2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) | 979-8-3503-5805-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/IDAACS58523.2023.10348918

The 12th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
7-9 September, 2023, Dortmund, Germany

Scanning of Web-applications: Algorithms and Software for Search of Vulnerabilities "Code Injection" and "Insecure Design"

Oles Yudin, Vyacheslav Kharchenko, Vladimir Pevnev
National Aerospace University "KhAI", 61070 Kharkiv, Ukraine,
o.yudin@student.csn.khai.edu, v.kharchenko@csn.khai.edu, v.pevnev@csn.khai.edu

Abstract—**The work is dedicated to the analysis of algorithms and software tools for scanning web applications with the aim of detecting vulnerabilities according to OWASP top 10. Incorrect or insecure programming can lead to the emergence of vulnerabilities that can be exploited by malicious actors to gain unauthorized access to data or render applications inoperable, thereby violating confidentiality, integrity, and availability. The main objective of this research is to analyze existing algorithms and software capable of automatically scanning web applications for vulnerabilities such as "code injection" and "insecure design". The work presents an overview of existing vulnerabilities in web applications, specifically "code injection" and "insecure design", as well as describing the main approaches and methods used for their detection. Subsequently, scanning algorithms based on the identified approaches and methods are developed and discussed their implementation.**

Keywords—pentest web-application; web-application vulnerability; code injection; SQL-injection; insecure design.

I. INTRODUCTION

It is impossible to imagine the development of a web application without security testing. This is because modern web applications can contain numerous vulnerabilities. The number of vulnerabilities in an application cannot be accurately counted, as it changes every day. Attackers constantly discover new flaws in components related to the computing system, and a significant number of vulnerabilities remain unknown to both developers and the security department. These system flaws are referred to as "zero-day vulnerabilities" [1].

According to statistics from Snyk, only 24% of companies pay attention to checking the deployed source code of software [2]. Additionally, statistics show that when implementing automated information security testing, a vulnerability discovered through automated testing can be mitigated within one day with a 76% likelihood. If security testing is performed manually, the likelihood of success is 59%. However, if a company conducts security testing after deploying the software into the production environment, the likelihood of success drops to 38% [2]. This is because a significant portion of vulnerabilities can be found during the development and testing phases of the software.

When vulnerabilities are discovered in the production environment, it becomes much more challenging to address the issues as such actions can introduce defects into the software. For example, in the case of a database vulnerability in a web application, removing or updating the software may cause the existing database to stop interacting with the web application, thereby depriving users of a seamless experience with the web application.

Objectives and structure of the work described below**:**

- analysis of investigations related to web cyber security, a specialy attacks on the most known vulnerabilities (Section II);
- examination of SQL injection types and a typical algorithm for detecting SQL injection vulnerabilities. Development of an algorithm based on the several types of search for vulnerabilities. Analysis of existing solutions for finding "insecure design" vulnerabilities (Section III);
- a practical comparison of the previously discussed tools for detecting SQL injection and insecure design vulnerabilities was conducted (Section IV);
- discussion of the obtained results and future research steps (Section V).

The main contribution that this work brings is to create a common scanning scheme for detecting code injection and insecure design vulnerabilities.

II. RELATED WORKS

A. Code-injection attacks

There are numbers of studies dedicated to vulnerabilities specifically related to Structure Query Language (SQL) injection, as well as more general code injections. According to the Open Web Application Security Project (OWASP), these include XSS-attacks, SQL, NoSQL, Blind SQL, LDAP and other types of injections [3]. The main idea is for the request party (client) to input a value into a submission form or the URL in such a way that the Database Server and the Information

Security Management System, such as a Web Application Firewall, consider the request legitimate and process it. Since submission forms and URL addresses interact directly with the database, they can retrieve certain values from the database. This makes it possible to send malicious payloads to the server, and if the application is vulnerable, the client can exploit the attack on confidentiality, integrity, or availability.

