DOI: 10.25140/2411-5363-2022-4(30)-113-120
UDC 004.056.53

Yevhenii Berloh[1], Andrii Rohovenko[2], Hanna Dyvnych[3]

[1]PhD student of the Department of Information and Computer Systems
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: evgeniy.berlog@gmail.com

[2]PhD in Technical Sciences, associate professor of the Department of Information and Computer Systems
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: arogovenko@gmail.com. **ORCID:** https://orcid.org/0000-0003-4594-5692. **ResearcherID:** G-3926-2014

[3]PhD in Public Administration, associate professor of the Department of Foreign Philology
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: anyahaidai@gmail.com. **ORCID:** https://orcid.org/0000-0003-4240-5391. **ResearcherID:** R-1613-2016

RESEARCH OF METHODS OF AUTOMATED SEARCH OF "SQL INJECTION" TYPE VULNERABILITIES IN WEB APPLICATIONS

The article presents the results of a scientific and methodological study of the methods of automated search for SQL vulnerabilities in web applications. An example of an attack using a typical SQL injection is provided. The classification of web application security assessment methods based on penetration testing is given. The results of practical studies of the operation of the most widely used web scanners for automated vulnerability testing of web applications are given. Based on the results, a comparison of the effectiveness of penetration testing methods has been made. The possible directions of further research into the methods of automated search for SQL vulnerabilities in web applications are substantiated, taking into account the results obtained, in particular the values of the Youden Index.

Key words: *SQL; SQL injection; OWASP; SPA; WVS; Black Box.*
Table: 1. Fig.: 3. References: 16.

Urgency of the research. Modern society is more and more relying on web applications, transferring its life to a digital level, which is raising the level of security requirements. Due to the complexity of modern web applications, finding vulnerabilities is a very difficult task, the solution of which does not always give positive results. Both for the research community and for the data security industry, it is important to develop research methods for finding vulnerabilities in web applications, taking into account the degree of impact of web technologies on the life of our society. According to Internet Live Stats as of February 2019 [3], hundreds of millions of websites are attacked every day, causing significant harm to a large number of people.

According to the leading web security organization Open Web Application Security Project (OWASP), one of the most common security risks to web applications is code injection [1], such as cross-site scripting (XSS) and structured query language injection (SQL) [2].

Target setting. In the case of site testing for vulnerabilities, either automated tools or manual tests are used. Manual testing has a number of disadvantages, of which the most significant should be highlighted:

- Time-consuming. A person significantly loses to computer systems in terms of speed.
- High probability of low-quality testing. The quality of the tests depends on the skills of the specialist who develops them. There is a risk that some of the vulnerabilities will be missed due to the human factor.

Thus, taking into account the volume and variety of modern web applications, the use of automated testing is extremely relevant. However, at the moment, there is no clear information to what extent existing methods and automated web application testing tools effectively detect code injection-based vulnerabilities. To determine the effectiveness, it is necessary to conduct a series of tests using existing automated testing tools.

Actual scientific researches and issues analysis. Many works have been devoted to the issue of the effectiveness of automated tools designed to find vulnerabilities in web applications built on the basis of various methods.[1] Most of them are characterized by low efficiency. At the same time, there are many publications on the possible improvements of automated tools designed to find vulnerabilities in web applications [6; 9]. The existing test sets for most types

of vulnerabilities in web applications are described in great detail in the literature [12]. There is also some information on testing certain types of vulnerabilities [13]. According to the results of the analysis of recent studies, it can be concluded that there is enough information available to the general public about commercial tools for testing the vulnerability of web applications.

Uninvestigated parts of general matters defining. An unsolved task is to determine the effectiveness of open-source web vulnerability testing tools based on penetration testing methods. In this work, special emphasis is placed on the fact that in the case of low efficiency of open-source testing tools, it is possible to analyse the reasons that create this deficiency and propose a solution to the problem. It is necessary to conduct a study of existing open-source web scanners and perform an analysis of their performance according to the Youden Index.

