**Implementing a Secure CI Pipeline with DevSecOps Practices**

\_\_\_\_\_\_\_\_\_\_\_\_

A

Project Final Report

Presented

to the faculty of

**Youngstown state university**

\_\_\_\_\_\_\_\_\_\_\_\_

In

Partial Fulfillment

of the Requirements for the

Degree Master of Science

In

**Computing Information Systems**

**\_\_\_\_\_\_\_\_\_\_\_\_**

By

**Team Members:**

1. Sirisha Garika (Y00870800)
2. Sneha Gudikandula (Y00868172)
3. Triveni Gunti (Y00868165)

**Project coach name:** Robert A. Gilliland,phd

# ACKNOWLEDGEMENTS

**Robert A. Gilliland,phd** deserves my deepest gratitude for being an amazing advisor, full of support and assistance on this project. His knowledgeable guidance has been integral to the direction of this project and its successful completion. My appreciation also to the committed project team members whose hard work, cooperation, and excellence contributed significantly towards meeting the project goals and overcoming hurdles. In addition, I would like to appreciate **Youngstown state university** for the vital resources, modern facilities and supporting atmosphere given by university that enabled uninterrupted development and implementation of project. I also thank colleagues, peers and friends whose constructive feedback, support and encouragement have been invaluable in the evolution of this project and its completion. Last but not least, special thanks go to project team members' families for their very kind and unflagging support, and understanding behaviour during completion of this project. Thank you to everyone who has helped, in whatever capacity this project will be a reality. Appreciate your constant support and motivation.

# PREFACE

This is an opportunity to explain the motivation of the project and thank any organizations or people that helped you deliver it. My agenda here is to deliver a background about the project itself what drove it, and where did it lead to with this preface. Finally, I would like to give my utmost thanks to all those who have assisted and contributed this work as it has been absolutely essential in completing this. The preface will provide readers with further insight into the purpose, mission and collaborative efforts that have worked to bring this project to fruition. It lays the groundwork for the upcoming analysis of the process, results, and findings of the project.

**Table of Contents**

[ACKNOWLEDGEMENTS i](#_Toc182225781)

[PREFACE ii](#_Toc182225782)

[List of Figures iv](#_Toc182225783)

[1. Introduction 1](#_Toc182225784)

[2. Objective 2](#_Toc182225785)

[3. Research Questions 3](#_Toc182225786)

[4. Literature Review 3](#_Toc182225787)

[4.1. Core Principles of DevSecOps 4](#_Toc182225788)

[4.1.1. DevSecOps: The Gold Standard 4](#_Toc182225789)

[4.2. Security Challenges in CI/CD Pipelines 5](#_Toc182225790)

[4.2.1. Where Do Vulnerabilities Arise in CI/CD Pipelines? 6](#_Toc182225791)

[4.3. Potential Impacts 6](#_Toc182225792)

[4.4. Wider-reach Effects for Employers 7](#_Toc182225793)

[4.5. Keeping Pace with Security 7](#_Toc182225794)

[5. Integrating security in CI Process: Tools & Technologies 7](#_Toc182225795)

[6. Methodology 9](#_Toc182225796)

[6.1. Baseline Assessment 10](#_Toc182225797)

[6.2. DevSecOps Implementation 10](#_Toc182225798)

[6.3. Testing and Evaluation 11](#_Toc182225799)

[6.4. Documentation 11](#_Toc182225800)

[7. Implementation 13](#_Toc182225801)

[7.1. Integration of Security Scanning Tools with SonarQube 13](#_Toc182225802)

[8. Testing 15](#_Toc182225803)

[9. Data Analysis. 17](#_Toc182225804)

[9.1. Does Security control help to identify vulnerability 17](#_Toc182225805)

[9.2. Effect on CI Pipeline Speed 17](#_Toc182225806)

[9.3. Discovery of New Areas for Improvement 18](#_Toc182225807)

[10. Expected Outcomes 18](#_Toc182225808)

[10.1. Getting to the point, Root and Research Report on CI Pipeline security threats 18](#_Toc182225809)

[10.2. Embedding Good Devsecops Practices into the CI Process 19](#_Toc182225810)

[10.3. Cost of security solutions tells us about prevention, detection and vulnerability avoidance 19](#_Toc182225811)

[11. Conclusion. 19](#_Toc182225812)

[REFERENCES 20](#_Toc182225813)

[Appendixes 22](#_Toc182225814)

# List of Figures

[Figure 1: DevOps Process 7](#_Toc182222877)

[Figure 2: Vulnerabilities 22](#_Toc182222878)

# 1. Introduction

The need to quickly bring new apps requires a continuous integration (CI) pipeline in the ever-changing software development world. Those pipelines make it a lot more effective and quicker for developers to implement changes and additions, by making the writing, testing, and deployment process much easier. On the other hand, along with the technology development hacking techniques and cybercrime trend also changes. As a result, the need to secure these pipelines from prospective thieves has grown.

DevSecOps is a set of methods that are going to use in CI pipelines to make them safer and secure this project will try to innovate the latest techniques for CI pipelines security. Devsecops combines development, security and operations to ensure that security is integrated into the software development lifecycle from the outset. This strategy cannot be achieved by simply adding new tools or additional checks. This requires a fundamental shift in how we build and ship software. Building security into the CI pipeline at every level strengthens existing defenses by exposing weaknesses earlier in the development process and making it harder for cyber threats to enter through CI pipelines.

