National College of Ireland

Securing CI/CD Pipeline: Automating the detection of misconfigurations and integrating security tools

MSc Industrial Internship
MSc. Cybersecurity

Muskan Mangla
Student ID: X21162697

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Muskan Mangla
Student ID:	X21162697
Programme:	MSc. Cybersecurity **Year:** 2022-2023
Module:	MSc Industrial Internship
Supervisor:	Vikas Sahni
Submission Due Date:	06th January 2023
Project Title:	Securing CI/CD Pipeline: Automating the detection of misconfigurations and integrating security tools
Word Count:	5990 **Page Count**: 20

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: Muskan

Date: 04th January 2023

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):	

Securing CI/CD Pipeline: Automating the detection of misconfigurations and integrating security tools

Muskan Mangla
X21162697

Abstract

In recent years, the adoption of DevOps technology has become widespread in enterprises and private sectors. DevOps emphasizes communication and collaboration between development and operations teams to accelerate the delivery of software. One key component of DevOps is the use of continuous integration and continuous delivery (CI/CD) pipelines, which automate the process of building, testing, and deploying software. However, the use of CI/CD pipelines' lack of security oversights that introduces security risks due to the potential for security misconfigurations. These misconfigurations can lead to vulnerabilities that can be exploited by a malicious actor and thus making it important to identify and address them as part of the CI/CD process.

To address these security concerns, a DevSecOps approach was adopted, which integrates security into the CI/CD pipeline and ensures that the software being deployed is secure and free of vulnerabilities. This was achieved by continuous detection of security misconfigurations automatically in every stage of the CI/CD Pipeline and addressing them as part of the CI/CD process. By adopting a DevSecOps approach, organizations can provide continuous security assurance and strengthen the security of their CI/CD pipelines.

1 Introduction

The use of software development methodologies such as Agile, Waterfall, and DevOps has allowed organizations to efficiently deliver applications through scalable and automated processes. However, the rapid deployment of cloud applications has often led to security challenges in the production environment. To mitigate these risks, it is crucial for organizations to prioritize security in their DevOps pipeline, as this helps to prevent security issues from arising after deployment (Ahmed & Francis, 2019). By integrating security into the DevOps process, organizations can ensure that their applications are secure and free of vulnerabilities before they are deployed to the production environment (Kumar & Goyal, 2021)

The implementation of a CI/CD pipeline can pose significant security risks to an organization, including insecure code and misconfiguration. These threats can compromise the integrity of the pipeline and lead to the disclosure of sensitive information (Chernyshev, Baig, & Zeadally, 2021).

In this paper, a secure CI/ CD pipeline has been proposed by identifying security misconfigurations using the security testing tools in AWS services and Infrastructure as code

that is built and deployed in the CI and CD Pipeline. In addition, Source Code Scanning and Analysis (SCA) and Static Application Security Testing (SAST) will be conducted to identify the other major security risks. In this proposal, a count of ten security findings are manually validated to eliminate the false positives and the mitigation solutions are highlighted using AWS services to enhance the security of the CI and CD pipeline.

Research Question

RQ1: How to automatically detect and identify security misconfigurations in a CI/CD pipeline?

RQ2: Can the integration of security tools into a CI/CD pipeline improve security by automating the detection of security misconfigurations?

Research Objective

The purpose of this research paper was to implement the DevOps Pipeline to automate the code deployment and code build and enforce the security by detecting security misconfigurations and other vulnerabilities using SAST and code analysis techniques integrated into CI/CD Pipeline. This will help in assuring the continuous security of the CI/CD Pipeline. The major misconfigurations would be eliminated by implementing this proposed solution before the deployment stage.

Structure of the Paper

The remainder of the paper is structured as follows

Section 1 described the Introduction of the research topic along with the problem statement and the objective of the research paper. Section 2 is illustrated as a literature review of the previous work by cloud security researchers and authors. Section 3 provides the Research Methodology and Section 4 shows the design of the proposed model including the Security techniques and frameworks used to secure the CI/CD Pipeline. Further, Section 5 includes the implementation of the security solutions in the pipeline, and Section 6 highlighted the evaluation of DevOps and DevSecOps.

2 Related Work

2.1 DevOps, CI, and CD

DevOps is a way of collaborating DEV (Development) and OPS (Operations) personnel tasks to automate the processes unlikely the traditional deployment process. The DevOps implementation is outlined in this research paper (Ivanov & Smolander, 2018) where the authors have demonstrated the DevOps pipeline for serverless applications. DevOps benefits in the software development automation that includes source, test, deployment, and monitoring phases. CI is defined as the integration of code changes in the base code by the developers and post, that the changes are validated by quality testing tools and build the software. When this process is automated is known as CI. Whereas, the CD is a process of automating the code changes deploy to a staging or production environment. DevOps was implemented using Gitlab suite and Docker containers to run the jobs, However, this research

has not emphasized the security threats challenges, and mitigations in the deployed CI/CD pipeline. (Arachchi & Perera, 2018) described the CI/CD pipeline approach and the importance that has rapidly increased the efficiency of software development.

2.1.1 DevOps security challenges and existing solutions

DevSecOps is a movement that aims to modernize security methods for compatibility with DevOps. This study (Myrbakken & Colomo-Palacios, 2017) reviewed the literature on DevSecOps to provide an overview of its benefits and challenges. The results indicated that implementing security that can keep up with DevOps is challenging, but can offer significant benefits if successful. (Mao, et al., 2020) researched on DevSecOps practices that include process: Plan, Code, Build, Test, and Operation phases, Infrastructure: Secrets Management, Configuration Management, and Container security scanning.

