

DevOps v DevSecOps : A Comparative Analysis

Yash Mestry

M.S in Cybersecurity - Pace University, USA

M.S in Networking & Security - Monash University, AU

BSc Computer Engineering - Pune University, IND

Abstract—Cloud computing has completely transformed the way businesses manage their IT infrastructure. However as cloud services become more widely used, new security issues have emerged, forcing the implementation of a strong cloud security strategy by enterprises. The DevSecOps methodology emphasizes security as a shared organizational responsibility by incorporating security practices into the DevOps process. The advantages and challenges of using DevSecOps for cloud security are highlighted in this paper's summary.

DevSecOps is an effective approach that entails incorporating security measures from the beginning of the software development life cycle(SDLC) all the way through to product delivery. The study uses a comparative analysis and examples to demonstrate the best practices of DevOps and DevSecOps. The main objective is to implement security and related best practices in DevSecOps to better and securely deliver applications or software. This paper presents some of the best practices for leveraging DevSecOps for cloud security, including continuous security testing, automated security controls, and proactive threat monitoring. We additionally describe how adopting security in DevOps can leverage security in the cloud and its software product. The important aspect of incorporating security in the software development life cycle is also covered in the paper. The analysis emphasizes the value of fostering a security culture throughout the business and ensuring that security is prioritized at all levels.

Overall, this paper presents insightful information about using DevSecOps for cloud security, assisting organizations in better understanding the advantages and drawbacks of this strategy and adopting best practices for securing their cloud infrastructure into practice.

.Keywords—DevOps, DevSecOps, SDLC, SAST, DAST, Security, EC2, AWS, CI/CD, SonarQube, OWASP, Dynamic Testing, Static Testing, Bugs, Vulnerabilities, Infrastructure, Cloud, Infrastructure over cloud, SQL, SQLInjection, MySQLi, Php, Instances, Web Servers, Production Deployments, Development, Automation, Pipelines, IAM, Least Privileged access, RDS, Database

I. INTRODUCTION

The process of integrating information security technologies and policies across the DevOps value stream and lifecycle is known as DevOps security. Since DevOps involves every stage of the SDLC, having good security becomes even more important. Businesses are not particularly new to information security and when it comes to information technology, security is a top concern for IT specialists. As opposed to traditional IT frameworks, DevOps infrastructure marks a significant departure from them. Companies that use DevOps practices may provide software upgrades up to 500 times per day. If the security team is not fully involved, rapidly deployed software modifications are more likely to create security issues.[1] This article seeks to assist software professionals in better integrating cybersecurity with DevOps by drawing on lessons learned from applying security concepts in a DevOps environment. In the course of our investigation, we looked at a sample of Internet artifacts and talked with representatives from nine DevOps-using companies about their experiences with security protocols. According to our research, most software developers and operations specialists think that common DevOps activities, such as automated

monitoring, have the potential to improve system security. When a company combines DevOps with its entire security strategy, it frequently uses security settings and security needs analysis.[2] The security team then creates interaction between the development team and the operations team. Most of the time, there is no logical approach to combine security modules and DevOps. Security is a crucial component of every business, but it has been challenging to include into the DevOps process at every stage.[1] Due to a widespread lack of security knowledge, security implementations are often imbalanced, slowing down the environment's speed and agility affecting the product management lifecycle. It is said that optimal product management is based on three major factors; Time, Cost and Scope. Consider a triangle, If there is an overlap or imbalance in any of the sides the triangle will tip to one side. Similarly The triangle here in this case the product stays equilateral or rather successful if all three sides of the project are in-sync and work in harmony. When products leave security as an after-thought or something that can be fixed later and eventually be victims to cyber-attacks the triangle in this case becomes unbalanced and if not corrected immediately the project can lead to catastrophic failure; obviously the solution is simple, integrate security in the projects lifecycle since the beginning. This idea gives us the answer that is forming a partnership with the appropriate group to properly build up security measures. Furthermore, cloud services such as AWS and Azure have great inbuilt security across their platforms, but lack of knowledge and know-how for following correct security standards and documentation to implement these security services offered by the cloud service providers is often not put into practice correctly by developers, security consultants and analysts themselves. To leverage DevSecOps this paper aims to show a comparative analysis between the best practices of DevSecOps and DevOps by leveraging SAST (Static application security testing) and DAST (Dynamic application security testing). SAST and DAST are two of the most popular and successful approaches for locating application security flaws.[2] Furthermore, DevOps and its needs are high given

how organizations have been churning out software and features with a high pace where security is often considered as an after-thought. The need of implementing Security in DevOps is now greater than ever.

II. LITERATURE REVIEW

A. What is the Software Development Life Cycle?

SDLC Software Development Life Cycle guides through all stages of software development and it is a methodology that helps the software development process to be more robust, efficient, and optimized for cost. The primary seven stages of SDLC are Requirement analysis, Planning, Design, Development, Deployment, Testing, and Monitoring. In the first stage of SDLC, inputs are gathered from stakeholders, users, and experts. This phase helps to give the scope of the entire process, detailed requirements, risks associated with it, and also the timeline required to complete the project. In the planning stage, developers mapped out the schedule, system requirements, and plans to mitigate risks and evaluate prototypes. Design phase models on how the application will work and respond to inputs, what programming language will be used, the best-suited prototypes, and security measures to ensure that the software is safe like SSL encryption, password protection, and data storage. It will guide the overall design and the platforms it will host on. Development, in this phase a developer starts to build a consistent code of the software and put action into the plan and work towards the goal that is initialized in prior stages. In the testing phase, in this stage, a test is run several times to check the liability, performance, and capabilities of the code and to check for any bugs, security flaws, or errors in the code. In the Deployment phase integration of the different modules into primary source code and then the code is delivered to the production server and made available to the end-users. Monitoring phase, in this the software is maintained and monitored for any residual bugs or any issues from the users. If some changes are required to be made after deployment developers can implement them in this stage.

B. DevOps

DevOps is a combination of development and operations to build the application or software in a more seamless process. For a long time, development and operations were done in separate silos. However, under a DevOps approach, both teams are combined with engineers working throughout the full development lifecycle [1]. Implementing DevOps increases efficiency, and accelerates and improves the software development phases. DevOps supports CI/CD Continuous Integration and Continuous Delivery is used to automate, integrate, and test code. New code is integrated into the already existing code and then sent for testing and deployment. This CI practice helps the developers to detect bugs or errors in the early development stage which takes fewer resources and helps it time-consuming to fix the bugs. Continuous Delivery is a process where code is built, tested, and deployed to multiple production environments. Automated test process and release process drives frequent testing and deployments this ensures an increase in quality. The operation phase initiates once the application or software is available to end users. This phase monitors the performance of the software and ensures that the software is running without any interruptions. This workflow helps to integrate the latest functionality and helps to improve the service or product continuously. The six stages of the DevOps lifecycle, Planning, Development, Integration, Deployment, Operations, and Continuous Feedback along with these phases the processes of testing and compliance are continuously run throughout the lifecycle

C. DevSecOps

DevSecOps Development, security, and operations is a technique that integrates security throughout the lifecycle of software. Secure design and development are at the core of the design and development phase. In the end, it seems a little paradoxical that the code must be safe yet the process must be accelerated [1]. The synergy between release engineers and security teams may be achieved by creating

a work culture that stresses "security as code" [1]. Effective DevOps ensures rapid and frequent development cycles, but outdated security approaches can undo even the most efficient DevOps initiatives. In DevSecOps, implementing security is as important as continuous deployment and continuous testing. Its emphasis is on the need to build a secure foundation. Integrating security in DevOps requires more new tools, techniques, and some cultural changes in DevOps for effective continuous integration of security measures.

D. DevOps vs DevSecOps

Integration of security in DevOps is DevSecOps. DevOps focuses on efficient and quick delivery of the software whereas DevSecOps focuses on security from the very start of the SDLC. In DevOps security is an issue that is addressed after development, whereas in DevSecOps throughout the stages of the process security is implemented. Traditional security has always been about barring people from disclosing crucial information because of the security policies that were in place [3]. However, DevSecOps deals with implementing security from the initial stages of the application development lifecycle to avoid costly mistakes that could be encountered by implementing security afterthought [3]. Cost is reduced in DevSecOps by fixing any security issues or flaws before the product or application reaches its mature phase. The transformation from DevOps to DevSecOps requires a shift-left approach that will help to reduce errors, vulnerabilities and increase the system's overall efficiency.

SAST and DAST are both crucial elements of a thorough security testing methodology. SAST, which is a component of the CI/CD pipeline, can be integrated into the development process in a DevOps strategy, whereas DAST can be integrated into the testing and deployment process. The entire software development lifecycle, from development to testing to deployment and monitoring, can be integrated into a DevSecOps strategy. Here's a comparative analysis of DevOps vs. DevSecOps:

Definition: In order to increase the efficiency and dependability of software delivery, the DevOps methodology stresses collaboration, communication, and automation between development and operations teams. On the other hand, DevSecOps expands on the DevOps methodology by integrating security principles throughout the entire software development lifecycle, from design to production.

