National College of Ireland

Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools

MSc Internship
MSc in Cyber Security

Lyubka Dencheva
Student ID: x20195907

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Lyubka Dencheva
Student ID:	x20195907
Programme:	MSc in Cyber Security **Year:** 2022
Module:	Internship
Supervisor:	Mark Monaghan
Submission Due Date:	15/08/2022
Project Title:	Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools

Word Count: 10712 **Page Count: 25**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Lyubka Dencheva

Date: 15th August, 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools

Lyubka Dencheva
x20195907

Abstract

In the current age of fundamental science, technological progress and the fully globalized world of technology, Cybersecurity is extremely important to ensure a high level of protection in every single area of life. In addition, the challenges, and threats to securing protection in cyberspace are constantly increasing. There are many methods to prevent Cybersecurity, one of the fundamental and basic methods is performing security testing for vulnerabilities. It is the focus of this master's thesis, which aims to examine, research, analyse, compare, and summarize the two main methods for performing this type of testing, namely Static application security testing (SAST) and Dynamic application security testing (DAST). Based on the prepared comparative analysis, the advantages, and disadvantages of both types of tools are established, which can be used as a basis for modelling a solution to achieve the most detailed, comprehensive, productive, and efficient performance of security testing for vulnerabilities. Achieving this extremely important research goal, the master's thesis supports finding a solution for a complex combination of Static and Dynamic Application Security Testing tools, through which to examine web applications for vulnerabilities and to fix them, to increase the level of protection and ensure a high, reliable and effective Cybersecurity.

1 Introduction

In current globalized world, where technology and the global network are leading in every aspect, it is crucial to ensure a high level of protection in the field of cybersecurity. The challenges and threats to providing protection in cyberspace are constantly increasing, foreshadowed by the progress of digital transformation and the global environment.
In this regard, it is the Covid-19 pandemic, during which remote work and migration to cloud solutions took place worldwide, that significantly increased the challenges of providing protection and security in the network. This is confirmed by the annual report for the past year 2021 of The European Union Agency for Cybersecurity, ENISA[1] and many other organizations. Therefore, due to the increasing number of cyber-attack attempts, it is necessary to focus on ways to prevent and increase cybersecurity.
Based on this, one of the most important measures that can be taken to reduce cyberspace risks is to use tools to identify vulnerabilities in applications, which are subsequently removed. In this way, a higher level of data protection and security in the network will be ensured.

[1] ENISA Threat Landscape 2021: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021

It is for this purpose that the current master's thesis compares two of the most popular, widely used and extremely useful methodologies used to test the security of web applications (see Figure 1). In addition, the comparative analysis focuses on the use of open-source web application penetration testing tools, in order to increase reliability and efficiency, greater scalability and flexibility, but finally the fact that it is absolutely free and easily accessible to any user of cyberspace.

Figure 1: Application security testing methodologies SAST and DAST (created by: https://www.diagrameditor.com)

1.1 Background

One of the most widely used and effective methodologies for identifying application security vulnerabilities is Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST).

The first method, SAST, is also known as white-box security testing. The tester has the source code, is familiar with the complete information about the systems, networks, architectures and all the necessary details. Based on this, the ethical hacker simulates an attack from an internal source, which is supposed to have all the necessary information to carry out the breach. Through the implementation of Static Application Security Testing, security vulnerabilities can be identified in source code, which checks code coverage and does data flow testing, path testing, loop testing. [1] [2]

Dynamic Application Security Testing (DAST), also called black-box security testing, is the exact opposite of the previous method discussed. Namely, the tester does not have knowledge and information about the tested system. Also, it does not have access to the source code, and usually uses the user interface to perform testing while the application is running. This aims to identify vulnerabilities that can be exploited by malicious individuals.[1] [2]

From a brief review of the two types of methodologies, it can be seen that they are used differently, for different purposes and to prevent attacks at very different levels. Therefore, in certain cases, their combination and their joint phased implementation would provide a higher level of protection and security against possible cyber attacks.

1.2 Research Motivation & Justification

The development of the master's thesis on the comparative analysis between SAST and DAST is extremely useful and helpful, as it provides a detailed review of the main characteristics of both types of methodologies. In this way, the differences between the methods are clearly and precisely distinguished and it is easy and precise to determine in which situations it is appropriate to use each of them.

Also, the present thesis reveals the advantages and disadvantages of the methodologies - Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST), documented and proven by tests performed and case studies. In this way, the strengths and weaknesses of both types of methodologies stand out visually and practically. Based on them, it is easy and proper to determine which type of method is most appropriate to use in a given situation.
It is also essential that this research paper fill the gap in the literature for clear, accurate, but also practical, supported by examples and demonstrations when performing a comparative analysis between SAST and DAST. This paper will also provide research and structured documentation on the differences between the two methodologies, their advantages and disadvantages, proven by real examples.
This master's thesis is an extremely detailed, structured and practice-oriented comparative analysis between two of the most used, popular and effective methodologies for performing application testing. This study was conducted in order to formulate, present and demonstrate the differences between Static Application Security Testing and Dynamic Application Security Testing, as well as to highlight their strengths and weaknesses. This will help determine the right testing methodology in certain situations, but will also demonstrate how the combined use of both methods can provide a higher level of protection against cyber attacks.

1.3 Research Questions and Hypothesis

In order to carry out a well-structured, clearly presented and detailed scope of research papers, this master's thesis is based on and guided by the following research question:

- What is the difference between the two types of testing methodologies

Based on this research question, many sub-questions arise, which further help to perform a qualitative, detailed, and useful comparative analysis between the two types of methodologies. Some of these questions are:

- What is SAST and DAST
- What each of them requires
- How they are used and in what cases
- What types of vulnerabilities they detect
- What are their strengths and weaknesses
- What are the main differences between the two methodologies and how to measure them with Benchmark

Gradual focusing, answering these questions, and applying practical research help to develop research work at a high and quality level, which is extremely useful and helpful.

2 Related Work

This chapter of the master's research project focuses on performing a critical analysis of the related work in the field of security testing types, namely Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST). It begins by analyzing the scientific articles relating to the definitions of the two types of testing, as the clarification of the concepts is of utmost importance. Then, the characteristics of the described types of application security testing are examined, as well as the comparative analysis between them. The analysis of the literature review continues with defining the differences between open-source and commercial penetration testing, as well as with a more concrete examination of the types of open-source web application penetration testing tools.

2.1 Definitions of White-box (SAST) and Black-box (DAST) penetration testing

These types of penetration testing methodologies are widely used and discussed. That is why there are many reseach papers associated with these methods. However, the definitions of the concepts set out in the research developments in a different way and scope. In the topics focused on overall penetration testing, the definitions of the two methodologies are only marked, as is the case in the article developed by Hessa Mohammed Zaher Al Shebli and Babak D. Beheshti.[3] It does not provide detailed information on the related names of the methodologies, nor does it explain their principle of operation.

This information is presented in an extremely interesting, detailed, and accessible language in research papers published in the International Journal of Embedded Systems and Applications[4] and the 36th Central American and Panama Convention[1]. They define the methodologies, but also provide explanations of how they work, as well as the synonymous names with which they are known.

There are also research papers that provide extremely useful and detailed information that accompanies the definitions of both methodologies. Such is the paper prepared by K. Krishna Mohan, A. K. Verma and A. Srividya[2], as well as the articles published in the 2nd Smart Cities Symposium [5] and the International Journal of Electronics and Computer Science Engineering[6]. They provide both theoretically oriented information, but also provide practical guidance and extremely detailed information on White-box and Black-box penetration testing.

2.2 Characteristics of Static application security testing (SAST) and Dynamic application security testing (DAST)

In most research papers, the characteristics of the two types of methodologies are examined sequentially in order to identify differences between them, and also to establish that their combined use provides a higher level of protection against cyber attacks. This is the case with the article developed by Mateo Tudela, Juan-Ramón Bermejo Higuera, Javier Bermejo Higuera, Juan-Antonio Sicilia Montalvo and Michael I. Argyros[7], as well as the work of Michael Felderer, Matthias Büchler, Martin Johns, Achim D. Brucker, Ruth Breu and Alexander Pretschner[8]. These research papers provide extremely detailed, systematic and well-founded information on the characteristics of the SAST and DAST.

