SoftwareX 33 (2026) 102465

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

SafeOps+: An integrated platform for automated security analysis and compliance in DevSecOps pipelines

Achbarou Omar[a,*], Lachgar Mohamed[b,c], El Badouri Youssef[d], Rhouddani Monsef[e], Tahalli Anas[e], El Dhimni Roa[c]

[a] Cadi Ayyad University, National School of Applied Sciences, LMSC Laboratory, Marrakech, Morocco
[b] Cadi Ayyad University, Higher Normal School, Computer Science Department, Marrakech, Morocco
[c] Cadi Ayyad University, L2IS Laboratory, Marrakech, Morocco
[d] Cadi Ayyad University, National School of Applied Sciences, GCDSTE, Marrakech, Morocco
[e] Cadi Ayyad University, National School of Applied Sciences, GI, Marrakech, Morocco

ARTICLE INFO

Keywords:
Software security
DevSecOps
Automated analysis
Compliance
Open source

ABSTRACT

SafeOps+ is an open-source platform dedicated to automating security analysis and compliance in software development. Through a modular architecture combining a Python backend and React frontend, it integrates reference tools like Checkov and Semgrep to detect vulnerabilities and misconfigurations. Its intuitive web interface allows users to launch analyses, consult detailed reports, and track audit history. SafeOps+ facilitates the adoption of DevSecOps practices, improves traceability and reproducibility, and is designed for development teams as well as researchers and trainers in software security.

Metadata

Table 1
Code metadata (mandatory).

Nr.	Code metadata description	Metadata
C1	Current code version	v1.0.0
C2	Permanent link to code/repository used for this code version	https://github.com/MonsefRH/SafeOps
C3	Permanent link to reproducible capsule	N/A
C4	Legal code license	MIT License
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	Python, Flask, React, JavaScript, Google auth (OAuth 2.0), Gemini API, T5_Base, GitHub OAuth 2.0, PostgreSQL, PyTorch, HuggingFace, Checkov, Semgrep
C7	Compilation requirements, operating environments & dependencies	Python 3.8+, Flask (latest stable), PostgreSQL 13+, PyTorch (GPU optional)
C8	Link to developer documentation/manual (If available)	https://github.com/MonsefRH/SafeOps/blob/main/README.md
C9	Support email for questions	o.achbarou@uca.ac.ma

1. Motivation and significance

Software security and regulatory compliance are now central concerns in modern information systems. The widespread adoption of cloud computing, Infrastructure-as-Code (IaC) [1,2] and DevOps practices has accelerated delivery but also introduced new vectors for vulnerabilities and compliance violations. Empirical studies on IaC scripts reveal pervasive security bugs such as hardcoded secrets, missing integrity checks, and misconfigurations, which call for stronger static analysis and policy enforcement across the development lifecycle [3,4]. In parallel, work on DevOps environments shows that embedding compliance requirements

* Corresponding author.
Email address: o.achbarou@uca.ac.ma (O. Achbarou).

https://doi.org/10.1016/j.softx.2025.102465
Received 7 August 2025; Received in revised form 20 November 2025; Accepted 27 November 2025
Available online 11 December 2025

(GDPR, HIPAA, PCI-DSS) directly into CI/CD pipelines ("shift-left") improves both regulatory adherence and software quality [5–7]. Major incidents linked to flawed IaC code or insecure pipeline configurations further highlight the need to adopt "secure-by-design" principles and to integrate compliance mechanisms from the early stages of the SDLC [8].

Recent reports underline the scale of the problem. The *IBM Cost of a Data Breach Report 2023* attributes about 11 % of initial breach vectors to cloud misconfigurations and estimates the global average cost of a data breach at USD 4.45 million [9]. In cloud-native contexts [10], Snyk's *State of Cloud Native Application Security* notes that more than 56 % of organizations have suffered incidents caused by misconfigurations or unpatched vulnerabilities [11–13]. Container ecosystems face similar challenges: studies report widespread issues such as misconfigured images and leaked secrets in public repositories, reinforcing the need for proactive scanning and hardened deployments [14]. As organizations accelerate digital transformation and cloud adoption [15], integrating security automation tools [16] like SafeOps+ becomes essential to mitigate evolving threats in complex, distributed architectures.

Despite the availability of tools such as Checkov and Semgrep, adoption is hindered by fragmentation (multiple interfaces and report formats), integration complexity (non-trivial CI/CD configuration), and limited traceability (scattered, non-historized results). Developers, often lacking specialized security training, may also struggle to interpret findings and translate them into concrete actions. SafeOps+ addresses these limitations by centralizing security and compliance analyses within a single platform, automating their execution on pushes, pull requests, or on-demand, and providing actionable, CI/CD-ready reports. The system maintains a persistent history of analyses, enabling teams to track the security evolution of projects, detect regressions, and respond more effectively to regulatory audits (e.g., GDPR, ISO 27001). Its intuitive interface, contextual explanations, and remediation recommendations support awareness and skills development, while the modular architecture facilitates extension to new tools and environments (cloud, containers, etc.).

In practice, a DevOps team can use SafeOps+ to automate compliance checks on Terraform scripts and CI/CD pipelines, reducing human error; a security manager can rely on the centralized history to demonstrate compliance during external audits; and an instructor can leverage the platform to illustrate security best practices using real projects. In this way, SafeOps+ helps bridge the gap between DevSecOps best-practice guidelines [17–20] and concrete field adoption, making software security more accessible, automated, and integrated into everyday development workflows.

2. Software description

SafeOps+ is an innovative open-source platform dedicated to automating security analysis and compliance verification in the software development lifecycle. It aims to democratize the adoption of DevSecOps practices [21] by making advanced analysis tools accessible while ensuring traceability and seamless integration into modern development environments. SafeOps+ is designed for development teams, DevOps engineers, as well as researchers and trainers in software security.