The main objective of an attacker is to construct a query that will be evaluated as TRUE. Additionally, hackers may attempt other types of code injection attacks, such as Blind SQL Injection and Union Query. These attack types have been thoroughly investigated in the article [4].

The authors also considered the classification of SQL injection attacks. Work [5] examines different types of injections, as well as practical examples of exploiting vulnerabilities. During the testing of these examples in a vulnerable web application, unauthorized access to the web application was achieved, and confidential information stored in the MySQL Database was also exposed.

In article [6], methods of exploiting the "code injection" vulnerability are proposed, primarily from the perspective of an attacker. It would be reasonable to combine the exploitation methods from paper [6] to create a unified flowchart for the vulnerability detection algorithm for code injection.

B. Insecure design

Another significant vulnerability from the OWASP Top 10 list, which ranked fourth place in 2021, is called "insecure design". This is a new vulnerability that was not widely known before, although it existed since the early days of computing systems.

However, it gained more attention with the advancement of cloud computing and the focus on DevOps-engineering. In these contexts, security policies and international standards are often overlooked due to the complexity of configuring systems with best cybersecurity practices. Despite efforts made by cloud service providers such as Azure, Google, AWS, etc., ensuring secure system configurations and insecure design findings remains a challenging task.

Methods and vulnerability analysis for insecure design depend on the infrastructure and scale of the project.

The article [7] examines the security aspects of various platforms used for organizing eLearning processes. The analysis reveals that the main threats are the violation of accessibility and confidentiality of educational data, as well as the possibility of grade manipulation through the exploitation of vulnerabilities in the learning management system. The exploitation of the attacks discussed in the article can also be associated with "insecure design" vulnerabilities.

Paper [8] addresses the challenge of selecting the most appropriate tool for penetration testing of web applications by proposing a Web service that utilizes a neural network on the server side. The neural network is trained using data provided by experts in the field of penetration testing, enabling it to recommend tools based on specific requirements.

Article [9] discusses the potential application of neural networks in addressing the challenge of tool selection for penetration testing of web applications. The construction process of a question tree is outlined, which aims to minimize the number of questions users need to answer in a simplified questionnaire mode. Additionally, an expert mode is introduced, which includes more questions but provides a more precise formation of the requirements vector.

To timely identify security misconfigurations, it is recommended to divide the work into two stages: vulnerability scanning using static and dynamic scanners.

If the project's infrastructure is built using infrastructure as code (IaC) tools, static scanners like Checkov, Trivy, and Snyk [10] [11] [12] can be used.

As for dynamic scanning, where access to the application or infrastructure source code is not available, a network scanner like Nmap [13] is suggested. It helps uncover security weaknesses related to the network infrastructure.

One of the main reasons for the adoption of the Metasploit Framework (MSF) [14] by information security professionals is its open-source nature and active development. Unlike many other penetration testing tools, Metasploit provides flexible customization options, granting users full access to the source code and the ability to add their own modules. MSF allows testers to switch payloads in real-time using specific commands, such as "set payload", providing great flexibility when attempting to gain system access through shell-based access.

III. Algorithms of Vulnerabilities Search

The vulnerability search scheme for code injection can be based on performing several events:

1. Search for vulnerability presence based on a dictionary: This involves attempting to inject known malicious code or payloads from a predefined dictionary into the target application or database to identify if there is a vulnerability. In this context, various methods can be used to search for vulnerabilities, such as Error-based, Time-based injection, Union-based, and Boolean-based techniques.
2. Error-based vulnerability detection: By injecting specially crafted input, such as specific characters or SQL statements, the goal is to trigger an error response from the database, which can provide additional information that can be useful for further attacks or understanding the structure of the database.
3. UNION-based vulnerability detection: This technique involves leveraging the UNION operator in SQL queries to combine the results of multiple SELECT statements. By injecting UNION-based payloads, an attacker can attempt

to retrieve additional information from the database, such as accessing unauthorized data or bypassing authentication mechanisms.