The research objective. The purpose of this work is a practical study of tools for searching vulnerabilities in web applications, as well as an analysis of their work efficiency according to the Youden index and the determination of directions for the development of automated tools in order to further simplify the process of protecting web applications from attacks based on code injections.

The statement of basic materials. SQL injection is classified as a code injection attack where an attacker enters malicious SQL queries into an input field. An example of a typical SQL injection can be seen in Figure 1. All types of SQL injection are very similar to this example. By making an SQL injection, an attacker can either change the database or show the contents of those data that would normally be inaccessible [10].

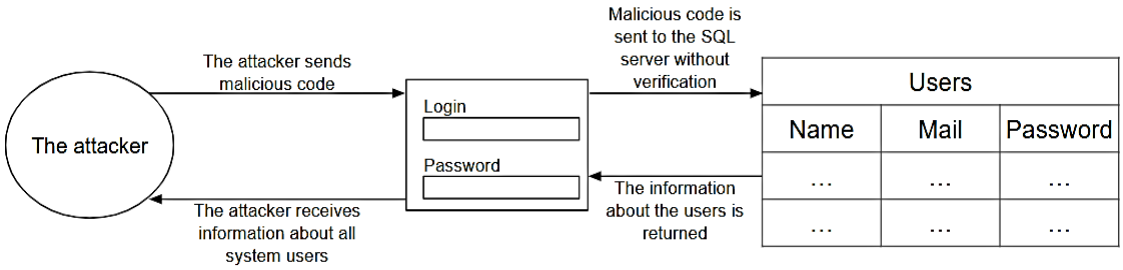

Fig. 1. An example of a typical SQL injection

Security assessment methods based on penetration testing (pentests) can be classified into several categories depending on the level of knowledge about the system to be tested. This knowledge may include the programming language, the type of database used in the application, the technologies used. In general, the tests are commonly divided into three categories:

- Black Box;
- White Box;
- Grey Box.

Black Box is a type of pentests when the developer knows only the address of the application that needs to be tested. In the case of web applications, this is the IP address (or domain name) and the port. In most cases, Black Box testing is a combination of manual and automated testing. The advantage of this type is that the tester is in the same conditions as the external hacker and therefore can find the same vulnerabilities. The disadvantage is that the tester does not have access to internal services.

White Box testing is the opposite of Black Box. In this case, the tester has full access to the code, documentation, architecture, user credentials, etc. The advantage of this type of testing is that the tester can find some vulnerabilities based on code analysis. Also, in this case, you can test both external and internal services. This type of testing is more resource-consuming compared to Black Box testing.

Grey Box testing is a mix of White Box and Black Box. In this case, the tester may have a certain piece of information that may be useful for testing [4].

One of the tools used for automated Black Box testing is the Web Vulnerability Scanner (WVS). WVS scans a web application for specific vulnerabilities. Usually, it consists of three modules: web scanner, fuzzer and analyzer [6].

The first to come into play is the web scanner, which is used to analyse the web application, to determine what data the web application can accept as input and to build the structure of the web application pages. When the first stage is completed, the fuzzer generates data that can be used to attack the web application and actually executes the attacks. Finally, the analyser evaluates the result to determine which attacks were successful and which were not. Based on these results, the analyser generates a report with a list of vulnerabilities that were found with a detailed description of how exactly this particular vulnerability was found.

The web scanner automatically searches the web application for URLs derived from the original URL. It is traditionally used in search engines such as Google and Bing. The scanner's workflow begins by loading a web page with an initial URL and searching the HTML DOM for related URLs. One of the resources that the scanner pays attention to is the robots.txt file, which is used to inform the scanner of which resources it should not access. Another type of file that the scanner analyses during this process is the XML sitemap. This file is a list of important web pages and is essential for building the website structure. The algorithm of the scanner can be seen in Fig. 2.

