ISSN: 2754-6659

Journal of Artificial Intelligence & Cloud Computing

SCIENTIFIC Research and Community

Review Article

Open Access

Application of AI and ML in the Field of DevSecOps

Khirod Chandra Panda[1]* and Shobhit Agrawal[2]

[1]Asurion Insurance, VA, USA

[2]Visa, WA, USA

ABSTRACT

Security is a paramount concern in DevOps. The adoption of Infrastructure as Code (IaC) has increased the potential impact of even minor flaws, particularly in critical domains like healthcare and maritime applications. Existing solutions typically focus on either Static Application Security Testing (SAST) or run-time behavior analysis. This paper introduces the IaC Scan Runner, an open-source tool developed in Python for inspecting various IaC languages during application design, and LOMOS, a run-time anomaly detection tool. Both tools work together to enhance the security of DevOps processes. In today's rapidly evolving technological landscape, the vulnerability of infrastructure and applications is growing due to a combination of factors. Attackers are becoming more sophisticated, leveraging improved intelligence to exploit weaknesses. At the same time, there is a lack of technical capability in many organizations to effectively secure their systems. This paper explores a dual approach to cybersecurity: static security monitoring through rule matching and the application of self-supervised machine learning. By combining these approaches, organizations can better defend against cyber threats. One area of focus is supply chain resilience and smart logistics, where the integration of these methods is particularly critical. This approach emphasizes a self-learning and self-healing approach, allowing systems to adapt and respond to new threats autonomously. Integrating Artificial Intelligence (AI) and Machine Learning (ML) into DevSecOps practices is essential for improving security, efficiency, and innovation in software development and deployment. This paper delves into strategies and best practices for leveraging AI/ML within the DevSecOps framework. It discusses automated threat detection, predictive analytics for vulnerability management, and intelligent automation for continuous integration and deployment. However, this integration also presents challenges, such as data privacy, algorithm transparency, and ethical implications. The paper addresses these challenges and showcases how organizations can use AI/ML to optimize their DevSecOps pipelines, mitigate security risks, and foster continuous improvement. The adoption of Infrastructure as Code (IaC) has increased the potential impact of even minor flaws, especially in critical domains like healthcare and maritime applications. Existing solutions typically focus on either Static Application Security Testing (SAST) or run-time behavior analysis.

***Corresponding author**

Khirod Chandra Panda, Asurion Insurance, VA, USA.

Received: November 15, 2022; **Accepted:** November 22, 2022; **Published:** November 30, 2022

Keywords: DevOps, DevSecOps, IaC, SAST, DAST, LOMOS, Machine Learning, Natural Language Processing, Self-Supervised Learning

Introduction

DevSecOps stands for Development, Security and Operations. It's the extension of DevOps practices where each team has various roles and responsibilities of software groups when they develop the applications. DevSecOps is the practice of integrating security testing at each phase of the application development process. It covers tools and procedures that motivate collaboration among developers, operation groups, and security specialists to develop efficient and secure software. It provides the cultural transformation that contributes to security with the shared responsibility for everyone developing the software. In the dynamic realm of technology, the fusion of Artificial Intelligence (AI) and Machine Learning (ML) with DevSecOps practices stands out as a pivotal catalyst for bolstering security, efficiency, and innovation in software development and deployment processes. This document explores effective strategies and optimal practices for maximizing the capabilities of AI/ML within the DevSecOps framework. Commencing with an overview of DevSecOps principles and the integral role of AI/ML, the document delves into specific tactics such as automated threat detection, predictive analytics for vulnerability management, and intelligent automation for continuous integration and deployment. Furthermore, it discusses key challenges and considerations in integrating AI/ML into DevSecOps, such as data privacy and ethical implications. Through illuminating case studies and real-world illustrations, the document showcases how organizations can leverage AI/ML technologies to streamline their DevSecOps pipelines, mitigate security risks, and cultivate a culture of ongoing enhancement. By embracing these strategies and adhering to best practices, organizations can harness the full potential of AI/ML to propel innovation, fortify resilience, and enhance agility in their DevSecOps endeavors. DevOps methodology in software engineering aims to automate operations related to development, testing, continuous integration and deployment in alignment with business goals and other aims of the involved stakeholders and organizations [1,2].

