the login form, the registration form and the data entry forms. In all cases, attempts to inject SQL commands failed, and the application returned validation errors without revealing the structure of the database queries. The tests confirmed the absence of direct logical vulnerabilities such as SQL injection or XSS at the application's main entry points.

4.2.3. Nuclei scan

Scanning the proprietary application using the Nuclei tool with OWASP Top 10 templates did not reveal any vulnerabilities. The absence of detected vulnerabilities confirms that the in-house application meets basic security standards regarding the most well-known and common classes of vulnerabilities, as defined by the templates used for the scan.

This is due to the nature of the application under test, which does not resemble typical products and software that usually contain some of the most common vulnerabilities. However, this lack of alignment with standard patterns does not imply the absence of any security vulnerabilities, which necessitates the use of a different type of scanner.

4.2.4. Summary results of tests on the proprietary application

A comparison of the testing process and the results of the three tools shows that each emphasises a slightly different aspect of security: OWASP ZAP focuses mainly on header configuration issues, Burp Suite CE did not detect any errors in the application's logic, and Nuclei did not find any known vulnerabilities from its template database. Scanning of the proprietary application did not register any high-risk alerts and did not reveal any logical vulnerabilities known as the OWASP Top 10. All identified issues relate solely to configuration and can be rectified by correctly setting HTTP security headers and CORS policies.

4.3. Comparison of the results from the tools for both applications

A comparison of the scan results from both applications allows us to assess the scope and specialisation of the tools used in relation to different types of vulnerabilities. The OWASP Juice Shop application, as a deliberately vulnerable environment, enabled us to verify the detection capabilities of each scanner in conditions where real security vulnerabilities were present. The proprietary application, on the other hand, was used to assess the behaviour of the tools in an environment similar to a production environment. Table 3 presents

a summary of the types of vulnerabilities detected, depending on the tool used.

Table 3: A comparison of the tools used in relations to the application under investigation

Scanning tool	Vulnerability type	Juice Shop	Proprietary application
OWASP ZAP	HTTP layer configuration	Detected	Detected
Burp Suite CE	Sql Injection	Detected	Not detected
Burp Suite CE	XSS	Detected	Not detected
Burp Suite CE	Attack chains / application logic	Detected	Not detected
Nuclei	CVE and configuration vulnerabilities	Detected	Not detected

An analysis of the scan results showed that none of the tools used detected all vulnerabilities on their own. OWASP ZAP was the most effective at automatically detecting a wide range of alerts, Burp Suite CE enabled the confirmation of critical logical vulnerabilities and attack chains through testing of specific endpoints, and Nuclei supplemented the results with vulnerabilities based on CVE templates. The complementary nature of the results from the individual tools confirms the need to use a combination of automated scanners and manual techniques in a comprehensive security assessment.

Furthermore, a comparison of the scan results for both applications reveals a clear difference in both the number and severity of the vulnerabilities detected, reflecting a fundamental difference in the intended purpose and security characteristics of the two systems. The in-house application was developed in accordance with the basic principles of secure software development, where OWASP Juice Shop was deliberately designed to replicate a multitude of real-world vulnerabilities corresponding to all categories of the OWASP Top 10.

5. Discussion

The studies conducted yielded mixed results, both in terms of the effectiveness of the individual tools and the safety of the two test sites. Analysis of the data obtained requires interpretation in terms of the differences between the tools used and the differences between the applications themselves.

Discrepancies in the results of individual scanners stem directly from differences in their types and the distinct behaviour of each. OWASP ZAP, operating in active scanning mode, automatically performs tests on

all detected resources, making it the most effective at detecting configuration vulnerabilities. Nuclei relies on predefined YAML templates tailored to known vulnerability patterns, meaning its effectiveness is strictly dependent on the up-to-date status of the template database. The greatest strength of this scanner is the detection of known configuration errors and publicly documented vulnerabilities, whilst it is practically useless against logical vulnerabilities. Burp Suite CE, used in semi-automatic mode with the Repeater module, was the only tool that required the active participation of the tester when handling requests. This is both its greatest limitation and its greatest advantage, as it allows for the detection of vulnerabilities that no automated scanner is capable of detecting, thanks to the deliberate selection of payloads and the examination of specific endpoints.

The nature of OWASP Juice Shop as a deliberately vulnerable platform explains the broad range of vulnerabilities detected in the first phase of the study. The vulnerabilities present in it are the result of deliberate design choices rather than implementation errors, making it an appropriate environment for verifying the detection capabilities of the scanners used.

The security level of the application supporting the organisation of skydiving is acceptable in relation to publicly known vulnerability classes and given that the application was not specifically hardened for cybersecurity during its development; however, it cannot be unequivocally stated that the application is secure. Nevertheless, the identified security issues can be addressed through server configuration adjustments without the need to overhaul the application's architecture. The identified flaws in the security headers still constitute a real attack vector that could be skilfully exploited.

OWASP ZAP generates a large number of alerts (9 for the skydiving application and 431 for Juice Shop), a significant proportion of which relate to issues of low or informational severity, such as timestamp disclosure, missing headers or warnings concerning static resources. These results are useful but require filtering and manual verification to account for false positives. Burp Suite CE returns few results, but these are of high value to the overall assessment due to manual testing and verification of the vulnerabilities found, which reduces the risk of false positives. Nuclei returns a moderate number of results focused on template patterns, where, in the case of Juice Shop, it detected the public availability of API documentation and the absence of integrity verification attributes (SRI). For the proprietary application, Nuclei returned no results; however, this does not indicate the absence of vulnerabilities, but merely a lack of match with the available templates.