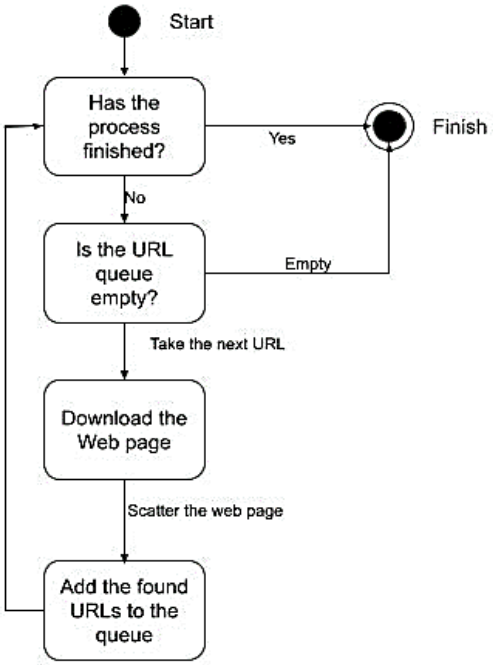

Fig. 2. Algorithm of the web scanner

Fuzzing is an automated testing technique that randomizes the input for programs and focuses on edge cases with the use of invalid data.[7] Fuzzing was first described by Miller [8] in a program called The Fuzz. This program generated random characters as input for testing UNIX utilities.

There are two traditional types of fuzzing techniques, which are listed below:

- Mutation-based fuzzing is based on collecting correct data, and randomly or heuristically changing that data. The data is then used to attack the system under test (SUT) and observe the behaviour of the web application. An example of mutation-based fuzzing with heuristics can be a change in the length of an input string.
- Another traditional type of fuzzing is generation-based fuzzing, which uses a specification describing the composition of the input data, such as a file format, to automatically generate semi-valid input data. This is often combined with various types of fuzzing heuristics, such as

very long empty strings. Mutation-based fuzzing is generally easier to get started with because it doesn't take much to know about SUTs. However, mutation-based fuzzing generally results in lower code coverage compared to generation-based fuzzing [14].

The analyser's job is to decide if a vulnerability exists in a web application. This is resolved based on the response received from a fuzzer attack on the web application [9]. The analyser tries to find common patterns in the error messages that indicate that this part of the web application has some kind of vulnerability.

Since the purpose of this article is to analyse the methods and efficiency of performance of scanners, several scanners should be selected as the subject of testing and analysis. Based on the information from the sources, a large number of available open-source scanners can be identified. Therefore, the following criteria were used as a basis for the selection of experimental specimens:

- The ability to work with SPA (Single Page Application) web applications;
- Code openness.
- Project support by the developer community.
- The ability to search for SQL injections.

As a result of the analysis, three scanners were selected.

OWASP Zap (Zed Attack Proxy) is one of the most popular WVS with a huge community behind it and so it was chosen as the first focus scanner [15]. It has shown good results according to some previous articles and consists of many different functionalities, making it an interesting tool for this project. Zap is created and maintained by the OWASP Foundation [1], runs on Windows, Linux, and Mac OS, and is written in Java. ZAP uses a GUI from which the user can perform scans, but it also acts as a proxy server so that the user can observe and manipulate all the traffic that passes through it. OWASP Zap can work with both traditional and AJAX scanners.

Arachni is a high-performance, free and open source ruby-based scanner designed to help administrators and testers assess the security of web applications. Arachni supports multiple platforms, including Windows, Linux, and Mac OS X, and can be instantly used on any system. Arachni includes both a command line interface (CLI) and a web user interface for multi-user management [16].

Wapiti is the third and final scanner selected for testing. Wapiti differs from the other two scanners in several ways:

It is written in Python and only works on Linux.

It has no GUI and does not act as a proxy server.

This scanner is different from the other two and also has active community support, making it an interesting choice for analysis.

