Exploring the Applicability of Open-Source Tools for Web Application Cybersecurity Improvement

Ivan Cvitić, Dragan Peraković(✉), Marko Periša, and Mario Sekondo

Faculty of Transport and Traffic Sciences, University of Zagreb, Vukelićeva 4, 10000 Zagreb, Croatia
{ivan.cvitic,dragan.perakovic,marko.perisa}@fpz.unizg.hr

Abstract. The security of the information-communication system is crucial to avoid potential cyber-attacks. Web applications are most vulnerable to attacks, so it is very important to determine the most common vulnerabilities and the best tools to improve the security of such applications. Vulnerabilities are potential flaws in the system that make it prone to potential attacks. These vulnerabilities can stem from various sources, such as programming languages with inherited security flaws, bad security coding practices, outdated or unpatched services etc. In order to improve security of web applications, the system as a whole needs to be assessed. One of the ways to improve the security is to hire a third-party company that specializies in pen-testing and security of such systems. But since security is complex and needs to be thoroughly tested, this service is rather expensive. So for a smaller web applications and projects this may not be the best or the smartest option. So in order to improve security one of the options is use of vulnerability assessment tools such as open-source vulnerability scanners. This paper will analyze technologies that are used for the development of web applications, the most common vulnerabilities encountered and open source tools that can be used to improve web application security.

Keywords: Web application · Security · Open source tools · Vulnerability

1 Introduction

In today's world, technology is an indispensable part of human lives. The development of the Internet and information technology has changed the way of life in which people now devote more and more free time to use various information and communication services for entertainment and personal needs or use them to do remote work. Businesses are expanding or changing the way they do business precisely because of the development of these technologies. Whether businesses are building their own web solutions or hiring companies to develop such solutions, it is important to keep in mind that the security and protection of the information and communications system is a key component.

D. Perakovic and L. Knapcikova (Eds.): FABULOUS 2022, LNICST 445, pp. 64–79, 2022.
https://doi.org/10.1007/978-3-031-15101-9_5

Web applications are increasingly the victims of cyber-attacks, and thus endanger their end users. Therefore, more and more businesses are hiring companies whose services are based on improving the security of information systems in order to validate and test their systems. Such companies typically have their own security assessment methodology developed and their own tools to implement their services, which ultimately results in costly service which may not be acceptable for smaller businesses in terms of cost. Therefore, it is important to consider alternative ways in which businesses could improve the security of their own applications or websites.

Web application vulnerabilities can arise from various sources, and addressing security vulnerabilities requires a certain amount of knowledge and resources to improve system security successfully. One way to improve the security of web applications is to use open source software to detect vulnerabilities. Usually such tools are free or have a reasonable price that suits smaller businesses. The main tasks of such tools are to find vulnerabilities and shortcomings in web applications, and they can provide automatic scanning or configured scanning parameters to match the needs of the web application.

The aim of this research is to determine the level of applicability of various vulnerability scanners in improving the security of web applications.

2 Previous Research

The field of web application development is fast growing. New technologies applicable in this area are developing rapidly, so there is a need to develop the security aspect of web applications and study existing vulnerabilities. Therefore, it is important to follow scientific research papers in order to be able to record trends and make recommendations for security solutions. In addition, many tools try to solve the issue of web application security by applying them. The fact is that any tool cannot completely solve the security issue of web applications, so the research can try to approximate the applicability of individual tools so that their potential can be maximized.

Statistical data collected through relevant research is an indicator of the prevalence of individual vulnerabilities in web applications and the trends that are developing in this area. Thus, such data can be used to improve the security of web applications and potentially select the appropriate tool to be used to improve security.

OWASP (The Open Web Application Security Project) is a non-profit organization that finds its purpose in improving software security. OWASP collects network security data from a variety of sources, and their OWASP Top Ten document is one of the most popular documents that acts as an indicator of the most critical security risks faced by web applications. The document is updated on average every third or fourth year, and the last edition is from 2017. According to this document, the most critical risks for web applications are Injections, Broken Authentication, Sensitive Data Exposure, XML External Entities, Broken Access Control, Security Misconfiguration, XSS - Cross Side Scripting, Insecure Deserialization, Components with Known Vulnerabilities, Insufficient Logging & Monitoring [1].

In addition to these common security risks, as many as 20% of businesses and organizations often encounter denial-of-service attacks, which was noted as a trend in a study by the European Union Agency for Cyber Security (ENISA). This research found

that in 2019, the number of attacks on web applications increased by 52%, and it was found that as much as 84% of vulnerabilities stem from security configuration errors. Figure 1 shows the data correlated with OWASP Top Ten safety risks [2].