In this context, many novel methods, concepts and techniques have emerged, striving to aid the automation of the underlying activities. One of corner stones is the introduction of Infrastructure as Code (IaC) that treats deployment, configuration, and update

instructions similarly as software source code [1]. For that purpose, usually, both human- and machine-readable scripts are leveraged to automatize the underlying operations, while eliminating the need of manual intervention as much as possible. This way, high degree of deployment repeatability and reusability is achieved, saving both the time and reducing the operational costs for the involved parties [1]. Additionally, modifications of IaC scripts for the purpose of application updates during the lifecycle is process prone to various errors and mistakes, e.g., exposing credentials, applying wrong/outdated settings or configuration parameters. Preventing the worst consequences by performing IaC inspection with Static Application Security Testing (SAST) tools covers only a subset of potential issues in design-time. Others can be detected only when applications is already deployed, running in production environment and facing the load of users and potential attacks.

In this application life-cycle stage we can apply Dynamic Application Security Testing (DAST) tools or monitoring the application to detect abnormalities as soon as possible . The paper proposes a proof-of-concept of tools contributing to SAST and complement the DAST approach in the DevSecOps wokrflow. First, we improved the DevSecOps design-time experience by developing a Python-based tool called IaC Scan Runner in order to integrate a variety of static component and security inspection check tools targeting state-of-art IaC languages [3]. In our case, the focus is on Ansible playbook-related case study including sophisticated component and security checks. On the other side, the dynamic/run-time aspects are covered by the proposed approach. For run-time phase we developed a VAT and AI-enabled log inspection tool called LOMOS. The paper is concluded by listing the domains where tools are applied and pointing out the future work.

Background and Related Work

DevOps Phases and Workflow

DevSecOps is differentiated into 5 phases: Plan, Develop, Test, Release and Operations.

The first plan stage includes the three essential security tasks that must be considered to generate and analyze the security essentials, develop threat models and implement a roadmap for success control.

Developing the code and getting peer review is part of the development phase.

Testing the security of the things that are planned and developed is the aspect of the Test phase of the DevSecOps cycle. In testing phase, the Static Application Security Test is conducted.

The software is developed, released, and deployed in the non-production phase and the production circumstances in the release and deployment. Dynamic application security testing, red testing, etc., are performed in release phase.

Once the application is deployed in production, maintaining, and monitoring the application occurs in the operations phase.

DevOps practices have been increasingly applied to software development and the machine learning lifecycle, known as MLOps. Implementing MLOps efficiently is crucial, but there is limited information in academic literature on how to do so effectively. To address this gap, Matsui and Goya propose five essentials. steps for implementing MLOps, serving as a reference guide for those interested in adopting MLOps practices [2,4]. Additionally, Gawre suggests integrating machine learning with DevOps through Continuous Integration/Continuous Deployment (CI/CD) and dynamic hyperparameter changes to achieve increased accuracy without human intervention. This approach is applicable to any type of machine learning model, with a focus on neural networks [5].

Moreschini et al. propose a graphical representation for MLOps, called MLOps, which combines the simplicity of DevOps with circular steps for ML incorporation, creating a self-maintained ML-based development subsystem [5]. Finally, Cankar et al. address the security concerns in DevOps by proposing IaC Scan Runner and LOMOS, tools that provide static analysis and runtime anomaly detection for Infrastructure as Code (IaC) [6].