The OWASP Benchmark test was launched in 2015 to evaluate the accuracy, coverage and speed of web application vulnerability scanners. Being an open-source program, organizations and researchers can use this framework to evaluate web vulnerability scanners using thousands of test cases provided by the OWASP Benchmark for eleven different vulnerability categories. These categories include cross-site scripting (XSS), insecure cookie, LDAP injection, structured query language (SQL) injection, and many others. OWASP Benchmark is implemented in Java and can be used to evaluate different types of scanners. Although OWASP Benchmark is a free and open-source program, it remains one of the most up-to-date because it is actively supported by the open-source community. Thus, the OWASP Benchmark can be considered an effective option for measuring the effectiveness of vulnerability scanners. It evaluates a tested scanner based on true positive rate, false positive rate, true negative rate, and false negative rate.

The score obtained by the OWASP Benchmark is the Youden index, which is a standard method that summarizes the accuracy of a set of tests [12]. OWASP Benchmark calculates an individual score for each category of test cases, called the Benchmark Accuracy Score, ranging from 0 to 100. The following example provides an overview of how OWASP Benchmark calculates the Scanner Accuracy Score.

Let's say the scanner showed a true positive rate (TPR) of 88 % and a false positive rate (FPR) of 15 %; This means that its sensitivity = TPR (0.88) and its specificity = 1-FPR (0.85). So, the Youden Index is (0.88+0.85) - 1 = 0.73 and the OWASP score is 73 because it normalizes the results to a range of 0 to 100.

Each of the scanners was run on the OWASP Benchmark tests and the scan results are shown in Table 1 below. Metrics including TP, FN, TN, FP, TPR and FNR and the Youden index have been calculated for each web vulnerability scanner.

Table 1 – The results of the study of scanners on the OWASP Benchmark

WVS	TP	FN	TN	FP	Total of tests	TP percentage	FP percentage	Youden Index
OWASP ZAP v2.12.0	160	112	215	17	504	58,82%	7,33%	51,50%
Arachni v1.6.1.3	136	136	232	0	504	50,00%	0,00%	50,00%
Wapiti v3.1.4	153	119	232	0	504	56,25%	0,00%	56,25%

Fig. 3 highlights the performance evaluation of each of the selected scanners based on the OWASP Benchmark tests for SQL Injection vulnerability.