Focus: The main goal of DevOps is to streamline and automate procedures between the development and operations teams in order to deliver software more quickly. On the other side, DevSecOps concentrates on incorporating safeguards into the software development lifecycle to recognize and reduce security risks early in the development process.

Goal: The goal of DevOps is to streamline software development and deployment processes and improve software delivery speed and quality. The goal of DevSecOps is to integrate security into the software development lifecycle to reduce security risks and improve overall software security.

Key Practices: Continuous integration (CI), continuous delivery (CD), automated testing, and infrastructure automation are all examples of DevOps principles. Threat modeling, code analysis, vulnerability scanning, penetration testing, and security automation are examples of DevSecOps practices.

Key Benefits: Improved communication between the development and operations teams, quicker and more dependable software delivery, and shorter time-to-market are all made possible by DevOps. These advantages are made safer by DevSecOps, which also increases software security, lowers the possibility of security breaches, and enhances regulatory compliance.

Challenges: The automation and integration of procedures between the development and operations teams provide difficulties for DevOps. In order to incorporate security practices into the software development lifecycle, DevSecOps must overcome additional obstacles, such as cultural opposition to change, a lack of security expertise, and difficulties upholding security throughout the development process.

In conclusion, DevSecOps expands on the goals of DevOps by integrating security principles into the development process. DevOps focuses on enhancing software delivery speed and quality. The decision between DevOps and DevSecOps is based on the particular demands and objectives of the organization, as each strategy has its own advantages and disadvantages.

E. Cloud Security

Cloud security is a set of practices and tools created to address both internal and external security threats to businesses. As they implement their digital transformation strategy and integrate cloud-based tools and services into their infrastructure, organizations need cloud security.

In recent years, the phrases "digital transformation" and "cloud migration" have become commonplace in business contexts. Both expressions are motivated by the need for change, even if their meanings can vary depending on the organization. As businesses adopt these ideas and work to improve their operational strategy, new difficulties in balancing security and productivity levels occur. While moving largely to cloud-based settings can have various ramifications if done insecurely, more contemporary technologies do enable firms to develop capabilities outside the boundaries of on-premise infrastructure. Understanding how contemporary businesses can profit from the use of linked cloud technology while implementing the finest cloud security policies is necessary for striking the proper balance.

F. SAST (Static Application security testing)

SAST-Static application security testing is used to secure applications by examining the source code to find potential flaws. Syntactic and semantic tests are the two primary methods which are used by

SAST programs to find potential vulnerabilities. Syntactic checks look at the source code to look for calls to unsafe functions or codes that go against security guidelines. Whereas, to find potential security flaws, semantic checks rely on an understanding of the semantics of the program, such as data flow or control flow. By giving quick feedback to developers on problems introduced into code during development, SAST lowers security risks in programs. With real-time access to suggestions and line-of-code navigation, it assists developers in learning about security as they work, facilitating quicker vulnerability detection and collaborative auditing. This makes it possible for developers to write more secure code, which results in a more secure program and less of a need for frequent upgrades and software modernization.

G. DAST (Dynamic Application security testing)

DAST- Dynamic Application security testing performs security checks on a program that is already in use instead of scanning the source code. DAST tools proactively submit created input to the application and examine the results to find any potential weaknesses. Dynamic taint analysis is a crucial method that is frequently used in DAST. The DAST tool records the output from the sink after sending a list of malicious payloads to a data source. The recorded response will match the defined response pattern in the security scanning rules of the DAST tool if the application is vulnerable. Fuzzing is another well-liked DAST testing method that involves feeding an application with randomly generated data until one of the inputs causes the application to crash. The fuzzing testing method is based on the idea that a function's failure signals incorrect input handling, which could result in a security flaw. Fuzzing tools produce testing data using a variety of fuzzing strategies, including random fuzzing, mutation-based fuzzing, and generation-based fuzzing. DAST technologies gives us an insight into how

web apps operate when they are in use, allowing us to resolve possible vulnerabilities before a hacker exploits them to launch an attack.

H. OWASP ZAP (Open Web Application Security Project Zed Attack Proxy)

OWASP ZAP (Zed Attack Proxy) is a popular open-source web application security scanner that helps developers identify security vulnerabilities in web applications during the development and testing phases. It is designed to be used by both security professionals and developers, and it provides a user-friendly graphical interface as well as an API for automation and integration with other tools. When used as a proxy server it allows the user to manipulate all of the traffic that passes through it, including traffic using HTTPS. Zap provides cross-platform i.e. it works across all OS (Linux, Mac, Windows). OWASP ZAP can be used to scan for a wide range of vulnerabilities, including injection flaws, cross-site scripting (XSS), broken authentication and session management, insecure communication, and more. OWASP ZAP provides a number of features to help users improve the security of their web applications, including a reporting system for generating detailed vulnerability reports, a passive scanning mode for monitoring applications in production, and support for a wide range of authentication mechanisms and web application frameworks.

I. SonarQube

SonarQube is an open-source platform, developed by SonarSource, it is a code quality assurance tool that inspects and/or analyzes the source code by performing static and dynamic analysis continuously with the help of tools in order to detect bugs or code smells and gives automotive code reviews and reports. It supports various languages including C, C++, C#, Java, JavaScript, Python, Kotlin, Ruby, Php, HTML

etc. It consists of four components: analyzers, plugins, server and database. It analyzes the source code from every aspect and through each layer of code and distinguishes whether the code has defects, error in styling, code coverage, code complexity or code duplication. SonarQube provides metric values and statistics on which part of the code the problem resides and which part requires improvement. It ensures code reliability, security and maintains the code clean. It also provides guidance on the issue and on how to fix the issues. It evaluates the source code based on a set of rules called quality profiles which can be configured for a specific project.

J. SonarScanner

Software developers can evaluate the quality of source code with SonarScanner. It is a component of the open-source SonarQube platform, which allows for continuous code quality examination. SonarScanner assists programmers in identifying and correcting bugs, vulnerabilities, and code smells (bad code that may result in future problems).

SonarScanner scans source code using techniques for static code analysis. Java, C#JavaScript, Python, and many other programming languages are among others it can examine. In addition, it may be incorporated with a number of build tools, notably Maven, Gradle, and MSBuild.

SonarScanner can assist developers in detecting issues early on in their development process prior to escalating into significant concerns. It is also useful in preserving a consistent level of quality of code throughout a project and facilitates developer collaboration and long-term code maintenance. You need to set up SonarScanner with your project and the code analysis criteria you want to employ before you are able to utilize it. When you run the scanner, a report will be generated that indicates any problems with the code and provides suggestions for how to fix them.

Overall, a sonar scanner is an effective tool for enhancing code quality and making sure that software applications are accurate and secure.

K. AWS CodePipeline

AWS Amazon Web Services (AWS) provides a fully managed continuous delivery service called CodePipeline. It enables software developers to accelerate and efficiently deliver updates and new features to their clients by automating their software release process. You may establish and perform the software release process with CodePipeline, which includes developing the source code, testing the applications, and delivering code updates.

In addition to an API for automated access, CodePipeline provides a graphical user interface for setting up, managing, and visualizing your release pipelines. It incorporates your existing tools and workflows into your release process easily because it connects with a wide range of AWS services and third-party solutions. To get started we have defined the pipeline using the AWS Command Line Interface (CLI). After that, we have chosen the source code repository to utilize, for instance, GitHub, and set the pipeline up to be activated when the repository is changed.

We have set up automatic testing and approval procedures as code passes through the pipeline to ensure that only high-quality code is released into production. Furthermore, CodePipeline offers thorough pipeline visibility, enabling users to maintain track of the status of their deployments and rapidly spot any potential problems.

AWS CodePipeline is a powerful tool for fast and secure software update delivery to users and for automating the software release process in general.

L. AWS code build

AWS CodeBuild is a completely managed continuous integration and delivery service that assembles source code, conducts tests, and generates software packages that are prepared for deployment. It is no longer required to provision, manage, and scale your own build servers.

CodeBuild can rapidly create and test the code using a number of different programming languages and technologies, including Java, Python, Ruby, Node.js, and Docker. To support end-to-end automated software delivery workflows, it also connects with other AWS services including Amazon S3, AWS CodeDeploy, and AWS CodePipeline. Small projects and huge corporate applications of any size can be managed by the flexible and scalable AWS CodeBuild solution. For organizations looking to streamline their release cycles and automate their software development process, it's an ideal choice.

Software applications can be built and tested at a low cost with CodeBuild. That can simply scale up or down according to requirements while only purchasing the computational resources that you actually utilize during the building process.

M. S3 bucket

Amazon Web Services offers Amazon S3 - Simple Storage Solution, a cloud-based object storage solution. An S3 bucket is a container used to store files and objects in the cloud. These objects could be any number of things, including papers, photos, movies, logs, backups, and more.

A unique key and URL are used to store and access each object in an S3 bucket. High object availability and durability are provided by S3, together with security features that restrict access to objects. S3 buckets can also be used as a data lake, to host static websites, and to store data for big data analytics. S3 is a popular option for many businesses since it is a highly scalable and economical way to store and retrieve data in the cloud.

