



# Evaluating and comparing the web application security testing tools: Identifying and Applying Key Metrics

Sanmay Bhavanish Thota  
Sai Ajit Jayasimha Vemula

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

**Contact Information:**

Authors:

Sanmay Bhavanish Thota

E-mail: sath21@student.bth.se

Sai Ajit Jayasimha Vemula

E-mail: save21@student.bth.se

University advisor:

Dr Usman Nasir

Department of Software Engineering

Faculty of Computing  
Blekinge Institute of Technology  
SE-371 79 Karlskrona, Sweden

Internet : [www.bth.se](http://www.bth.se)  
Phone : +46 455 38 50 00  
Fax : +46 455 38 50 57

---

# Abstract

**Background:**

Web application security (WAS) testing is crucial for protecting web applications from cyber threats. However, organizations often struggle to select effective WAS testing tools due to the lack of a well-defined set of evaluation criteria. This research aims to address this need by identifying the key metrics for evaluating and comparing WAS testing tools.

**Objectives:**

The primary objectives of this research are to identify the key metrics for comparing WAS testing tools, validate the significance of these metrics through semi-structured interviews, and perform a comparison between WAS testing tools using the validated metrics. This research aims to find a set of validated metrics for evaluating and comparing WAS testing tools.

**Methods:**

The research methodology consisted of three main phases: a literature review to compile a comprehensive set of technical and non-technical metrics commonly used for assessing and comparing WAS testing tools, semi-structured interviews with security experts to validate the significance of the identified metrics, and an experiment to compare three WAS testing tools - ZAP, Burp Suite, and Acunetix - using the OWASP Benchmark project. These three tools were selected based on the author's recommendations in the literature.

**Results:**

The initial literature review found 37 evaluation metrics for WAS testing tools. Through interviews, experts confirmed some of these were important, but also said some were not very useful. The experts additionally suggested some new metrics that were not in the literature. Incorporating this feedback, the final list was refined down to 35 metrics for evaluating WAS testing tools. An experiment was then conducted to compare three WAS testing tools - ZAP, Burp Suite, and Acunetix with the test subject as the OWASP Benchmark Project and by using the validated set of metrics. The results of this experiment revealed differences in the performance of the tools, with Burp Suite emerging as the best performer.

**Conclusions:**

This research has provided a valid set of metrics for comparing and evaluating WAS testing tools, empowering organizations to make more informed decisions. Security professionals can optimise their WAS testing tool selection by understanding the key metrics and their relative significance, as established through the literature and interviews. Based on the experimental analysis, Burp Suite performed better than other tools. Therefore, for organizations initiating the selection process of WAS testing tool, Burp Suite stands out as a good choice.

**Keywords:** Web application security, Web app security testing tool, vulnerability, Evaluation metrics.

---

## Acknowledgments

We want to express our gratitude to everyone who has contributed to the successful completion of our thesis.

First and foremost, we sincerely appreciate our supervisor, Usman Nasir, for his unwavering guidance, constructive feedback, and unwavering support throughout our research journey. His invaluable advice has played a pivotal role in shaping this thesis, and we are immensely grateful for his mentorship.

Our deepest gratitude goes to our parents and siblings for their constant support and encouragement, which has been a beacon of light during this journey. Their unwavering belief in us has served as a constant source of inspiration, enabling us to pursue our goals with passion and determination.

Additionally, we express our sincere thanks to our friends, family, and all those who participated in the interviews for their support, understanding, and patience during this phase of our research. Their encouragement has been invaluable in helping us overcome challenges and achieve our objectives.

---

# Contents

|  |            |
|--|------------|
| <b>Abstract</b>  | <b>i</b>   |
| <b>Acknowledgments</b>   | <b>ii</b>  |
| <b>Abbreviations</b>   | <b>vii</b> |
| <b>1 Introduction</b>  | <b>1</b>   |
| 1.1 Scope  | 2          |
| 1.2 Background   | 3          |
| 1.2.1 Web Application Security Testing                                 | 3          |
| 1.2.2 Web Application Security Testing Tools                           | 3          |
| 1.2.3 What are Metrics?  | 4          |
| 1.2.4 OWASP Benchmark Project  | 4          |
| 1.3 Outline  | 5          |
| <b>2 Related Work and Study Design</b>                                 | <b>7</b>   |
| 2.1 Related work   | 7          |
| 2.2 Research Gap   | 8          |
| 2.3 Study Design   | 9          |
| 2.4 Overview of Research Methods                                       | 9          |
| 2.4.1 Literature Review  | 10         |
| 2.4.2 Interviews   | 10         |
| 2.4.3 Experiment   | 12         |
| 2.5 Alternative Methods  | 12         |
| <b>3 Literature Review</b>   | <b>13</b>  |
| 3.1 Snowballing Implementation:  | 13         |
| 3.2 Literature Review Results  | 14         |
| 3.2.1 Snowballing Results  | 14         |
| 3.2.2 Metrics Used in Evaluating WAS Testing Tools                     | 16         |
| 3.3 Literature Review Analysis   | 22         |
| 3.3.1 Analysis and Justification for Choosing Tools for our Experiment | 22         |
| <b>4 Interviews</b>  | <b>24</b>  |
| 4.1 Participant Selection  | 24         |
| 4.2 Interview Questions  | 24         |
| 4.3 Thematic Analysis  | 26         |
| 4.4 Refining the Evaluation Metrics for WAS Testing Tools              | 35         |

|          |  |           |
|----------|--|-----------|
| <b>5</b> | <b>Experiment</b>  | <b>37</b> |
| 5.0.1    | Variables . . . . .                                      | 37        |
| 5.0.2    | Objects and Subjects . . . . .                           | 37        |
| 5.0.3    | Treatment . . . . .                                      | 38        |
| 5.0.4    | Environment Setup . . . . .                              | 38        |
| 5.0.5    | Implementation . . . . .                                 | 38        |
| 5.1      | Experiment Results . . . . .                             | 39        |
| 5.1.1    | ZAP when tested OWASP Benchmark Project . . . . .        | 39        |
| 5.1.2    | Burp suite when tested OWASP Benchmark Project . . . . . | 39        |
| 5.1.3    | Acunetix when tested OWASP Benchmark Project . . . . .   | 40        |
| 5.1.4    | Experiment Analysis . . . . .                            | 40        |
| <b>6</b> | <b>Discussion</b>  | <b>46</b> |
| 6.1      | Threats to validity . . . . .                            | 48        |
| 6.1.1    | Internal Validity Threats . . . . .                      | 48        |
| 6.1.2    | External Validity Threats . . . . .                      | 49        |
| 6.1.3    | Construct Validity Threats . . . . .                     | 49        |
| 6.1.4    | Conclusion Validity Threats . . . . .                    | 49        |
| <b>7</b> | <b>Conclusions and Future Work</b>                       | <b>50</b> |
| 7.1      | Conclusions . . . . .                                    | 50        |
| 7.2      | Future Work . . . . .                                    | 51        |
|          | <b>References</b>  | <b>52</b> |
| <b>A</b> | <b>Supplemental Information</b>                          | <b>57</b> |

---

## List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | Study Design . . . . .                                      | 9  |
| 4.1 | Steps we followed in Thematic analysis . . . . .            | 27 |
| 4.2 | Commonly Prioritized Technical Metrics . . . . .            | 27 |
| 4.3 | Commonly Prioritized Non-Technical Metrics . . . . .        | 29 |
| 4.4 | Additional metrics suggested by interviewees . . . . .      | 30 |
| 4.5 | Metrics Considered Less Important . . . . .                 | 32 |
| 4.6 | Metrics Trade-Offs and Situational Considerations . . . . . | 33 |
| 4.7 | Evolving Landscape of Web App Security . . . . .            | 34 |
| 5.1 | OWASP Benchmark Project experiment setup . . . . .          | 39 |
| 5.2 | Time taken to scan by each tool in min . . . . .            | 41 |
| 5.3 | Results of Tools for FPR and FNR . . . . .                  | 43 |
| 5.4 | Results of Tools for TPR and TNR . . . . .                  | 43 |

---

## List of Tables

|     |   |    |
|-----|---|----|
| 3.1 | Metrics . . . . .   | 17 |
| 3.2 | WAS testing Tools . . . . .                                     | 22 |
| 4.1 | Interviewee details . . . . .                                   | 24 |
| 4.2 | Metrics List . . . . .  | 36 |
| 5.1 | Environment Setup . . . . .                                     | 38 |
| 5.2 | Results of ZAP, Burp suite and Acunetix for OSWAP benchmark . . | 45 |
| A.1 | Start set . . . . .   | 57 |
| A.2 | 1st Iteration Forward Snowballing . . . . .                     | 58 |
| A.3 | 1st Iteration Backward Snowballing . . . . .                    | 59 |
| A.4 | 2nd Iteration Forward Snowballing . . . . .                     | 59 |
| A.5 | 2nd Iteration Backward Snowballing . . . . .                    | 60 |
| A.6 | 3rd Iteration Forward Snowballing . . . . .                     | 60 |
| A.7 | 3rd Iteration Backward Snowballing . . . . .                    | 61 |

---

# Abbreviations

- **OWASP** - Open Web Application Security Project
- **WAS** - Web application security
- **SQLI** - Structured Query Language Injection
- **XSS** - Cross-site scripting
- **CWE** - Common Weakness Enumeration
- **TP** - True Positives
- **FP** - False Positives
- **FN** - False Negatives
- **TN** - True Negatives
- **GUI** - Graphical User Interface
- **CLI** - Command Line Interface
- **HTML** - HyperText Markup Language
- **XML** - Extensible Markup Language
- **PDF** - Portable Document Format
- **DAST** - Dynamic Application Security Testing
- **FNR** - False Negative Rate
- **FPR** - False Positive Rate
- **IAST** - Interactive Application Security Testing
- **TPR** - True Positive Rate
- **TNR** - True Negative Rate
- **ZAP** - Zed Attack Proxy



In today's digital world, web applications have become indispensable for various tasks, ranging from online shopping and banking to accessing crucial services. However, these applications are vulnerable to different types of sophisticated cyber attacks, where malicious actors/persons exploit weaknesses to steal sensitive information or disrupt critical operations [42]. Safeguarding web applications from such threats is paramount, as a successful attack can have severe consequences, including financial losses, compromised user data, and erosion of public trust.

To address this challenge, robust Web Application Security (WAS) testing tools are vital [17]. These tools evaluate the security of web applications by identifying vulnerabilities, empowering organizations to take proactive measures to mitigate risks and safeguard against potential vulnerabilities. WAS tools are crucial because web applications are frequent targets for cyber attacks due to the sensitive data they handle and their critical role in business operations. By employing these tools, organizations can ensure compliance with regulations, protect user data, and maintain trust in their online services.

With numerous WAS testing tools available in the market, selecting the most suitable one for an organization's specific needs is a critical decision. To assess and compare WAS testing tools effectively and make a well-informed decision, it is crucial to conduct a comprehensive comparison [34]. By comparing the capabilities and effectiveness of different tools, organizations can determine which WAS testing tool is most aligned with their requirements and security objectives.

If the criteria used for comparison are effective, then the selection of a tool based on the comparison will also be effective [22]. In this context, metrics serve as the standards or criteria employed to assess these tools. Organizations can make well-informed decisions about which tool best suits their needs by identifying and prioritizing key metrics, such as the accuracy of vulnerability detection and types of vulnerability detection.

WAS testing tools are essential for protecting web applications from cyber attacks [31]. They help strengthen digital defences and ensure online security. Similarly, using the right metrics to compare these tools is important for making smart choices about which tool to use. This ensures that organizations can effectively safeguard their online presence and protect against hacking.

In this study, our objective is to examine WAS tools and ascertain the key metrics necessary for their evaluation. To accomplish this, we will engage with industry professionals who possess practical experience in testing with these tools. Using their expertise, we aim to identify the metrics essential for effectively assessing and com-

paring WAS tools. Subsequently, we will compare various WAS tools utilizing these identified metrics. We aim to facilitate informed decision-making for organizations seeking to enhance their web application security.

## 1.1 Scope

This thesis is scoped to systematically identify, validate, and apply key metrics to compare the capabilities of web application security testing(WAS) tools, to guide practitioners in selecting the most effective tools for their needs.

However, it's important to note that this study does not extensively explore non-web application security testing tools.

**Aim:** This thesis aims to identify key metrics for comparing web application security WAS testing tools. Additionally, it aims to compare WAS testing tools using the identified metrics.

### Objectives:

- Identify metrics utilized in existing literature for comparing WAS testing tools, including identifying positively rated WAS testing tools mentioned by the authors.
- Validate the identified metrics through discussions with experts in the field.
- Compare the identified WAS testing tools based on the validated metrics and provide recommendations accordingly.

### **RQ1. What are the metrics employed for comparing web application security (WAS) testing tools?**

**Justification:** *Understanding the metrics commonly employed for comparing WAS testing tools is crucial for evaluating different tools in various contexts. By identifying these metrics, one can make informed decisions when selecting the most appropriate tool for their specific security testing needs. Additionally, knowledge of these metrics can facilitate the development of standardized evaluation criteria and benchmarks for assessing the performance of WAS testing tools*

### **RQ2. Among the metrics identified in RQ1, which ones are considered important, and what are the reasons for their significance?**

**Justification:** *This question is significant because various research studies use different metrics to assess WAS testing tools. However, some of these metrics lack clear definitions, leading to confusion. By identifying which metrics are considered important and understanding why, we can gain clarity on the most relevant evaluation criteria. This helps simplify the evaluation process and ensures that the chosen metrics are well-understood and suitable for the field. This improves the effectiveness of assessing WAS testing tools.*

### **RQ3. Which WAS testing tool demonstrates better performance based**

**on the metrics obtained from RQ2?**

**Justification:** *Choosing the right WAS testing tool is crucial for accurately assessing vulnerabilities. This question is essential to understand which tools perform better when assessed using the important metrics from RQ2. This knowledge assists in making informed decisions about which tools to use, ensuring better security for web applications.*

## 1.2 Background

To provide the necessary context for this research, the following section presents background information on WAS testing and the use of evaluation metrics.

### 1.2.1 Web Application Security Testing

Web Application Security Testing is the process of evaluating and assessing the security posture of web applications. It involves identifying vulnerabilities, weaknesses, or potential entry points that malicious actors could exploit to gain unauthorized access, steal data, or disrupt the application's functionality [3].

The primary objective of WAS testing is to uncover security flaws and vulnerabilities in web applications before cybercriminals can exploit them [3]. This proactive approach to security helps organizations mitigate risks and protect their web applications, as well as the sensitive data and critical operations they support.

WAS testing encompasses a wide range of techniques and methodologies, including static code analysis, dynamic application security testing (DAST), and interactive application security testing (IAST) [2]. These methods are designed to assess different aspects of web application security, such as input validation, authentication mechanisms, session management, and access control, among others.

### 1.2.2 Web Application Security Testing Tools

WAS testing tools are specialized software applications designed to automate and facilitate the process of identifying and analyzing vulnerabilities in web applications [37]. These tools are specifically developed to assess the security of web-based systems and applications.

WAS testing tools typically employ a combination of techniques, including crawling and mapping the application's structure, injecting simulated attacks, and analyzing the application's responses to detect potential vulnerabilities. They can scan for a wide range of vulnerabilities, such as cross-site scripting (XSS), SQL injection, insecure configurations, and broken authentication or authorization mechanisms [37].

One of the key advantages of WAS testing tools is their ability to automate and streamline the security testing process, reducing the time and effort [20]. These tools can also provide comprehensive reports and detailed findings, enabling organizations to prioritize and address identified vulnerabilities effectively.

Additionally, WAS testing tools often offer features such as integration with other security tools, customizable testing configurations, and support for various applica-

tion frameworks and programming languages, making them versatile and adaptable to different web application environments.