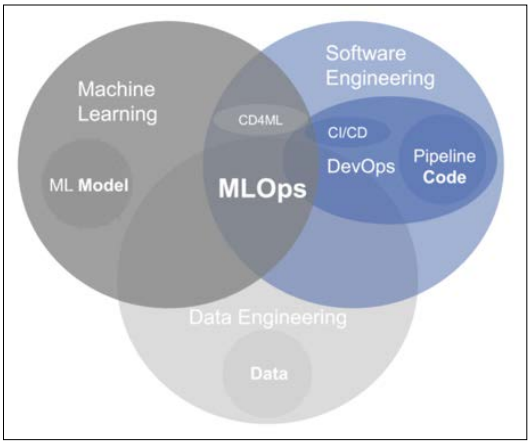


Figure 1: Intersection of Disciplines of the MLOps Paradigm

Approach

In this paper, we adopt an approach which tackles the previously mentioned issues related to security and trustworthiness within the scope of DevOps workflows in both the design and run-time phase, based on DevSecOps philosophy [2]. In the design phase, we rely on (i) a service for static code scanning, integrating many independent tools and in run-time phase, we developed (ii) an NLP-based service for detecting anomalies and therefore potential issues in system logs. Both services are applicable on wide set of IaC related formats and standard and provide a summary of the scans to the final user. Figure 1 depicts the proposed DevSecOps workflow based covering the aspects of both design and run-time trustworthiness, leveraging the proposed (i) and (ii) approaches in synergy with other DevSecOps steps. In the first step, when user has already designed an application, she provides the desired archive containing IaC scripts and submit it for static scanning. Here, user is able to notice if issues exist, and correct the code, accordingly. After user intervention and code correction, the IaC archive is checked once again and deployed in case that no problems were detected. After the successful deployment, when the infrastructure is up and running the services, the infrastructure or application logs are acquired. These logs unveil a lot of potential security issues that is known described by experts.

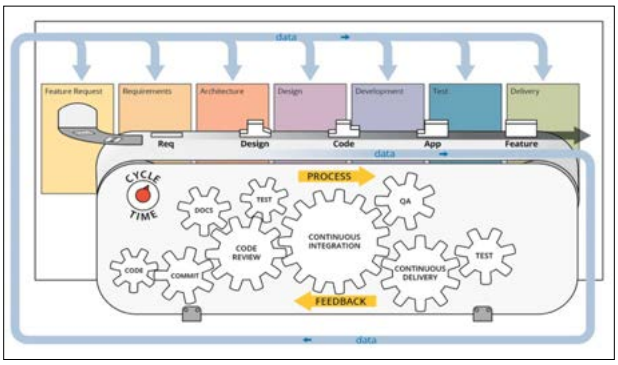
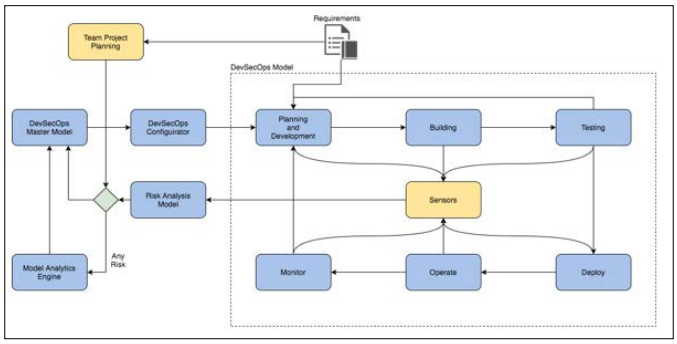


Figure 2: Generic DevSecOps Pipeline that Accommodates all the IEEE Processes [7]

To identify unknown problems and to label potential issues, an additional AI-based analysis service is processing the logs and detecting anomalies, ranking them with an evaluation score, so users can focus only on the parts of history logs that potentially present a threat.



A more complex scenario may present two pipelines that share common characteristics and use the switches to evaluate the quality of the identified ML patterns. Figure 3 above illustrates our example.