N. EC2 instance

An EC2 - Elastic Compute Cloud instance is a virtual server that is offered by Amazon Web Services. Users can rent virtual machines from EC2, which offers scalable computing capacity in the cloud, and use those machines to run their applications. EC2 instances can be deployed in several AWS regions and availability zones and come in a variety of sizes with varied CPU, memory, storage, and network resource allocations. Users can choose an EC2 instance that best suits their computing needs and only pay for the resources they actually utilize.

Web applications, mobile backends, batch processing, and big data analytics are just a few of the workloads that frequently use EC2 instances. Using the AWS Management Console, AWS CLI, or other AWS tools and services, EC2 instances can be managed. In addition, customers can alter the images of their EC2 instances so that they can launch new instances with the same setup and options.

P. RDS database

A fully managed database service - Amazon RDS - Relational Database Service offered by Amazon Web Services is an easy way to set up, operate, and scale a relational database in the cloud. Users may quickly start a database instance in the cloud with RDS, set up automatic backups and software patches, and compute storage resources as necessary. It also provides built-in security features to protect data.

Many applications that need a relational database, like online applications, mobile apps, gaming apps, and enterprise applications, choose RDS as their relational database provider. With RDS, users can concentrate on their data and applications while leaving the maintenance of the databases to AWS.

O. Github

GitHub is a web-based platform for version control and collaboration that enables software developers to host, review, and manage their code. It gives developers a centralized platform to work together on code, keep track of changes, and control project workflows. It provides integrations with other tools and services, such as continuous integration and deployment (CI/CD) pipelines and cloud platforms, to streamline the software development process.

III. METHODOLOGY

DevOps and DevSecOps are two approaches that have seen enormous developments in the world of software development in recent years. With the aim of accelerating the pace and efficiency of software delivery, DevOps focuses a significant value on collaboration and integration between IT operations teams and software development teams. In order to establish more resilient and secure systems, DevSecOps goes one step further and incorporates safety measures into the DevOps process from the very beginning.

We set up two web servers with static PHP code for PMY Cosmetics, a website for cosmetics, to conduct an evaluation between DevOps and DevSecOps. A DevOps approach will be used to set up the first web server, and a DevSecOps approach will be used to set up the second web server. The objective of this demonstration is to illustrate how these techniques can affect the software development process and the final product. We demonstrated the fundamental distinctions between DevOps and DevSecOps by establishing these two web servers and evaluating the techniques used in their development and deployment cycle. This comparison analysis will offer insightful information on which approach is best suited for software development projects and how businesses can improve their software development procedures for increased productivity, security, and teamwork.

In the context of the current project, we deliberately designed the PMY Cosmetics online login page to be susceptible to a vulnerability that allows SQL injection. This is being carried out in order to evaluate and verify how effectively the DevSecOps pipeline works to find and prevent such vulnerabilities. We monitor how the DevSecOps pipeline responds and what steps can be taken to mitigate a vulnerability by intentionally introducing a known vulnerability. In addition, we are introducing an intentionally susceptible hardcoded password connection to the RDS database. As a result, the DevSecOps pipeline detects the vulnerability and notifies the development and operations teams, enabling them to take necessary measures and stop any potential exploitation of this vulnerability.

Overall, by deliberately injecting vulnerabilities, we are able to comprehend how DevOps and DevSecOps vary in their approaches to security and how each strategy can assist minimize potential vulnerabilities or bugs and defend against cyber attacks.

The listed tasks in the table below are required for the following analysis.

TABLE 1 .TASKS

Sr. No	Task	Sub-Task	Description	Team Member	Hardware/Software	Completion Date	Progress
1	Documentation		Documenting the entire project	Yash Mestry Mitwa Mistry	N/A	04/26/2023	Done

				Purvika Gaikar			
1.2		Detailed Architecture and step-by-step setup	Detailed architecture of infra over cloud and having every step listed for deploying initial infra and both DevOps and DevSecOps Pipelines	Yash Mestry Purvika Gaika	N/A	04/25/2023	Done
1.3		Literature review	A comprehe	Purvika Gaikar		04/25/2023	Done

			nsive overview on the topic	Mitwa Mistry			
1.4		Methodology	Detailed approach, set of methods and procedure conducted for this project	Purvika Gaikar Yash Mestry		04/26/2023	Done
1.5		Results and Analysis	Detailed outcomes of the project	Yash Mestry Purvika Gaikat		04/26/2023	Done
1.6		Formatting		Purvika Gaikar		04/27/2023	Done
1.7		PPT		Mitwa Mistry		04/27/2023	Done

				Yash Mestry			
2	Setting up AWS infra				Hardware		Done
2.1		Creating Aws accounts	Creating IAM users for all team members on AWS	Yash Mestry		03/04/2023	Done
2.3		Setting up EC2 instance to host website	Create 2 ec2 instance with some server code that is vulnerable	Yash Mestry	Hardware over cloud	04/04/2023	Done
2.4		Setting up Database Cluster for	Creating an RDS cluster	Yash Mestry	Hardware over cloud	04/16/2023	Done

		hosting web server	with a database for vulnerable website				
2.5		Setting up SonarQube Instance		Mitwa Mistry	Hardware over cloud	04/22/2023	Done
2.6		Setting up OWASP ZAP instance		Purvika Gaikar	Hardware over cloud	04/16/2023	Done
2.7		Setting up Code Build environmen ts		Yash Mestry	Hardware over cloud		Done
2.8		Setting up Codepipeli ne with only DevOps		Yash Mestry	Hardware over cloud	04/20/2023	Done

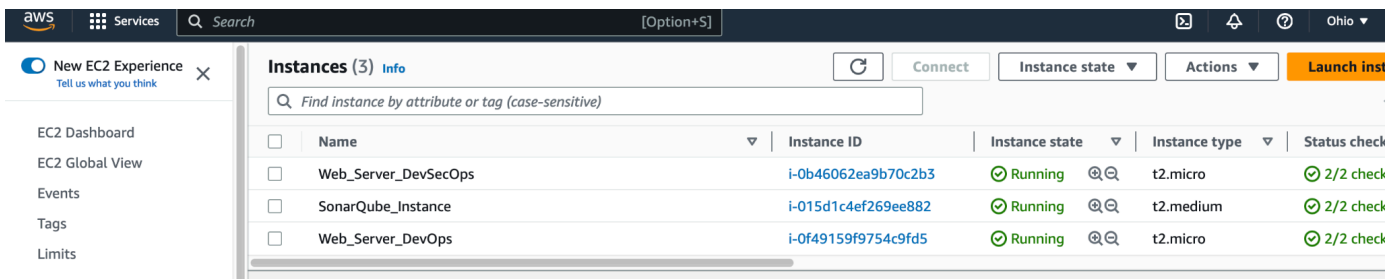
2.9		Setting up Codepipeline with DevSecOps		Yash Mestry Purvika Gaikar Mitwa Mistry	Hardware over cloud	04/22/2023	Done
3	Setting up Project Code	Setting up project code for deployment on EC2 instance on AWS		Purvika Gaikar Mitwa Mistry	Software	04/18/2023	Done
4.	Deploying Project Code	Using CI/CD pipelines to deploy code on ec2 instances		Yash Mestry Purvika Gaikar	Software	04/22/2023	Done
5.	Conducting	Conducting		Yash	Reporting	04/24/2023	Done

	Analysis	Analysis between DevSecOps and DevOps pipelines		Mestry Mitwa Mistry			
6.	Drafting Conclusions	Drafting and formulating conclusions to show analysis on code		Yash Mestry Purvika Gaikar Mitwa Mistry		04/24/2023	Done

A. Infrastructure Implementation

Initially, we have a basic Virtual Private Cloud (VPC) setup on Amazon Web Services (AWS). It additionally permits us to keep track of traffic to and from the EC2 instances by enabling the construction of public subnets. After setting up and creating, we created six separate subnets: 3 public subnets and 3 private ones. All the EC2 instances are in the public subnet while our RDS cluster is in the private subnet. The high amount of subnets provide better availability across AWS edge locations and is the recommended default configuration for AWS. One of the Ec2 instances named “SonarQube_Instance” we have Installed

and configured Sonarqube on a docker container to perform code analysis on the static code during the DevSecOps Pipeline's SAST stage. For the DevOps instance, we then install and set up software development technologies such as Git and use ssh and scp protocols to deploy and manage our applications. Finally, infrastructure deployment for two WebServers in the form of EC2 instances: EC2 dev instance (DevOps), and EC2 prod instance (DevSecOps), necessitates meticulous design, configuration, and monitoring. The Figure below describes the instances deployed on AWS for our Web-Servers and SonarQube Analysis.



Name	Instance ID	Instance state	Instance type	Status check
Web_Server_DevSecOps	i-0b46062ea9b70c2b3	Running	t2.micro	2/2 check
SonarQube_Instance	i-015d1c4ef269ee882	Running	t2.medium	2/2 check
Web_Server_DevOps	i-0f49159f9754c9fd5	Running	t2.micro	2/2 check

FIG 1. INSTANCES

B. MYSQL RDS cluster

For our database we have added an RDS cluster that has our MySQL Database for the website. It has a database named “tables” that has all the schemas required for our website for the demo.