### 1.2.3 What are Metrics?

In evaluating and comparing WAS testing tools, metrics refer to the specific measurements or criteria used to assess these tools. Metrics provide a standardized way to evaluate and compare different tools based on specific attributes or characteristics.

Metrics can be broadly categorized into two types:

#### 1. Technical Metrics:

These are metrics that directly measure the tool's ability to accurately identify and report vulnerabilities, which is the primary objective of WAS testing. Some examples include:

- True Positive Rate (TPR): The ratio of correctly identified vulnerabilities to the total number of actual vulnerabilities [34].
- False Positive Rate (FPR): The ratio of incorrectly identified vulnerabilities to the total number of non-vulnerabilities [34].
- False Negative Rate (FNR): The ratio of missed vulnerabilities to the total number of actual vulnerabilities [34].

#### 2. Non-technical Metrics:

These are metrics that evaluate other aspects of the WAS testing tool. Examples include:

- Pause and Resume Scans: The ability of the tool to pause the scan and resume from where it stopped.
- Results report: Automated tools can generate reports in different formats, including HTML, XML, and PDF.
- Usability: The ease of use and user-friendliness of the tool's interface and functionality.

By identifying and prioritizing the key metrics, organizations can make more informed choices when selecting WAS testing tools, ensuring that the chosen tools align with their specific security requirements and risk tolerance levels. Additionally, a standardized set of metrics can promote consistency in the evaluation process and facilitate the development of benchmarks and best practices within the industry.

### 1.2.4 OWASP Benchmark Project

The OWASP Benchmark Project is a Java-based web application designed to assess and compare the accuracy, and coverage of automated WAS testing tools [1]. It

provides a standardized test suite with thousands of purposely exploited test cases, each associated with a unique Common Weakness Enumeration (CWE) identifier. The project covers a wide range of security vulnerabilities and offers developers and security professionals a means to evaluate the effectiveness of various WAS testing tools. The test cases simulate real-world attack scenarios and undergo regular review and updates by a dedicated community. Balume Mburano et al. recommended that one must use the OWASP Benchmark as the primary target for evaluating WAS testing tools [34].

To access the OWASP Benchmark Project application, you can use the following URL: <https://owasp.org/www-project-benchmark/>

## 1.3 Outline

This section outlines the structure of the thesis.

Chapter 1: This chapter provides a detailed introduction to the thesis and introduces the aim and objectives of the research.

Chapter 2: This chapter provides an exploration of the related work in the field, an overview of the study design followed in this thesis, and an overview of the methods used.

Chapter 3: This chapter presents the implementation and analysis of the literature review.

Chapter 4: This chapter presents the interviews, includes the interview responses, and provides an interview analysis.

Chapter 5: This chapter presents the experiment implementation, results and analysis.

Chapter 6: Discussion of our thesis, featuring a sub-section on threats to the validity.

Chapter 7: The final chapter offers a conclusion and discusses potential future work for this thesis.



## Chapter 2

---

# Related Work and Study Design

## 2.1 Related work

Few researchers have explored the importance of evaluating and comparing web application security (WAS) testing tools to select the most effective tool for identifying vulnerabilities. In this section, we review and critically analyze the existing literature related to our research, highlighting the similarities, differences, and potential gaps.

The study by Nuno Antunes et al. [12] discussed a significant challenge in benchmarking WAS testing tools: different types of tools report vulnerabilities in distinct ways. For instance, while some tools identify vulnerabilities based on the application's response, others examine the code for potential security issues. This variance makes it difficult to compare the effectiveness of tools solely based on the number of reported vulnerabilities. Our research recognizes this challenge and strives to establish a comprehensive set of metrics to evaluate WAS testing tools beyond the reported vulnerabilities.

In another study, S. El Idrissi et al. [24] evaluated the effectiveness of various commercial and open-source web application analysis tools in addressing security issues. While their work provided insights into the performance of these tools, it did not explicitly focus on the metrics or criteria used for evaluation. Our research aims to address this by identifying and validating the most relevant metrics for assessing WAS testing tools, enabling a more standardized and effective evaluation process.

Marco Vieira et al. [15] explored the process of selecting suitable metrics for benchmarking software vulnerability detection tools. The authors gathered and analyzed a wide range of metrics, established criteria for good metrics, and evaluated the initial set of metrics based on the defined criteria. Although they considered different new metrics the selection and validation of these metrics were not thoroughly discussed. They haven't given proper reasons for which metrics to choose and which metrics not to choose. Our research builds upon this work by engaging with industry experts to validate the importance and relevance of specific metrics for evaluating WAS testing tools.

Few other studies, such as [24] and [12], have utilized established metrics like True Positive, False Positive, False Negative rates, precision, recall, and F-measure to evaluate the performance of WAS testing tools. However, the justification for the usage of these specific metrics is often lacking, highlighting the need for a more comprehensive and validated approach.

To develop a more comprehensive evaluation, some researchers have introduced new frameworks with additional parameters. For instance, Marwan et al. [9] pro-

posed a framework that includes parameters such as interface usability, types of crawling, scanning speed, types of scans supported, and coverage of OWASP Top 10 vulnerabilities. Similarly, Mandar [49] examined existing evaluation frameworks and introduced a new framework encompassing a broader range of parameters, including the type of tool, penetration test, crawling methods, coverage, scanning speed, scan types, detection rate, false positives, true positives, reporting features, addons, ease of configuration, scan logs, and cost.

While these frameworks offer a comprehensive approach to evaluating WAS testing tools, the reasons for selecting the additional metrics beyond those included in existing frameworks are not clearly explained. Our research aims to address this gap by engaging with industry practitioners to understand the essential metrics and the reasons behind their significance in evaluating WAS testing tools.

In addition to the studies mentioned above, Ruya et al. [46] addressed the challenge of selecting the appropriate web-based automated tool for specific testing processes. The authors compared 14 web-based automated tools across 20 different criteria, including cost, license, technical support, language support, user experience, documentation, browser support, environment support, testing type, and hardware requirements. However, the significance of these criteria beyond the ones already established in the literature was not clearly explained.

By reviewing the existing literature, it is evident that while researchers have made efforts to compare the WAS testing tools, there is a lack of consensus and clear justification for the selection of specific metrics that are used for comparison. Our research aims to bridge this gap by engaging with industry experts to identify the key metrics and understand the rationale behind their importance in evaluating WAS testing tools effectively.

## 2.2 Research Gap

The existing research on comparing WAS testing tools did not establish standard metrics for evaluation. A few studies used different criteria like accuracy, precision, and recall, but there was no widely accepted standard. Additionally, some researchers created new benchmarks without sufficient evidence that they were truly important or useful. This made it difficult for people to know which metrics were crucial for evaluating these tools. To fill this gap, this thesis aimed to identify the key metrics for evaluating WAS testing tools. This was achieved by reviewing previous research and gathering input from experts in the field. By identifying the essential metrics, this research facilitated a better understanding and selection of WAS testing tools, ultimately improving the assessment of web application security and aiding in better decision-making when choosing WAS testing tools.

## 2.3 Study Design

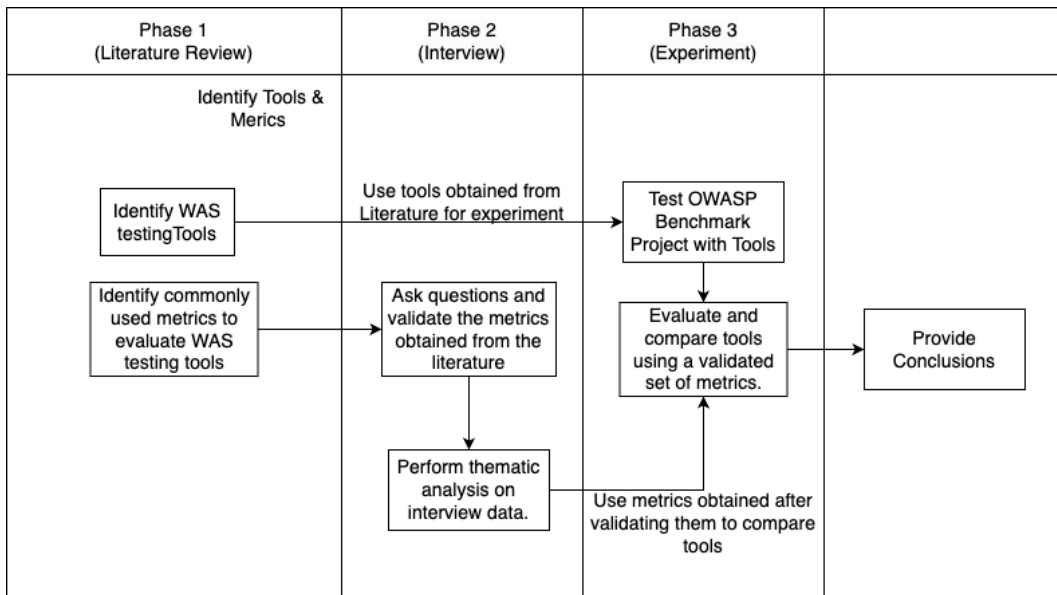


Figure 2.1: Study Design

Our research follows the approach illustrated in Figure 2.1.

In the first literature review phase, we identified the metrics commonly used to evaluate WAS testing tools and the WAS testing tools that were positively rated. This involved determining the WAS testing tools mentioned in the existing literature and compiling a list of the commonly used metrics for assessing these tools.

The second phase focused on the interviews. We used the metrics identified in the literature review as a starting point and asked questions to experts to validate the significance of these metrics. Through the interviews, we gained deeper insights into the practical importance of the various metrics and uncovered any additional metrics that the experts considered crucial in their decision-making process.

The third and final phase involved an experiment. We used the validated set of metrics obtained from the previous phases to evaluate and compare the performance of selected WAS testing tools. This experiment used the OWASP Benchmark Project as the test subject, allowing us to quantitatively assess the tools' capabilities across the identified key metrics.

The results of this research provided a comprehensive understanding of the important metrics to consider when evaluating and selecting WAS testing tools, offering valuable insights for both researchers and practitioners in the field of web application security.

## 2.4 Overview of Research Methods

We have chosen three methods: literature review, interviews, and experiments to answer our research questions. The methods used in this study are described in the sections 2.4.1, 2.4.2, 2.4.3.

### 2.4.1 Literature Review

We chose to conduct a literature review to answer our RQ1. This method allowed us to identify the metrics commonly used for comparing Web Application Security (WAS) testing tools. By examining existing studies, we thoroughly investigated the different metrics employed in this area. Through this review, we aimed to uncover both established and recent metrics that researchers use to evaluate the capabilities of WAS testing tools.

#### **Snowballing for Literature Review:**

This technique is helpful while performing a literature review, where the objective is to locate and compile all relevant research on a specific subject. There are two main types of snowballing techniques in research. They are forward snowballing and backward snowballing [56].

Claes Wohlin [56] suggests that snowballing is an important approach that can complement or even replace database searches to improve the reliability and thoroughness of systematic literature studies in software engineering.

**Forward Snowballing:** Forward snowballing refers to identifying new papers to include based on the papers that cite the paper being examined [56].

**Backward Snowballing:** Backward snowballing refers to using the reference list of a paper to identify new papers to include [56].

Both forward and backward snowballing techniques are valuable for ensuring a thorough exploration of the existing literature. By systematically tracing both forward and backward citation paths, researchers can identify a comprehensive range of relevant studies, thereby minimizing the risk of overlooking important sources of information.

### 2.4.2 Interviews

Interviews serve as an appropriate method for addressing RQ2 due to their inherent ability to elicit rich insights from participants. By engaging in direct conversation with individuals experienced in WAS testing, interviews offer the opportunity to delve deeper into their perspectives on the significance of various metrics.

Interviews allow for an exchange of ideas, enabling participants to elaborate on their experiences, preferences, and reasoning behind the selection of specific metrics. Through open-ended questioning, interviewers can explore the underlying factors that influence participants' perceptions of metric importance, such as practical relevance, impact on decision-making, and alignment with organizational goals. There are three types of interviews - unstructured, semi-structured, and fully structured [48].

We have chosen a semi-structured interview format. In a semi-structured interview, questions are prepared beforehand like in fully structured interviews, but they're not necessarily asked in the same order. The interviewer can change the

order based on how the conversation unfolds during the interview, making sure to cover all the planned questions. The interviewer adapts the conversation based on the subject's interests and level of engagement [48].

### **Thematic Analysis**

Thematic analysis is a widely used qualitative data analysis method that involves identifying, analyzing, and reporting patterns or themes within data. It is a flexible approach that is not tied to a particular perspective, making it a valuable tool for researchers across various disciplines [30].

This analytical method was well-suited for the current study, as it allowed for an in-depth exploration of the perspectives shared by the expert participants. The step-by-step thematic analysis process enabled us to engage with the interview transcripts thoroughly and derive key themes, rather than imposing a pre-determined coding framework.

The steps in the thematic analysis approach are:

**Step 1: Become familiar with the data** [30]: The first step in thematic analysis is to become thoroughly familiar with the data. This involves reading and re-reading the transcripts, making notes, and writing down initial impressions. This phase lays the foundation for the analysis by ensuring the researcher has a deep understanding of the data.

**Step 2: Generate Initial Codes** [30]: In this step, the researcher systematically codes relevant or interesting segments of the data. Coding reduces the data into manageable chunks of meaning. The coding can be driven by the research question (top-down, theoretical) or more data-driven (bottom-up, inductive) [30].

**Step 3: Search for Themes** [30]: The next phase involves identifying patterns of meaning, or themes, within the coded data. Themes are broader than codes and capture something significant or interesting about the data in relation to the research question.

**Step 4: Review Themes** [30]: During this step, the researcher reviews and refines the preliminary themes identified in the previous step. This involves considering whether the themes make sense if the data supports them, and if there are any overlapping or redundant themes.

**Step 5: Define Themes** [30]: In this phase, the researcher defines and names the final themes. This involves determining the scope and focus of each theme and ensuring they are distinct from one another.

**Step 6: Write-up** [30]: The final step is to produce a coherent, and compelling written account of the analysis. This should provide the reader with a clear understanding of the themes and how they address the research question.

The thematic analysis process is often iterative, with the researcher moving back

and forth between the different steps as the analysis progresses. This flexibility allows the researcher to refine the themes and ensure they accurately reflect the data.

### 2.4.3 Experiment

Experimental methodology refers to the systematic approach used to design and conduct experiments to observe and measure variables and draw conclusions [57]. We chose experiment methodology for RQ3 to examine the outcomes obtained from testing the web application with different WAS testing tools. After testing the tools, we assessed their performance based on the validated set of evaluation metrics that we had identified from the literature review and refined through the interviews.

## 2.5 Alternative Methods

- **Conducting a survey:**

The decision to use semi-structured interviews instead of a survey was based on the need for a more in-depth exploration of expert opinions. Interviews allowed us to investigate the reasons behind the identified metrics, ask follow-up questions, and uncover additional insights that may have been missed in a survey format.

- **Performing a more extensive comparison analysis of WAS testing tools:**

The primary focus of this thesis is on the identification and validation of the key metrics, rather than a comprehensive comparative analysis of tools. Expanding the experiment section may divert the attention from the main research objectives.

To identify the metrics for evaluating WAS testing tools, a literature review was conducted as the first phase of this research. As discussed in the method chapter, we used snowballing for our literature review. In this chapter, we present the detailed methodology and results of our snowballing process.

### 3.1 Snowballing Implementation:

It has only two steps, one is to identify the start set, and the other is to implement the forward and backward snowballing.

#### **Start Set Identification:**

We chose specific keywords and search strings to find relevant papers for our initial set. Utilizing these keywords and strings, we conducted searches across various databases such as Google Scholar, IEEE, Research Gate, etc.

