DevSecOps approach: Integrating security into development pipeline

Filip Pospisil

Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru
Email: filip.pospisil@student.um.si

Change detection in Synthetic Aperture Radar (SAR) imagery is crucial for environmental monitoring and disaster assessment but is hindered by speckle noise and acquisition inconsistencies. The study "Improved Difference Images for Change Detection Classifiers in SAR Imagery Using Deep Learning" proposed a neural network-based mapping transformation to generate artificial SAR images with consistent acquisition conditions, improving difference image (DI) quality and reducing false positives. This paper tries to replicate the original experiment using the same dataset and validation methods but with a reduced computational scope. The replication follows a structured SAR change detection pipeline: preprocessing for noise reduction and alignment, DI formation using deep learning instead of traditional algebraic methods, and classification using thresholding and Support Vector Machines (SVMs). A subset of the dataset is used to compare deep learning-enhanced DIs against conventional methods. Performance is evaluated using Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) scores. This replication assesses the feasibility of deep learning-based SAR change detection on computationally constrained systems, validating the original findings and exploring broader applicability

Keywords: devsecops, pipelines, security, github actions;

1. INTRODUCTION

1.1. Problem statement

Synthetic Aperture Radar (SAR) has become a crucial tool in remote sensing change detection, enabling consistent monitoring of Earth's surface under diverse weather conditions. Unlike optical imaging, which offers higher spatial resolution but is limited by cloud cover and daylight availability, SAR operates in the microwave spectrum, allowing data acquisition regardless of atmospheric conditions. This makes SAR particularly valuable for applications such as post-disaster damage assessment, forest damage detection, and long-term environmental monitoring, including deforestation and glacier melting.

Despite its advantages, SAR imagery presents unique challenges. The technique relies on a moving antenna to synthesize a virtual aperture, which results in speckle noise—a grainy pattern caused by random backscatter within each resolution cell. Even with advanced noise-suppression algorithms, speckle remains an issue, reducing the clarity of the difference image (DI) generated by subtracting or ratioing temporal SAR acquisitions. To improve DI quality and enhance the accuracy of change detection, additional pre-processing steps are required.

- **Preprocessing**: Applying speckle filtering and radiometric calibration to ensure accurate comparisons.
- **Difference Image (DI) Formation**: Generating a DI using algebraic operations (e.g., subtraction, ratio, or log-ratio) between two SAR images.
- **Classification**: Identifying changed regions in the DI through clustering or machine learning classifiers.

To address the challenges of DI quality, Alatalo et al. proposed a neural network-based mapping transformation function that improves SAR image comparability by normalizing acquisition conditions, considering imaging angles, digital elevation models (DEMs), and weather data. Their method, tested using Sentinel-1 SAR imagery over Finland, demonstrated significant improvements in DI quality and change detection accuracy compared to conventional approaches.

In this study, we aim to replicate the core methodology of Alatalo et al. using the same dataset and validation techniques while adapting the computational scale due to hardware constraints. Given the resource-intensive nature of deep learning models, our approach will focus on a subset of the available data to assess whether similar improvements in DI quality and classifier performance can be achieved under limited computational resources. By systematically evaluating noise suppression, DI formation, and

classification steps, this replication study seeks to validate the effectiveness of deep learning-based SAR image processing and explore its feasibility for practical applications on less powerful computing systems.

Manual Security Processes

Many organizations (especially those with legacy systems) still use manual security reviews. These are:

- **Time-intensive**: Security audits can be lengthy and require special resources which are slowing development.
- **Error-prone**: Human mistakes during manual reviews can miss vulnerabilities. (Especially in complex projects.)
- **Inconsistent**: The quality of security checks depends on the reviewer's skills and how carefully they follow procedures.

Increasing Complexity of Software Supply Chains

Modern software often depends on external libraries, containerized environments and distributed architectures, which makes security harder to manage:

- **Dependency vulnerabilities**: Projects use libraries with hidden dependencies, making it almost imposible to find and fix security issues.
- **Dynamic configurations**: Container orchestration tools adds extra layers of complexity in securing settings.
- **Lack of visibility**: It can be difficult to see the overall security status of the entire supply chain.