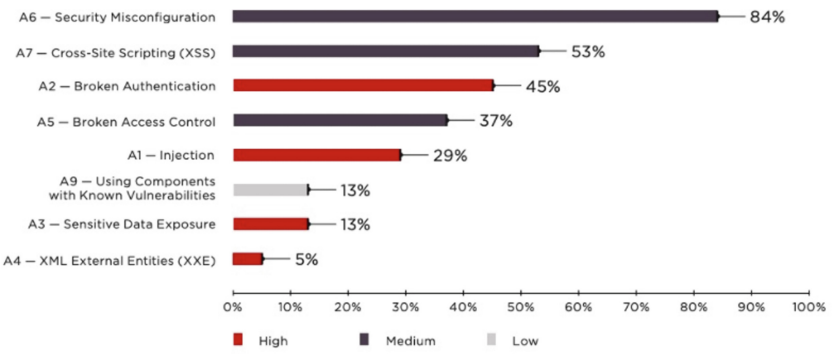

Fig.1. Statistical data correlated to OWASP top ten vulnerabilities [2]

The prevalence of denial-of-service attacks is also confirmed by the scientific paper *Cyber Security Threats and Vulnerabilities: A Systematic Mapping Study*, which maps data from scientific articles from publicly available sources. In this article, a total of 78 scientific articles that passed the final selection were analysed and it was found that a denial-of-service attack occurs in 37% of cases [3].

Security testing of web applications or websites requires pre-defined strategy and planning due to the choice of tools and constant new challenges in the security sector in the information industry. Therefore, the choice of the tool itself must correspond to the ultimate goal for which the security testing is carried out. In order to choose a tool, it is important to know the vulnerabilities and how the average web application works.

Web applications are considered any program that runs in a browser and can be accessed over the Internet. Web applications provide various functionalities to users, and due to the development of new technologies they are becoming more and more complex. Applications are usually divided into three layers. The first layer is the user, which may consist of a browser through which the user loads the web application's content. Then the second layer is the server, which serves the user and dynamically loads the content. The third layer consists of the final structure of the web application, i.e. "backend", which is used to store data [4].

Security tools that perform penetration testing are usually defined as scanners. They are considered automated tools that analyse web applications and websites, and thus try to find potential security vulnerabilities. In addition to simple search and scanning, such tools can also search for source code errors that could allow other attack vectors, such as buffer overflows. Web application scanners typically use the black box method in penetration testing [5].

In the work of *Baykar, M. Investigation and Comparison of Web Application Vulnerabilities Test tools*, six different tools were analysed. The tools consisted of commercial and open source tools, and a vulnerable application created by the Netsparker team

at http://aspnet.testsparker.com was used for testing. The tools Netsparker, Acunetix, Vega, OWASP ZAP, Wapiti and Iron WASP were tested. Comparative work analysis is divided into nine categories: tool scanning time, number of scanned vulnerabilities, tool report capabilities, module scan capabilities, manual scan capabilities, graphical interface availability, availability on specific operating systems, tool fee, and using area [5].

Looking solely at performance, the Vega tool had the fastest scan time of 45 min, while the Wapiti tool had the longest scan time of 13 h. The IronWASP tool detected the most vulnerabilities, 213, while the OWASP ZAP tool detected only 7 vulnerabilities, the least of all tools. Although free open source tools have achieved the best performance in this research, the author of the paper suggests that the choice of tools is not so simple, but that each tool has its purpose [5].

A similar suggestion is found in the work of *A. Alzahrani, A. Alqazzaz, Y. Zhu, H. Fu and N. Almashfi, "Web Application Security Tools Analysis",* where the authors tested ten types of security tools depending on their purposes. In this tool, tools that are more focused on individual areas of vulnerability, such as deficiencies in transport layer protection, information leakage vulnerabilities, XSS vulnerabilities, and injection vulnerabilities, are tested. The authors of this paper concluded that security testing of web applications or pages requires a carefully designed testing process and plan, and careful selection of tools given that each tool has advantages and disadvantages [4].

3 Related Technologies and Security Threats in Web Applications

Most of today's web applications and websites are designed through a group of technologies. Technology groups or more commonly referred to as "stacks" are groups of tools used to implement certain ideas in terms of web applications. Stacks consist of programming languages, frameworks, programming libraries, and various development tools. The choice of a particular technology group will depend on the application itself and the web application's needs, which may include its functionality, scalability, sustainability, security, etc. [6].

