National College of Ireland

Enhancing Web App Security in CI/CD Pipeline: A DevSecOps Framework with Open-Source Tools

MSc Research Project

Cybersecurity

Arjun Variammattu Sasi

Student ID: x22180770

School of Computing

National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: **Arjun Variammattu Sasi**

Student ID: **X22180770**

Programme: **MSc Cybersecurity** **Year:** **2024**

Module: **MSc Research Project**

Lecturer: **Michael Pantridge**

Submission Due Date: 25/04/2024

Project Title: **Enhancing Web App Security in CI/CD Pipeline: A DevSecOps Framework with Open-Source Tools**

Word Count: **8527** **Page Count: 21**

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

Signature: Arjun Variammattu Sasi
Date: 25/04/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

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

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

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

Enhancing Web App Security in CI/CD Pipeline: A DevSecOps Framework with Open-Source Tools

Arjun Variammattu Sasi

22180770

Abstract

The thesis provides a well thought-out DevSecOps framework for web applications that specifically addresses the urgent need to integrate security into CI/CD pipeline in an unobtrusive manner. The framework automates the process of security testing through SAST, DAST while at the same time adopting manual penetration testing approaches. Implemented on AWS platforms through the GitLab CI/CD pipeline, it simplifies security assessment and improves deployment effectiveness. This is facilitated by a major aspect of the framework which is its capability to combine tool outputs thereby enabling unified security audit report generation. Empirical evaluation and case studies demonstrate the practicality and efficacy of the proposed solution in enhancing the security posture of web applications. By developing an exhaustive approach to security testing based on tools, this thesis advances DevSecOps practices thus filling an important void in existing literature for developers and stakeholders with concrete takeaways.

1 Introduction

In today's digital age, web applications are increasingly important to connect users to a wide range of services, providing a seamless experience that is nevertheless shrouded in lurking threats of cyberattack Titled the results of the. This pipeline represents software development maturity, moving beyond normal coding to a more comprehensive life cycle, which crucially includes security from the get-go

The paper aims to fill a significant gap in existing research by exploring how open source tools can be added to the DevSecOps pipeline to not only secure web applications but also enhance their implementation This need stemmed from the neglect of security early in the development cycle, which was mostly left until the end; which resulted in a non-committal approach and reacted to security risks The literature review examines the emergence of DevOps, highlighting the shift towards agility but still failing to secure a will be integrated quickly. The DevSecOps concept that incorporates security at every stage of development represents a paradigm shift in which security is an intrinsic part of the entire software lifecycle.

This study revolves around the important question: How can the addition of open source SAST and DAST tools to the DevSecOps pipeline improve the security efficiency of web applications? The CI/CD pipeline developed mirrors the DevSecOps approach, emphasizing security integration at every step. With automated security tools for dependency scanning and dynamic application testing, the pipeline continues from build to test, to production phase,

using SAST and DAST methodologies for comprehensive security analysis Network the addition of security scanning further supports the dynamics of the pipeline. This design is consistent with the evaluation goal of enabling security in the CI/CD workflow, ensuring that the secure web application is deployed on Amazon Web Services as well, a custom Python script with a pipeline in aggregates security reports from multiple tools into a single vulnerability statistics report as well Automates issue creation in GitLab for identified vulnerabilities, prioritizing them according to how severe This facilitates simple and effective security responses risks, and increase the use of secure web applications.

It is important to acknowledge the limitations and limitations of the study that may affect the results. Despite controlled and random sampling efforts, some variables may influence the results. The learning proceeds through the knowledge of underlying concepts such as the reliable operation of selected open-source tools.

This report is well structured to guide you through the detailed analysis of security integration in the CI/CD pipeline. The article begins with the introduction in the first chapter, setting the initial context and identifying the research question. Chapter 2, Related Work, sets this research into a larger scholarly discourse by providing a critical literature review. Chapter 3 presents the methodology and detailed descriptions of qualitative and quantitative methods that were used. The Design Specification continues in Chapter 4 followed by Implementation in Chapter 5 and finally the results are tested in Chapter 6 Evaluation to measure the effectiveness and efficiency of the system. In section 6.5, Discussion involves further in-depth analysis while focusing on meaning, importance and relevance of findings. Finally, concluding remarks and future work are discussed in Chapter 7 which also points on areas still requiring further research following this report. This method guarantees clarity as well as logical progression thus enabling readers to appreciate what research has contributed to the field of cyber security.

2 Related Work

2.1 DevOps vs DevSecOps

In "DevSecOps: A Boon to the IT Industry,"(Mittal *et al.*, 2021) by Kriti Mitta explores how DevOps can be fortified with security. It suggests transforming CI/CD into CI/CD/Continuous Security (CI/CD/CS) to weave in security checks throughout development. As their study is methodological, they use an AppSec pipeline for continuous security. In the thesis written by Bakary Jammeh called "Enhancing Web App Security in CI/CD Pipeline: A DevSecOps Framework with Open-Source Tools."(Jammeh, no date) the author stresses that we must integrate security into CI/CD but emphasizes the use of open-source tools for more automation and streamlined deployment of secure web apps. The current thesis extends the discussion by detailing how open-source SAST and DAST tools, alongside container security testing with Trivy, can be integrated into the pipeline to automate security testing.

Bakary Jammeh's paper "DevSecOps: Security Expertise a Key to Automated Testing in CI/CD Pipeline"(Ness, Rangaraju and Dharmalingam, 2023) from Bournemouth University delves into automated testing within a DevSecOps framework. He points out that we need security expertise throughout and then outlines the principles of DevSecOps. The author argues that automation is key to matching the speed of DevOps practices when it comes to security testing. Sakthiswaran Rangaraju and his team focus on cloud security in their paper “Incorporating AI-Driven Strategies in DevSecOps for Robust Cloud Security.” (Rajapakse *et al.*, 2022). They

look at how artificial intelligence (AI) can make this branch of technology even stronger. Challenges and considerations are discussed to help embed AI-driven tools within these existing practices so that we can automate as much as possible - all within the workflow defined by CI/CD. “Challenges and solutions when adopting DevSecOps: A systematic review” (Rajapakse *et al.*, 2022) by Roshan N. Rajapakse et.al, looks at combining the strengths of DevOps and security. Their solutions include Interactive Application Security Testing (IAST) tools, orchestrating other tools for continuous vulnerability assessment, adopting Infrastructure as Code (IaC), and building a collaborative culture.

2.2 Automated Security Testing in CI/CD Pipelines

Zachary Wadhams et.al's (Wadhams, Reinhold and Izurieta, no date) study discusses Static Application Security Testing (SAST) tools and their role in identifying and fixing vulnerabilities in software development lifecycles. They propose an automated process that integrates SAST tool outputs into developers' issue-tracking systems so they can manage any vulnerabilities quickly. The thesis tells us that integrating security testing into CI/CD pipelines is super important, but takes it a step further by exploring open-source tools for automation and streamlining of secure web application deployment.

Fiorella Zampetti (Zampetti *et al.*, 2021) et.al's study highlights how CI/CD pipelines have been restructured over time across many open-source projects, going over trends and actions taken to make code more readable, build matrices simpler, and security more robust. It focuses on continuous adaptation and improvement of CI/CD processes, which is directly connected to what this thesis project aims to do. Rammohan Vadavalasa's (Vadavalasa and Vadavalasa, 2020) paper puts machine learning (ML) applications side-by-side with traditional ones as far as developing, deploying, and continuously improving them goes. It calls for a secure pipeline that supports continuous integration so ML applications can be delivered and deployed safely. Rammohan goes through tools and practices for building secure pipelines specifically for ML apps. Open-source tools are explored here as well as methodologies for securing CI/CD pipelines in the current thesis.