There are also research papers that focus only on the explore of one of the two methodologies. Examples of similar research that focus on Static Application Security Testing have been conducted by Ronald Croft, Dominic Newlands, Ziyu Chen And M. Ali Babar[9], but also a paper published in the 41st International Conference on Software Engineering[10]. They examine only the methods of static security testing, but are presented in a larger and more comprehensive scope. Other sample research papers exist for focused research only on dynamic application security testing, such as the article developed by Diana M. Ochoa [11]. It presents in great detail the characteristics of dynamic security testing.

In addition to separate research papers on the types of security testing, there are also many that cover both types together, looking at their main characteristics, methodology and working principles. An example of such development is the scientific research paper published at the 10th Region Symposium in Indonesia[12]. It describes not only the characteristics of the two types of testing, but their design methodology is also extremely well illustrated. In this way, their principle of operation can be understood easily and comprehensibly. In addition to them, the article also provides examples that complement the overall idea of the functioning of type testing, but they are limited to their use in applications developed in a specific programming language.

As can be seen, the scientific articles that consider the types of security testing have different orientations. Some focus only on various techniques of white and black box testing, such as the article published in the International Journal of Emerging Technologies and Innovative Research[13]. In this article, the techniques are presented in a synthesized form, highlighting their main features, which provides an overview but not a detailed analysis.
On the other hand, scientific articles are not only distinguished by a certain focus, but also by the scope of the information. Some of them focus on the detailed presentation of information, while others - with a synthesized and generalized look. The first type provides extremely detailed information, such as the article published in the International Journal of Advance Research in Computer Science and Management Studies [14]. It particularly presents the characteristics of white and black box testing, the technologies and tools that are used. In addition, the advantages and disadvantages of the types of testing are also described. Thus, the scientific development provides an extremely detailed analysis and helps to gain a comprehensive understanding of the different types used for security testing in applications. The second type of scientific articles provide basic knowledge and synthesized information, such as the article by Suresh Jat and Pradeep Sharma, published in the International Journal of Scientific Research in Computer Science and Engineering[15]. It characterizes, illustrates and compares the types of security testing technologies in an extremely structured, clear and summarized form, which helps to understand the information more easily, but does not provide more extensive and detailed information. As can be seen, both types of research papers have strengths and weaknesses, and their use is determined by the purpose for which they are needed. In summary, there are research papers for each type of security testing methodology, as well as summary papers for both types. Each type of research paper has a specific scope, focus and structure. Based on them, both its advantages and disadvantages can be determined for each scientific development. Their use is determined by their specific purpose and according to the needs for which it is intended.

2.3 Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST)

The research papers that perform a comparative analysis between Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are numerous, but also different from each other in scope and focus. Some of them are exclusively focused on the theoretical aspect of analysis. Such are the research papers published in the International Journal of Computer Science and Engineering [16], the International Journal of Advanced Computer Science and Applications [17] and the International Journal of Allied Practice Research and Review[18]. They examine in detail the characteristics of the two methodologies, compare them with each other and highlight their advantages and disadvantages. In this way, these papers are extremely useful, structured, presented in detail and systematically highlight the significant differences between the two types of methodologies. Based on the above scientific developments and published articles by Thomas Hamilton [19] and Apoorva Phadke[20] on the key differences between sast and dast in a theoretical aspect, the results can be summarized in a tabular form (see Table 1) that clearly, precisely and easily readable to illustrate the essential differences in these two types of security testing.

Criteria	Static application security testing (SAST) / White Box Testing	Dynamic application security testing (DAST) / Black Box Testing
Definition	full knowledge of internal structure (frameworks, design, and implementation)	without knowledge of the internal structure (technologies and frameworks)

Alias	white box testing, structural testing, clear box testing, code-based testing, or glass box testing	black box testing, data-driven, box testing, data-, and functional testing
Test type	test from the inside out	test from the outside in
Approach	developer approach	hacker approach
Base of testing	on internal behavior	on external expectations- internal behavior is unknown
Usage	for a lower level of testing like Unit Testing, Integration testing	for higher levels of testing like System Testing, Acceptance testing
Objective	to check the quality of the code	to check functionality of the system
Programming and implementation knowledge (Skill level)	needed/required	not needed/not required
Tested by	tester and developers	end user, developer, and tester
Granularity	high	low
Time	exhaustive and time-consuming	less exhaustive and time-consuming
Requirements	source code	running application
Basis for test cases	after preparing for Detail design document	after preparing requirement specification document
Finds vulnerabilities	earlier in the SDLC	toward the end of the SDLC
Fix vulnerabilities	less expensive	more expensive
Discover run-time and environment-related issues	can’t	can
Typically scan	all kind of software	only apps like web applications and web services
Benefit	allows removing the extra lines of code, which can bring in hidden defects	well suited and efficient for large code segments

Table 1: Comparison between SAST and DAST

As can be noticed, the above table presents the differences between white and black box testing based on theoretical knowledge, but they are not substantiated with practical examples and experiments to prove and demonstrate the technical differences in the results of the two types of testing. There are many scientific developments, the scope of which is not focused on theory, but on practice. Examples of such papers are the Proceedings of the 38th IEEE International Conference on Software Engineering and Proceedings of the 2018 Building Performance Modeling Conference[21] and SimBuild jointly organized by ASHRAE and IBPSA-USA[22] as well and the paper developed by Octavio Loyola-Gonzalez[23]. They make a comparative analysis of the two types of methodologies applied in a real practical example and the conclusions are based on them. However, with them, the practical experiments are narrowly specialized in a certain direction or to specific instruments, and there is no real comparison of the basic characteristics of white box testing and black box testing.

Therefore, both theoretically oriented research papers and practical ones have different advantages and benefits. The best option is to combine both types, as this gives a comprehensive and in-depth view of both types of methodologies, their strengths and weaknesses, and an idea of when and how to apply them to be most effective in providing prevention against cyberattacks.
Consequently, it is extremely important to analyze and compare the results of performing security tests with SAST and DAST. Accordingly current master's thesis specializes in analyzing the these two types of testing methodologies to highlight the significant differences in results obtained when using them for web application vulnerability analysis.

2.4 Benchmark

The purpose to compare these two methods of security testing and their effectiveness, it is necessary to analyse the results obtained after using SAST and DAST tools. Therefore, one of the techniques recommended for this purpose is the use of Benchmark. The Benchmark is used to execute performance evaluation of web vulnerability scanners, as clearly explained in a publication from the 26th International Conference on Systems Engineering[24]. It examines, explains, and compares various benchmark tools, as well as performs and analyses specific samples and experiments. Articles published in the International Journal of Computer Applications[25] and in the Security and Communication Networks[26] are also in a similar direction. They perform an extremely detailed comparative analysis of various benchmark tools, but these research papers do not compare specific results of using SAST and DAST tools, as they are specialized and focused on different orientations. Therefore, based on the above research papers, this master thesis aims to use a benchmark in analyzing and compare black box security testing and white box security testing, thus further developing and improving the scope, specificity and specialization of this field.

2.5 Differences between open-source and commercial penetration testing tools

There are many articles, publications and research papers in the public network that look at open-source and commercial penetration testing tools and the significant differences between them. It is extremely important that the comparative analysis between the two types of tools is presented in a systematic, clear and precise way, so that a parallel can be drawn between them quickly and easily. Research papers that are constructed in this way have been published in the International Journal of Scientific & Engineering Research[27] and International Journal of Computer Applications[28]. They are extremely well structured, show in tabular form the differences between the two types of tools, cover all the necessary information on the topic and can be extremely useful and very helpful.

2.6 Open-source web application penetration testing tools

The current master's thesis has chosen the use of open-source testing software, because this type of testing tools are widely available, free, flexible, provides customization and provides speed and security in testing. Like any other thing, open-source testing tools have weaknesses, but for this paper we will focus on the benefits it offers, as the cost-effectiveness and affordability that these testing tools provide is crucial. An article published in the Journal of Interaction Science[29] discusses in detail the advantages and limitations of open-source testing tools, but this paper does not specifically address the use of these tools for application testing. This is the focus of the research paper published on the 36th Annual Computer Software and

Applications Conference Workshops[30]. It examines the application of open-source tools in an extremely detailed, systematic and structured way in order to perform security testing.

2.7 Conclusion

The scientific research papers that cover the topics of security testing are extremely numerous and diverse, but they are focused on individual components or are specified in a certain direction. Therefore, a multi-aspect collaborative research paper has not been developed that examines and analyses these types of security testing in extreme detail. It is for this reason that the current topic of the master's thesis was chosen to be developed. In this research paper, testing with two types of tools is performed on one project and the results obtained are gradually analysed through a special benchmark. They mainly focus on calculating the accuracy, range, and accuracy of the results. In this way, the differences between SAST and DAST instruments are clearly and accurately distinguished.
Therefore, this master's thesis aims to summarize the available information, develop it, and bring together many other components for testing the security of web applications. In this way, a multi-component, complex and extremely well-defined scientific work is built, which includes many components combined, used, and practically applied in combination.