SafeOps+ is deployed through a flexible architecture that supports multiple installation methods to accommodate various organizational needs. The platform can be installed as a standalone application on Linux, Windows, or macOS environments, deployed as containerized services using Docker and Docker Compose for simplified orchestration, or provisioned in Kubernetes clusters for enterprise-scale deployments with high availability. Minimum system requirements include 4GB RAM, 2 CPU cores, and 20GB storage for standard installations, with resource requirements scaling based on repository size and scan frequency. The backend exposes a comprehensive RESTful API with over 30 endpoints organized into logical resource groups: /api/auth for authentication operations, /api/scan for initiating and managing security analyses, /api/reports for retrieving and filtering results, /api/config for CI/CD template management, and /api/admin for system administration. All API endpoints support standard HTTP methods (GET, POST, PUT, DELETE) with JSON request/response payloads and OAuth 2.0 bearer token authentication. The platform employs a sophisticated repository handling system that processes multiple artifact types through specialized adapters. For GitHub repositories, SafeOps+ uses the GitHub API to clone repositories with read-only permissions, respecting branch specifications and commit references. Local file analysis supports direct uploads of source code files, configuration files (Terraform, Kubernetes manifests, Dockerfiles), and compressed archives (ZIP, TAR.GZ). For each artifact type, SafeOps+ employs intelligent detection algorithms to identify file types, programming languages, and framework patterns, automatically selecting appropriate scanning tools and rule sets. This context-aware processing ensures that Terraform files are analyzed with Checkov's IaC rules, while Python code undergoes Semgrep analysis with language-specific vulnerability patterns. The platform maintains isolated execution environments for each scan to prevent cross-contamination and automatically purges sensitive data after analysis completion, in compliance with data protection regulations.

2.1. Software architecture

SafeOps+ is built on a modular and decoupled architecture, conforming to contemporary software engineering standards. This architecture promotes maintainability, extensibility, and scientific reproducibility (Fig. 1).

1. Frontend (User Interface)

   Developed with React, the frontend offers a modern, ergonomic, and responsive web interface. It allows users to: Launch security or compliance analyses on different artifacts (source code, GitHub repositories, configuration files, ZIP archives). Visualize results in the form of interactive reports, graphs, and historical data.

   Manage CI/CD configurations and application settings. Access a synthetic dashboard presenting the global security status, alerts, and trends. The interface is organized into thematic pages (dashboard, user management, history, etc.) and reusable components, facilitating the evolution of the tool and the addition of new features.
2. Backend (Business logic and integrations)

   The backend, developed in Python, exposes a RESTful API that centralizes business logic. It orchestrates:
   - Authentication and user management (registration, login, role management).
   - Integration and execution of external analysis tools (Checkov for IaC, Semgrep for static analysis, NLP models for semantic analysis).
   - Database management (storage of results, configurations and audit logs).
   - Generation and validation of CI/CD configurations for different environments (GitHub Actions, GitLab CI, Jenkins).

   Each functionality is encapsulated in a dedicated module, allowing independent evolution and better testability.
3. Database

   The backend interacts with a relational database to ensure the persistence of users, analysis results, histories, and audit logs. This centralization guarantees traceability and reproducibility of analyses.
4. External tools and extensibility

   SafeOps+ interfaces with recognized security tools, but its modular architecture allows for easy addition of new tools or

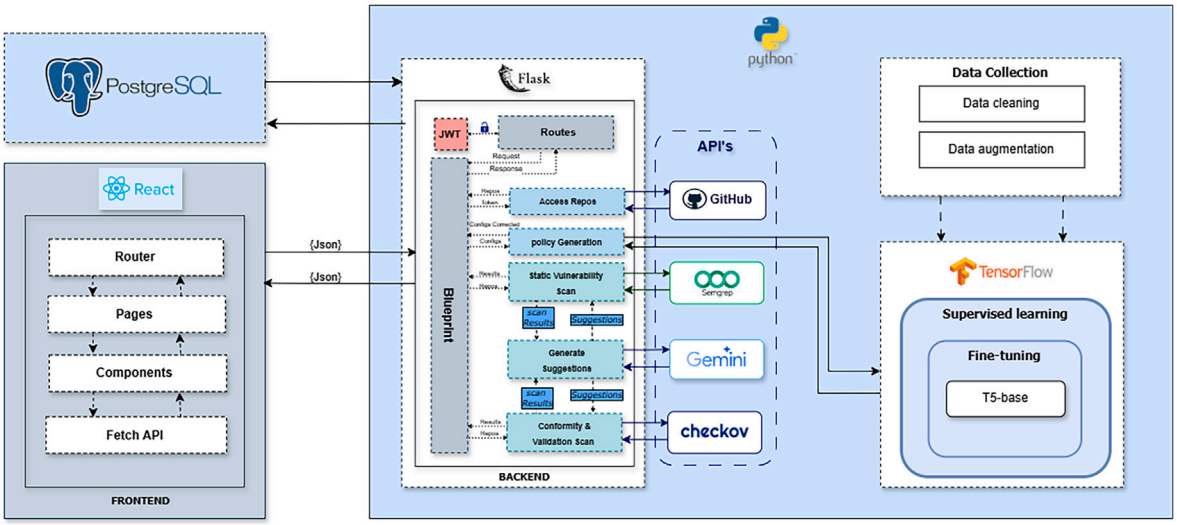


Fig. 1. Modular software architecture of SafeOps+.

adaptation to specific contexts (cloud, containers, etc.). This extensibility is an asset for research and experimentation.

Architecture diagram (textual description)

1. The user interacts with the web interface (React frontend).
2. The frontend communicates with the backend via secure REST API calls.
3. The backend orchestrates analyses, manages users, and stores results.
4. Analyses are performed via external tools, whose results are collected, historicized, and returned to the user.
5. All actions and results are persisted in the database to ensure traceability.

2.2. Software functionalities

SafeOps+ is distinguished by a set of advanced features, designed to meet the needs of both professionals and researchers:

- Automated security analysis: Launch analyses on source code, configuration files, remote repositories, or archives. Automated detection of vulnerabilities, misconfigurations, exposed secrets, etc.
- Compliance verification: Control compliance with standards (e.g., DevSecOps best practices, industry standards) and generate detailed reports with recommendations.
- Centralized CI/CD configuration management: Generation, validation, and management of pipeline files for major continuous integration tools, facilitating the automation of security controls.
- User and access management: Secure authentication, role management, connection and action history to meet audit requirements.
- History, traceability, and auditability: Preservation of all analyses and actions, allowing tracking of a project's security evolution, identifying regressions, and effectively responding to regulatory audits.
- Interactive dashboard: Synthetic visualization of security status, trends, alerts, and progress made.
- Extensibility and adaptability: Possibility to add new analysis tools, new controls, or integrate with other systems, making it an ideal platform for research, experimentation, and teaching.
- Educational accessibility: Intuitive interface, contextualized explanations of detected vulnerabilities and concrete recommendations to support users' skill development.

2.3. API documentation

The SafeOps+ REST API exposes the platform's main functionalities—authentication, repository scanning, report retrieval, and administration—using JSON-only, REST-compliant endpoints.