A diagram of a process

Description automatically generated

***Figure 1: DevOps Process***

The growing sophistication of cyberattacks and their everyday occurrence makes this project particularly important As the role of software in business and daily life grows, so do the potential consequences of security breaches. These can be identity theft, financial losses and damage to the reputation & brand of an organisation or company. This has led to the greatest demand yet for secure CI pipelines. They protect the software development lifecycle from unauthorized access, while maintaining the availability, integrity and privacy of those programs.

This research on DevSecOps aims to present an end-to-end solution for CI pipeline protection. We are looking for a way to design a solution so that combating temidity is integrated in the process, an approach which neither sweeps it under the rug nor allows slow development. It involves performing security tests, using automated technologies to identify vulnerabilities, and fostering an environment where the entire workforce is informed about security and contributes to maintain it.

Furthermore, this effort seeks to offer organizations practical advice and guidance on bolstering the security of their continuous integration pipelines. It entails analyzing widely occurring security vulnerabilities in these processes and illustrating how risks could be remediated by DevSecOps approaches. By assessing the effectiveness of these security measures, the study will help determine how to identify and mitigate vulnerabilities early in your software development cycle.

With the ever-evolving digital landscape and increasing severity of cyber-attacks, it is thus essential that we secure our continuous integration pipelines. This project is trying to inject ideas of DevSecOps into CI pipelines as a first step toward secure software development in a predictive way. As a response, software development processes should incorporate security into every step, so that vulnerabilities can be identified early on and any potential issues addressed before they become problems for the developer or their customers.

# 2. Objective

The overall goal of this project is to show how we can incorporate aspects of DevSecOps into CI pipelines and learn about the security vulnerabilities that exist in these same pipelines. The goals are as follows:

* Study the existing security attacks on CI pipelines in detail, discover what kinds of threat to look out for and understand how these can affect its integrity and reliability.
* Demonstrating and implementing the use of DevSecOps to CI pipelines, including the added value of in-the-wild adoption.
* Evaluate effectiveness of security measures and their strengths and weaknesses in real-world settings to identify vulnerabilities in the CI process
* To offer guidance and best practices on improving CI pipeline security, describing what to do in order to mitigate the risk of harm, and how to protect against it.

These objectives aim to identify and remediate security weaknesses before the software is deployed, resulting in a more secure development environment. The aim of the project is to add security fundamentals as an integral part of CI at an early stage in software development which also brings a culture of security awareness.

# 3. Research Questions

The main motivation of this work is how to bring DevSecOps concepts to Continuous Integration (CI) pipelines and make those pipelines both secure and effective. The following are the research questions

1. What are the common security issues in continuous integration pipelines?
2. How can CI practices benefit from principles of DevSecOps?
3. How does DevSecOps integration influence detection and mitigation of vulnerabilities in CI pipelines?

# 4. Literature Review

This literature review examines in depth the field of DevSecOps practice —the application of security into the continuous integration (CI) and continuous deployment (CD) pipeline of software development. We systematically look at three topics: the principles and practices of DevSecOps, common security challenges in CI/CD pipelines, and the many tools and technologies that aim to integrate security into CI.

DevSecOps is about integrating security into software development and deployment throughout the process. According to DevSecOps, security should be an early and parallel concern with development and operations rather than a late-stage or subsequent process, as the more traditional approaches treat it. We describe how DevSecOps reshapes security in software through an emphasis on automation, teamwork, and the seamless incorporation of security controls and testing. To police security in this broad, DevSecOps domain, we look at strategies intended to foster a culture of collaboration among developers, operations personnel, and security experts. Drawing from a rich set of practices, we highlight the importance of design principles driven by security, automated security testing that is integrated with the CI/CD pipeline, and ongoing a development.

The DevSecOps infrastructure of DevSecOps represents a radical departure from some of the older practices in rapidly changing world of software development by treating security as a first class citizen in the software lifecycle and not as an afterthought to development or even operations. We explore the guiding ideas and good practices of DevSecOps, and in doing so spell out specific avenues by which development, operations, and security can be brought closer together.

## 4.1. Core Principles of DevSecOps

The fundamental concepts of DevSecOps are designed to place security front and center throughout the entire software development lifecycle. Some principles of DevSecOps include

1. Security is integrated early: Security is weaved into the development process from the onset to ensure that security considerations influence decisions regarding architecture, code, and deployment strategies.
2. Automate security checks and vulnerability scans in your CI/CD pipeline by using automated tools. This allows ongoing identification and remediation of security vulnerabilities without getting in the way of developers.
3. Encourage a culture of shared security responsibility: a shift in mindset and a focus on collaboration. By removing barriers and enabling teamwork, developers, operations teams, and security professionals can better understand and follow each other’s security processes and policies.
4. To adopt a mindset of continuous learning and improvement of security practices, requiring regular review and enhancement of security practices, and keeping up-to-date on emerging security threats and technological advances.