3 Research Methodology

In this chapter of the master's thesis, the methodology is detailed and discussed which has been applied to perform comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools. The methodology is presented and illustrated in Figure 2 below.

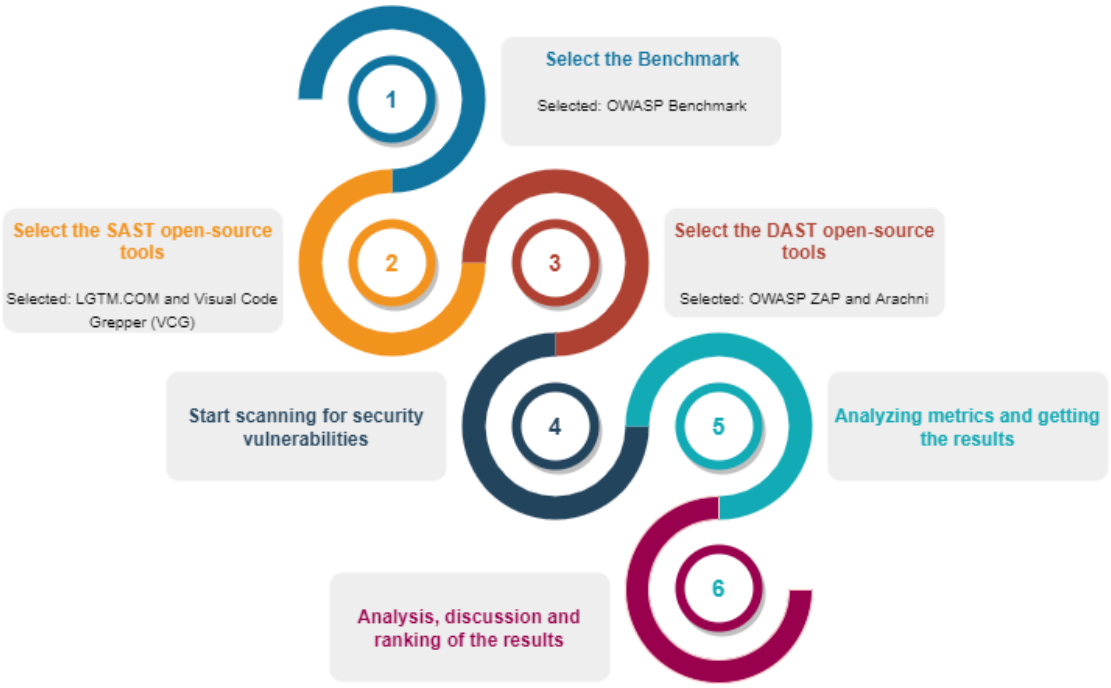

Figure 2: Methodology (created by: https://www.diagrameditor.com/)

3.1 Select the Benchmark

The methodology applied in current master's thesis begins with the selection of a benchmark by which to compare the effectiveness of SAST and DAST security tools. Based on a research paper by Balume Mburano and Weisheng Si[24], which extremely detailed and precisely

evaluates the two main types of benchmarks - Web Application Vulnerability Security Evaluation Project (WAVSEP) and Open Web Application Security Project (OWASP). As a result of the research, it was found that there are significant differences in the performance of the benchmark. These are reflected in the fact that after the trials and comparisons, the results of the WAVSEP benchmark are much higher than those of the OWASP benchmark. Therefore, the OWASP benchmark is more challenging than the WAVSEP benchmark, which makes it a more preferred and more functional choice, as confirmed by the authors of the research paper. Also, other scholarly authors examine and compare these two types of benchmarks as well. An example of such a research paper is the published paper of the 42nd Annual Computer Software and Applications Conference (COMPSAC)[31]. It is developed with a slightly different focus, but also highlights the advantages of choosing the OWASP benchmark, which can fully satisfy the needs and encourage its use for various specific research papers and scientific works where it is necessary to perform an evaluation and comparative analysis.
Referring to the above research papers and the arguments discussed, the OWASP benchmark is selected in the present thesis. It stands out for its outstanding functionality, accuracy and reliability, but also of utmost importance is the fact that it is a fully functional open-source web application that provides numerous benefits.
The main purpose of the OWASP benchmark, as defined by its official page[2], is to evaluate the accuracy, coverage and speed of software vulnerability detection tools. These characteristics are extremely important, because based on them, a comparison of the used testing tools is made, which is the main goal of this master's thesis. Therefore, for research purposes, a benchmark that contains exploitable test cases that can be analyzed by both SAST and DAST tools is required. It is this functionality that OWASP Benchmark has, but it also includes multiple scorecard generators that are extremely useful for purpose of comparative analysis because they visually illustrate the results obtained.

3.2 Select the SAST open-source tools

In the next step of the methodology of this research paper, a selection of SAST tools is carried out, whose effectiveness will be investigated. These tools scan the source code of applications and find security vulnerabilities in it. There are numerous such tools, both commercial and open source. It is the second type, open source, that are used in this project, because they not only allow free research to be done, but also provide many advantages and functionalities.The first selected tool is Visual Code Grepper (VCG), referring to the Tom Carr's article[32] and scientific paper of the 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)[33]. They define VCG's ability to provide optimal code security, since when performing a source code scan, the tool has the functionality to detect the most popular and common security vulnerabilities according to the OWASP classification. Other outstanding advantages of VCG is that it offers customization for defining scan areas, extremely user-friendly interface for performing vulnerability scans, but also presents the output in an easy, clear and understandable form accompanied by an illustrated diagram with the reported results.
The second SAST tool that was chosen to perform a vulnerability scan and subsequent analysis and comparison of the results is LGTM.COM. It is an automated tool for reviewing source code and looking for security vulnerabilities, and it is completely free for all open-source projects. Based on Eyal Katz's article[34], the tool not only checks for Common Vulnerabilities and Exposures (CVE), but it is also backed by the trusted code query language- CodeQL, which is supported by Microsoft and Google. Also, the LGTM.COM has functionality that presents

[2] OWASP Foundation, OWASP Benchmark: https://owasp.org/www-project-benchmark/

the information and the obtained results in an exceptional way, which allows the subsequent processing of the data in order to perform a comparative analysis, which is the main focus of this research paper.

3.3 Select the DAST open-source tools

After the selection of SAST tools, it is necessary to select DAST tools, the effectiveness of which should be examined and compared with the previous ones. With the help of dynamic testing tools, active applications are examined by performing an external attack. The goal is to help detect and identify security vulnerabilities and subsequently remediate them.

In the present research work, two DAST tools were selected, used and investigated. The first selected is one of the most popular tools of this type, namely OWASP ZAP. Based on an article by Jakub Woźniak[35], this tool has many advantages one of which is that it is open source and a multifunctional tool for testing web application security. Also, it has many tools that automate the scanning process and thus it is possible to detect as many errors as possible. These advantages are also confirmed by a penetration tool analysis published in the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)[36], which documents that OWASP ZAP manages to detect more serious threats that are missed from similar instruments.

The second selected DAST tool is Arachni, the advantages of which are systematically presented in the research paper of Felix Brombacher[37]. It defines the most valuable features that this tool has, some of which are: free and open source tool, but it also integrates extremely easily with most platforms and allows many different checks for vulnerability analysis.Therefore, this tool is a highly automated, distributed penetration testing platform with multiple features. In this way, Arachni provides the highest levels of resistance, accuracy and reliability. All these advantages that this tool provides are examined and discussed in detail in the article by Tasos Laskos[38], and other useful features that Arachni possesses are also added. Therefore, this tool is highly suitable for the purposes of this research paper.

3.4 Start Testing

Once the SAST and DAST tools, whose effectiveness will be investigated, have already been selected, it is necessary to start their security testing. For this purpose, the selected tools are installed and configured, after which the vulnerability scan itself is launched, specifying the OWASP Benchmark itself. When the scan is finished, the tools that performed it download a document with the final results in xml format, as it is accepted by the selected benchmark. The generated files are placed in the OWASP Benchmark directory and the generation of scorecards is started to summarize and calculate the necessary metrics for subsequent comparative analysis.