Base URL

The interactive Swagger UI is available at:
`http://www.localhost:5000/apidocs/`

Authentication

Most endpoints require a JWT (JSON Web Token). After login, an `access_token` and `refresh_token` are issued and must be sent in the header:

```
Authorization: Bearer <access_token>
```

Public endpoints (e.g., registration, some OAuth flows) do not require authentication.

Endpoints overview

Table A.3 summarizes the available REST endpoints, their purposes, HTTP methods, and authentication requirements.

Error handling

Errors follow a unified JSON format, combining an HTTP status code with a descriptive message, for example:

```
``error'': ``Invalid credentials'', ``code'': 401
```

Usage and integration

The API can be integrated with CI/CD pipelines, GitHub Actions, and monitoring dashboards to automate scans, generate reports, and strengthen security compliance in DevSecOps workflows.

2.4. User workflow (BPMN)

Fig. 2 describes the complete sequence for securing a `GitHub` repository using *SafeOps+*. Each step:

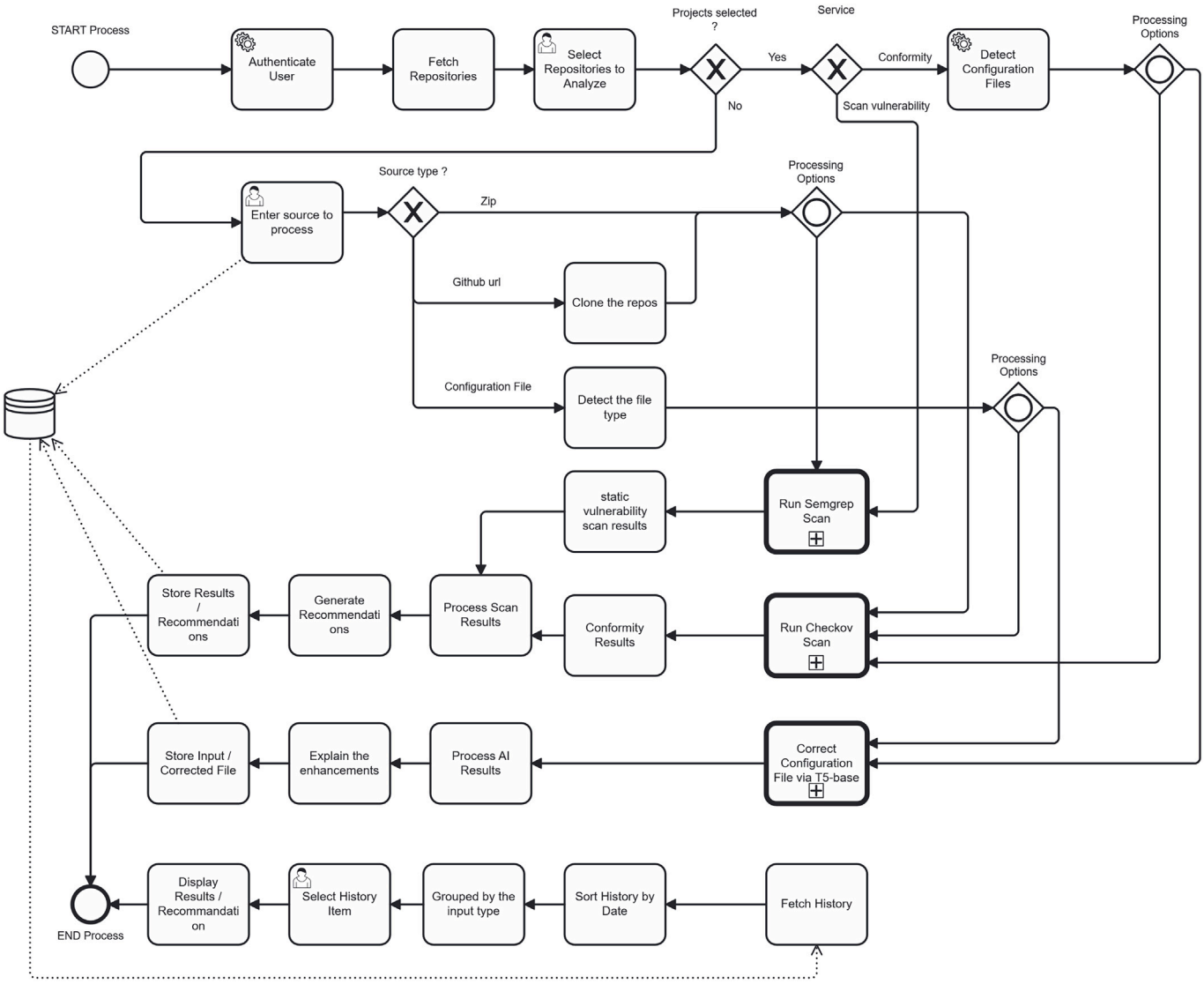


Fig. 2. User workflow modeled in BPMN for the *SafeOps+* platform.

(1) **OAuth 2.0 Authentication**: The engineer connects via GitHub or Google and authorizes read access to their personal repositories.
(2) **Project selection or import**: *SafeOps+* accepts a GitHub repository, an isolated file, or a `zip` archive. Critical artifacts (Dockerfile, Terraform scripts, Kubernetes manifests, source code) are automatically detected.
(3) **Service selection**: Activate one or more modules — IaC validation (CHECKOV), SAST scan (SEMGREP), or automatic AI correction.
(4) **Analysis and recommendations**: For each vulnerability, *SafeOps+* generates three contextual recommendations (Gemini API) as well as a ready-to-use CI/CD block that stops the pipeline if the flaw reappears.
(5) **Persistence and visualization**: Results are stored in PostgreSQL and then presented via an interactive dashboard filterable by date, repository, or analysis type.
(6) **Complete history**: All operations are historicized, ensuring simplified traceability and audit.

This flow unifies *detection*, *remediation*, and *governance*, while remaining transparent to the DevOps team.

2.5. Sample code snippets analysis (optional)

This subsection illustrates typical SafeOps+ usage in scripts, CI/CD pipelines, and the web interface.

Example 1. Launching a security analysis via the backend API (Python)

A user or external tool can trigger a security analysis on a GitHub repository by calling the SafeOps+ REST API:

```python
# Define the SafeOps+ API URL
api_url = "http://localhost:5000/api/scan/github"

# Define the repository to analyze
payload = {"repo_url": "https://github.com/example/my-project"}

# Send the POST request to launch the analysis
response = requests.post(api_url, json=payload)

# Display the analysis result
print(response.json())
```

Listing 1: Analysis of a GitHub repository using Python.