The application security techniques with limitations are examined and future opportunities are highlighted in this paper (Chernyshev, et al., 2021). The underlined Application security techniques examined were Threat Modeling to identify potential vulnerabilities by following OWASP ASVS, Manual code reviews, SAST (static analysis) by running the tools such as Sonarqube, Snyk code analyzer, DAST(dynamic analysis), and Sofware composition analysis to identify known vulnerable dependencies in DevOps using dependency scanning tool. The shortcomings of this paper were missing the practical real-world scenarios implementation and analysis. The novelty of the research was not adequate
In this paper (Dhaya Sindhu, 2021), the DevOps benefits and challenges are highlighted along with mitigation strategies to overcome the shortcomings. The major issues that are mentioned include the communication gap between the development and security teams that leads to insecure and delayed delivery often. Fast delivery and short development cycles are the main focus areas of DevOps that can cause the business to a single vulnerability which further leads to security breaches and malicious assaults.
The security risks associated with the cloud infrastructure and deployment are lacked to detect due to the utilization of open source tools and innovative technologies. A simple bug or misconfiguration might lead to sensitive information disclosure. In addition, poor access and secrecy management are critical in the continuous delivery pipeline that has the possibility of exposing credentials to the internet that can further be exploited and gained admin access. In order to mitigate the highlighted challenges, the researcher has provided solutions that included the adoption of the DevSecOps model, efficient Privilege Access Management system, and enforce Secure Policies to ensure the continuous security of the DevOps infrastructure.
However, the limitation of this research lacked the integration of security tools to detect security issues. The detection of vulnerabilities was not performed for the DevOps infrastructure and illustrated the research only at the theoretical level. (Mohan, et al., 2018) detailed the security concerns in the Continuous deployment process that included manual security tests, participation of the security team, separation of roles, audit, security guidelines, and enforcement of access controls. The recommended solution was SecDevOps which

included proper enforcement of separation of duties, secret management, logging, and monitoring.
The objective of this paper (Battina & Sindhu, 2017) highlighted the security of DevOps and definitions for DevOps and DevSecOps. The security policies enforced throughout the DevOps process to detect and mitigate security issues are illustrated. The best practices included are identity and access management (IAM), configuration, vulnerability scanning, and restricted access. The research scope was not wide to carry out the research. The research paper highlighted only the theoretical level and missing real-world case studies or scenarios.

2.1.2 Misconfiguration attacks in CI/CD

The main focus of this research (Ahmed & Francis, 2019) tends to the integration of DevSecOps to avoid insecure services and misconfigurations while deploying the application using the CI/ CD pipeline. This research paper's methodology is to implement DevSecOps in the DevOps practice by adding security from the initialization stage. The security tools that consist of the Snyk dependency check were utilized to identify the vulnerabilities before the deployment of the application to secure the application. However, the research was performed in a limited scope and on a smaller scale. In addition, the mitigation for the discovered findings was not included.

The efficient configuration policies to mitigate the security threats that can occur by exploiting misconfiguration vulnerabilities are proposed in this paper (Torkura, et al., 2021). The authors have developed the secure baseline configuration by applying the security best practices from the Centre for Internet Security (CIS) cloud security benchmarks. Further, the proposed method which was Continuous security monitoring and auditing was implemented for providing continuous security assurance. This was achieved by continuously detecting the vulnerabilities related to misconfigurations on cloud resources. Therefore, the security posture of the monitored cloud infrastructure can be analyzed. The limitation of this research lacked the implementation of detection rules of misconfigurations for other Cloud services (Lambda, databases).

This research paper is significant to a novel Secure by Design methodology that was integrated into the development process (Casola, et al., 2020). The automated security testing process included security assessment and risk analysis to determine the potential threats using the STRIDE threat model. (Casola, et al., 2020) emphasized the Security SLA model to share the responsibility for security issues between Cloud Providers and Customers using the NIST, Cloud Control Matrix. In addition, the technical security metrics as well as operations security controls were underlined to ensure the configured policy or controls are implemented correctly. The paper highlighted the automated techniques for the risk analysis and security assessment of cloud applications and this research paper aimed to calculate the efficiency, usability, and time consumption for the evaluation of the results, they utilized online questionnaires and surveys to receive the feedback and analyze the efforts to deploy the tool by selecting the evaluation team who are not skilled in the security aspects. This research has

not highlighted detecting vulnerabilities using scanning tools and their usage and how the vulnerabilities were mitigated.

2.1.3 Continuous Security Testing in CI/CD Pipeline

In this paper (Rangnau et al. in 2020), security solutions for DevOps practices were proposed and demonstrated through the integration of three automated security testing tools. This practical, security-focused study highlighted techniques for automating dynamic security testing and examined the approach to implementing a CI/CD pipeline, including the selection of tools such as OWASP ZAP. Additionally, the integration and implementation of DAST in the CI/CD pipeline were examined to identify security issues. The objective of the research was successfully achieved, though challenges were encountered in meeting the requirements for DAST implementation. A case study involving the integration of three common security testing tools was conducted to address these challenges and provide solutions.

(Ullah, et al., 2017) has evaluated the security between a secure CD process and a non-secure CD Process. This paper aims to design a secure CD process by utilizing security tactics. The five security tactics have been proposed to secure the CD process by leveraging the AWS IAM ecosystem and use of a private SSH key Further, the analysis was performed using Qualys OWASP and ZAP scanner. (Sojan, et al., 2021) summarized and evaluated the need for security monitoring solutions to detect and monitor security threats and vulnerabilities.

To summarize, the previous papers described the principles and theoretical concepts of DevSecOps. The researchers highlighted the security challenges, risks, and opportunities for securing the DevOps Pipeline, however, the tools, benchmarks, frameworks, and methodology and their usage were lacked to for practical applicability. The key concept was to implement the security in DevOps Pipeline. However, most literature focused on the SAST and SCA tools to detect the vulnerabilities in the CI/CD Pipeline. In addition, the security mitigations were not enabled to secure the Pipeline from the misconfiguration attacks. Less is known about how to integrate security tools and techniques into the Code Pipeline. In this proposed solution, the limitations of the included related work were achieved by enabling the security tools for identifying the security issues and implementing the security solutions by AWS services and monitoring solutions to detect the malicious activities. This research method and solution will help the other researchers/ individuals/organization and authors to implement the Secure CodePipleline using AWS as a cloud service provider.