#### **Keywords:**

1. Web application security
2. Web application vulnerability scanner
3. Metrics
4. Evaluation
5. Testing tools
6. Comparision

#### **Search Strings:**

- ("Evaluation" OR "Comparision" ) AND - of - ("Web application vulnerability scanner")
- ("Evaluation" OR "Comparision" ) AND ("Metrics")- for -("Web application security") AND ("Testing tools")

We chose our initial set following the guidelines proposed by Claes Wohlin [56], which emphasized the importance of diversity. Accordingly, our start set included works from various publishers and authors to ensure a broad representation.

#### **Inclusion criteria:**

- Only papers that compare WAS testing tools using any metrics.
- Abstracts related to WAS testing tools comparison.
- Papers written in English.

**Exclusion criteria:**

- Repeated and duplicate papers will not be included
- Papers that discuss WAS testing tools without comparison.
- Papers that discuss only WAS.

**Data Extraction:**

The relevant information from all the research papers found through the snowballing process has been gathered and recorded in an Excel sheet named "Research-Papers". The data extraction form contains various details essential for our study.

- Title
- Link to paper
- Name of the authors
- Tools they have used
- Web application they have used to test the tools
- Number of references
- Number of citations
- Year of publication
- Best tool according to authors
- Metrics used by authors to compare WAS testing tools

## 3.2 Literature Review Results

The literature review process yielded relevant studies that formed the foundation for our research. The following subsections detail the results of the start set, as well as the subsequent forward and backward snowballing results.

### 3.2.1 Snowballing Results

#### 3.2.1.1 Start Set

We initiated our search using terms related to web vulnerability testing tools, which we further refined using the provided list. From this search, we identified three relevant papers, which served as the starting point for our subsequent forward and backward snowballing searches. These papers were chosen because they did not reference each other. Table A.1 presents the papers included in our start set.

### 3.2.1.2 First Iteration

In the first forward snowballing iteration, we identified 88 papers, which were evaluated for relevancy through abstract and full-text reviews. Following the application of inclusion and exclusion criteria, 11 papers were selected for further examination. Among the papers not selected for further examination, 24 were in different languages, 7 were duplicates, and 2 were inaccessible. Table A.2 in the appendix outlines the outcomes of the initial forward snowballing iteration.

During the first iteration of backward snowballing, we reviewed a total of 59 papers. After applying inclusion and exclusion criteria, 9 papers were deemed relevant to our study. Among the papers not selected for further examination, 5 were duplicates and 2 were inaccessible. The results of the first backward snowballing iteration are detailed in Table A.3 in the appendix.

### 3.2.1.3 Second Iteration

In the second iteration of forward snowballing, we identified 1137 papers. After applying inclusion and exclusion criteria, 5 papers were found to be relevant to our study. Among the papers not selected for further examination, 6 were inaccessible, 34 were in different languages, and 41 were duplicates. The results of the forward snowballing iteration are presented in Table A.4 in the appendix.

During the second iteration of backward snowballing, we identified 667 papers. After applying inclusion and exclusion criteria, 11 papers were deemed relevant to our study. Among the papers not selected for further examination, 6 were inaccessible, 9 were in different languages, and 101 were duplicates. The results of the backward snowballing iteration are outlined in Table A.5 in the appendix.

### 3.2.1.4 Third Iteration

During the third iteration of forward snowballing, we identified 662 papers. After applying inclusion and exclusion criteria, 5 papers were found to be relevant to our study. Among the papers not selected for further examination, 40 were in different languages, and 65 were duplicates. The results of the forward snowballing iteration are presented in Table A.6 in the appendix.

During the third iteration of backward snowballing, we identified 441 papers. After applying inclusion and exclusion criteria, 3 papers were deemed relevant to our study. Among the papers not selected for further examination, 8 were inaccessible, 16 were in different languages, and 60 were duplicates. The results of the backward snowballing iteration are outlined in Table A.7 in the appendix.

### 3.2.1.5 Fourth Iteration

Following the third iteration of snowballing, we identified 66 papers for forward snowballing. After applying inclusion and exclusion criteria, none were found to be relevant to our study.

During the fourth iteration of backward snowballing, we reviewed a total of 236 papers. After applying inclusion and exclusion criteria, we did not find any papers that met our preferences. Consequently, the snowballing procedure was terminated

as no suitable papers were encountered.

After thoroughly analyzing 47 relevant papers for our thesis, we explored all the metrics used to compare WAS testing tools. Our focus was on understanding the metrics used to evaluate these tools. These metrics will be discussed in the following sections.

### 3.2.2 Metrics Used in Evaluating WAS Testing Tools

In our exploration of the literature, we identified many metrics employed by researchers to evaluate WAS testing tools. These metrics provide insights into the effectiveness and overall performance of these tools.

We have divided the metrics into technical and non-technical for better understanding. Table 3.1 presents a comprehensive overview of these metrics, both technical and non-technical, used by researchers to assess WAS testing tools.

This structured categorization of the evaluation metrics allows for a more thorough and balanced assessment of the capabilities and characteristics of the WAS testing tools being considered.

Table 3.1: Metrics

*This table showcases a list of metrics that researchers utilized to assess WAS testing tools.*

| Evaluation Metrics for WAS Testing Tools    |                               |
|---|-------------------------------|
| Technical Metrics                           | Non-Technical Metrics         |
| Precision                                   | Automation Level              |
| Recall                                      | Pause and Resume Scan         |
| F-measure                                   | Reporting Features            |
| True positives                              | Interface                     |
| False negatives                             | Addons and Extension features |
| False positives                             | Type of Penetration test      |
| Accuracy                                    | Platform Compatibility        |
| Types of different vulnerabilities detected | Browser Compatibility         |
| True positive rate                          | Technical Support             |
| False positive rate                         | OWASP Top 10                  |
| True negatives                              | Ease of configuration         |
| True negative rate                          | Protocol Support              |
| Time taken to scan                          | Tool cost                     |
| Markedness                                  | Crawling Type                 |
| Informedness                                |                               |
| Youden index                                |                               |
| A number of vulnerabilities were detected   |                               |
| Vulnerability Detection Rate                |                               |
| Scan logs                                   |                               |
| Fx Score                                    |                               |
| Specificity                                 |                               |
| Matthews Correlation Coefficient            |                               |
| False negative rate                         |                               |

Below are the definitions of metrics that are used in reviewed papers:

- **True Positives (TP):** True Positives is the number of true vulnerabilities detected when a tool reports a vulnerability where one is present in the code [21] [6].
- **False Positives (FP):** False Positives is the number of vulnerabilities detected that do not exist [21] [6].
- **False Negatives (FN):** False negatives happen when a tool fails to find a vulnerability that is present in the code. This means the tool misses a problem that exists [21] [6].

- **Precision:** The ratio of correctly detected vulnerabilities to the number of all detected vulnerabilities. Also referred to as true positive accuracy, positive predictive value, or confidence [50]. It can be represented by the following formula:

$$\text{Precision} = \frac{TP}{TP + FP} * 100$$

- **Specificity:** Rate of negative cases that is correctly classified negative. Also known as true negative rate or inverse recall [15].

$$\text{Specificity} = \frac{TN}{FP + TN} * 100$$

- **Recall:** A ratio of correctly detected vulnerabilities to the number of all known vulnerabilities. Recall is also called as Sensitivity [6]. It can be represented by the following formula:

$$\text{Recall} = \frac{TP}{TP + FN} * 100$$

- **F-Measure:** Represents the harmonic mean of precision and recall. Equivalent to the F1 Score [6]. The formula for the F-measure is:

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} * 100$$

- **Accuracy:** Accuracy measures the overall correctness of a system or tool in correctly identifying both true positives (TP) and true negatives (TN) relative to all predictions [7]. It is calculated as the ratio of correct predictions to the total number of predictions [19]. The formula for accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} * 100$$

- **Types of Different Vulnerabilities Detected:** This metric evaluates the diversity of vulnerabilities detected by the tool. It provides insight into the tool's ability to identify various types of security threats commonly found in web applications [50].
- **False Positive Rate (FPR):** False positive rate measures the proportion of non-existent vulnerabilities incorrectly identified by the tool among all vulnerabilities flagged. Lower false positive rates indicate better accuracy in avoiding false alarms [34].

$$\text{False positive rate} = \frac{\text{FalsePositives}}{\text{FalsePositives} + \text{TrueNegatives}} * 100$$

- **False Negative Rate (FNR):** False negative rate measures the proportion of actual vulnerabilities missed by the tool among all vulnerabilities present in the application. A lower false negative rate indicates better sensitivity in detecting vulnerabilities [32].

$$\text{False Negative Rate} = \frac{\text{FalseNegatives}}{\text{TruePositives} + \text{FalseNegatives}} * 100$$

- **True Negative Rate (TNR):** The true negative rate measures the proportion of non-existent vulnerabilities correctly identified as negative by the tool among all non-vulnerable instances. It represents the tool's ability to avoid false alarms for non-existent vulnerabilities [32].

$$\text{True Negative Rate} = \frac{\text{TrueNegatives}}{\text{TrueNegatives} + \text{FalsePositives}} * 100$$

- **True Positives Rate (TPR):** True positive rate measures the proportion of actual vulnerabilities correctly identified by the tool among all vulnerabilities present in the application. It indicates the tool's effectiveness in capturing genuine security issues [34].

$$\text{True Positive Rate} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}} * 100$$

- **Time Taken to Scan:** This metric measures the time it takes for the tool to complete the scanning process. It reflects the speed of the tool in assessing the security posture of the web application [9] [54].
- **Markedness:** Markedness penalizes the ratio of true positives considering the ratio of false positives and false negatives. It provides a measure of how well the tool balances between correctly identifying vulnerabilities (true positives) and avoiding false positives [15].

$$\text{Youden index (J)} = \frac{TP}{TP + FN} + \frac{TN}{FP + TN} - 1$$

- **Informedness:** Informedness computes all positive and negative cases found and adjusts the result based on the presence or absence of true positives. For each true positive, the result is increased in a ratio of 1/P, while if not detected, it is decreased in a ratio of 1/N, where P is the number of positives (test cases with vulnerabilities) and N is the number of negatives (test cases with no vulnerabilities) [15].

$$\text{Informedness (BM)} = \frac{TP}{TP + FN} - \frac{FP}{FP + TN}$$

- **Youden Index (J):** The Youden Index, also known as Youden's J statistic, is a single metric that captures the overall performance of a binary classifier, such as a vulnerability detection tool. It is calculated as the difference between the true positive rate (sensitivity) and the false positive rate (1-specificity) [47].

$$\text{Youden index (J)} = \frac{TP}{TP + FN} + \frac{TN}{FP + TN} - 1$$

- **Number of Vulnerabilities Detected:** This metric simply counts the total number of vulnerabilities identified by the WAS testing tool during the scanning process. It provides a quantitative measure of the tool's effectiveness in uncovering security threats [45].

- **Tool Cost:** Tool cost evaluates the financial expense associated with acquiring and using the WAS testing tool. It includes factors such as licensing fees, subscription costs, and additional charges for support or updates [9].
- **Crawling Type:** Crawling, a form of scanning, involves systematically navigating an application by sending requests and cataloguing discovered links. These links are then used for further scanning. Two types of crawling exist: active and passive. Active crawlers engage with the application by sending requests to obtain active links, while passive crawlers operate quietly without directly interacting with the application. Passive crawling occurs while manually navigating through the application, allowing for broader coverage of links and paths within the application [9].
- **Protocol Support:** Protocol Support refers to the range of network protocols that the web application testing tool can interact with during scanning. It assesses the tool's capability to handle various communication protocols used in web applications, such as GET, POST, COOKIE, HEADER, SECRET, PName, Custom, PROXY, GZIP, DEFLATE, SSL a total of 11 [43].
- **Vulnerability Detection Rate:** Vulnerability Detection Rate measures the percentage of vulnerabilities identified by the tool compared to the total number of vulnerabilities present in the web application. It indicates the tool's effectiveness in detecting security threats relative to the overall vulnerability landscape [8].

$$\text{VDR} = \frac{\text{Number of vulnerabilities detected by the scanner}}{\text{Number of vulnerabilities present in the benchmark}} * 100$$

- **Ease of Configuration:** Ease of Configuration assesses how straightforward it is to set up and configure the web application testing tool for use. It considers factors such as documentation clarity, and intuitiveness of configuration options [49].
- **Scan Logs:** Scan Logs refer to the detailed records generated by the tool during the scanning process. They include information about detected vulnerabilities, scan progress, errors encountered, and other relevant data for analysis and reporting [49].
- **Type of Penetration Test:** Type of Penetration Test specifies the nature and scope of the security assessment conducted by the tool. It may include options for black-box testing, white-box testing, Grey testing or a combination of these approaches [49].
- **Platform Compatibility:** Platform Compatibility evaluates whether the tool is compatible with various operating systems and environments. It ensures that the tool can be deployed and used across different platforms, such as Windows, Linux, macOS, etc [46].

- **Browser Compatibility:** Browser Compatibility assesses whether the tool can effectively analyze web applications across different web browsers. It ensures that the tool can handle variations in browser rendering and behaviour during scanning [46].
- **Technical Support:** Technical Support evaluates the availability and quality of support services provided by the tool’s developers or vendors. It includes options for documentation, helpdesk assistance, training materials, and community forums [46].
- **OWASP Top 10:** This metric counts the total number of vulnerabilities identified by the tool that are classified within the OWASP Top 10 list of the most critical web application security risks. It indicates the tool’s coverage of high-priority vulnerabilities outlined by OWASP [23].
- **Fx Score:** The Fx Score is a composite metric that combines precision and recall (or any other pair of metrics) into a single score using a specific weighting factor. It provides a balanced assessment of a tool’s performance, considering both the ability to avoid false positives and false negatives [15].

$$\text{Fx Score} = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}}$$

- **Matthews Correlation Coefficient (MCC):** Represents a correlation coefficient between the true classes and the classified results. It is also equivalent to the geometric mean of markedness and informedness [15].

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$$

- **Automation Level:** Automation Level assesses the capability of the web application testing tool to conduct scans autonomously, without requiring manual intervention from penetration testers [9].
- **Pause and Resume Scans:** The ability to pause and resume the scan from the same point is a strength factor for the scanner and it helps the tester reduce the time for rescanning the web application [9].
- **Reporting Features:** Automated tools generate reports in multiple formats, including PDF, HTML, and XML, which are commonly offered by scanners. Additionally, scanners provide compliance reports for various standards like PCI-DSS, HIPAA, and OWASP Top 10. These reports enable users to swiftly assess if they have met the desired standards. [9].
- **Interface:** The interface metric assesses whether a WAS testing tool provides both a Graphical User Interface (GUI) and a Command-line Interface (CLI). This metric compares the availability of both interfaces within the tool’s functionality.

These observations highlight the different metrics used by researchers to evaluate and compare WAS testing tools, reflecting the multifaceted nature of web application security assessment.

### 3.2.2.1 Web Applications Security testing tools

According to the findings of papers from the literature review, the tools listed in table 3.2 emerged as the top choices based on the author's testing. The table also denotes the number of papers in which each tool was identified as the best option. In some research papers, the authors have not specified a tool that performed better than others. Instead, they have indicated that the most effective tool may depend on the specific situation or context being examined.

Table 3.2: WAS testing Tools

| Best tool mentioned in paper | Number of papers |
|------------------------------|------------------|
| OWASP ZAP                    | 18               |
| Vega                         | 1                |
| Burp Suite                   | 9                |
| Skipfish                     | 5                |
| Acunetix                     | 5                |
| NetSparker                   | 1                |
| Arachni                      | 2                |

## 3.3 Literature Review Analysis

After reviewing 47 papers in the literature, We compiled a list of 37 metrics that authors used to compare WAS testing tools. However, the literature did not provide a standard justification for selecting these metrics.

To validate these metrics and understand the reasoning behind their selection, We conducted interviews. Based on the insights gained from the interviews, performed an experiment to compare WAS testing tools and evaluate their performance using the validated metrics.

For the experiment, We selected the WAS testing tools that were recommended as the best by the authors in the research papers. Below is a detailed explanation of the process used to choose the tools for the experiment.