The screenshot displays the AWS RDS console interface for a database instance named 'database-1'. The top navigation bar includes the AWS logo, a search bar, and user information. The breadcrumb trail shows 'RDS > Databases > database-1'. The main header for the instance is 'database-1', with 'Modify' and 'Actions' buttons. Below this is a 'Summary' section with a table of instance details:

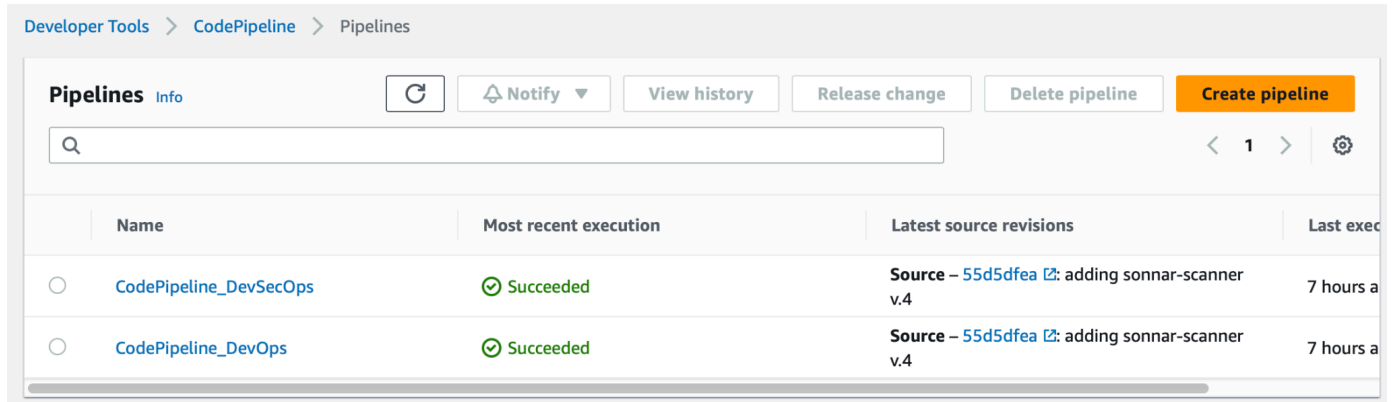
Summary			
DB identifier database-1	CPU 2.78%	Status Available	Class db.t3.micro
Role Instance	Current activity 0 Connections	Engine MySQL Community	Region & AZ us-east-2c

Below the summary is a horizontal menu with tabs: 'Connectivity & security' (selected), 'Monitoring', 'Logs & events', 'Configuration', 'Maintenance & backups', and 'Tags'. The 'Connectivity & security' tab is active, showing a table with three columns: 'Endpoint & port', 'Networking', and 'Security'.

Connectivity & security		
Endpoint & port	Networking	Security
Endpoint database-1.c5hsdzkic3gc.us-east-2.rds.amazonaws.com	Availability Zone us-east-2c	VPC security groups launch-wizard-2 (sg-0e801dfd4ba79a7c0)
Port 3306	VPC vpc-0ddfab22a5fd9d181	Active
	Subnet group default-vpc-0ddfab22a5fd9d181	Publicly accessible Yes

FIG 2. DATABASE

After that we use AWS CodePipeline to create our CI/CD pipelines. AWS CodePipeline is a fully managed continuous delivery solution that allows modeling, visualization, and automate the steps involved in the software release process. For the actual build stage in the pipeline we use AWS CodeBuild; a fully managed build service that compiles your source code, performs tests, and generates ready-to-deploy software packages. Together, CodePipeline and CodeBuild provide us two DevOps and DevSecOps pipelines as seen in the screenshot below.



Name	Most recent execution	Latest source revisions	Last execution
CodePipeline_DevSecOps	✓ Succeeded	Source – 55d5dfea link : adding sonnar-scanner v.4	7 hours ago
CodePipeline_DevOps	✓ Succeeded	Source – 55d5dfea link : adding sonnar-scanner v.4	7 hours ago

FIG 3. DEVOPS & DEVSECOPS PIPELINES

A sequence of stages is performed in a DevOps pipeline to deploy code from source to build and then deploy to the server. For our purpose, the source code is initially committed on a GitHub repository, which acts as the pipeline's source. A build stage is then added to the pipeline, where the code is compiled and assembled in a deployable manner using AWS CodeBuild. The code would then be deployed to the EC2 dev instance using our ssh key and scp transfer protocols.

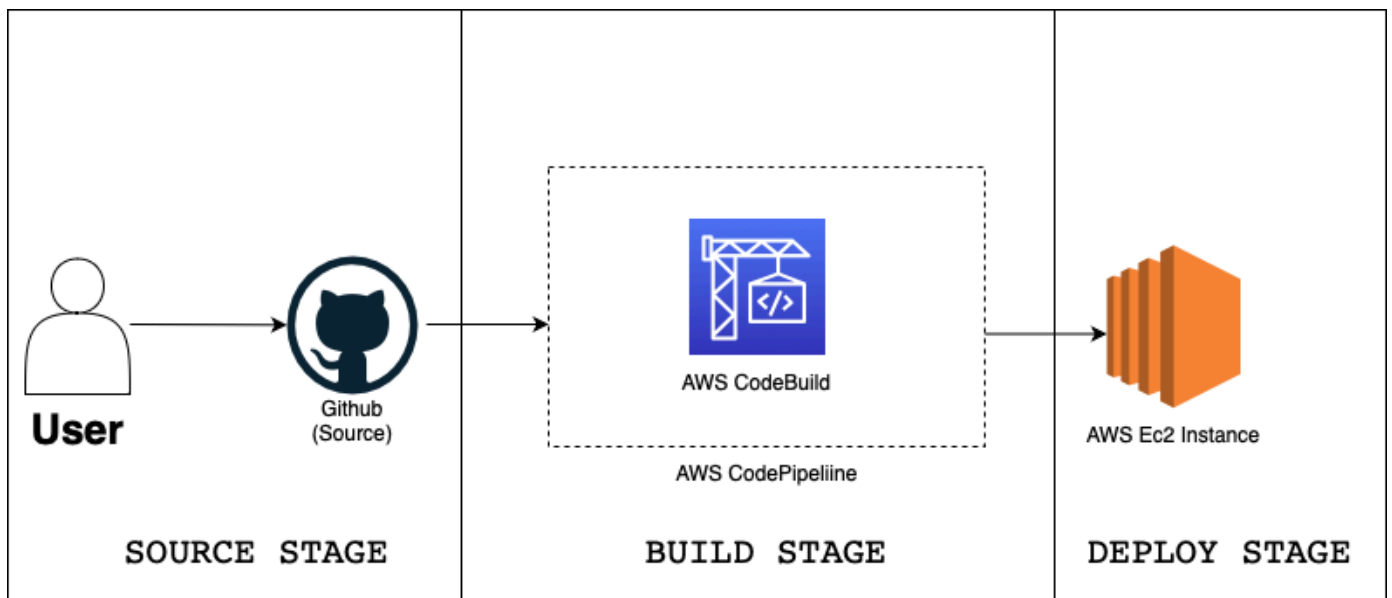


FIG 4. DEVOPS STAGES

The code-snippet below is of the 'buildspec.yaml' file which is essentially a script for AWS CodeBuild and it depicts the website application's build and deployment process. The package repository on the build server is first updated, and then a private key file from an S3 bucket is copied to the build server. This key is used to ssh into our web server. To secure private key files, permissions are modified so that only the owner can read them. The contents of the web server directory are then copied to a ubuntu based ec2 instance. Once the files are copied to the instance, ssh protocol is used to connect to the remote Ec2 instance and the script copy_to_server.sh is executed.

```
version: 0.2

phases:

  install:

    runtime-versions:
      php: 8.1

    commands:
      - yum update -y

  build:

    commands:
      - echo Deploying Code to WebServer
      - aws s3 cp s3://devopsbucketstorage/Keys/DevOps_Key.pem .
      - chmod 400 DevOps_Key.pem
```

```
- scp -i DevOps_Key.pem -o StrictHostKeyChecking=no -r ./Web_Server/*  
ubuntu@13.59.252.207:./Web_Server  
- ssh -i DevOps_Key.pem ubuntu@13.59.252.207 'sudo ./copy_to_server.sh'
```

FIG 5. BUILD & DEPLOYMENT PROCESS

The `copy_to_server.sh` is a bash script that copies and overwrites files from the `Web_Server` directory to the Apache web server's document root directory at `/var/www/html`. The `-R` flag instructs `cp` to recursively copy directories, while the `-f` flag instructs it to overwrite files without prompting for confirmation.

The code snippet of the `Copy_to_Server.sh` script is given below:

```
#!/bin/bash  
  
# Copy and overwrite files from Web_Server to /var/www/html  
sudo cp -Rf Web_Server/* /var/www/html/
```

Once the build is complete we can then see the changes committed on the web server instance named `WebServer_DevOps`.

On the other hand the designed basic DevSecOps pipeline contains several crucial stages to support secure software development and deployment. Each step is broken down below:

- Source: The source code is stored in a GitHub repository and acts as the pipeline's source.
- Static Analysis: Before deploying the code, it is statically examined utilizing SonarQube to discover any possible security flaws or code quality issues.
- Deploy: After the code has passed static analysis, it is deployed to an EC2 production instance.

