École Polytechnique Fédérale de Lausanne

Application security: governance, tools and secure development practices, a focus on Security Champions and DAST tools

by Adrien Fermeli-Furic

Master Thesis

Approved by the Examining Committee:

Prof. Dr. sc. ETH Mathias Payer
Thesis Advisor

Hamza Boughemza
External Expert

Hamza Boughemza
Thesis Supervisor

EPFL IC IINFCOM HEXHIVE
BC 160 (Bâtiment BC)
Station 14
CH-1015 Lausanne

August 25, 2023

Acknowledgments

I would like to thank Hamza Boughemza, my thesis supervisor within the company, for his continuous guidance throughout the project and for his invaluable advice.

I would also like to express my gratitude to Enea Bell for sharing his expertise in application security with me.

I would also like to thank Mathias Payer, my professor, for supervising my master's thesis.

Lastly, I extend my heartfelt gratitude to Wavestone for granting me this internship opportunity and permitting me to undertake this master thesis on application security.

Lausanne, August 25, 2023 Adrien Fermeli-Furic

Abstract

This project delves into two primary concerns of DevSecOps: the selection of suitable DAST tools and the effective implementation of a Security Champions program.

Existing studies on DAST tools often lack transparency. Our research, through a detailed comparative study of five open-source DAST tools, reveals that most vulnerabilities are missed by the tools. Besides, while tools like Owasp Zap performed significantly better than others, no single tool identified all the vulnerabilities of others, suggesting the potential merit of tool combinations. Ultimately, the study offers a ranking of the tools, but it also objectively provides all the necessary data to allow individuals to choose the tool that is best suited to their specific needs.

On the organizational front, while most resources provide surface-level advice and don't cover the whole picture, our interactive guide offers actionable insights for every stage of a Security Champions program.
The guide provides actionable solutions to the main challenges in implementing a Security Champions program, emphasizing the importance of management buy-in, champion recruitment, effective training, and sustained motivation. Notably, it champions the selection of passionate and communicative individuals, clear KPI establishment, and regular feedback, all presented in an interactive format distinct from conventional paper-based resources.

This dual contribution aims to equip application security professionals with robust data on DAST tool performance and a comprehensive guide for championing security within development teams.

Contents

Chapter 1

Introduction

Application security is an expansive and intricate domain. It encompasses technical challenges, such as integrating security tools into the CI/CD pipeline, as well as human-oriented challenges like fostering a security culture within the development team. In our project, we endeavored to address both these facets. However, given the breadth of these areas, we had to refine our focus. We honed in on prevalent dilemmas companies encounter when transitioning to DevSecOps. From a technical standpoint, the pivotal question was: which DAST tool to choose ? Organizationally, the key query was: how can we effectively conduct a Security Champions program?

From a technical standpoint, the market is flooded with numerous DAST tools, making the selection process challenging. Security professionals often express dissatisfaction with their DAST tools due to various reasons: a high number of false positives, overlooked vulnerabilities, low report quality, and extended scanning durations.
After a DAST tool completes its scan of an application, its report requires manual analysis by a security team member. An abundance of false positives means more time spent by the security personnel sifting through the tool's output, leading to wasted hours and financial implications for the company.
Additionally, prolonged scanning times can financially strain a company by potentially causing delays in development cycles. Naturally, the primary objective of this security layer is to identify as many vulnerabilities as possible, emphasizing the importance of choosing an effective tool.

From an organizational perspective, the human element is pivotal in the successful implementation of DevSecOps. Security champions, when guided appropriately, can be instrumental in cultivating a security-centric culture within the development team. However, the reality often falls short of this ideal. Implementing a Security Champions program comes with its set of challenges. A prevalent issue is the dwindling motivation of security champions, even though they initially volunteered for the role. In the program's nascent stages, persuading management to allocate the

requisite resources can be another hurdle. While a well-executed Security Champions program can yield significant benefits for the team, a poorly managed one can drain both time and resources. Hence, it's imperative for application security professionals to receive guidance throughout every phase of their Security Champions initiative.

For both of these challenges, current research is insufficient.

There are several resources that give advice about security champions, including the OWASP playbook [26]. However, many of these resources offer surface-level an don't cover the whole picture, falling short of being comprehensive guides. One freely accessible guide is "The ultimate guide to building security champions" [3]. This guide shares a lot of good advice and solutions to conduct a securitiy champions program, but once again, I consider that this guide is not complete. A significant omission is its lack of discussion on KPIs or evaluation metrics for the program. Such metrics are crucial not only for gaining management's approval, ensuring the program's widespread adoption across teams, but also for providing valuable feedback to the champions. This feedback mechanism can motivate [30] champions and guide their continuous improvement.
Finally, in contrast to other resources on security champions that are presented in a traditional paper format, our guide is designed to be interactive, offering a more engaging and user-friendly experience.

On the other hand, DAST landscape features several online rankings, notably from platforms like Comparitech [6] and Trust Radius [31]. However, these rankings often suffer from a transparency deficit, presenting pros, cons, and tool rankings without sharing the underlying data. Moreover, such studies occasionally appear promotional, emphasizing free trials and specific solutions, which can raise questions about their objectivity and credibility.

To address these concerns, I propose the following two solutions:

In the realm of DAST tools, I undertook a detailed comparative study of five open-source DAST tools: Owasp Zap, Wapiti, Nikto, Golismero, and Skipfish. Unlike existing comparative studies on DAST tools, my study stands out due to its complete transparency and exclusive focus on open-source tools. I established seven distinct evaluation criteria, each carrying a significance weight. The data was collected by scanning two vulnerable web applications using these tools.
The primary insights from the study were somewhat concerning: a significant number of vulnerabilities went undetected by the tools. When combined, the five tools could only identify 22% of the vulnerabilities in the first web application and 60% in the second.
A notable observation was that while most of the tools efficiently detected all the XSS vulnerabilities across both web applications, none could pinpoint any SQL injection vulnerabilities, even though many of them claim to support detection of this vulnerability type. A description of the different vulnerability types is written in the Appendix.

In terms of individual performance, Owasp Zap outshone the others. However, it's worth noting that no single tool could detect all the vulnerabilities identified by the collective group, suggesting that there's potential value in utilizing a combination of these tools in certain scenarios.

In addressing the challenges of implementing a Security Champions program, I've crafted a comprehensive guide titled "How to Create a Successful Security Champions Program?".
This guide is tailored to assist individuals at any phase of their Security Champions journey, be it initiation, design, or maintenance. Organized chronologically, the guide is built around four pivotal challenges: gaining management buy-in, effective recruitment of security champions, providing robust training for the champions and ensuring sustained motivation among the champions. These challenges represent the core hurdles often encountered in the implementation of a Security Champions program.
The guide offers actionable advice and solutions to navigate these challenges. Key recommendations include selecting champions who exhibit a genuine passion for security and possess exemplary communication skills, establishing clear KPIs, and consistently offering feedback and rewards to champions to foster engagement and growth.
Distinctively, our guide is designed to be interactive, setting it apart from traditional paper-format resources on the topic.

This thesis offers two contributions to the field of application security.

The first contribution is the provision of detailed data regarding the performance of various open source DAST tools. I've constructed a ranking system for these tools, based on the weights I assigned to each evaluation criterion.
What sets this study apart from similar comparative analyses is the comprehensive disclosure of metrics and details measured during the scans. This approach facilitates the creation of a unique profile for each DAST tool, derived from its performance scores across the different evaluation criteria. Consequently, individuals can adjust the weights based on their specific priorities and recalibrate the rankings to suit their needs. This aspect of the study is particularly valuable for those seeking to select a DAST tool from the array of free options, especially considering the hefty price tags often associated with paid versions.
Beyond the comparison of the tools, this study also contributes to analyzing how good DAST tools perform on vulnerable web applications.

The thesis's second contribution centers on aiding application security professionals in the execution of their security champions program. As previously mentioned, the guide is structured to cater to individuals at any phase of their security champions initiative.
Each section of the guide stands alone, allowing users to directly access and focus on the segment most relevant to their current needs. This design ensures that users can swiftly identify a suite of

potential solutions tailored to address their specific challenges and subsequently implement these strategies to rectify their concerns.
Beyond offering actionable advice, the guide also serves a dual purpose as a maturity assessment tool in the realm of Security Champions, enabling professionals to gauge and enhance their program's effectiveness over time.

Chapter 2

Background

Before delving into the intricacies of DevSecOps, it's imperative to first grasp the concept of DevOps. DevOps [19] emerged as a response to the limitations of the Waterfall model, which was the prevailing approach to application development for a considerable period. The Waterfall model was a linear and sequential design approach where each phase depended on the deliverables of the previous one. This model had its merits but was often criticized for its rigidity and lack of adaptability to changes.

Historically, the development landscape was divided into two primary teams: the development (dev) team and the operations (ops) team. The dev team was primarily responsible for writing the code and ensuring its functionality, while the ops team focused on deployment, infrastructure management, and ensuring the application's availability to end-users. These two teams often operated in silos, with minimal communication and collaboration. This separation not only slowed down the software delivery process but also led to numerous challenges, especially when it came to integrating code into production environments.

It was against this backdrop that DevOps was introduced. DevOps sought to bridge the gap between development and operations, emphasizing collaboration, automation, and continuous integration. The core idea was to view the development pipeline as a continuous cycle rather than a linear waterfall. This shift facilitated faster development cycles and more frequent releases.

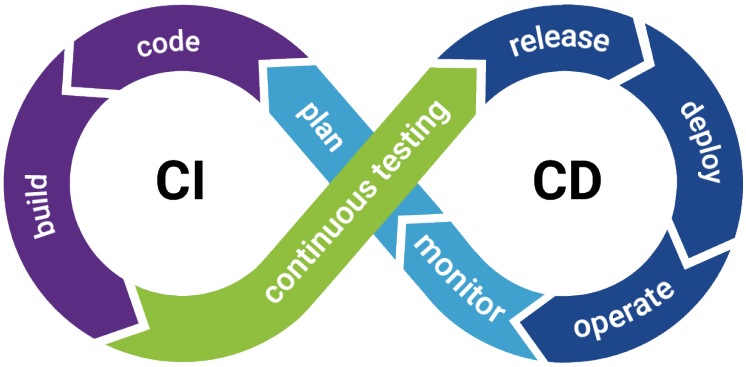

Figure 1: Diagram of the CI/CD pipeline

While DevOps brought about significant improvements in software delivery and collaboration, it inadvertently sidelined a critical aspect: security. In the rush to deploy rapidly, security checks were often postponed until the end of the development cycle. This approach posed several challenges:

First, vulnerabilities were only identified just before a new version was released. If any vulnerability was overlooked, it would be deployed directly, exposing systems to potential breaches.
Then, comprehensive security checks, including scans and penetration testing, are time-consuming. When juxtaposed with short development cycles and frequent releases, this led to a situation where, by the time a vulnerability was identified and fixed, multiple versions of the application, all carrying the same vulnerability, were queued up for deployment.

Recognizing these challenges, there was a clear need to "shift security to the left" in the development pipeline. This led to the emergence of DevSecOps. DevSecOps signifies an essential and logical progression in how development teams address security. DevSecOps promotes the idea that security should not be an afterthought. Instead, it should be integrated at every stage of the Continuous Integration/Continuous Deployment (CI/CD) pipeline. By doing so, security becomes a shared responsibility, ensuring that applications are not only functional and available but also secure from potential threats.

The main benefits of DevSecOps are the following:

Prevention of Security Incidents Before They Happen

By seamlessly integrating DevSecOps within the CI/CD toolchain [22], teams can proactively detect and resolve potential issues, ensuring they don't manifest in the production environment.