2.2 Research Niche

Table 1: Summary of Literature Review

Related Work(s)	Strength	Limitations
(Ivanov & Smolander, 2018)	Described the DevOps, CI and CD	The limitations of DevOps were not highlighted in the paper
(Arachchi & Perera, 2018)	Elaborated on CI/CD approach and methodology to adopt	Lack of security integration

(Myrbakken & Colomo-Palacios, 2017)	The DevOps implementation process was outlined	The security challenges are not highlighted
(Myrbakken & Colomo-Palacios, 2017)	DevOps challenges and solutions are discussed in this paper. The challenges to implementing the security have been evaluated	Lack of evaluation methodology
(Mao, et al., 2020)	Secret management and Configuration Management was considered for the implementation in the security phase	Security tools are not integrated into the DevSecOps Pipeline to identify misconfigurations and other vulnerabilities
(Chernyshev, et al., 2021)	Utilized security tools such as Snyk code analyzer, and DAST and followed OWASP ASVS to identify potential vulnerabilities in Cloud application	This paper has not elaborated on continuous security and the novelty of this paper was not considered
(Dhaya Sindhu, 2021)	In this paper, the importance of detecting security misconfigurations is highlighted. Recommendations on the DevSecOps model were provided	The limitation of this research lacked the integration of security tools to detect security issues
(Mohan, et al., 2018)	Security concerns and recommendations in the CD process were mentioned	Lack of implementation
(Battina & Sindhu, 2017)	Security policies were enforced to identify and mitigate the vulnerabilities in DevOps. The best practices for IAM were highlighted	However, the research paper does not consist of a wider scope to carry out the research and in addition, future work was missing
(Ahmed & Francis, 2019)	DevSecOps model was adopted to avoid insecure code and misconfigurations. Snyk dependency check was utilized	The list of misconfigurations was not highlighted in the paper and its evaluation.
(Torkura, et al., 2021)	Secure baseline configuration with regards to CIS benchmark was proposed	Lack of detection rules implementation to discover misconfigurations
(Casola, et al., 2020)	Security assessment and risk analysis using the STRIDE model were conducted	The detection of misconfigurations using mentioned security tools was not performed
(Rangnau, et al., 2020)	Integration of security tools in CI/CD pipeline and detection of vulnerabilities using tools was proposed	However, the detection of misconfigurations was not performed
(Ullah, et al., 2017)	Evaluation between a secure and non-secure Continuous Deployment process was analyzed. The security tactics	Misconfigurations were not defined in this paper

	were enforced to secure a CD process.	
(Sojan, et al., 2021)	Highlighted the security monitory solution to identify the security issues	Only at the theoretical level

3 Research Methodology

Preliminary research on the implementation of DevOps has emphasized the importance of incorporating security into the early stages of software development (Ahmed & Francis, 2019). DevSecOps practices can help prevent insecure coding and misconfigurations, which can lead to zero-day attacks, data breaches, and loss of confidentiality, integrity, and availability. It is therefore crucial to incorporate DevSecOps in ongoing deployments to mitigate these risks. Then, the importance of detecting misconfigurations and errors in Infrastructure as code resulted in vulnerabilities and expanded attack surface for the cloud application mentioned by (Alonso, et al., 2023). The component dependency check must be verified using SCA tools and SAST tools and an active knowledge database.
Therefore, the search string used for the research was "DevSecOps" and "Misconfiguration in Infrastructure as code". The research was carried out based on the related works by multiple authors (Battina & Sindhu, 2017), (Casola, et al., 2020), (Kumar & Goyal, 2021).

To avoid misconfiguration and vulnerabilities in the infrastructure as code, AWS Services, or CI/CD Pipeline, the Secure CI/CD pipeline was adopted and illustrated misconfigurations caused while deploying the application automatically. Therefore, a secure CI/CD pipeline has been proposed and the following methodology was implemented:

Stages of a CI/CD Pipeline Security

Continuous Integration: Terraform code was applied to create or modify the cloud environment and GitHub Actions was used to set up the CI workflow and merged the code changes into a central repository.

Continuous Security: A continuous security workflow using GitHub Actions was implemented to ensure the security of a codebase. The GitHub Actions workflow was adopted by authors (Benedetti, et al., 2022) and (Kinsman, et al., 2021) and discussed the importance of using GitHub Actions as the latest technology introduced by GitHub to automate the workflows. The workflows are specified in .yml or .yaml file extensions. (Kinsman, et al., 2021) compared the commits of each workflow file from other studies and analyzed the data from 3,910 GitHub repositories. When a specified trigger event occurred, such as the pushing of code to a particular branch or the creation of a pull request, a series of actions are automatically taken. These actions included the use of static analysis tools to detect vulnerabilities in the code and the running of tools to check for misconfigurations. The continuous security workflow was configured to run these actions automatically and alerted notifications of the triggered actions or failed actions. In this way, the implementation of a

continuous security workflow using GitHub Actions helped to secure the codebase and identify and fix issues early in the development process.
Code analysis: Static code analysis tools were used to scan the codebase for potential vulnerabilities. This included checking for insecure coding practices, such as the use of hardcoded passwords or the inclusion of sensitive data in the codebase. The tools utilized are Semgrep and tfsec.

Continuous Delivery: In this stage, the code is deployed to a staging environment where it can be tested in a production-like environment. This involved additional security checks, such as AWS Security Hub, and Cloud Inspector, and utilized various tools such as Prowler and ScouteSuite and AWS-service-enum script for identifying misconfigurations in AWS services and infrastructure.
In this phase, the terraform that is infrastructure as a code tool was utilized and integrated with GitHub Actions.