### 3.3.1 Analysis and Justification for Choosing Tools for our Experiment

1. **ZAP(Zed Attack Proxy):** This tool was prominently mentioned in 18 papers, indicating its widespread recognition among researchers. ZAP's popularity as the best tool can be attributed to its comprehensive features, reliability, and strong

community support, making it a preferred option for many in the field. Given ZAP's established reputation, we have decided to use it as one of the tools for our experiment. This will allow us to compare it against other tools using metrics that have been finalized based on our literature review and interviews. It is noteworthy that the separation of ZAP from OWASP took place in August 2023, leading to the re-branding of OWASP ZAP as "ZAP."

**2. Burp Suite:** Noteworthy in 9 papers, Burp Suite emerged as a favoured choice due to its versatility and extensive functionalities. Its reputation for effectiveness and flexibility contributes to its continued popularity among security professionals and researchers. Given Burp Suite's demonstrated capabilities, we have also decided to include it in our experiment.

**3. Skipfish:** Skipfish was identified as the preferred tool in 5 papers, valued for its efficient vulnerability detection. However, it is worth noting that Skipfish was last updated **12 years ago**, which is a relatively long period in the cybersecurity landscape. This raises concerns about Skipfish's ability to detect the latest vulnerabilities that have emerged in recent years. Given the potential limitations of an outdated tool, we have decided not to include Skipfish in our experiment.

**4. Acunetix:** Acunetix was also identified as the preferred tool in 5 papers, with its user-friendly interface and thorough scanning capabilities contributing to its frequent endorsements. Given Acunetix's positive recognition in the literature and its active development, we have decided to include it in our experiment, alongside ZAP and Burp Suite.

**4. Arachni:** Although less frequently mentioned, Arachni was recognized as the optimal tool in 2 papers. However, it has come to our attention that the Arachni scanner has been replaced by a different scanner by the company. Due to this change, we have decided not to include Arachni in our current experiment, as we want to focus on tools that are actively maintained and supported.

**5. Vega:** Vega was mentioned once in the literature, indicating its presence in the landscape of WAS testing tools. However, it is worth noting that Vega was last updated **8 years ago**, which raises concerns about its ability to detect the latest vulnerabilities. Given the outdated nature of Vega, we have decided not to use this tool in our experiment.

**6. NetSparker:** NetSparker was mentioned twice in the literature, also indicating its presence in the WAS testing tool ecosystem. When we contacted the NetSparker team to request access to the tool for our experiment, they informed us that they do not provide tool access to students. Given the lack of access to NetSparker, we have decided not to include these tools in our experiment.

Based on the above analysis, we have decided to choose **ZAP, Acunetix, and Burp Suite** for our experiment.

After conducting the literature review, we obtained a list of metrics as displayed in Table 3.1. Using these metrics as a base, we prepared interview questions and selected participants for the interview. We validated these metrics using semi-structured interviews. The methodology and analysis of the interviews are presented below.

### 4.1 Participant Selection

We selected participants with experience in WAS testing tools from various companies and organizations to conduct the interviews. We searched for individuals on LinkedIn and sent them messages related to our interview. Additionally, we joined a security community on Slack and shared our request for interviews there as well.

We formulated a set of 12 questions and crafted an interview template. We interviewed six people with varying experience from 8 to 30 years. The interview also included demographic inquiries regarding the participant's roles and experience related to WAS testing. Table 4.1 presented the information about the individuals who were interviewed.

Table 4.1: Interviewee details

| S.No.        | Role  | Experience |
|--------------|---|------------|
| Respondent 1 | Penetration Tester  | 12 years   |
| Respondent 2 | Application Security Architect (contract) / OWASP London Chapter Leader | 25 years   |
| Respondent 3 | ZAP Project Lead  | 14 years   |
| Respondent 4 | Principal Security Architect  | 12 years   |
| Respondent 5 | Chair of the Global Board of Directors at OWASP                         | 30 years   |
| Respondent 6 | Security Analyst  | 8 years    |

### 4.2 Interview Questions

To gather insights from security experts, a set of interview questions was developed. The purpose of these questions was to validate the metrics obtained from the literature review. During the interview, we presented the list of metrics obtained from the

literature review by sharing our screen and then proceeded to ask the questions. The questions were designed to gather insights from people experienced in WAS testing. We aimed to understand their perspectives on the selection and evaluation of WAS testing tools.

Firstly, we asked participants to introduce themselves and their roles in the field of cybersecurity, including their experience with WAS testing. This helped establish their expertise, ensuring their insights were relevant and valuable.

Next, we asked about the key factors or criteria they consider when assessing or choosing web application security testing tools. This provided valuable insights into the decision-making process from experienced professionals. Participants were then asked to identify the most important metrics for comparing WAS testing tools, based on their experience. This helped prioritize metrics that are widely regarded as crucial in the field.

We also sought input on any potentially missing metrics that participants believed were important for evaluating WAS testing tools. This helped ensure comprehensive coverage of relevant evaluation criteria.

Additionally, we explored potential trade-offs between different metrics and asked participants to prioritize metrics in conflicting situations. This shed light on decision-making strategies when faced with competing evaluation criteria.

Below are the questions we prepared:

1. *Could you please introduce yourself and your role in the field of cybersecurity, particularly in web application security testing?*

**Metrics Selection:**

2. *Based on your experience, What are the key factors or criteria you consider when assessing or choosing WAS testing tools?*
3. *Based on the list of metrics (got from literature review) displayed on the screen from your experience, which ones do you consider most important for comparing web application security testing tools?*
  - (a) *Why do you consider those particular metrics important for comparing web application security testing tools?*
4. *Are there any metrics that you believe are crucial for evaluating web application security testing tools but are missing from the list? If so, what are they and why do you consider them important?*
5. *In your experience, Are there any metrics from the list that you believe are less important and aren't used regularly? If there are, could you explain why you think so?*
6. *When evaluating a tool's performance, do you primarily focus on metrics like TP, FP, FN, and TN, or do you also consider derived metrics such as accuracy, TPR and precision?*
  - (a) *If considering other derived metrics, which ones should be included from the following list and why? TPR, FPR, TNR, FNR, Precision, Recall, F-measure, Accuracy, Markedness, Informedness, Youden Index, Fx Score, Specificity, Matthews Correlation Coefficient?*

**Metrics Trade-offs:**

7. *Have you encountered situations where different metrics conflict with each other when evaluating web application security testing tools?*
  - (a) *If there is a situation which metric should be prioritized for evaluation?*

**Other Factors:**

8. *Beyond technical metrics, what other factors or considerations do you take into account when evaluating web application security testing tools? (e.g., usability, scalability, integration capabilities)*
9. *In your opinion, what external factors or industry trends should be considered when selecting and evaluating web application security testing tools?*

These questions acted as a starting point for the interview. However, as the interview progressed, we explored the participant's responses and thoughts further to gain deeper insights into the topic. While the interviews were recorded, we also took notes to capture the key points of the discussion. This note-taking process enabled us to further engage with the participants and explore the topics beyond the initial set of questions, allowing for more in-depth answers.

### 4.3 Thematic Analysis

The responses collected from the interviews were then subjected to a thematic analysis to identify patterns and themes. In conducting a thematic analysis for our thesis, we followed several steps to analyze and interpret the data collected from interviews. Firstly, we transcribed the interview recordings into written transcripts. Then, we reformatted the transcripts with grammatically correct sentences to enhance our understanding. Afterwards, we familiarized ourselves with the interview transcripts to gain insight into the content. Subsequently, we systematically identified patterns, recurring topics, and ideas within the data. These were labelled as "codes" and assigned to relevant segments of the transcripts. Once all the data was coded, we grouped similar codes together to form "themes" that captured broader concepts or emerging patterns from the data. These themes were then refined and organized into a coherent narrative representing the key findings of the analysis. Throughout the process, we ensured that the analysis remained faithful to the original data and reflected the perspectives and experiences of the participants. The figure 4.1 shows a visual representation of our thematic analysis process [30].

The identified themes are as follows: Commonly Prioritized Technical Metrics, Commonly Prioritized Non-Technical Metrics, Additional metrics suggested by interviewees, Metrics Considered Less Important, Metrics Trade-Offs and Situational Considerations, and the Evolving Landscape of Web App Security.

Details about these themes are provided below. The star-like diagrams in the following sections, such as fig. 4.2 to fig. 4.7, represent these themes and include all

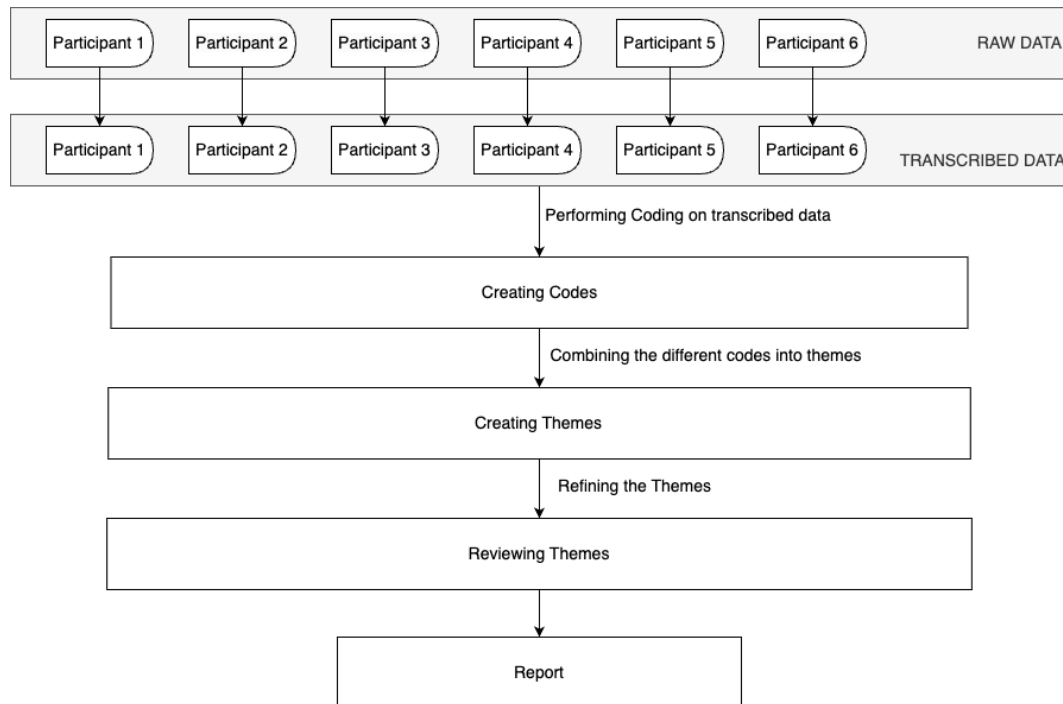


Figure 4.1: Steps we followed in Thematic analysis

the codes that make up each theme, where each oval in the figures represents themes and the curved rectangles represent codes.

### 1. Commonly Prioritized Technical Metrics:

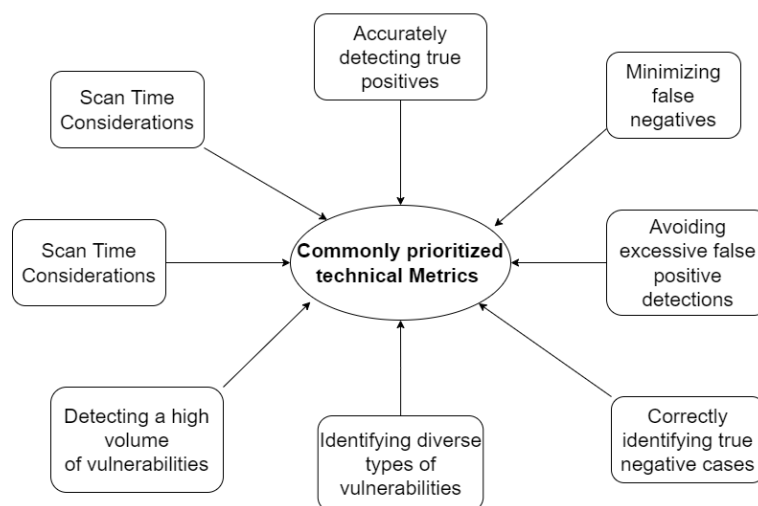


Figure 4.2: Commonly Prioritized Technical Metrics

At the core of the evaluation process are the technical metrics, which provide a fundamental baseline for the tool's performance. Accurately detecting true positive vulnerabilities, minimizing false negative detections, and avoiding excessive false positive detections were unanimously cited as essential factors. As one expert noted,

"These core metrics give us a clear understanding of the tool's ability to identify real security issues without generating too many false alarms."  
(Respondent 1)

Additionally, correctly identifying true negative cases, though not weighted as heavily as the other core metrics, was seen as contributing to a more comprehensive understanding of the tool's overall capabilities. Participant explained,

"Correctly identifying normal application functions that are not vulnerabilities is just as important as detecting the actual security issues. This helps us concentrate our efforts on the areas that truly require attention and remediation. (Respondent 1 )"

Beyond these core metrics, the experts also emphasized the importance of identifying diverse types of vulnerabilities, going beyond just the raw number of detected issues. As one security professional explained,

"The breadth of vulnerability coverage is crucial - we need tools that can uncover a wide range of security flaws, not just the most common ones.  
(Respondent 3 )"

The total number of vulnerabilities detected and the overall vulnerability detection rate were also viewed as relevant technical metrics, providing insights into the tool's ability to identify security issues comprehensively.

The interview findings also highlighted the Scan Time as a crucial technical metric. As one expert stated,

"The time taken to scan is very important in the built environment or DevOps because if you implement this tool in the build pipeline and it takes a long time to scan, it will have many impacts. (Respondent 2 )"

Another participant added,

"Faster scan times allow us to integrate the security testing more seamlessly into our development and deployment processes, which is essential for maintaining a responsive security posture. (Respondent 2)"

The security professionals also emphasized the importance of derived metrics, such as accuracy, precision, and the F1-score, which provide a more holistic understanding of the tool's performance. As one expert commented,

"These aggregate metrics give us a better sense of the tool's overall effectiveness, beyond just the individual true positive, false negative, and false positive rates.(Respondent 6)"

## 2. Commonly Prioritized Non-Technical Metrics:

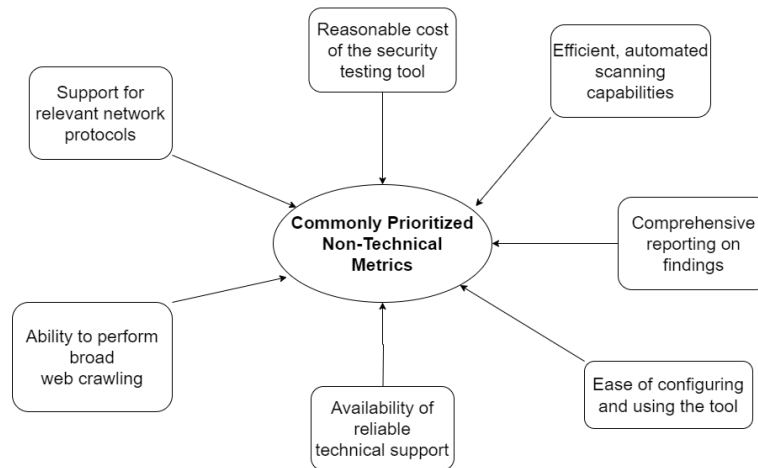


Figure 4.3: Commonly Prioritized Non-Technical Metrics

In addition to the technical metrics, the security experts also highlighted the importance of various non-technical elements when evaluating WAS testing tools. These non-technical considerations help ensure the tool's practical usefulness and alignment with the organization's security needs.