Faster Response to Security Issues

Continuous assessments in the DevSecOps model enhance the security focus, providing action-

able data. This data-driven approach empowers teams to make informed decisions regarding the security posture of applications, whether they're in development or gearing up for production.

Accelerated Feature Velocity [2]

Equipped with the right data and tools, DevSecOps teams can better anticipate and mitigate unforeseen risks. This proactive stance ensures a smoother development process and faster feature releases.

Lower Security Budget

The streamlined resources, solutions, and processes inherent in the DevSecOps approach simplify the development lifecycle. By designing with security in mind from the outset, organizations can achieve robust security without inflating their budgets.

Implementing DevSecOps comes with its set of challenges. On the technical side, there's the task of integrating automated security tools into the CI/CD pipeline. But there are also organizational and human challenges to consider. Just as DevOps was designed to improve communication and collaboration between development and operations teams, DevSecOps aims to enhance the interaction between the security team and both the development and operations teams. For DevSecOps to truly work, it's crucial to build a security-focused mindset across all teams. Security shouldn't be just the security team's responsibility; every team should prioritize it.

In this thesis, we chose to focus on one technical and one organizational aspect.

The technical aspect is the integration of DAST tools to the CI/CD pipeline.
Dynamic Application Security Testing (DAST) tools [21] are specialized software solutions designed to identify vulnerabilities in web applications while they are running in a live environment. Unlike static application security testing (SAST) tools, which analyze the source code, bytecode, or binary code of an application without executing it, DAST tools operate from the outside-in, simulating external cyberattacks.
DAST tools test applications in their running state, often in real-time, to identify vulnerabilities that may not be evident in the static code but become exploitable when the application is operational. They don't have access to the underlying source code, they are often referred to as "black box" testing tools. They assess the application much like an external attacker would, without knowledge of the internal workings.
DAST tools can detect a wide range of vulnerabilities, including those related to session management, authentication, data input validation, and other runtime processes.
Numerous DAST tools exist in the market, and to assist application security experts in selecting the most suitable one, we conducted a comparative analysis of open-source DAST tools.

Regarding the organizational aspect, we made a focus on Security champions programs.
A Security champion program is an initiative within organizations [26] where specific individuals, often from non-security teams, are designated as "champions" to promote and enhance security practices within their respective departments or teams. These champions act as a bridge between the security team and other departments, ensuring that security best practices are integrated into daily operations and projects. They are typically equipped with additional security training and are responsible for advocating for security, raising awareness about potential risks, and ensuring that their teams adhere to the organization's security policies.
By having these champions in place, organizations aim to create a culture of security awareness and responsibility across all levels and departments, ensuring a more proactive approach to cybersecurity.
Conducting a Security champion program is not an easy task, you might struggle with support from management, finding the right people to be security champions, or even keeping the champions motivated. To address these challenges effectively we implemented an interactive and complete guide called "How to create a successful Security champions program ?".

Chapter 3

Design

3.1 Open source DAST tools comparative study

3.1.1 Tools

In this subsection we introduce the tools that we will compare in the study. These tools are suggested by OWASP list of vulnerability scanning tools [28]. They have been selected based on online reputation and on compatibility and ease of installation on the study environment.

OWASP Zap

The OWASP Zed Attack Proxy (ZAP) stands as one of the most prevalent tools globally for Dynamic Application Security Testing (DAST). Being a free and open-source instrument, it enjoys active maintenance from a global team of volunteer contributors and has earned recognition as a top 1000 project on GitHub. It is also presented among the best DAST tools in the Magic Quadrant for Application Security Testing [21] of Gartner 2023.

With a concentration on the DAST component, ZAP is engineered to detect security flaws in web applications throughout the development and testing stages. It equips users with automated scanners and an assortment of tools for manual discovery of security vulnerabilities. ZAP generally search for widespread security problems such as:

- SQL and OS command injection flaws
- Cross-Site Scripting (XSS)

- Weaknesses in authentication and session management
- Insecure direct object references
- Misconfigured security settings
- Exposure of sensitive information
- Cross-Site Request Forgery (CSRF)
- Unchecked redirects and forwards

Wapiti

Wapiti is a freely available web application vulnerability scanner specializing in Dynamic Application Security Testing (DAST). Its purpose is to uncover various security weaknesses in web applications through simulated attacks.

A closer look at Wapiti's DAST functionality reveals:

Wapiti is capable of identifying an extensive array of vulnerabilities, encompassing but not restricted to:

- SQL Injection
- Cross-Site Scripting (XSS)
- Inclusion of Files
- Command Execution
- XXE (XML External Entity) assaults
- SSRF (Server-Side Request Forgery)
- CRLF Injection

The scanning procedure of Wapiti involves navigating through the web application to collect all URLs, forms, and input fields. Subsequently, it introduces payloads to probe for vulnerabilities. This approach differs from static analysis, as it doesn't examine the source code but rather engages with the live application.

Nikto

Nikto is a widely recognized open-source web server scanner, utilized predominantly for Dynamic Application Security Testing (DAST).

Categories of Vulnerabilities Detected:

- Server software that is no longer current
- Particular misconfigurations within the server
- Probable dilemmas with file permissions
- Files and programs that lack security
- XSS (Cross-Site Scripting)
- Various additional web server-related vulnerabilities

Nikto's operation involves conducting exhaustive assessments against web servers, scrutinizing multiple elements. This includes a search for over 6700 files and programs that could pose a threat, as well as an examination for outdated versions across more than 1250 servers and version-specific problems in over 270 servers.

Golismero

GoLismero is an open-source framework tailored for security testing, with an emphasis on web security.
GoLismero has the ability to gather and consolidate the outcomes of renowned tools like sqlmap, xsser, openvas, dnsrecon, theharvester, among others.
It aligns with established security benchmarks by integrating with standards such as CWE, CVE, and OWASP.
With a focus on web security, GoLismero's scanning abilities can be extended to conduct various types of examinations, encompassing vulnerability assessments for matters like SQL injection, XSS assaults, DNS exploration, and more.

Skipfish

Skipfish is a specialized automated web application security scanner, crafted with the intention of conducting security reconnaissance and aiding in security evaluations.

The vulnerabilities that Skipfish can detect in web applications encompass a wide range, including but not confined to:

- SQL Injection
- Cross-Site Scripting (XSS)
- Cross-Site Request Forgery (CSRF)
- Execution of server-side code
- Injection of shell commands
- Path traversal
- Flaws in authentication and authorization

Employing a heuristic methodology, Skipfish identifies vulnerabilities by constructing and sustaining an exhaustive site map throughout the scanning process. This method enables more discerning probing and minimizes the occurrence of false positives.

3.1.2 Targets

In the pursuit of a comprehensive comparative study of DAST tools, the research methodology necessitated the utilization of vulnerable web applications.
These applications, designed with intentional vulnerabilities, offer a realistic environment for evaluating the performance of various DAST tools. Such an approach enables the simulation of real-world scenarios, where security vulnerabilities are prevalent, providing a tangible measure of the effectiveness and efficiency of the tools under examination.

The selection of these vulnerable web applications was made with careful consideration of their diverse range of vulnerabilities. This diversity ensures a rigorous and exhaustive comparison, allowing for the discernment of the strengths and weaknesses of the tools being evaluated.
These two web applications that have been chosen are OWASP juice shop [25] and Google gruyere [20] that I will introduce in the next paragraphs.

Target 1 : OWASP Juice Shop

The OWASP Juice Shop [25] is a unique and intentionally insecure web application designed to resemble a small online shop selling fruit and vegetable juice products.

However, beneath its seemingly ordinary façade lies a complex platform containing 102 challenges of varying difficulty, where users are encouraged to exploit underlying security vulnerabilities. These vulnerabilities are intentionally planted in a manner that reflects real-life web development scenarios.

The Juice Shop's development began in September 2014 as a modern security training exercise environment, reflecting the reality of current web technology. Over the years, it has evolved, incorporating the latest web technologies and adding more vulnerabilities.
In September 2016, the Juice Shop was accepted as an OWASP Tool Project, and by July 2018, it reached the final Flagship maturity stage for OWASP projects.
The application serves not only as a training ground for hackers and security enthusiasts but also as a testing environment for penetration testing tools and automated security scanners.

It contains many types of vulnerabilities including XSS, Sensitive Data Exposure, Improper Input Validation, Broken Access Control, Vulnerable components, Broken, Authentication, Security through Obscurity, Insecure Deserialization, Broken Anti Automation, Injection, Security Misconfiguration and Cryptographic Issues. A description of vulnerability types is written in the Appendix. A summary of the proportion of each of these vulnerability categories is shown on Figure 2 below.

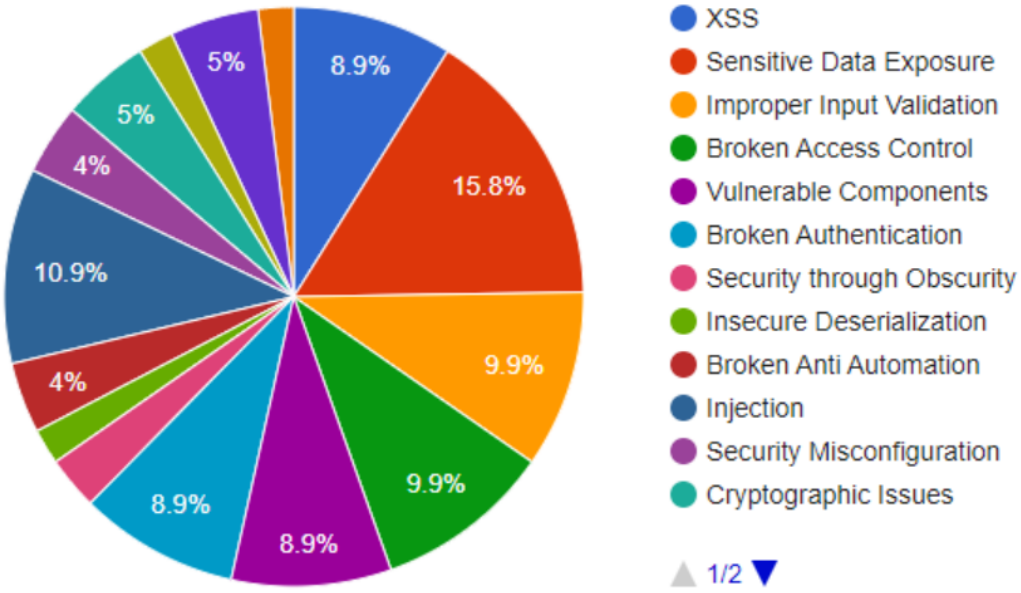

Figure 2: Categories of vulnerabilities in OWASP Juice Shop and their proportions

Target 2 : Google gruyere

Google Gruyere [20] is a small, intentionally insecure web application that serves as an educational platform for understanding web application vulnerabilities.
Created as a part of a codelab by Bruce Leban, Mugdha Bendre, and Parisa Tabriz, Gruyere allows users to publish snippets of text and store assorted files.
However, it is riddled with multiple security bugs that range from cross-site scripting (XSS) and cross-site request forgery (XSRF) to information disclosure, denial of service (DoS), and remote code execution.

The goal of Gruyere is to guide users through discovering these bugs and learning ways to fix them. The codelab is organized by types of vulnerabilities, and participants are encouraged to play the role of a malicious hacker to find and exploit the security bugs. The challenges within Gruyere can be approached through both black-box hacking, where users experiment with the application without access to the source code, and white-box hacking, where users have access to the source code and can analyze it to identify bugs.

Gruyere is designed to provide hands-on experience in understanding how an application can be attacked using common web security vulnerabilities and how to find, fix, and avoid these common vulnerabilities.

Google gruyere contains 20 vulnerabilities/challenges. A description of vulnerability types is written in the Appendix. A summary of the proportion of each of these vulnerability categories is shown on Figure 3 below.