In "Ambush From All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines," Ziyue Pan (Pan *et al.*, 2024) et.al examines the security threats that CI/CD pipelines face, especially in open-source software. They also show how vulnerable these pipelines are when hosted on platforms like GitHub. The results of their study make it clear just how important a security-first mindset is for web application development and deployment within a DevSecOps framework, which matches what the current thesis project is all about

2.3 SAST (Snyk) , DAST(ZAP) , Pentesting tools

The paper “To Detect and Mitigate the Risk in Continuous Integration and Continues Deployments (CI/CD) Pipelines in Supply Chain using Snyk tool” (Sushma *et al.*, 2023)by Sushma D et.al, introduces Snyk into the CI / CD pipeline to enhance security against vulnerabilities and attacks. It identifies attack vectors such as malicious code injection and unauthorized access that can compromise software integrity, confidentiality, and availability. The system of this study uses CodeQL, Slaying The Software Supply-Chain Dragon (SLSA), etc., to evaluate vulnerabilities within the pipeline. As a result, it finds unsafe dependencies and outdated libraries. This study presents how detailed reports from Snyk facilitate a swift response when problems occur. This research supports the thesis's aim to automate security within the CI/CD workflow.

“Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results” (Sonmez and Kilic, 2021) by Ferda Özdemir Sönmez et.al introduces innovative visualization of Dynamic Application Security Testing (DAST) results to create a web application security visualization system that has not been implemented before. In this paper, while emphasizing integration with Static Application Security Testing (SAST) tools and DAST tools in DevSecOps framework, we propose practical visualization tools for test results that reduce the operational load of administrative managers through efficiency improvement. “An Analysis System to Test Security of Software on Continuous Integration-Continuous Delivery Pipeline” (Aparo *et al.*, 2023) by Carmelo Aparo et.al is a modular architecture that integrates existing AST tools into a Continuous Integration-Continuous Deployment (CI / CD) pipeline using Docker containerization. In this study, we used the OWASP Benchmark suite to confirm its effectiveness compared to other systems. This is different from the current thesis where open source SAST tool and DAST tool are used for automation of web application security testing.

“Review of the Benefits of DAST (Dynamic Application Security Testing) Versus SAST” (Sharma, 2021) by Manish Sharma is a paper in which the author discusses the differences between DAST and SAST. It compares the detection methods, and effects on web application security based on those detection methods and develops a robust security framework. “An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities” (Albahar, Alansari and Jurcut, 2022) by Marwan Albahar et.al is an empirical study that evaluates commercial and non-commercial Pen-Testing tools for web application vulnerability identification. In this thesis, by focusing on the open-source SAST tool and DAST tool, we created a system that can evaluate vulnerabilities at once to achieve automated efficient security testing.

2.4 Container security

Marwan Albahar’s study titled “An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities” (Bhardwaj, 2023) highlights the world of penetration testing tools, both commercial and non-commercial, that are used to locate weaknesses in web applications. The aim is to ensure that web applications are not exposed to cyber threats. In this survey, Mr. Albahar studied tools such as Burp Suite Professional and OWASP ZAP. Nevertheless, it is worth mentioning that our current thesis is different because it focuses on open-source SAST and DAST tools only. This will include Snyk and OWASP ZAP which are being incorporated into the DevSecOps framework. In addition, employing a more automated and efficient security testing process earlier in development shows some promise.

“Security Analysis of Docker Containers for ARM Architecture” (Haq, Tosun and Korkmaz, 2022) by Md Sadun Haq. It should be noted IoT (Internet of Things) devices as well as edge computing extensively use Docker containers tailored for the ARM architecture specifically. However, there has not been enough attention directed towards their security aspect. By doing this project we help identify these vulnerabilities specific to ARM containers along with other challenges tied to scenarios concerning edge computing. Based on running a variety of security tools on official ARM images from DockerHub researchers found from their analysis that 72% of vulnerabilities across multiple tools have different levels of severity. This led them to conclude that no one tool could detect at least 80% of all vulnerabilities singly — thus multiple resources are required to effectively secure these environments in any case. On its part, however, this work strengthens our discourse by addressing specific issues related to CI/CD

pipelines using SAST and DAST with DevSecOps integration through a case study revolving around ARM-based container testing(Trivy)

2.5 Gitlab CI/CD integration

Jeffery Fairbanks, study “Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab” (Fairbanks, Tharigonda and Eisty, 2023) looks at how continuous integration and continuous delivery (CI/CD) pipelines impact software development practices of open-source repositories in quantitative terms. By looking over more than 12,000 repositories, they discover that CI/CD raises the velocity of commits by an astonishing 141.19%, and increases reported issues by 321.21%, thereby highlighting its dual effect as a catalytic mechanism for faster development cycles and prompting rigorous issue identification. The current thesis project also bears a similar interest to the above but focuses on enhancing web application security through automated security testing. Even though Fairbanks et al.’s research speaks to the wider impacts of CI/CD, it indirectly supports the premise of the thesis that CI/CD can be used to streamline workflows during the development process. Their findings suggest that while increasing counts of issues could indicate a more proactive approach towards issue detection within CI/CD, this finding aligns well with DevSecOps’ objective of early identification mitigation for vulnerabilities.

“Continuous Integration Using Gitlab” (Arefeen *et al.*, 2019) is a piece written by Mohammed Shainsul Arefeeu and Michael Schiller which compels us to understand that GitLab plays a significant role in supporting agile software development without compromising code quality. Therefore, insights into GitLab’s CI/CD tools provide views about pipeline optimization aimed at delivering efficient and secure software as outlined in the thesis. The challenges involved in configuring CD pipelines are discussed under “Configuration Smells in Continuous Delivery Pipelines: A Linter and a Six-Month Study on GitLab” (Vassallo *et al.*, 2020) by Carmine Vassallo et.al where CD-Linter was introduced to detect “CD smells”. This approach to security testing within CI/CD pipelines matches the perspective of the thesis project, which is geared towards optimizing pipelines for web application development.

2.6 Security testing in existing Web Application of Choice: OWASP Juice Shop Over DVWA or DVNA

The aim of the paper "Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities" (Althunayyan *et al.*, 2022) by Muzun Althunayyan et.al is to test out how effective black-box web application vulnerability scanners are against modern and sophisticated web apps. The researchers put a lot of focus on injection vulnerabilities like SQLi, NoSQL, and Server-Side Template Injection (SSTI). They run some tests on the OWASP Juice Shop which is an intentionally insecure web app and try to see how accurately these five popular black-box scanners can detect any prevalent vulnerabilities. The current thesis project tries to improve web app safety by adding SAST and DAST tools to the DevSecOps framework.

In "Evaluation of Static Web Vulnerability Analysis Tools" (Tyagi and Kumar, 2018) taken from the Distributed and Grid Computing (PDGC-2018) conference by Shobha Tyagi et.al explore if static analysis tools can be used effectively in identifying web vulnerabilities. They specifically focus on two open-source tools called OWASP WAP and RIPS using the Damn Vulnerable Web Application (DVWA) and A Buggy Web Application (bWAPP) as their test beds. The current thesis plans on integrating tools just like these into CI/CD pipelines.