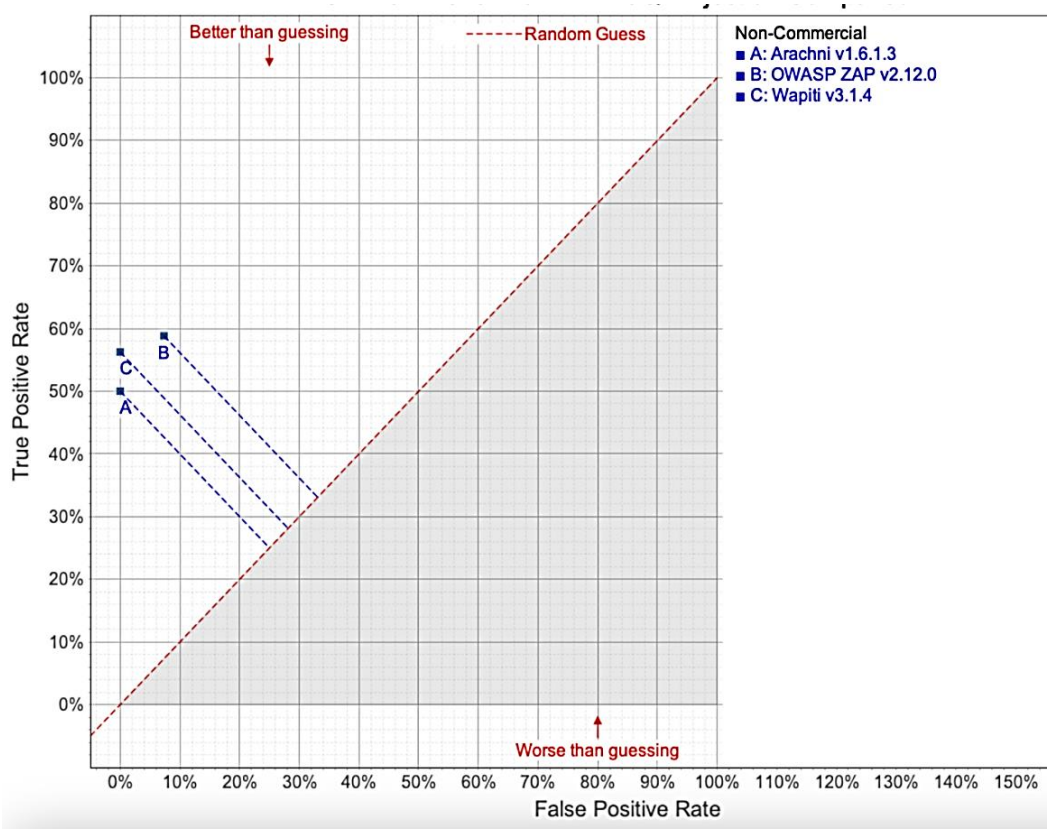

Fig. 3. Performance evaluation of the scanners based on the OWASP Benchmark

As a result, Wapiti has shown the best results with a Youden index of 56.25, ZAP is in second place with an indicator of 51.50 and Arachni is in third place with a result of 50.00.

Conclusions.

- An example of an attack on a web application is given and the main elements and stages of a typical SQL injection are highlighted;
- The classification of web application security assessment methods based on penetration testing is provided.

- The results of practical studies of the operation of the most widely used open-source web scanners for automated vulnerability testing of web applications are presented.

The calculated Youden index allows us to conclude that the Wapiti scanner has the highest efficiency, that is, it has detected the largest number of vulnerabilities. But even this scanner has a Youden coefficient of less than 60%, so the task of improving the efficiency of searching for SQL injection vulnerabilities is highly relevant for further research. In addition, during the study, it was determined that the process of scanning for vulnerabilities of the "SQL injection" type is more complicated for applications of the SPA (Single Page Application) type, which also requires a solution as a special case of the problem of increasing the effectiveness of the search for vulnerabilities of web applications.

Список використаних джерел

OWASP, Top 10 2021 [Electronic resource]. – 2021. – Available: https://owasp.org/www-chapter-minneapolis-st-paul/download/20211216_OWASP-MSP_OWASP_Top_Ten_2021.pdf?raw=true.

Halfond W. G. J. A classification of SQL injection attacks and countermeasures / W. G. J. Halfond, J. Viegas, A. Orso // Proceedings of the IEEE international symposium on secure software engineering. – 2006. – Vol. 1. – Pp. 13-15.

W3C, Internet live stats [Electronic resource]. – 2019. – Available: http://www.internetlivestats.com.

Mohd. Ehmer Khan. A Comparative Study of White Box, Black Box, and Grey Box Testing Techniques / Mohd Ehmer Khan, Fareema Hkaan // International Journal of Advanced Computer Science and Applications. – 2012. – Vol. 3, No 6. – Pp. 12-14. – Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.261.1758&rep=re p1&type=pdf.

Benjamin Eriksson. Black Widow: Black-box Data-driven Web Scanning / Benjamin Eriksson, Giancarlo Pellegrino, Andrei Sabelfeld // 2021 IEEE Symposium on Security and Privacy (SP). – 2021. – Pp. 1125–1142. – DOI: 10.1109/SP40001.2021.00022.

Siham el Idrissi. Performance evaluation of web application security scanners for prevention and protection against vulnerabilities / Siham el Idrissi, Naoual Berbiche, Fatima Guerouate, Sbihi Mohamed // International Journal of Applied Engineering Research. – Jan. 2017. – Vol. 12 – Pp. 11068–11076.

Oehlert P. Violating assumptions with fuzzing" / Peter Oehlert // IEEE Security Privacy. – 2005. – Vol. 3.2. – Pp. 58–62. – DOI: 10.1109/MSP.2005.55.