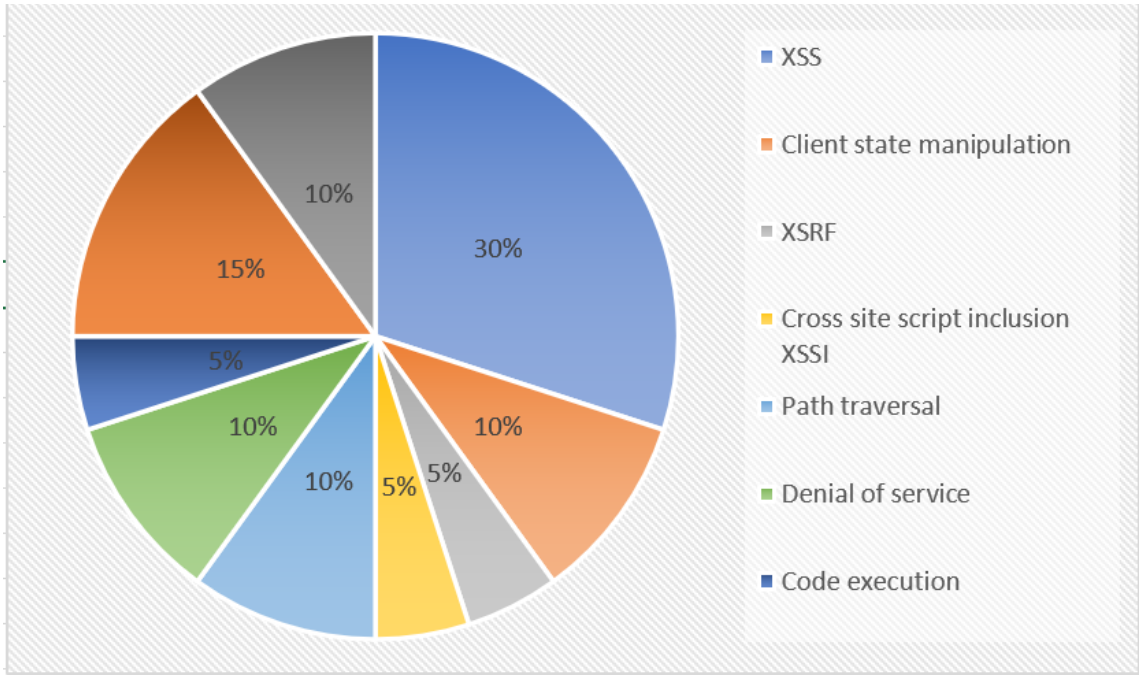

Figure 3: Categories of vulnerabilities in Google gruyere and their proportions

3.1.3 Evaluation criteria

Weights

In order to evaluate the tools, I chose 7 evaluation criteria that I will introduce below.
Each criteria has an weight from 1 to 5 which represents its importance. The final notation is the sum of the scores for each criterion multiplied by the weight of the criterion divided by the sum of weights.

Ease of use

Ease of use is a crucial factor when integrating a tool into the pipeline.
As highlighted by Gartner in their article "How to Select DevSecOps Tools for Secure Software Delivery" [5], one of their findings emphasizes that when selecting a tool, the ease of integration and the enhancement of the developer experience should be given as much importance as the tool's effectiveness.
However, this criterion has been assigned the lowest weight of 1 due to its subjective nature.
Additionally, familiarity with the tool over time can make it easier to use, reducing the initial challenges faced by developers.

Maintainability of the project

Maintainability is a significant factor to consider when evaluating a tool.
The ever-changing landscape of vulnerabilities in applications, as illustrated by the evolution of the OWASP Top Ten [27] from 2017 to 2021 in Figure 4 below, underscores the importance of tool maintainability. A tool that isn't well-maintained might not address current security concerns or could become obsolete over time.
However, this criterion is given a weight of 2, which is lower than some other criteria. This is because a tool that effectively identifies vulnerabilities in web applications retains its value, even if it isn't actively maintained

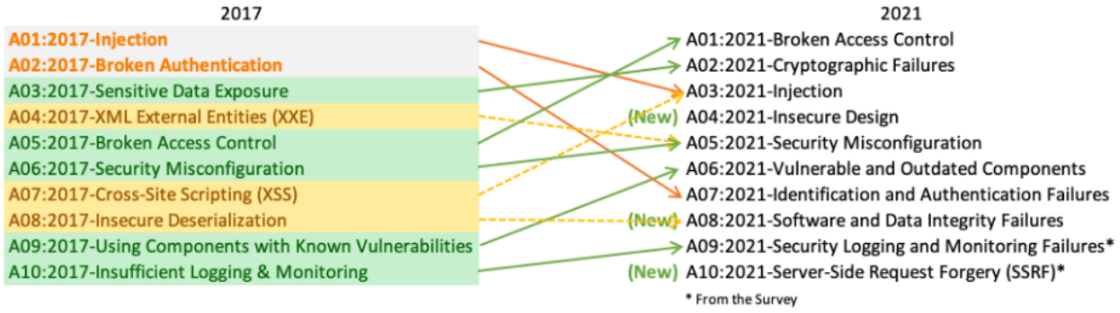

Figure 4 : Evolution of OWASP top ten from 2017 to 2021

Performance

Performance, in this context, refers to the duration a tool requires to scan a web application. This is a pivotal criterion, especially when the tool is integrated into a CI/CD pipeline. A swift tool ensures that the development cycle isn't unnecessarily prolonged. Indeed, one of the primary advantages of DevSecOps is the elimination of security [2] as a bottleneck, leading to shorter development cycles when security measures are shifted left. Hence, the emphasis on a tool's speed and efficiency. For these reasons, the weight of the performance criterion is 3.

Quality of Report

The quality of the report generated post-scan is of paramount importance.
After a tool completes its scan of a web application, it produces a report detailing its findings. This report is subsequently analyzed by developers or members of the security team. It's essential for the report to articulate its findings in a clear and comprehensible manner, enabling swift and easy understanding by the reader.
Additionally, the report should offer effective remediation suggestions for the identified issues, ensuring actionable insights.
Hence, the quality of report has a weight of 3.

False positives

False positives are a prevalent concern with DAST tools. This criterion is accorded the maximum weight of 5 due to the critical importance of minimizing false positives for an effective DAST tool. As previously mentioned, the reports generated by a DAST tool are scrutinized by developers or security team members. The time of these professionals is invaluable to the organization, and it should not be squandered on sifting through numerous false positives. Reducing false positives ensures that the team can focus on genuine threats and vulnerabilities, optimizing their efficiency and contribution.

True positives

The true positives criterion is given the highest weight of 5, emphasizing its significance. The number of true positives detected is a pivotal metric as it indicates the tool's capability to identify genuine vulnerabilities within the web application. An effective DAST tool should be adept at uncovering the majority of the real issues present in the web application, ensuring comprehensive security assessments and safeguarding against potential threats.

Vulnerabilities found

Given that the study focuses on vulnerable websites, assessing the range of vulnerabilities a tool can detect within these web apps is crucial. This criterion offers a distinct perspective compared to the true positives metric. While true positives indicate genuine vulnerabilities detected, the "vulnerabilities found" metric sheds light on the tool's ability to uncover a diverse array of vulnerabilities. Some true positives might lead to the discovery of a broader spectrum of vulnerabilities than others. This criterion is assigned the highest weight of 5, reflecting its paramount importance. The primary objective of a DAST tool is to identify and highlight critical vulnerabilities in a web application, ensuring robust security measures.

A summary of the weights of each criterion is shown below in table 3.1.

	Weigths
Maintainability of the project	2
Report quality	3
True positives	5
False positives	5
Number of vulnerabilities found	5
Performance	3
Ease of use	1
Total	24

Table 3.1: The caption without a number

Computation of the score

Each criterion is evaluated with a score from 0 to 5.

The simplest criteria scores to compute are for the 'Ease of Use' and 'Report Quality' criteria.
The 'Ease of Use' is directly assigned a grade from 0 to 5, which I choose based on my personal experience with the tool.
For the 'Report Quality' criterion, I first assign a score from 0 to 5 to two sub-criteria: the 'Remediation Guidance' score and the 'Clear Reporting of Findings' score. The weight of 1 is given to 'Remediation Guidance' and 2 to 'Clear Reporting of Findings'. Even though remediation guidance is an essential part of the report, it's more crucial for the reader to clearly understand the vulnerabilities present in the web application. Finally, I compute the 'Report Quality' score by averaging these two sub-criteria based on their respective weights.

For the other criteria, we adopt a distinct approach. Metrics such as the number of false positives, the number of vulnerabilities detected, and the scan duration don't naturally fall within a 0 to 5 range. To standardize these values and obtain scores within this range, we employ the Min–max normalization technique.
If we measure a value x for a specific criterion, its Min-max normalized value x' is calculated as follow:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

where x_{min} is the minimum value that a tool performed on this criterion, and x_{max} the maximum. Therefore we have $0 \leq x' \leq 1$ with $x' = 0$ for the tool with the minimum on the criterion and $x' = 1$ for the tool with the maximum.

For the criteria, there are two categories to consider. Specifically, for criteria where a larger value is more favorable, such as the 'Number of True Positives' and the 'Number of Vulnerabilities Found', we compute the score as follow:

$$Score = x' * 5 \tag{3.1}$$

For the other criteria, where a smaller value is deemed optimal, we have metrics like 'Duration of the Scan' (representing the Performance criterion) and 'Number of False Positives'. Their score is computed as follow:

$$Score = (1 - x') * 5 \tag{3.2}$$

The 'Maintainability' criterion is derived from two distinct sub-criteria.
The first is the 'Number of Years Since the Last Commit.' For tools that are still receiving updates in 2023, this value is set to 0. This sub-criterion serves to distinguish projects that are no longer receiving updates. A larger value here indicates potential obsolescence, suggesting the tool might not be attuned to current vulnerability detection needs.
The second sub-criterion is the 'Number of Commits in 2023 (up to July).' This metric differentiates projects that are still actively updated. A higher number of commits indicates that contributors are actively refining and updating the tool, which is a positive sign of its relevance and adaptability.
Both these sub-criteria values undergo Min–max normalization. The score for 'Number of Years Since the Last Commit' is determined using formula 3.2, while the score for 'Number of Commits in 2023' is derived from equation 3.1.
In the final step, the 'Maintainability' score of the project is computed as the weighted average of these two sub-criteria scores, with both carrying an equal weight of 1.

Finally, the final scores for each tool concerning the criteria 'Vulnerabilities Found', 'False Positives', 'True Positives', and 'Performance' are determined by taking the average of the scores obtained from evaluations on both the OWASP Juice Shop and Google Gruyere.

3.2 Security champions program guide

The guide titled "How to Have a Successful Security Champions Program" is designed to provide comprehensive advice for those navigating the complexities of a security champions program, regardless of its stage. Whether you're initiating the program or maintaining it, this guide is tailored to assist.

It is structured chronologically and it addresses four following main challenges: securing management support, the recruitment of security champions, their effective training, and ensuring their sustained motivation. These challenges are commonly encountered in the realm of security champion programs. To address them, the guide offers solutions and advice, equipping application security professionals to tackle these issues effectively.

This guide stands apart from typical lengthy documents. It's crafted to be interactive and user-friendly. Whether you're at the program's inception, in the midst of its implementation, or in the maintenance phase, this guide offers valuable insights.
It serves as a reliable resource for those striving to optimize their security champions program.

The details and the content of the guide are presented in the implementation part.

Chapter 4

Implementation

4.1 Open source DAST tools comparative study

In this section, we provide detailed guidelines to enable someone to replicate the study.

The study has been conducted on a kali linux vm installed in Virtual box. Kali linux is the reference distribution for pen testing tools. The details of the VM configuration can be seen below on figure 5.