One key non-technical metric was the Cost of the security testing tool. The experts emphasized that the tool's pricing, including any ongoing fees or hidden costs, needs to be reasonable and fit within the organization's security budget. As one participant said,

"We can't afford an expensive tool, even if it has advanced features. The cost has to make sense for our situation. (Respondent 5)"

The Automation of the scanning capabilities were also cited as crucial non-technical factors. The security teams want tools that can streamline the testing process and minimize manual effort. As an expert explained,

"We need automated scans and other efficient features so our analysts can focus on analyzing the results, rather than just running the tests. (Respondent 4)"

Different types of vulnerability reporting are other important non-technical metrics. The experts said the tool's reports need to provide clear, actionable insights into the identified vulnerabilities, including their potential impact and guidance for addressing them. As one participant noted,

"Reporting features are important because I am always specifically looking for a report format that can be aggregated and read by a computer. (Respondent 6)"

The Ease of Use for the tool was also highlighted as an important non-technical metric. If the tool is overly complex or difficult to set up, it can create barriers to adoption and reduce the effectiveness of the security testing program. As an expert mentioned,

"Ease of configuration is important. Complicated tools can be difficult to work with. The easier it is to configure and the more configurable it is, the better, as long as you don't have to spend too much time configuring it before you can start using it. (Respondent 3)"

The availability of Reliable Technical Support from the vendor and the tool's compatibility with the organization's existing technology were also seen as important non-technical metrics. As one expert noted,

"Responsive and knowledgeable support from the tool vendor is essential, especially during the initial setup and if any issues come up. We need to be able to quickly resolve problems and get the most out of the tool's capabilities. (Respondent 6)"

By considering this comprehensive set of technical and non-technical factors, security teams can make informed decisions when selecting the most suitable WAS testing tool for their organization's needs.

### 3. Additional metrics suggested by interviewees:

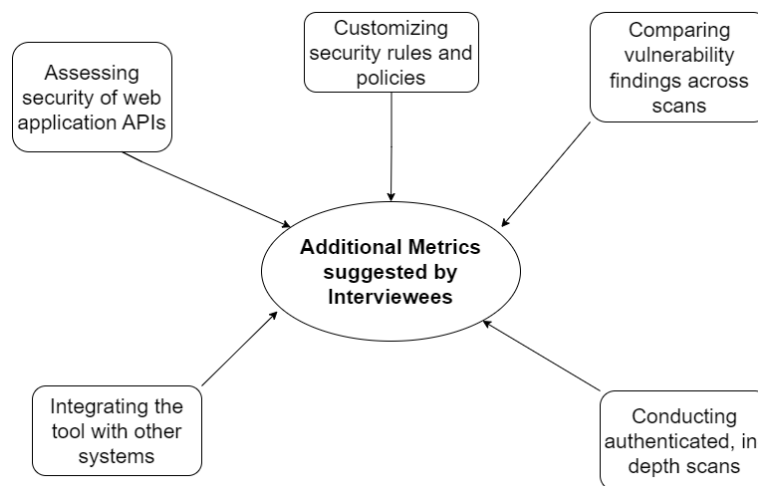


Figure 4.4: Additional metrics suggested by interviewees

As we have a list of commonly used metrics for evaluating WAS testing tools from the literature review that are shown in Table 3.1, we asked the experts if there are any additional metrics that they feel are important but missing from the list. Building on this, experts explained the importance of some metrics that are not used by authors in the literature.

The ability to customize the security testing tool's rules and policies to align with the specific needs and requirements of the organization is crucial. As one expert explained,

"The ability to customize the security testing tool's rules and policies to align with the specific needs of the organization is crucial. Since each

company has unique priorities, this customization ensures the tool effectively addresses our concerns, enhancing its overall utility. (Respondent 2)"

Metrics related to the tool's ability to compare results between different scans over time are important to track progress and identify changes. As one of the experts noted,

"Tracking changes in each scan is essential for monitoring the effectiveness of the tool and identifying any trends or improvements in the security posture of the web applications.(Respondent 2)"

The capability to perform authenticated scans is a crucial metric, as these scans enable the tool to detect vulnerabilities that may require authentication to access, providing a more comprehensive and accurate assessment of the web application's security, as experts mentioned.

The tool's integration capabilities with other security and development tools are an important consideration. As an expert mentioned,

"Integration with CI/CD tools enables a more cohesive security workflow, allowing for better collaboration. (Respondent 6)"

Metrics related to the tool's coverage of API-specific security testing are crucial, as APIs are becoming increasingly important in modern web applications. As one expert stated,

"Evaluating the tool's ability to identify vulnerabilities in the API layer is essential for ensuring the overall security of the web application. (Respondent 4)"

By highlighting these additional metrics, the experts provided valuable insights beyond the literature, showcasing the evolving needs and priorities in the WAS testing domain. These metrics underscore the importance of customization, cross-tool integration, and the growing emphasis on API security, which organizations should consider when selecting and evaluating their WAS testing tools.

#### **4. Metrics Considered Less Important:**

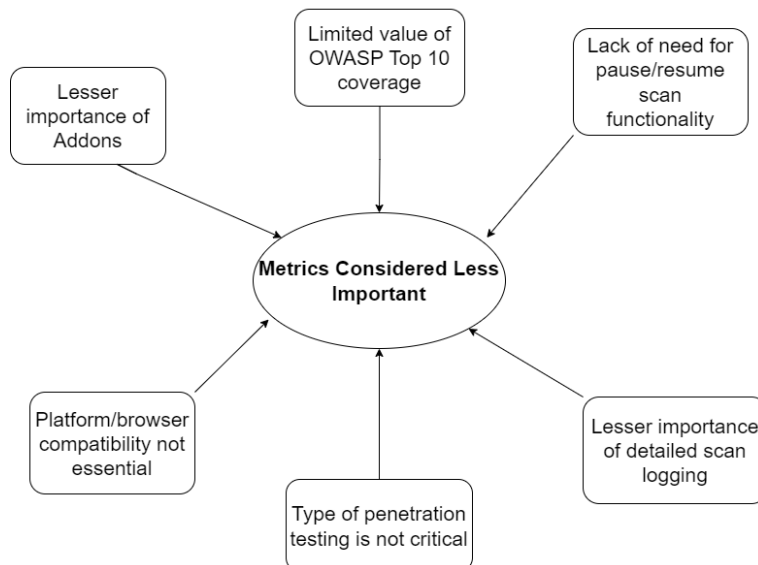


Figure 4.5: Metrics Considered Less Important

While several additional important metrics were highlighted beyond the literature review, certain metrics were considered less critical when evaluating WAS testing tools.

Coverage of the OWASP Top 10 vulnerabilities was one such metric deemed less important. As one expert explained,

"The OWASP Top 10 is a good starting point, but we found that it doesn't necessarily reflect the unique risks and vulnerabilities in our environment. Focusing too heavily on just the OWASP Top 10 would give us an incomplete picture of our application security posture. (Respondent 3)"

Instead, the recommendation was to take a more holistic approach, considering a wider range of metrics tailored to the organization's specific needs and threat landscape. The ability to pause and resume security scans was also not a high priority, as an expert said,

"In our experience, we rarely need to pause and resume scans. The testing process typically completes without interruption, so this functionality isn't a critical requirement for us. (Respondent 4)"

The Scan logs were less important than other metrics. As one expert stated,

"While comprehensive logging can be useful for some organizations, we found that the detailed scan logs weren't a major factor in our decision-making process. We're more focused on the actionable insights and remediation guidance provided by the tool. (Respondent 1)"

The specific methodology or type of penetration testing performed by the security tool was not a crucial consideration. An expert noted,

"Penetration testing is a human-led activity where testers use the results of vulnerability scanners to see if they can exploit identified vulnerabilities. For example, while a scanner might detect a potential SQL injection, a penetration tester would try to exploit it to confirm its existence. Therefore, it's important to note that type of penetration testing is unimportant in this case. (Respondent 6)"

Compatibility with different platforms and web browsers was not an essential metric. As one expert explained,

"Platform compatibility is not a major concern because if you containerize something, you can run it anywhere. Browser compatibility is also not important. (Respondent 4)"

In addition to the metrics mentioned above, the experts also considered the availability of add-ons or plugins to be a less important factor. As one expert commented,

"While add-ons can provide additional functionality, we found that the core capabilities of the security tool were more important to us than the availability of add-ons. (Respondent 1)"

The experts emphasized that their priority was to identify a security testing tool that effectively addressed their core needs, such as identifying vulnerabilities, providing detailed remediation guidance, and integrating with their existing security workflows. The availability of add-ons or plugins, while potentially useful, was not a critical factor in their decision-making process.

By understanding which metrics are considered less critical, organizations can concentrate their evaluation and selection efforts on the most essential factors that align with their specific security requirements and priorities.

## 5. Metrics Trade-Offs and Situational Considerations:



Figure 4.6: Metrics Trade-Offs and Situational Considerations

The interview gave us insights into what metrics must be considered when there are trade-offs or when prioritizing which metrics to focus on. The experts highlighted that when it comes to evaluating WAS testing tools, there is no one-size-fits-all approach. Organizations have different priorities and needs, so the security metrics they focus on should reflect that.

The experts emphasised the need to prioritize the security metrics based on their specific goals and objectives. As one expert mentioned,

"The metrics we care about will depend on our organization's main security priorities. If we're more concerned with regulatory compliance, certain metrics will take precedence over others. (Respondent 5)"

Some companies might want to improve their overall security, while others want just to meet certain requirements. Aligning the security metrics with your specific goals will help you choose the right testing tools.

By considering these trade-offs and situational factors, organizations can ensure that the security metrics they use are tailored to their unique needs, effectively supporting their overall security goals and objectives.

## 6. Evolving Landscape of Web App Security:



Figure 4.7: Evolving Landscape of Web App Security

When evaluating security testing tools, organizations also need to consider the changing nature of web application security. This includes two key areas:

As more companies move their web apps to the cloud, the security tools need to adapt. More than traditional scanning methods may be needed to handle the dynamic and distributed nature of cloud-hosted applications.

Organizations need security solutions that can seamlessly integrate with cloud platforms. These tools should provide visibility into cloud resources and services, and be able to adapt as the cloud environment changes. Features like automated cloud monitoring and integration with cloud security services are important. As one expert said,

"With the growing adoption of cloud, we need security tools that can handle these new challenges. The old ways of scanning may not be sufficient for our cloud-based infrastructure. (Respondent 6)"

The web application security landscape is constantly changing, with new attack techniques and vulnerabilities always emerging. Security tools need to be able to keep up with these evolving threats.

As one expert explained,

"The threat landscape is always shifting, with hackers finding new ways to exploit web app weaknesses. Our security tools need to not only find known issues but also detect and respond to these emerging threats. (Respondent 3)"

By considering both cloud security and the evolving threat landscape, organizations can select security testing tools that are equipped to handle both current and future web application security challenges.

## 4.4 Refining the Evaluation Metrics for WAS Testing Tools

The initial literature review identified a set of metrics for evaluating web application security (WAS) testing tools, as presented in Table 3.1. However, after further analysis and incorporating insights from the expert interviews and thematic analysis, some adjustments were made to this evaluation criteria.

Specifically, the expert inputs suggested that certain metrics from the initial list, such as OWASP Top 10, Pause and Resume Scan, Scan Logs, Type of Penetration Testing, and Platform/Browser Compatibility, were less crucial for assessing the capabilities of these tools.

At the same time, the experts highlighted the importance of several additional metrics that were not extensively covered in the prior literature. These included Customize Rules, Comparing Between Scans, Authenticate Scans, Integration Capabilities, and API Security Testing.

By removing the less crucial metrics from the initial list and adding the newly identified important ones, the final set of evaluation metrics is represented in Table 4.2. This refined, expert-informed list provides a comprehensive standard for assessing the capabilities of WAS testing tools based on the real-world needs and experiences of practitioners in the field.

Table 4.2: Metrics List

| <b>Metrics List</b>                         |                               |
|---|-------------------------------|
| <b>Technical Metrics</b>                    | <b>Non-Technical Metrics</b>  |
| Precision                                   | Automation Level              |
| Recall                                      | API security testing          |
| F-measure                                   | Reporting Features            |
| True Positives                              | Interface                     |
| False Negatives                             | Addons and Extension features |
| False Positives                             | Integration capabilities      |
| Accuracy                                    | Comparing between scans       |
| Types of different vulnerabilities detected | Customize rules               |
| True positive rate                          | Technical Support             |
| False positive rate                         | Authenticate scans            |
| True Negatives                              | Ease of configuration         |
| True negative rate                          | Protocol Support              |
| Time taken to scan                          | Tool cost                     |
| Markedness                                  | Crawling Type                 |
| Informedness                                |                               |
| Specificity                                 |                               |
| A number of vulnerabilities were detected   |                               |
| Vulnerability Detection Rate                |                               |
| Matthews Correlation Coefficient            |                               |
| Fx Score                                    |                               |
| False negative rate                         |                               |

After refining the set of evaluation metrics through the interviews, as shown in Table 4.2, the next step was to conduct an experiment to compare the selected WAS testing tools using the validated metrics. As discussed in the above sections, we chose ZAP, Acunetix, and Burp Suite for our experiment based on the literature review.

### 5.0.1 Variables

There are mainly two types of variables in an experiment

- **Independent Variables:**

These are the variables that we manipulate or control in our experiments [57]. In the context of our research, the independent variables are:

1. **Security Testing Tools:** The three WAS testing tools Acunetix, Burp Suite and ZAP are independent variables. We have used these tools under different configurations to compare them.
2. **Web Applications:** The Web Application 'OWASP Benchmark Project' is also considered an independent variable. We evaluated how each WAS testing tool performed when tested against the OWASP Benchmark Project.

- **Dependent Variables:**

These are the outcomes or measurements that we observe or record as a result of changes in the independent variables [57]. In our research, the dependent variables are the metrics.

### 5.0.2 Objects and Subjects

- **Objects:** The objects are the specific entities or items that are being tested, examined, or interacted with during your research [57]. They serve as the target of our experiments. In our case, the objects are the web applications and WAS testing tools themselves.
- **Subjects:** The subjects or participants are the individuals actively involved in the experiments. They are the ones responsible for applying the treatments and interacting with the objects [57]. In our case, subjects can be individuals (authors themselves) who are conducting security tests using WAS testing tools.

### 5.0.3 Treatment

In the context of our thesis, which focuses on evaluating and comparing WAS testing tools treatment can be defined as follows:

**Treatment:**

Varying the Configuration Settings of WAS Testing Tools.

**Description:**

The treatment involves setting up a proxy server in some cases and gradually changing configuration parameters, such as the scan depth of each chosen security testing tool ( Acunetix, Burp Suite, ZAP), to examine their effects on vulnerability detection. Before performing security tests, each tool's configuration settings are adjusted for the OWASP Benchmark project.

### 5.0.4 Environment Setup

The installation and testing environment for our tools is described in the table below:

Table 5.1: Environment Setup

| Tool Name  | Tool Host Environment   |
|------------|---|
| ZAP        | Kali Linux Debian 2023.1, 6-core Intel Core i7 12th gen, 64-bit, 1.70 GHz, 16GB RAM |
| Burp Suite | Kali Linux Debian 2023.1, 6-core Intel Core i7 12th gen, 64-bit, 1.70 GHz, 16GB RAM |
| Acunetix   | Kali Linux Debian 2023.1, 6-core Intel Core i7 12th gen, 64-bit, 1.70 GHz, 16GB RAM |

### 5.0.5 Implementation

**OWASP Benchmark Project:**

To test the OWASP Benchmark Project in a virtual machine using Acunetix, ZAP, and Burp Suite, we followed the steps outlined in figure 5.1:

- We installed the required software and packages and cloned the OWASP Benchmark Project repository.
- Ensured all three tools were installed in the environment.
- We referred to the tips on the OWASP Benchmark website to properly scan the OWASP Benchmark Project using the tools.
- Requested the results in XML format.
- Moved the results files of three tools to the "results" folder in the OWASP Benchmark Project folder.