These problems highlight the urgent need for a new way of handling security in software development.

2. IMPORTANCE OF DEVSECOPS

2.1. What is DevSecOps?

DevSecOps is a method that integrates security into DevOps workflows, ensuring security is part of every phase of the Software Development Lifecycle. Unlike older methods where security is treated separately at the end, DevSecOps shifts security to the beginning. This means security risk are recognized early and continuously handled.

2.2. How Does DevSecOps Work?

- **Automation**: Security tasks are automated in the CI/CD pipeline for consistent and repeatable results.
- **Collaboration**: Developers, operations teams, and security experts work together to keep good security without slowing down work.
- **Continuous monitoring**: DevSecOps pipelines constantly check for vulnerabilities, giving quick feedback and supporting ongoing improvements.

2.3. Benefits of Integrating Security into CI/CD Pipelines

Early detection and prevention

- Vulnerabilities are found during development.
- Tools like Static Application Security Testing and dependency scanning help keep code quality high and reduce risks.

Efficiency

- Automated tools like Snyk and Dependabot detect vulnerabilities and update dependencies automatically.
- This reduces the need for manual security checks and speeds up delivery.

Enhanced collaboration

- Development, operations, and security teams share responsibility and can communicate better.

Scalability and adaptability

- Automated pipelines handle large or small projects and can secure dynamic environments.

Reduced risk exposure

- Continuous feedback loops helps to manage risks early, preventing vulnerabilities from reaching production.

3. SNYK

Snyk is a tool designed to help developers find and fix security issues in their projects. It focuses on making security part of the development process instead of treating it as a separate step. Snyk supports modern software practices like Agile, microservices, and DevOps, making it a key tool for improving security without slowing down development.

3.1. Key features of Snyk

Snyk provides several features that help developers secure their projects at every stage of the Software Development Lifecycle:

Dependency scanning

- Snyk checks project dependencies to find vulnerabilities in libraries and frameworks. This includes direct dependencies and transitive ones.
- It uses a reliable database of known vulnerabilities to ensure up-to-date and accurate results.
- Suggests fixes such as updating to a safer version of the dependency.

Code scanning

- Snyk can analyze your code and detect common security issues and insecure configurations.
- It helps developers write safer code from the start.

Container security

- Snyk can even scan container images for vulnerabilities in the base images or installed packages.
- It provides guidance to create more secure containers and reduce risks in production environments.

Integration with CI/CD pipelines

- Snyk integrates with Continuous Integration and Continuous Delivery tools like Jenkins, GitHub Actions, ...
- Automates security checks, ensuring vulnerabilities are caught early in the development process.

3.2. How Snyk fits into DevSecOps

Snyk is a great example of how security can be integrated into DevSecOps. Its approach ensures that security is considered during every phase of development.

Early detection of vulnerabilities

- By scanning dependencies, code and configurations during development Snyk allows developers to fix issues early. Th
- Tools like Snyk make it possible to detect risks before they become serious vulnerabilities.

Automation and efficiency

- Security tasks are automated, so developers don't need to manually scan for issues.
- This reduces delays and ensures consistent results across projects.

Friendly design

- Snyk focuses on being easy to use for developers. Its interface and tools are designed to fit into the development process.
- It provides clear and actionable guidance so that developers can fix issues without needing deep security expertise.

Continuous monitoring

- After deployment, Snyk continues to monitor your projects and alerts you if new vulnerabilities are discovered in your dependencies or containers.

3.3. Conclusion

Snyk is a powerful tool that helps developers integrate security into their workflows. By automating security checks, providing clear guidance, and working with modern tools, Snyk makes it easier to build secure software without slowing down the process speed. Its ability to detect and fix vulnerabilities early aligns perfectly with DevSecOps principles, making it an essential part of any secure development process.

4. DEPENDABOT

Dependabot is a native tool on GitHub designed to help developers keep their project dependencies secure and up-to-date. Automates the process of checking for outdated or vulnerable dependencies and generates pull requests to update them. Dependabot ensures that projects stay secure and compliant with minimal manual effort, making it a valuable addition to modern development workflows.