By performing these events, the search scheme aims to identify the presence of code injection vulnerabilities and gather information that can be used for further exploitation or remediation.

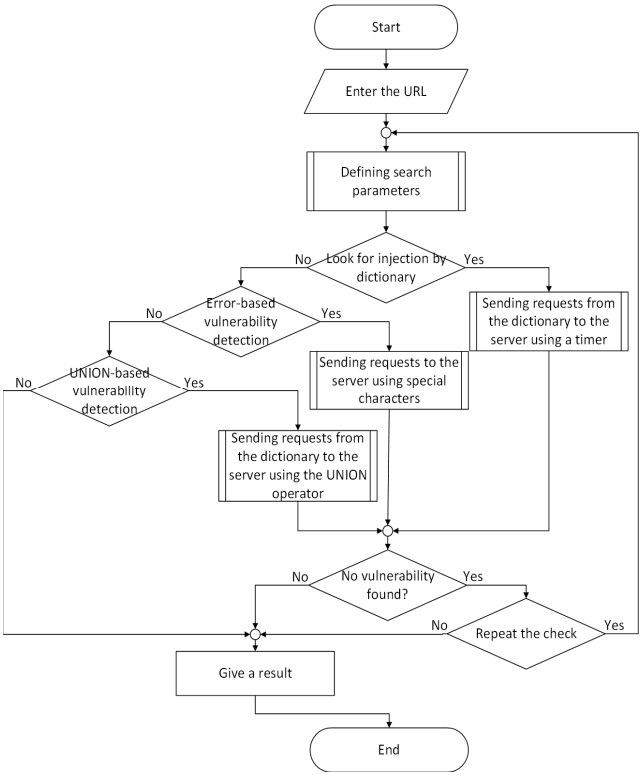

Figure 1. Block diagram of the code injection vulnerability search algorithm

This block diagram outlines a high-level algorithm for detecting "code injection" vulnerabilities, specifically focusing on SQL-injection attacks.

A dictionary-based search can help identify typical vulnerabilities in web applications and databases that lack proper protection or sanitization. This type of search does not require in-depth knowledge from the attacker's side, as it becomes possible to use ready-made programs for SQL injection detection. Some popular programs that can be mentioned include:

1. SQLMap [15] is a tool for automated detection of SQL injection vulnerabilities. It is the most popular program for identifying weaknesses in systems compared to its counterparts. The program allows for the detection of various types of vulnerabilities including Error-Based, Time-Based, Boolean-Based, Union-Based, and Command Injection.
2. Blisqy [16] is a tool designed to assist web security researchers in finding Time-based Blind SQL injection vulnerabilities in HTTP headers, as well as exploiting the same vulnerability. The exploitation involves the slow extraction of data from a database, currently supporting MySQL/MariaDB only, using bitwise operations on printable ASCII characters via blind SQL injection. The modules provided within Blisqy can be imported into other Python-based scripts, allowing users to create their own scripts for specific penetration testing tasks.
3. jSQL Injection [17] is an open-source tool specifically designed for detecting and exploiting SQL injection vulnerabilities in web applications. It provides a range of features to automate injection attacks and exploit default configuration weaknesses in databases. The main purpose of jSQL Injection is to identify and exploit SQL injection vulnerabilities in web applications that use SQL databases. It offers the following key features: Automated Enumeration and Cloning Attacks, Scanning Subnets and IP Lists, Password Cracking, JavaScript Injection Techniques, Timing-Based Attacks. It exploits time delays in responses to infer the success or failure of injected code.

The following programs are suggested for examining security vulnerabilities related to "insecure design":

- Checkov [18];
- Trivy [11];
- Snyk [12];
- Nmap [13];
- Metasploit Framework [14].