- **Dynamic Analysis:** The deployed code is then subjected to dynamic analysis using an OWASP ZAP docker container. Using the Dynamic testing and automated crawlers on ZAP named spider and ajax-spiders we are able to identify vulnerabilities and security-hotspots on the Web Server after the code has been deployed.
- **Storage:** The results of the dynamic analysis are saved in an S3 bucket, where they can be promptly accessed and reviewed.
- **Rollback:** If a vulnerability is detected during the dynamic analysis, the pipeline is designed to roll back to the most recent stable state. This assures that any security vulnerabilities are resolved before releasing the code into production.

By including both static and dynamic analysis in the pipeline, potential vulnerabilities are detected at an early stage of the development process while providing safeguards in the instance that any vulnerabilities are discovered later in the deployment. Furthermore, the results of the dynamic analysis are stored in an S3 bucket, and monitoring the development of the application's security becomes manageable over time, which helps to discover trends and potential areas for improvement.

Refer to figure below for DevSecOps Pipeline:

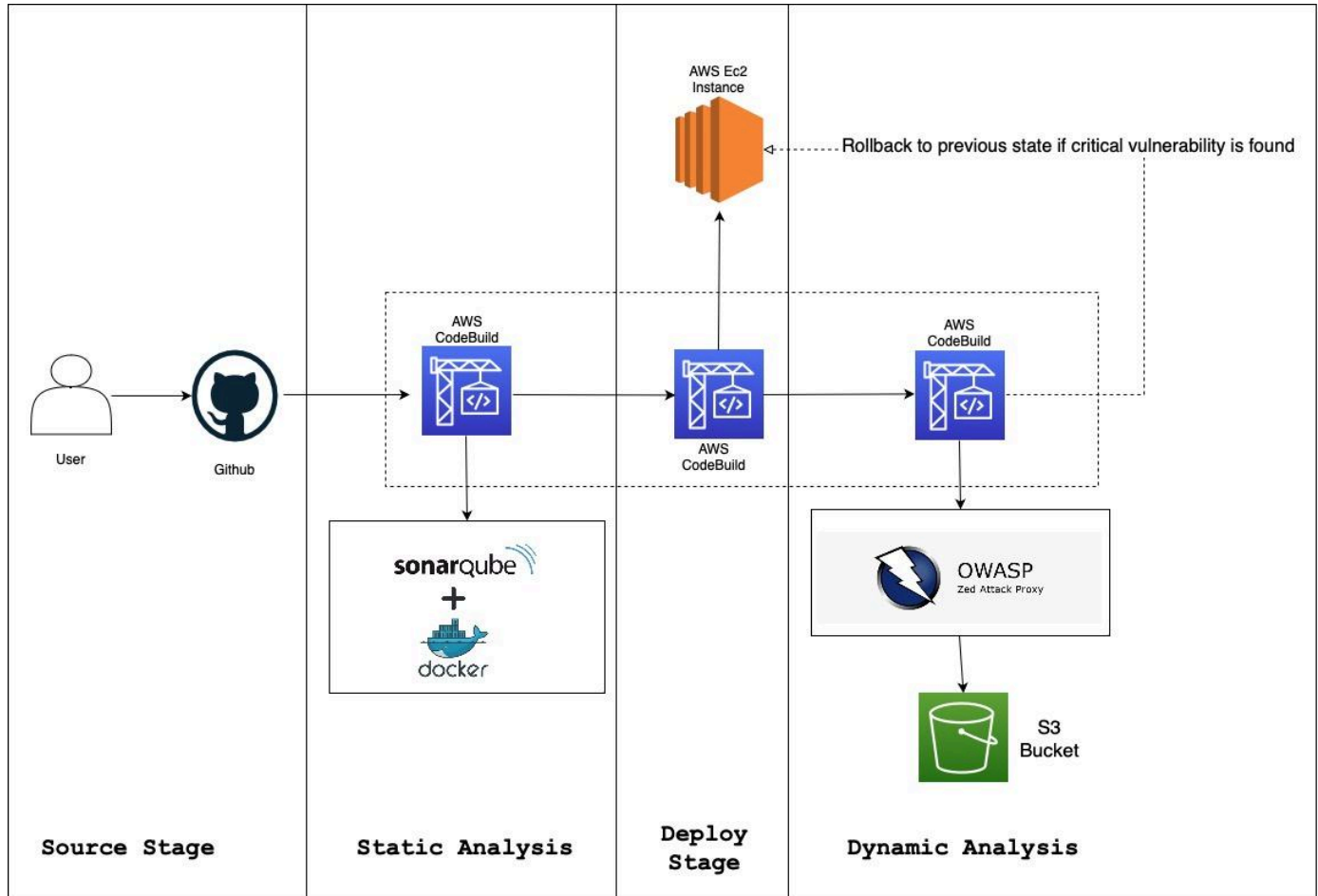


FIG 6. BASIC DEVSECOPS CI/CD PIPELINE WITH SAST AND DAST

The following is the ‘buildspec yaml file’ script, which includes steps for installing the required software, conducting static and dynamic analysis, deploying the code to a web server, and performing a security scan.

The following is a breakdown of each phase and its steps:

The install phase installs the pipeline's required runtime versions and packages. Specifically, it installs PHP 8.1 and Docker version 20.

The tasks in the `pre_build` phase deploy SonarScanner and conduct SAST (Static Application Security Testing) on the code. It installs the SonarScanner CLI, configures multiple permissions and environment variables, and then executes the scanner on the instance, sending the results to a specified URL. It also provides a message that directs to the report.

The `Build` phase is responsible for deploying the code to the web server. It first copies the `DevOps_Key.pem` file from an S3 bucket, subsequently sets the appropriate permissions, and transfers the contents to a specified server using SCP. Following that, it executes a script on the server through ssh to implement the changes.

During the `Post_build` phase, the OWASP ZAP tool is utilized to conduct dynamic analysis. It performs a complete scan on the web server using the Ajax spider and stores the results in a file. If a critical vulnerability is discovered, it prints a message, initiates a rollback to the last stable state, and outputs a message with instructions to review the most recent report for additional information. The report gets uploaded to an S3 bucket, and a message is displayed indicating the build failed. Otherwise, it displays a success message and quits with the code 0.

Overall, this pipeline establishes a DevSecOps approach to detect vulnerabilities in code by using both static and dynamic analysis techniques. It also includes a technique for promptly restoring to the last known stable state if a critical vulnerability is discovered during the dynamic analysis phase.

```
version: 0.2

phases:

install:

runtime-versions:
```

```
php: 8.1

docker: 20

pre_build:

commands:

- echo Installing Docker

- yum update -y

- echo Installing SonarScanner and conducting SAST

- wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.6.2.2472-linux.zip

- unzip sonar-scanner-cli-4.6.2.2472-linux.zip

- mv sonar-scanner-4.6.2.2472-linux /opt/sonar-scanner

- chmod -R 775 /opt/sonar-scanner

- /opt/sonar-scanner/bin/sonar-scanner -Dsonar.projectKey=PMY-COSMETICS

-Dsonar.host.url=http://3.142.246.193:9000

-Dsonar.token=sqp_d936b90e10fbb9eb9778f7a1387860dbe3407aaf

- echo "STATIC ANALYSIS IS DONE PLEASE GO to http://3.142.246.193:9000 to get your report"

build:

commands:

- echo Deploying Code to WebServer

- aws s3 cp s3://devopsbucketstorage/Keys/DevOps_Key.pem .

- chmod 400 DevOps_Key.pem
```

```
- scp -i DevOps_Key.pem -o StrictHostKeyChecking=no -r ./Web_Server/*
ubuntu@3.12.136.142:./Web_Server

- ssh -i DevOps_Key.pem ubuntu@3.12.136.142 'sudo ./copy_to_server.sh'

post_build:

commands:

- echo Running ZAP Full Scan with Ajax Spider for conducting DAST

- |

set +e # disable the 'exit immediately on error' option

docker run -v $(pwd):/zap/wrk/:rw -t owasp/zap2docker-stable zap-full-scan.py -t http://3.12.136.142
-c configfiles/tests.conf -d > report.txt

if grep -q -E 'FAIL-NEW: [1-9]+' report.txt; then # check if any new failures were detected

aws s3 cp report.txt s3://devopsbucketstorage/Reports/OWASP_ZAP_Report_$(date
+"%Y-%m-%d-%H-%M-%S").txt

echo "BUILD FAILED AND THE LAST KNOWN STABLE STATE WAS RESTORED BECAUSE
A CRITICAL VULNERABILITY WAS DETECTED. PLEASE LOOK AT THE LATEST REPORT FOR
FURTHER ANALYSIS"

ssh -i DevOps_Key.pem ubuntu@3.12.136.142 'sudo ./implement_rollback.sh'

exit 0

else

aws s3 cp report.txt s3://devopsbucketstorage/Reports/OWASP_ZAP_Report_$(date
+"%Y-%m-%d-%H-%M-%S").txt

echo "BUILD WAS SUCCESSFUL and all security standards are met and no critical vulnerabilities
were detected"
```



```
exit 0  
  
fi  
  
set -e
```

FIG 7. DEPLOYMENT SCRIPT

Script to Copy and Overwrite Files from Web_Server to /var/www/html for devsecops:

```
#!/bin/bash  
  
# Copy and overwrite files from Web_Server to /var/www/html  
  
sudo cp -Rf /var/www/html/* rollback/  
  
sudo cp -Rf Web_Server/* /var/www/html/
```

Script for Rolling Back Files from Web_Server to /var/www/html:

```
#!/bin/bash  
  
# Copy and overwrite files from Web_Server to /var/www/html  
  
sudo rm -r /var/www/html/*  
  
sudo cp -Rf rollback/* /var/www/html/
```

IV. RESULTS AND ANALYSIS

For the purpose of testing we pushed multiple UI changes to the website. When the code is committed both pipelines(CodePipeline_DevOps and CodePipeline_DevSecOps)trigger simultaneously.