4.1. Key features of Dependabot

Dependabot provides essential features that simplify dependency management and improve project security:

Automated dependency updates and vulnerability alerts

- Dependabot regularly scans your project dependencies and identifies outdated versions.
- It automatically generates pull requests to update dependencies to the latest versions.
- It provides detailed information about the vulnerability, including its severity and recommended remediation.

Custom configuration

- Developers can configure Dependabot to check for updates daily, weekly, or monthly, depending on project needs.
- It allows users to exclude specific dependencies or set rules for update behavior, ensuring flexibility.

Integration with GitHub

- Dependabot is built into GitHub which making it easy to set up and use within GitHub repositories.
- It integrates with CI/CD workflows and enabling automated testing of dependency updates.

Compatibility testing

- Dependabot ensures that updates do not break your project by running tests as part of the PR process.
- This guarantees that updated dependencies are safe and compatible with the existing codebase.

4.2. How Dependabot fits into DevSecOps

Dependabot aligns with DevSecOps principles by automating security tasks and enabling faster detection

and resolution of vulnerabilities in dependencies. It simplifies dependency management and ensures that security remains a priority throughout the development cycle.

Early detection of vulnerabilities

- Dependabot helps detect vulnerabilities in external libraries before they become a risk to production.
- Notifies developers as soon as vulnerabilities are discovered.

Automation and efficiency

- By automating dependency updates will Dependabot reduces the manual effort needed to monitor and update libraries.

Friendly design

- Dependabot is easy to use and requires minimal setup. Developers can enable it directly in their GitHub repositories.
- The generated pull requests are clear, actionable and providing all the necessary details to make informed decisions.

Continuous monitoring

- Dependabot continuously scans for vulnerabilities and outdated dependencies, ensuring projects remain secure over time.

4.3. Conclusion

Dependabot is an effective tool for automating dependency management and improving security in software projects. Its ability to detect vulnerabilities early, automate updates, and seamlessly integrate with GitHub makes it an essential component of DevSecOps practices. By reducing manual effort and ensuring up-to-date dependencies Dependabot helps developer maintain secure, high-quality software without slowing down the process speed.

5. SONARQUBE

SonarQube is a powerful platform for continuous inspection of code quality and security. It helps developers and organizations detect code smells, bugs, and vulnerabilities in their source code. By integrating with modern development workflows, SonarQube ensures that high quality and secure code is delivered throughout the Software Development Lifecycle. Its ability to support multiple programming languages and provide actionable insights makes it a valuable tool in any DevSecOps pipeline.

5.1. Key features of SonarQube

SonarQube offers a wide range of features to improve code quality and enhance security:

Static code analysis

- SonarQube scans source code to detect bugs, security vulnerabilities, and code smells in multiple programming languages.

Support for multiple languages

- SonarQube supports over 25 programming languages, including Java, Python, JavaScript, TypeScript, ...

Security rules

- SonarQube includes rules that cover the OWASP Top 10 and SANS Top 25 vulnerabilities.
- It provides clear explanations of each issue and guidance on how to fix them.

Integration with CI/CD pipelines

- SonarQube integrates with Continuous Integration and Continuous Delivery tools like Jenkins, GitHub Actions, ...
- This allows developers to enforce code quality gates as part of the build process.

Quality gates

- Quality gates define criteria that code must meet to pass inspections, such as acceptable levels of code coverage, maintainability, and security ratings.
- They act as checkpoints to prevent bad code.

Developer feedback

- SonarQube provides detailed feedback on issues directly in the code which makes it easy for developers to understand and resolve problems.

5.2. How SonarQube fits into DevSecOps

SonarQube aligns with DevSecOps principles by enabling continuous code quality and security monitoring. Promotes a proactive approach to identifying and resolving issues early in the development process.

Early detection of issues

- SonarQube helps catch bugs, vulnerabilities, and code smells early in development, reducing the cost and effort of fixing them later.
- Ensures that security risks are addressed before production begins.