- To generate scorecards for all the result files located in the /results directory, we executed the following command:

```
./createScorecard.sh
```

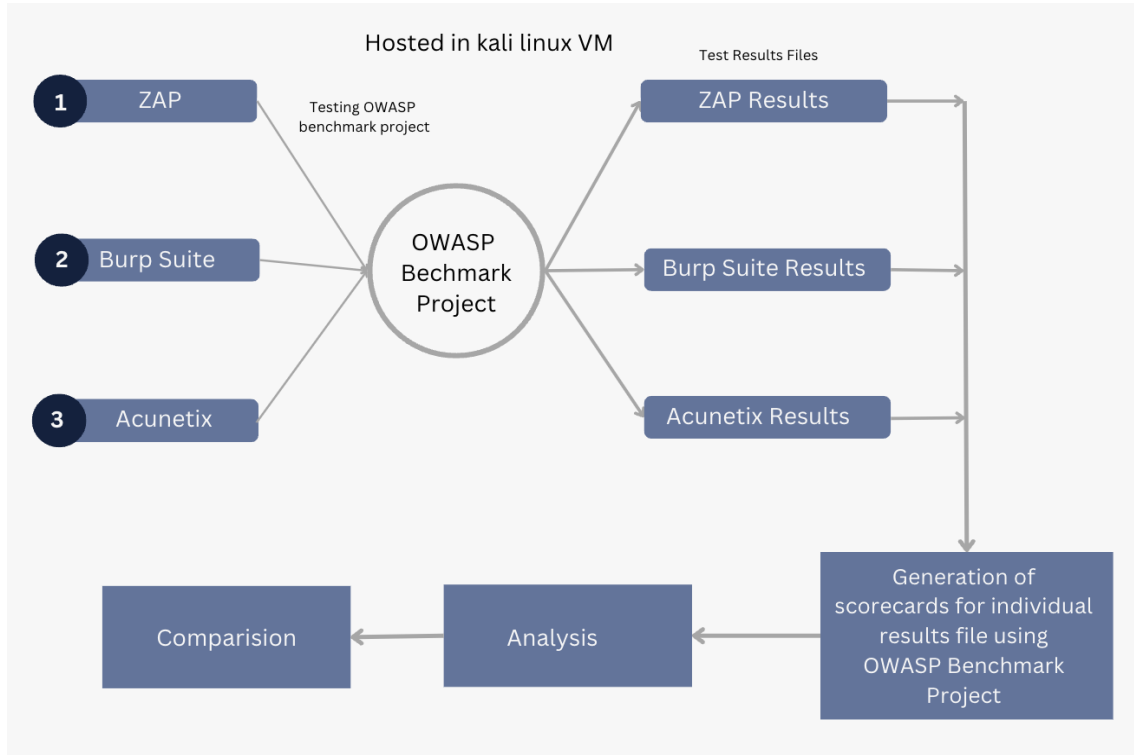


Figure 5.1: OWASP Benchmark Project experiment setup

## 5.1 Experiment Results

The experiment was conducted by testing the selected web application security tools - ZAP, Burp Suite, and Acunetix - against the OWASP Benchmark Project, with the following results:

### 5.1.1 ZAP when tested OWASP Benchmark Project

ZAP demonstrated a user-friendly GUI, supported active and passive crawling, and completed the scan in approximately 17 hours and 21 minutes. It was easy to configure and offered multiple types of UI for generating reports. ZAP allowed pause and resume of scans and detected 441 true positives. ZAP supported active, passive, and policy scans, resulting in a vulnerability detection rate of 34.91%. The details of the ZAP against metrics are presented in the table 5.2.

### 5.1.2 Burp suite when tested OWASP Benchmark Project

Burp Suite, a GUI tool, supported both active and passive crawling. The scan was completed in approximately 4 hours and 18 minutes. Configuration was user-friendly,

and standard reports in HTML and XML format were available. Burp suite allowed for pause and resume of scans, demonstrated moderate automation and detected 497 true positives. The rate of vulnerability detection was 35.11%. Details of the metrics are provided in the table 5.2.

### 5.1.3 Acunetix when tested OWASP Benchmark Project

Acunetix, a GUI tool, supported both active and passive crawling. The scan was completed in 129 minutes and 18 seconds. Configuration was user-friendly, and comprehensive scan reports, including OWASP Top 10 compliance, were available. Acunetix demonstrated high automation and detected 306 true positives. The rate of vulnerability detection was 22.83%. Details of the metrics and scores are provided in the table 5.2.

### 5.1.4 Experiment Analysis

In this section, we provide an analysis of the results obtained from our experiment. Each metric offers valuable insights into the performance, capabilities, and effectiveness of these tools in identifying vulnerabilities within web applications.

The analysis examines various metrics outlined in the table 4.2. By examining these metrics in detail, we aim to gain a deeper understanding of the strengths and limitations of each tool.

Table 5.2 presents a comprehensive analysis of three WAS testing tools: ZAP, Burp Suite, and Acunetix, showcasing their outcomes across various metrics. The table includes several key performance indicators and features, allowing for a detailed comparison of the tools.

- **True positives:** Burp Suite was the most effective at correctly identifying vulnerabilities, with 497 true positives. Following closely behind was ZAP, which found 441 true positives. Acunetix identified 306 true positives, indicating it had the lowest effectiveness among the three tools.
- **False Negatives:** Acunetix performed the worst in this metric, missing the most vulnerabilities, with 1109 false negatives. ZAP followed closely behind with 974 false negatives, while the Burp suite had 918 false negatives. Therefore, in terms of missing vulnerabilities, Acunetix ranked lowest among the three tools.
- **False positives:** Burp Suite excelled in this metric by not reporting any false positives. Acunetix had 17 false positives, while ZAP had 53. Therefore, Burp Suite demonstrated the best performance in correctly identifying vulnerabilities, with zero false positives.
- **True negatives:** Burp Suite performed the best on this metric, correctly identifying 1325 non-vulnerabilities as such. Acunetix followed closely with 1308

true negatives, while ZAP had 1272. Therefore, in terms of correctly identifying non-vulnerabilities, Burp Suite demonstrated the highest performance among the three tools.

- **Types of different vulnerabilities detected:** ZAP was better at identifying a broader range of vulnerability types compared to the other two tools. Specifically, ZAP detected 45 different types of vulnerabilities, while Burp Suite detected only 15 and Acunetix detected 5. This broader coverage suggests that ZAP has a more comprehensive approach to identifying vulnerabilities across a diverse range of scenarios and potential threats.
- **Time taken to scan:** ZAP took the longest time to complete the scan at 17 hours and 21 minutes, while Acunetix was the fastest at 129 minutes. Burp Suite fell in the middle at 4 hours and 18 minutes. This suggests that Acunetix is the fastest at scanning. Figure 5.2 presents these results in a bar graph format.

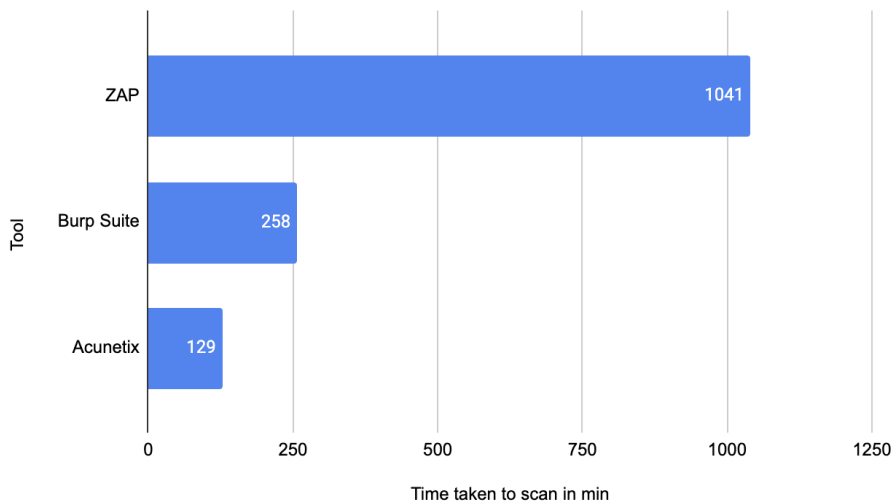


Figure 5.2: Time taken to scan by each tool in min

- **Number of vulnerabilities detected:** Burp Suite and ZAP found the most vulnerabilities overall. Specifically, Burp Suite detected 497 vulnerabilities, while ZAP detected 494. Acunetix detected fewer vulnerabilities, totalling 323. Therefore, both Burp Suite and ZAP demonstrated a higher capability in identifying vulnerabilities compared to Acunetix.
- **Tool cost:** ZAP is a free and open-source tool, while Burp Suite and Acunetix are commercial tools that require a paid license.
- **Crawling Type:** All three tools can use both active and passive crawling techniques.
- **Protocol Support:** All three tools have the capability to support 11 different protocols, namely GET, POST, COOKIE, HEADER, SECRET, PName, Custom, PROXY, GZIP, DEFLATE, and SSL.

- **Automation Level:** The level of automation for the Burp suite was classified as "medium" because the scanning process involved a combination of automated steps and manual intervention. Unlike tools like ZAP and Acunetix which were able to fully automate the vulnerability scanning of the target application, the Burp suite workflow required us to navigate and scan the application page-by-page manually.
- **Reporting Features:** Acunetix stood out as the best option among the three tools it offered various user interface results options, including HTML reports and compliance standard reports like OWASP Top 10. In contrast, Zap and Burp Suite lacked compliance standard reports like OWASP Top 10. However, all three tools provided results in PDF, XML, and HTML formats, ensuring compatibility and flexibility in accessing and sharing vulnerability assessment results.
- **Interface:** All three tools had a graphical user interface (GUI).
- **Vulnerability Detection Rate:** Burp Suite had the highest vulnerability detection rate at 35.11%, followed by ZAP at 34.91% and Acunetix at 22.83%.
- **Ease of configuration:** All three tools were easy to configure.
- **Technical Support:** All three tools offers technical support.
- **True positive rate:** Burp Suite had the highest true positive rate at 35.12%, followed by ZAP at 31% and Acunetix at 21.6%. Figure 5.4 presents these results in a bar graph format.
- **False positive Rate:** Burp Suite had the lowest false positive rate at 0%, followed by Acunetix at 1.28% and ZAP at 4%. Figure 5.3 presents these results in a bar graph format.
- **True Negative rate:** Burp Suite had the highest true negative rate at 100%, followed by Acunetix at 98.7% and ZAP at 96%. Figure 5.4 presents these results in a bar graph format.
- **False negative rate:** Burp Suite had the lowest false negative rate at 64.87%, followed by ZAP at 68.8% and Acunetix at 78.3%. Figure 5.3 presents these results in a bar graph format.
- **F-measure:** Burp Suite achieved the highest F-measure at 51.8%, indicating a balanced performance between precision and recall. ZAP followed with an F-measure of 45.9%, while Acunetix had the lowest F-measure at 34.3%.
- **Accuracy:** Burp Suite achieved the highest accuracy at 66.4%. ZAP followed with an accuracy of 62.51%, while Acunetix had the lowest accuracy at 58.9%.
- **Recall:** Burp Suite had the highest recall rate at 35%, indicating its ability to correctly identify vulnerabilities when they were present. ZAP followed with a recall of 31%, while Acunetix had the lowest recall rate at 21.6%.

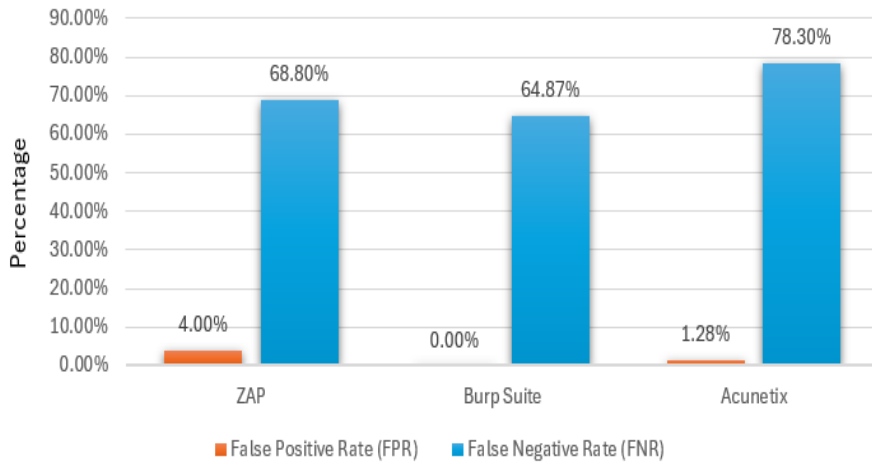


Figure 5.3: Results of Tools for FPR and FNR

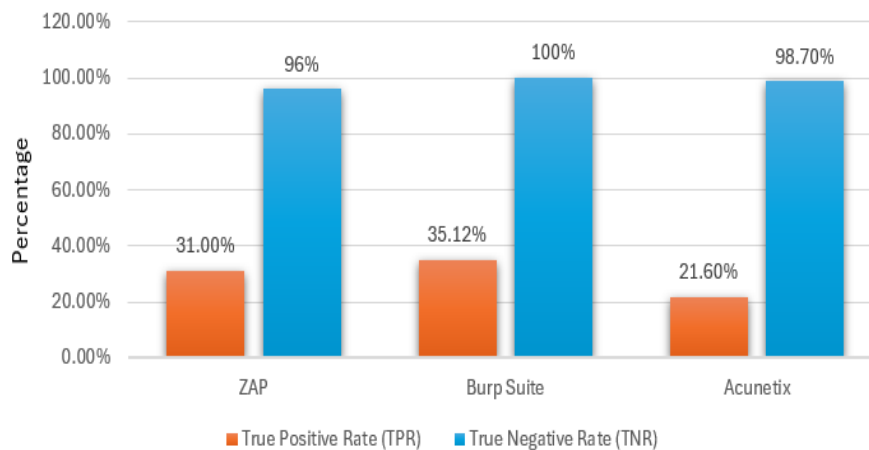


Figure 5.4: Results of Tools for TPR and TNR

- Precision:** Burp Suite achieved perfect precision at 100%, indicating that when it flagged a vulnerability, it was almost always correct. Acunetix followed with a precision of 94.7%, while ZAP had a precision of 89%.
- Matthews Correlation Coefficient(MCC):** Burp Suite achieved the highest MCC at 45.5%, indicating a strong correlation between its predictions and the actual outcomes. ZAP followed with an MCC of 35.31%, while Acunetix had the lowest MCC at 31.5%.
- Specificity:** Burp Suite had perfect specificity at 100%, indicating its ability to correctly identify non-vulnerabilities. Acunetix followed with a specificity of 98.7%, while ZAP had a specificity of 96%.
- Fx Score:** Burp Suite achieved the highest Fx Score at 51.9%. ZAP followed with an Fx Score of 46.2%, while Acunetix had the lowest Fx Score at 35.21%.

- **Markedness:** Burp Suite achieved perfect markedness at 100%, reflecting the quality of its positive predictions. Acunetix followed with a markedness of 93.4%, while ZAP had a markedness of 85.2%.
- **Informedness:** Burp Suite had the highest informedness at 35.1%, reflecting the effectiveness of the tool in making correct positive and negative predictions. ZAP followed with an informedness of 27.1%, while Acunetix had the lowest informedness at 20.3%.
- **Customize rules:** All three tools can customize rules for scanning.
- **Authenticate scans:** All three tools can perform authenticate scans.
- **API security testing:** All three tools can perform API security testing.
- **Comparing between scans:** All three tools provide a comparison between scans.
- **Integration capabilities:** All three tools can integrate with other tools.

In conclusion, our analysis suggests that Burp Suite emerges as the effective tool among the three, exhibiting the highest true positive rate, lowest false positive rate, and achieving a balanced precision and recall. Therefore, for users beginning the selection process of a WAS testing tool, Burp Suite stands out as a promising option.