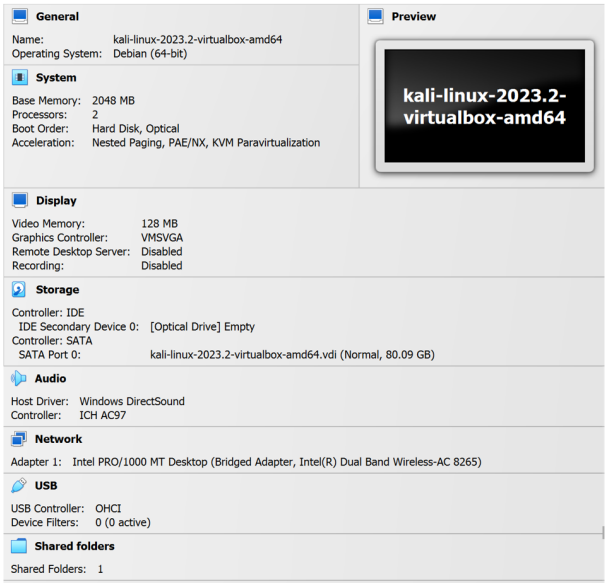

Figure 5: Configuration of the kali linux virtual machine in virtual box

4.1.1 Targets setup

OWASP juice shop

The OWASP Juice Shop project was cloned from its GitHub repository and then executed on localhost using npm. After the server is initiated, the Juice Shop becomes available on localhost at port 3000.

Google gruyere

To set up Google gruyere we first visit the following url https://google-gruyere.appspot.com/start. Then Google Gruyere sets up a new instance for you, generating a unique URL. You can then immediately begin scanning the web application at the provided URL.

4.1.2 Tools setup

The following paragraphs explain with what commands and arguments the tools have been run.

OWASP Zap

OWASP Zap is the only among the tools to have a GUI. In order to run a scan in OWASP Zap we select "Automated scan" and enter the chosen url. We enable the "Traditional spider" but also the "Ajax spider" with firefox as parameter.

The other tools are run from the terminal. Here are the commands that we used to run them:

Wapiti

Here is the command that we use to run Wapiti:

```
$ wapiti -v2 -u <url> --scope domain -f html
```

We modify the default scope which was only the page in order to attack the whole domain. This command creates a new folder .wapiti in the directory it is executed. This folder contains the HTML report of Wapiti.

Nikto

Here is the command that we use to run Nikto:

```
$ nikto -h  <url> -Format htm -outpout report.html
```

This command generates an HTML report with the name given in parameter in the directory it is exectued.

Golismero

Here is the command that we use to run Golismero:

```
$ golismero scan <url> --forbid-subdomains -o report.html
```

For the Golismero scan, we restricted the search to exclude subdomains, as there was only one domain to examine for both targets. Without this limitation, Golismero would have attempted to probe numerous subdomains, potentially taking a lot of time to complete. Finally this command generates an HTML report with the name given in parameter in the directory it is exectued.

Skipfish

Here is the command that we use to run Skipfish:

```
$ skipfish -o <output directory> <url>
```

This command creates a new folder with the name given for the parameter -o. This folder contains a file called index.html which is the scan report.

4.2 Security champions program guide

In order to make the guide interactive, we chose to implement it using the presentation tool prezi [9]. The url of the guide is written in the README of the project [12]. It contains four main parts corresponding to the 4 main challenges given in the design section, this way in function of the maturity of someone on his security champion program, it can simply zoom on the challenge he feels more concerned about. An overview of the guide is shown below on figure 6.

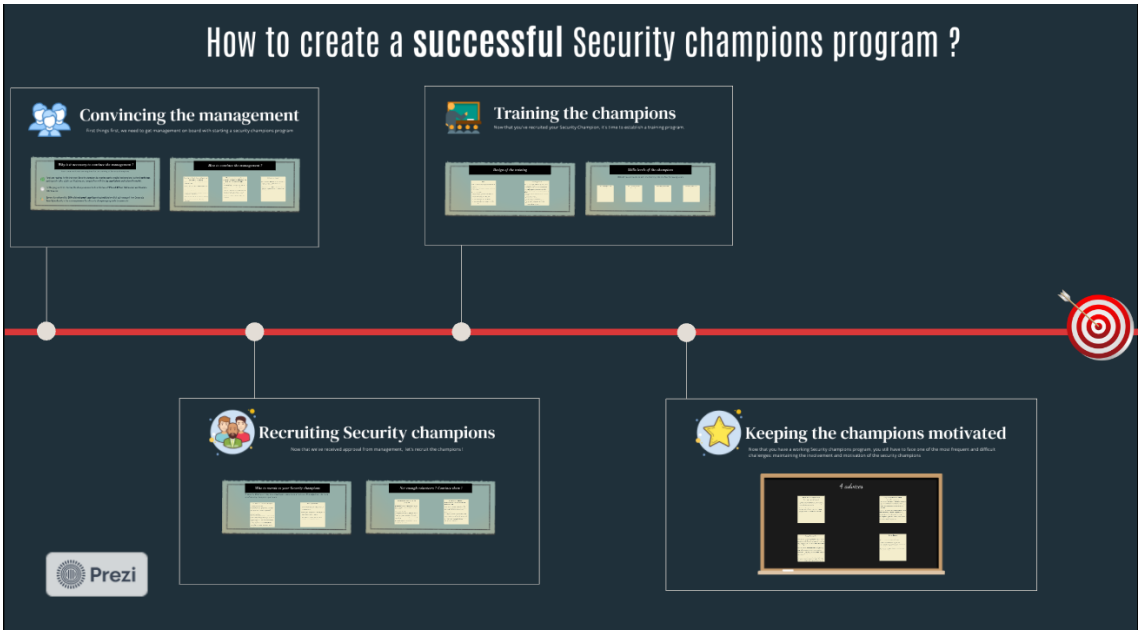

Figure 6: Overview of the Security champions program guide

In the following parts we give a description of the content of the guide, following the same outline as the guide.

4.2.1 Convincing the management

Why is it necessary to convince the management ?

The necessity to gain management's support in the training of Security Champions is paramount [24], as it involves significant financial and time commitments. Funds are essential for organizing events [3], enabling champions to attend workshops, participate in talks, obtain certifications, and equip them with the most advanced security tools available in the market. Moreover, for the program to be effective, the champions must allocate between 20% and 30% [24] [1] of their regular work hours to their new role, a shift that requires managerial approval. The importance of management's active support is further underscored by surveys, where 56% of development security professionals [32] identified it as the primary requirement for the success of a Security Champions program. This alignment with corporate security leadership not only ensures the necessary resources but also fosters an organizational culture that prioritizes and invests in security competence.

How to convince the management ?

Describe in details the contribution of the Security champion

Security Champions make essential contributions within an organization. They serve as a connecting link between the Development team and the Security team, ensuring smooth communication [32]. They're the go-to person for developers with security questions, helping to build a security-aware culture within the development team. They also take on the responsibility of finding and reporting security issues in their projects and products, and they're involved in checking the code to make sure it's secure through activities like code reviews and threat modeling [1]. Their work doesn't stop there; they also actively look for ways to make security standards better. In a nutshell, Security Champions are central to creating a secure development environment, and their role is key to improving security practices in line with modern technological needs.

Explain how it would benefit the company

Investing time in security pays off, especially when you consider the cost of fixing security issues down the line [24]. By finding vulnerabilities early, we can avoid the higher cost of fixing them later and reduce the chance of those problems spreading in the final product. It's a proactive approach that makes financial and practical sense. Security Champions play a big part in this by taking on some tasks that would normally fall to the security team. This frees up the security team to focus on the bigger picture of the overall security strategy . Plus, having solid security assurance helps build trust with customers [3]. They know they can rely on your product, and that trust is invaluable. In the end, the time and effort put into security are more than worth it, benefiting the organization in multiple ways.

If the management is still not convinced after those arguments here are two ideas to explore:
First idea: create security awareness for the management team to ensure that they understand the importance of security. [3]
Second idea : propose to conduct a Proof of Concept (POC) on one team to display the effectiveness of the program before implementing it for every team. [3]

4.2.2 Recruiting Security champions

The skills of a Security champion

A Security Champion must possess a genuine interest in security. For professionals in development security, this enthusiasm is often regarded as the essential quality [32], hence the preference for volunteers who are truly passionate about the subject.
Besides, technical expertise, while valuable, is not the sole criterion. Soft skills, especially communication and leadership, play a pivotal role [1]. A Security Champion should guide developers

toward proper security practices and facilitate communication between the development and security teams [3]. While a comprehensive understanding of security is advantageous, the ability to articulate ideas and lead effectively often distinguishes a successful Security Champion. Their role transcends mere enforcement of rules; they must inspire collaboration and foster a shared understanding of security within the development process.

Clearly communicate about the position

Opening applications for the role of Security Champion must be accompanied by clear and detailed communication that elucidates the responsibilities and expectations of the position. This ensures that volunteers are fully aware of what they are committing to and can make an informed decision.
To further gauge interest and identify potential candidates, organizing security workshops or talks can be an effective strategy [3]. These events provide an opportunity to observe who among the team members are most engaged and demonstrate a genuine interest in security. They serve as a platform for potential Security Champions to showcase their enthusiasm and aptitude.
Additionally, direct discussions with the team can provide valuable insights into who might be interested in taking on the role. Engaging in one-on-one conversations and investigating individual inclinations allows for a more personalized approach. By understanding the unique interests and capabilities of each team member, the selection process can be tailored to identify those who are best suited to become Security Champions.

Convincing the team in case of missing volunteers

One of the primary concerns that may inhibit individuals from volunteering as Security Champions is the apprehension of additional work [24]. It's vital to articulate clearly that becoming a Security Champion does not entail extra responsibilities but rather presents a structured pathway for personal and professional enhancement.
The rewards of participation should be emphasized to attract volunteers. Joining the program equates to allocated time within one's schedule specifically for training. This is a significant incentive, as surveys indicate that 84% [32] of workers are open to changing jobs if learning opportunities are provided by a new employer.
Additionally, the program facilitates the acquisition of certifications, further enriching the participants' professional profiles. But the true allure of becoming a Security Champion lies in the unparalleled opportunity to upskill and advance one's career [1].

If you are still missing volunteers after this here are two ideas that can help :
Propose a Limited Commitment [3]: Offering the option to commit for a defined period, such as one year or six months, can alleviate apprehensions about long-term obligations. This limited

engagement allows potential volunteers to experience the program without feeling locked in. Even if they choose not to continue, the knowledge and skills gained in security will remain valuable assets.
Create a Brand: Inspired by the idea of Christopher Romeo [29], creating a brand for the Security Champions program can instill a sense of identity and pride. Designing a logo, adopting a mascot, or crafting other branding elements can forge a unique identity for the program. This branding not only makes the program more recognizable but also inspires developers to join, enhancing their motivation and fostering a sense of team spirit.

4.2.3 Training the Security champions

Design of the training

KPIs

First and foremost, it's crucial to find a method to measure the performance and growth of the Security Champions. This requires determining several Key Performance Indicators (KPIs) that align with the priorities of the security program. These KPIs can provide a clear picture of how the Security Champions are performing and where improvements can be made. Possible KPIs might include [24] the number of threat models or threat modeling activities conducted, findings from those models, code reviews carried out, findings from code reviews, findings from penetration testing, findings from vulnerability scanning, and the number of participants in security events.
By tracking these specific metrics, an organization can gain valuable insights into the effectiveness of the Security Champions and make informed decisions to enhance the security posture.
It's a systematic approach that ensures that the efforts of the Security Champions are aligned with the organization's goals and are contributing to the continuous improvement of security practices.

