Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps

Matija Cankar
matija.cankar@xlab.si
XLAB d.o.o.
Ljubljana, Slovenia

Nenad Petrović
nenad.petrovic@xlab.si
XLAB d.o.o.
Ljubljana, Slovenia
University of Niš
Niš, Serbia

Joao Pita Costa
Aleš Černivec
Jan Antić
Tomaž Martinčič
Dejan Štepec
XLAB d.o.o.
Ljubljana, Slovenia

ABSTRACT

Security represents one of the crucial concerns when it comes to DevOps methodology-empowered software development and service delivery process. Considering the adoption of Infrastructure as Code (IaC), even minor flaws could potentially cause fatal consequences, especially in sensitive domains such as healthcare and maritime applications. However, most of the existing solutions tackle either Static Application Security Testing (SAST) or run-time behavior analysis distinctly. In this paper, we propose a) IaC Scan Runner, an open-source solution developed in Python for inspecting a variety of state-of-the-art IaC languages in application design time and b) the run time anomaly detection tool called LOMOS. Both tools work in synergy and provide a valuable contribution to a DevSecOps tool set. The proposed approach is demonstrated and their results will be demonstrated on various case studies showcasing the capabilities of static analysis tool IaC Scan Runner combined with LOMOS – log analysis artificial intelligence-enabled framework.

CCS CONCEPTS

• **Security and privacy → Software security engineering**.

KEYWORDS

DevOps, DevSecOps, IaC, SAST, DAST, Machine Learning, Natural Language Processing, Self-supervised Learning

ACM Reference Format:
Matija Cankar, Nenad Petrović, Joao Pita Costa, Aleš Černivec, Jan Antić, Tomaž Martinčič, and Dejan Štepec. 2023. Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3578245.3584943

1 INTRODUCTION

DevOps methodology in the area of software engineering targets to enable the automatization of operations related to development, testing, continuous integration and deployment in alignment with business goals and other aims of the involved stakeholders and organizations [2], [1]. 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) [1] that treats deployment, configuration and update instructions similarly as software source code. For that purpose, usually, both human- and machine-readable scripts are leveraged in order 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 [2].

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 [11] in order to integrate a variety of static component and security inspection check tools targeting state-of-art IaC languages. 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.

2 BACKGROUND AND RELATED WORK

2.1 DevSecOps workflow

Life-cycle management of the application from the configuration and deployment phase to the daily repetitive updates and upgrades has been recognised as a Dev(Sec)Ops workflow [2] containing

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3584943

multiple stages where different roles of experts and technicians need to collaborate. The stages can be divided in *design phase* – including Plan, Create, Verify and Package – and *run-time phase* – including Release, Configure and Monitor. Phases are meant to be repetitive and focused on special tasks. Considering the possibility of security and other issues in design phase, verification of IaC script trustworthiness is applied as a measure to tackle them. On the other side, in run-time phase, monitoring of real-time behavior of a system is performed in order to detected deviations from normal operation. In this paper, our focus is on two phases aiming to ensure trustworthiness of the application by improving: *a) static verification and security analysis before the deployment* in design phase and *b) adoption of machine learning-based approach to anomalies detection within logs* in run-time, on the other side.

2.2 Security in design phase of DevSecOps

Before the deployment, still in the design phase of the application, the verification stage checks consistency and inspects for vulnerabilities by applying SAST tools. Our goal is to detect known vulnerabilities, often caused by improper configuration, parameter values, type errors and non-compliance with common good practices and particular specification language standards. However, it is quite often the case that IaC-based deployment depends on multiple components, provided by either community or specific organizations that work on IaC languages and standards. Therefore, it is also highly relevant to take into account the previously mentioned concerns and apply them to IaC-related libraries, templates and collections (as for Ansible). For that reason, additional steps are be performed such as verification of dependencies or usage of outdated, vulnerable libraries, which is referred to as component inspection within the scope of this paper. The component inspection can find new vulnerabilities even if the IaC has not changed, therefore this task is continuously repeated also after the application is running. The available of-the-shelf solutions – also called checks – can base on the expert knowledge integrated in the tool [6] or rely on machine learning approaches [4].

Some of the notable existing SAST solutions considering component and security inspection are the following: Mega-Linter[1] - open-Source, offers quality and consistency analysis checks for wide set of languages covered, outputs detailed reports and support auto-fixing; Super-linter[2] - GitHub-integrated workflow that represents a combination of multiple linter tools; Snyk[3] - continuous component checking of dependencies covering project build tools such as Maven for Java and npm for Node.js. Moreover, Open Web Application Security Project (OWASP)[4] holds an extensive list of open source and commercial SAST tools.