2.7 AWS Elastic Beanstalk for Web Application Deployment

The paper "Implementation and Empirical Assessment of a Web Application Cloud Deployment Tool" (Mendonça *et al.*, no date) by Américo Sampaio aims to describe how the authors developed and evaluated a cloud deployment tool referred to as TREXCLOUD, which is aimed at simplifying web application deployments on infrastructure-as-a-service (IaaS) clouds. Consequently, this will reduce the complexity of configuring virtual machine images and application components for cloud deployment. For example, nine participants were involved in an empirical assessment that included their deploying two Java web applications which proved that TREXCLOUD can reduce deployment effort by up to 90% in optimal scenarios. Therefore, one of the major concerns of this study is a reduction in complexities and manual steps involved in setting up virtual machine configurations and other application details during cloud deployments. Consequently, this research supports the thesis project goal which was improvement of website security by showing how cloud deployments can be more efficient.

This study by Michael O. Ogbole (Nazir *et al.*, 2020) et.al gives a detailed overview of cloud computing concerning service model classification; evaluation and comparative analysis between main cloud service providers such as AWS, GCP, and Azure. It also examines various scalability solutions provided by Google's App Engine including its peculiarities compared with those supplied by Microsoft Azure or Amazon Elastic Beanstalk. Furthermore, these insights should be considered while designing an integrated CI/CD pipeline architecture for automated tests run against web applications deployed on different platforms such as WebLogic Server and OpenShift Container Platform connected via Assured SLDC channels primarily intended for Acme Widgets' customers that would end up with a unified platform managed by only one vendor. A comprehensive comparison between CSPs like AWS Oracle Compute Azure GCP IBM Cloud has been given by Anurag Choudhary Pradeep Kumar Verma Piyush Rai under "Comparative Study of Various Cloud Service Providers: A Review" (Choudhary, Verma and Rai, 2022). The importance when choosing a CSP provider depends on its suitability regarding the type of organization it operates in line with its needs at that particular time concerning security postures plus operational objectives. Hence, this research will be helpful in this thesis project for its deployment of web applications using AWS Elastic Beanstalk in a DevSecOps framework.

3 Research Methodology

3.1 Review of Literature and Framework Formulation

The beginning of this project involved an extensive literature review. The purpose was to figure out the dimensions of the current DevSecOps ecosystem. This part is important because it shows how one can build upon existing knowledge and find potential integration of security measures into Continuous Integration/Continuous Deployment (CI/CD) pipeline internally, which is a core principle of DevSecOps. To understand how existing security practices are embedded in software development cycles, multiple sources were used for the literature review. These sources included academic journals, industry reports, and case studies.

The survey with a large focus on gaps and challenges for organizations in implementing open-source tools for security audits and penetration testing in their DevSecOps pipeline revealed a recurring theme: while the intention was clear, practical implementations often weren't. Many

factors led to this gap like difficult configurations and maintenance of security tools, weak integration with already existing development workflows, and lack of skills as well as knowledge needed to effectively use these tools for security purposes.

From insights drawn from the literature review, it is clear that there is a need to develop a DevSecOps framework. This framework was developed to address these challenges but also leverage power from open-source tools so that they can provide efficient and effective scalable solutions where a safe environment is considered an integral part of the development process instead of an afterthought.

3.2 Stages of CI/CD Pipeline

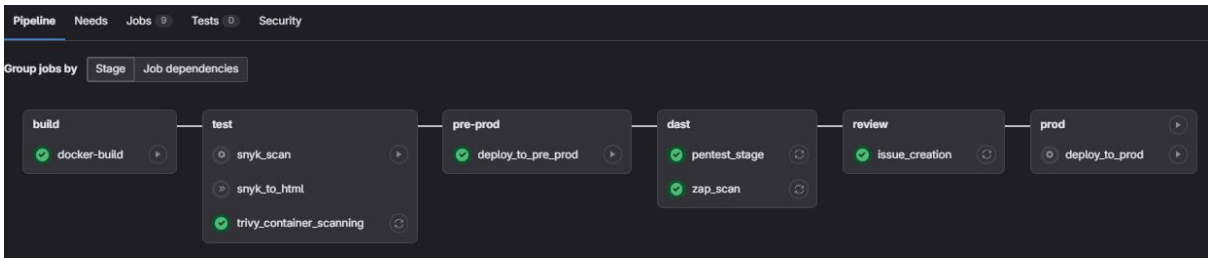

Figure 1: GitLab Pipeline Stages

3.2.1 Pre-Deployment

The pre-deployment CI/CD pipeline lays the foundation for integrating safety and performance requirements early in the development cycle. The ***before_script*** section ensures that the necessary directories for storing artifacts are created, and sets the stage for a smooth transition to the Continuous Integration phase. This preparation step is important for planning subsequent activities and facilitating proper inventory management.

3.2.2 Continuous Integration

The Continuous Integration phase, identified by the build task, involves compiling source code and building a Docker image, which is then pushed to the registry This phase contains the basic principles of CI, with code-commit and auto-built testing each leading to immediate success or failure of the integration process Ensure that responsive Docker architecture processing at this point is necessary to maintain accuracy and speed up the development process.

3.2.3 Continuous Security

Continuous security is woven through the pipeline, from Snyk for SAST scanning, OWASP ZAP for dynamic security testing and custom scripts for additional ***penetration_testing*** It also uses Trivy for container scanning, checking that Docker images have no known vulnerabilities. Integrating these tools directly into the CI/CD pipeline allows for functional security assessments that align with the DevSecOps philosophy of security embedding throughout the lifecycle.

3.2.4 Continuous Delivery

Here, AWS Elastic Beanstalk is used to deploy the application to both pre-production and production environments. This stage ensures that the application, having been built, tested, deployed at pre-production environment and secured, is deployed to the production environment, making it available to end-users. The use of AWS Elastic Beanstalk simplifies

the deployment process, leveraging AWS's capabilities for load balancing, auto-scaling, and application health monitoring, thereby ensuring high availability and reliability.

3.3 Tool Selection and Rationale

The tools for this framework were carefully chosen using input from the literature review. From having a full security inclusion to being easy to integrate, support for automation, and even developer friendly. GitLab was the choice of tool because it is so integrated with DevSecOps features, and has an easy integration with all other security tools. It works well with Docker and can be continuously integrated and deployed which is essential for this study. Through GitLab we can also configure the issue tracking system to automate fixing certain vulnerabilities based on severity levels. Alongside its robust API and documentation making custom pipeline customization a breeze, such as our Python script for report consolidation.

3.3.1 Web Application of Choice: OWASP Juice Shop Over DVWA or DVNA

Instead of DVWA or DVNA, OWASP Juice Shop was chosen due to its modern technology stack in Node.js which is widely used currently. The Juice Shop's architecture provides only one Docker image that needs deployment within cloud environments—making it simpler than other web apps we considered like DVWA or DVNA. Those alternatives may require multiple containers or more complex configurations that would hinder a rapid setup or integration within our GitLab CI/CD pipeline:

- **Modern Technology Stack**: OWASP Juice Shop is built using Node.js, representing a modern and widely used technology stack.