Training format

When it comes to learning security, prioritizing participatory teaching methods over passive ones can make a significant difference. Research supports this approach, showing that participatory methods enable much greater information retention. For example, traditional lectures might only result in a 5% retention rate, while hands-on practice can lead to a remarkable 75% retention [23]. This stark contrast highlights the importance of engaging, interactive learning experiences. Examples of relevant activities for learning security include [24] attending security conferences, participating in security interest groups, engaging with security video and audio shows, enrolling in security training courses and workshops, exploring secure coding labs and pen testing labs, working with vulnerable applications, and competing in Capture the Flag (CTF) events.
These activities not only foster a deeper understanding of security concepts but also provide practical experience that can be directly applied in the field.

Skill levels of the champions

OWASP security champions playbook [26] recommends to split the training into the four following levels.

Level 1: Awareness training

The initial step in becoming a champion is achieving awareness. This basic stage includes understanding the threats that applications may face and recognizing the results of software weaknesses. In Level 1, champions are introduced to the main aspects of security, learning the fundamentals like important terms, simple ideas, and common vulnerabilities, without going into the ways of implementation.

Level 2: General topics

Level 2 marks the stage where champions, having finished training in broad topics, begin to engage in more practical actions. They start to explore subjects that can be directly applied to their daily work routines. During this phase, champions will perform practical training on general security areas by performing threat modelling, code reviews, penetration testing and more.

Level 3: Specialized training

Level 3 marks a significant shift in the training process for Security Champions. At this stage, the training becomes more personalized and tailored to the individual needs of the champions. Since different project teams are likely using various platforms, languages, cloud providers, and other technologies, the training for each champion must be customized to fit their specific tech environment. This specialized training delves into the unique characteristics and challenges of the particular platforms and technologies that the champions are working with. By focusing on the specific context in which the champions operate, Level 3 ensures that they develop the skills and knowledge needed to address the security concerns that are most relevant to their projects. It's a targeted approach that enhances the effectiveness of the Security Champions, equipping them with the tools to tackle real-world security issues in their particular technological landscape.

Level 4: 6 months or more of being a Security Champion

This level marks 6 months as a Security Champion, a stage where the champion's strong security skills should be evident. This is when the training's impact on the team becomes clear, and its success can be measured using KPIs. The champion is now ready to work closely with the security team, finding weaknesses in their product and actively looking for ways to fix them. This level shows that the champion is fully prepared and effective in their role, highlighting the success of the training program.

4.2.4 Keeping the Security champions motivated

Show the champions that they have a real impact

Use KPIs to celebrate successes and also point out areas that need work. Before diving into a new security concept or activity, take time to explain why it's relevant and why it matters [1]. This approach helps champions understand their value and connects their efforts to the bigger picture of security within the organization.

Keep them informed about security concerns [3]

For example, keep champions informed and engaged by distributing a newsletter on security events. Share your security activities with them and ask for their thoughts on possible improvements or changes. If you can, let them be the first to know about new security tools or processes in the company. As developers, they might spot something the security team missed, giving them a sense of ownership over the wider security efforts in your organization. This inclusive approach fosters collaboration and empowers champions to actively contribute to the security landscape.

Keep them active

Keep Security Champions active by organizing activities that foster a sense of belonging and identity. Reinforce a shared identity [29] and team spirit through tournaments, contests, workshops, talks, interactive quizzes, and bug bounties. These activities encourage friendly competition, learning, and collaboration, adding an engaging and practical dimension to the experience. Together, they help Security Champions feel more connected to their role and each other, boosting the overall success of the program. By creating a cohesive and motivated group, these activities enhance the effectiveness and impact of the Security Champions within the organization.

Reward them

Reward Security Champions by allowing them to pursue certifications and offering symbolic acknowledgments like certificates [3] to celebrate their progress and achievements. These rewards recognize their hard work and dedication, reinforcing their positive impact. Additionally, keep managers informed about the valuable contributions the champions are making. This not only validates the champions' efforts but also fosters a supportive environment that acknowledges and promotes the importance of security within the organization.

Chapter 5

Evaluation

5.1 Results of the open source DAST tools study

In this section, we showcase the performance of the tools on the two designated targets. The outcomes are delineated in tables and illustrated through figures, at the end of the section.

A few overarching observations regarding the results are noteworthy:
Initially, it's evident that the tools don't directly address most challenges. For instance, within the OWASP Juice Shop, a prompt emerges when a challenge is successfully tackled. Merely two such prompts were activated by the tools, specifically for the 'Error Handling' and 'Confidential Documents' challenges. Nonetheless, the tools do provide insights that hint at potential solutions. These hints might pertain to intriguing URLs, misconfigurations, absent security mechanisms, among others. Thus, a vulnerability is deemed 'identified' by a tool if its report furnishes sufficient information to enable an individual to overcome the challenge.

Another observation is the pronounced disparity in the number of false positives generated by the tools, with Nikto being by far the tool with the greater number of false positives. This discrepancy meant that other tools, due to the Min-max normalization, saw inflated scores in the 'False Positives' category. The difference is so stark that Nikto registered a score of zero in this category, while its counterparts hovered close to the maximum score of 5. For a more nuanced comparison, I also conducted an analysis excluding Nikto, focusing solely on the four tools: ZAP, Wapiti, Golismero, and Skipfish. The outcomes of this secondary analysis are detailed in Table 5.4 and Figure 14. Figures 7, 8, 9, 11, and 12 depict the tool profiles based on their scores across various criteria. For all tools, barring Nikto, these visual representations are derived from the results of the secondary analysis.

We will now analyze the performance tool by tool.

5.1.1 OWASP Zap

Ease of Use score : 5

Zap boasts a user-friendly interface. Users can initiate a scan immediately upon launching the app for the first time, showcasing its intuitive nature. The GUI aids in understanding and managing the tool, making tasks like report generation straightforward with a dedicated button available post-scan. Additionally, Zap provides real-time updates on the scan's progress, indicating the percentage completed.

Remediation Guidance score : 5

Zap offers comprehensive and detailed solutions for each finding.

Clear Reporting of Findings score : 5

The report produced by Zap is clean and well-organized, featuring tables, a summary, and additional statistics. It incorporates hyperlinks and dropdown lists for enhanced user experience. [11] [13]

Maintainability score : 5

ZAP continues to be updated in 2023 and has recorded 127 commits from January to July 2023, the highest among the tools evaluated. As a result, it achieves the maximum maintainability score of 5.

Juice shop

ZAP did well on the Juice Shop. As we can see in table 5.1, it reported 9 alerts, all of which were true positives, and found 16 vulnerabilities. This makes ZAP the best in terms of true positives, false negatives, and vulnerabilities, which are the top three criteria. Its scan time was 6m40s, which is close to the average time of 6m.

Google gruyere

ZAP also did well on Google Gruyere. As indicated in table 5.2 it had the second-best score for true positives with 16, false positives with 2, and found 9 vulnerabilities. Its scan time was 15m41s, which is better than the average of 34m14s.

Conclusion

Overall, ZAP is the best tool in this test. It did better than the others in most areas. The table shows that ZAP is top in almost all criteria, except for performance and false positives where Wapiti did better. ZAP's final score is 4.54 in the first test and 4.61 in the second, which is much higher than the other tools.

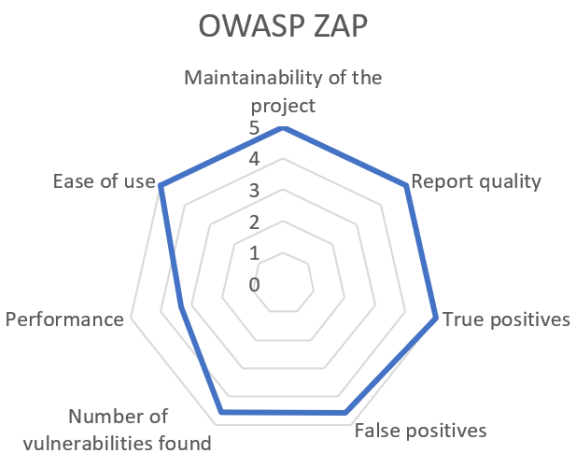

Figure 7: Summary of the performance of OWASP zap based on table "without nikto"

5.1.2 Wapiti

Ease of Use score : 4

Wapiti doesn't have a GUI, but its documentation is adequate. The console's help command was particularly useful in determining the right parameters.

Remediation Guidance score : 3

Wapiti occasionally provides solutions, but these are typically brief and not always available.

Clear Reporting of Findings score : 4

Wapiti's findings are clearly presented, but the overall design of the report is basic. Despite its minimalist appearance, the report is well-structured. [18] [16]

Maintainability score : 4.29

Wapiti is still being updated in 2023. There were 91 commits counted from January to July 2023, earning it a maintainability score of 4.29.

Juice shop

Wapiti was the quickest on the Juice Shop, taking only 5 seconds, compared to 2m16s for the next fastest tool. Another positive is that it tied with OWASP ZAP with 0 false positives, a crucial criterion. On the downside, Wapiti only detected 3 true positives, the lowest among the tools. These led to the

identification of 9 vulnerabilities, slightly below the average of 11.2. The only vulnerabilities Wapiti detected on the Juice Shop were XSS vulnerabilities.

Google gruyere

Wapiti was again very fast, completing the scan in just 8 seconds, while the average was 34m14s and the next best time after Wapiti was 8m1s. It also stood out by reporting no false positives. Wapiti identified 8 true positives, which uncovered 6 vulnerabilities. Both these numbers are a bit below the average. The vulnerabilities Wapiti found were again limited to XSS vulnerabilities.

Conclusion

Wapiti stands out as the fastest tool. Its ability to quickly detect some vulnerabilities makes it valuable. A major plus is its zero false positives, meaning analyzing its report is quick and straightforward. While it tops in performance and false positives, it falls short in true positives and vulnerabilities detected. This places it in 2nd position, right after OWASP ZAP. Its main drawback, leading to its 2nd place ranking, is the low number of true positives and vulnerabilities it identified.

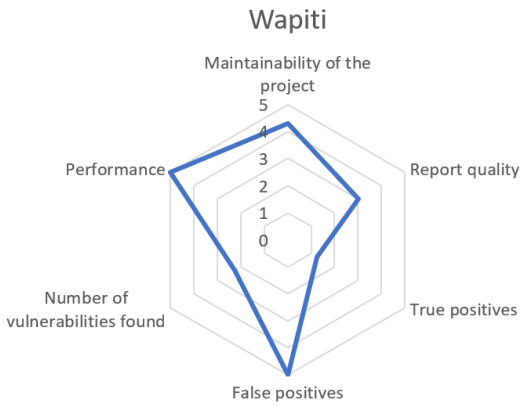

Figure 8: Summary of the performance of Wapiti based on table "without nikto"

5.1.3 Nikto

Ease of Use score : 4

While Nikto lacks a GUI, its operation is straightforward, and there's detailed documentation available online.

Remediation Guidance score : 1

Nikto provides minimal remediation guidance, often just a reference link for the vulnerability.

Clear Reporting of Findings score : 1

The information in Nikto's report is highly condensed, lacking structure and presenting findings as an extensive list. [10] [17]

Maintainability score : 1.59

Nikto hasn't been updated recently, with the last commit dating back to 2019. Using the calculation method described in the design, it receives a maintainability score of 1.59.

Juice shop

Nikto flagged 151 findings, of which 7 were true positives and a whopping 144 were false positives. While its true positives are average, its false positives are a significant concern, especially when the next highest tool has only 6. In terms of vulnerabilities, Nikto missed all XSS vulnerabilities but did detect some that only one other tool identified. Interestingly, it complements Wapiti, as together they detect 19 vulnerabilities, more than any single tool. Its scan duration was 7m6s, slightly above the average.