3.1 Web Application Architecture

The web application architecture can be divided into two parts, the client part and the server part. The client part, better known as the frontend, is responsible for presenting data to the user, receiving requests and data from the user, and directing it to the server. The server part of the web application, better known by the term backend, is credited with request processing, client management, storage and data processing. Different technologies are often used to make frontends and backends [6].

Programming languages such as HTML/HTML5, CSS, and JavaScript are used to create the front-end interface and working environments and programming libraries such as ReactJS, AngularJS, React, Node.js jQuery.

For the server, backend part, programming languages such as C#, Java, PHP, Python, Objective-C and many other programming languages are used, along with web server frameworks created through programming languages in order to enable scalability and easier application development. Examples of such frameworks are Node.js, .NET,

Django, etc. Databases such as SQL, MySQL, PostgreSQL, Oracle, MongoDB are used to store and manage data. Applications such as Apache and Nginx are used to run web servers, and cloud servers are often used for additional services such as AWS, Microsoft Azure, Google Cloud [6].

Currently the two most popular stacks are MERN and MEAN. These two technology stacks refer to a collection of technologies based on the JavaScript programming language. In the StackOverflow survey, as many as 68.62% of respondents use JavaScript in a professional environment for a long time, which makes this programming language the most used for the ninth year in a row especially in web development. For the web server framework, the choice in these technology groups is Node.js (36.19%), and for web development environments the most commonly used are React.js (41.4%), jQuery (34.52%), Angular (26, 23%) and Express.js (23.6%). MongoDB (28.03%) is the main choice of tools for working with databases in these technological groups [7].

MEAN and MERN are technology stacks that are based on JavaScript programming language. These technology stacks are both free and open source, and are used as a framework for development and design of web applications. The main benefit of using these stacks is that they allow developers to work in one programming language, applying already existing programming concepts for that language [8, 9].

These technology stacks are consisted of four key technologies and they represent layers of the stack respectfully.

MEAN stack is consisted of [8]: MongoDB - database, Express.js - web application framework, Angular.js – client-side framework, Node.js - web server.

MERN stack is consisted of [9]: MongoDB - database, Express.js - web application framework, React.js – client-side framework, Node.js - web server.

3.2 Security Aspects of Web Applications

The principles of information system security are important to maintain in order to protect the information and communication systems. This implies the application of a series of measures, standards and procedures in order to maintain the desired level of security of the information and communication system [10].

There are six basic principles of security according to [10]:

- **Confidentiality** is a characteristic of an information system that ensures the disclosure of information and data exclusively to authorized persons, entities and processes, at a defined time and by a defined procedure.
- **Integrity** of system and information means protection against intentional or accidental unauthorized modification caused by human influence or system malfunction.
- **Availability** refers to the availability of information to authorized users at the requested time and under specified conditions.
- **Authentication** is the verification of the legitimacy of a user who requests access to certain information system resources.
- **Authorization** refers to the assignment of a certain level of access rights to the system after the user has been authenticated.
- **Audit** is a process of evaluating the effectiveness of implemented security mechanisms.

Web application security threats are any possible occurrences, malicious or not, that could harm web applications in some way. Vulnerability is the weakness that makes a threat possible. Vulnerabilities can be caused by poor design, configuration error, or insecure and untested coding techniques. An attack is an action that exploits vulnerabilities in a web application.

4 Open-Source Tools Evaluation and Analysis of Applicability

Vulnerability scanners are automated tools used to detect vulnerabilities within computers, networks, or applications. Depending on the type of tool, scanners may have different functionalities and different vulnerability detection techniques. Some of the key features will be analysed later in this paper. The tools that are going to be used for applicability analysis are security scanners for dynamically testing the security of web applications. The tools to be analysed are OWASP ZAP, Vega, Arachni and Nikto. The main division of application security testing is divided into static and dynamic analysis. Static Application Security Testing (SAST) often uses various types of static analysis techniques to detect vulnerabilities, while Dynamic Application Security Testing (DAST) uses attack graph implementation techniques.

Static analysis is a fast and reliable technique. This technique focuses on the analysis of the program structure within the application, primarily on the source code itself in order to detect possible vulnerabilities within the application. This technique is considered very effective for detecting vulnerabilities. Static analysis often uses program libraries or databases for comparison with analysed code to verify program code. One of the disadvantages of such an analysis is that in the case of discovering a new unknown vulnerability it is not possible to make any comparison with the program library to verify the security of the program code [11].