Logging and Monitoring: In this stage, the development team monitors the code in production and performs regular maintenance and updates to address any security vulnerabilities or misconfigurations that may be discovered. AWS CloudTrail was used as a monitoring solution provided by AWS. The use of CloudTrail for monitoring assisted in the identification of misconfigurations or potential security vulnerabilities in the AWS environment and facilitate timely action to address them.

4 Design Specification

The below architecture was proposed as a secure module of the CI/CD pipeline that was utilized to automate the application deployment process.

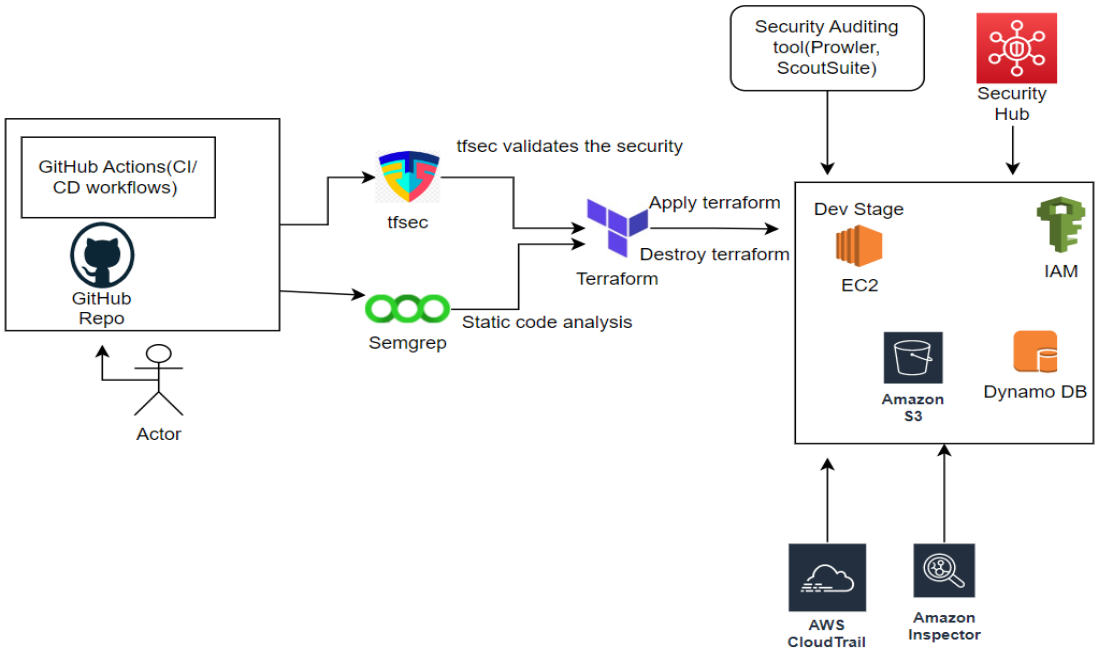

Figure 1: Secure Design

Semgrep[1] is a static analysis tool that uses pattern matching to identify vulnerabilities and security issues in code. It supports a wide range of programming languages, including Python, JavaScript, Java, and C++. Semgrep uses a custom language called Semgrep Patterns to define the patterns it should look for in the codebase. These patterns can be customized to match the specific security concerns of the project. The choice of the tool was selected based on the analysis performed by the Semgrep authors, supported by the OWASP community, and pinpointed by other great security forums mentioned in the article[1.]

Tfsec is a static analysis tool specifically designed for Terraform code. It was used for scanning Terraform configurations for potential security issues, such as the use of insecure resources or misconfigured resource settings. tfsec is written in Go and integrated into the GitHub action workflows. This was used in the previous research by (Ibrahim, et al., 2022) to secure the terraform configurations.

Terraform was chosen based on the comparison of the tools by (Ibrahim, et al., 2022) as shown in Figure 2. Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. It uses configuration files in HashiCorp Configuration Language (HCL) or JSON to describe the desired state of the infrastructure resources, and then it automatically creates, updates, and deletes those resources in a cloud provider or on-premises.

The AWS resources were deployed and managed by Terraform including the following AWS resources :

- **EC2 Instance**: Amazon EC2 (Elastic Compute Cloud)[2] is a cloud computing service that provides resizable compute capacity in the form of virtual servers (called "instances") that can be easily launched and terminated as needed.
- **Networking resources such as VPCs, subnets, and security groups**: Amazon Virtual Private Cloud (VPC) is a service that allows users to create a logically isolated section of the AWS cloud to launch resources in a virtual network that they define. A subnet is a range of IP addresses within a larger network. A security group is a virtual firewall that controls inbound and outbound traffic to resources in a virtual private cloud (VPC).
- **Storage resources such as S3 buckets and volumes**: Amazon Simple Storage Service (S3)[2] is a scalable, durable, and secure object storage service that allows users to store and retrieve any amount of data from anywhere on the web, organized into logical containers called "buckets."
- **Database instances such as DynamoDB**: Amazon DynamoDB[2] is a fully managed, NoSQL database service that provides fast and predictable performance with seamless scalability.
- **IAM**: Amazon Identity and Access Management (IAM)[2] is a web service that helps to secure control access to AWS resources. IAM enables users to create and

[1]https://portswigger.net/daily-swig/semgrep-static-code-analysis-tool-helps-eliminate-entire-classes-of-vulnerabilities

manage AWS users, roles, and groups, and use permissions to allow and deny their access to AWS resources.

factor	Tools		
	Terraform	*CloudFormation*	*Azure Resource manager*
providers	All Providers	AWS	AZURE
Language	DSL	YAML	YAML
Appraoch	Declarative and imperative	imperative	imperative

Figure 2: Tools Comparison

AWS Security Hub is a central place to manage security and compliance across AWS accounts and workloads provided on AWS documentation[2]. It provides a comprehensive view of security posture and enables automation of the process of finding and fixing security issues.

AWS Inspector[2] is a security assessment service that helps to identify vulnerabilities and deviations from best practices. Inspector performs automated security assessments of applications, networks, and Amazon Machine Images (AMIs), and provides recommendations for improving their security.