Automation and eficiency

- By integrating with CI/CD pipelines SonarQube automates code analysis and enforces quality gates.
- This reduces manual effort and ensures consistent results across projects.

Improved code quality

- SonarQube promotes clean and maintainable code by providing actionable insights into technical debt and maintainability issues.

5.3. Conclusion

SonarQube is a comprehensive tool for ensuring code quality and security. By integrating static code analysis, quality gates, and actionable insights into development workflows, it supports DevSecOps practices and enables teams to deliver secure, high quality software. Its ability to detect issues early, automate checks, and provide continuous feedback makes it an essential part of modern software development pipelines.

6. SONARCLOUD

SonarCloud is a cloud based platform provided by the developers of SonarQube. It offers similar features for continuous code quality and security inspection but is designed to work entirely in the cloud. SonarCloud eliminates the need for managing infrastructure, making it particularly appealing for organizations that prefer cloud-native solutions.

6.1. Key features of SonarCloud

SonarCloud shares many features with SonarQube, but it is optimized for cloud-based workflows:

Cloud based service

- SonarCloud operates entirely in the cloud so requiring no local server installation or maintenance.
- It is accessible via a web interface and integrates seamlessly with cloud-hosted repositories like GitHub, GitLab, ...

Scalability

- As a cloud service SonarCloud scales automatically based on project size and analysis demands.

CI support

- SonarCloud integrates with CI/CD pipelines for automated code analysis.
- It supports tools like GitHub Actions, GitLab CI, ...

6.2. SonarCloud vs. SonarQube

While SonarCloud and SonarQube share the same core functionalities, they differ in their deployment model and target audience. Table 1 summarizes the key differences between the two.

6.3. Use cases for SonarCloud and SonarQube

SonarCloud:

- Ideal for teams using cloud-hosted repositories and CI/CD pipelines.
- Suitable for startups and small teams that prefer minimal setup and maintenance.

SonarQube:

- Suitable for enterprises with strict data privacy requirements or existing on-premises infrastructure.
- A good choice for teams requiring extensive customization of rules and configurations.

6.4. Conclusion

SonarCloud and SonarQube are powerful tools for code quality and security, but they cater to different needs. SonarCloud is perfect for organizations looking for a cloud-native, hassle-free solution, while SonarQube offers more control for those with specific infrastructure and data privacy requirements. Choosing between the two depends on the organization's workflow, budget, and security considerations.

7. COMPARISON OF SNYK, DEPENDABOT, AND SONARQUBE

Snyk, Dependabot, and SonarQube are three distinct tools that address different aspects of software security and quality. Although all three contribute to DevSecOps practices, their focus areas and functionalities vary. This section provides a detailed comparison of these tools and their roles in modern software development. Table 2 is summaring main features of each component.

7.1. Overview of Key Differences

Focus area

- **Snyk:** Focuses on identifying and remediating vulnerabilities in dependencies, container images, and Infrastructure as Code configurations.
- **Dependabot:** Automates dependency updates and ensures that projects use the latest secure versions of libraries.
- **SonarQube:** Focuses on improving code quality and identifying security vulnerabilities in source code through static analysis.

TABLE 1: Comparison of SonarCloud and SonarQube

Feature	SonarCloud	SonarQube
Deployment model	dully cloud-based	self-hosted or on-premises
Maintenance	no maintenance required; managed by SonarSource	requires installation, updates, and server management
Integration	seamless integration with cloud platforms like GitHub, GitLab, ...	integrates with both cloud and on-premises CI/CD pipelines
Scalability	automatic, managed by SonarSource	requires manual configuration and resource allocation
Cost model	subscription-based pricing, often per codebase or user	license-based pricing for the server instance, often tiered by lines of code
Data privacy	code and analysis results stored in the cloud	data remains in the organization's infrastructure
Target audience	teams and organizations seeking cloud-native solutions	organizations with strict data control requirements or on-premises workflows

Integration

- Snyk integrates with CI/CD pipelines, repositories, and container registries for comprehensive security.
- Dependabot is built into GitHub and seamlessly integrates with its ecosystem.
- SonarQube supports integration with CI/CD tools and IDEs for continuous code quality monitoring.