- **Ease of Deployment**: As a result, deployment of the Juice Shop is simplified by its architecture which is contained within a single Docker image in cloud environments. This simplification helps to speed up the process of setting up things quickly and integrating them into the GitLab CI/CD pipeline. While on the other hand, DVWA and DVNA might need more complex configuration steps or even multiple containers before they work properly hence making their implementation in a cloud-based setup quite difficult

- **Comprehensive Vulnerability Coverage**: The OWASP Juice Shop's vulnerabilities include at least all those listed in OWASP TOP 10, thus providing a full security testing platform. Thus, for an exhaustive security audit, this wide-ranging vulnerability coverage is essential since it creates realistic conditions that are difficult to test out how well SAST integrated with DAST tools perform.

3.3.2 Selection of AWS Elastic Beanstalk for Web Application Deployment

AWS Elastic Beanstalk was chosen over other platforms such as Azure due to its unique mix of features that line up perfectly with what we want to achieve here: Modern app support (Node.js), simple deployment process (capacity provisioning, load balancing and auto-scaling handled by AWS), simple maintenance (Elastic Beanstalk manages application health monitoring). Not only does it provide an all-in-one solution, but it's also cost-efficient thanks to the pay-as-you-go pricing model and free educational tier.

- **Modern Application Support**: Elastic Beanstalk support for Node.js apps simplifies the deployment process of modern web applications such as OWASP Juice Shop.

- **Ease of Deployment**: Elastic Beanstalk makes the deployment process easier by taking it upon itself to carry out deployments, starting from capacity provisioning, load balancing, autoscaling and application health monitoring.

- **Cost-Effectiveness and Educational Tier**: A pay-as-you-go pricing plan by AWS Elastic Beanstalk coupled with an educational free tier is very affordable in terms of deploying pre-production and production environments.

3.3.3 SAST Tools of Choice: Snyk Over SonarQube

Even though SonarQube provides many great features itself such as code coverage analysis not found in Snyk, we have decided Snyk will work better for us within this project due to its ability to identify defects without accommodations at an early stage of development. This decision was made after considering various factors about both tools: SonarQube's analysis capability, Snyk's ability to predict new security challenges.

- **Vulnerability Database**: Snyk's proprietary database is constantly updated with new security threats, providing unrivalled vulnerability information It is essential to anticipate the emerging security challenges.

- Integration with the collaboration of developers: Snyk has been designed to integrate seamlessly into the CI/CD pipeline. This way, security scans do not interrupt development performance or slow it down. It fits perfectly with DevSecOps philosophy that includes security as part of the development process without adding much effort and time-consuming work.

- Automatic fix notifications: Unlike SonarQube, which prioritizes code quality and code-based vulnerabilities only, Snyk provides real-time alerts and automated recommendations for fixing detected vulnerabilities. Using this approach will allow you narrow down risks and respond quickly to safety accidents or incidents.

3.3.4 Container Security with Trivy

Trivy, the tool that was chosen to assess container models for vulnerabilities and inconsistencies, has been described as a solid choice. In this day and age of packaging, when it comes to securing those images, you can't be too careful. The many benefits of

Trivy for Container security are listed below:

- It is great at detecting vulnerabilities across OS packages and application dependencies
- Its ability to integrate with CI/CD pipelines like GitLab makes automated checks a breeze
- Scanning is said to be fast as all get out which allows developers to gain information quickly. This quick turnaround time also comes in handy with DevSecOps due to its need for speed.
- It automatically updates its vulnerability database so you don't have to worry about checking for them yourself.

- When it generates reports, they're detailed by including both the vulnerabilities found and their respective fixes. This gives development teams and security teams the feedback they need to address these issues efficiently

3.3.5 DAST Tool of Choice: OWASP ZAP On Burp Suite

When deciding between OWASP ZAP and Burp Suite for DAST (Dynamic Application Security Testing), there were a few things that set them apart from each other.

- **CI/CD pipeline integration**: OWASP ZAP provides easy integration into CI/CD pipelines with minimal complexity compared to Burp Suite, mainly because no additional API is required for integration, making it easier for DevOps practices in addition, ZAP provides pre-built Docker images, with Burp Suite In contrast, it is easier and more efficient to use in environments, where configuration and integration can be resource-intensive

- **Automation-friendly**: This is an automation-friendly tool because it has a good API and can be customized so it can be used to automate security testing in a DevSecOps workflow

- **Developer and QA access**: In addition to security professionals, ZAP has been made accessible to the developer QA team thus encouraging an integrated approach.

3.3.5.1 Nmap:

The penetration testing stage was selected through a methodological analysis of its relevance to our security objectives and capabilities

- Thorough scanning: Nmap's comprehensive scanning capabilities allow comprehensive network vulnerability assessments.
- CI/CD Integration: Enhances automated security functionality, easily integrated into the CI/CD pipeline.
- Attack Simulation: Simulates real-world attacks by finding exploitable vulnerabilities.
- Lua Scripting: Lua supports scripting, provides customizable scans and extended functionality.

3.3.6 Data Analysis and Result Interpretation

Analysing data and interpreting results is key to the DevSecOps framework. By breaking down and understanding the output generated by Snyk, Trivy, and OWASP ZAP tools we are able to identify any vulnerabilities that may exist. Both JSON and HTML formats are used in our analysis of these reports to ensure a smooth process. Our approach is done in two parts- creating a unified format for security assessment purposes and pinpointing automated tracking issues for any detected vulnerabilities.

3.3.6.1 Unified Data Format

- **Consolidation of JSON Reports**: To get an accurate read on where our digital weak points may be, JSON data from Snyk, Trivy, and ZAP goes through a consolidation

process using Python script. This tool picks out info on the detected vulnerability's severity, location, and suggested fixes.

- **Generation of Unified Report**: Extracted data then gets compiled into a unified HTML report format. This file provides an overview of the application's security posture so we can clearly see where our immediate attention should go when it comes to patches.

3.3.6.2 Automated Issue Tracking:

- **Issue Creation in GitLab**: The same Python script runs a check with with GitLab's API for each identified vulnerability issue found in order to automatically create issues in GitLab. This will help us streamline the remediation process so vulnerabilities are addressed ASAP before deploying anything vulnerable or dangerous into production stages.

- **Severity-Based Prioritization**: As part of the sorting process, the script sets priority levels based on severity ratings from tool reports mentioned above. By doing this it helps make sure resources are properly allocated by focusing efforts first on mitigating those most critical vulnerabilities.

3.3.7 Conclusion

The research method chosen for this project was created by using a very detailed literature review. By selecting the tools like GitLab, AWS Elastic Beanstalk, Snyk, Trivy and OWASP ZAP our team aimed to make it possible to integrate security within the CI/CD pipeline that is central to DevSecOps. One of the main focuses was on tool effectiveness and integration ease and automation. The goal of this methodology was to create a rigorous process of enhancing software development security with a foundation laid for future cybersecurity research to improve development pipeline resilience and security.

4 Design Specification & Implementation

4.1 Design Specification

The design specification for the CI/CD pipeline, written in the GitLab CI configuration file, encapsulates a sophisticated architecture crafted to integrate comprehensive security assessments seamlessly into the software development lifecycle. This specification details the underlying techniques, architecture, and framework that drive the implementation, alongside the associated requirements necessary for its execution.