5 Implementation

The results obtained using the methodology and proposed design identified misconfigurations and addressed them in the early development cycle. The CI/CD Pipeline with continuous security was implemented using GitHub Actions workflows. The Terraform integrated with GitHub Actions enabled the automation of the process of Infrastructure as code (IAC) deployment with changes. The changes to Terraform configuration files (main. tf) were committed to a Git repository hosted on GitHub. A GitHub Action was triggered by the commit that includes a Terraform plan step to preview the proposed changes to the infrastructure. The Action includes Terraform apply step to apply the changes for the deployment. To add continuous security in CI/CD process, the tfsec GitHub Action(tfsec.yml) was configured to the su-muskan/AWSGoat repository to scan the terraform code for security issues or misconfigurations. This ensured the Terraform code is checked for security issues before it is deployed to production, helping to reduce the risk of vulnerabilities in the infrastructure. Further, Semgrep was configured as a SAST tool using GitHub Actions to identify the security misconfigurations in the code that included Insecure IAM Permissions, and an S3 bucket open to public access. After the deployment, AWS Security Hub and Inspector are utilized to monitor the security of your environment and identify potential security risks.
In addition, Prowler, and ScoutSuite tools were used to perform security assessments of AWS accounts.

[2] https://docs.aws.amazon.com/

It checks for security best practices and potential vulnerabilities and provides recommendations for improvement. This will further help in securing AWS environments and reduces the risk to CI/CD pipeline and compliance with industry standards and best practices. CloudTrail as a monitoring solution was also combined in this DevSecOps pipeline to identify any unusual or suspicious activity and take action to mitigate potential security risks.

By running static code analysis with tools like semgrep and tfsec, it was possible to detect a wide range of vulnerabilities in the codebase. Some examples of misconfigurations that these tools detected include:

- **Misconfigured resources**, such as insecure network settings or S3 buckets public access, or misconfigured security groups
- **Insecure coding practices**, such as the use of hardcoded passwords or the inclusion of sensitive data in the codebase
- **Unsafe handling of user input**, such as insufficient input validation or the use of insecure functions
- **Insecure IAM policy permissions**

By detecting these misconfigurations vulnerabilities early in the development process, it was possible to improve the security and stability of the software and reduce the risk of security breaches.

Security Hub detected 9 critical, and 11 High misconfigurations on AWS S3 buckets, IAM Policy, security groups, and MFA was not enabled for the root account. In addition, Prowler and Scout Suite tools detected the major misconfigurations on VPC, EC2, IAM, and S3.

6 Evaluation

This experiment aimed to detect misconfigurations in the Infrastructure as code as a part of the CI/CD Pipeline. The secure module of the CI/CD Pipeline was built by integrating security tools to identify security misconfigurations in every stage of the CI/CD pipeline. The evaluation was performed between the CI/CD Pipeline and the Secure CI/CD Pipeline. The pipeline without security does not detect any misconfigurations or vulnerabilities. However, this proposed pipeline with security detected the security misconfigurations or vulnerabilities that benefit compliance and security. The output produced from the security tools and GitHub Actions workflows showed the security misconfigurations on AWS resources which are EC2, S3, IAM, security groups, VPC, etc.

6.1 tfsec

The misconfigurations detected in the repository using tfsec were 155 as shown in Figure 3. The majority of the misconfigurations were regarding S3 bucket unauthorized access, IAM policy subjected to excessive permissions, and code security issues.

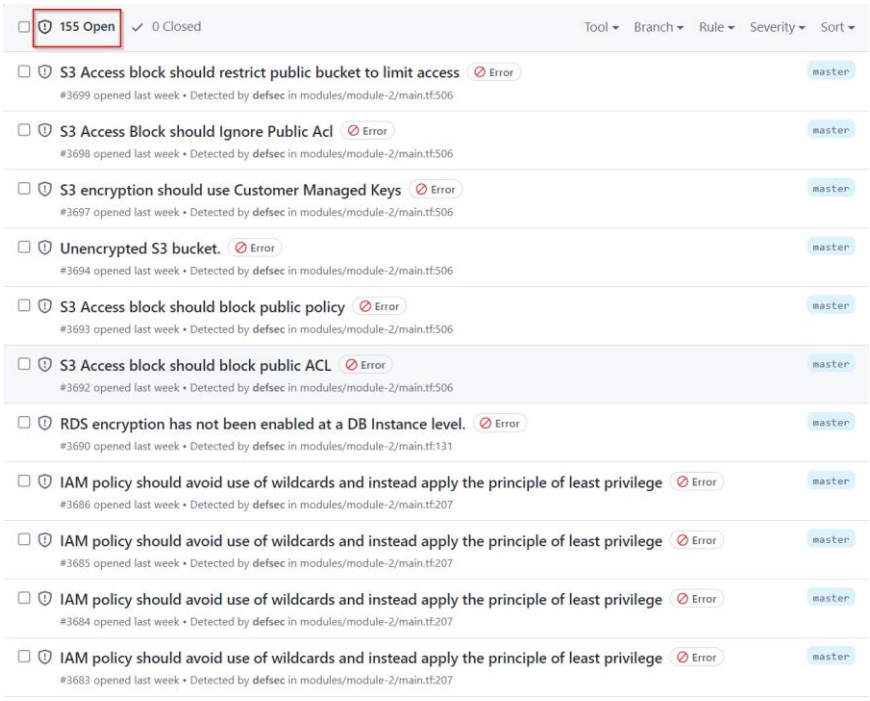

Figure 3: tfsec scan result

6.2 Semgrep

Below are the identified security misconfigurations from the Semgrep tool that checks for any secrets leaked in plaintext or other misconfigurations. Figure 4 represents the SQL errors, S3 access misconfigured, input handling error, and misconfigured IAM policy.