Barton P. Miller. An Empirical Study of the Reliability of UNIX Utilities / Barton P. Miller, Louis Fredriksen, Bryan So // Commun. ACM. – Dec. 1990. – Vol. 33.12. – Pp. 32–44. – DOI: 10.1145/96267.96279.

Adam Doupé. Why Johnny can't pentest: An analysis of black-box web vulnerability scanners / Adam Doupé, Marco Cova, and Giovanni Vigna // International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. – Springer, 2010. – Pp. 111–131.

William G. Halfond. A classification of SQL-injection attacks and countermeasures / William G. Halfond, Jeremy Viegas, and Alessandro Orso // Proceedings of the IEEE international symposium on secure software engineering. IEEE. – 2006. – Vol. 1. – Pp. 13–15.

OWASP. (2022). OWASP Benchmark [Electronic resource]. – Available: https://www.owasp.org/index.php/Benchmark.

Vulnerability Scanning Tools, 2022 [Electronic resource]. – Available: https://owasp.org/www-community/Vulnerability_Scanning_Tools.

Matti E. Evaluation of open source web vulnerability scanners and their techniques used to find SQL injection and cross-site scripting vulnerabilities : Student thesis. – Linköpings universitet, Institutionen för datavetenskap, 2021.

OWASP. (2016). Fuzzing [Electronic resource]. – Available: https://www.owasp.org/index.php/Fuzzing.

OWASP ZAP [Electronic resource]. – URL: https://github.com/zaproxy.

Laskos T. Arachni Application Security Scanner Framework [Electronic resource] / T. Laskos. – 2017. – Available: http://www.arachni-scanner.com.

References

1. OWASP Top 10 2021, 2021. https://owasp.org/www-chapter-minneapolis-st-paul/download/20211216_OWASP-MSP_OWASP_Top_Ten_2021.pdf?raw=true.

2. Halfond, W.G.J., Viegas, J., Orso, A. (2006). A classification of SQL injection attacks and countermeasures, *1*, 13-15.

3. W3C, Internet live stats. (2019). http://www.internetlivestats.com.

4. Mohd, Ehmer Khan, Fareema, Hkaan. (2012). A Comparative Study of White Box, Black Box, and Grey Box Testing Techniques. *International Journal of Advanced Computer Science and Applications, 3*(6), 12-14. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.261.1758&rep=rep1&type=pdf.

5. Benjamin, Eriksson, Giancarlo, Pellegrino, & Andrei Sabelfeld. (2021). Black Widow: Black-box Data-driven Web Scanning". *2021 IEEE Symposium on Security and Privacy (SP)* (pp. 1125–1142). doi: 10.1109/SP40001.2021.00022.

6. Siham, el Idrissi, Naoual, Berbiche, Fatima, Guerouate, & Sbihi, Mohamed. (Jan. 2017). Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. *International Journal of Applied Engineering Research, 12*, 11068–11076.

7. Oehlert, P. (2005). Violating assumptions with fuzzing. *IEEE Security Privacy*, 3.2, 58–62. doi: 10.1109/MSP.2005.55.

8. Barton, P. Miller, Louis, Fredriksen, & Bryan, So. (Dec. 1990). An Empirical Study of the Reliability of UNIX Utilities. *Commun. ACM*, 33.12, 32–44. doi: 10.1145/96267.96279.

9. Adam, Doupé, Marco, Cova, & Giovanni, Vigna. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 111–131). Springer.

10. William, G. Halfond, Jeremy, Viegas, & Alessandro, Orso. (2006). A classification of SQL-injection attacks and countermeasures. *Proceedings of the IEEE international symposium on secure software engineering.* IEEE, *1*, 13–15.