3.5 Analysing metrics and getting the results

The methodology used in the master's thesis continues with the determination of the criteria and metrics by which the comparison of the SAST and DAST tools will be carried out. They are based on the chosen OWASP Benchmark used because it has functionality to define, calculate and present extremely important indicators and metrics. Through them, a complete picture of the effectiveness of the testing tools is obtained and by comparing them, a detailed, correct, and reliable comparative analysis can be done between the used static and dynamic application security testing tools.

The current project uses indicators and metrics to evaluate scanners based on the already selected benchmark. These metrics are automatically calculated and visualized using the OWASP benchmark's score carding functionality. The metrics and indicators that are

calculated and presented using the benchmark are detailed analysed and described in a paper published at the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks[39]. In it, the information is extremely detailed and extensive, but a more specific, summarized, and systematized analysis of the metrics used is presented in the research work of Balume Mburano and Weisheng Si[24]. Based on the above documents, the metrics and indicators used in the master's thesis are presented in tabular form (see Table 2), which provides information about each indicator and its calculation method.

Name	**Formula**	**Description**	**Description in OWASP Benchmark**
True Positives (TP)	automatically generated after the scan	actual vulnerability that needs to be fixed	Tests with real vulnerabilities that were correctly reported as vulnerable by the tool
False Negatives (FN)		actual vulnerability is not detected by the tool -> serious issue	Tests with real vulnerabilities that were not correctly reported as vulnerable by the tool
True Negatives (TN)		code is a non-flawed code -> no need action	Tests with fake vulnerabilities that were correctly not reported as vulnerable by the tool
False Positives (FP)		non-flawed code is reported as a flaw -> this warning should be ignored	Tests with fake vulnerabilities that were incorrectly reported as vulnerable by the tool
True Positive Rate (TPR)	$\frac{TP\ (True\ Positive)}{TP\ (True\ Positive)+FN(False\ Negative)}$	Proportion of positive cases that are correctly classified as positive. Also called Recall or sensitivity.	The rate at which the tool correctly reports real vulnerabilities
False Positive Rate (FPR)	$\frac{FP\ (False\ Positive)}{FP\ (False\ Positive)+TN(True\ Negative)}$	Represents the ratio of negatives that is incorrectly classifies as positives. Also called fall-out.	The rate at which the tool incorrectly reports fake vulnerabilities as real

Table 2: Summary of the used metrics

Every single indicator researched is of utmost importance for determining the effectiveness of the tools used, as their assessment is based on True Positive Rate (TPR) and False Positive Rate (FPR). These indicators are particularly important, as they not only determine the time and ability to detect vulnerabilities but are also a major part of determining the accuracy of

security analysis tools using Youden's Index, which is part of the scorecards prepared by the OWASP Benchmark. The above index is calculated according to the following formula:

$$Youden\ index\ (Score) = TPR\ (True\ Positive\ Rate) - FPR\ (False\ Postive\ Rate)$$

After performing the calculation of the formula, the result obtained is extremely important in determining the accuracy of the SAST and DAST tools used. This is a percentage value that determines how accurate the results obtained from the scan performed are. The analysis of a sample resulting index is detailed and explained in an article by Fabian Yamaguchi[40]. It states that the higher the percentage of this index, the more accurate the scan result.
In addition, the obtained results are generated as a score card, which contains a graph that visually depicts the results and helps to make a comparative analysis between used secure testing tools easier and more accurate.
All the above and analysed metrics are automatically generated by OWASP Benchmark based on the provided results from the performed scans of white box and black box security testing tools. For each tool, an html file is generated with the final examined results, containing general statistics and details of the scans. In addition to them, this document contains information on the above-mentioned indicators and metrics, which are the main basis and necessity for performing a qualitative and accurate comparative analysis between SAST and DAST tools.

3.6 Analysis, discussion and ranking of the results

In previous phases of this methodology, a scorecard was generated, metrics were analysed, and results were obtained to be used for comparative analysis. Based on them, a comparison of the obtained results is made, and a diagram is developed using MS Excel to visually illustrate the differences in the selected tools. In addition, the present research paper also carries out subsequent analyses, comparisons, summaries, and final conclusions to complete the overall comparative analysis of SAST and DAST instruments and highlight the essential differences between the two types of methodologies for security testing in web applications.
In order to carry out a complete, accurate and accurate analysis, parameters for comparison have been selected, which are extremely important not only for determining the characteristics of the tools used, but also for analysing their overall efficiency and productivity. Based on them, the master's thesis carried out a complex comparative analysis of static and dynamic tools for testing the security of applications, the performance of which was studied according to the parameters presented in Table 3:

№	Parameter	Description
1	Type of security testing	SAST or DAST tools (white box or black box security testing)
2	Cost of the tool	Commercial or open-source tools
3	Type of interface	Graphic user interface (GUI) based or Command line interface (CLI) tools
4	Programming Languages Support	Which programming languages are supported by selected tools
5	Scan time required (min)	How long the security vulnerability scan takes
6	Numbers of detected security alerts	What is the number of detected alerts after the scan
7	Total test cases	what is the total number of investigated test cases

8	True positive, false positive, true negative, false negative	What are the obtained values for these basic parameters
9	True positive rate and false positive rate (%)	What are the values obtained for these rates
10	Score (%)	What is the value of the Score, which is the Youden's Index, representing the accuracy of security analysis tools

Table 3: Parameters for comparative analysis

The above parameters are used as the criteria by which the comparative analysis of SAST and DAST tools is performed. They are visualized in tabular form to provide an easier, structured, and visible mapping of the parameters to facilitate the analysis of the obtained results generated based on the performed security vulnerability scans and the subsequent generation of scorecards to be used for accurate, clear, and accurate comparative analysis of white and black box security analysis tools.

4 Design Specification

4.1 Oracle VM VirtualBox and Kali Linux

Launching the research work begins with providing a Linus-based operating system that is needed to run the project. For this purpose, open-source software - Oracle VM VirtualBox Manager version 6.1 is installed and configured on an available Windows operating system machine. It allows the use of multiple guest operating systems on one host, which is necessary for the master thesis. A Debian Linux-based Kali Linux distribution was selected for it, which is open source, contains the possibility of built-in functionalities, and is penetration-oriented, making it the most suitable for the current project. It uses Kali Linux version 5.9, which is also installed as a virtual machine in VirtualBox and is configured with the following basic parameters: 4 processors, 7451 MB base memory and 16 MB video memory. Then, the Linux virtual machine is started and the necessary components and software products that will be used to perform the research are provided.

4.2 OWASP Benchmark

In order to perform a precise and accurate comparative analysis of the used SAST and DAST tools, the open source tool OWASP Benchmark Project is used. It is designed to test and investigate the speed and accuracy of vulnerability detection tools, which is a major goal in this research paper. For this purpose, the OWASP Benchmark files are downloaded and uploaded to the virtual machine. Then, from the Linux console, the necessary files are executed and the benchmark is started. It is positioned on localhost and loaded into the web browser from which it can be accessed.

4.3 SAST tools → LGTM.COM and Visual Code Grepper (VCG)

The other necessary resource for carrying out the research is the already selected SAST tools. The first of these, LGTM.COM , is an online tool that is a code analysis platform for identifying vulnerabilities and preventing them from reaching production. In it, the OWASP Benchmark project is loaded and the scan starts. Once the scan is complete, its results are generated and visualized. It also provides the possibility to export them, thus allowing them to be loaded and analyzed in the selected benchmark.
The second of the SAST tools used for this research is the Visual Code Grepper (VCG). Its installation files are downloaded to the host Windows OS machine and the tool is installed.

After launching the tool, the programming language in which the target files are written is configured, then the files are loaded and the scan begins. After its completion, the results are visualized and the possibility to download them in xml format is provided, which can be processed by the selected OWASP Benchmark.

4.4 DAST tools → OWASP ZAP and Arachni

For the purpose of the master's thesis, in addition to SAST tools, DAST tools are also needed. The first selected is OWASP ZAP v 2.9.0, which is an integrated penetration testing tool for finding vulnerabilities in web applications. It is installed and launched from the Kali Linux console, then set to scan the local host on which the benchmark is positioned. After the scan is completed, it is possible to download the report in xml format to be processed by the OWASP Benchmark.
The second used Dynamic Testing tool is Arachni, which is installed and launched from the Kali Linux console. After that, this tool and its admin panel are accessible through the web browser, from where the vulnerability scan target is set. When the scan is complete, the results can be downloaded into a format supported by the OWASP Benchmark and subsequently analyzed by it for performing a comparative analysis between the SAST and DAST tools.