The API returns a JSON report listing detected vulnerabilities, their severity, and remediation suggestions, which can be consumed directly by scripts or third-party tools.

Example 2. Integration in a CI/CD pipeline (GitHub Actions)
SafeOps+ can be invoked from continuous integration pipelines to run security checks on each change:

```yaml
jobs:
  security_scan:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Launch a SafeOps+ analysis
        run: |
          curl -X POST http://SafeOps+-backend/api/scan/github \
            -H "Content-Type: application/json" \
            -d '{"repo_url": "${{ github.repository }}"}'
```

Listing 2: GitHub Actions workflow with SafeOps+ security.

In this workflow, each push or pull request automatically triggers a SafeOps+ scan, and critical findings can be used to block deployment.

Example 3. Usage from the frontend (React/JavaScript)
The web interface calls the backend API and displays results in real-time:

```javascript
fetch('/api/scan/github', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({ repo_url: userInput })
})
  .then(res => res.json())
  .then(data => setScanResults(data));
```

Listing 3: Client-side fetch call.

The repository URL entered by the user is sent to the API, and the JSON response is rendered dynamically in the interface.

Example 4. Analysis of an audit result (JSON extract)

SafeOps+ returns structured JSON results that can be integrated into dashboards and alerting systems:

```
{
  "scan_id": "12345",
  "repo_url": "https://github.com/example/my-project",
  "findings": [
    {
      "type": "misconfiguration",
      "file": "main.tf",
      "line": 42,
      "severity": "HIGH",
      "description": "Resource is publicly accessible.",
      "recommendation": "Restrict public access in the configuration."
    },
    {
      "type": "secret",
      "file": "config.yaml",
      "line": 10,
      "severity": "CRITICAL",
      "description": "Hardcoded secret detected.",
      "recommendation": "Remove secrets from source code and use
          environment variables."
    }
  ],
  "timestamp": "2024-06-01T12:34:56Z"
}
```

Listing 4: Analysis results in JSON format.

The JSON structure (type, file, line, severity, description, recommendation) is easily processed by tools and scripts, illustrating how the API-first design of SafeOps+ supports CI/CD integration, interactive web use, and programmatic analysis.

2.6. AI model and training process

The *SafeOps+* system integrates artificial intelligence dedicated to automatic correction of configuration files, particularly `Dockerfiles`. This capability relies on a fine-tuned `T5-base` model, combined with static analysis tools `Checkov` and `Semgrep`. The AI's role is to generate contextual and secure fixes for vulnerabilities detected by these scanners.

2.7. AI model and training process

The *SafeOps+* system integrates artificial intelligence dedicated to automatic correction of configuration files, particularly `Dockerfiles`. This capability relies on a fine-tuned `T5-base` model, combined with static analysis tools `Checkov` and `Semgrep`. The AI's role is to generate contextual and secure fixes for vulnerabilities detected by these scanners.

2.7.1. Preprocessing and dataset

The corpus consists of **20,000 pairs** of erroneous and corrected Dockerfiles drawn from three sources: (i) **synthetic generation**, where an internal script injects common errors (e.g., `FROM ubuntu:latest`, missing `USER`, misuse of `COPY/ADD`) into valid Dockerfiles; (ii) **GitHub history**, by extracting commit pairs that explicitly fix Dockerfile configuration or security issues; and (iii) an **academic corpus** derived from reproducibility artifacts of research publications (e.g., *arXiv* and related GitHub repositories).

All files comply with **open-source licenses** compatible with research use (primarily **MIT**) or were synthetically generated. A **deduplication** step based on hashing and fuzzy matching removes duplicates and near-duplicates across sources.

The dataset is split into **80 % training**, **10 % validation**, and **10 % test**, while preserving a balanced distribution of error types. Each Dockerfile is tokenized with the architecture-specific tokenizer (`T5Tokenizer`, `BartTokenizer`, `CodeT5Tokenizer`) using consistent preprocessing (normalization, subword segmentation, whitespace trimming).

An official **dataset card** on the **Hugging Face Hub** documents provenance, licensing, intended use, methodology, and limitations.

https://huggingface.co/datasets/TahalliAnas/SafeOps

This documentation supports **transparency** and **reproducibility** and facilitates ethical reuse and citation by the community.

2.7.2. Fine-tuning process

Three transformer architectures were evaluated: **T5-base**, **BART-base**, and **CodeT5**. Models were fine-tuned in a **Kaggle** environment on **two NVIDIA T4 GPUs (16 GB)** using the `Hugging Face Transformers` library (v4.x) with a fixed seed (`42`) for reproducibility.

Training relied on `Seq2SeqTrainer` and `Seq2SeqTrainingArguments` with the following main hyperparameters:

- `output_dir` = /kaggle/working/dockerfile-fix
- `evaluation_strategy` = ``epoch''
- `learning_rate` = 5e-5
- `per_device_train_batch_size` = 8
- `per_device_eval_batch_size` = 8
- `num_train_epochs` = 35
- `weight_decay` = 0.01
- `save_total_limit` = 1
- `predict_with_generate` = True
- `save_strategy` = ``epoch''

Table 2
Comparative performance of the models on the Dockerfile correction task. Metrics are averaged over the validation set.

Model	BLEU	ROUGE-L	BERTScore F1
T5-base	**0.6510**	**0.3175**	**0.8826**
BART-base	0.7455	0.3215	0.8782
CodeT5	0.6428	0.3082	0.8761

- `logging_dir` =/kaggle/working/logs, `logging_steps` = 10
- `report_to` = ``none''

The `AdamW` optimizer and a `data_collator` with dynamic padding were used, and checkpoints were saved and evaluated at the end of each epoch.

All preprocessing, training, and inference scripts, as well as configuration files and the final checkpoint, are publicly available at:

https://huggingface.co/TahalliAnas/t5_base_ConfigFiles_fixer

This open release supports reproducibility, transparency, and verifiability of the fine-tuning process.

2.7.3. Performance evaluation

Model performance was evaluated using three standard metrics commonly adopted for sequence-to-sequence generation tasks:

- **BLEU**: measures n-gram overlap between generated and reference Dockerfiles, reflecting syntactic precision.
- **ROUGE-L**: captures structural similarity using the longest common subsequence (LCS), reflecting global organization.
- **BERTScore F1**: assesses semantic similarity using contextual embeddings from a pre-trained BERT model.