Design-Time IaC and Component Inspection

During the design-time service, the user submits an IaC archive for scanning by the IaC Inspector. The IaC Scan Runner then evaluates the user-selected checks with the archive and automatically identifies the compatible checks to perform [8]. Our significant contribution in this process is the development of the Ansible component inspector. Our analysis identified a gap where Ansible IaC code relies on multiple Ansible Collections that offer specific functionality. However, including each collection introduces potential risks, as collections could be outdated or vulnerable. To address this, we developed a tool with the following features:

- **Parameter Checking:** Identifies incorrect configurations and ensures the correct parameters are used, considering their relationships.
- **Best Practices Adoption:** Ensures anti-patterns are avoided.
- **Module Checking:** Identifies name changes and redirects, checks for fully qualified names, and ensures only certified and approved modules are used.
- **Correction Recommendations:** The error assistant guides users through hard-to-detect errors, with errors and warnings distinguished by colors.

This framework, which combines IaC and component checks, is implemented in Python and offers both a web-based REST interface based on FastAPI and a command-line interface (CLI) for easier integration with DevOps pipelines. The OpenAPI specification can be used with the SwaggerUI graphical interface to interact with the deployed service. The framework covers a variety of check tools, from basic linters (e.g., pylint for Python, YAMLlint for TOSCA and Ansible YAML files, Hadolint for Docker files) to more advanced security-related tools (e.g., Terrascan and tfsec for Terraform, Steampunk Spotter for Ansible, xOpera TOSCA parser for TOSCA YAML). Additionally, informational tools that provide IaC archive-related statistics, such as cloc, are included. The service, named IaC Scan Runner, is open-source software available on GitHub, aimed at consolidating various IaC-related static script scanning tools into a unified web-based API. To simplify its use, the component inspection tool is integrated into the IaC Scan Runner and can be initiated along with other supported scans. A professional version of the component inspection tool is also available separately under the commercial name Steampunk Spotter. In addition to static IaC analysis, the tool assists in automation code writing and offers recommendations for Ansible Playbooks. It can be easily integrated into GitHub CI/CD workflows using the command-line interface.

LOMOS

A log analysis tool that leverages AI-powered log analysis to enhance dynamic security. This tool automatically analyzes system or application logs, providing valuable insights into the current and past status of monitored assets. Based on LogBERT and implementing self-supervised NLP methods like Masked Language Modelling, LOMOS uses deep learning techniques to consider various log aspects, such as message semantics and sequential information [8]. This AI-based approach can analyze messages automatically based on historical log records, considering factors like severity and occurrence frequency. It enables unsupervised differentiation between normal flow and abnormalities, sending notifications to the user when unexpected behavior or incidents occur. Traditional log monitoring solutions are limited to rule based (manual) analysis of time series data. In contrast, LOMOS makes use of state-of-the-art Natural Language Processing architectures to model log streams and capture their normal operating conditions. This enables the implementation of a monitoring system that does not depend on any manually defined rules or human intervention, but that relies on that behavioral model to automatically detect deviations that would represent any kind of abnormal situations, including potential security threats. Our approach automatically analyses system or application logs and provides valuable insights regarding the current and past status of the monitored assets. The technology uses deep learning techniques, to compute anomaly score on sequences of log templates, applying NLP models to the IDs defined for the templates. Without any manual preprocessing of raw logs from unstructured data, LOMOS aims at learning patterns in logs and identifying anomalous behavior. To that aim, LOMOS tries to get some structure by identifying what are the log templates that would match the log. The algorithm behind this tries to identify parameters (ids, services, ports, etc.) transforming the unstructured logs to structured log templates broken down according to the tree structure (with wildcards for parameters changing from log to log). Then, LOMOS observes the sequence of templates trying to learn what is the normal behavior and provides an anomaly score (including aggregation and specific counts) that should be low for normal logs.