11. OWASP. (2022). OWASP Benchmark. https://www.owasp.org/index.php/Benchmark.

12. Vulnerability Scanning Tools (2022). https://owasp.org/www-community/Vulnerability_Scanning_Tools.

13. Erik, Matti. (2021). *Evaluation of open source web vulnerability scanners and their techniques used to find SQL injection and cross-site scripting*. Vulnerabilities.

14. OWASP. (2016). Fuzzing. https://www.owasp.org/index.php/Fuzzing.

15. OWASP ZAP. https://github.com/zaproxy.

16. Laskos, T. (2017). *Arachni Application Security Scanner Framework*. http://www.arachni-scanner.com.

Отримано 15.11.2022

УДК 004.056.53

Євгеній Берлог[1], Андрій Роговенко[2], Ганна Дивнич[3]

[1]аспірант кафедри інформаційних та комп'ютерних систем
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: evgeniy.berlog@gmail.com

[2]кандидат технічних наук, доцент кафедри інформаційних та комп'ютерних систем
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: arogovenko@gmail.com. **ORCID:** https://orcid.org/0000-0003-4594-5692. **ResearcherID:** G-3926-2014

[3]кандидат наук з державного управління, доцент кафедри іноземної філології
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: anyahaidai@gmail.com. **ORCID:** https://orcid.org/0000-0003-4240-5391. **ResearcherID:** R-1613-2016

ДОСЛІДЖЕННЯ МЕТОДІВ АВТОМАТИЗОВАНОГО ПОШУКУ ВРАЗЛИВОСТЕЙ ТИПУ «SQL INJECTION» У ВЕБДОДАТКАХ

У статті представлено результати науково-методичного дослідження методів автоматизованого пошуку SQL вразливостей у вебдодатках.

Наведено приклад атаки з застосуванням типової SQL ін'єкції. Усі SQL ін'єкції будуються за подібною структурою, тобто результатом SQL ін'єкції, може бути несанкціонована зміна вмісту бази даних, або отримання доступу до інформації, яка за звичайних умов недоступна.

Наведено класифікацію методів оцінки безпеки web-додатків на основі тестування на проникнення. Зокрема, при класифікації враховуються мова програмування, тип бази даних яка використовується в додатку та використані технології. Виділено три категорії тестів, та наведені відмінності між ними.

Розглянуто підходи до тестування на проникнення такі як Black Box, White Box та Grey Box. Беручи до уваги необхідність автоматизації тестування та максимальної імітації хакерської атаки ззовні було обрано підхід Black Box підхід тестування.

Розглянуто алгоритм роботи засобів пошуку вразливостей у web-додатках, зокрема веб-сканера (WVS). Також розглянуто особливості використання традиційних методів знаходження вразливостей у web-додатках беручи до уваги типову архітектуру WVS та беручи до уваги типову відмінність у роботі WVS при роботи з традиційними вебдодатками та SPA додатками.

Наведено результати практичних досліджень роботи найуживаніших вебсканерів для автоматизованого тестування вразливості вебдодатків, зокрема досліджено сканери OWASP Zap, Arachni, Wapiti.

Обчислено індекс Юдена, який дав змогу зробити висновок, що найбільша ефективність є у сканера Wapiti, тобто він виявив найбільшу кількість вразливостей. Але навіть цей сканер має коефіцієнт Юдена менш ніж 60 %, отже задача підвищення ефективності пошуку вразливостей типу "SQL ін'єкція" є дуже актуальним для подальшого дослідження. Додатково при дослідженні було визначено, що процес сканування на предмет вразливостей типу "SQL-ін'єкція" є складнішим для додатків типу SPA (Single Page Application), що також потребує рішення в якості окремого випадку задачі підвищення ефективності пошуку вразливостей web-додатків.

***Ключові слова**: SQL; SQL-ін'єкція; OWASP; SPA; WVS; Black Box.*

Berloh, Ye., Rohovenko, A., Dyvnych, H. (2022). Research of methods of automated search of "SQL injection" type vulnerabilities in web applications. *Technical sciences and technologies*, (4(30)), 113-120.