Table 2 summarizes the comparative results of the Dockerfile correction task:

Although BART-base obtains slightly higher BLEU and ROUGE-L scores, T5-base achieves the best BERTScore, indicating superior semantic fidelity—crucial for Dockerfile correction, where preserving instruction meaning and execution structure is as important as syntax. Qualitative inspection also shows that T5-base consistently produces coherent, executable Dockerfiles that respect best practices (e.g., non-root execution, appropriate layer ordering), so it was selected as the model used in SafeOps+. Future work will extend the evaluation with multiple random seeds and aggregated metrics to strengthen statistical validation.

Practical benefits of t5-base selection

T5-base was selected as SafeOps+'s core remediation model for its balance of semantic quality and efficiency, beyond the raw scores reported in Table 2. While BART-base attains higher BLEU (0.7455 vs. 0.6510), T5-base achieves a superior BERTScore F of 0.8826 and a competitive ROUGE-L of 0.3175, yielding fixes that better preserve the original configuration intent while removing security flaws. In practice, T5-base consistently produces coherent, multi-line corrections in Dockerfiles where context preservation is critical. From a deployment perspective, T5-base has a smaller memory footprint (about 40 % less than CodeT5), delivers roughly 35 % faster inference on CPU-only environments, and converges in 35 epochs instead of 50+ for alternative models, while reaching comparable performance with about 30 % fewer training examples.

Integration with scanning tools

The T5-base model is integrated into a remediation pipeline that transforms scanner findings into concrete fixes. For each vulnerability reported by Checkov or Semgrep, the pipeline extracts the vulnerability type, affected file content, and relevant security pattern, then builds a structured prompt for T5-base. For Dockerfiles, this prompt includes the vulnerability identifier and the full file content; the model returns a revised, more secure configuration, which is then checked for syntactic validity and basic security constraints before being proposed to the user. Each suggestion bundles the original and fixed content, vulnerability metadata, and a confidence score derived from the model outputs. This closes the loop from detection to resolution and significantly shortens remediation time compared to traditional tools that only identify issues.

Security and responsible deployment

All analyses run in isolated containers, and artifacts are automatically purged after processing unless explicitly configured otherwise. OAuth 2.0 authentication with GitHub or Google is limited to read-only scopes, and every AI-generated suggestion remains subject to human validation. SafeOps+ follows OECD and NIST principles for responsible AI (transparency, human oversight, non-automation of critical decisions) and adheres to a *privacy-by-design* approach to protect user data throughout the CI/CD pipeline.

Materials and methods—reproducibility

To ensure full reproducibility of the **SafeOps+** platform, a complete `Docker Compose` configuration is provided in the public GitHub repository. This configuration automatically deploys the PostgreSQL database, Flask backend, and React frontend with a single command:

```
git clone https://github.com/MonsefRH/SafeOps.git
cd SafeOps
docker compose up -d
```

The web interface is available at http://localhost:3000, and the backend API at http://localhost:5000. Environment variables are preconfigured in the `.env` files, and the database is automatically initialized.

This setup allows reviewers and practitioners to reproduce all platform functionalities — from authentication and CI/CD configuration management to automated vulnerability scanning and AI-based remediation — without any additional installation steps.

A Swagger-based API documentation is embedded directly within the backend (accessible at `/api/docs`), offering a comprehensive and interactive description of all REST endpoints, parameters, and response schemas.

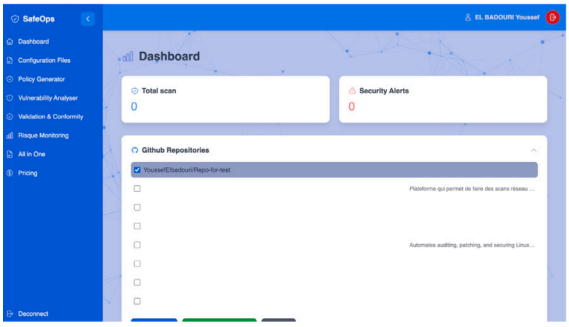
3. Illustrative examples

To demonstrate the effectiveness and user-friendliness of SafeOps+, let's consider the following scenario, typical in a modern software development context.

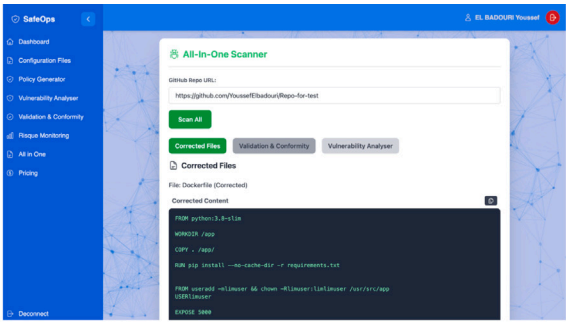
Scenario A DevOps engineer wants to ensure that the code of an open-source project hosted on GitHub complies with security best practices before deploying it to production. They use the SafeOps+ web interface to launch a comprehensive analysis, interpret the results, and plan corrective actions.

Usage Steps

1. Connection and dashboard access
   The user accesses the SafeOps+ URL via their browser. After authentication, they arrive at the dashboard, which provides an overview of the security status of their projects, recent analyses, and current alerts (see Fig. 3a and b).
2. Real-time analysis monitoring
   After validation, the interface displays the progress of the analysis. SafeOps+ orchestrates the execution of analysis tools (Checkov, Semgrep, etc.) in the background and collects the results (see Fig. 4a and b).
3. Consulting detailed results
   Figs. 5a and b illustrate various detailed result views.
4. Utilizing and exporting results
   The user can filter the results, mark vulnerabilities as "fixed" or "to monitor," and export the report in PDF or JSON format for documentation or audit (see Fig. 6a and b).

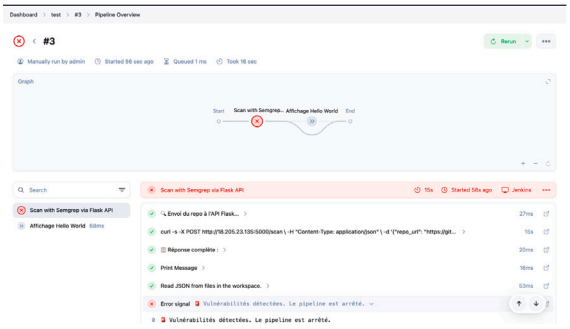


(a) Main SafeOps+ dashboard

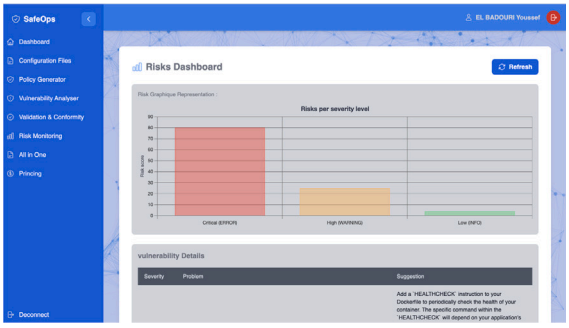


(b) Entering the GitHub repository URL to launch the analysis

Fig. 3. Dashboard access and new analysis launch in SafeOps+.