Google gruyere

Nikto identified a massive 2327 findings, with 23 being true positives and 2304 being false positives. While 23 true positives is the highest among the tools, its false positives are again a major issue, especially when the next highest is only 12 by Skipfish. These true positives led to the discovery of 6 vulnerabilities, which is below average. The vulnerabilities Nikto detected were XSS vulnerabilities. Its scan time was notably long at 56m20s, making it the second slowest.

Conclusion

Nikto ends up in the last position. Its high false positives and poor report quality were major drawbacks, causing it to rank last in both these criteria. A significant problem with Nikto is its tendency to flag URLs ending with filenames like cert.pem as potential findings, even if they don't return an HTTP error. However, in terms of vulnerabilities detected and true positives, Nikto is second only to OWASP ZAP. While it does identify some valuable findings, they are buried in extensive and challenging-to-analyze reports amidst a sea of false positives, making the analysis process tedious and time-consuming.

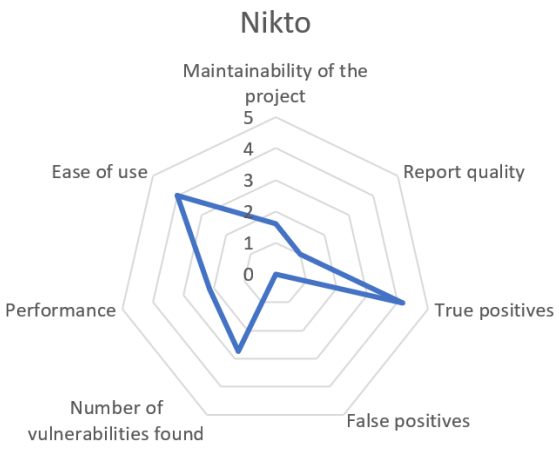

Figure 9: Summary of the performance of Nikto based on table "with nikto"

5.1.4 Golismero

Ease of Use score : 2

Golismero doesn't have a GUI, and its documentation is lacking. Finding the correct parameters, especially for generating an HTML report, was challenging due to conflicting online resources.

Remediation Guidance score : 3

Golismero provides solutions, but they are typically brief, often just a single sentence.

Clear Reporting of Findings score : 5

Golismero's report is excellently structured, featuring charts and tables that offer statistics on the findings, as it is shown below on figure 10. Each finding is distinctly separated and clearly presented. [8] [7]

Maintainability score : 1.82

Golismero hasn't been updated in 2023, with its last update occurring in 2023. This results in a maintainability score of 1.82

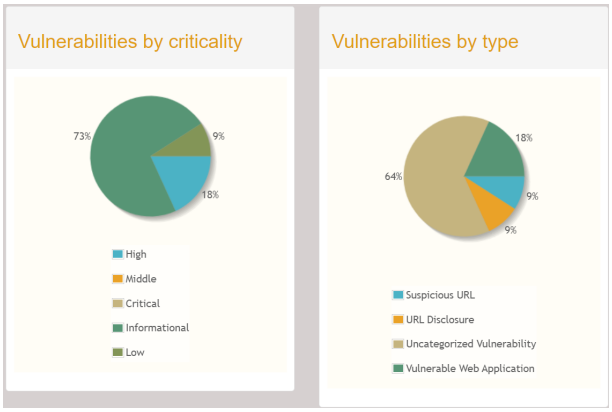

Figure 10: Overview of the vulnerabilities from Golismero's report on juice shop

Juice shop

Golismero delivered average results on the Juice Shop. It flagged 11 findings, with 6 being true positives and 5 being false positives. These findings resulted in the identification of 11 vulnerabilities, which matches the average. In terms of speed, Golismero was the second quickest, completing the scan in 2m16s, compared to the average time of 5m56s.

Google gruyere

Golismero didn't fare well on Google Gruyere. It identified 11 findings, all labeled as "suspicious url", but none were accurate. This resulted in 0 true positives and no vulnerabilities detected, making it the least effective tool for Google Gruyere.

Conclusion

Golismero takes the 3rd spot in the ranking. It has a commendable report quality and speed. However, as shown in table 5.3, its scores in true positives, false positives, and vulnerabilities detected – the key criteria – are lacking. While it seemed promising with its average performance on the OWASP Juice Shop and its decent report quality, its inability to effectively scan Google Gruyere pulled it down.

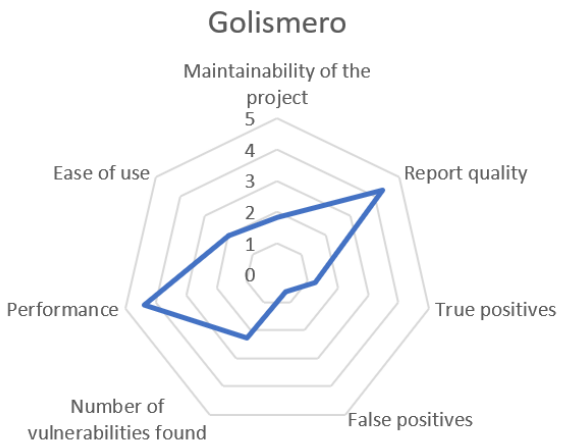

Figure 11: Summary of the performance of Golismero based on table "without nikto"

5.1.5 Skipfish

Ease of Use score : 3

Skipfish lacks a GUI and, being an older project, its documentation isn't so comprehensive.

Remediation Guidance score : 0

Skipfish doesn't offer any solutions for its findings.

Clear Reporting of Findings score : 3

Findings in Skipfish's report are categorized and presented in dropdown menus, ensuring a good structure. However, the details provided for each finding are minimal, often requiring reliance on the title alone to understand the vulnerability. [15] [14]

Maintainability score : 0

Skipfish hasn't seen an update since 2012, making it the longest-unchanged project among those evaluated. Consequently, it receives the lowest maintainability score of 0.

Juice shop

Skipfish's performance on the OWASP Juice Shop was subpar. It flagged 11 findings, of which 5 were true positives, falling below the average. It also reported 6 false positives, the second-highest among the tools. Furthermore, it detected only 5 vulnerabilities, the lowest count among all tools. Its scan duration was the longest at 13m14s. While Skipfish missed the XSS challenges, it was the

sole tool to detect the Outdated Allowlist challenge and Missing encoding. It outperformed others in identifying hidden directories within the web application.

Google gruyere

Skipfish shone brighter on Google Gruyere, detecting the highest number of vulnerabilities at 11, while the combined efforts of the other four tools found only 9. Again, Skipfish excelled in uncovering interesting directories and URLs, aiding in solving some challenges. However, its performance was the poorest, taking a staggering 1h31m1s, nearly three times the average. Additionally, with 11 false positives, it was second only to Nikto in this undesirable metric.

Conclusion

Skipfish lands in the 4th position. Several factors contributed to its lower ranking. Primarily, its scan durations were considerably longer than its competitors. Its lack of recent maintenance also weighed it down. Coupled with a substandard report quality and being second in false positives after Nikto, its overall performance was less than ideal.

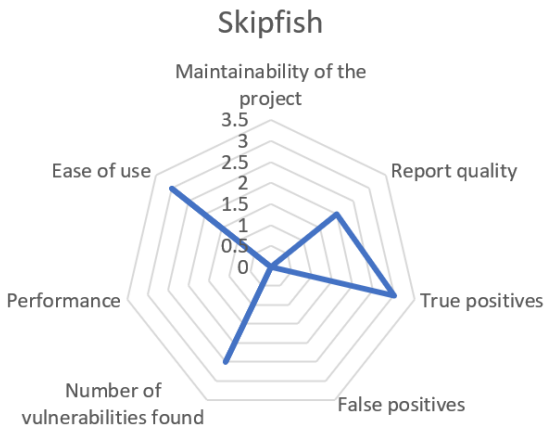

Figure 12: Summary of the performance of Skipfish based on table "without nikto"

	Number of alerts	True Positives	False positives	Duration	Vulnerabilities found
Owasp Zap	9	9	0	6m40s	16
Wapiti	3	3	0	5s	9
Nitko	151	7	144	7m6s	11
Golismero	11	6	5	2m16s	15
Skipfish	11	5	6	13m34s	5
Average	37	6	31	5m56s	11.2

Table 5.1: Results of the scan of the tools on OWASP juice shop

	Number of alerts	True positives	False positives	Duration	Vulnerabilities found
Owasp Zap	18	16	2	15m41s	9
Wapiti	8	8	0	8s	6
Nitko	2327	23	2304	56m20s	6
Golismero	11	0	11	8m1s	0
Skipfish	26	14	12	1h31m1s	11
Average	478	12.2	465.8	34m14s	6.4

Table 5.2: Results of the scan of the tools on Google gruyere

	Owasp Zap	Wapiti	Nikto	Golismero	Skipfish	Average
Maintainability of the project	5.00	4.29	1.59	1.82	0.00	2.54
Report quality	5.00	3.00	1.00	4.33	2.00	3.07
True positives	4.24	0.87	4.17	1.25	2.36	2.58
False positives	5.00	5.00	0.00	4.90	4.88	3.96
Number of vulnerabilities found	4.55	2.27	2.73	2.27	2.50	2.86
Performance	3.35	5.00	2.15	4.38	0.00	2.98
Ease of use	5.00	4.00	4.00	2.00	3.00	3.60

Table 5.3: Detailed scores of the study

	Owasp Zap	Wapiti	Golismero	Skipfish	Average
Maintainability of the project	5.00	4.29	1.82	0.00	2.78
Report quality	5.00	3.00	4.33	2.00	3.58
True positives	5.00	1.25	1.25	3.02	2.63
False positives	4.58	5.00	0.63	0.00	2.55
Number of vulnerabilities found	4.55	2.27	2.27	2.50	2.90
Performance	3.35	5.00	4.38	0.00	3.18
Ease of use	5.00	4.00	2.00	3.00	3.50

Table 5.4: Detailed scores of the second study without Nikto

	Owasp Zap	Wapiti	Nikto	Golismero	Skipfish
API-only XSS	x	x		x	
Bonus Payload	x	x		x	
Client-side XSS Protection	x	x		x	
CSP Bypass	x	x		x	
DOM XSS	x	x		x	
HTTP-Header XSS	x	x		x	
Reflected XSS	x	x		x	
Server-side XSS Protection	x	x		x	
Video XSS	x	x		x	
Confidential document	x		x	x	x
CSRF	x		x		
Access log			x	x	
Easter egg			x	x	
Score board	x				x
Outdated Allowlist challenge					x
Missing encoding					x
Email leak	x		x		
Error handling	x				x
Forgotten Sales Backup			x	x	
Deprecated Interface	x		x		
Forgotten Developer Backup			x	x	
Cross site imaging	x		x		
Misplaced Signature File			x	x	

Table 5.5: Details of the vulnerabilities found by the tools in the OWASP juice shop web application

	Owasp Zap	Wapiti	Nikto	Golismero	Skipfish
File Upload XSS	x	x	x		x
Reflected XSS	x	x	x		x
Stored XSS	x	x	x		x
Stored XSS via HTML Attribute	x	x	x		x
Stored XSS via AJAX	x	x	x		x
Reflected XSS via AJAX	x	x	x		x
XSRF	x				x
Cookie manipulation	x				
Elevation of privilege	x				x
DoS - Quit the Server					x
DoS - Overloading the Server					x
Information disclosure #1					x

Table 5.6: Details of the vulnerabilities found by the tools in the google gruyere web application