An attack graph is defined as a summary of all the paths an attacker follows in the network to achieve the desired state. The desired condition may include network corruption, theft of network packets, or full access to it to determine what is happening on the network. Attack graphs help identify security vulnerabilities that lie within an application, they are usually quite large because they represent the entire application, so they are quite complex to understand and analyse. Attack graph generation is implemented within the vulnerability scanner to identify underlying application vulnerabilities, and then establish overall attack graphs to analyse the strength of an individual attack [11].

Vulnerability assessment means identifying vulnerabilities in the system before they can be exploited by anyone whose intentions are malicious. This is a proactive approach to security where vulnerabilities are detected and addressed so that no one can exploit them maliciously. Vulnerabilities do not only arise from the application, the platforms on which the application is located, operating systems, middleware that connects various parts of the application system, etc. can also be vulnerable. Therefore, it is necessary to scan the entire system including the network and software used by the application [11].

4.1 Vulnerable Application Configuration

This research aims to determine the level of applicability of various vulnerability scanners in improving the security of web applications. In order to obtain objective results

when scanning vulnerabilities on a web application, a predefined intentionally vulnerable web application OWASP Mutilidae II was used.

OWASP Mutilidae II is a free, intentionally vulnerable open source web application, which provides a simulation service of a real vulnerable web application and serves as a target of attack for potential users who want to learn more about web security. Mutilidae can be installed on Linux and Windows operating systems using one of a group of server solutions for serving at a local address, such as LAMP, WAMP, or XAMPP [12].

For the purposes of this paper, the web application will run in XAMPP on Windows 7 (x64). XAMPP is a free cross-platform server solution package with multiple open source platforms developed by Apache Friends, and consists mainly of Apache HTTP servers, a MariaDB database, and an interpreter for scripts written in the PHP and Perl programming languages [13].

To serve a vulnerable web application at a local address, it is needed to run the Apache server and MySQL database in the XAMPP user interface. After starting the server and database, it is necessary to move the directory with the source code of the Mutilidae application to the XAMPP executable directory on the local disk. In this case, it's at the C:\xampp\htdocs location in the Windows 7 operating system called mutilidae. The application can then be accessed at the local URL address 127.0.0.1/mutilidae.

4.2 OWASP Zed Attack Proxy (ZAP)

Zed Attack Proxy (ZAP) is a free open source tool used for penetration testing developed and maintained under the OWASP organization. ZAP is designed and developed specifically to test the security of web applications and be very flexible and extensible. ZAP works on the principle of man-in-the-middle proxy. In this way, ZAP intercepts and reviews requests and responses between the tester's browser and the web application, and can modify the contents of the request as needed and forward it to the destination. If there is a network proxy between the ZAP tool and the web application, then ZAP can be configured to connect to the proxy as shown in Fig. 2 [14].

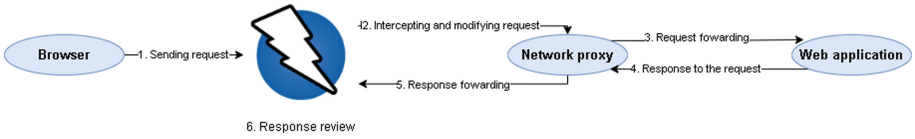

Fig. 2. Demonstration of ZAP tool operation between web application and browser [14]

ZAP offers a number of functionalities that advanced developers can use, but also beginners in security testing. It is available on every popular operating system such as Linux, Windows and OS X, and is expandable through additional features that can be downloaded freely from the ZAP Marketplace.

To start automatic scanning, user needs to press the Automated Scan button. User is then prompted to enter the desired scan target and select additional scan options. After selecting the options, it is necessary to press the Attack button to start scanning. After

running the scan, the tool first performs a passive scan, exploring possible attack pathways along with application and system information. After the passive scan is complete, the tool switches to the active scan. During the active scanning, it is possible to monitor information about the operation of the tool, and already found vulnerabilities are available via the Alerts menu in the information window. It is also possible to monitor the detailed progress of the scan by pressing the Show scan progress details button in the Active Scan menu.

Upon completing the automatic scan, the results are arranged in the Alerts menu in the information window. The total scan time was 30 min, and the ZAP tool found a total of 44 vulnerabilities in the Mutilidae application. The Alerts menu sorts vulnerabilities by severity and priority from high, medium and low levels, and information vulnerabilities that can potentially be vulnerabilities or reveal unnecessary information about the web application. Figure 3 represents the Alerts menu showing the vulnerabilities found in the Mutilidae web application.