Automation

- Snyk automates vulnerability scanning and remediation through patches and pull requests.
- Dependabot automates the creation of pull requests to update outdated or vulnerable dependencies.
- SonarQube automates code quality checks through static analysis and enforces quality gates.

7.2. Using Snyk and Dependabot Together

Snyk and Dependabot might seem to look almost the same as they are complementary tools that enhance software security and dependency management. While both address vulnerabilities, their unique features make them valuable when used in tandem.

Overview of Key Benefits

Focus Area

- **Snyk:** Identifies and remediates vulnerabilities in dependencies, containers, and Infrastructure as Code.
- **Dependabot:** Automates dependency updates, ensuring projects use the latest secure versions.

Automation

- **Snyk:** Automates vulnerability scanning and generates security fix pull requests.
- **Dependabot:** Automates pull requests for dependency updates, including minor and patch versions.

Benefits of using both combined

- Proactive dependency updates with Dependabot.
- Deep vulnerability scanning and remediation with Snyk.
- Comprehensive security coverage with minimal maintenance effort.

8. PROOF OF CONCEPT

This pipeline is designed to implement DevSecOps practices by automating dependency updates, vulnerability scans, and code quality analysis. Each job addresses a specific aspect of securing and improving the software development lifecycle. The pipeline is available in the GitHub repository at `https://github.com/fifa-um-si/ais-project-poc`, where its functionality will be presented and demonstrated. Additionally, the repository contains an example project used to showcase the pipeline's features in a practical context. Content of this pipeline including example project is also attachment of this paper in eStudij. Table 3 contains code of main pipeline, table 4 contains code of dependabot settings.

8.1. Dependabot Dependency Updates

This job runs when Dependabot creates a pull request. It checks out the code and fetches metadata for dependencies using the Dependabot fetch-metadata action. The primary purpose is to automate dependency updates and ensure the project uses the latest secure versions.

TABLE 2: Comparison of Snyk, Dependabot, and SonarQube

Feature	Snyk	Dependabot	SonarQube
Primary focus	dependency, container, and IaC security	automated dependency updates	code quality and security
Automation	vulnerability scanning and fixes	dependency update pull requests	static code analysis
Integration	CI/CD, repositories, cloud platforms	GitHub ecosystem	CI/CD, IDEs
Feedback type	vulnerability details and fixes	update suggestions	detailed code insights
Use case	comprehensive security monitoring	automated dependency management	continuous code quality improvement

8.2. Snyk Vulnerability Scan

This job performs a vulnerability scan using Snyk to identify security issues in the codebase. It checks out the code, installs dependencies, authenticates with the Snyk CLI, and runs a vulnerability test. Additionally, it monitors the project for new vulnerabilities over time by submitting the project to Snyk for continuous analysis.

8.3. SonarCloud Code Quality and Security Analysis

This job integrates with SonarCloud to analyze the project's code for quality and security issues. It checks out the code, sets up a Java 17 environment, and executes the SonarCloud scan. The scan evaluates the codebase for potential bugs, security vulnerabilities, and adherence to coding standards, ensuring high code quality and security.

8.4. Example Project

The repository also includes an example project that demonstrates the real-world application of the DevSecOps pipeline. This project serves as a practical implementation, allowing users to observe how the pipeline automates dependency updates, scans for vulnerabilities, and assesses code quality, ensuring a secure and efficient development process.

.