The service we are proposing combines code scans and also component inspections, which means that also holds the knowledge of potential issues that can materialise when a specific component is in use. Beside existing scans we focused on implementing component scans which has not been yet covered by the community.

[1]https://megalinter.io/latest/, accessed on 18 January 2023
[2]https://github.com/github/super-linter, accessed on 18 January 2023
[3]https://snyk.io/, accessed on 18 January 2023
[4]https://owasp.org/www-community/Source_Code_Analysis_Tools/, (January 2023)

2.3 Security in run-time phase of DevSecOps

Once the deployment defined by IaC scripts is done, applications and services go live. While they are up and running, during their usage life-cycle phase, various events (such as malicious behavior, attacks and anomalies) can occur and have impact on many aspects related to their availability, performance, data safety and overall integrity [2]. A set of methods aiming to detect such occasions is referred to as Dynamic Application Security Testing (DAST).

For this purpose, the monitoring components record various messages, errors and info about the state changes, generating logs as output. Usually, log analysis methods are applied in order to discover events, which could affect the system, such as security treats or failures. In order to achieve this goal, various tools and approaches are used, as some of significant examples are: WALASAM [8] - web server log analysis using data mining methods based on classification and clustering; nfer [9] - rule-based event-stream abstraction and processing; CMS-NLP [10] - NLP-based technique for log analysis aiming to reduce the operational workload and delays within a CMS solution. Other prominent methodologies in the state-of-the-art of anomaly detection on logs use BERT-based pre-trained methods [3] and the Masked Language Model (MLM) in combination with BERT[12], as well as deep learning based on auto-encoder networks [5].

Our approach leveraged an existing machine learning approach, complementing a traditional DAST methodology, to log monitoring relying on deep learning techniques designed for Natural Language Processing (NLP), which has been already approved by enabling robust distinction between normal system operation and abnormalities [10].

3 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 as presented in Section 2.3. 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. 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.

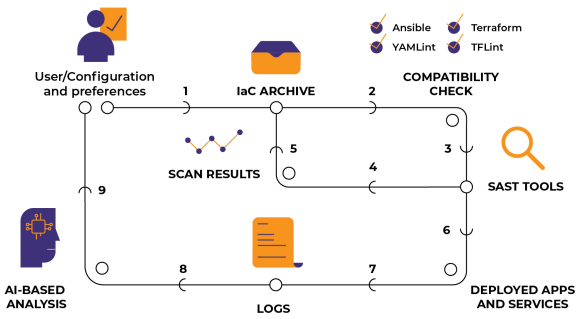

Figure 1: Workflow enabling design and run-time trustworthiness in DevSecOps: 1 – upload IaC; 2 – archive scan 3 - list of compatible check tools 4 – generate HTML summary and persist results 5- correcting with respect to reported problems 6 – IaC deployment 7– run-time events and logging 8 – log analysis 9 – event detection and alerts

4 THE SERVICES OF INSPECTION

We developed a design-time and run-time services for security inspection. A design-time can do a multi-pass over IaC by initiating known open-source and proprietary IaC scan tools together with our own implemented inspection services for deep component check of Ansible IaC. The design service resulted in two parts, IaC Inspector and IaC Component inspector that we combined in the Restful Service. For the run-time services an AI powered log inspection tool, called LOMOS, is developed.

4.1 Design-time IaC and component inspection

The design-time service we developed works as presented in Figure 1 and covers steps 1-6 as mentioned in previous sections. The user provides the IaC archive that is about to be scanned by the IaC Inspector. The IaC Scan runner analyzes the user selected checks together with the archive and automatically recognize the compatible checks to performed. In this step our crucial contribution is the development of the Ansible component inspector. Our gap analysis revealed that in case of Ansible, IaC code relies on multiple Ansible Collections that provide specific functionality. However, inclusion of each collection presents new potential risks, as collections could be outdated and/or vulnerable. This led us to develop a tool with the following set of features: 1) parameter checking - wrong configuration identification, making sure that the correct parameters are used, considering their relationships 2) best practices adoption - ensures that anti-patterns are avoided 3) module checking - identifies name changes and redirects, checks for fully qualified names, and ensure we are using only certified and approved modules 4) correction recommendations - error assistant will guide us through the hard-to-catch errors, while errors and warnings can be distinguished by colors.