5 Implementation

In the previous section, the Design Specifications of the used products were examined and discussed, and based on them, this section presents their use and implementation. This complex process is presented in Figure 3.

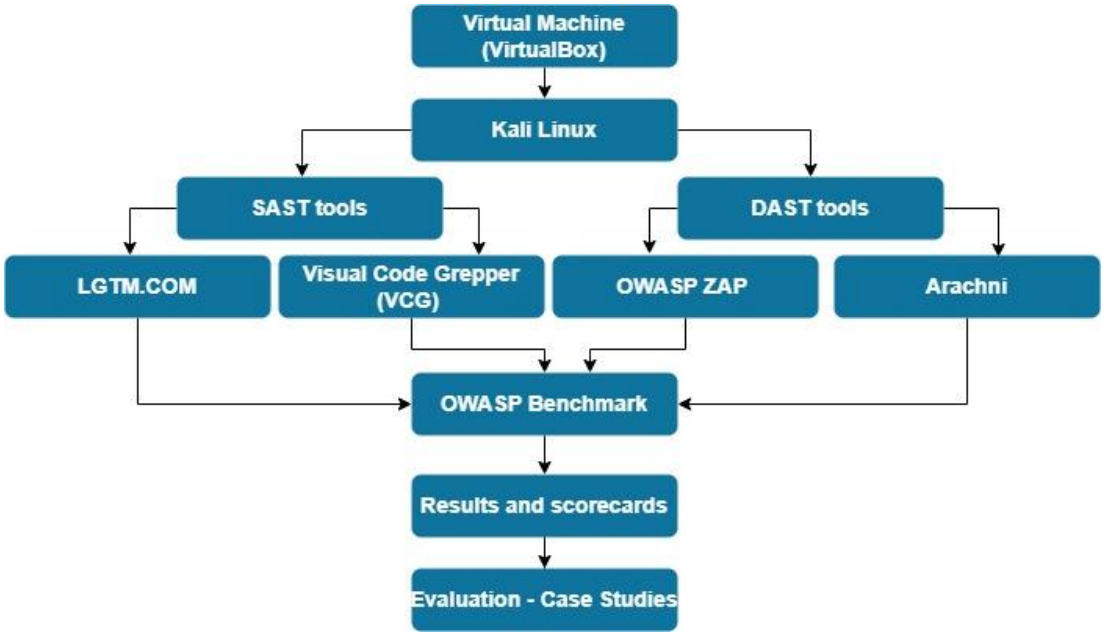

Figure 3: Implementation process (created by: https://www.diagrameditor.com/)

5.1 Environment Setup

Step 1: The primary host machine is a Windows OS, but since multiple scripts are running, a Linux operating system would provide more flexibility and functionality. That is why the implementation process starts with the installation and configuration of the Kali Linux virtual machine, through VirtualBox. For this purpose, a new virtual machine is created, on which the

installation files and its main parameters are set. After starting and configuring the operation system, the Kali Linux virtual machine is ready for use.
Step 2: In the Kali Linux virtual machine, the OWASP Benchmark files are placed, as well as the OWASP ZAP and Arachni tools are installed using commands in the terminal. As for LGTM.COM tool no installation is required as it is used online on its official site. The other scanning tool Visual Code Grepper (VCG) runs on the host OS Windows machine for faster and more convenient scanning.
Step 3: The next step is a scan for security vulnerabilities by using SAST tools. The selected tools load the OWASP Benchmark files and start the source code scans. Once completed, the results are exported in xml format by Visual Code Grepper (VCG) and in sarif format by LGTM.COM, which are accepted by the Benchmark. The generated files are placed in its directory, in the results folder.
Step 4: Scanning for security vulnerabilities followed, but this time using Dynamic Application Security Testing tools. The first step is to run the OWASP Benchmark, which will be scanned by the tools. It is started by sequentially executing the buildDockerImage.sh and runDockerImage.sh files, using the Linux console. It is accessed through the web browser at the localhost URL on port 8443 and the benchmark directory as follows: https://localhost:8443/benchmark/.
Step 5: Once the Benchmark is running, its URL address is set in the attack URL field in the already installed and configured OWASP ZAP, and the scan is started. In the other DAST tool Arachni, the situation is similar, but with the following differences: access to the tool and the admin panel is done through the web browser at the address http://localhost:9292 and login with the basic username and password. After that, the URL of the benchmark is set for the target address, but with the difference that instead of localhost, the local IP address is written to prevent conflicts when performing scans.
After the security vulnerability scans are completed, the performing results are exported in xml format and placed in the results folder of the OWASP Benchmark directory, as was the process and results with the SAST tools.

5.2 Results and scorecards setup

In this step of the implementation, a script is run from the console to generate scorecards based on the already placed files with the results obtained during the scanning with SAST and DAST tools. This is accomplished by executing the createScorecards.sh file from the OWASP Benchmark directory. After its execution, scorecards are generated in html format, which contain detailed information about the obtained results, as well as illustrating them through graphics. In addition to this file, the script generate two more files as follow: the first one is a png image of the aforementioned graph, and the other is a csv file containing the amount of data from the tests and analyses. Therefore, based on the generated scorecards, the comparative analysis between Static and Dynamic Application Security Testing tools is performed.

6 Evaluation

This chapter presents and analyses the results of the scan for security vulnerabilities of OWASP Benchmark through open-source Static and Dynamic Application Security Testing tools. Their results were analysed separately, after which a comprehensive comparative analysis was carried out between the results obtained from the scanning of the selected tolls and the prepared scorecards.
After performing a scan for security vulnerabilities, scorecards are generated by OWASP Benchmark, which present and visualize the obtained results in a form that allows easy, precise, and accurate analysis. For this purpose, the methodology described in the research paper is

used to analyse the results. Also, MS Excel is used to visualize the obtained parameters and present them graphically. Through it, diagrams are created that show the main parameters needed to perform the comparative analysis. In this way, the essential differences between the tools used can be extremely clearly visualized and highlighted.

6.1 Case Study 1: Analysing the results of the SAST tools

6.1.1. LGTM.COM

Based on the generated scorecard (see Figure 4) and applying the described analysis methodology, the obtained results can be summarized in the template table described in the methodology chapter as follows:

№	Parameter	Results
1	Type of security testing	SAST (white box security testing)
2	Cost of the tool	Open-source tool
3	Type of interface	Graphic user interface (GUI)
4	Programming Languages Support	C and C++, C#, Go, Java, JavaScript/TypeScript, and Python
5	Scan time required (min)	obtained at the moment, because lgtm.com has a library of already analyzed open-source projects, among which is the OWASP Benchmark Java
6	Numbers of detected security alerts	3042 security alerts, from which 2641 errors and 401 warnings
7	Total test cases	2740
8	True positive, false positive, true negative, false negative	True positive (TP) = 934 False positive (FP) = 457 True negative (TN) = 868 False negative (FN) = 481
9	True positive rate and false positive rate (%)	True Positive Rate (TPR) = 69.08% False Positive Rate (FPR) = 36.04%
10	Score (%)	Score= TPR- FPR = 33.03%

Table 4: LGTM.COM results

Based on the above characteristics, it is visible that the LGTM.COM tool has a high level with correctly reports real vulnerabilities, almost 70% out of a maximum of 100, which makes it extremely effective. However, the tool also scores poorly on the functionality for incorrectly reports fake vulnerabilities as real. It is quite high, 36.04%, which shows that LGTM.COM does not have that good performance in this direction. The difference from the parameters analysed above provides information about the Youden's Index, which evaluates the accuracy of the tool used. In this case, it is rated 33.03%, indicating that the level of accuracy is slightly above the average commercial level, therefore this tool is an acceptable choice for performing analysis.

Analysing in more detail the generated scorecard for LGTM.COM for each category of security vulnerabilities, it is noticed that the tool has 100% efficiency for detecting Insecure Cookies and Weak Encryption Algorithm. At the other extreme, with 0%, the ineffective detection of securities fraud is noted in the following categories: Trust Boundary, Weak Hashing Algorithm and Weak Randomness.