4.1.1 Architecture Overview:

The DevSecOps framework is structured around GitLab's CI/CD pipeline capabilities, leveraging GitLab's integrated environment to implement continuous integration and continuous deployment processes. The pipeline architecture is defined in the ***.gitlab-ci.yml*** file, which is divided into distinct stages ***build***, ***test***, ***pre-prod***, ***dast***, ***review***, and ***prod*** each tailored to fulfil specific roles in the continuous integration, delivery, and deployment

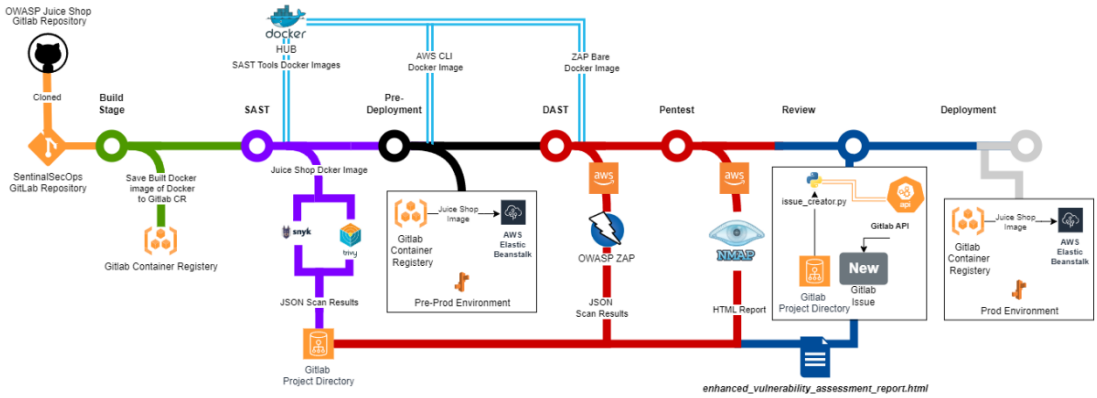

Figure 2: *Architecture Diagram of Pipeline*

processes. This structure facilitates a systematic approach to software development, where code changes are automatically built, tested, and prepared for deployment, ensuring the consistent release of high-quality and secure software.

The architecture diagram depicts an advanced CI/CD pipeline for the OWASP Juice Shop, illustrating a seamless integration of security testing and deployment. Below is a detailed explanation of each component and its role in the workflow:

- **Build Stage**: The first stage in the pipeline is ***built***. OWASP Juice Shop's code is fetched from its GitLab repository with this build. A Docker image that contains the application and its environment is built using this code. This image is then stored in the GitLab Container Registry.

- **Static Application Security Testing (SAST)**: Both Snyk and Trivy scan the Docker image in this stage for known vulnerabilities. With Snyk, dependencies are checked, while Trivy scans the container image for vulnerabilities. The JSON scan output generated by these tools provides a structured record of identified security issues. HTML reports of *snyk* result are created using *snyk-to-html* plugin.

- **Pre-Deployment**: AWS Elastic Beanstalk deploys the Docker image stored in the GitLab Container Registry to a pre-production environment automatically in this stage. Provisioning, load balancing, auto scaling etc processes are managed by this service as well.

- **Dynamic Application Security Testing (DAST)**: In pre-prod environment, ZAP performs dynamic scanning to identify potential security vulnerabilities that can be exploited when app goes live on production environment. It tests the app like an attacker would do to find weaknesses and exploits them accordingly. Results of dynamic scanning are saved as JSON scan results.

- **Penetration Testing**: Network exploration tool Nmap is used during penetration-testing phase to assess network security posture by simulating a malicious attacker

trying to find openings in it. The *nmap-bootstrap.xsl* creates an HTML report that details network's security stance in depth.

- **Review**: Python script ***issue_creator.py*** takes Security reports created from SAST & DAST stages and converts them into actionable GitLab issues, with summarized HTML report generated. ***python-gitlab*** library interacts with Gitlab API securely using its API token and severity of vulnerabilities decides the priority of these issues. This allows the team to work on them on a methodical order.

- **Deployment**: Setup and configuration of pre-prod and prod environments is necessary for AWS deployment phase. Proper IAM permissions are required to manage and automate the deployment processes, which is achieved by creating a dedicated gitlab-ci user in AWS IAM, using aws-cli command integration with GitLab CI/CD service. Once security issues are resolved after review stage, OWASP Juice Shop is deployed to prod environment via AWS Elastic Beanstalk, which makes it live for public access with assurance that all known security issues have been addressed.

This pipeline automates the build, test, and deployment processes. While doing so, it also embeds rigorous security testing at each stage. By adhering to a DevSecOps approach, it ensures continuous security considerations throughout the software creation process.

The end result is the ***enhanced_vulnerability_assessment_report.html*** and ***nmap_scan.html*** files that give transparent explanations of the application's security health for users.

4.1.2 Job Dependencies

The dependency diagram of the CI/CD pipeline illustrates the flow and relationship between various jobs within the GitLab pipeline, showcasing the sequence and conditions under which each job is executed.

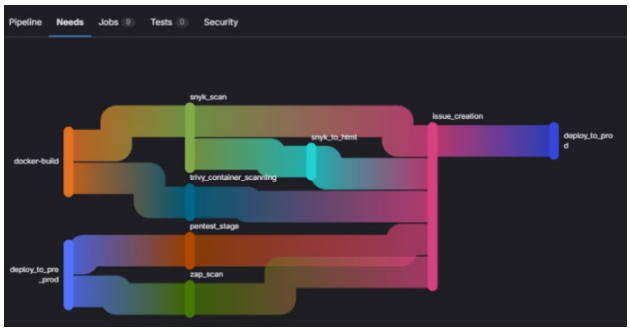

Figure 3: *Pipeline Job Dependencies*

Job Name	Stage	Dependency	Description
`docker-build`	*Build*	None	Initiates the pipeline by building the Docker image for the OWASP Juice Shop application.
`snyk_scan`	*Test*	`docker-build`	Runs a Snyk vulnerability scan on the dependencies, using the Docker image built in the `docker-build` job.
`trivy_container_scanning`	*Test*	`docker-build`	Performs a Trivy scan on the Docker container alongside the Snyk scan, requiring the Docker image from `docker-build`.

`snyk_to_html`	*Test*	`snyk_scan`	Converts the JSON results from the Snyk scan into a readable HTML format, following the **`snyk_scan`** job.
`deploy_to_pre_prod`	*Pre-Prod*	`snyk_to_html`	Deploys the application to a pre-production environment on AWS, contingent on the completion of the **`snyk_to_html`** job.
`pentest_stage`	*DAST*	`deploy_to_pre_prod`	Conducts penetration testing on the pre-production environment set up by the **`deploy_to_pre_prod`** job.
`zap_scan`	*DAST*	`deploy_to_pre_prod`	Executes OWASP ZAP for dynamic security scanning on the pre-production deployment.
`issue_creation`	*Review*	`pentest_stage`, `zap_scan`	Processes the results from security scans to create and prioritize issues in GitLab for vulnerabilities.
`deploy_to_prod`	*Prod*	`issue_creation`	Deploys the application to the production environment after vulnerabilities have been reviewed and addressed.

Table 1: *Pipeline Jobs*

This design specification lays out an extensive plan for integrating security into the CI/CD pipeline. That way, we can ensure it's not just an afterthought, but a normal part of the software development cycle. By using GitLab for CI/CD, Docker to containerize apps, and our own Python script for vulnerability management, we can streamline how we identify, prioritize and fix security vulnerabilities in web applications.