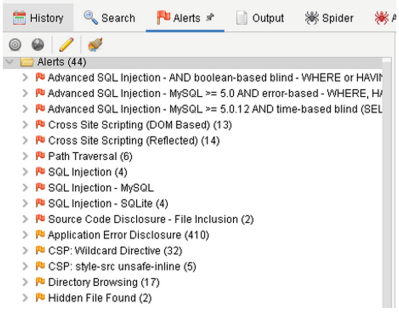

Fig. 3. Vulnerabilities shown in alerts menu after automatic scan

For a more detailed analysis of individual vulnerabilities, it is possible to obtain more information by clicking on the vulnerabilities and selecting a specific application request or response. In addition to information about the attack, the information window also contains auxiliary information for resolving the vulnerability in that case. In addition to that, additional resources are included so that web application users can investigate specific vulnerabilities and find an adequate solution for their own applications.

4.3 Vega

Vega is a platform for testing the security of web applications. Vega is a free open source tool written in the Java programming language and can run on Windows, Linux and OS X operating systems. It is based on a graphical interface and can be easily extended using modules written in Javascript. It was developed by Subgraph, a company that deals exclusively with open source information security [15].

The total scan time was 18 min, and the Vega tool found 198 different vulnerabilities within the Mutilidae application. The Vega tool also ranks vulnerabilities by severity and sorts them by priority in the Scan Alerts window as shown in Fig. 4. Within this

window it is possible to view all vulnerabilities found within the application, and the requests or responses used to find them.

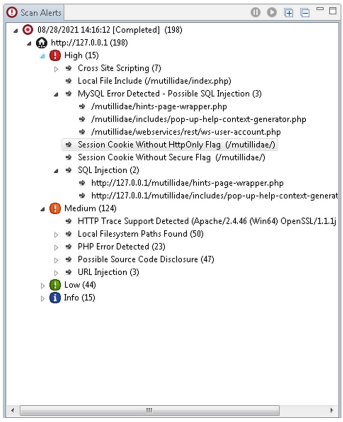

Fig. 4. Vulnerabilities shown in the *Scan Alerts* windows after automated scan

For more information on each vulnerability, user has to click on the request or vulnerability within the Scan Alerts window. The Scan Info window then provides the user with a more detailed overview of the vulnerability and information about that vulnerability and links to potential solutions to prevent such vulnerabilities. If the user wants to analyse in more detail the request or response sent that revealed the application vulnerabilities, then it is necessary to click on the request within the Request section in the Scan Info window.

4.4 Arachni

Arachni is a high-performance modular Ruby framework filled with features designed to help penetration testers and administrators assess the security of modern web applications. It is free and open source, supported on Linux, Windows and OS X operating systems and distributed through portable packages that allow immediate deployment. Implementation options are via the Ruby program library, available through Command Line Interface (CLI), the WebUI, and the distributed system using remote agents [16].

Arachni is a very flexible and versatile tool that covers many use cases for web application security. It can be used as a simple utility tool via the command-line interface for scanning, up to a global network of high-performance scanners, thanks to the REST API, integration is easy [16].

The user can start automated scanning via the header menu or by clicking the New scan button on the workspace. User is then asked to enter the URL of the scan target, and in addition, can select the scan profile, additional advanced options and scan description, and with whom to share the scan. Arachni cannot scan via a local address, and requires entering the private IP address of the computer on which the application is served. In

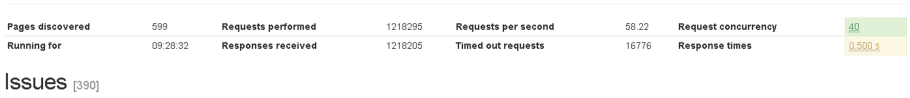

Fig. 5. Arachni scanner performance after automated scan

this case, the application will be scanned at the URL http://192.168.8.129/mutillidae/. After that it is necessary to press the Go! and the scan will begin.

During the automated scan, the user can observe the scanning process and the found vulnerabilities are available for inspection. In addition, Arachni offers a simpler display of information about scan performance such as the number of requests sent and responses received, response time, pages found, etc. Upon completing the active scan, Arachni conducts a meta-analysis of the scanned results to potentially identify possible false-positive vulnerabilities.

The total scan time was 9 h and 28 min and Arachni found a total of 390 potential vulnerabilities. In addition, Arachni discovered 599 different pages or paths within the application, and recorded a large number of requests and responses, as shown in Fig. 5.