For example we pushed a change to update the background image of the main homepage of the website.

When the changes were pushed, the DevOps pipeline ran the script accordingly and pushed the changes to the WebServer_DevOps Instance quite quickly and notified the developer that the build succeeded.

The figure below depicts how the DevOps pipeline is expected to achieve success as it does not include any testing or analysis.

```
[Container] 2023/04/22 19:40:12 Running command echo Deploying Code to WebServer
Deploying Code to WebServer

[Container] 2023/04/22 19:40:12 Running command aws s3 cp s3://devopsbucketstorage/Keys/DevOps_Key.pem .
Completed 1.6 KiB/1.6 KiB (23.5 KiB/s) with 1 file(s) remaining
download: s3://devopsbucketstorage/Keys/DevOps_Key.pem to ./DevOps_Key.pem

[Container] 2023/04/22 19:40:23 Running command chmod 400 DevOps_Key.pem

[Container] 2023/04/22 19:40:23 Running command scp -i DevOps_Key.pem -o StrictHostKeyChecking=no -r ./Web_Server/*
ubuntu@13.59.252.207: ./Web_Server
Warning: Permanently added '13.59.252.207' (ECDSA) to the list of known hosts.

[Container] 2023/04/22 19:40:24 Running command ssh -i DevOps_Key.pem ubuntu@13.59.252.207 'sudo ./copy_to_server.sh'

[Container] 2023/04/22 19:40:24 Phase complete: BUILD State: SUCCEEDED
[Container] 2023/04/22 19:40:24 Phase context status code: Message:
[Container] 2023/04/22 19:40:24 Entering phase POST_BUILD
[Container] 2023/04/22 19:40:24 Phase complete: POST_BUILD State: SUCCEEDED
[Container] 2023/04/22 19:40:24 Phase context status code: Message:
```

FIG 8. DEVOPS

The Figure below depicts how the background image of the website changes once the DevOps Pipeline succeeds.

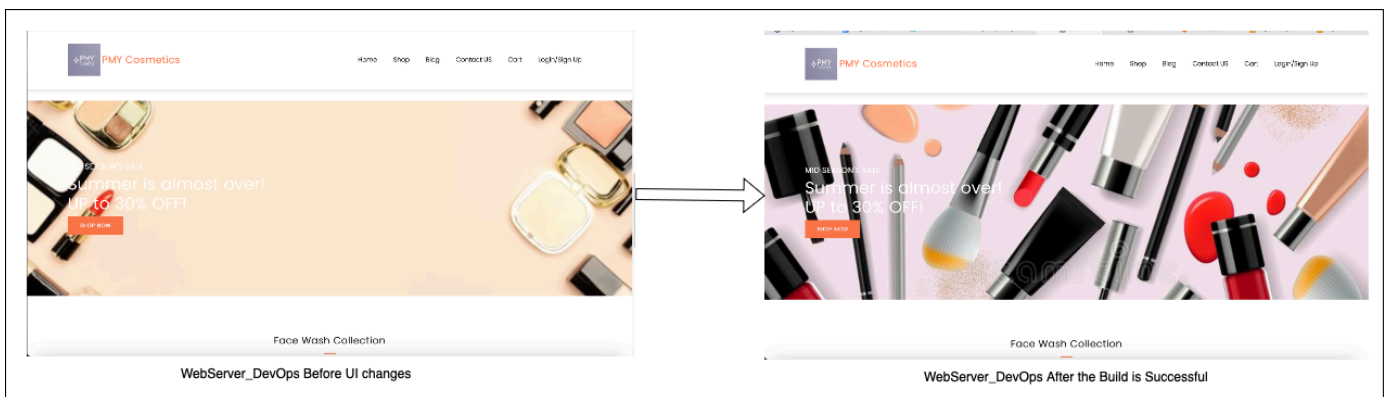


FIG 9. DEVOPS PIPELINE STAGES ON WEBSITE

On the other hand however, the DevSecOps pipeline will initially deploy the changes after conducting Static Analysis but will fail during Dynamic analysis due to the presence of a SQL injection vulnerability. Due to this the pipeline will implement rollback after it detects the vulnerability during dynamic analysis and revert the code back to the last known stable secure state. The figure below shows how the build notifies the developer that it failed because a security vulnerability was detected and that it restored it to the original state.

```

982 4cd4db6b10d5: Pull complete
983 4f4fb700ef54: Pull complete
984 b5d2d6820096: Pull complete
985 34b52258d45e: Pull complete
986 05230fb900de: Pull complete
987 b4fa603a1b81: Pull complete
988 21208d8d30d1: Pull complete
989 a57760bd63c6: Pull complete
990 5824ef294a16: Pull complete
991 545082a40d3c: Pull complete
992 ad463afee06e: Pull complete
993 400d42be1c85: Pull complete
994 c0cee4208540: Pull complete
995 Digest: sha256:58fc221e9db954a1188a49592774310a5def902536f2ad0e24bf7d4d72a4981c
996 Status: Downloaded newer image for owasp/zap2docker-stable:latest
997 Completed 61.3 KiB/61.3 KiB (769.6 KiB/s) with 1 file(s) remaining
998 upload: ./report.txt to s3://devopsbucketstorage/Reports/OWASP_ZAP_Report_2023-04-22-19-47-26.txt
999 BUILD FAILED AND THE LAST KNOWN STABLE STATE WAS RESTORED BECAUSE A CRITICAL VULNERABILITY WAS DETECTED. PLEASE LOOK AT THE LATEST
    REPORT FOR FURTHER ANALYSIS

```

FIG 10. DEVSECOPS PIPELINE FAILING

The image presented depicts the WebServer_DevSecOps and the modifications it undergoes as the pipeline operates.

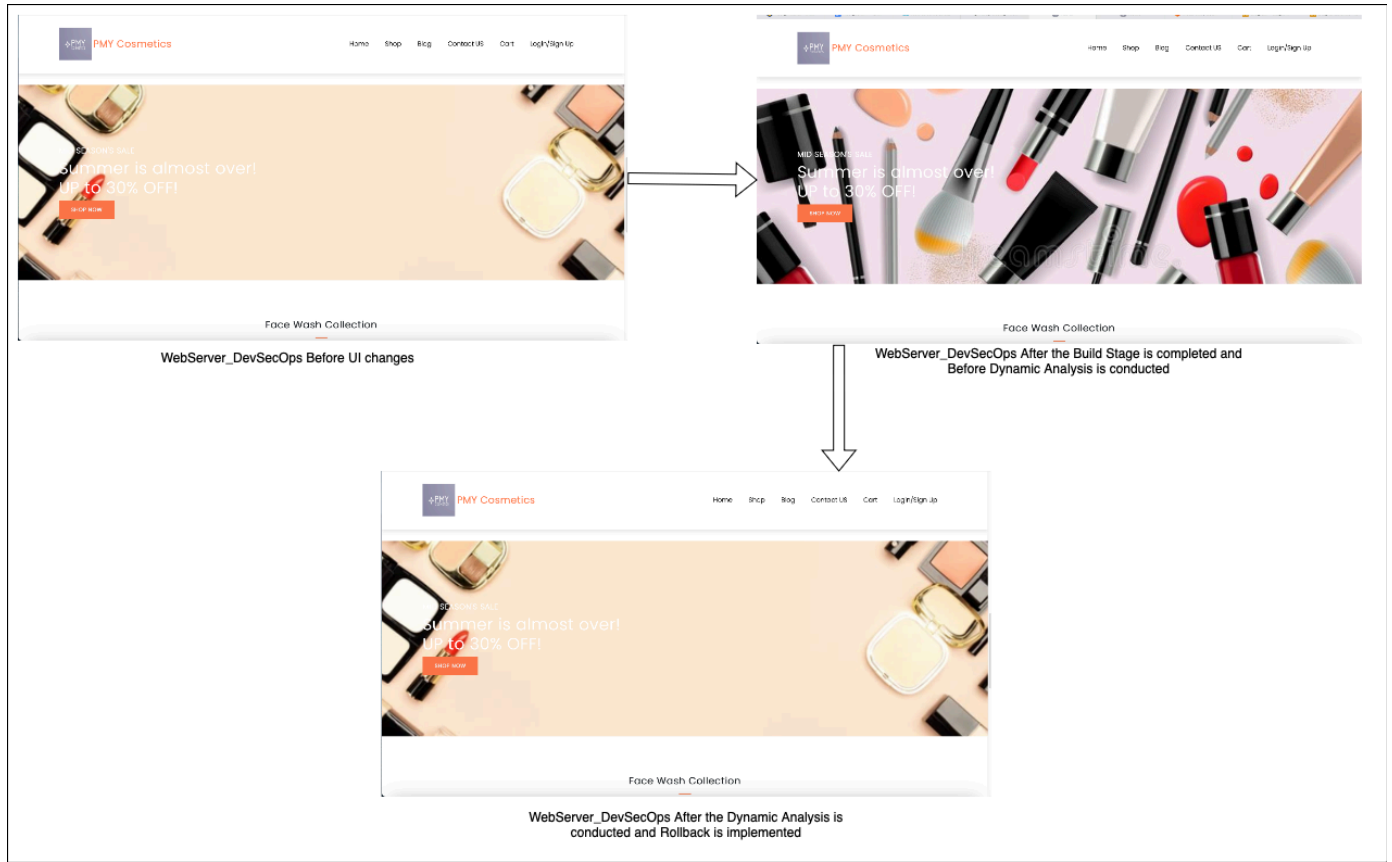


FIG 11. DEVSECOps PIPELINE STAGES ON WEBSITE

Once the vulnerability has been fixed and both pipelines have been executed successfully, the modifications will be visible on both of the web servers.