Checkov is an open-source static analysis tool designed for infrastructure as code (IaC) security [18]. It helps in identifying and preventing security misconfigurations in cloud environments. Checkov is primarily focused on scanning IaC templates written in popular formats such as Terraform, AWS CloudFormation, and Kubernetes YAML files. The most valuable feature for developers is the built-in support for popular integrated development environments (IDEs) such as Visual Studio Code (VS Code) and JetBrains IDEs. From a DevOps engineer's perspective, Snyk provides support for CI/CD, both for free systems and proprietary resources like AWS CodePipeline [19] and AWS CodeBuild [20].

Trivy is an open-source vulnerability scanner specifically designed for containerized environments. It focuses on identifying security vulnerabilities in container images and helps ensure that containers are free from known vulnerabilities before deployment [11]. The main advantage of the static scanner Trivy is its container-focused approach. Trivy is specifically designed to scan and identify vulnerabilities in container images. It analyzes the layers and dependencies within the container image to identify potential security issues and provides detailed

reports on vulnerabilities found. This makes Trivy particularly useful for securing containerized environments and ensuring the integrity of container images used in deployment pipelines.

Snyk is a platform that allows scanning and prioritizing information security, detecting vulnerabilities in modules hosted and/or not hosted in a tested repository, and identifying vulnerabilities in containers and infrastructure configurations as code (IaC) [21]. This tool is an enterprise solution, but users are provided with free access to limited scanning capabilities. The scanner can work with popular programming languages used for developing web applications. The list of supported programming languages for scanning can be found at [22].

Nmap [23] is commonly used for security auditing, but many system administrators find it useful for regular network management tasks as well, such as network exploration, service and schedule monitoring, and host or service uptime tracking. The main task of Nmap is to conduct reconnaissance on the internal network of a web application in order to have an understanding of the target and identify its vulnerabilities before an attack. It is possible to integrate Nmap into the CI/CD cycle. However, as mentioned earlier, the primary purpose of Nmap is network scanning. Since the CI/CD server is typically located within the internal infrastructure of the application and, following best security practices, does not have public access to the internet, there may not be a need to incorporate Nmap into the CI/CD cycle by default. Nmap already has access to the internal network by default. Nevertheless, network scanning can be performed after setting up the internal infrastructure of the web application to ensure that components of the web application that should not have public access to the internet, such as a database server, do not have privileged access.

Metasploit Framework [14] is a penetration testing platform that enables the discovery, exploitation, and validation of vulnerabilities. It is a popular tool used by both malicious hackers and ethical hackers. The Metasploit project includes anti-forensic and remediation tools, some of which are built into the Metasploit Framework. With Metasploit, security professionals can simulate real-world attacks to assess the security posture of systems and applications. It provides a wide range of exploits, payloads, and auxiliary modules that can be used to test and validate vulnerabilities in target systems. Metasploit Framework offers a command-line interface as well as a web-based interface called the Metasploit Console. It allows security practitioners to leverage a vast collection of known exploits and techniques, making it a valuable tool for penetration testing, vulnerability assessment, and security research. However, it is important to note that the use of Metasploit should always comply with legal and ethical guidelines, and it should only be used with proper authorization and consent.

The recommended tools have been used both separately and together to achieve more accurate results, because each tool has its own advantages and results, and in order to open a large attack spread, it is recommended to use several tools that combine them by type. For example, use the Snyk scanner for statistical analysis and the Metasploit Framework scanner as a complete pentesting tool in the cloud and locally.

IV. Case-Study

To identify SQL injection vulnerabilities related to "code injection" vulnerabilities, it is proposed to compare the previously mentioned vulnerability scanning tools such as SQLMap, Blisqy, and jSQL-injection. The authors conducted an independent experiment using the vulnerable web application Damn Vulnerable Web Application (DVWA) [24]. The app has been white box tested since the source code of the DVWA application was available during the testing phase.