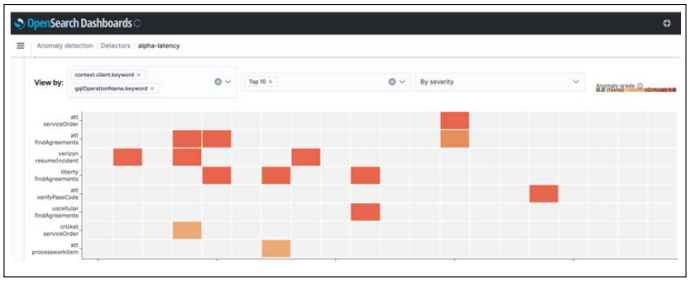


Figure 4: LOMOS Dashboard (OpenSearch)

In figure 4 above, we can see the LOMOS dashboard showcasing the dated identified threats, their ranking and the call for action (orange/red). There is a training period and a monitoring period ensuring a rapid response, based on the extension of LogBERT algorithms, using Drain for log template parsing, and OpenSearch (opensearch.org) or Grafana (grafana.com) to setup alerts and send them to specific services [8]. Real-time anomaly detection would require incremental learning (learn sample by sample and get immediate updates to the ML model) but in this case it does not make sense as we cannot update the model at every log.

Conclusion

The integration of static rule matching with dynamic self-supervised machine learning in log analysis presents a promising approach for detecting anomalies. By combining open-source security monitoring technology with machine learning-based anomaly detection, this approach shows potential for enhancing supply chain resilience and improving DevSecOps frameworks. A key research question is how anomalies identified through this process can be transformed into Security Information and Event Management (SIEM) rules, with the LOMOS tool playing a crucial role in this transformation. Integrating LOMOS results into the SIEM could enhance its ability to adapt to new threats and provide valuable insights for more accurate event creation.

In the realm of DevOps, this paper demonstrates the importance of integrating various tools for static code analysis and anomaly detection to ensure trustworthiness in both design and run-time aspects. The approach focuses on leveraging automatic code correction in static analysis, complemented by a dynamic approach based on machine learning methods. Future work includes automating procedures for extending design-time analysis tools with new Infrastructure as Code (IaC) checks and further developing run-time security approaches for supply chain cybersecurity and failure detection. This approach is also planned for incorporation into platform engineering, promoting collaboration and automated infrastructure management.

While this paper emphasizes incremental accuracy improvements, it also highlights the potential for creating comprehensive end-to-end automated MLOps pipelines. By integrating feature engineering and feed-forwarding techniques, these pipelines can optimize time utilization for data scientists and ML engineers, allowing them to focus on innovation and research. This approach streamlines the deployment process for ML models, ensuring efficient allocation of company resources and enabling analysts to concentrate on business objectives rather than technological uncertainties.

References

1. Juncal Alonso, Christophe Joubert, Leire Orue-Echevarria, Matteo Pradella, Daniel Vladušic (2021) Programming trustworthy Infrastructure As Code in a Secure Framework. In First SWForum workshop on Trustworthy Software and Open Source 1-8.
2. Garg S, Pundir P, Rathee G, Gupta PK, Garg S, et al. (2021) On Continuous Integration/Continuous Delivery for automated deployment of machine learning models using MLOps. In: 2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE). Laguna Hills, USA: IEEE 25-28.
3. He P, Zhu J, Zheng Z, Lyu MR (2017) Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE international conference on web services (ICWS) 33-40.
4. Mäkinen S, Skogström H, Laaksonen E, Mikkonen T (2021) Who needs MLOps: What data scientists seek to accomplish and how can MLOps help? In: 2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN). Madrid, Spain: IEEE 109-112.
5. Gupta S, Bhatia M, Memoria M, Manani P (2022) Prevalence of GitOps, DevOps in Fast CI/CD Cycles. In: 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON). Faridabad, India: IEEE 589-596.
6. 12207-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes. IEEE Xplore https://ieeexplore.ieee.org/document/8100771.
7. Nenad Petrovic, Matija Cankar, Anže Luzar (2022) Automated Approach to IaC Code Inspection Using Python-Based DevSecOps Tool. 2022 30th Telecommunications Forum (TELFOR) 1-4.
8. Haixuan Guo, Shuhan Yuan, Xintao Wu (2021) LogBERT: Log Anomaly Detection via BERT. Arxiv https://arxiv.org/abs/2103.04475.

Copyright: ©2022 Khirod Chandra Panda. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.