A. Comparative Analysis between DevOps and DevSecOps :

TABLE 2. COMPARISON BETWEEN DEVOPS & DEVSECOps

Sr. no	Aspects	DevOps	DevSecOps
1.	Definition	A software development	A software development

		methodology that prioritizes collaboration, communication, and integration among software developers and IT operations teams.	methodology that incorporates security into the DevOps process, ensuring that security is taken into account throughout the software development lifecycle.
2.	Security consideration	Although security is not the primary focus, it is integrated into the process to guarantee that security concerns are addressed	Security is a top priority, and it is included in every level of the software development lifecycle.
3.	Goal	To deliver software in a fast and effective way, with an emphasis on continuous integration and delivery (CI/CD) and automation	Secure software fast and efficiently, with a focus on security integration and automation in the CI/CD pipeline.
4.	Pipeline time	An average of 3 ½	Took an average of 8 ½

		minutes to run.	minutes to run (based on codebase size and time required to run SAST and DAST respectively)
5.	Deployment of Vulnerable Code	Vulnerable code was deployed on instance	Vulnerable code was not pushed
6.	Security Measures	Security was not the primary focus	Security was integrated from the beginning
7.	Prevention of Security Vulnerability	Vulnerabilities were not prevented	SQL injection was prevented
8.	Notification of Developer	No notification was given	Developers were notified and informed why

9.	Vulnerability Detection	Post-Deployment	Pre-Deployment
10.	Vulnerability Prevention	Not Prevented	Prevented
11.	Overall Security	Security was an afterthought	Security was integrated from the beginning
12.	Risk Mitigation	Reactive	Proactive

B. SonarQube

In this test, SonarQube discovered that the host url of the RDS Cluster, the login and password and the database named “tables” for "database-1" were hardcoded into the code during a static analysis scan of a software application. This is a significant security risk because anyone with access to the code may quickly obtain these credentials and potentially gain unauthorized access to the database. Hard Coding credentials

in code is typically regarded as bad practice because it can lead to security issues and makes modifying or updating the credentials difficult when necessary.

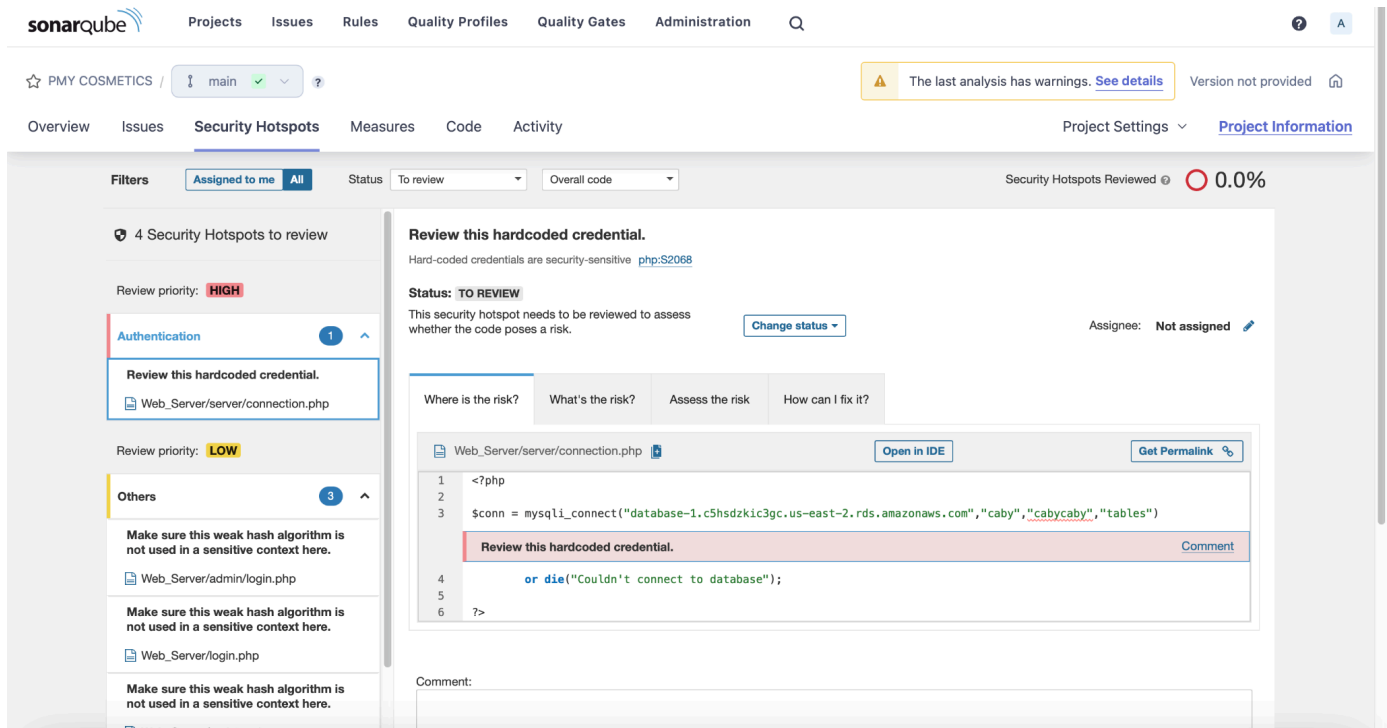


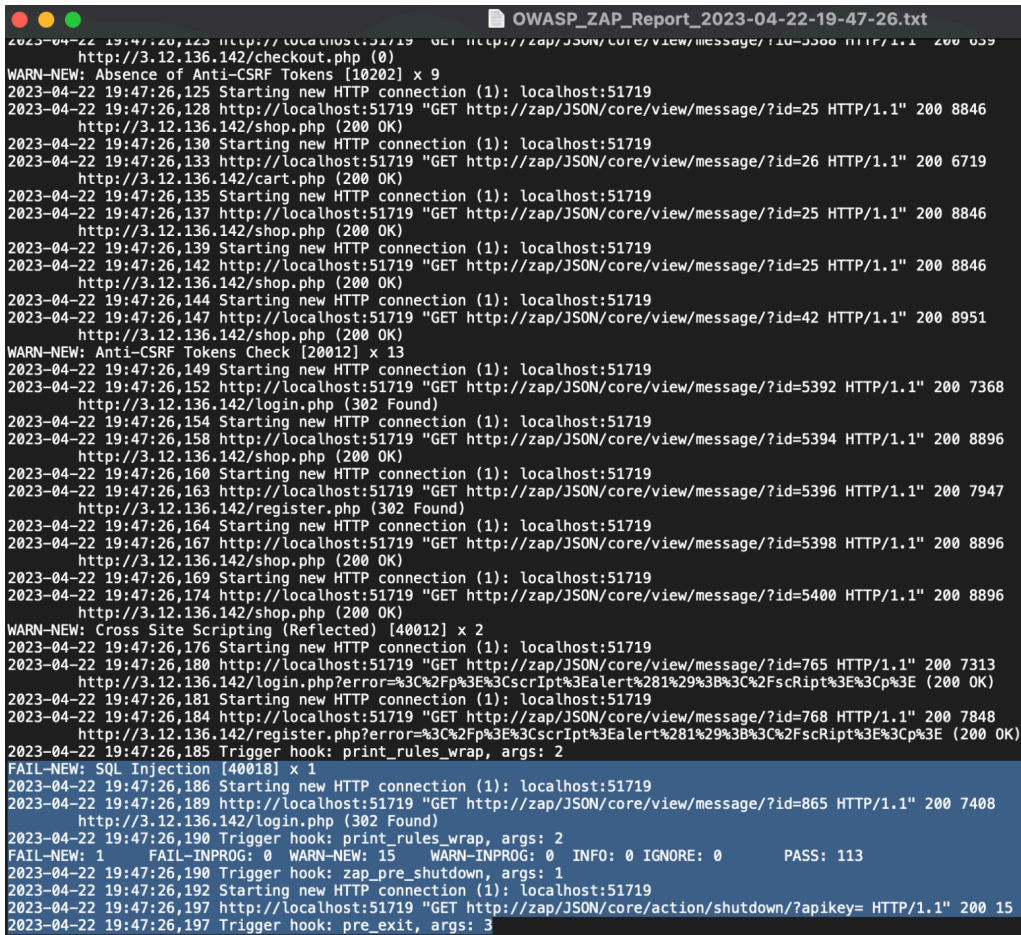
FIG 12. SONARQUBE REVIEW

C. ZAP Dynamic tests

A ZAP scan involved a number of tests, and the program generated a report with information on the detected vulnerabilities. A ZAP scan was done in the stipulated scenario, and the findings were examined. The term "trigger hook" refers to a test that was run. The parameters shown below indicate the number of vulnerabilities discovered during the scan:

"fail-new:1" - One new vulnerability that failed the scan criteria was identified, "fail-inprog:0" - No previously tracked vulnerabilities failed the scan criteria, "warn-new:15" identifies 15 new vulnerabilities as warnings, and "pass:123" - A total of 123 vulnerabilities passed the scan criteria without being marked

as potential security threats. Overall, this information provides insight into the types and magnitudes of vulnerabilities detected during the ZAP scan.



```

OWASP_ZAP_Report_2023-04-22-19-47-26.txt
2023-04-22 19:47:26,123 http://localhost:51719 GET http://zap/JSON/core/view/message/?id=3300 HTTP/1.1" 200 639
http://3.12.136.142/checkout.php (0)
WARN-NEW: Absence of Anti-CSRF Tokens [10202] x 9
2023-04-22 19:47:26,125 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,128 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=25 HTTP/1.1" 200 8846
http://3.12.136.142/shop.php (200 OK)
2023-04-22 19:47:26,130 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,133 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=26 HTTP/1.1" 200 6719
http://3.12.136.142/cart.php (200 OK)
2023-04-22 19:47:26,135 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,137 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=25 HTTP/1.1" 200 8846
http://3.12.136.142/shop.php (200 OK)
2023-04-22 19:47:26,139 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,142 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=25 HTTP/1.1" 200 8846
http://3.12.136.142/shop.php (200 OK)
2023-04-22 19:47:26,144 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,147 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=42 HTTP/1.1" 200 8951
http://3.12.136.142/shop.php (200 OK)
WARN-NEW: Anti-CSRF Tokens Check [20012] x 13
2023-04-22 19:47:26,149 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,152 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=5392 HTTP/1.1" 200 7368
http://3.12.136.142/login.php (302 Found)
2023-04-22 19:47:26,154 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,158 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=5394 HTTP/1.1" 200 8896
http://3.12.136.142/shop.php (200 OK)
2023-04-22 19:47:26,160 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,163 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=5396 HTTP/1.1" 200 7947
http://3.12.136.142/register.php (302 Found)
2023-04-22 19:47:26,164 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,167 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=5398 HTTP/1.1" 200 8896
http://3.12.136.142/shop.php (200 OK)
2023-04-22 19:47:26,169 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,174 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=5400 HTTP/1.1" 200 8896
http://3.12.136.142/shop.php (200 OK)
WARN-NEW: Cross Site Scripting (Reflected) [40012] x 2
2023-04-22 19:47:26,176 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,180 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=765 HTTP/1.1" 200 7313
http://3.12.136.142/login.php?error=%3C%2Fp%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cp%3E (200 OK)
2023-04-22 19:47:26,181 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,184 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=768 HTTP/1.1" 200 7848
http://3.12.136.142/register.php?error=%3C%2Fp%3E%3Cscript%3Ealert%281%29%3B%3C%2Fscript%3E%3Cp%3E (200 OK)
2023-04-22 19:47:26,185 Trigger hook: print_rules_wrap, args: 2
FAIL-NEW: SQL Injection [40018] x 1
2023-04-22 19:47:26,186 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,189 http://localhost:51719 "GET http://zap/JSON/core/view/message/?id=865 HTTP/1.1" 200 7408
http://3.12.136.142/login.php (302 Found)
2023-04-22 19:47:26,190 Trigger hook: print_rules_wrap, args: 2
FAIL-NEW: 1 FAIL-INPROG: 0 WARN-NEW: 15 WARN-INPROG: 0 INFO: 0 IGNORE: 0 PASS: 113
2023-04-22 19:47:26,190 Trigger hook: zap_pre_shutdown, args: 1
2023-04-22 19:47:26,192 Starting new HTTP connection (1): localhost:51719
2023-04-22 19:47:26,197 http://localhost:51719 "GET http://zap/JSON/core/action/shutdown/?apikey= HTTP/1.1" 200 15
2023-04-22 19:47:26,197 Trigger hook: pre_exit, args: 3

```

FIG 13. ZAP SCAN

D. Benefits

Leveraging DevSecOps for cloud security can provide numerous benefits, including improved security posture, faster detection and response times, and reduced security costs. However, implementation challenges and the need for effective tooling and automation must be carefully considered to achieve optimal results. Software development teams are able to deliver code more rapidly, effectively, and error-free through the use of DevSecOps pipelines. Teams can reduce the time it takes to advance from development to production by utilizing automation and collaboration tools to organize their development workflows.

Enhanced security: DevSecOps pipelines assist teams in identifying and addressing possible security concerns early in the development cycle by integrating security into the development process. Code that is secure as a result is less vulnerable to threats and exploits. Collaboration between development, security, and operations teams is encouraged via DevSecOps pipelines, which promote a shared responsibility for the delivery and security of software. As a result, communication and teamwork are improved and silos are broken down. Better software: DevSecOps pipelines assist teams to build more efficient software with fewer defects by discovering bugs and security issues early in the development cycle. As a result, there are fewer production-related problems, better customer experiences, and better commercial results.

Continuous improvement is fostered through DevSecOps pipelines, which encourage teams to seek for methods to streamline and optimize their procedures. Teams could deliver software more quickly, securely, and of better quality as a result.

V. CONCLUSION

Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are two key tools used in DevOps and DevSecOps to identify and address potential security vulnerabilities in software applications. While both approaches can be used in both DevOps and DevSecOps, the difference lies in how they are integrated into the development process. In DevOps, SAST and DAST tools are typically used to identify vulnerabilities at various stages of the development cycle, from code creation to testing and deployment. The focus is on identifying and addressing these vulnerabilities as quickly as possible to ensure the smooth and efficient delivery of software. In DevSecOps, SAST and DAST tools are integrated into the development process from the start, with a strong emphasis on continuous testing and integration. The goal is to identify and address potential security issues as soon as possible, rather than waiting until the end of the development cycle.

In conclusion both DevOps and DevSecOps can benefit from the use of SAST and DAST tools, but DevSecOps takes a more proactive approach by integrating these tools from the start of the development process. By doing so, DevSecOps teams can identify and address potential security issues more quickly, resulting in more secure software and a better overall development process.

REFERENCES

- [1] Dhaya Sindhu Battina, “BEST PRACTICES FOR ENSURING SECURITY IN DEVOPS: A CASE STUDY APPROACH”, Int. j. innov. eng. res. technol., vol. 4, no. 11, pp. 38–45, Apr. 2017.
- [2] Lyubka Dencheva “Comparative analysis of Static application security testing (SAST) and Dynamic application security testing (DAST) by using open-source web application penetration testing tools”, <https://norma.ncirl.ie/5956/1/lyubkadencheva.pdf> .
- [3] “Best Practices for Ensuring Security in DevOps: A Case Study Approach” Rajavi Desai and T N Nisha 2021 J. Phys.: Conf. Ser. 1964 042045
- [4] On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications , Appl. Sci. 2020, 10(24), 9119; <https://doi.org/10.3390/app10249119>
- [5] DevSecOps:A Multivocal Literature Review https://www.researchgate.net/publication/319633880_DevSecOps_A_Multivocal_Literature_Review
- [6] Security Tools in DevSecOps - A Systematic Literature Review <https://lu.diva-portal.org/smash/get/diva2:1727554/FULLTEXT01.pdf>
- [7] Challenges and solutions when adopting DevSecOps: A systematic review Roshan N. Rajapaksea,b,* , Mansooreh Zahedia , M. Ali Babara,b, Haifeng Shenc <https://arxiv.org/pdf/2103.08266.pdf>
- [8] <https://mydeveloperplanet.com/2021/05/11/automate-zap-with-docker/>
- [9] <https://docs.sonarqube.org/9.8/try-out-sonarqube/>
- [10] <https://aws.amazon.com/blogs/devops/integrating-sonarcloud-with-aws-codepipeline-using-aws-code-build/>

- [11] <https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html>
- [12] <https://aws.amazon.com/ec2/>
- [13] <https://aws.amazon.com/rds/>
- [14] <https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html>
- [15] <https://aws.amazon.com/what-is/devsecops/>
- [16] <https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-dast-tools/>
- [17] <https://www.xenonstack.com/blog/devsecops-pipeline-aws>
- [18] <https://www.zaproxy.org/docs/docker/full-scan/>
- [19] <https://karthickcse05.medium.com/api-web-application-security-scan-using-owasp-zap-in-jenkins-aws-code-build-d77c8af48478>