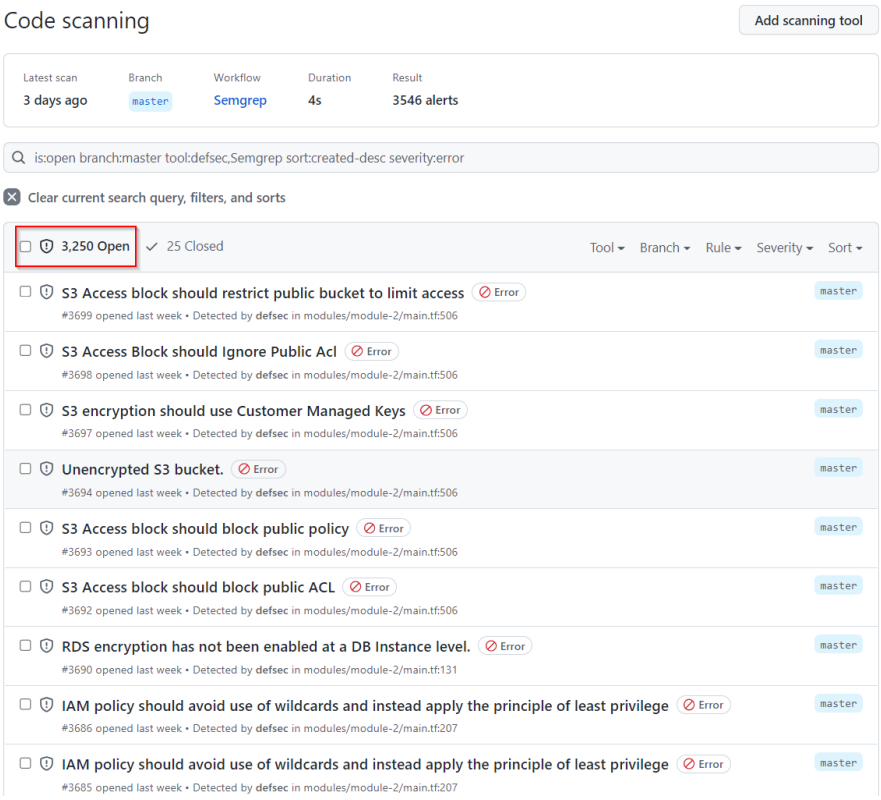

Figure 4: Semgrep scan result

6.3 Security Hub

Security Hub scan found misconfigured AWS services and resources following the CIS and AWS security best practices as shown in Figure 5. The majority of the misconfigurations were related to AWS IAM, EC2, S3, and Dynamo DB.

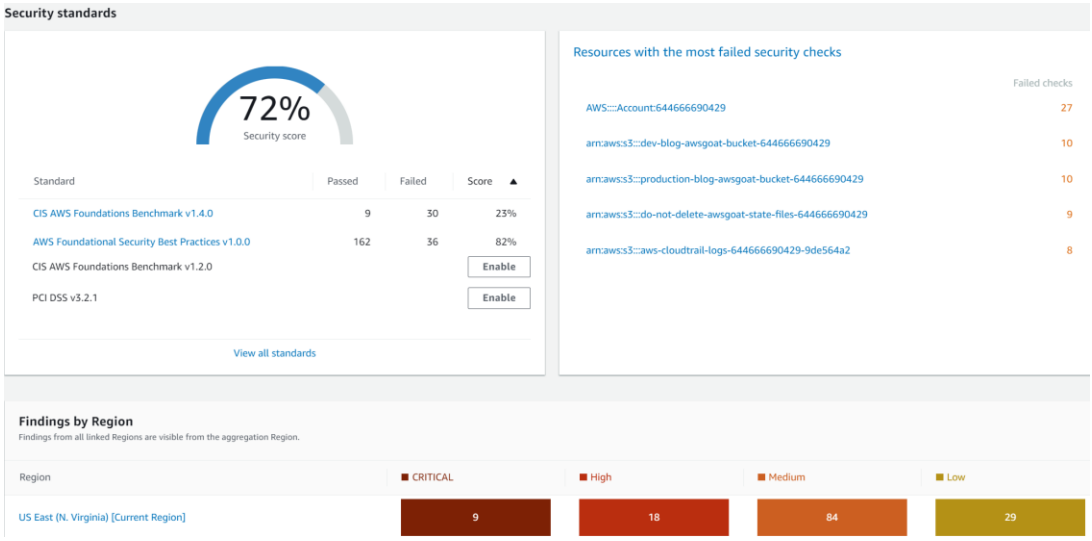

Figure 5: Security Hub scan result

6.4 Prowler

The below results from Prowler after the deployment. Figure 6 displayed the output from the Prowler security tool that has identified security configuration findings.