Each software was examined for the main types of SQL injection attacks, such as Error-Based, Time-Based, Boolean-Based, Union-Based, and Stacked Queries SQL injection.

SQLMap. With the help of this software, it was possible to detect all the vulnerabilities mentioned above. It is worth noting that SQLMap is capable of working with cookies, which means that testing can be conducted even if authentication is required in the web application. If a specialist performs testing that requires prior authentication and does not enter correct cookie data or ignores them altogether, the request will be processed incorrectly as the program script will not be able to navigate within the application. Another significant advantage of SQLMap is its ability to bypass firewalls in search of SQL injection vulnerabilities by generating random user-agent client strings.

Blisqy. It utilizes advanced algorithms and techniques to thoroughly analyze application inputs and behaviors, allowing it to identify potential points of exploitation. In addition to vulnerability detection, Blisqy offers exploitation features that enable security professionals to gain unauthorized access to databases and manipulate data.

It provides a seamless experience, even for users without extensive knowledge in information security. Blisqy supports various scan types, including Error-Based, Time-Based, Boolean-Based, Union-Based, and Stacked Queries SQL injection techniques. This comprehensive coverage ensures the detection and assessment of different types of SQL injection vulnerabilities.

While Blisqy offers powerful capabilities, it may have some limitations. For example, it may not support certain features like cookie handling, which can be essential for testing web applications that require authentication. Nonetheless, Blisqy's advanced detection algorithms, exploitation capabilities, user-friendly interface, and support for various scan types make it a valuable tool for

security professionals seeking to identify and exploit SQL injection vulnerabilities in web applications.

jSQL Injection has a user-friendly graphical interface and allows users to quickly test vulnerable web applications without requiring additional knowledge in the field of information security. After inputting the vulnerable payload, the user can choose from available scan types such as Error-Based, Time-Based, and Blind-Based. However, due to its overly simplified usage, it lacks additional settings. For example, it does not support the use of cookies, making it unsuitable for web applications that require user authentication before the initial load.

The vulnerable application was also tested using a script based on the algorithm shown in Figure 1. A dictionary was incorporated into the script to assist in detecting SQL injection vulnerabilities.

The values in the dictionary can be categorized into several types, namely:

- Error-based injection – Special characters are utilized that deliberately cause errors, allowing the detection of vulnerabilities based on the response from the web application. If the application generates an error message or behaves unexpectedly, it indicates the presence of a vulnerability, specifically an SQL injection vulnerability;
- Time-based injection – queries are utilized to determine the existence of a vulnerability in a web application by observing the delay in the response. By analyzing the time, it takes for the web application to process and respond to these queries, it becomes possible to identify whether a vulnerability exists or not;
- Union-based injection – Various variations of the UNION operator are used, and if a web application is vulnerable, the user can receive a message that displays the injected SQL query, exploiting the SQL injection vulnerability.

It should be noted that the program cannot gain unauthorized access to the web application or attack the system in a way that would compromise confidentiality, integrity, or availability. Table 1 presents an example of SQL injections that are used in the dictionary.

Those are examples of queries that can be used in the dictionary to determine the presence of a vulnerability. In each example, the query ends with the symbol -- (two dashes), which represents a comment sign and is used to remove the remaining part of the expression in order to avoid triggering any syntax checks that may be implemented in the application.

The word presented in Table 1 is only an example and was compiled based on the description of the type of vulnerabilities for SQL injection [4, 5, 6].

The tools and methods for identifying errors and flaws in the "insecure design" vulnerability may vary depending on the server's operating system and additional infrastructure components. Therefore, it is impossible to present a single tool that will provide acceptable results in different projects and infrastructure solutions. Using the dictionary, it was possible to identify 3 types of SQL injection.