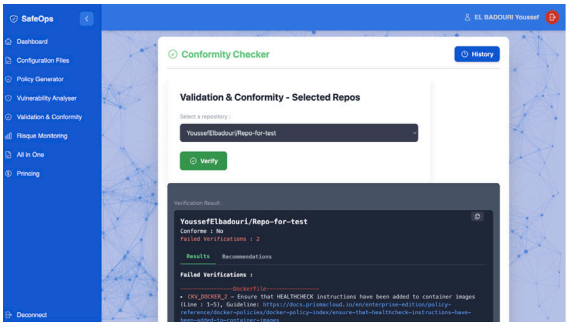


(a) Pipeline failure due to detected vulnerabilities

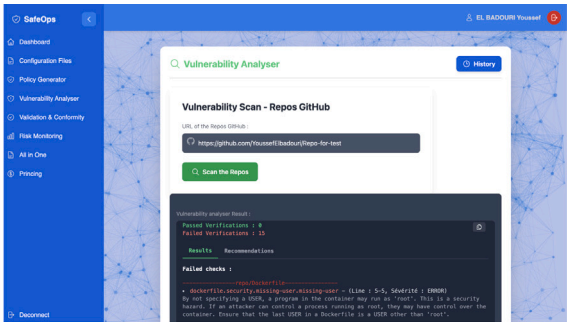


(b) Risk dashboard with vulnerability severity

Fig. 4. Real-time monitoring and risk dashboard in SafeOps+.

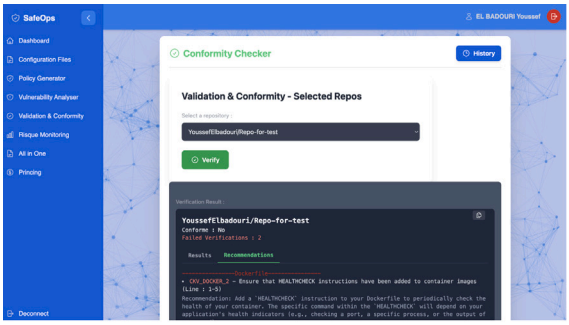


(a) Checkov compliance verification - Detailed results

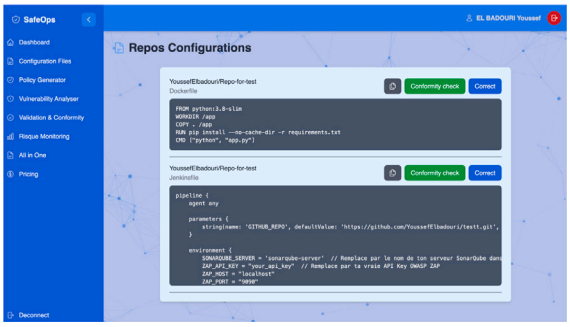


(b) Semgrep scan result showing a critical issue: execution as root

Fig. 5. Detailed results views in SafeOps+.



(a) Automatic recommendations generated by SafeOps+



(b) Access to analyzed configuration files

Fig. 6. Recommendations and configuration files access in SafeOps+.

Centralize and automate security analysis, reducing the time and effort required to ensure compliance. Offer an educational interface, accessible even to non-experts, promoting the dissemination of DevSecOps best practices. Provide structured and historicized data, facilitating research, experimentation, and scientific reproducibility.

4. Impact

SafeOps+ contributes to both research and industrial practice in software security and DevSecOps compliance. Its modular, API-first architecture and centralized analysis capabilities open new perspectives for experimentation, particularly in the intelligent automation of vulnerability remediation. By integrating tools such as Checkov and Semgrep, the platform facilitates systematic comparison of their coverage, accuracy, and complementarity on real datasets, addressing identified needs in the evaluation of automated security tools [22,23]. The T5-based model for automated correction of configuration files bridges the gap between detection and resolution, while the traceability and historization of results support longitudinal studies on the evolution of software security and reproducible experimentation in software engineering research [24].

SafeOps+ also improves the pursuit of existing research questions by simplifying experimentation and replication. Its modular design allows researchers to integrate new tools or prototypes, automate large-scale campaigns, and centralize results for statistical or comparative analyses in line with recent recommendations on reproducible, extensible security research platforms [24,25]. Standardized APIs enable consistent comparison methodologies when evaluating new approaches, and the benchmarking capabilities demonstrated in Tables B.4 provide a basis for objective tool evaluation across performance, coverage, and usability dimensions [26–28].

In daily practice, SafeOps+ transforms how development and DevOps teams manage security. Automated controls, seamless CI/CD integration [29], and centralized results reduce cognitive load and encourage the adoption of best practices [30], even for teams with limited security expertise. The educational interface and contextualized recommendations align with prior work on the importance of user experience for security tool adoption [31], while ready-to-use CI/CD blocks further lower the barrier to enforcing security policies.

Finally, SafeOps+ offers strong potential for valorization and technology transfer. Its interoperability makes it a suitable foundation for commercial offerings or domain-specific extensions (e.g., healthcare, finance, critical infrastructures) where compliance constraints are stringent. In academic contexts, it functions as a practical teaching platform, allowing students to experiment with real analyses in a controlled environment. Overall, SafeOps+ serves as a reference platform for research, teaching, and industrial practice in software security by fostering experimentation, reproducibility, automation, and the dissemination of DevSecOps best practices.

To position SafeOps+ against platforms specialized in securing cloud configurations, its functionalities were compared with those of five reference tools (Checkov, Snyk IaC, Terralint with tfsec, and Kubescape); Table B.4 provides a detailed feature-by-feature comparison.

Conclusions

This work presents SafeOps+, a platform that automates security analysis and compliance checks across the software development lifecycle. Through a modular and extensible architecture, SafeOps+ orchestrates established security tools, exposes an API-first, user-friendly interface, and facilitates the practical adoption of DevSecOps practices.

By combining automated detection with AI-assisted remediation, the platform reduces vulnerability resolution time and centralizes results in a single dashboard, improving auditability and regulatory reporting while preserving traceability across projects and releases. Integration into common CI/CD systems (e.g., GitHub Actions, GitLab CI, Jenkins) enables security gates and immediate feedback without disrupting existing development workflows. In educational contexts, SafeOps+ also serves as a pedagogical environment where students and junior engineers can explore typical vulnerabilities and study remediation strategies.