4.2 Implementation

The implementation of the CI/CD pipeline brings speed from start to finish. If you use AWS Elastic Beanstalk for deployment while building things on Docker with Snyk and Trivy running alongside OWASP ZAP, you'll get Docker images and vulnerability reports as output. A custom Python script will also automatically track issues on GitLab based on scan results. With this framework in place we're showing off what DevSecOps principles are all about: delivering a secure product fast

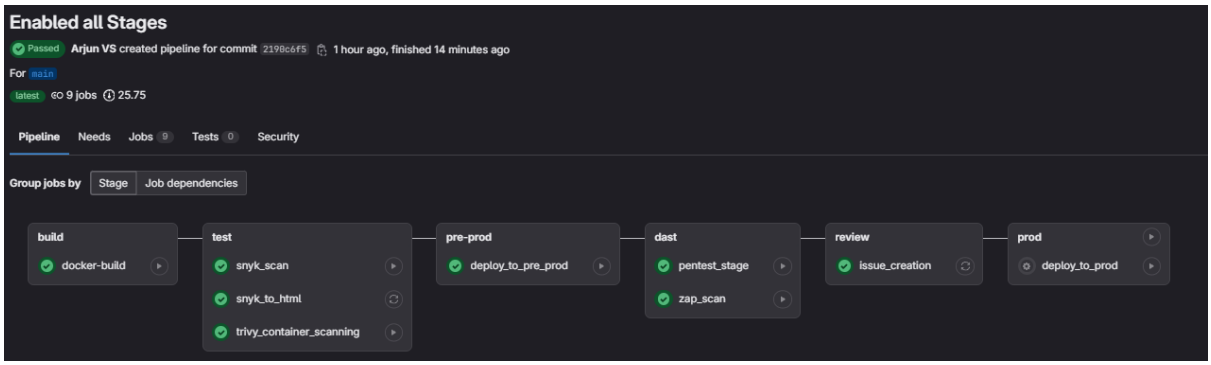

Figure 4 :*Implementation of CI/CD Pipeline*

4.2.1 Building the CI/CD Pipeline:

- **Continuous Integration and Build Process**: The Docker was used to keep application images consistent across different stages of development so they work well when tested or ultimately put into production environments. The ***docker-build*** job takes care of it

all by making new images before pushing them onto your private ***GitLab Container Registry.***

- **Static Application Security Testing (SAST)**: Alongside Snyk and Trivy in this stage you'll find tools that perform Static Application Security Testing (SAST) and container vulnerability scanning. This step helps us keep security front and centre by embedding automated security checks directly into our workflow.

- **Dynamic Application Security Testing (DAST)**: After being deployed on AWS, OWASP ZAP takes over with penetration testing (*nmap scripts*). It all happens in the dast stage, and it's designed to catch runtime vulnerabilities before apps get put into production.

- **Automated Review and Issue Tracking**: The custom Python script run here is able to extract information from security reports and automatically create GitLab issues for each vulnerability. This step helps fix the problem quickly by ensuring everyone knows what's wrong and when it needs to be fixed.

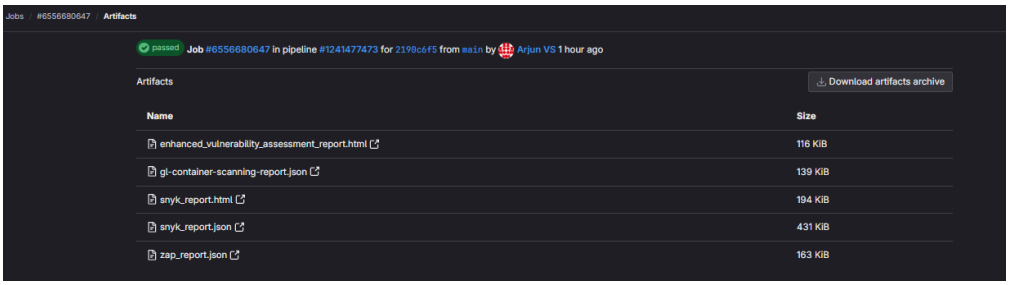

Figure 5: *Generated Artifacts*

- **Custom Python Script (*issue_creator.py*):** Information about vulnerabilities identified by Snyk, Trivy, and OWASP ZAP are extracted using this script. With its access to severity ratings determined by these reports, the script can prioritize remediation efforts so you know which problems need fixing now and which ones can wait. Interacting with the GitLab API is made possible through the ***python-gitlab*** library.

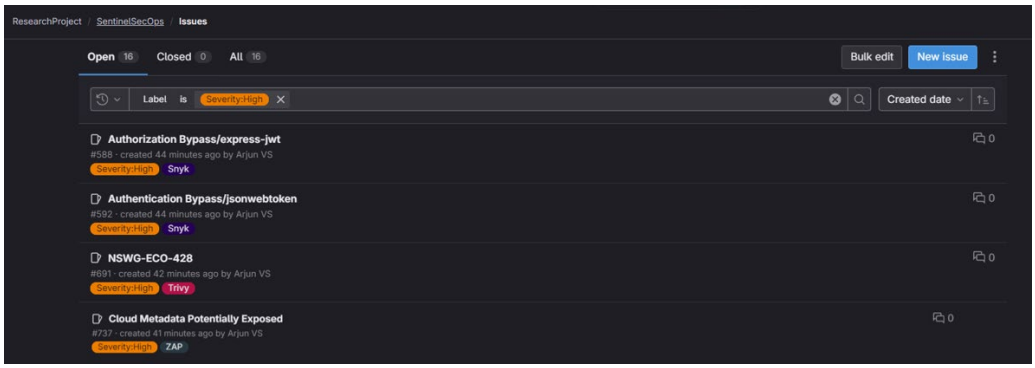

Figure 6: *Vulnerability Issues created in GitLab*

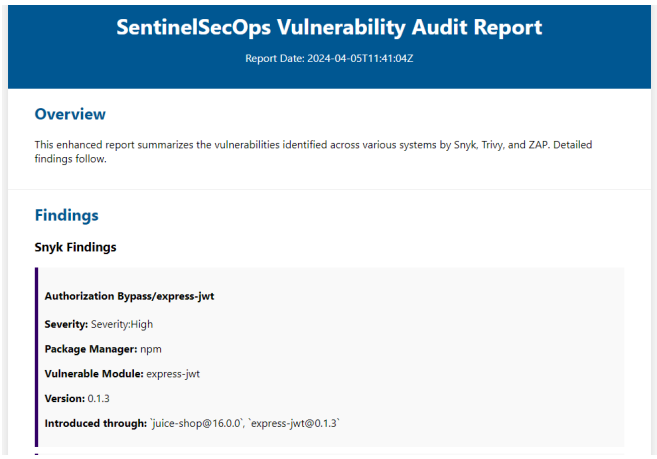

Figure 7 : *Audit report generated by script*

- **Deployment**: Split into *pre-prod* and *prod stages*, we built an *AWS CLI* docker image that should make deployment a breeze regardless. That way you don't have to worry about whether your application is deployable at any time.

- **Penetration Testing:** The ***pentest_stage*** job is put in the DAST stage for pipeline efficiency and parallel execution. Using **Nmap**, it performs vulnerability scanning to simulate attacks against the web application. By running scans that closely mimic potential attacker methodologies, the pipeline can run tests without slowing down. After a scan is completed, **xsltproc** processes the Nmap output into a readable HTML report.