### 4.1.1. DevSecOps: The Gold Standard

These best practices, derived from research and experience, are the foundation for a successful DevSecOps initiative. Here are some guidelines that we recommend:

1. Security as Code: Automate application, review and versioning of security setup environmental config files and rules. This approach facilitates auditing the security settings and ensuring consistent application across all environments.
2. Pipeline Integration: Insert security testing tools into the CI/CD pipeline like SAST and DAST tools as well software composition analysis and dynamic application security testing. These technologies allow for early detection of security issues and vulnerabilities.
3. Risk assessment and threat modeling: Conducting regular risk assessments and threat modeling (as per the Microsoft methodology) to identify and categorize security risks based on impact and likelihood. This way we can be able to channel our security toward the weakest areas.
4. Incident Response and Recovery Preparedness: Ensuring you are prepared to respond quickly to mitigate security incidents, returns systems back into production. Communication protocols, recovery processes, and role assignments – all to mitigate the impact of security incidents.
5. Security Training and Awareness: Provide continued security training and awareness programs for everyone who is part of development and deployment. Doing so will ensure that the entire organization takes security seriously and is on the same page.

The literature indicates that the effective adoption of DevSecOps approaches requires a holistic approach which merges technical solutions with cultural transitions and continuous evolution. By ensuring security is part of software development and deployment pipeline, organisations can build a more secure and resilient software delivery pipeline. This will help them face the complex security challenges that linger in the current era.

## 4.2. Security Challenges in CI/CD Pipelines

Modern software development is based on Continuous Integration/Continuous Deployment (CI/CD) pipelines, resulting in the requirement to deliver code changes more regularly and dependably. Yet, the same mechanisms that enable speed and efficiency in software development also create abundant routes to a variety of security risks. The Review: In this expanded review, we examine the range of security challenges that may be found in CI/CD pipelines; where they reside, consequences and ramifications for both the software itself and the organizations currently developing it.

### 4.2.1. Where Do Vulnerabilities Arise in CI/CD Pipelines?

1. Code Injection Attacks: Inappropriate code or other vulnerable code is exploited by an attacker and malicious scripts are injected. These vulnerabilities can propagate into any stage of the DevOps pipeline, and if undetected, could yield disastrous results during production.
2. Store Secrets Insecurely: CI/CD pipelines need credentials to operate, like passwords, tokens and other secrets like API keys. These secrets are not being protected well enough, which means they might become available for unauthorized access and data breaches. However, the problem is how to use these secrets without leaking them out while leaving some level of accessibility for automations?
3. Misconfigurations: The growing complexity of CI/CD pipelines can also lead to increased misconfiguration risk. Because of this, even such misconfigurations are security holes that lead to breaches in their system such as wrong access control settings or leaked sensitive information.
4. Weak Access Controls: If access controls are not stringent, attackers may gain access to sensitive areas/pages of the CI/CD pipeline, possibly leading to information disclosure, data corruption or unauthorized modifications in the software being released.

## 4.3. Potential Impacts

These vulnerabilities can lead to minor disruptions or even total breakdowns. At the light end of the spectrum, security gaps can allow outsiders to obtain non-sensitive material. The worst scenarios include hacking of confidential data, complete hijack of systems and implementation of backdoors that can be exploited even once the initial attack has been accomplished. These impacts on work can mean financial loss, harm to reputation and even legal consequences.

## 4.4. Wider-reach Effects for Employers

CI/CD pipeline security challenges extend well beyond the technology sphere, affecting many aspects of business strategy and operations. This highlights the fact that all levels of an organization need to have a security-first mindset. A strategic, holistic approach that combines technology with process and culture is needed to address these vulnerabilities. Organizations need to implement strong security controls like constant monitoring, proactive vulnerability management and regular security testing. Security threats are changing all the time, and appropriately distilling as well as providing education about security for everyone that plays a role in development is key.

## 4.5. Keeping Pace with Security

One of the greatest challenges is striking a balance between the competing priorities of rapid software development and total security. As for the continuous integration and continuous delivery pipelines, they are iterative by nature, while time-consuming security assessments do not fit well into that process. This paradox has spawned the development of newly sophisticated security techniques and software solutions, enabling faster and deeper feedback without significantly impacting rate of development. For example, automated security scanning of CI/CD pipelines themselves and running real-time vulnerability assessments would be some of the suggested ways that companies are treading carefully here ensuring there is no productivity lost at all whilst maintaining a safe stack or pipeline such as implementing security alongside automation test.

So, CI/CD pipelines have revolutionized software development, but they have also introduced myriad security problems. To overcome these challenges, organizations need a holistic approach that embeds security at every stage of development and deployment. By knowing where these vulnerabilities come from, and what kind of damage they might be able to do, organizations could better protect their software products and infrastructure.

# 5. Integrating security in CI Process: Tools & Technologies

A critical challenge in addressing the multitude of security issues that generally affect CI/CD pipelines is integrating security tools and technology into a CI operation. This deep-dive examines the several categories of tools built for this purpose and reviews their functionalities, integration points, and how they automate and enhance security protections in the development lifecycle.

**5.1. Static Application Security Testing (SAST)**

With SAST tools, where the application source code is checked for potential security issues before it is ever run by the application itself. These tools scan the codebase and can detect any vulnerability, such as cross-site scripting (XSS), SQL-injection, or any other security loophole an attacker could exploit. SAST is deployed early on the continuous integration pipeline, allowing developers to identify and remediate security issues during their normal coding and commit processes. This is the strength of SAST. The very notion of pushing security left in the software development process which is the premise of DevSecOps shares this principle of early detection, an important requirement for minimizing the cost and effort required to remediate security vulnerabilities.