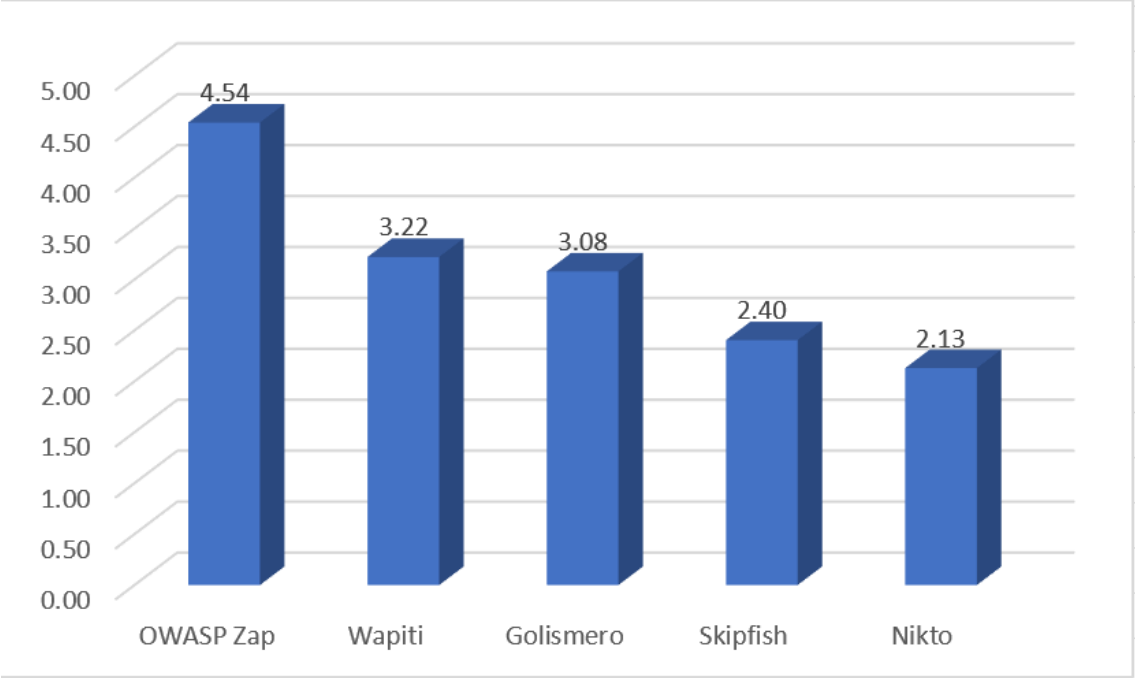

Figure 13: Final scores the study

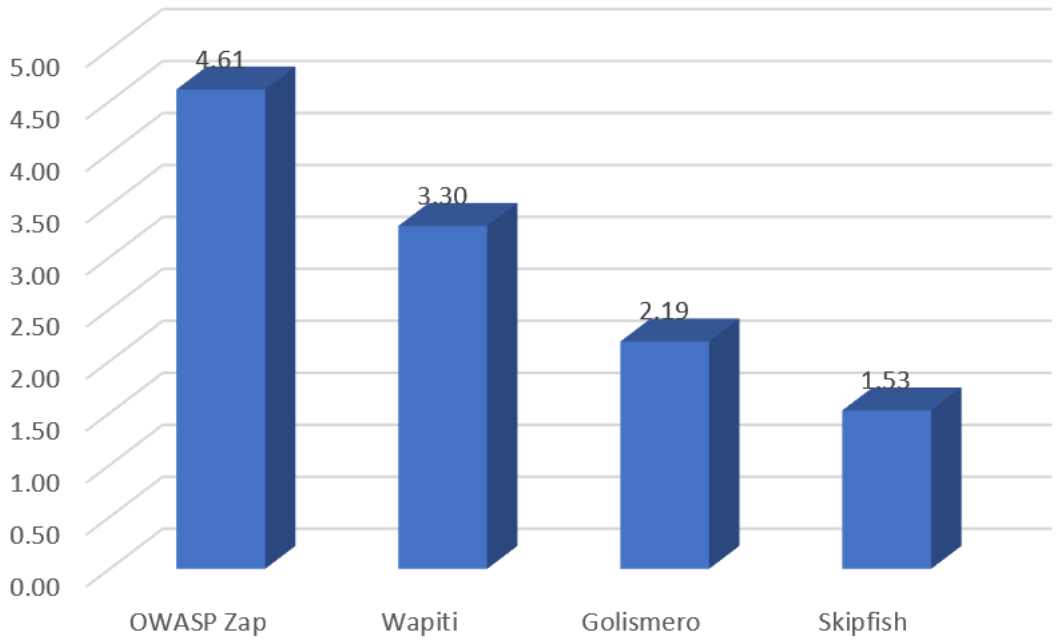

Figure 14: Final scores the second study without Nikto

To summarize the study, it is evident that OWASP Zap stands out as the superior tool among its peers. Each tool, as illustrated in the report's figures, possesses a unique profile. Depending on individual preferences for performance, false positives, and report quality, these profiles can guide users in selecting the most suitable tool for their needs.

Interestingly, none of the tools detected SQL injections, but they exhibited commendable performance in identifying XSS vulnerabilities. No single tool managed to uncover all the vulnerabilities that others did, suggesting that there's potential in using a combination of tools for a more comprehensive assessment. This complementary nature of the tools can be advantageous.

However, it's crucial to note that the tools only identified a fraction of the vulnerabilities present. For instance, on the OWASP juice shop, OWASP ZAP, which detected the highest number of vulnerabilities, only found 16 out of the 102 present, accounting for approximately 15.6%. When all the tools were combined, they identified 23 vulnerabilities, which is less than a quarter of the total.
On Google Gruyere, the combined tools detected 12 out of the 20 vulnerabilities, amounting to 60%. This underscores the fact that while these tools offer valuable insights, they cannot replace the expertise of a pen tester. A web application passing the tools' tests doesn't necessarily deem it safe. As said Dijkstra, "Testing can show the presence of bugs, but never their absence."
The limited detection rate of the tools reinforces the idea that they alone are insufficient for ensuring a web application's security.

5.2 Limitations

5.2.1 Open source DAST tools comparative study

Here are the main limitations of this comparative study on Open source DAST tools

Scope of DAST Tools

The analysis conducted in this study was confined to open-source DAST tools. While this focus provided valuable insights into freely available resources, it inherently limited the comparative scope. An examination that includes paid DAST tools could have offered a more comprehensive perspective, allowing for a broader understanding of the overall landscape of DAST tools.

Nature of Vulnerabilities

The vulnerabilities targeted in the web applications for this study were a mix of real-life scenario vulnerabilities and those more akin to Capture The Flag (CTF) vulnerabilities. While the inclusion of real-life scenarios added authenticity to the study, the presence of CTF-like vulnerabilities may not fully represent the complexities and nuances of vulnerabilities found in real-world applications.

Performance Evaluation Variance

The tools were assessed based on their performance, a metric that can exhibit significant variance. DAST tools operate by sending hundreds of HTTP requests across the network to identify vulnerabilities. Consequently, their performance is highly contingent on network congestion. Although the tools were run on the same Wi-Fi network, they were not executed simultaneously. This discrepancy in timing might have introduced inconsistencies in the evaluation, potentially impacting the comparative performance assessment.

5.2.2 Security training guide

The training guide, has not been subjected to evaluation in a real-world scenario. To truly assess its effectiveness and applicability, it would need to be implemented within a development team, with results observed over an extended period. Such an evaluation would provide insights into how the guide's principles translate into practice and the tangible impact they have on a team's security posture.

Chapter 6

Related Work

6.1 Open source DAST tools comparative study

In my research, I couldn't locate a comparative study conducted in the manner I've approached, which involves defining specific metrics, executing the tools on web applications, and subsequently comparing the tools. Most online resources offer rankings of DAST tools rather than detailed comparisons. A significant concern with these rankings is their opacity. While they provide pros and cons for each tool and assign them ranks, they don't disclose the underlying data that informed these decisions.

In contrast, my study is rooted in objectivity. It presents raw numbers and provides commentary on them, allowing readers to form their own rankings based on their unique criteria, should they differ from mine. This transparency is further emphasized by the detailed breakdown of the study and the profiles of the tools, showcasing their performance across various criteria.

Several rankings I came across include:

Astra [4]: This platform doesn't delve into the specifics of its study and, interestingly, places its own tool at the top of its ranking.

Trust Radius [31]: Their ranking methodology is based on user ratings, which can be subjective.

Comparitech [6]: They evaluate tools based on several criteria, such as:

- Integration into CI/CD pipelines
- Continuous testing

- Black box unit testing
- Integration testing
- Issue tracker integration
- Availability of a free trial or demo package
- Value for money, represented by a comprehensive testing system at a fair price

While these criteria are pertinent, the platform doesn't provide a detailed breakdown of how each tool fares. This raises concerns about the credibility of such sources, especially when they promote free trials of specific tools.

6.2 Security champions program guide

This section iterates on studies related to Security champions and explains why they couldn't replace our guide.

6.2.1 OWASP Security culture [24]

One notable source on the topic of security culture is the "OWASP Security Culture" document. This comprehensive framework is tailored for AppSec professionals aiming to cultivate and enhance a security culture within their organizations. The document delves into the advantages, challenges, and best practices associated with evaluating, strategizing, executing, and gauging a security culture initiative. Additionally, it offers valuable insights and resources for encouraging employees to embrace a security-centric mindset. This guidance is grounded in the research and experiences of security experts and practitioners spanning various sectors, regions, and security maturity stages.

While the OWASP document provides valuable recommendations on fostering a cybersecurity culture, it particularly shines in suggesting metrics and KPIs to gauge the efficacy of security training. Moreover, it offers excellent ideas for participatory activities designed to impart security knowledge. However, this source doesn't focus on security champions. It doesn't delve deeply into the specific role and nuances of security champions within the broader security culture, making it less comprehensive in this particular area.

6.2.2 Security Champions: Empowering Heroes to Unite Security and DevOps [32]

Another significant source in the realm of security culture is the "Empowering Sec Champions" document. This report is essentially a survey that delves into the role and influence of Security

Champions. The report sheds light on the advantages, challenges, and effective practices associated with establishing and managing a Security Champions initiative within an organization. The insights presented in this report are derived from an online survey that garnered responses from 99 security and development professionals. These professionals hail from a diverse range of industries, regions, and security maturity levels.

This study offers intriguing data regarding what application security professionals believe about security champions and the conditions required for a Security Champions program to thrive. One of the primary findings of this research is that the most crucial quality of a security champion is their genuine interest in security. However, the study's scope is somewhat limited, focusing only on certain aspects and conditions that contribute to the success of a Security Champions program.
As such, while it provides valuable insights, it isn't comprehensive enough to serve as a standalone guide for those looking to establish a robust Security Champions initiative.

6.2.3 OWASP security champions play book [26]

Another resource worth mentioning is the "OWASP Security Champions Playbook." This playbook originated as a precursor to the "Security Champions 2.0" discussion held at the 2017 OWASP Bucharest AppSec Conference. The document provides a clear roadmap detailing the essential steps to swiftly establish a Security Champions initiative. It's designed to be adaptable, catering to organizations of all sizes and regardless of where they stand in terms of security process maturity.

The OWASP playbook offers guidance on various phases of the Security Champions program, from nominating champions to other integral processes. However, its scope is somewhat limited, preventing it from being classified as a comprehensive guide. Notably, it overlooks some pivotal aspects of executing a Security Champions program. A glaring omission is the lack of guidance on how to persuade management to endorse such a program. This is a crucial step, as many security professionals concur that management's buy-in is indispensable for the success of a Security Champions initiative [32]. While the playbook does present some interesting concepts, it lacks depth in its explanations and details.

6.2.4 The ultimate guide to building security champions [3]

Another document that delves into the realm of security champions is "The Ultimate Guide to Building Security Champions."
This guide is tailored for developers who possess a distinct interest and expertise in application security. It sheds light on the advantages, obstacles, and effective strategies for instituting and managing a security champions program within an organization. Additionally, it offers valuable insights and resources for training, motivating, and actively involving security champions.