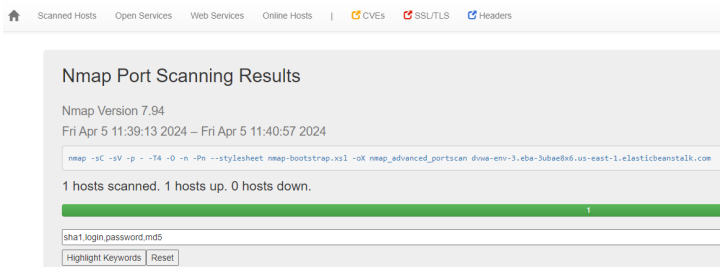

Figure 8: *Nmap Scan Report*

5 Evaluation

This research is taking the first step forward in integrating dynamic application security testing (DAST) and manual penetration testing into the DevSecOps pipeline. As of now, it's mainly focused on container scanning. The evaluation is meant to prove how effective this framework can be for deploying automated security processes, such as SAST tools Snyk and Trivy. While also working alongside DAST tools like OWASP ZAP, for a comprehensive security audit of the OWASP Juice Shop web application.

Through empirical analysis, this research hopes to show what specific tools are better at finding vulnerabilities. Tools like Trivy are great at this and identify a number of vulnerabilities that other tools wouldn't notice. This shows how crucial combining different tools is in order to

approach security from multiple angles. When comparing vulnerabilities discovered by Trivy and Clair, we'll have a more quantitative understanding of how effective this framework really is across different docker images.

The next phase does things manually with penetration testing because there are always issues that automated scans miss. And they give developers advice on how they can further improve the app's security proactively.

These findings also allow researchers to see how adaptable their framework actually is when used across various web applications and CI/CD environments. It highlights the practical implications behind these results so companies know whether or not to invest in it.

This research bridges the gap mentioned earlier in the base project by providing an evaluation that proves just how significant this framework can be within the CI/CD pipeline specifically. By operationalizing dynamic analysis and manual security testing, it presents a huge leap forward for deploying applications securely through DevSecOps practices.

5.1 Case Study 1

5.1.1 SAST Implementation with Snyk

The integration of Snyk into the CI/CD pipeline aimed at the early identification of vulnerabilities within application dependencies during the development stage.

- The deployment of Snyk, a Static Application Security Testing (SAST) tool, was designed to scrutinize and uncover vulnerabilities in the dependencies of the application. The OWASP Juice Shop, purposefully developed with security vulnerabilities, was selected as the subject for this analysis, providing a broad spectrum of security flaws for detection.

- The CI/CD pipeline, facilitated through GitLab, was augmented with Snyk to conduct scans on the OWASP Juice Shop's codebase. Configured to halt the build process upon discovering high-severity vulnerabilities, Snyk served as a critical checkpoint in the development workflow.

Severity	Count
Critical	1
High	2
Medium	10
Low	70
Informational	0

Table 2: *Vulnerabilities found by Snyk*

5.2 Case Study 2

5.2.1 Container Vulnerability Scanning with Trivy

The security of Docker images used in the deployment of the project is evaluated through the integration of Trivy, a comprehensive container scanning solution. Docker containers was employed for deploying the OWASP Juice Shop, necessitate rigorous security assessments to

safeguard against vulnerabilities. Trivy was selected for its adeptness at scanning containers for vulnerabilities, encompassing both operating system packages and application dependencies.

- Within the CI/CD pipeline, Trivy was deployed to scan Docker images of the OWASP Juice Shop for vulnerabilities. The scans targeted both the container's operating system packages and its encapsulated application dependencies.

- Trivy identified a range of vulnerabilities within the Docker images, including both high and medium severity issues. It provided detailed insights into each vulnerability, facilitating targeted remediation efforts aimed at securing the container images prior to their deployment.

Severity	Count
Critical	8
High	13
Medium	10
Low	33
Informational	0

Table 3: *Vulnerabilities found by Trivy*

5.3 Case Study 3

5.3.1 DAST Implementation with OWASP ZAP

The goal was to dynamically assess the deployed web application for exploitable vulnerabilities through the integration of OWASP ZAP for Dynamic Application Security Testing (DAST). DAST complements SAST and container scanning by evaluating the application in its operational state. OWASP ZAP was selected for its proficiency in emulating real world attacks against web applications.

- OWASP ZAP was integrated into the pipeline to conduct automated dynamic scans against OWASP Juice Shop's pre-production deployment. The tool aimed to uncover a variety of vulnerabilities, from injection flaws to authentication issues.
- The tool successfully detected multiple vulnerabilities that were not identified through static analysis or container scanning. OWASP ZAP provided exhaustive reports on each detected vulnerability.

Severity	Count
Critical	0
High	1
Medium	3
Low	3
Informational	2

Table 4: *Vulnerabilities found by ZAP*

5.4 Discussion

Integration of Snyk, Trivy and OWASP ZAP into the DevSecOps pipeline shows that web app security must take various forms. The three tools however represent distinct aspects of security testing reflecting the complexity of protecting applications from constantly changing threats.

Snyk's major advantage lies in early vulnerability detection in dependencies which is consistent with DevSecOps approach to speed and security. This "shift left" method reduces risks earlier in the development cycle thus reinforcing the need for proactive security measures.

The importance of container security in modern application deployment is underlined by Trivy's ability to scan container images. Significantly, it ensures that both OS packages and application dependencies are examined for vulnerabilities acknowledging container security as a foundation for application safety.

OWASP ZAP's DAST function uncovers runtime environment vulnerabilities thus providing insights on how an application can fare against real world cyber threats. This supplements static analysis as well as container scanning and prepares applications to face real attacks.

These integrated tools show that no single methodology can fully address all aspects of software security at once. Hence, a layered approach including SAST, DAST, manual testing supplemented with container scanning provides a comprehensive view on the status of an application's security posture.

6 Conclusion and Future Work

6.1 Conclusion

This project sought to answer how integration of open-source Static and Dynamic Application Security Testing (SAST & DAST) tools along with manual penetration testing can improve the security posture while streamlining deployment enhances web apps within DevSecOps paradigm. Some objectives aimed at deploying a holistic security structure using tools like Snyk, Trivy, OWASP ZAP among others coupled with manual penetration testing strategies to strengthen CI/CD pipelines for web-based applications.

That question has been addressed by this framework development process entirely successfully. By integrating certain chosen tools within GitLab CI/CD pipeline while assessing their efficacy through case studies and empirical analysis; this project has clearly improved the security posture of OWASP Juice Shop deployed web application. The important findings include:

- Using a combination of SAST, DAST and manual penetration testing, this framework detects several vulnerabilities in different areas thereby demonstrating that layered security is paramount.
- The framework is scalable and adaptable as it can be applied to various web applications and CI/CD environments.

- It was confirmed that including security testing in the CI/CD pipeline greatly strengthens an application's security posture by giving developers actionable insights for proactively enhancing security measures.

Apart from being confined within academia boundaries, this research brings tangible contributions to the industry. Informs a practical approach towards assessing and enhancing overall web app security for developers, compliance teams as well as security analysts. Thus, filling gaps in existing literature and practice related to effective and integrated security testing within CI/CD Pipeline is how this project contributes to DevSecOps community.