Table 5.2: Results of ZAP, Burp suite and Acunetix for OSWAP benchmark

| <b>Metric</b>                               | <b>ZAP</b>                        | <b>Burp suite</b>          | <b>Acunetix</b>  |
|---|-----------------------------------|----------------------------|--|
| True positives                              | 441                               | 497                        | 306  |
| False Negatives                             | 974                               | 918                        | 1109   |
| False positives                             | 53                                | 0                          | 17   |
| True negatives                              | 1272                              | 1325                       | 1308   |
| Types of different vulnerabilities detected | 45                                | 15                         | 5  |
| Time taken to scan                          | 17hrs 21min                       | 4hrs 18min                 | 129min   |
| Number of vulnerabilities detected          | 494                               | 497                        | 323  |
| Tool cost                                   | Free Tool                         | Commercial Tool            | Commercial Tool  |
| Crawling Type                               | Active and Passive                | Active and Passive         | Active and Passive   |
| Protocol Support                            | 11                                | 11                         | 11   |
| Automation Level                            | High                              | Medium                     | High   |
| Reporting Features                          | different types of UI for reports | standard reports HTML, PDF | different types of UI for reports, along with compliance standard reports. |
| Interface                                   | GUI                               | GUI                        | GUI  |
| Vulnerability Detection Rate                | 34.91%                            | 35.11%                     | 22.83%   |
| Ease of configuration                       | Easy                              | Easy                       | Easy   |
| Technical Support                           | Yes                               | Yes                        | Yes  |
| True positive rate                          | 31%                               | 35.12%                     | 21.6%  |
| False positive Rate                         | 4%                                | 0%                         | 1.28%  |
| True Negative rate                          | 96%                               | 100%                       | 98.7%  |
| False negative rate                         | 68.8%                             | 64.87%                     | 78.3%  |
| F-measure                                   | 45.9%                             | 51.8%                      | 34.3%  |
| Accuracy                                    | 62.51%                            | 66.4%                      | 58.9%  |
| Recall                                      | 31%                               | 35%                        | 21.6%  |
| Precision                                   | 89%                               | 100%                       | 94.7%  |
| Matthews Correlation Coefficient            | 35.31%                            | 45.5%                      | 31.5%  |
| Specificity                                 | 96%                               | 100%                       | 98.7%  |
| Fx Score                                    | 46.2%                             | 51.9%                      | 35.21%   |
| Markedness                                  | 85.2%                             | 100%                       | 93.4%  |
| Informedness                                | 27.1%                             | 35.1%                      | 20.3%  |
| Customize rules                             | Yes                               | Yes                        | Yes  |
| Authenticate scans                          | Yes                               | Yes                        | Yes  |
| API security testing                        | Yes                               | Yes                        | Yes  |
| Comparing between scans                     | Yes                               | Yes                        | Yes  |
| Integration capabilities                    | Yes                               | Yes                        | Yes  |

This study aims to identify key metrics for comprehensively comparing and evaluating web application security (WAS) testing tools to guide effective tool selection. We addressed three main research questions using three methods: literature review, interviews, and experiment.

We conducted a literature review using the snowballing technique to address the first research question **RQ1: What are the metrics employed for comparing web application security (WAS) testing tools?** This allowed us to gather a comprehensive set of 37 metrics that have been used to assess and compare WAS testing tools, as highlighted in Table 3.1. These metrics covered both technical factors, such as accuracy in finding vulnerabilities, and non-technical aspects, like tool cost and integration capabilities. However, the literature review revealed that most previous studies did not provide clear explanations for their choice of specific evaluation metrics. We also selected the tools for testing in our experiment (ZAP, Burp Suite, and Acunetix) as they are stated as the best tools by most of the authors in the literature review when they conducted their evaluations.

To build on the literature review findings, we conducted semi-structured interviews with industry experts to answer our **RQ2: Among the metrics identified in RQ1, which ones are considered important, and what are the reasons for their significance?** We gained insights into the key metrics they consider crucial when evaluating and selecting WAS testing tools. The expert input helped validate and prioritize the initial list of metrics from the literature review. The experts unanimously emphasized the critical significance of metrics such as true positive rate (TPR), false positive rate (FPR), and false negative rate (FNR) as the primary technical evaluation criteria. They explained that these metrics directly reflect a tool's ability to accurately identify and report vulnerabilities, which is the core purpose of WAS testing. A high TPR, low FPR, and low FNR are essential for ensuring the effectiveness and trustworthiness of the testing process.

The experts mentioned that while derived metrics like recall, accuracy, F-measure, and Matthews Correlation Coefficient are important when conducting a one-time tool evaluation, they are not as crucial in their day-to-day decision-making process. The expert's reasoning for this prioritization was based on their practical experience and the specific needs of their organizations.

In addition to the technical metrics identified in the literature review, the experts also highlighted the importance of non-technical factors when comparing WAS testing tools. These non-technical metrics included attributes such as ease of use, reporting capabilities, integration with other security tools, etc. The experts em-

phasized that these non-technical factors can also significantly impact the overall efficiency and workflow integration of the WAS testing process within an organization. User-friendly tools, that provide comprehensive reporting, and seamlessly integrate with an organization's existing security infrastructure are more likely to see long-term adoption and effective usage.

Interestingly, the experts considered some metrics, such as alignment with the OWASP Top 10, scan logs, platform compatibility and browser compatibility, as less crucial for their specific needs. They argued that while these metrics may be relevant in certain contexts, they do not directly reflect the core performance and effectiveness of the security testing tools. The experts emphasized that their primary focus is on the tool's ability to accurately identify and report vulnerabilities, as these are the most critical factors for enhancing their organization's web application security posture.

The expert's decision to lessen the significance of these metrics came from their practical experiences. They explained that alignment with the OWASP Top 10 may not always be a reliable indicator of a tool's capabilities, as the Top 10 list can change over time and may not cover all the relevant vulnerabilities for their applications. Similarly, they considered scan logs and platform/browser compatibility as more of a basic requirement rather than a distinguishing factor when choosing a WAS testing tool. In addition to highlighting the importance of various metrics, the experts also discussed situations where certain metrics might conflict. In such cases, the experts explained they would need to prioritize the metrics based on their specific organizational needs and the WAS testing context.

One notable finding from the interviews was the importance of metrics that were not frequently mentioned by the authors in the literature, such as False Negatives (FN), True Negative Rate, technical support provided by the tool vendors, interface, and the automation level of the tools. The experts highlighted these metrics as crucial in their practical decision-making process, emphasizing that a tool's ability to minimize false negatives and provide robust technical support are key factors in ensuring the effectiveness and long-term adoption of WAS testing tools within their organizations. So, we also consider these metrics important and recommend practitioners use them when evaluating WAS testing tools.

The insights gained from the interviews, combined with the literature review, provided a deeper understanding of the relative significance and reasoning behind different evaluation metrics. After validating the initial list with experts, the final set of key metrics totalled 35, as highlighted in Table 4.2.

To address the third research question **RQ3: Which WAS testing tool demonstrates better performance based on the metrics obtained from RQ2?**, we conducted an experimental evaluation of selected WAS testing tools (ZAP, Burp Suite, and Acunetix) using the OWASP Benchmark Project. This allowed us to quantitatively assess the performance of these tools based on the 35 validated key metrics from the previous steps. We selected these tools for testing in our experiment (ZAP, Burp Suite, and Acunetix) as they are stated as the best tools by most of the authors in the literature review when they conducted their evaluations.

The results revealed that Burp Suite outperformed the other tested tools across the evaluation metrics. Compared to the other tools, Burp Suite demonstrated a relatively high true positive rate (TPR) in detecting vulnerabilities, while also main-

taining an impressively low false positive rate (FPR) of 0%. This indicates that Burp Suite has an excellent ability to minimize false positives, reducing the need for unnecessary investigation and remediation efforts.

ZAP showed a lower TPR than Burp Suite and also had an elevated FPR, which could lead to more time-consuming false positive assessments. Acunetix, on the other hand, exhibited a more balanced performance, with a moderate TPR and FPR. Overall, the results clearly show that Burp Suite provided the best performance across the validated set of evaluation metrics. Its combination of high TPR and extremely low FPR makes it a highly reliable and effective choice for WAS testing, outshining the other tools evaluated in this study.

These findings provide valuable insights for practitioners when selecting the suitable WAS testing tool. Organizations with a strong focus on reducing false positives may find Burp Suite to be a more appropriate choice, while those prioritizing comprehensive vulnerability detection may lean towards ZAP, despite its higher FPR. Per the experimental analysis, Burp Suite performed better, so for users beginning the selection process of a WAS testing tool, Burp Suite stands out as a promising option.

It's important to mention that we encountered some challenges during our experiments when using the Burp Suite tool for scanning. To overcome these challenges, we reached out to an author of a relevant research paper mentioned in the literature review [28] for guidance. With their help, we were able to use the tool successfully.

Overall, this research has provided a comprehensive understanding of the important metrics to consider when evaluating and comparing WAS testing tools. By combining literature reviews and expert input we identified and validated a set of 35 key metrics spanning technical and non-technical factors. The study's findings offer a foundation for future benchmarking efforts and tool comparisons in research. For practitioners, our results provide guidance on what metrics to consider when selecting and comparing tools for effective WAS testing.

## 6.1 Threats to validity

### 6.1.1 Internal Validity Threats

Certain factors regarding internal validity may influence our study's results. One potential threat is the consistency of our experimental setup. While we took great care to maintain consistent conditions during the testing of different WAS testing tools, variations in environmental factors or unforeseen technical issues could impact the internal validity of our findings.

One potential threat is the Selection Bias of interview participants. This occurs if the chosen participants do not adequately represent a diverse range of perspectives and experiences in the field of WAS testing. To mitigate this threat, we ensured the inclusion of relevant participants in the interviews. Our participant selection process encompassed individuals with various backgrounds in WAS testing and experience levels ranging from 8 to 30 years.

The comprehensiveness and representativeness of the OWASP Benchmark Project in covering real-world web application vulnerabilities could be a threat to the valid-

ity of the experimental results. The OWASP Benchmark Project is designed as a standardized test suite to assess the accuracy of WAS testing tools, but it may not fully capture the diversity and complexity of vulnerabilities found in real-world web applications. As a result, the performance of the WAS testing tools evaluated using the OWASP Benchmark Project may not fully reflect their effectiveness in detecting and addressing vulnerabilities in actual web applications.

### 6.1.2 External Validity Threats

The performance and capabilities of the tools evaluated in this thesis may have changed since the research was conducted, limiting the longer-term applicability of the findings from the experiment. WAS testing tools are subject to frequent updates and improvements. As a result, the relative strengths and weaknesses of the tools assessed in this study may have shifted over time. To address this threat to external validity, it would be prudent to acknowledge the dynamic nature of the WAS testing tool market and recommend periodic re-evaluation of the tools, perhaps through follow-up studies. Additionally, we encourage readers to verify the current state of the tools and consider the potential impact of any significant changes that may have occurred since the research was conducted.

### 6.1.3 Construct Validity Threats

Metrics like "ease of configuration", and "automation level" may be more subjective and could be interpreted differently by different stakeholders. These types of metrics can be challenging to measure and compare objectively across various WAS testing tools. The assessment of ease of configuration or ease of use may depend on the individual preferences and experiences of the users evaluating the tools. To mitigate this threat to construct validity, it would be important to establish clear, standardized criteria for evaluating these subjective metrics in future, potentially incorporating input from a diverse set of users.

### 6.1.4 Conclusion Validity Threats

The experiment was conducted in a particular environment, which may restrict the applicability of the findings to other settings. The effectiveness of automated WAS testing tools could differ in various environments. Therefore, it's important to interpret the conclusions within the context of this study's specific conditions.

## 7.1 Conclusions

This thesis set out to identify key metrics for evaluating and comparing WAS testing tools, to guide the selection of the most effective tools for WAS testing.

As highlighted in the introductory chapter, robust WAS testing tools are critical in protecting web applications from cyber threats, and organizations need a well-defined set of evaluation and comparing criteria to make informed decisions about tool selection. This research aimed to address this need by identifying the key metrics for comparing WAS testing tools, to provide a well-defined set of evaluation criteria to guide practitioners.

To address the first research question RQ1, we conducted a literature review. As seen in Table 3.1, through this comprehensive review of the existing research, we were able to compile a detailed set of technical and non-technical metrics commonly used for assessing the capabilities of WAS testing tools. These included accuracy metrics like TPR, FPR, and FNR, as well as other important factors such as ease of use, reporting capabilities, and automation level.

To validate the significance of these identified metrics, as per RQ2, we conducted semi-structured interviews with security experts and practitioners who had 8 to 30 years of experience. As discussed in Section 4.3, the interviews provided valuable validation and insights into the relative importance of these metrics. The experts emphasized that accuracy metrics are critical in evaluating a tool's ability to effectively identify and report vulnerabilities - the primary purpose of WAS testing. They also highlighted the significance of non-technical factors such as usability and reporting, which can impact the overall efficiency and practicality of deploying these tools in real-world environments. After validating the initial list from the literature review with experts, the final set of key metrics totaled 35, as highlighted in Table 4.2.

Building on these findings, we performed an experiment to compare three WAS testing tools - ZAP, Burp Suite, and Acunetix - using the OWASP Benchmark project(RQ3). The results, summarized in Table 5.2, revealed notable differences in the performance of these tools across the key metrics.

Notably, the analysis showed that Burp Suite performance was considerably effective, particularly in terms of its low FPR of 0%, compared to 4% for ZAP and 1.28% for Acunetix. This aligns with the feedback from the security experts, who stressed the crucial importance of FPR as a critical accuracy metric in evaluating the effectiveness of WAS testing tools.

In conclusion, this research has provided a valid set of metrics for comparing and evaluating WAS testing tools, empowering organizations to make more informed decisions that align with their specific security requirements and constraints. Security professionals can optimise their WAS testing tool selection by understanding the key metrics and their relative significance, as established through the literature and interviews. Based on the experimental analysis, it is evident that Burp Suite performed better than other tools. Therefore, Burp Suite stands out as a good choice for users initiating a WAS testing tool selection process. It provides strong capabilities and dependable performance.

## 7.2 Future Work

This research has provided valuable insights into the key metrics for evaluating and comparing WAS testing tools. However, there are several ways to build upon the findings of this study.

One possible direction is to evaluate more tools. This study focused on three main WAS testing tools - OWASP ZAP, Burp Suite, and Acunetix. Future research could look at a broader range of both commercial and open-source WAS testing tools. Evaluating more tools would help us better understand the capabilities within the WAS testing field. Security practitioners can make more informed decisions about which tools best meet their organizational needs by assessing more options.

Another valuable approach is to conduct long-term studies. As web application attack methods and tool functionalities change over time, long-term analyses could track WAS testing tools' performance and evolving capabilities. This approach would highlight trends, shifts in tool strengths, and new vulnerabilities that need to be addressed. Such evaluations would give security teams a better understanding of the evolving WAS testing tool ecosystem.

By pursuing these future research directions, the security community can continue to improve knowledge about WAS testing. These efforts will help organizations choose and use appropriate WAS testing tools, strengthening the security of web applications against the ever-changing threat landscape.