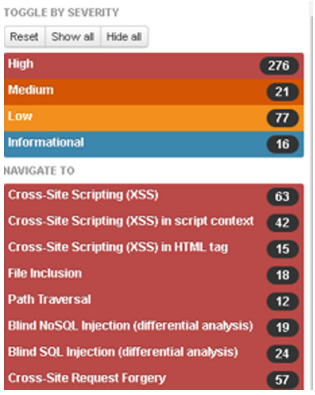

Fig. 6. Vulnerabilities shown in the Arachni *WebUI*

Similar to other tools, vulnerabilities are ranked by severity as shown in Fig. 6. Within this menu, all vulnerabilities can be searched and more detailed information about the vulnerability found can be accessed in the main menu. Within the vulnerability information window, an explanation of the vulnerability is available and the location in the web application where the vulnerability was found. For a more detailed analysis, user can click the button next to the request to determine the entire request and response process.

4.5 Nikto

Nikto is a free open source tool for scanning web server vulnerabilities. Nikto conducts extensive testing of web servers for multiple items, including more than 6,700 potentially

dangerous files/programs, checking outdated versions of over 1250 servers, and version-specific issues on more than 270 servers. It also checks for server configuration items, such as the presence of multiple index files, HTTP server options, and will try to identify installed web servers and software. Scan items and plugins are frequently updated and can be updated automatically. Nikto is written in the Perl programming language and can run on all Unix-based operating systems. Nikto is used exclusively through the command line interface [17].

The goal of the Nikto is to examine a web server to find potential problems and security vulnerabilities that include incorrect server and software configurations, default files and programs, insecure files and programs, outdated servers and programs etc. Nikto points to some information about found vulnerabilities if the report saving function is used in some of the formats available, but serves more as a tool that provides pointers to users to get a better result via manual pen-testing [17].

To scan a web server, the user needs to enter the URL of the web application, the IP address of the server, and the port number. In this case, the URL of the Mutilidae application will be scanned by entering the command nikto.pl -h https://127.0.0.1/mutilidae.

Nikto reported 166 different potential vulnerabilities in just 59 s of scanning. Scanning time on a local server is slightly faster than scanning on a web server due to the use of local resources. Nikto prints the results and information within the command-line interface, if the results are printed in one of the possible formats, then the results remain saved and can be accessed at any time in a more accessible format. Nikto prints the IP address and target port number information first, followed by the SSL information and the start time of the scan.

Nikto recognized the type of server, operating system, and programming language used within the web application. After that, Nikto prints out the vulnerabilities. When analysing the results, it is important to pay attention to the OSVDB vulnerabilities. These are already discovered vulnerabilities in similar open source technologies that can be explored within the Open Source Vulnerability Database Project repository.

5 Results Analysis and Discussion

The analysis of statistics from relevant research concluded that the number of attacks on web applications is increasing, and the most common vulnerabilities stem from security configuration errors. Four different open source tools were analyzed to examine their applicability to improve web applications' security. The research aims to determine whether open source tools can be an alternative to commercial tools or third-party penetration testing. The tool research was conducted on the vulnerable web application Mutillidae. Through the research, an automated scanning method was used to show the tools' functionalities and get a rough overview of the performance of the tools. Table 1 maps the data on the basic functionalities of these tools.

The selection of tools within this research was based on potential applications of these tools within the web application development process. The intentionally vulnerable Mutillidae application was selected as an indicator of the capabilities of these tools, and the data on the scans performed will serve as suggestions on how to implement the tools

Table 1. Basic functionalities of tools used in research

Tool name	Scan mode	Operating system	Module extensibility	User interface	Report generation
OWASP Zap	Automated and manual scanning	Linux Windows OS X	Yes	GUI	Yes
Vega	Automated scanning	Linux Windows OS X	Yes	GUI	Yes
Arachni	Automated scanning	Linux Windows OS X	Yes	WebUI	Yes
Nikto	Automated scanning	Unix-based systems	Yes	CLI	Yes

before producing the applications to improve their security. Table 2 maps the Mutillidae application scan data.

In the first case, the ZAP tool developed by the OWASP organization was used. ZAP offers automated and manual scanning capabilities. Through an automated scan on the Mutilidae application, ZAP found 44 types of different vulnerabilities in a 30-min scan. ZAP offers manual application scanning capabilities which can be useful for penetration testing in different steps within the application development process making it a very flexible tool. In addition, it offers report generation, access to in-depth information and documentation on vulnerabilities, and the tool itself, which can make it easier for developers with very little experience in security testing to make an application more secure.