6.2 Future Work:

In addition to the groundwork set by this project, there are numerous possibilities for further research that might make improvements in the security framework and indicate new dimensions of application security within the DevSecOps philosophy. Here are some important directions to consider for future work:

- Future research may expand the range of tools and applications covered by the framework thereby making it more comprehensive useful.
- The study focused on OWASP Juice Shop, which is mainly a JavaScript-based application. Further studies could extend the framework to support other programming languages and frameworks such as: Python, Java, .NET etc., thus widening its audience among developers and target applications.
- However, identifying vulnerabilities alone is not enough; automated remediation recommendations or patches ought to be provided. Building a part of this framework that does not just identify vulnerabilities but provides solutions will greatly increase how fast applications can be secured.
- To facilitate broad adoption by dev teams and security professionals, there is need for better UI/UX in-built in these tools. For example, among other things an easier interface coupled with integrated educational resources on best practices would democratize security information.
- In future research one might try out comprehensive benchmarking exercises comparing performance impact of integrated security testing across different CI/CD pipelines & environments. It reveals what changes should be made in order not to hamper development workflow hence optimization of the pipeline.

To sum up, this research lays a foundation stone for further development towards secure practices under DevSecOps paradigm providing valuable input into constant debates about smooth integration between production lifecycle and safety measures. The concept of open sourcing it as a tool summarizes collaborative endeavor leading to innovation aimed at supporting web applications' vulnerability control within fast changing digital environment.

In conclusion, this research has laid a solid foundation for advancing security practices within the DevSecOps paradigm, offering a valuable contribution to the ongoing dialogue on integrating security seamlessly into the software development lifecycle. The envisioned release of the framework as an open-source tool encapsulates the spirit of collaboration and innovation, promising to bolster the security of web applications in the rapidly evolving digital landscape.

References

Albahar, M., Alansari, D. and Jurcut, A. (2022) 'An Empirical Comparison of Pen-Testing Tools for Detecting Web App Vulnerabilities', *Electronics 2022, Vol. 11, Page 2991*, 11(19), p. 2991. Available at: https://doi.org/10.3390/ELECTRONICS11192991.

Althunayyan, M. *et al.* (2022) 'Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities', *Electronics 2022, Vol. 11, Page 2049*, 11(13), p. 2049. Available at: https://doi.org/10.3390/ELECTRONICS11132049.

Aparo, C. *et al.* (2023) 'An Analysis System to Test Security of Software on Continuous Integration-Continuous Delivery Pipeline', *Proceedings - 8th IEEE European Symposium on Security and Privacy Workshops, Euro S and PW 2023*, pp. 58–67. Available at: https://doi.org/10.1109/EUROSPW59978.2023.00012.

Arefeen, M.S. *et al.* (2019) 'Continuous Integration Using Gitlab', *Undergraduate Research in Natural and Clinical Science and Technology Journal*, 3(1–11), pp. 1–6. Available at: https://doi.org/10.26685/URNCST.152.

Bhardwaj, P. (2023) 'Detecting Container vulnerabilities leveraging the CICD pipeline'.
Choudhary, A., Verma, P.K. and Rai, P. (2022) 'Comparative Study of Various Cloud Service Providers: A Review', *3rd International Conference on Power, Energy, Control and Transmission Systems, ICPECTS 2022 - Proceedings* [Preprint]. Available at: https://doi.org/10.1109/ICPECTS56089.2022.10047594.

Fairbanks, J., Tharigonda, A. and Eisty, N.U. (2023) 'Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab', *Proceedings - 2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications, SERA 2023*, pp. 176–181. Available at: https://doi.org/10.1109/SERA57763.2023.10197778.

Haq, M.S., Tosun, A.S. and Korkmaz, T. (2022) 'Security Analysis of Docker Containers for ARM Architecture', *Proceedings - 2022 IEEE/ACM 7th Symposium on Edge Computing, SEC 2022*, pp. 224–236. Available at: https://doi.org/10.1109/SEC54971.2022.00025.
Jammeh, B. (no date) 'DevSecOps: Security Expertise a Key to Automated Testing in CI/CD Pipeline'. Available at: https://www.researchgate.net/publication/347441415 (Accessed: 5 April 2024).

Mendonça, N.C. *et al.* (no date) 'Implementation and Empirical Assessment of a Web Application Cloud Deployment Tool', *International Journal of Cloud Computing*, 1(1). Available at: https://doi.org/10.29268/stcc.2013.0004.

Mittal, K. *et al.* (2021) 'DevSecOps: A Boon to the IT Industry', *SSRN Electronic Journal* [Preprint]. Available at: https://doi.org/10.2139/SSRN.3834132.

Nazir, R. *et al.* (2020) 'Cloud Computing Applications: A Review', *EAI Endorsed Transactions on Cloud Systems*, 6(17), pp. e5–e5. Available at: https://doi.org/10.4108/EAI.22-5-2020.164667.

Ness, S., Rangaraju, S. and Dharmalingam, R. (2023) 'Incorporating AI-Driven Strategies in DevSecOps for Robust Cloud Security Incorporating AI-Driven Strategies in DevSecOps for

Robust Cloud Security Product Security Leader at Pure Storage', *Article in International Journal of Innovative Science and Research Technology*, 8(11). Available at: https://doi.org/10.5281/zenodo.10361289.

Pan, Z. *et al.* (2024) 'Ambush from All Sides: Understanding Security Threats in Open-Source Software CI/CD Pipelines', *IEEE Transactions on Dependable and Secure Computing*, 21(1), pp. 403–418. Available at: https://doi.org/10.1109/TDSC.2023.3253572.

Rajapakse, R.N. *et al.* (2022) 'Challenges and solutions when adopting DevSecOps: A systematic review', *Information and Software Technology*, 141, p. 106700. Available at: https://doi.org/10.1016/J.INFSOF.2021.106700.

Sharma, M. (2021) 'Review of the Benefits of DAST (Dynamic Application Security Testing) Versus SAST', *INTERNATIONAL JOURNAL OF MANAGEMENT AND ENGINEERING RESEARCH*, 1(1), pp. 05–08. Available at: http://www.ijmer.org/index.php/journal/article/view/2 (Accessed: 5 April 2024).

Sonmez, F.O. and Kilic, B.G. (2021) 'Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results', *IEEE Access*, 9, pp. 25858–25884. Available at: https://doi.org/10.1109/ACCESS.2021.3057044.

Sushma, D. *et al.* (2023) 'To Detect and Mitigate the Risk in Continuous Integration and Continues Deployments (CI/CD) Pipelines in Supply Chain Using Snyk tool', *7th IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions, CSITSS 2023 - Proceedings* [Preprint]. Available at: https://doi.org/10.1109/CSITSS60515.2023.10334136.

Tyagi, S. and Kumar, K. (2018) 'Evaluation of static web vulnerability analysis tools', *PDGC 2018 - 2018 5th International Conference on Parallel, Distributed and Grid Computing*, pp. 1–6. Available at: https://doi.org/10.1109/PDGC.2018.8745996.

Vadavalasa, R. and Vadavalasa, R.M. (2020) 'End to end CI/CD pipeline for Machine Learning', *International Journal of Advance Research* [Preprint]. Available at: https://www.researchgate.net/publication/351022405 (Accessed: 5 April 2024).

Vassallo, C. *et al.* (2020) 'Configuration smells in continuous delivery pipelines: A linter and a six-month study on GitLab', *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 327–337. Available at: https://doi.org/10.1145/3368089.3409709.

Wadhams, Z., Reinhold, A.M. and Izurieta, C. (no date) 'Automating Static Code Analysis Through CI/CD Pipeline Integration'.

Zampetti, F. *et al.* (2021) 'CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study', *Proceedings - 2021 IEEE International Conference on Software Maintenance and Evolution, ICSME 2021*, pp. 471–482. Available at: https://doi.org/10.1109/ICSME52107.2021.00048.