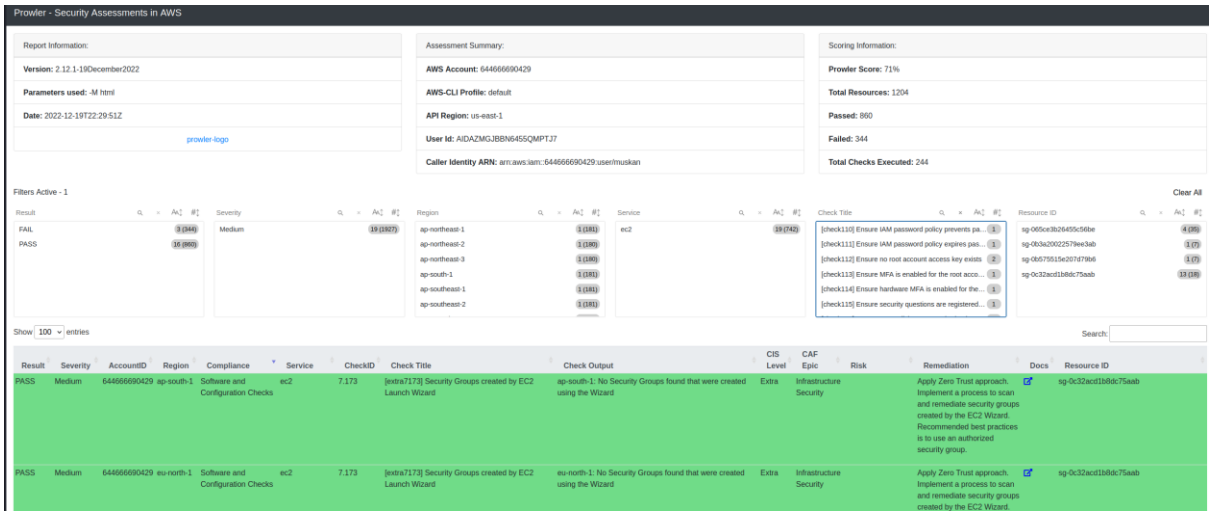

Figure 6: Prowler scan result

Using the above-mentioned tools, it was successfully identified security misconfigurations in AWS Infrastructure as code and this could help the developers or cloud architects to remediate the security misconfigurations in the initial stage before deployment which minimizes the risk from vulnerabilities, exploitation, and exfiltration of secrets. However, false positives were also detected using these tools that can be manually validated. Therefore, revisited 10 identified misconfigurations to verify the false positives using AWS Command Line Interface(CLI).

6.5 Manual Validation of the Findings

The list of security misconfigurations that are analyzed manually is shown in Table 2.

Table 2: Manual Analysis of security Misconfigurations

Misconfigurations	Description	Severity Rating	Mitigations	Affected Resources
S3 buckets allow public read access	Anonymous users will be able to list the objects in an Amazon S3 bucket that gives everyone online access to READ (LIST) and utilize the knowledge gathered to discover possible objects with misconfigured permissions and exploit them.	High	Deny PUBLIC READ access to Amazon S3 buckets by default	S3 bucket: production-blog-awsgoat-bucket-644666690429
IAM: Password Policy not enabled	No rules on the IAM Password policy can leave the account vulnerable to password-related security issues, such as weak or easily guessable passwords, or passwords that are reused across multiple accounts.	High	Configure Password complexity and other minimum requirements that are password reuse, expiry	IAM users, roles and policies
Root Account without MFA	Enabling multi-factor authentication (MFA) for the root account of the Amazon Web Services (AWS) account is an important security best practice because it adds an extra layer of protection to the AWS account. No MFA leads to unauthorized access	High	Enable MFA for root account	root_account
IAM Policy with wildcards	The use of '*' in the IAM Policy allows access to all the resources which might result in potential privilege escalation vulnerabilities	High	Apply the Least of Privilege Principle or enable Zero Trust	aws-goat-instance-boundary-policy

Security Group Opens All Ports to All	It was detected that all ports in the security group are open, and any source IP address could send traffic to these ports, which creates a wider attack surface for resources assigned to it.	High	Open ports should be reduced to the minimum needed to correctly operate and, when possible, source address restrictions should be implemented .	AWS_GOAT_sg
Security Group Opens SSH Port to All	Every port in the security group is open, allowing traffic to come from any source IP address.	High	Restrict Open ports and enforce source address restrictions	AWS_GOAT_sg
AWS EC2 AMI not Encrypted	Amazon Machine Images (AMIs) are encrypted to fulfill compliance requirements for data-at-rest encryption. When dealing with production data that is crucial for businesses, it is required to implement data encryption to protect it from attackers or unauthorized personnel.	Medium	Encrypt any unencrypted Amazon Machine Images available within your AWS account.	amzn2-ami-hvm-2.0.20221210.1-x86_64-ebs
AWS S3 bucket allows HTTP communication	Communications between the client and the S3 bucket are unencrypted which could reveal sensitive information in cleartext	Medium	Use of HTTPS	do-not-delete-awsgoat-state-files-644666690429
Unencrypted S3 bucket	Enforcing Server-Side Encryption will guarantee that your AWS S3 buckets are	Medium	Enable Server side encryption to S3 buckets	dev-blog-awsgoat-bucket-644666690429

	safeguarding their sensitive data while it is at rest.			
Private key detected	The sensitive credential is hardcoded	Medium	Remove the hardcoded .pem keys	/.ssh/keys/john.pem, /.ssh/keys/mary.pem, /.ssh/keys/charles.pem

6.6 Discussion

Figure 7 represented the results of an experiment that evaluated the effectiveness of tools, tfsec and Semgrep, in identifying misconfigurations in a pipeline integrated with security. The results of the experiment showed that both tools identified a relatively high number of misconfigurations. This was significant because these tools were used before deployment, which that stated any misconfigurations identified could be fixed in the early development cycle. However, it was also important to consider how to maintain the security of the pipeline after it had been deployed. This was achieved by using tools such as prowler and AWS Security Hub to continuously monitor the pipeline and identify potential security issues. By doing so, it was possible to quickly address any issues that were encountered and maintain the overall security of the pipeline over time. In summary, a combination of proactive measures in the development phase, such as the use of tfsec and Semgrep, and continuous monitoring after deployment using tools like prowler and Security Hub, could help ensure the security of the pipeline and reduce the risk of attacks.