LITERATURE

[1] B. Jammeh *et al.*, "Devsecops: Security expertise a key to automated testing in ci/cd pipeline," *International Journal of Scientific Research*, vol. 12, no. 12, pp. 2073–2077, 2023, accessed: 2025-01-07. [Online]. Available: https://www.ijsr.net/archive/v12i12/SR24304171058.pdf

[2] Deloitte, "Integrating and automating security into a devsecops model," *Deloitte Reports*, 2024, accessed: 2025-01-04. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/us/Documents/risk/us-integrating-and-automating-security-into-a-devsecops-model.pdf

[3] Deloitte Insights, "Embedding security into devops pipelines," *Deloitte Insights*, 2024, accessed: 2025-01-03. [Online]. Available: https://www2.deloitte.com/us/en/insights/focus/tech-trends/2019/embedding-security-devops-pipelines-devsecops.html

[4] DevOps Door, "Devsecops - integrating security into the ci/cd pipeline," *DevOps Door*, 2024, accessed: 2025-01-05. [Online]. Available: https://devopsdoor.com/blog/devops/leaders/beginners/2024/05/06/DevSecOps/

[5] E. Wahed, "Evaluation of sast and dast tools for web application security," *NUST Institutional Repository*, 2023, accessed: 2025-01-02. [Online]. Available: https://repositories.nust.edu.pk/xmlui/handle/123456789/44613

[6] Java Code Geeks, "Devsecops: Integrating security into the ci/cd pipeline," *Java Code Geeks*, 2024, accessed: 2025-01-07. [Online]. Available: https://www.javacodegeeks.com/2024/12/devsecops-integrating-security-into-the-ci-cd-pipeline.html

[7] J. Desai, "A comprehensive comparison of sast tools – sonarqube vs snyk," *Jinal Desai Blog*, 2024, accessed: 2025-01-10. [Online]. Available: https://jinaldesai.com/a-comprehensive-comparison-of-sast-tools-sonarqube-vs-snyk/

[8] Microsoft Security, "What is devsecops? definition and best practices," *Microsoft Security*, 2024, accessed: 2025-01-09. [Online]. Available: https://www.microsoft.com/en-us/security/business/security-101/what-is-devsecops

[9] Practical DevSecOps, "What is devsecops pipelines? - easy guide to understand," 2024, accessed: 2025-01-05. [Online]. Available: https://www.practical-devsecops.com/what-is-devsecops-pipelines/

[10] R. Chandramouli and F. Kautz, "Strategies for the integration of software supply chain security in devsecops ci/cd pipelines," *NIST Special Publication*, vol. 800-204D, 2024, accessed: 2025-01-04. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-204D

[11] Security Boulevard, "Overcome ast noise to find and fix software vulnerabilities," *Security Boulevard*, 2025, accessed: 2025-01-03. [Online]. Available: https://securityboulevard.com/2025/01/overcome-ast-noise-to-find-and-fix-software-vulnerabilities/

TABLE 3: .github/workflows/devsecops-pipeline.yml

```yaml
name: DevSecOps Pipeline

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
  schedule:
    - cron: '0 3 * * *'

jobs:
  dependabot:
      name: Dependabot Dependency Updates
      if: github.actor == 'dependabot[bot]'
      runs-on: ubuntu-latest
      steps:
        - name: Checkout code
          uses: actions/checkout@v3

        - name: Fetch Dependabot Metadata
          uses: dependabot/fetch-metadata@v1
          with:
            github-token: ${{ secrets.GITHUB_TOKEN }}

  snyk:
    name: Snyk Vulnerability Scan
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Install dependencies
        run: npm install

      - name: Install Snyk CLI
        run: npm install -g snyk

      - name: Authenticate Snyk
        run: snyk auth ${{ secrets.SNYK_TOKEN }}

      - name: Snyk Test for Vulnerabilities
        run: snyk test

      - name: Snyk Monitor Project
        run: snyk monitor

  sonarcloud:
    name: SonarCloud Code Quality and Security Analysis
    if: github.actor != 'dependabot[bot]'
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'

      - name: Run SonarCloud Scan
        uses: sonarsource/sonarcloud-github-action@v4.0.0
        env:
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
        with:
          args: >
            -Dsonar.token=${SONAR_TOKEN}
            -Dsonar.projectKey=fifa-um-si_ais-project-poc
            -Dsonar.organization=fifa-um-si
            -Dsonar.host.url=https://sonarcloud.io
```

TABLE 4: .github/dependabot.yml

```
version: 2
updates:
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "daily"
```