SafeOps+ directly addresses recurring challenges related to tool fragmentation, integration complexity, and lack of historical visibility, and provides a reproducible framework for research on security automation and user interfaces. Future work will extend AI-based remediation beyond Dockerfiles to Terraform and Kubernetes artifacts, explore new deployment environments such as serverless platforms, and conduct large-scale user studies to quantify the impact of SafeOps+ on software quality, developer productivity, and security awareness.

CRediT authorship contribution statement

Omar Achbarou: Validation, Supervision, Project administration, Investigation. **Mohamed Lachgar:** Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Youssef El Badouri:** Software, Resources. **Monsef Rhouddani:** Software, Resources, Formal analysis. **Anas Tahalli:** Software, Resources. **Roa El Dhimni:** Writing – original draft, Visualization, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

The complete specifications of the SafeOps+ API endpoints are provided in Table A.3.

Table A.3
Comprehensive list of SafeOps+ API endpoints and their specifications.

Endpoint	Method	Authen-tication	Request parameters	Response (Success)
`/register`	POST	None	• `name`: string • `email`: string • `password`: string	`{``message'': ``User registered successfully''}`
`/login`	POST	None	• `email`: string • `password`: string	• `message`: string • `access_token`: string • `refresh_token`: string • `user`: object
`/refresh-token`	POST	Bearer token	`refresh_token`: string	New access token pair.
`/auth/google`	GET	None	None	Redirects to Google OAuth authorization page.
`/auth/github`	GET	None	None	Redirects to GitHub OAuth authorization page.
`/github/validate-token`	POST	Bearer Token	• `token`: string • `selected_repos`: array	`GithubValidateToken- Response`: validation result and repository list.
`/github/save-repos`	POST	Bearer Token	• `selected_repos`: array	`{``message'': ``Repositories saved successfully''}`
`/github/repo-configs`	GET	Bearer Token	None	List of connected repositories with their configurations.
`/checkov`	POST	Bearer Token	• `content`: string or • `repo_url`: string	`CheckovResponse`: compliance results and risk score.
`/semgrep`	POST	Bearer Token	• `repo_url`: string	`SemgrepResponse`: code vulnerability results.
`/t5-analysis`	POST	Bearer Token	• `description`: string	`T5Response`: AI-generated summary and improvement suggestions.
`/full-scan`	POST	Bearer Token	• `repo_url`: string	Aggregated scan combining Checkov, Semgrep, and T5 outputs.
`/history`	GET	Bearer Token	• `scan_type` (optional)	Array of `ScanHistoryResponse` containing previous results.
`/api/admin-stats`	GET	Bearer Token	None	`AdminStatsResponse`: includes user count, total scans, and most active users.
`/api/config/templates`	GET	Bearer Token	None	Predefined configuration templates for CI/CD security pipelines.
`/reports`	GET	Bearer Token	`filter` (optional)	List of saved reports in JSON format.
`/reports/:id`	GET	Bearer Token	`id`: integer	Detailed report information for a specific scan.
`/admin/users`	GET	Bearer Token	None	List of all registered users (admin privilege required).
`/admin/users/:id`	DELETE	Bearer Token	`id`: integer	`{``message'': ``User deleted successfully''}`

Appendix B

A functional comparison between SafeOps+ and existing IaC platforms is provided in Table B.4.

Table B.4
Functional comparison between SafeOps+ and existing IaC platforms.

Feature/criteria	SafeOps+	Checkov[1]	Snyk IaC[2]	Terralint + tfsec[3]	Kubescape[4]
Identification method	T5 fine-tuned + Gemini AI	Static rules	Static + explainability	Linters (static)	Rego rules + OPA
Configuration fix suggestions	Full rewrite (T5) + Gemini	Partial	Guided suggestions	Alerts only	Fix hints only
CI/CD integration templates	GitHub, GitLab, Jenkins	GitHub, GitLab, CircleCI	GitHub, GitLab, Bitbucket	CLI-based	GitHub, GitLab, ArgoCD
Vulnerability scanning scope	IaC (Terraform, Dockerfile, K8s) + code (Semgrep)	Terraform, K8s, CloudFormation	Terraform, K8s, AWS IAM	Terraform only	K8s (YAML, Helm)
AI-based policy generation	Yes (T5-base)	No	No	No	No
Code Security Scan (SAST)	Semgrep + Gemini	No	Basic SAST	No	No
Custom dashboard	React-based UI	Web UI	Web UI	None	Rich UI
Historic traceability	Full filters (repo, date)	Limited	Audit trails	Logs only	Scan history
Auto pipeline block on critical	Yes	Yes	Yes	Manual	Yes

[1] https://www.checkov.io
[2] https://snyk.io/fr/product/infrastructure-as-code-security/
[3] https://github.com/aquasecurity/tfsec
[4] https://kubescape.io

References

[1] Opdebeeck R, Zerouali A, De Roover C. Control and data flow in security smell detection for infrastructure as code: is it worth the effort?. In: Proceedings of the 20th International Conference on mining software repositories (MSR). IEEE; 2023. p. 259–70. https://doi.org/10.1109/MSR59073.2023.00079

[2] War A, Habib A, Diallo A, Klein J, André TB. Vulnerabilities in infrastructure as code: what, how many, and who?. Empir Softw Eng 2025 empirical study finds IaC scripts plagued by security bugs across OWASP categories. https://doi.org/10.1007/s10664-025-10672-8

[3] Rahman A, Parnin C, Williams L. The seven SINS: security smells in infrastructure as code scripts. In: Proceedings of the 41st International Conference on software Engineering (ICSE). IEEE/ACM; 2019. p. 164–75. https://doi.org/10.1109/ICSE.2019.00030

[4] Fu M, Pasuksmit J, Tantithamthavorn C. AI for devsecops: a landscape and future opportunities. ACM Trans Softw Eng Methodol 2025;34(4). https://doi.org/10.1145/3712190

[5] Nagpal A, Pothineni B, Parthi AG, Maruthavanan D, Banarse AR, Veerapaneni PK, Sankiti SR, Jayaram V. Framework for automating compliance verification in ci/cd pipelines. Int J Comput Sci Inf Technol Res 2024;5(4):17–27.