Table 2. Results of the scan performed on the Mutilidae application

Tool name	Number of vulnerabilities found	Scan time	Types of vulnerabilities found	Number of vulnerabilities by severity
OWASP Zap	44	30 min	• SQL injection, • XSS injection, • Directory traversal, • Security misconfiguration, • Sensitive data disclosure, • Broken access control, • Source code and error detection	**High level** 10 **Medium level** 12 **Low level** 13 **Information level** 9

(*continued*)

Table 2. (*continued*)

Tool name	Number of vulnerabilities found	Scan time	Types of vulnerabilities found	Number of vulnerabilities by severity
Vega	198	18 min	• Security misconfiguration, • SQL injection, • URL injection, • XSS injection, • Sensitive data disclosure, • Cookie information, • Source code and error detection	**High level** 15 **Medium level** 124 **Low level** 44 **Information level** 15
Arachni	390	9 h and 28 min	• SQL injection, • XSS injection, • CSRF attacks, • OS and URL injection, • Sensitive data disclosure, • Unsafe cookies, • Security misconfiguration, • Broken access control, • Broken authentication	**High level** 267 **Medium level** 21 **Low level** 77 **Information level** 16
Nikto	166	59 s	• Insecure cookies, • Sensitive data disclosure, • Detection of unsafe configurations, • Remote command execution, • XST vulnerabilites	N/A

In the second case, the Vega tool was used. Vega has proven to be a very simple tool for dynamically scanning the security of web applications. It offers a very simple user interface and scanning which can also help developers in analyzing the security of a web application. Vega found 198 different vulnerabilities in just 18 min. The ability to scan and intercept proxy requests allows it to tactically analyze the security of web applications, which can later be used to cross-check vulnerabilities with other security tools.

In the third case, the Arachni security framework was used. Arachni offers a wide range of features and functionalities that can be implemented within the project. This makes Arachni a somewhat more complex tool to use, but offers advantages such as automation of penetration testing within the entire lifecycle of a web application. Arachni is a very thorough tool, as evidenced by the fact that the automated scan of the Mutilidae application took 9 h and 28 min, and Arachni found 390 different vulnerabilities.

In the fourth case, the Nikto tool was used to scan the web server of the Mutilidae application. Nikto is simple enough for initial use, but flexible enough for more complex scans. In this case, Nikto scanned the web server for only 59 s and found 166 potential vulnerabilities. Although Nikto offers the ability to automatically scan, the lack of a user interface makes it harder to access vulnerability information.

It is important to note that the security of a web application itself will depend on several factors. This may include the choice of technologies to be used to create the web application, the expertise and competence of the developers and other contributors, the resources available, the complexity of the project, and many other factors. When it comes to the choice of technologies, it is important to choose technologies for creating a web application that will be consistent with the size and complexity of the project and the capabilities of developers who will use these technologies. Therefore, it is important to thoroughly explore the possibilities of individual technologies in order to avoid unnecessary complications and select appropriate tools. After selecting the appropriate tools, it is important to adhere to best source code writing practices, and acceptable security coding techniques to avoid security vulnerabilities. In this paper, dynamic web application security scanners were researched, so the proposal to implement open source tools will be based on them. But along with dynamic scanners, it is essential to explore and consider the use of other types of security scanners, such as static security scanners that analyze source code.

After researching and selecting appropriate tools to be used for security testing, open source tools can be implemented in the initial steps of application design. When the application is first served to the server, Nikto can scan for vulnerabilities within the active server. Scan results can be used to manually fix vulnerabilities, and it is possible to keep documentation of scan reports after each implementation of new components within the web application to the server. Depending on the complexity of the project and application, there may be multiple servers and domains used, so Nikto security scans provide a way to implement a security audit overall servers within a project.

During web application design, ZAP and Vega can be used to scan vulnerabilities within a web application. Designing a web application is a complicated process that can involve writing a large number of components and functionalities within the application. Using the ZAP tool, it is possible to manually test new components that are implemented within the application and conclude whether there are vulnerabilities that potential attackers can use as an attack vector. This allows developers to determine if there are security vulnerabilities within the source code before the application is produced, thus giving them the ability to implement a better security coding technique. In addition, the advantage of combining multiple dynamic security scanners within a web application allows cross-analysis of scan results, which can provide a better insight into the security of the web application.

Arachni tool can serve as a security framework for full security scanning and monitoring of web application security. As a tool, Arachni offers the ability to automate the entire security system. It is also possible to adjust the performance and use the "intelligence" of the tool to obtain extensive scan results for the purpose of analyzing the security of the web application. This allows developers to perform a thorough web application security audit after each stage of the application development process. And with additional features such as writing new modules and add-ons, it allows developers to tailor security testing to the needs of the application and the entire project.