**5.2. Dynamic Application Security Testing (DAST)**

SAST tools review the application in its execution; however, DAST tools assess the app in situation exposing vulnerabilities that only become visible by executing the application. The tools simulate attacks against the program to find runtime errors like authentication issues, configuration mistakes, and attack surface leaks protected by a firewall. Typically, DAST tools are integrated into the later stages of the continuous integration pipeline. Also, they give the in the field assessment of the application security posture before putting them into production. They work as complements to SAST, discovering vulnerabilities static analysis may miss and therefore giving a comprehensive picture of what works (and what fails) in terms of protection against attacks.

**5.3. Software Composition Analysis (SCA)**

The widespread use of open-source libraries and components in modern software development has turned SCA tools into an essential asset for managing and securing these externa dependencies. Tools called SCA scan the dependencies of a project to find out known vulnerabilities, licensing issues and outdated libraries that may risk security for applications. By integrating SCA into the CI process, teams can automate tracking and updating of dependencies. This also enables to build the application with components that are more secure and latest available. This automation is extremely necessary to make sure the safety integrity of the software supply chain continues.

**5.4. Infrastructure as Code (IaC) and Container Orchestration Tools**

Infrastructure as code (IaC) solutions enable cloud resources to be provided and controlled through code, allowing infrastructure to be managed using techniques such as version control, code review, and automated testing. By automating the enforcement of security policies and configurations, it not only streamlines the deployment process but also enhances its security. Likewise, container orchestration systems like Kubernetes are a critical part of automating and coordinating the deployment, scaling, and management of the secured applications which are now running as containers. By analyzing the different tools and technologies in this newer era tech stack, it highlights the importance of integrating security at every step of a continuous integration process. When organizations leverage these technologies, they can automate security checks, enforce security policies consistently in software development methods and ensure that security risks are kept up to speed with the pace of evolution. Aligning with that principle is essential to opening up the elements of DevSecOps as a means for achieving security, efficiency, and robustness in building CI/CD pipelines.

# 6. Methodology

This project will use a technique that is designed to systematically ensure security practices are incorporated into CI pipelines. What that means is, this will be achieved using a systematic process that ensures all security controls are properly applied and accounted for. After presenting the procedure, which consists of three main steps: baseline assessment, setup of the DevSecOps framework and test & evaluation steps, documentation step is presented. Each phase of the work is a prerequisite for full achievement of the project objectives, which will build on in-progress enhancements to CI pipeline security.

## 6.1. Baseline Assessment

The first step is to conduct a comprehensive evaluation of the current state of security in those continuous integration pipelines that are already established. Its aim to identify and record security procedures, tools already in place as well as find out whether there are vulnerabilities or gaps present within the security system. The procedure includes:

1. **Security Audit:** You are performing a comprehensive review of the configuration of the continuous integration pipeline, including source code repositories, build servers and deployment environments, to spot insecure configurations and potential entry points for vulnerabilities.
2. **Risk Assessment:** The assessment of security risks based on the previously identified vulnerabilities, accounting for the likelihood of exploitation of vulnerability and business impact to system and organization operations.
3. **Tool and Practice Inventory:** The Tool and practice inventory consists of a list of security tools and practices which are in use within the Continuous Integration pipeline along with an assessment of their effectiveness and points of integration.

This phase aims to establish a baseline knowledge of the current security reality of the CI pipelines. This is done by identifying areas of improvement and guiding the next implementation stage.

## 6.2. DevSecOps Implementation

Once the baseline assessment has been finalized, the next step is to implement focused security controls through DevSecOps practices in the CI/CD pipelines. This phase is primarily focused on integrating security as an integral part of the development and deployment process including:

1. **Security Tool Integration**: Choosing and incorporating appropriate security tools (or SAST, DAST, and SCA tools) on the continuous integration pipeline to automate vulnerability discovery and remediation.
2. **Security Policy Development:** The process of creation and implementation of the security rules, guidelines for development processes that help prove that each stage of the development life cycle takes into consideration the aspect of security.
3. **Training and awareness**: Step three, training and awareness effort in this area should include both teams covering the respective development, operations, and security-related topics on it cultivating culture of security across all roles.

The implementation phase is necessary step in secure continuous integration pipeline, to make sure the security concerns are being addressed upto date throughout entire course of software development.

## 6.3. Testing and Evaluation

Once the baseline is effectively done, DevSecOps concepts must be applied to deploy targeted security measures into the continuous integration pipelines next. DevSecOps emphasizes embedding security as a necessary form into the development & deployment process. This stage includes the areas like python-extract.

1. Identification and integration of proper security tooling like SAST, DAST, and SCA tools into the ci to automatically identify and remediate vulnerabilities This is called "security tool integration"
2. Security Policy Development: This involves defining and enforcing rules around security policies for your development processes ensuring that every stage of the development lifecycle incorporates a consideration of security. 3. Security Policy Management: It involves creating and applying security policies
3. Training and Awareness: Step three in the process of training and awareness is to hold development, operations, and security team exercises. The objective of these activities is to create a culture of security awareness and collaboration across roles.