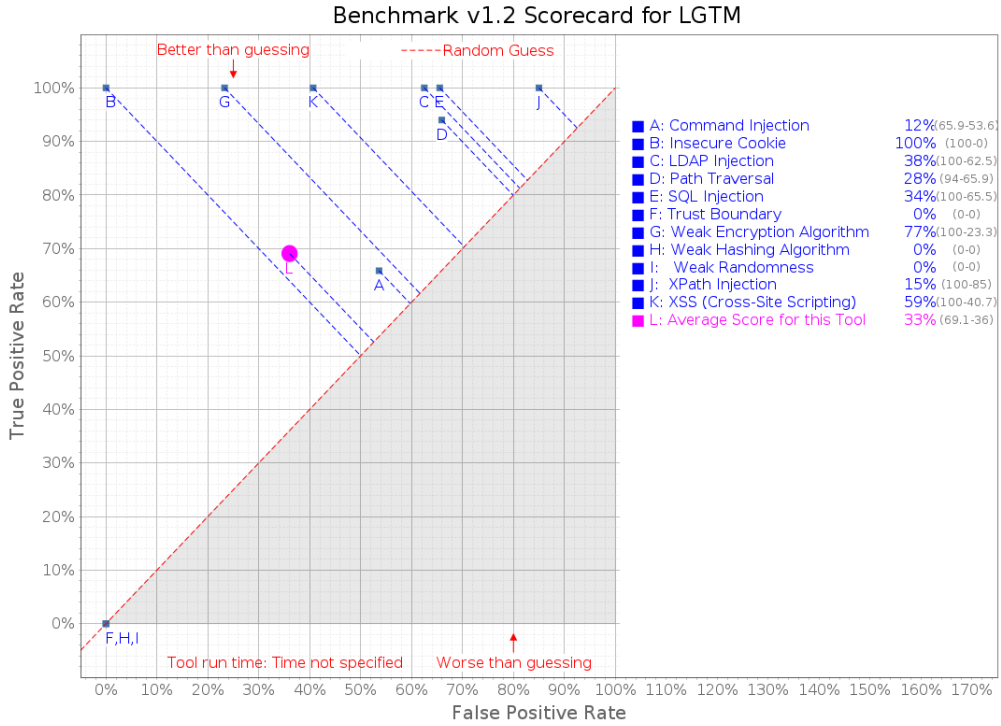

Figure 4: LGTM.COM scorecard

6.1.2. Visual Code Grepper (VCG)

The obtained results of the performed security vulnerability scan can be summarized in the table described in the methodology of the research paper as follows (see Table 5):

№	Parameter	Results
1	Type of security testing	SAST (white box security testing)
2	Cost of the tool	Open-source tool
3	Type of interface	Graphic user interface (GUI)
4	Programming Languages Support	C/C++, C#, VB, PHP, Java, and PL/SQL
5	Scan time required (min)	performed extremely quickly, within less than 5 min
6	Numbers of detected security alerts	3912 security alerts, from which 230 potentially dangerous code
7	Total test cases	2740
8	True positive, false positive, true negative, false negative	True positive (TP) = 81 False positive (FP) = 2 True negative (TN) = 1323 False negative (FN) = 1334
9	True positive rate and false positive rate (%)	True Positive Rate (TPR) = 7.94 % False Positive Rate (FPR) = 0.08 %
10	Score (%)	Score= TPR- FPR = 7.86 %

Table 5: Visual Code Grepper (VCG) results

Therefore, as can be seen from the table, the 6.Visual Code Grepper (VCG) generates a significant number of False Negative (FN) and True Negative (TN), which lowers its productivity. Accordingly, this also affects its final efficiency rating, which is below 1% as measured by Youden's Index. Analyzing in more detail the parameters for each type of security

vulnerability, it was found that the VCG tool has a rather low efficiency for all the categories covered. For a large part of them, the tool manages to detect alerts in an extremely low percentage between 1 and 4%, and for the Insecure Cookie category it even has a negative value. Therefore, the SAST tool Visual Code Grepper does not have a high efficiency for detecting security vulnerabilities.

6.2 Case Study 2: Analysing the results of the DAST tools

6.2.1 Arachni

In a similar way, as in the previous chapter, the results obtained from this tool are also presented. The aim is to provide a better visual representation and easy comparison of the parameters presented in Table 6 below.

№	Parameter	Results
1	Type of security testing	DAST (black box security testing)
2	Cost of the tool	Open-source tool
3	Type of interface	Graphic user interface (GUI)
4	Programming Languages Support	PHP, ASP, ASPX, Java, Python and Ruby
5	Scan time required (min)	extremely slow, more than 18 hours
6	Numbers of detected security alerts	71 security issues, from which 43 are of high threat level
7	Total test cases	2740
8	True positive, false positive, true negative, false negative	True positive (TP) = 43 False positive (FP) = 0 True negative (TN) = 1325 False negative (FN) = 1372
9	True positive rate and false positive rate (%)	True Positive Rate (TPR) = 3.10 % False Positive Rate (FPR) = 0.00 %
10	Score (%)	Score= TPR- FPR = 3.10 %

Table 6: Arachni results

As can be seen from the tabular view of the obtained results, the number of detected security alerts is not large, but on the other hand, no False Positives (FP) were reported, which is extremely important for efficiency. On the other hand, the number of False Negative (FN) is quite high, indicating that Arachni tool reported false vulnerabilities incorrectly as real ones. Also, analysing the main score of the tool, which determines its effectiveness, it is noticed that it has a rather low value - a little over 3%. This stems from the fact that in all scanned categories the rate was reported as 0%, except for Command Injection. It has a value of almost 35%, which makes Arachni tool effective for detecting this type of security vulnerabilities.

6.2.2 OWASP ZAP

The results obtained from the performed scan for security vulnerabilities through OWASP ZAP is presented in tabular form as follows (see Table 7):

№	Parameter	Results
1	Type of security testing	DAST (black box security testing)
2	Cost of the tool	Open-source tool
3	Type of interface	Graphic user interface (GUI)
4	Programming Languages Support	supports multiple languages including Java, JavaScript, Zest, Python, Ruby, C# etc.

5	Scan time required (min)	medium speed, about 45 minutes
6	Numbers of detected security alerts	577 security alerts
7	Total test cases	2740
8	True positive, false positive, true negative, false negative	True positive (TP) = 81 False positive (FP) = 2 True negative (TN) = 1323 False negative (FN) = 1334
9	True positive rate and false positive rate (%)	True Positive Rate (TPR) = 7.93% False Positive Rate (FPR) = 0.08 %
10	Score (%)	Score= TPR- FPR = 7.86 %

Table 7: OWASP ZAP results

Based on the results presented above, OWASP ZAP, like the other DAST tool, reports an extremely low False Positive (FP) score close to 0%. This shows a good performance of the tool, but it is affected by the high number of False Negative (FN) and finally its overall performance score is estimated at 7.86%. Therefore, it is significantly reduced by the low level and even zero values of the True Positive Rate (TPR) parameter, which are reported for most of the investigated categories. The only exceptions are Command Injection and Insecure Cookie, which register quite good results and opportunities to detect security vulnerabilities by using OWASP ZAP.

6.3 Discussion: Comparative analysis between SAST and DAST tools

Based on the above-presented and analysed tabular data for the used SAST and DAST tools, the comparative analysis between them is carried out for each of the criteria set in the methodology of the research work. The obtained analyzed results and summaries are grouped in the two sections - similarities and differences. In this way, the components in the comparative analysis of Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) tools are clearly highlighted, differentiated and qualified.

6.3.1 Similarities

1) Cost of the tool

All tools used and analyzed are open source, as they are not only free, but also provide higher functionality, flexibility and independence when performing security vulnerability scans.

2) Type of interface

Both SAST and DAST tools have a Graphic User Interface (GUI). It provides an opportunity for extremely easy, convenient, understandable and multitasking scanning for security vulnerabilities. It also allows provides a user-friendly environment through which the configuration and management of tests can be easily performed.

3) Programming Languages Support

All selected and used tools support multiple programming languages, but the one required to perform the research in the master's thesis is Java. It is supported by all four selected security testing tools.

4) Total test cases

The sum of the performed test cases is the same for all tools used, regardless of whether they are for static or dynamic application security testing. This is based on the fact that all of them perform vulnerability scans on the same files and directories from the selected OWASP benchmark.

6.3.2 Differences

1) Type of security testing

The selected and research tools are the two main categories: Static application security testing (SAST), known as White Box Testing tools, and Dynamic application security testing (DAST), known as Black Box Testing tools. The first type includes LGTM.COM and Visual Code Grepper (VCG), analyzing the source code for security vulnerabilities. The other type of tools perform functional testing scans for security vulnerabilities of active web applications. Therefore, the DAST tools used and analyzed in the master's thesis are Arachni and OWASP ZAP.