[6] Port D, Taber B, Emkani P. Investigating effectiveness and compliance to devops policies and practices for managing productivity and quality variability. J Syst Softw 2024. https://doi.org/10.1016/j.jss.2024.112030

[7] Nimmagadda S. Integrating machine learning into the security of containerized workloads. J Comput Sci Technol Stud 2025;7(9):135–42. https://doi.org/10.32996/jcsts.2025.7.9.17

[8] Du X, Liu M, Wang K, Wang H, Liu J, Chen Y, Feng J, Sha C, Peng X, Lou Y. Evaluating large language models in class-level code generation. In: 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE). IEEE/ACM; 2024. p. 81:1–81:13. https://doi.org/10.1145/3639478.3639505

[9] IBM Security. Cost of a data breach report 2023, Tech. rep., IBM Corporation; 2023. https://www.ibm.com/reports/data-breach/2023.

[10] Red Hat. Guide to NIST SP 800-190 compliance in container environments. May 2024, https://www.redhat.com/en/resources/guide-nist-compliance-container-environments-detail [Accessed on Oct 21, 2025].

[11] Snyk Ltd. State of cloud native application security report. 2023 [Accessed on 5 Jan 2025] https://snyk.io/blog/cloud-security-challenges

[12] Wang P, Liu X, Xiao C. CVE-bench: benchmarking LLM-based software engineering agent's ability to repair real-world CVE vulnerabilities. In: Proceedings of the 2025 conference of the North American chapter of the Association for Computational Linguistics: human language Technologies (volume 1: long papers). Association for Computational Linguistics; 2025. p. 4207–24.

[13] Zhang L, Zou Q, Singhal A, Sun X, Liu P. Evaluating large language models for real world vulnerability repair in c/c++ code, Tech. rep., National Institute of Standards and Technology (NIST). Jun 2024. https://www.nist.gov/publications/evaluating-large-language-models-real-world-vulnerability-repair-cc-code.

[14] Dahlmanns M, Sander C, Decker R, Wehrle K. Secrets revealed in container images: an internet-wide study on occurrence and impact. In: Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security (ASIA CCS). ACM; 2023. https://doi.org/10.1145/3579856.3590329

[15] Kon PTJ, Liu J, Qiu Y, Fan W, He T, Lin L, Zhang H, Park OM, Elengikal GS, Kang Y, Chen A, Chowdhury M, Lee M, Wang X. Iac-eval: a code generation benchmark for cloud infrastructure-as-code programs. In: Advances in neural information processing systems 37 (neurips 2024) datasets and benchmarks track; 2024. p. 134488–506.

[16] Buckner M. AI-powered devsecops: navigating automation, risk and compliance in a zero-trust world. May 2025 https://devops.com/ai-powered-devsecops-navigating-automation-risk-and-compliance-in-a-zero-trust-world/ [Accessed on Oct 21, 2025].

[17] OWASP Foundation. LLM and gen AI data security best practices 2025 v1.0, Tech. rep., OWASP Foundation. Apr 2025. https://www.scribd.com/document/849703118/LLM-and-Gen-AI-Data-Security-Best-Practices-2025-v1-0.

[18] Desai R, Nisha TN. Best practices for ensuring security in devops: a case study approach. J Phys Conf Ser 2021;1964(4):042045. https://doi.org/10.1088/1742-6596/1964/4/042045

[19] Kalva R. The evolution of devsecops with AI. Nov 2024 [Accessed on Oct 21, 2025], https://cloudsecurityalliance.org/blog/2024/11/22/the-evolution-of-devsecops-with-ai

[20] Karthick R. A unified framework for devsecops-driven AI applications in multi-cloud environments, Preprints.orgPreprint version; submitted 15 July 2025, posted 17 July 2025 (2025). https://doi.org/10.20944/preprints202507.1486.v1. https://www.preprints.org/manuscript/202507.1486

[21] Garcia J, et al Devsecops: a multivocal literature review. Comput Stand Interfaces 2020;69:103403. https://doi.org/10.1016/j.csi.2019.103403

[22] Kalouptsoglou I, Siavvas M, Ampatzoglou A, Kehagias D, Chatzigeorgiou A. Software vulnerability prediction: a systematic mapping study. Inf Softw Technol 2023;147:107303. https://doi.org/10.1016/j.infsof.2023.107303

[23] Beller M, Gousios G, Zaidman A. Oops, my tests broke the build: an empirical study of Travis CI with Github. PeerJ Comput Sci 2017;3:e127. https://doi.org/10.7717/peerj-cs.127

[24] Lamb C, Zacchiroli S. Reproducible builds: increasing the integrity of software supply chains. IEEE Software 2022;39(2):62–70.

[25] Gajbhiye B, Aggarwal A, Jain S. Automated security testing in devops environments using AI and ML. Int J Res Publ Seminar 2024;15(2):259–66. https://doi.org/10.36676/jrps.v15.i2.1472

[26] Yildiz A, Teo SG, Lou Y, Feng Y, Wang C, Divakaran DM. Benchmarking llms and LLM-based agents in practical vulnerability detection for code repositories. In: Proceedings of the 63rd annual meeting of the Association for Computational Linguistics. Vienna, Austria: Association for Computational Linguistics; 2025. p. 30848–65.

[27] Davidson S, Sun L, Bhasker B, Callot L, Deoras A. Multi-IaC-Eval: Benchmarking Cloud Infrastructure as Code Across Multiple Formats. arXiv preprint arXiv:2509.05303, 2025. https://doi.org/10.48550/arXiv.2509.05303

[28] Lian K, Wang B, Zhang L, Chen L, Wang J, Zhao Z, Yang Y, Lin M, Duan H, Zhao H, Liao S, Guo M, Quan J, Zhong Y, He C, Chen Z, Wu J, Li H, Li Z, Yu J, Li H, Zhang D. A.S.E: A Repository-Level Benchmark for Evaluating Security in AI-Generated Code, arXiv preprint arXiv:2508.18106, 2025. https://doi.org/10.48550/arXiv.2508.18106

[29] GitLab Inc. Gitlab announces AI-powered devsecops platform with Gitlab 16. May 2023 https://about.gitlab.com/press/releases/2023-05-22-gitlab-16-announces-ai-powered-devsecops-platform/ [Accessed on Oct 21, 2025].

[30] Nanjundappa P. Devsecops in 2025: the AI-powered future of security and efficiency. Mar 2025 [Accessed on Oct 21, 2025], https://www.chef.io/blog/devsecops-in-2025-ai-powered-future-security-efficiency

[31] Witschey J, et al Quantifying developers' adoption of security tools. In: Proceedings of the 2015 symposium and bootcamp on the science of security; 2015.