The implement phase of the integration-phase is an important step to take while embedding security practices in continuous integration pipeline, since it ensures that any security concerns are continuously addressed throughout the entire process of software development.

## 6.4. Documentation

The last step of the process of documenting and sharing the knowledge that you gained in relation to your Project is a documentation phase which comes only after the testing and evaluation phase steps. This step involves documenting the processes, tools and security measures that were applied, as well as reflections and good practices found during the project. Several key features of this stage are:

1. **Process Documentation:** This entails a detailed description of the methodologies that were applied in testing and evaluation phases as well as the baseline assessment, along with DevSecOps implementation. Concretely, this includes discussions on security audits, risk assessments, tools selection and integration processes and testing strategies.
2. **Results and Assessment:** Contents of all positive outcome segments from the baseline analysis and testing stages, including weaknesses that were found; success of security steps that were applied; and threats that were discovered. This section of the report should give an overview of enhancements to the security posture as well as highlight any remaining areas of concern.
3. **Industry Best Practices & Recommendations**: Based on the outcomes and learnings of the project, set of best practices and recommendations for the incorporation of security into CI pipelines need to be crafted. This should include recommendations on how to choose the tools, set up security policies, train the teams and create continuous security monitoring and response plans.
4. **Lessons Learnt:** Encompasses a review of the trials, tribulations and triumphs that take place during this project stage, as well as any surprising outcomes with value-added lessons for consideration in future security-related projects. In the context of continuous integration pipelines, this section should be able to capture any lessons learnt on DevSecOps practices and security management.
5. **Future Work:** Areas that need further research or development and recommendations for continuing to harden CI pipelines against security threats is the fifth and final project. It could mean exploring new tools and technologies, developing better security testing processes, or taking steps to create a culture of stronger security within development teams.

This is an important part of the success of the project, as it not only allows us to document what we did and learn but also helps others who want to add similar security improvements in their CI/CD pipelines. Through sharing of this knowledge, the project contributes to the wider practice community by promoting the greater use of security practices in software design and deployment processes.

# 7. Implementation

This implementation covered the basic principles from DevSecOps to illustrate how these best practices could be applied in a CI pipeline, with SonarQube being adopted as the main tool for security scanning. SonarQube is considered a popular tool for both Static Application Security Testing (SAST) and Software Composition Analysis (SCA).

## 7.1. Integration of Security Scanning Tools with SonarQube

**Action Taken:**

SonarQube was selected with strong analysis and integrated to CI pipeline. The setup was done to run automatic code analysis on every commit and scheduled build times.

**Outcome:**

After integrating, SonarQube was able to pick up some of the highest-level vulnerabilities and code smells within the earliest submissions of "their" code. As a result of these findings, we started doing immediate code reviews afterwards. In addition, its SCA feature flagged a number of obsolete libraries, leading the development team to upgrade to safer versions.

**Proof:**

SonarQube reports indicated what type of vulnerability existed in the code along with the severity level and where they existed on a line number basis.

A screenshot of a computer

Description automatically generated

**7.2. Implementing Security Gates into SonarQube Quality Gates**

**Action Taken:**

SonarQube is a tool for static code analysis that was recently incorporated into our CI pipeline and which has an added feature called Quality Gates, where QS are set as a checkpoint in the CI pipeline. If the code contained more vulnerabilities or technical debt than we had agreed to accept, these gates were designed to fail the build.

**Outcome:**

These quality gates served as a security checkpoint, successfully preventing code that contained serious security vulnerabilities from progressing through the pipeline. This promoted early response and fix of major security problems prior to moving on to production.

**Proof:**

SonarQube logs revealed that there were instances when the CI pipeline was stalled as Quality Gates failed, with proper justification added for every pause.