2) Scan time required (min)

The process of scanning for security vulnerabilities performed by testing tools requires technology time for their performance. Based on the above data, it is found that Static Application Security Testing tools, which analyzes the source code, requires slightly less time to scan for security vulnerabilities, compared to DAST. It takes significantly longer, and in the particular study the minimum difference in scan time between the two types of tools used was about 40 minutes. Therefore, SAST tools feature a higher scanning speed compared to Dynamic Application Security Testing tools, which is visualized in Figure 5.

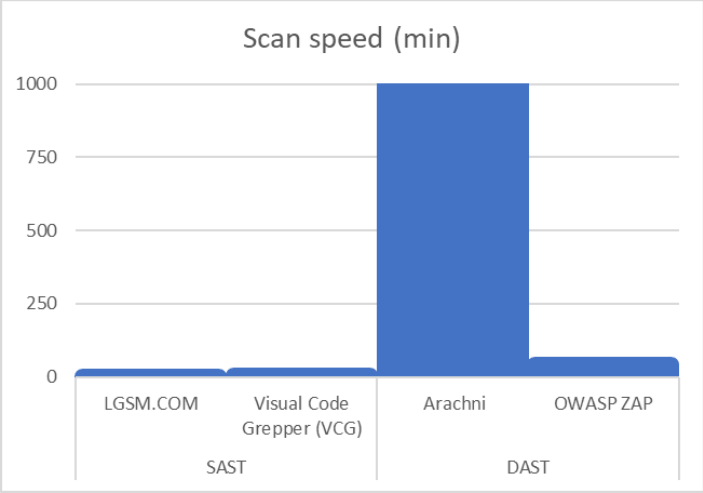

Figure 5: Graph of required scan time (min)

3) Numbers of detected security alerts and following parameters: true positive, false positive, true negative and false negative

Each tool has different capabilities for detecting security vulnerabilities. In this research paper, the Static Application Security Testing tools found significantly more security vulnerabilities compared to DAST tools. This is a relative parameter, as it is of utmost importance how many of them are truly real threats and how many are mislabelled. These specificities are determined by the values of true positive, false positive, true negative and false negative, visualized in Figure 8. It shows that the SAST and DAST tools generate approximately the same number of true negatives and false negatives but have a visible difference in the values of true positives and false positives. It is reflected in the fact that the Dynamic application security testing tools generated significantly lower numbers in the research paper.

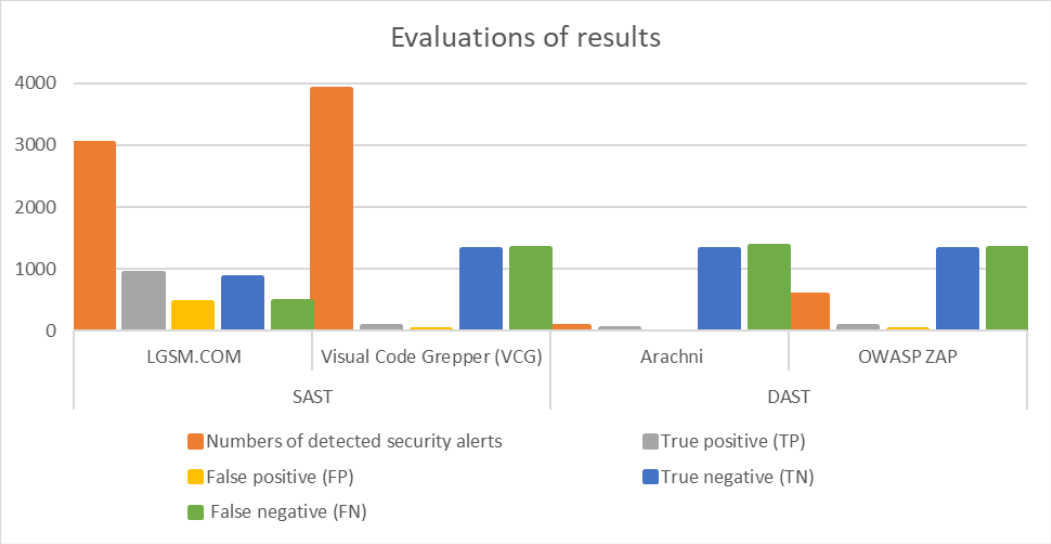

Figure 6: Diagram of evaluations of results

4) True positive rate, false positive rate, and score (%)

Based on the analysis of the previously discussed parameters, the True Positive Rate (TPR), False Positive Rate (FPR) and the total Score, which is the difference between the two, are also calculated. They evaluate the effectiveness and productivity of the types of tools used. As can be seen from figure 7, the SAST tool LGTM.COM has significantly higher results compared to the basic tools, which makes it the most effective of the considered ones. The remaining tools, both SAST and DAST tools have approximately the same results for the these parameters. Therefore, although they show similar results, it should be taken into account that their security vulnerability scans differ in functionality and the way they are performed. The first type, Static Application Security Testing tools, is used when source code is available and no active application is required to perform the scan, as is the case with Dynamic Application Security Testing tools. Based on this and the obtained results, it is established that although both types of security testing tools have approximately the same productivity and efficiency, their complex use is necessary to ensure the highest possible level of Cybersecurity.

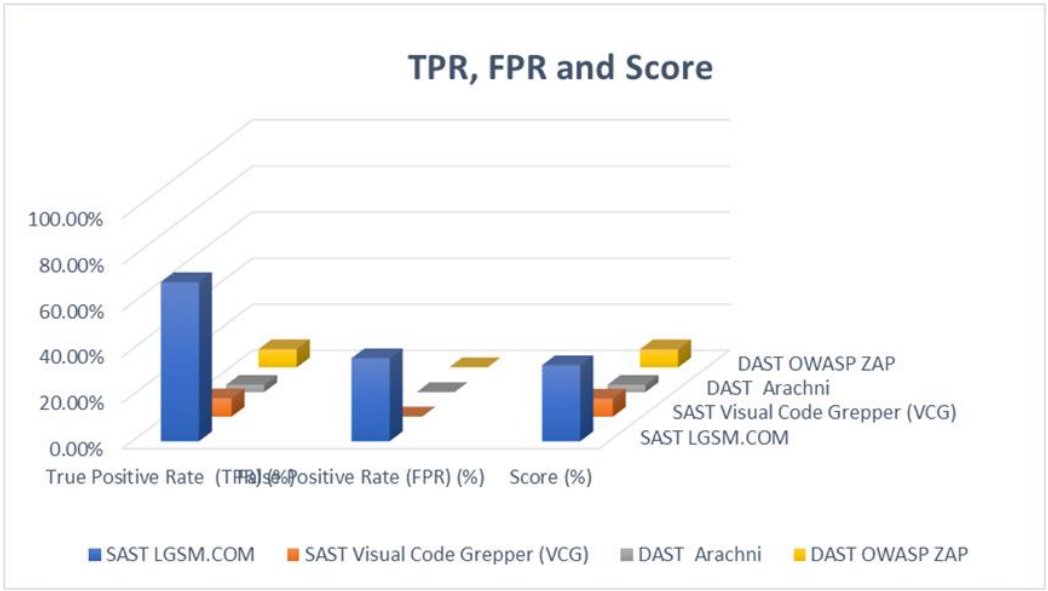

Figure 7: Diagram of TPR, FPR and Score

7 Conclusion and Future Work

The main objective of this master's thesis was to perform a comparative analysis between Static Application Security Testing tools and Dynamic Application Security Testing tools to establish not only their advantages and disadvantages, but also productivity and efficiency. In this way, it can determine which type of testing provides more opportunities and which is more reliable. For this purpose, analyses of two SAST and two DAST open-source tools were performed using OWASP Benchmark, which generated scorecards with the obtained results. It was reported that these tools have relatively similar results for some of the results, but there are also parameters on which they differ significantly. These are perfectly acceptable results of the research paper, as SAST and DAST tools scan web applications differently, with different functionality, and at a different stage of product deployment.

In the obtained results of the research, it is noticed that there are opportunities for the improvement of the research. This is aimed at Future work of this research paper. Examples of further development of the present topic are numerous, as is the case with the document. One of the directions in which this project can develop is to investigate other Static Application Security Testing tools and Dynamic Application Security Testing tools, whose effectiveness can be developed and compared with the available data. Another option is to use a different Benchmark to calculate, summarize and visualize the obtained results. In this way, an opportunity is provided to expand the master's thesis, helping to produce a more comprehensive, accurate and efficient result.