TABLE I. DICTIONARY SQL QUERYES

Type of SQL-injection	SQL query with injection
Error-based	'OR 1=1/0 -- 'OR 'abc'='abc' AND 1=1/0 -- '; EXEC non_existent_procedure; -- 'UNION ALL SELECT 1, @@VERSION – 'OR (SELECT COUNT(*) FROM information_schema.tables) = 0 -- 'OR ROW(1,1)>(SELECT COUNT(*),CONCAT(CHAR(58,58,58), version(), CHAR(58,58,58))x FROM information_schema.tables GROUP BY x) -- 'AND (SELECT TOP 1 column_name FROM information_schema.columns) = 'username' --
Time-based	'OR SLEEP(5) -- 'AND IF(1=1, SLEEP(5), NULL) – 'AND IF(1=1, (SELECT SLEEP(5)), NULL) -- '; SELECT CASE WHEN (1=1) THEN pg_sleep(5) ELSE NULL END; -- 'AND IF(1=1, SLEEP(5), BENCHMARK(1000000,MD5(1))) -- 'UNION ALL SELECT NULL, IF(1=1, (SELECT SLEEP(5)), NULL) -- 'UNION ALL SELECT NULL, IF(1=1, SLEEP(5), BENCHMARK(1000000,MD5(1))) --
Union-based	'UNION SELECT null -- 'UNION SELECT column_name FROM information_schema.columns -- ' UNION SELECT username FROM users -- 'UNION SELECT password FROM users WHERE username='admin' -- 'UNION SELECT credit_card_number FROM customers -- 'UNION SELECT NULL, CONCAT(username, ':', password) FROM users -- 'UNION SELECT LOAD_FILE('/etc/passwd') --

The study analyzed two static and two dynamic vulnerability scanners for web applications. It is worth noting that the obtained results may not be relevant as the comparison is made between programs that employ static and dynamic approaches simultaneously.

According to statistics [25], static scanners tend to produce a significantly higher number of false positives compared to dynamic scanners. However, the detection of vulnerabilities using a dynamic scanner takes much more time, but the time invested is compensated by the results obtained.

V. CONCLUSIONS

A. *Discussion*

This article defines types of attacks for exploiting SQL injection, which fall under the category of "code injection" vulnerabilities. The outcome of the research is the examination of existing software solutions for detecting

SQL injection vulnerabilities. An analysis was made of various tools for searching SQL injection. As a result of the analysis, the advantages and disadvantages of each tool are considered.

It is important to note that the dictionary with the script can be used as an initial check for the presence of SQL injection vulnerabilities. From the algorithm, it is evident that the script belongs to Static Application Security Testing (SAST) security tools. In the future, the script can be improved and integrated into the CI/CD process of the software development lifecycle (SDLC) to obtain vulnerability information in a timely manner.

Identifying insecure design vulnerabilities is a very difficult task and there is no universal scanner that will detect this vulnerability without false positives. However, you can use as example Metasploit Framework that could close the bulk of the vulnerabilities. The Metasploit Framework can detect the presence of SQL vulnerabilities. After that, investigate in detail the place where the vulnerability was found, you can close the gap in the security system and architectural design of applications. It also depends a lot on the «ecosystem» where the application is deployed. For example, the Amazon Web Services Cloud (AWS) provides many backend securities services such as AWS Guard Duty, Cognito, Security Hub, Inspector, WAF, and other.

To identify vulnerabilities related to "insecure design", there are plans to conduct research and testing in the AWS cloud environment using existing solutions.

The results of this work will be filtered and processed to develop an algorithm for detecting "insecure design" vulnerabilities. This algorithm will help in identifying common design flaws and weaknesses in web applications hosted on AWS, allowing for appropriate remediation measures to be implemented.

B. Research and development directions

Further research will focus on filtering, processing, and analyzing the obtained data. Additionally, there are plans to develop software for detecting SQL injection vulnerabilities of specific types, including Error-Based, Time-Based, Boolean-Based, and Union-Based.

The software will aim to automate the detection process and enhance the efficiency of identifying and mitigating SQL injection vulnerabilities in web applications and databases.