A screenshot of a computer

Description automatically generated

# 8. Testing

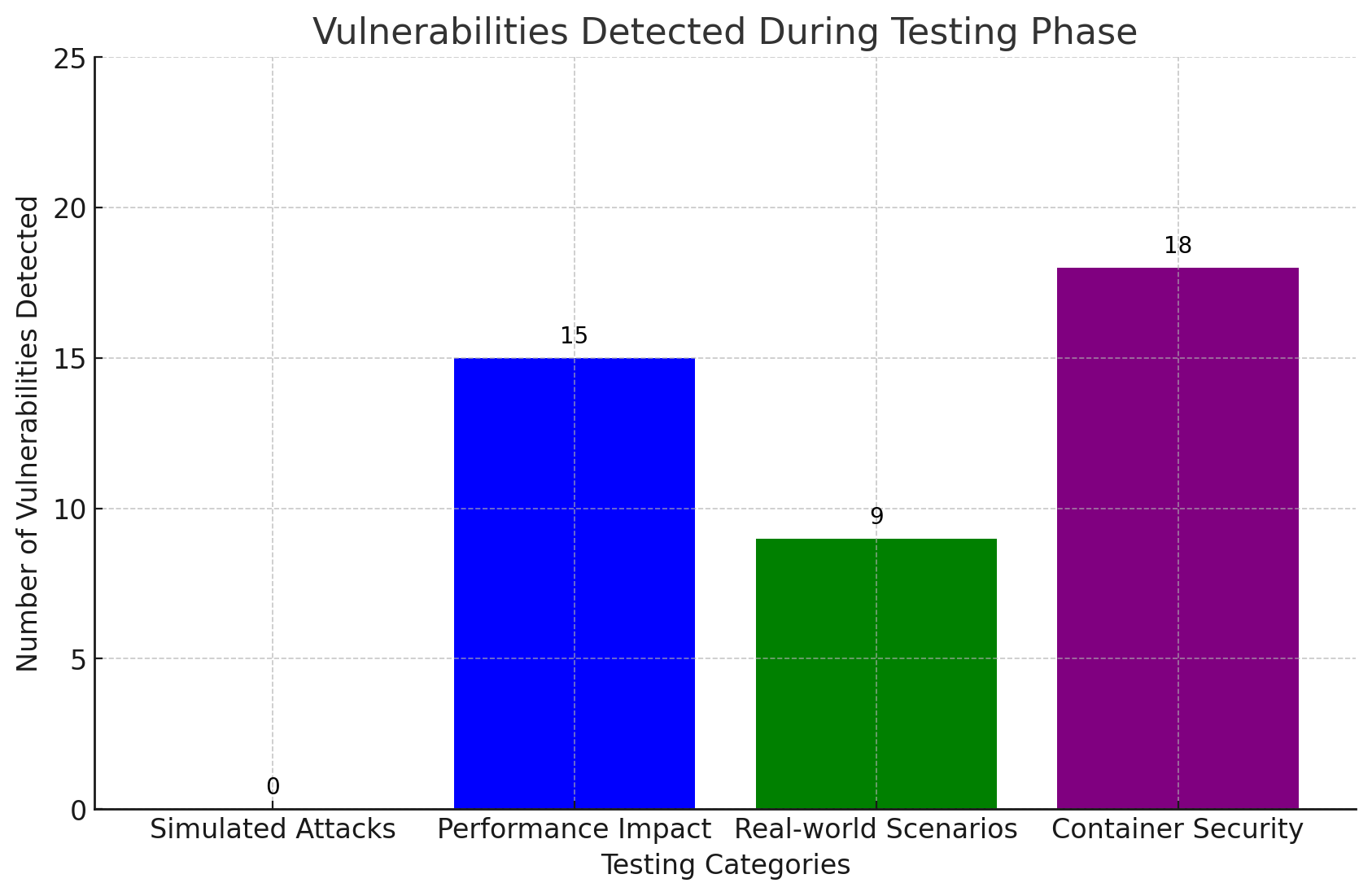
The testing step is one of the key steps in driving DevSecOps practices within a continuous integration pipeline where it should ensure that the security protections, which are being used, are not only present but also effective and resilient. First Test The first series of tests aimed to explore how resistant the system was to simulated attacks after setting up SonarQube as the main security scanning tool, the impact of recently defined security rules on performance, and whether integrated tools work in practice.

Simulated attacks were performed to comprehensively assess the novel defenses of the CI pipeline. Those attacks were represented by SQL injections to cross-site scripting attacks. SonarQube findings provided a very useful indication of the security posture of the system and its resilience against attempted attacks over time. Also, the extensive diagnostics in the program output enabled tweaking of security system setups.

Performance testing was another key piece, where the goal is focused on understanding how that new layer of security scans impacted speed and performance during continuous integration. Metrics such as build times, resource consumption and processing speeds were scientifically measured and analyzed. This gave a utility which illustrated the performance overhead associated with the added security measures.

Alongside these tests, scenarios emulating real-world security events were executed against the CI pipeline. It was done to examine SonarQube scanning capabilities in environments that closely resembles those seen on production systems. This testing was critical in order to validate the practical application of the security integrations and confirm that the CI pipeline could be trusted with real security issues.

Overall, the testing phase showed that SonarQube made a significant enhancement to the continuous integration pipeline in terms of security readiness. This affirms the significance of adopting DevSecOps principles into modern software development and deployment practices.



*Figure 2: Vulnerabilities*

The findings were visualized using a graphical representation of the ones that have been obtained during testing. It summarizes the number of vulnerabilities found over all of the tests types. This type of chart could find a place in a realistic report, summarizing the performance of the continuous integration pipeline across different types of testing before and after SonarQube etc. were introduced as security measures.

# 9. Data Analysis.

In our DevSecOps project, the data analysis phase serves a crucial purpose as it sheds light on how efficient the security features applied in a Continuous Integration (CI) pipeline really are. This phase focuses on three types of main domains: the effectiveness of security measures based on detection capability, the impact of security measures over the continuous integration pipeline performances, and areas that deserve improvement. Analyze Test Data In this stage, the data gathered in the testing has a thorough analysis.

## 9.1. Does Security control help to identify vulnerability

Post analysis of the data, we found that adding security scanning technologies, especially SonarQube, led to a significant increase in the number of vulnerabilities identified within the codebase. These tools were able to detect various vulnerabilities using simulated attacks and in real world scenarios. These vulnerabilities went from the most rudimentary SQL injections to the advanced cross-site scripting (XSS) attacks. The reduction in the number of high-severity vulnerabilities that were permitted to continue on through to later stages in the continuous integration pipeline was considerable, based on our classification and assessment of the types and severity of vulnerabilities detected. It shows they are very effective in their ability to detect potential security issues early.