References

[1] López de Jiménez, R. E. (2016). Pentesting on web applications using ethical—Hacking. *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*, 1–6. https://doi.org/10.1109/CONCAPAN.2016.7942364

[2] Mohan, K. K., Verma, A. K., & Srividya, A. (2010). Software reliability estimation through black box and white box testing at prototype level. *2010 2nd International Conference on Reliability, Safety and Hazard - Risk-Based Technologies and Physics-of-Failure Methods (ICRESH)*, 517–522. https://doi.org/10.1109/ICRESH.2010.5779604

[3] Shebli, H. M. Z. A., & Beheshti, B. D. (2018). A study on penetration testing process and tools. *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 1–7. https://doi.org/10.1109/LISAT.2018.8378035

[4] Nidhra, S., & Dondeti2, J. (2012). Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)*, *2*(2), 29–50.

[5] Hamza, Z. A., & Hammad, M. (2019). Web and mobile applications' testing using black and white box approaches. *2nd Smart Cities Symposium (SCS 2019)*, 1–4. https://doi.org/10.1049/cp.2019.0210

[6] Acharya, S., & Pandya, V. (2012). Bridge between black box and white box—Gray box testing technique. *International Journal of Electronics and Computer Science Engineering*, *2*(1), 175–185.

[7] Mateo Tudela, F., Bermejo Higuera, J.-R., Bermejo Higuera, J., Sicilia Montalvo, J.-A., & Argyros, M. I. (2020). On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications. *Applied Sciences*, *10*(24), 9119. https://doi.org/10.3390/app10249119

[8] Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., & Pretschner, A. (2016). Chapter one - security testing: A survey. In A. Memon (Ed.), *Advances in Computers* (Vol. 101, pp. 1–51). Elsevier. https://doi.org/10.1016/bs.adcom.2015.11.003

[9] Croft, R., Newlands, D., Chen, Z., & Ali Babar, M. (2021). *An Empirical Study of Rule-Based and Learning-Based Approaches for Static Application Security Testing*.

[10] Yang, J., Tan, L., Peyton, J., & A Duer, K. (2019). Towards better utilizing static application security testing. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 51–60. https://doi.org/10.1109/ICSE-SEIP.2019.00014

[11] Ochoa M., D. (2021). *Dynamic application security testing (DAST)*.

[12] Nouman, M., Pervez, U., Hasan, O., & Saghar, K. (2016). Software testing: A survey and tutorial on white and black-box testing of C/C++ programs. *2016 IEEE Region 10 Symposium (TENSYMP)*, 225–230. https://doi.org/10.1109/TENCONSpring.2016.7519409

[13] Dhawan, D. & Heena. (2018). A study of white box and black box software testing. *International Journal of Emerging Technologies and Innovative Research, 5*(6), 679–683.

[14] Kumar, M., Singh, S. K., & Dwivedi, Dr. R. K. (2015). A comparative study of black box testing and white box testing techniques. *International Journal of Advance Research in Computer Science and Management Studies, 3*(10), 32–44.

[15] Jat, S., & Sharma, P. (2017). Analysis of Different Software Testing Techniques. *International Journal of Scientific Research in Computer Science and Engineering*, *5*(2), 77–80.

[16] Verma, A., Khatana, A., & Chaudhary, S. (2017). A Comparative Study of Black Box Testing and White Box Testing. *International Journal of Computer Sciences and Engineering*, 301–304. https://doi.org/10.26438/ijcse/v5i12.301304

[17] Ehmer Khan, M., & Khan, F. (n.d.). A comparative study of white box, black box and grey box testing techniques. *International Journal of Computer Sciences and Engineering*, *3*, 2012. https://doi.org/10.14569/IJACSA.2012.030603

[18] Hussain, T., & Singh, Dr. S. (2015). A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing. *International Journal of Allied Practice Research and Review*, *2*(5), 1–8.

[19] Hamilton, T. (2022). Black Box Testing Vs. White Box Testing: Key Differences. In *Guru99 Blog*. https://www.guru99.com/back-box-vs-white-box-testing.html

[20] Phadke, A. (2016). SAST vs. DAST: What's the best method for application security testing. In *Synopsys Blog*. https://www.synopsys.com/blogs/software-security/sast-vs-dast-difference/

[21] Henard, C., Papadakis, M., Harman, M., Jia, Y., & Le Traon, Y. (2016). Comparing white-box and black-box test prioritization. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 523–534. https://doi.org/10.1145/2884781.2884791

[22] Arendt, K., Jradi, M., Shaker, H. R., & Veje, C. T. (2018). Comparative analysis of white-, gray-and black-box models for thermal simulation of indoor environment: Teaching building case study. *Proceedings of the 2018 Building Performance Modeling Conference and Sim-Build Co-Organized by ASHRAE and IBPSA-USA, Chicago, IL, USA*, 26–28.

[23] Loyola-González, O. (2019). Black-box vs. White-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE Access*, *7*, 154096–154113. https://doi.org/10.1109/ACCESS.2019.2949286

[24] Mburano, B., & Si, W. (2018). Evaluation of web vulnerability scanners based on owasp benchmark. *2018 26th International Conference on Systems Engineering (ICSEng)*, 1–6. https://doi.org/10.1109/ICSENG.2018.8638176

[25] Sarpong, P. A., Larbi, L. S., Korsah, D. P., Abdulai, I. B., Amankwah, R., & Amponsah, A. (2021). *Performance Evaluation of Open Source Web Application Vulnerability*

Scanners based on OWASP Benchmark. *174*. https://doi.org/10.1109/ICAICST53116.2021.9497831

[26] Alsaleh, M., Alomar, N., Alshreef, M., Alarifi, A., & Al-Salman, A. (2017). Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners. *Security and Communication Networks*. https://doi.org/10.1155/2017/6158107

[27] Singh, S., Verma, M., & Kumar N, N. (2017). Open-Source Software Vs Proprietary Software. *International Journal of Scientific & Engineering Research*, *3*(12).

[28] Singh, A., Bansal, R. K., & Jha, N. (2015). Open-Source Software vs Proprietary Software. *International Journal of Computer Applications*, *114*(18).

[29] Heron, M., Hanson, V. L., & Ricketts, I. (2013). Open source and accessibility: Advantages and limitations. *Journal of Interaction Science*, *1*(1). https://doi.org/10.1186/2194-0827-1-2

[30] Abdou, T., Grogono, P., & Kamthan, P. (2012). A conceptual framework for open source software test process. *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, 458–463. https://doi.org/10.1109/COMPSACW.2012.87

[31] M. Parizi, R., Qian, K., Shahriar, H., Wu, F., & Tao, L. (2018). Benchmark requirements for assessing software security vulnerability testing tools. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, *01*, 825–826. https://doi.org/10.1109/COMPSAC.2018.00139

[32] Carr, T. (2022). Static Analysis: A bringup, overview and additions that you will actually use. In *XRLO — eXtended Reality Lowdown Blog*. https://medium.com/xrlo-extended-reality-lowdown/static-analysis-a-bringup-overview-and-additions-that-you-will-actually-use-24713ad7b1e4

[33] Shaukat, R., Shahoor, A., & Urooj, A. (2018). Probing into code analysis tools: A comparison of C# supporting static code analyzers. *2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 455–464. https://doi.org/10.1109/IBCAST.2018.8312264

[34] Katz, E. (2021). Top 10 Static Application Security Testing (SAST) Tools in 2021. In *Spectral Blog*. https://spectralops.io/blog/top-10-static-application-security-testing-sast-tools-in-2021/

[35] Woźniak, J. (2022). OWASP ZAP: Tool Description, Key Functionalities, and Useful Resources. In *Droptica Blog*. https://www.droptica.com/blog/owasp-zap-tool-description-key-functionalities-and-useful-resources/

[36] Holik, F., & Neradova, S. (2017). Vulnerabilities of modern web applications. *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1256–1261. https://doi.org/10.23919/MIPRO.2017.7973616

[37] Brombacher, F. (2022). Free and Inexpensive Security Testing Tools for DevOps. In *Crashtest Security Blog*. https://crashtest-security.com/devops-security-tools/#dasttools

[38] Laskos, T. (2011). Web application testing with Arachni. In *Infosec Blog*. https://resources.infosecinstitute.com/topic/web-application-testing-with-arachni/

[39] Antunes, N., & Vieira, M. (2015). On the metrics for benchmarking vulnerability detection tools. *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 505–516. https://doi.org/10.1109/DSN.2015.30

[40] Yamaguchi, F. (2018). Beating the OWASP Benchmark. In *ShiftLeft Blog*. https://blog.shiftleft.io/beating-the-owasp-benchmark-24a7b1601031