However, this guide has its limitations.
Notably, it doesn't provide any Key Performance Indicators (KPIs) to gauge the effectiveness of the program. Such metrics are crucial for demonstrating the program's success to management, piloting the program before broadening its scope to other teams, and offering feedback to the champions. Feedback is especially vital for maintaining the motivation [30] of security champions, given that a lack of motivation is a significant challenge in such programs.
Another point of contention is the guide's apparent commercial angle, as it seems to promote the solutions offered by AppSec Engineer, thereby raising questions about its impartiality.

Chapter 7

Conclusion

7.1 Security champions program guide

In summary, the creation and success of a Security Champions program depend on a well-rounded approach that includes management support, careful recruitment, structured training, ongoing motivation, and regular evaluation. The journey begins by persuading management of the program's necessity and benefits, thereby securing the needed resources. The selection of Security Champions focuses on both technical expertise and interpersonal skills, with an emphasis on clear communication and a sincere passion for security. The training is methodically designed and tiered to accommodate various skill levels and specific technological needs, with a strong emphasis on hands-on learning. Keeping the champions engaged and motivated involves a combination of feedback, teamwork, acknowledgment, and engaging activities that build a sense of identity and teamwork. The program also integrates mechanisms for tracking performance through KPIs, aligning with the organization's objectives. Collectively, these components form a solid and effective Security Champions program that not only strengthens the organization's security stance but also encourages a culture of ongoing learning, collaboration, and enhancement in security practices. The strategies detailed in this guide offer a thorough roadmap for organizations aiming to develop Security Champions, utilizing their distinct abilities and interests to foster significant improvements in security.

However the training has not been evaluated, which would be necessary to prove is functionality and effectiveness.

7.2 Open source DAST tools study

In our DAST study, OWASP Zap emerged as the standout tool. Each tool showcased distinct characteristics, offering users varied insights based on their specific needs. Notably, none of the tools could detect SQL injections, though they performed well in identifying XSS vulnerabilities. Using a combination of tools can provide a more comprehensive assessment due to their complementary nature. However, despite their utility, these tools identified only a fraction of the total vulnerabilities, emphasizing that they cannot replace the expertise of a pen tester. Thus, passing the tools' tests doesn't guarantee a web application's security.

This study also has limitations of several dimensions.
Firstly, our analysis was restricted to open-source DAST tools, excluding the broader perspective that paid tools might offer.
Secondly, the vulnerabilities we examined combined real-life scenarios with Capture The Flag (CTF) style vulnerabilities. While the former adds real-world relevance, the latter might not capture the intricacies of genuine application vulnerabilities.
Lastly, the performance evaluation of the tools, based on their HTTP requests, could be influenced by network conditions. Even though all tools operated on the same network, they weren't run concurrently, possibly introducing evaluation inconsistencies.

7.2.1 Further work

Here are some suggestions of how this study could be pushed further.

First, it would be interesting to reproduce this study with non-open source tools, examining their capabilities in a manner akin to our current study on open-source counterparts.

Another path that could be explored is the realm of IAST (Interactive Application Security Testing) tools. IAST represents a new approach to application testing, operating from within the application in real-time, thereby offering the advantage of pinpointing vulnerabilities without generating false positives. This unique attribute of IAST could potentially revolutionize vulnerability detection.

Furthermore, to ensure the practicality and effectiveness of our training guide, the next step would be to evaluate it rigorously. This would involve implementing a proof of concept with a dedicated team that adheres strictly to the guide's advice and recommendations.

Bibliography

[1] Hege Aalvik, Anh Nguyen-Duc, Daniela Soares Cruzes, and Monica Iovan. "Establishing a Security Champion in Agile Software Teams: A Systematic Literature Review". In: *Lecture Notes in Networks and Systems*. Springer Nature Switzerland, 2023, pp. 796–810. DOI: 10.1007/978-3-031-28073-3_53. URL: https://doi.org/10.1007/978-3-031-28073-3_53.

[2] Dhaval Anjaria and Mugdha Kulkarni. "Effective DevSecOps Implementation: A Systematic Literature Review". In: *Revista Gestão Inovação e Tecnologias* 11.4 (Aug. 2021), pp. 4931–4945. DOI: 10.47059/revistageintec.v11i4.2514. URL: https://doi.org/10.47059/revistageintec.v11i4.2514.

[3] AppSecEngineer. "The Ultimate Guide to Building Security Champions". In: (). URL: https://www.appsecengineer.com/e-books/the-ultimate-guide-to-building-security-champions.

[4] astra. *14 Best Dynamic Application Security Testing Software [DAST Tools] in 2023*. https://www.getastra.com/blog/security-audit/top-dast-tools/.

[5] Manjunath Bhat. "How to Select DevSecOps Tools for Secure Software Delivery". In: (Feb. 2023).

[6] Stephen Cooper. *The Best DAST Tools for 2023*. https://www.comparitech.com/net-admin/dast-tools/.

[7] Adrien Fermeli-Furic. *Golismero report on google gruyere*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/google-gruyere/GoLismero_Report.pdf.

[8] Adrien Fermeli-Furic. *Golismero report on juice shop*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/juice-shop/GoLismero_Report.pdf.

[9] Adrien Fermeli-Furic. *Guide: How to create a successful Security champions program ?* https://prezi.com/view/dbqhWYKxAM0xv0ZN12KB/. 2023.

[10] Adrien Fermeli-Furic. *Nikto report on juice shop*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/juice-shop/niktoreport.pdf.

[11] Adrien Fermeli-Furic. *Owasp ZAP report on juice shop*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/juice-shop/ZAP_Scanning_Report.pdf.

[12] Adrien Fermeli-Furic. *Project repository*. https://github.com/Fermeli/DevSecOps.

[13] Adrien Fermeli-Furic. *Skipfish report on google gruyere*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/google-gruyere/ZAP_Scanning_Report.pdf.

[14] Adrien Fermeli-Furic. *Skipfish report on google gruyere*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/google-gruyere/Skipfish-scan-results-browser.pdf.

[15] Adrien Fermeli-Furic. *Skipfish report on juice shop*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/juice-shop/Skipfish-scan-results-browser.pdf.

[16] Adrien Fermeli-Furic. *Wapiti report on google gruyere*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/google-gruyere/Wapiti_scan_report.pdf.

[17] Adrien Fermeli-Furic. *Wapiti report on google gruyere*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/google-gruyere/Nikto_Report.pdf.

[18] Adrien Fermeli-Furic. *Wapiti report on juice shop*. https://github.com/Fermeli/DevSecOps/blob/main/Open_source_DAST_tools_study/juice-shop/Wapiti_scan_report.pdf.

[19] Mayank Gokarna and Raju Singh. "DevOps A Historical Review and Future Works". In: (Dec. 2020).

[20] Google. *Google gruyere*. https://google-gruyere.appspot.com/.

[21] Mark Horvath. "Magic Quadrant for Application Security Testing". In: (May 2023).

[22] Will Kelly. *A guide to implementing DevSecOps*. https://opensource.com/sites/default/files/2022-03/osdc_Will-Kelly_Guide-to-implementing-DevSecOps.pdf.

[23] *Learning pyramid*. https://thepeakperformancecenter.com/educational-learning/learning/principles-of-learning/learning-pyramid/.

[24] Nick Miller. "Security Culture 1.0". In: (Apr. 2022). URL: https://owasp.org/www-project-security-culture/v10/.

[25] OWASP. *OWASP juice shop*. https://pwning.owasp-juice.shop/.

[26] OWASP. *OWASP Security Champions Playbook*. https://github.com/c0rdis/security-champions-playbook/blob/master/README.md.

[27] OWASP. *OWASP top 10*. https://owasp.org/www-project-top-ten/.

[28] OWASP. *Vulnerability Scanning Tools*. https://owasp.org/www-community/Vulnerability_Scanning_Tools.

[29] Christopher Romeo. *Elite Security Champions Build Strong Security Culture in a DevSecOps World*. https://www.youtube.com/watch?v=9gVM93a1H1I&t=1280s. Aug. 2022.

[30] Elizabeth Tricomi and Samantha DePasque. "The Role of Feedback in Learning and Motivation". In: *Advances in Motivation and Achievement*. Emerald Group Publishing Limited, Nov. 2016, pp. 175–202. DOI: 10.1108/s0749-742320160000019015. URL: https://doi.org/10.1108/s0749-742320160000019015.

[31] TrustRadius. *Dynamic Application Security Testing (DAST) Tools*. https://www.trustradius.com/dynamic-application-security-testing-dast.

[32] ZeroNorth. "Security Champions: Empowering Heroes to Unite Security and DevOps". In: (Oct. 2020). URL: https://www.zeronorth.io/wp-content/uploads/2020/10/Security-Champions.pdf.

Appendix A

Description of vulnerability types

XSS (Cross-Site Scripting) XSS vulnerabilities allow attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal information, deface websites, or perform actions on behalf of the user without their consent.

Sensitive Data Exposure This vulnerability involves the exposure of sensitive data due to lack of encryption, weak algorithms, or insecure storage. Attackers can exploit this to steal confidential information like passwords, credit card numbers, or personal details.

Improper Input Validation This occurs when an application doesn't validate or improperly validates input before processing it. This can lead to various vulnerabilities, including SQL injection, XSS, and more.

Broken Access Control This vulnerability allows unauthorized users to gain access to certain parts of an application or data they shouldn't be able to access. It can result from misconfigurations or flawed logic in the application.

Vulnerable Components This refers to the use of outdated or vulnerable software components, libraries, or frameworks in an application. Attackers can exploit known vulnerabilities in these components to compromise the application.

Broken Authentication This vulnerability can allow attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume the identity of other users.

Security through Obscurity This is a flawed security principle where secrecy of the design or implementation is relied upon for security. True security should not depend solely on keeping things hidden.

Insecure Deserialization This can lead to remote code execution, replay attacks, injection attacks, and privilege escalation attacks. It occurs when untrusted data is used to reconstruct an object.

Broken Anti Automation This vulnerability refers to the lack of mechanisms to prevent automated attacks like brute force attacks, web scraping, or automated form submissions.

Injection This occurs when untrusted data is sent to an interpreter as part of a command or query. The most common example is SQL injection, where malicious SQL statements are inserted into an entry field for execution.

Security Misconfiguration This can occur when security settings are defined, implemented, and maintained improperly. It can expose sensitive information or grant unauthorized access to attackers.

Cryptographic Issues This category deals with vulnerabilities related to the improper use or implementation of cryptographic functions. This can include using outdated algorithms, improper key management, or not using encryption where it's needed.

Client State Manipulation This refers to the alteration of the state information stored on the client side, often in cookies or hidden fields, to gain unauthorized access or privileges.

XSRF (Cross-Site Request Forgery) This attack tricks the victim into executing unwanted actions on a web application in which they're authenticated, potentially leading to data loss or unauthorized actions.

Cross Site Script Inclusion (XSSI) XSSI vulnerabilities allow attackers to include scripts from one site into another, leading to data theft or other malicious actions.

Path Traversal This vulnerability allows attackers to access files and directories that are stored outside the web root folder by manipulating variables that reference files with ".." sequences and its variations.

Denial of Service This attack aims to make a machine or network resource unavailable, disrupting the services of a host connected to the internet.

Code Execution This allows attackers to run arbitrary code on the server, leading to full system compromise.

Configuration Vulnerabilities These vulnerabilities arise from insecure settings or misconfigurations in the application or its environment.

Ajax Vulnerabilities AJAX (Asynchronous JavaScript and XML) is a technique for creating fast and dynamic web pages. Vulnerabilities in AJAX can lead to a wide range of attacks, including XSS and data theft.