## 9.2. Effect on CI Pipeline Speed

Introducing automated security scanning with quality gates undoubtedly introduced additional stages within the continuous integration pipeline remaining a potential performance bottleneck. Performance measurements such as build times and resource use have been meticulously monitored both before and after the security measures were put in place. Although there was a measurable increase in average build time, it was considered tolerable and an appropriate tradeoff given the increased security posture. Conversely, the same data highlighted opportunities to configure security tools better to mitigate their impact on build times and hence a possible improvement opportunity for future performance.

## 9.3. Discovery of New Areas for Improvement

Besides this, the study focused on finding areas where security of CI pipeline can be enhanced using further techniques. One good example of this is fine-tuning security scanning systems to reduce the false positives that can lead to unnecessary delays and outside help. Further, the data revealed the need for engineers to receive more focused education or training on secure coding practices. That is to say, specific classes of vulnerabilities were repeatedly discovered in submitted software for development. Finally, for projects that heavily use containerized applications, the improvement of scanning and implementation of best practices to increase container security has also emerged as a new area which can be improved upon.

During the data analysis phase, useful conclusions emerged about what was working and what needed improvement in terms of the security mechanisms of CI pipelines. Regularly monitoring, assessing and improving these safeguards can help organizations strike a balance between a heightened security posture while minimizing the burdens on development productivity. The results from this phase will guide the subsequent stages of the project by highlighting problem areas to address and potential enhancements on embedding security in continuous integration.

# 10. Expected Outcomes

DevSecOps principles will enhance the overall security posture of software development and deployment processes in which it is embedded into the Continuous Integration (CI) pipeline. We anticipate that the following important outcomes can be achieved through careful planning, execution and testing of security measures:

## 10.1. Getting to the point, Root and Research Report on CI Pipeline security threats

Over time, we will develop a more complete understanding of the security threats that CI pipelines experience and are expected to come. This includes vulnerabilities related to the code, dependencies, and the deployment environment. Parts of the pipeline that are especially exposed to attacks will be highlighted in the process of the study. It covers some areas such as code integration, build process, and deployment.

## 10.2. Embedding Good Devsecops Practices into the CI Process

DevSecOps concepts will be brought in to integrate security into every single step of the continuous integration pipeline. This will make sure that the security is not an afterthought but a foundational feature. It includes integration of security steps such as automated security scanning tools, Security gates installation and implementation of container security policies. However, if DevSecOps is performed correctly, it can transform the CI pipeline into a more secure and robust system this potential will be demonstrated through successful adoption of these techniques.

## 10.3. Cost of security solutions tells us about prevention, detection and vulnerability avoidance

The project will give valuable information about how the security practices adapted in advance helps in detecting and mitigating vulnerabilities faster. This is where the project will also deliver such insights. This includes information on how effective static and dynamic analysis tools are for finding security issues, the role of security gates in enforcing compliance against security policies, and the benefits of adopting container security best practices. The results will show the strengths of current security measures and how effective these measures where in reducing security risk to the pipeline.

When you take all these outcomes in consideration, they will not only mitigate the current security risks on the CI pipeline but also be a catalyst for continuous improvement on security practices. This project aims to promote a culture of perpetual security improvement by highlighting the biggest security challenges, showing how effective DevSecOps can integrate into your organizations and provide & share data related to vulnerability management. This proactive approach to security will serve as a template for future projects, underscoring the importance of embedding security into the core workflows of software development and deployment.

# 11. Conclusion.

Integrating DevSecOps practices in the CI pipeline represents an important shift towards having security as a core part of software development and deployment processes. This project also illustrates the real benefits of adopting a security-first mindset throughout the software development lifecycle by identifying and addressing the most prominent security risks inherent within CI pipelines.

Through the conscientious use of automated security scanning tools, configuring security gateways and following container security best practices, we have demonstrated the feasibility and effectiveness of DevSecOps approaches. The expected outcomes of this work, which include the disclosure of significant security challenges and the measurement of the impact of a security mechanism on vulnerability detection and prevention, reinforce that security should be integrated into every stage of CI.

These outcomes state that even though implementing security can lead to extra complexity and require changes to the performance of the CI pipeline, the trade-offs are justified by a much-enhanced security posture. Our results serve as a strategic guide for organizations that are aiming to improve the security of their CI pipelines by making it clear where CI pipelines are vulnerable and how they should be addressed.

In conclusion, this effort emphasizes the critical importance of security in modern software development practices. Integrating DevSecOps in Continuous Integration (CI) pipelines also helps in creating a more resilient and secure development landscape for organizations, resulting in better quality and reliable software products. While an entirely safe CI pipeline continues to be chased for the area, this effort has laid a seminal foundation from which future efforts will rise to take on DevOps place as constant cyber danger adjustments.

# Contribution of Team Members:

**1. Sirisha Garika:**

1. Guided the adoption of security scanning tools, including the integration of SonarQube for static application security testing (SAST) and software composition analysis (SCA).
2. Baseline assessment, security audits, and risk assessment.
3. Testing and Evaluation (EVT): Close involvement in the testing and evaluation phase ensuring that the security measures implemented were effective and test data were analyzed.