6 Conclusion

The Internet has undoubtedly become a daily occurrence for a large number of people. A large number of websites have been redesigned in recent years into interactive web applications that attract an increasing number of users. Web applications can have different purposes, but they often have in common that they request certain information from the user in order to use the web application, which later in the event of an attack can result in negative consequences for both the user and the application owner.

With the popularity of the Internet and web applications, the risk of cyber attacks has also increased. Trends show that attacks on web applications are on the rise, and the only parameters that are changing are the ways in which attackers exploit vulnerabilities in web applications. Given the already mentioned popularity, businesses are often forced to expand their business in the form of web applications. Often having limited resources the security of web applications is not a priority. Open source security tools are often free, so this paper aimed to show their applicability to improve the security of web applications.

Through this paper, four different open source tools ZAP, Vega, Arachni and Nikto were analyzed. These tools were chosen because of their availability and different ways of application, which can be a very advantageous solution for businesses with limited resources. Chapter 4 presents the tools and their scan results. Using the vulnerable application, tools were used to understand their application in improving the security of web applications. The data from the research of these tools were then mapped to better understand their applicability and propose implementing these tools for the application development process.

The final conclusion is that the security of users, their data, but also the security of the web application itself is very important, so it is necessary to protect it as much as possible. Before developing an application, it is necessary to research technologies that will suit the function and size of the application, apply appropriate security techniques for writing source code, and apply open source tools to improve security through the application development process. Developers can use security tools to implement security protections to prevent attacks and remove vulnerabilities based on scan results to improve the security of web applications.

References

1. OWASP Top Ten. https://owasp.org/www-project-top-ten/. Accessed 31 July 2021
2. ENISA Threat Landscape 2020 - Web application attacks. https://www.enisa.europa.eu/publications/web-application-attacks. Accessed 31 July 2021
3. Humayun, M., Niazi, M., Jhanjhi, N.Z., Alshayeb, M., Mahmood, S.: Cyber security threats and vulnerabilities: a systematic mapping study. Arab. J. Sci. Eng. **45**(4), 3171–3189 (2020). https://doi.org/10.1007/s13369-019-04319-2
4. Alzahrani, A., Alqazzaz, A., Zhu, Y., Fu, H., Almashfi N.: Web application security tools analysis. In: IEEE 3rd International Conference, pp. 237–242 (2017)
5. Baykara, M.: Investigation and comparison of web application vulnerabilities test tools. Int. J. Comput. Sci. Mob. Comput. (IJCSMC) **7**(12), 197–212 (2018)
6. Suschevich, A., Birukova, D.: What is a technology stack? Choosing the right tech stack for your web project. https://www.intexsoft.com/blog/post/tech-stack.html. Accessed 2 Aug 2021
7. StackOverflow: 2021 Developer Survey. https://insights.stackoverflow.com/survey/2021. Accessed 5 Aug 2021
8. MongoDB: What is MEAN Stack? https://www.mongodb.com/mean-stack. Accessed 5 Aug 2021
9. MongoDB: What is MERN Stack? https://www.mongodb.com/mern-stack. Accessed 6 Aug 2021
10. Cvitic, I., Perakovic, D., Perisa, M., Botica, M.: Definition of the IoT device classes based on network traffic flow features. In: Knapcikova, L., Balog, M., Perakovic, D., Perisa, M. (eds.) 4th EAI International Conference on Management of Manufacturing Systems. EICC, pp. 1–17. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-34272-2_1
11. Bairwa, S., Mewara, B., Gajrani, J.: Vulnerability scanners: a proactive approach to assess web application security. Int. J. Comput. Sci. Appl. **4** (2014). https://doi.org/10.5121/ijcsa.2014.411. Accessed 23 Aug 2021
12. GitHub: OWASP Mutilidae II. https://github.com/webpwnized/mutillidae. Accessed 24 Aug 2021
13. XAMPP Official Page. https://www.apachefriends.org/index.html. Accessed 24 Aug 2021
14. OWASP Zed Attack Proxy (ZAP) Official page. https://www.zaproxy.org/. Accessed 24 Aug 2021
15. Vega Official page. https://subgraph.com/vega/. Accessed 25 Aug 2021
16. Arachni Official page. https://www.arachni-scanner.com/. pristupljeno 26 Aug 2021
17. Nikto Official page. https://cirt.net/Nikto2. Accessed 30 Aug 2021