REFERENCES

[1] Mitigate zero-day vulnerabilities, *Microsoft*. URL: https://learn.microsoft.com/en-us/microsoft-365/security/defender-vulnerability-management/tvm-zero-day-vulnerabilities .

[2] Snyk research report, Infrastructure as Code Security Insights, Snyk, February 2021.

[3] Code Injection. *OWASP*. URL: https://owasp.org/www-community/attacks/Code_Injection.

[4] H. R. Yasith Wimukthi, H. Kottegoda, D. Andaraweera, P. Palihena, "A comprehensive review of methods for SQL injection attack detection and prevention," *International Journal of Scientific Research in Science and Technology IJSRST*, 2022. URL: https://www.researchgate.net/publication/364935556_A_comprehensive_review_of_methods_for_SQL_injection_attack_detection_and_prevention.

[5] A. Tetskyi, "Examining methods for retrieving database content using SQL injection," *Open Information and Computer Integrated Technologies*, no. 66, 2014.

[6] D. O. Aduragbemi, A. A. M. Arabi, E. N. Tadiwa, "SQLIA types and techniques – A systematic analysis of effective performance metrics for SQL injection vulnerability mitigation techniques," *International Journal of Infrastructure Research and Management*, vol. 10, issue 1, pp. 16–28, June 2022.

[7] A. Tetskyi, O. Morozova, "Cybersecurity aspects of e-learning platforms," *Radioelectronic and Computer Systems*, no. 4(96), pp. 93-97, 2020. DOI: 10.32620/reks.2020.4.08.

[8] A. Tetskyi, V. Kharchenko, D. Uzun, A. Nechausov, "Architecture and model of neural network based service for choice of the penetration testing tools, *International Journal of Computing*, vol. 20, issue 4, pp. 513-518, 2021. DOI: 10.47839/ijc.20.4.2438.

[9] A. Tetskyi, V. Kharchenko and D. Uzun, "Neural networks based choice of tools for penetration testing of web applications," *Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv, Ukraine, 2018, pp. 402-405, doi: 10.1109/DESSERT.2018.8409167.

[10] Policy-as-code for everyone. Checkov. URL: https://www.checkov.io

[11] Trivy. Aqua Trivy. URL: https://aquasecurity.github.io/trivy/v0.19.2 .

[12] Open Source Vulnerability Database. Snyk. URL: https://security.snyk.io.

[13] Network Mapper. Nmap. URL: https://nmap.org .

[14] The world's most used penetration testing framework. Metasploit Framework. URL: https://www.metasploit.com .

[15] Automatic SQL injection and database takeover tool. SQLMap. URL: https://sqlmap.org.

[16] Penetration Testing for Blind SQL Injection using BBQSQL. Hacking Loops. URL: https://www.hackingloops.com/penetration-testing-for-blind-sql-injection-using-bbqsql .

[17] jsql-injection. Kali org. URL: https://www.kali.org/tools/jsql .

[18] Checkov quick start. Checkov. URL: https://www.checkov.io/1.Welcome/Quick%20Start.html.

[19] AWS CodePipeline. Amazone Web Services. URL: https://aws.amazon.com/codepipeline .

[20] AWS CodeBuild. AWS. URL: https://aws.amazon.com/codebuild .

[21] Introduction to Snyk, Snyk Docs. Snyk. URL: https://docs.snyk.io/introducing-snyk .

[22] Snyk supported languages for application scanner. Snyk. URL: https://docs.snyk.io/introducing-snyk/snyk-languages-and-integrations.

[23] Nmap reference guide. Nmap. URL: https://nmap.org/book/man.html#man-description .

[24] Description of Damn Vulnerable Web Application. Kali Linux. URL: https://kali.tools/?p=1820 .

[25] L. Dencheva, *Comparative Analysis of Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) by using Open-source Web Application Penetration Testing Tools*, National College of Ireland, 2022.