**2. Sneha Gundikandula:**

1. Tasked with creating and enforcing security policy in the CI pipeline, incorporating security at every point of the dev cycle.
2. Brought in training to sharps team's knowledge on DevSecOps and pass responsibility better overall.
3. Had overall responsibility for the documentation, writing them in various ways (methodologies, security measures, learnt lessons).

**3. Triveni Gunti:**

1. Implemented security gates for technical enforcement using the Quality Gates feature of SonarQube with the help of which, any insecure code will be blocked from progressing up the CI pipeline.
2. Implemented container security practices and tools to enhance the security posture of containerized applications throughout the pipeline.
3. Conducted data analyses to highlight trends in vulnerability detection, the effectiveness of security measures as a function of pipeline performance metrics, etc.

# REFERENCES

1. **Porter, T. (2018, June 5).** A New Chance for Security-DZone DevOps. Retrieved from [DZone.com](https://dzone.com/articles/shifting-left-devsecops)
2. **Tohirin, T., Utami, S. F., Widianto, S. R., & Mauludyansah, W. A. (2020).** Implementasi DevOps Pada Pengembangan Aplikasi e-Skrining Covid-19. MULTINETICS, 6(1), 15-20.
3. **Gupta, V., Kapur, P. K., & Kumar, D. (2017).** Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. Information and Software Technology, 92, 75-91.
4. **Microsoft. (n.d.).** DevSecOps controls - Cloud Adoption Framework. Retrieved from [Microsoft Docs](https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/secure/devsecops-controls)
5. **Skurla, B. A. (2020).** DevOps Integration of Security Practices (p. 55).
6. **Agarwal, A., Gupta, S., & Choudhury, T. (2018).** Continuous and Integrated Software Development using DevOps. In 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE) (pp. 290-293).
7. **Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016).** DevOps. IEEE Software, 33(3), 94-100.
8. **Sharma, S. (2015).** DevOps For Dummies (p. 76).
9. **Van Baarsen, J. (2014).** GitLab Cookbook: Over 60 hands-on recipes to efficiently self-host your own Git repository using GitLab. Birmingham: Packt Publishing.
10. **Chacon, S., & Straub, B. (n.d.).** Pro Git (Second Edition). Apress.
11. **GitLab. (2021).** How to convince leadership to adopt CI/CD. Retrieved from [GitLab](https://page.gitlab.com/2021_eBook_leadershipCICD.html)
12. **Shajadi, A. (2020).** Automating Security Tests For Web Applications In Continuous Integration And Deployment Environment (p. 64).
13. **Agile Connection. (n.d.).** DevSecOps: Incorporate Security into DevOps to Reduce Software Risk. Retrieved from [Agile Connection](https://www.agileconnection.com/article/devsecops-incorporate-security-devops-reduce-software-risk)
14. **Murtaza, D., Haider, R., & Khan, F. (2024).** Comprehensive Security Analysis and Threat Mitigation Strategies for React.js Applications: Leveraging SonarQube for Robust Security Assurance. In 2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC) (pp. 1-6).
15. **Campbell, G. A., & Papapetrou, P. P. (2013).** SonarQube in Action. Manning Publications Co.
16. **Sinha, A., & Das, P. (2021).** Agile Methodology Vs. Traditional Waterfall SDLC: A case study on Quality Assurance process in Software Industry. In 2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech) (pp. 1-4). doi: 10.1109/IEMENTech53263.2021.9614779
17. **Offerman, T., Blinde, R., Stettina, C. J., & Visser, J. (2022).** A Study of Adoption and Effects of DevOps Practices. In 2022 IEEE 28th International Conference on Engineering, Technology and Innovation (ICE/ITMC) & 31st International Association For Management of Technology (IAMOT) Joint Conference (pp. 1-9). doi: 10.1109/ICE/ITMC-IAMOT55089.2022.10033313
18. **Marandi, M., Bertia, A., & Silas, S. (2023).** Implementing and Automating Security Scanning to a DevSecOps CI/CD Pipeline. In 2023 World Conference on Communication & Computing (WCONF) (pp. 1-6). doi: 10.1109/WCONF58270.2023.10235015
19. **Efendi, M., Raharjo, T., & Suhanto, A. (2021).** DevSecOps Approach in Software Development Case Study: Public Company Logistic Agency. In 2021 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS) (pp. 96-101). doi: 10.1109/ICIMCIS53775.2021.9699316
20. **Ahmed, Z., & Francis, S. C. (2019).** Integrating Security with DevSecOps: Techniques and Challenges. In 2019 International Conference on Digitization (ICD) (pp. 178-182). doi: 10.1109/ICD47981.2019.9105789
21. **Zampetti, F., Geremia, S., Bavota, G., & Di Penta, M. (2021).** CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study. In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 471-482). doi: 10.1109/ICSME52107.2021.00048

# Appendixes

1. **Appendix A: Tool Selection and Setup** 
   * Criteria for choosing DevSecOps tools.
   * Steps for installing and setting up tools.
2. **Appendix B: Data Collection and Analysis** 
   * Data from testing phases.
   * Data analysis methodologies.
3. **Appendix C: Security Assessments** 
   * Reports before and after DevSecOps practices.
4. **Appendix D: Manuals and Best Practices** 
   * Guide to DevSecOps integration.
   * Security best practices checklist.