---

## References

- [1] “Owasp website.” [Online]. Available: <https://owasp.org/www-project-benchmark/>
- [2] “Owasp, information on sast, dast, iast,” 2024. [Online]. Available: <https://owasp.org/www-project-devsecops-guideline/latest/02c-Interactive-Application-Security-Testing>
- [3] “Owasp web application security testing,” 2024. [Online]. Available: [https://owasp.org/www-project-web-security-testing-guide/latest/4-Web\\_Application\\_Security\\_Testing/](https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/)
- [4] K. Abdulghaffar, N. Elmrabit, and M. Yousefi, “Enhancing web application security through automated penetration testing with multiple vulnerability scanners,” *Computers*, vol. 12, no. 11, 2023. [Online]. Available: <https://www.mdpi.com/2073-431X/12/11/235>
- [5] H. S. Abdullah, “Evaluation of open source web application vulnerability scanners,” *Academic Journal of Nawroz University*, vol. 9, no. 1, pp. 47–52, 2020.
- [6] A. Al Anhar and Y. Suryanto, “Evaluation of web application vulnerability scanner for modern web application,” in *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*. IEEE, 2021, pp. 200–204.
- [7] S. Alassmi, P. Zavorsky, D. Lindskog, R. Ruhl, A. Alasiri, M. Alzaidi *et al.*, “An analysis of the effectiveness of black-box web application scanners in detection of stored xssi vulnerabilities,” *International Journal of Information Technology and Computer Science*, vol. 4, no. 1, 2012.
- [8] S. Alazmi and D. C. De Leon, “A systematic literature review on the characteristics and effectiveness of web application vulnerability scanners,” *IEEE Access*, 2022.
- [9] M. Albahar, D. Alansari, and A. Jurcut, “An empirical comparison of pen-testing tools for detecting web app vulnerabilities,” *Electronics*, vol. 11, no. 19, p. 2991, 2022.
- [10] M. Alsaleh, N. Alomar, M. Alshreef, A. Alarifi, and A. Al-Salman, “Performance-based comparative assessment of open source web vulnerability scanners,” *Security and Communication Networks*, vol. 2017, 2017.
- [11] M. Althunayyan, N. Saxena, S. Li, and P. Gope, “Evaluation of black-box web application security scanners in detecting injection vulnerabilities,” *Electronics*, vol. 11, no. 13, p. 2049, 2022.

- [12] R. Amankwah, J. Chen, P. K. Kudjo, and D. Towey, “An empirical comparison of commercial and open-source web vulnerability scanners,” *Software: Practice and Experience*, vol. 50, no. 9, pp. 1842–1857, 2020.
- [13] K. Anagandula and P. Zavorsky, “An analysis of effectiveness of black-box web application scanners in detection of stored sql injection and stored xss vulnerabilities,” in *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*, 2020, pp. 40–48.
- [14] N. Antunes and M. Vieira, “Assessing and comparing vulnerability detection tools for web services: Benchmarking approach and examples,” *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 269–283, 2014.
- [15] V. M. Antunes, Nuno, “On the metrics for benchmarking vulnerability detection tools,” in *2015 45th Annual IEEE/IFIP international conference on dependable systems and networks*. IEEE, 2015, pp. 505–516.
- [16] Daud, N. Izyani, Bakar, K. A. Abu, Hasan, and M. S. Md, “A case study on web application vulnerability scanning tools,” in *2014 Science and Information Conference*. IEEE, 2014, pp. 595–600.
- [17] S. Disawal and U. Suman, “An analysis and classification of vulnerabilities in web-based application development,” in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2021, pp. 782–785.
- [18] A. Doupé, M. Cova, and G. Vigna, “Why johnny can’t pentest: An analysis of black-box web vulnerability scanners,” in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2010, pp. 111–131.
- [19] M. El, E. McMahon, S. Samtani, M. Patton, and H. Chen, “Benchmarking vulnerability scanners: An experiment on scada devices and scientific instruments,” in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 83–88.
- [20] A. M. Ferreira and H. Kleppe, “Effectiveness of automated application penetration testing tools,” Master dissertation., Master Education SNE/OS3, University of Amsterdam . . . , 2011.
- [21] S. Idrissi, N. Berbiche, F. Guerouate, and M. Shibi, “Performance evaluation of web application security scanners for prevention and protection against vulnerabilities,” *International Journal of Applied Engineering Research*, vol. 12, no. 21, pp. 11 068–11 076, 2017.
- [22] A. Ishizaka and S. Siraj, “Are multi-criteria decision-making tools useful? an experimental comparative study of three methods,” *European Journal of Operational Research*, vol. 264, no. 2, pp. 462–471, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377221717304885>
- [23] P. Jarupunphol, S. Seatun, and W. Buathong, “Measuring vulnerability assessment tools’ performance on the university web application.” *Pertanika Journal of Science & Technology*, vol. 31, no. 6, 2023.

- [24] C. Joshi and U. K. Singh, "Performance evaluation of web application security scanners for more effective defense," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 6, no. 6, pp. 660–667, 2016.
- [25] F. Kagorora, J. Li, D. Hanyurwimfura, and L. Camara, "Effectiveness of web application security scanners at detecting vulnerabilities behind ajax/json," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 4, no. 6, pp. 4179–4188, 2015.
- [26] C. Kalaani, "Owasp zap vs snort for sqli vulnerability scanning," 2023.
- [27] N. Khoury, P. Zavarisky, D. Lindskog, and R. Ruhl, "Testing and assessing web vulnerability scanners for persistent sql injection attacks," in *proceedings of the first international workshop on security and privacy preserving in e-societies*, 2011, pp. 12–18.
- [28] E. Lavens, P. Philippaerts, and W. Joosen, "A quantitative assessment of the detection performance of web vulnerability scanners," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–10.
- [29] A. Lis, "Comparison and analysis of web vulnerability scanners," B.S. thesis, 2019.
- [30] M. Maguire and B. Delahunt, "Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars." *All Ireland journal of higher education*, vol. 9, no. 3, 2017.
- [31] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1. IEEE, 2015, pp. 399–402.
- [32] F. Mateo Tudela, J.-R. Bermejo Higuera, J. Bermejo Higuera, J.-A. Sicilia Montalvo, and M. I. Argyros, "On combining static, dynamic and interactive analysis security testing tools to improve owasp top ten security vulnerability detection in web applications," *Applied Sciences*, vol. 10, no. 24, p. 9119, 2020.
- [33] Matti and Erik, "Evaluation of open source web vulnerability scanners and their techniques used to find sql injection and cross-site scripting vulnerabilities," 2021.
- [34] B. Mburano and W. Si, "Evaluation of web vulnerability scanners based on owasp benchmark," in *2018 26th International Conference on Systems Engineering (ICSEng)*. IEEE, 2018, pp. 1–6.
- [35] K. McQuade, "Open source web vulnerability scanners: the cost effective choice?" in *Proceedings of the Conference for Information Systems Applied Research*, vol. 2167, 2014, p. 1508.
- [36] R. Mohammed, "Assessment of web scanner tools," *International Journal of Computer Applications*, vol. 133, no. 5, pp. 1–4, 2016.
- [37] OWASP, "Vulnerability scanning tools," 2024. [Online]. Available: [https://owasp.org/www-community/Vulnerability\\_Scanning\\_Tools](https://owasp.org/www-community/Vulnerability_Scanning_Tools)

- [38] B. Pala, L. Pisu, S. L. Sanna, D. Maiorca, G. Giacinto *et al.*, “A targeted assessment of cross-site scripting detection tools,” in *CEUR WORKSHOP PROCEEDINGS*, 2023.
- [39] M. Qasaimeh, A. Shamlawi, and T. Khairallah, “Black box evaluation of web application scanners: Standards mapping approach,” *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 14, pp. 4584–4596, 2018.
- [40] U. Ravindran and R. V. Potukuchi, “A review on web application vulnerability assessment and penetration testing.” *Review of Computer Engineering Studies*, vol. 9, no. 1, 2022.
- [41] M. Rennhard, D. Esposito, L. Ruf, and A. Wagner, “Improving the effectiveness of web application vulnerability scanning,” *International Journal on Advances in Internet Technology*, vol. 12, no. 1/2, pp. 12–27, 2019.
- [42] Y. Sadqi and Y. Maleh, “A systematic review and taxonomy of web applications threats,” vol. 31, no. 1. Taylor & Francis, 2022, pp. 1–27. [Online]. Available: <https://doi.org/10.1080/19393555.2020.1853855>
- [43] Saeed and F. Abbas, “Using wassec to evaluate commercial web application security scanners,” *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 4, no. 1, pp. 177–181, 2014.
- [44] F. A. Saeed and E. A. Elgabar, “Assessment of open source web application security scanners,” *Journal of Theoretical and Applied Information Technology*, vol. 61, no. 2, pp. 281–287, 2014.
- [45] D. Sagar, S. Kukreja, J. Brahma, S. Tyagi, and P. Jain, “Studying open source vulnerability scanners for vulnerabilities in web applications,” *IIOAB JOURNAL*, vol. 9, no. 2, pp. 43–49, 2018.
- [46] R. Samlı and Z. Orman, “A comprehensive overview of web-based automated testing tools,” *İleri Mühendislik Çalışmaları ve Teknolojileri Dergisi*, vol. 4, no. 1, pp. 13–28, 2023.
- [47] P. A. Sarpong, L. S. Larbi, D. Paa, I. B. Abdulai, R. Amankwah, and A. Ampomah, “Performance evaluation of open source web application vulnerability scanners based on owasp benchmark,” *International Journal of Computer Applications*, vol. 174, no. 02, 2021.
- [48] C. Seaman, “Qualitative methods in empirical studies of software engineering,” *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.
- [49] M. P. Shah, “Comparative analysis of the automated penetration testing tools,” Ph.D. dissertation, Dublin, National College of Ireland, 2020.
- [50] J. Shahid, M. K. Hameed, I. T. Javed, K. N. Qureshi, M. Ali, and N. Crespi, “A comparative study of web application security parameters: Current trends and future directions,” 2022.
- [51] N. Suteva, D. Zlatkovski, and A. Mileva, “Evaluation and testing of several free/open source web vulnerability scanners,” 2013.

- [52] Y.-H. Tung, S.-S. Tseng, J.-F. Shih, and H.-L. Shan, "A cost-effective approach to evaluating security vulnerability scanner," in *2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2013, pp. 1–3.
- [53] S. Tyagi and K. Kumar, "Evaluation of static web vulnerability analysis tools," in *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*. IEEE, 2018, pp. 1–6.
- [54] Vega, E. A. Armas, Orozco, A. L. Sandoval, Villalba, and L. J. García, "Benchmarking of pentesting tools," *International Journal of Computer and Information Engineering*, vol. 11, no. 5, pp. 602–605, 2017.
- [55] M. Vieira, N. Antunes, and H. Madeira, "Using web security scanners to detect vulnerabilities in web services," in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 566–571.
- [56] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.
- [57] C. Wohlin, P. R. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, "Experimentation in software engineering," 2012. [Online]. Available: [https://edisciplinas.usp.br/pluginfile.php/7713736/mod\\_resource/content/1/Wohlin.pdf](https://edisciplinas.usp.br/pluginfile.php/7713736/mod_resource/content/1/Wohlin.pdf)
- [58] B. Zukran and M. M. Siraj, "Performance comparison on sql injection and xss detection using open source vulnerability scanners," in *2021 International Conference on Data Science and Its Applications (ICoDSA)*. IEEE, 2021, pp. 61–65.

## Appendix A

---

# Supplemental Information

Table A.1: Start set

| Paper No. | Title   | Reference |
|-----------|---|-----------|
| 1         | Performance Evaluation of Web Application Security Scanners for Prevention and Protection against Vulnerabilities | [21]      |
| 2         | Evaluation of Web Application Vulnerability Scanner for Modern Web Application                                    | [6]       |
| 3         | Assessment of Web Scanner Tools   | [36]      |

Table A.2: 1st Iteration Forward Snowballing

| Paper No. | Title  | Reference |
|-----------|--|-----------|
| 4         | A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions   | [50]      |
| 5         | Evaluation of Web Vulnerability Scanners Based on OWASP Benchmark  | [34]      |
| 6         | On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications | [32]      |
| 7         | Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities   | [11]      |
| 8         | Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners  | [4]       |
| 9         | Performance Evaluation of Open Source Web Application Vulnerability Scanners based on OWASP Benchmark  | [47]      |
| 10        | Improving the Effectiveness of Web Application Vulnerability Scanning  | [41]      |
| 11        | A Quantitative Assessment of the Detection Performance of Web Vulnerability Scanners   | [28]      |
| 12        | OWASP ZAP vs Snort for SQLi Vulnerability Scanning   | [26]      |
| 13        | A Targeted Assessment of Cross-Site Scripting Detection Tools  | [38]      |
| 14        | Studying Open Source Vulnerability Scanners for Vulnerabilities in Web Applications  | [45]      |

Table A.3: 1st Iteration Backward Snowballing

| Paper No. | Title  | Reference |
|-----------|--|-----------|
| 15        | Effectiveness of Web Application Security Scanners at Detecting Vulnerabilities behind AJAX/JSON | [25]      |
| 16        | Effectiveness of Automated Application Penetration Testing Tools                                 | [20]      |
| 17        | Assessment of Open Source Web Application Security Scanners                                      | [44]      |
| 18        | Evaluation of web vulnerability scanners   | [31]      |
| 19        | Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners                    | [18]      |
| 20        | Open Source Web Vulnerability Scanners: The Cost Effective Choice?                               | [35]      |
| 21        | Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners               | [10]      |
| 22        | Evaluation of Open Source Web Application Vulnerability Scanners                                 | [5]       |
| 23        | Using WASSEC to Evaluate Commercial Web Application Security Scanners                            | [43]      |

Table A.4: 2nd Iteration Forward Snowballing

| Paper No. | Title   | Reference |
|-----------|---|-----------|
| 24        | An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities  | [9]       |
| 25        | A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners                             | [8]       |
| 26        | An empirical comparison of commercial and open-source web vulnerability scanners  | [12]      |
| 27        | Evaluation of open source web vulnerability scanners and their techniques used to find SQL injection and cross-site scripting vulnerabilities | [33]      |
| 28        | Measuring Vulnerability Assessment Tools' Performance on the University Web Application   | [23]      |

Table A.5: 2nd Iteration Backward Snowballing

| Paper No. | Title   | Reference |
|-----------|---|-----------|
| 29        | BLACK BOX EVALUATION OF WEB APPLICATION SCANNERS: STANDARDS MAPPING APPROACH                                      | [39]      |
| 30        | EVALUATION AND TESTING OF SEVERAL FREE/OPEN SOURCE WEB VULNERABILITY SCANNERS                                     | [51]      |
| 31        | Benchmarking vulnerability scanners: An experiment on SCADA devices and scientific instruments                    | [19]      |
| 32        | Assessing and Comparing Vulnerability Detection Tools for Web Services: Benchmarking Approach and Examples        | [14]      |
| 33        | Benchmarking of Pentesting Tools  | [54]      |
| 34        | Testing and Assessing Web Vulnerability Scanners for Persistent SQL Injection Attacks                             | [27]      |
| 35        | On the Metrics for Benchmarking Vulnerability Detection Tools   | [15]      |
| 36        | A case study on web application vulnerability scanning tools  | [16]      |
| 37        | An analysis of the Effectiveness of Black-box Web Application Scanners in Detection of Stored XSS Vulnerabilities | [7]       |
| 38        | A cost-effective approach to evaluating security vulnerability scanner  | [52]      |
| 39        | Using web security scanners to detect vulnerabilities in web services   | [55]      |

Table A.6: 3rd Iteration Forward Snowballing

| Paper No. | Title   | Reference |
|-----------|---|-----------|
| 40        | Performance Comparison on SQL Injection and XSS Detection using Open Source Vulnerability Scanners                | [58]      |
| 41        | An analysis of the Effectiveness of Black-box Web Application Scanners in Detection of Stored XSS Vulnerabilities | [13]      |
| 42        | A Comprehensive Overview of Web-Based Automated Testing Tools   | [46]      |
| 43        | Comparison and Analysis of Web Vulnerability Scanners   | [29]      |
| 44        | COMPARATIVE STUDY OF WEB APPLICATION PENETRATION TESTING TOOLS  | [52]      |

Table A.7: 3rd Iteration Backward Snowballing

| Paper No. | Title  | Reference |
|-----------|--|-----------|
| 45        | Comparative Analysis of the Automated Penetration Testing Tools              | [49]      |
| 46        | A Review on Web Application Vulnerability Assessment and Penetration Testing | [40]      |
| 47        | Evaluation of Static Web Vulnerability Analysis Tools                        | [53]      |