Comparing the results obtained with industry data, according to report [8], over 33% of vulnerabilities detected in typical production applications fall into the high or critical-risk categories, and SQL injection remains the most common critical vulnerability. No high-risk vulnerabilities or any of the key logical classes included in the OWASP Top 10 were found in the proprietary application under investigation, which allows

its security level to be assessed as clearly above the average for production applications described in the literature. The identified issues relate solely to the configuration of HTTP headers, which constitute a category of configuration vulnerabilities commonly found even in well-secured production applications.

The research conducted has several significant limitations. The tests were carried out in a local, isolated environment, which eliminated factors typically found in production environments, such as Web Application Firewall mechanisms. Furthermore, the scope of the manual tests was limited exclusively to endpoints selected by the author, meaning that certain elements of the application were not examined, and the version of Burp Suite used did not have a built-in automated scanner. Nuclei, on the other hand, severely limits comprehensive vulnerability testing by using strictly defined templates, which necessitates the use of a combination of different scanners.

None of the tools used had access to the source code of the tested applications, which ruled out the detection of vulnerabilities present solely at the implementation level. Neither ZAP nor Nuclei required authentication to perform a scan, meaning that some resources accessible only after logging in remained out of reach of these tools. Burp Suite CE tested both public resources and those requiring authorisation, but only at manually selected endpoints, where, prior to testing a given endpoint, a login to a specific account type was performed. Furthermore, ZAP's results are heavily dependent on the crawler's default configuration and the scanning settings applied, as different scanning aggressiveness settings would potentially yield different results. Nuclei is directly dependent on the up-to-date status of the template database and the application of a given template to the application. Vulnerabilities discovered after the date of the last template update will not be detected.

6. Conclusions

The proprietary web application demonstrated an acceptable level of security with regard to publicly known vulnerabilities and logical errors. No logical vulnerabilities such as SQL injection, XSS or broken access control were found. None of the three tools used confirmed the presence of such vulnerabilities in the tested endpoints. The only issues detected were deficiencies in the configuration of HTTP headers. The absence of high-risk alerts and a zero score in the OWASP Top 10 scan indicate that the application's main entry points are properly secured.

As a dedicated educational tool, OWASP Juice Shop has proven its worth as a testing and validation environment for the scanning tools used. Vulnerabilities deliberately implemented to correspond to the OWASP Top 10 categories have enabled the detection capabilities of each scanner to be assessed under controlled conditions simulating the presence of real security vulnerabilities.

In addition to the application testing itself, the study confirmed that none of the scanners used was capable of

detecting all types of vulnerabilities on its own. OWASP ZAP stood out for its superior universal automated scanning, Burp Suite CE performed best in verifying logical vulnerabilities and in semi-automated tests dedicated to selected endpoints, whilst Nuclei effectively supplemented the results with template-based vulnerabilities. This study confirms the usefulness of combining automated and manual vulnerability scanners in the context of a comprehensive security assessment of the applications under test.

Further research into application security analysis should include extending manual testing to all endpoints of the proprietary application and repeating the tests using Burp Suite, but in the Professional version with the automatic scanner. This would allow for a comparison of the effectiveness of both versions of the scanner and a comparison of the scan results with those from the ZAP tool. A broader range of results for analysis could also be obtained by extending the methodology to include static application security testing (SAST), which enables the detection of vulnerabilities in the source code.

Bibliography

- [1] S. Qadir, E. Waheed, A. Khanum, S. Jehan, Comparative evaluation of approaches & tools for effective security testing of Web applications, *PeerJ Computer Science* 11 (2025) e2821, <https://doi.org/10.7717/peerj-cs.2821>.
- [2] OWASP Top 10, <https://owasp.org/Top10/2025/>, [08.05.2026].
- [3] S. Alazmi, D. Conte de Leon, A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability, *IEEE Access* 10 (2022) 33200–33219, <https://doi.org/10.1109/ACCESS.2022.3161522>.
- [4] J. Shahid, M.K. Hameed, I.T. Javed, K.N. Qureshi, M. Ali, N. Crespi, A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. *Applied Sciences* 12(8) (2022) 4077 <https://doi.org/10.3390/app12084077>.
- [5] P. Zamościński, G. Koziel, Analysis of security CMS platforms by vulnerability scanners. *Journal of Computer Sciences Institute* 15 (2020) 252-260, <https://doi.org/10.35784/jcsi.2020>.
- [6] P. Jarupunphol, S. Seatun, W. Buathong, Measuring Vulnerability Assessment Tools Performance on the University Web Application, *Pertanika Journal of Science & Technology*, 31(6) (2023) 2973–2993, <https://doi.org/10.47836/pjst.31.6.19>.
- [7] A. Kondraciuk, A. Bartos, B. Pańczyk, Comparative analysis of the effectiveness of OWASP ZAP, Burp Suite, Nikto and Skipfish in testing the security of web applications. *Journal of Computer Sciences Institute* 24 (2022) 176-180, <https://doi.org/10.35784/jcsi.2929>.
- [8] Edgescan, 2024 Vulnerability Statistics Report, <https://www.edgescan.com/wp-content/uploads/2025/04/2024-Vulnerability-Statistics-Report.pdf>, [08.05.2026].
- [9] OWASP Juice Shop, <https://juice-shop.github.io/>, [08.05.2026].
- [10] M. Wisnu, B. Soewito, Security Assessment Based on OWASP Top 10 Using SonarQube and ZAP on Export and Import Applications in the LNSW, *INTENSIF: Jurnal Ilmiah Penelitian dan Penerapan Teknologi Sistem Informasi* 10(1) (2026) 36–52, <https://doi.org/10.29407/intensif.v10i1.25294>.