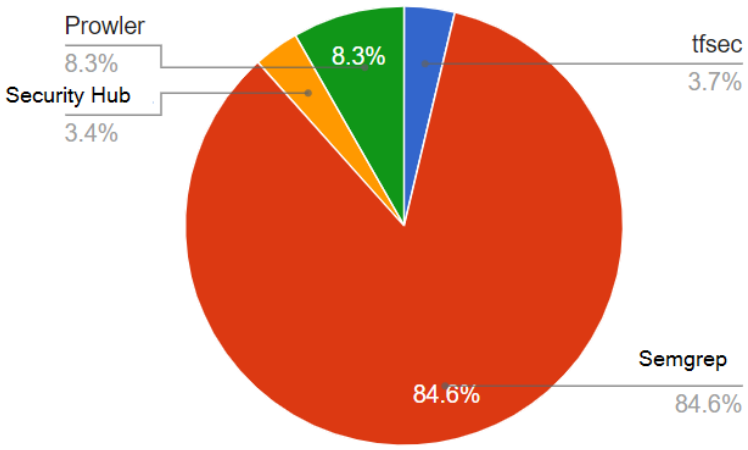

Figure 7: Quantitative analysis of automatic detection of misconfigurations

7 Conclusion and Future Work

The first Research Question (RQ1) was successfully achieved using the security tools and discovered misconfigurations. Further, RQ2 was also successfully implemented by integrating security tools in every stage of the CI/CD Pipeline and detecting misconfigurations before and post-deployment. The tfsec and Semgrep GitHub Action workflow was integrated and successfully detected and identified security misconfigurations before the deployment. The other tools such as Prowler, Scout Suite, and Security Hub detected the security issues post-deployment and minimized the attack surface for the CI/CD pipeline before deploying into the production environment. This ensured continuous security in the CI/CD pipeline and helps the cloud architects and developers to implement this proposed secure module of the pipeline to prevent security breaches or loss of data.

The limitation of this research was in the implementation of the 4 -stage DevOps Pipeline that included Jenkins integration, Code Build, CodePipeline, and Code Deploy. The challenge was to integrate the terraform with this 4-stage DevOps pipeline during the provided timeframe. Therefore, another approach was utilized to implement CI/CD pipeline using GitHub Actions.

One potential area of future work is to develop and implement a mitigation solution that can automatically mitigate vulnerabilities based on industry guidelines such as the Center for Internet Security (CIS) and the Open Web Application Security Project (OWASP). This solution could potentially include features such as automated remediation of vulnerabilities and real-time monitoring and alerting to identify and respond to potential threats or incidents. By integrating this solution into the CI/CD process, organizations can further improve the security of their systems and applications and reduce the risk of data breaches or other security incidents.

References

Ahmed, Z. & Francis, S. C., 2019. Integrating Security with DevSecOps: Techniques and Challenges. *2019 International Conference on Digitization (ICD),* pp. 178-182.

Alonso, J., Piliszek, R. & Cankar, M., 2023. Embracing IaC Through the DevSecOps Philosophy: Concepts, Challenges, and a Reference Framework. *IEEE Software,* pp. 56-62.

Arachchi, S. & Perera, I., 2018. Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. *2018 Moratuwa Engineering Research Conference (MERCon),* Issue 10.1109/MERCon.2018.8421965, pp. 156-161.

Battina & Sindhu, D., 2017. BEST PRACTICES FOR ENSURING SECURITY IN DEVOPS: A CASE STUDY APPROACH. *International Journal of Innovations in Engineering Research and Technology,* pp. 38--45.

Benedetti, G., Verderame, L. & Merlo, A., 2022. Automatic Security Assessment of GitHub Actions Workflows. Issue 10.48550/arXiv.2208.03837.

Casola, V., De Benedictis, A., Rak, M. & Villano, U., 2020. A novel Security-by-Design methodology: Modeling and assessing security by SLAs with a quantitative approach. *Journal of Systems and Software,* Volume 163, pp. 110-537.

Chaudhary, A. et al., 2021. Cloud DevOps CI -CD Pipeline.

Chernyshev, M., Baig, Z. & Zeadally, S., 2021. Cloud-Native Application Security: Risks, Opportunities, and Challenges in Securing the Evolving Attack Surface. *Computer,* pp. 47-57.

Dhaya Sindhu, B., 2021. The Challenges and Mitigation Strategies of Using DevOps during Software Development. *International Journal of Creative Research Thoughts (IJCRT),* pp. 4760-4765.

Ibrahim, A., Yousef, A. H. & Medhat, W., 2022. DevSecOps: A Security Model for Infrastructure as Code Over the Cloud. *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC),* Issue 10.1109/MIUCC55081.2022.9781709, pp. 284-288.

Ivanov, V. & Smolander, K., 2018. Implementation of a DevOps pipeline for serverless applications. In: *International conference on product-focused software process improvement.* s.l.:Springer, pp. 48-64.

Kinsman, T., Wessel, M., Gerosa, M. A. & Treude, C., 2021. How Do Software Developers Use GitHub Actions to Automate Their Workflows?. *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR).*

Kumar, R. & Goyal, R., 2021. When Security Meets Velocity: Modeling Continuous Security for Cloud Applications using DevSecOps. *Springer,* pp. 415--432.

Mao, R. et al., 2020. Preliminary Findings about DevSecOps from Grey Literature. *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS),* pp. 450-457.

Mohan, V., ben Othmane, L. & Kres, A., 2018. BP: Security Concerns and Best Practices for Automation of Software Deployment Processes: An Industrial Case Study. *2018 IEEE Cybersecurity Development (SecDev),* Issue 10.1109/SecDev.2018.00011, pp. 21-28.

Myrbakken, H. & Colomo-Palacios, R., 2017. DevSecOps: A Multivocal Literature Review. *Springer International Publishing,* pp. 17-29.

Rangnau, T., Buijtenen, R. v., Fransen, F. & Turkmen, F., 2020. Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines. In: *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC).* s.l.:s.n., pp. 145-154.

Sojan, A., Rajan, R. & Kuvaja, P., 2021. Monitoring solution for cloud-native DevSecOps. *2021 IEEE 6th International Conference on Smart Cloud (SmartCloud),* Issue 10.1109/SmartCloud52277.2021.00029, pp. 125-131.

Torkura, K. A., Sukmana, M. I., Cheng, F. & Meinel, C., 2021. Continuous auditing and threat detection in multi-cloud infrastructure. *Computers \& Security,* pp. 102-124.

Ullah, F. et al., 2017. Security Support in Continuous Deployment Pipeline. *Proceedings of 12th International Conference on Evaluation of Novel Approaches to Software Engineering.*