This framework combining IaC and component check is implemented in Python programming language and offers both web-based REST interface relying on FastAPI and command-line interface (CLI) for easier integration with DevOps pipelines. The OpenAPI specification[5] can be used with SwaggerUI graphical interface to interact with deployed service. A variety of check tools is covered - from basic linters (pylint – for Pythom, YAMLlint – for TOSCA and Ansible YAML files, Hadolint – for Docker files) to more sophisticated security-related tools (such as Terrascan and tfsec for Terraform; Steampunk Spotter for Ansible; xOpera TOSCA parser for TOSCA YAML). Apart from that, informational tools that provide IaC archive-related statistics are included as well, such as cloc. The list of currently supported static analysis tools for specific IaC-related file types can be found here[6].

We named the presented service as IaC Scan Runner [11]. It is an open-source software, publicly available on GitHub[7], with a goal to aggregate various types of IaC-related static script scanning tools put together into unified web-based API. To ease the usage the component inspection tool is integrated in the IaC Scan Runner and can be initiated among other supported scans. The professional version of component inspection tool is available also separately under the commercial name Steampunk Spotter[8]. Beside the static IaC analysis provides an assisted automation code writing and offers recommendations for Ansible Playbooks. Tool can be simply integrated within GitHub CI/CD workflows using command-line interface.

4.2 Design-time IaC Scan results

When the IaC is processed by all scans the outputs are ranked and displayed to a user. The output lists result summary in four levels: 1) Passed – the IaC is clear with no problems, 2) Error – issues found 3) Warnings and 4) Not performed – scan not performed as there was no associated file. To ease the managing all scan results for the user, the ranks define prioritisation list, displaying checks with more serious issues on top and less important issues later.

The list of the scans is limited to the ones that current version of IaC Scan Runner supports, however, we are aware that DevOps paradigm is evolving and new scans will appear in the future, to cover new vulnerabilities. To make the tool more future-proof, we prepared the instructions on our GitHub location[9] that can be used to add any new scan in the IaC Scan Runner.

4.3 Run-time service inspection

The run-time inspection is focused on vulnerability assessment tools, including the continuous checking of the system safety and system information management systems that check system historic logs. In the following sections we will present the VAT and LOMOS approach.

4.3.1 VAT - static vulnerability assessment from rule matching. In complex systems, verification is not only a matter concerning the

[5]https://xlab-si.github.io/iac-scanner-docs/
[6]https://xlab-si.github.io/iac-scanner-docs/02-runner.html
[7]https://github.com/xlab-si/iac-scan-runner, accessed on 13 January 2023
[8]https://steampunk.si/blog/how-to-use-steampunk-spotter-cli-to-audit-your-playbook/, accessed on 13 January 2023
[9]https://github.com/xlab-si/iac-scan-runner#readme, accessed on 20 January 2023

pre-operation steps, but it also encompasses the operation of the system. In particular, it is of paramount importance to make sure that a system undergoes through a continuous run-time verification for what concerns any security violation. Such violations can be recognized by monitoring of various security metrics (e.g., file integrity, network configuration changes, usage of software reported as vulnerable, malware detection) and integrated with the system's monitoring component to recognize violations of defined security policies and alert DevSecOps teams to address and eliminate threats as fast as possible. Through security verification and threat detection on multiple levels, the PIACERE framework empowers deployed applications by helping to prevent abuse and leakage of data.

We have developed a toolset able to verify any security violation at run-time, feeding self-learning and self-healing mechanisms. The monitoring system is capable of detecting security-related events and incidents in the deployed application's environment[10]. It is (to the extent possible) deployable automatically and notifies users about security alerts. The system is then able to automatically deploy security monitoring agents, integrated into the monitoring mechanisms and notify about security threats according to the policies.

4.3.2 LOMOS - dynamic security with AI-powered log analysis. To complement and enhance the static analysis with VAT as discussed above in Section 4.3.1, we have developed a log analysis tool - LOMOS - that provides the automatic analyses of system or application logs and provides valuable insights regarding the current and past status of the monitored assets. It is based on LogBERT [7] and implements self-supervised NLP methods, such as Masked Language Modelling, relying on deep learning techniques and taking into account various aspects relevant to logs, such as semantics of their messages and sequential information. The adopted AI-based approach is able to perform automatic message analysis based on historical log records, taking into account factors, such as their severity and occurrence frequency. It allows for unsupervised distinction between the normal flow and abnormalities, while corresponding notification are sent to the user when unexpected behavior or incident happens (see the workflow in Figure 2).

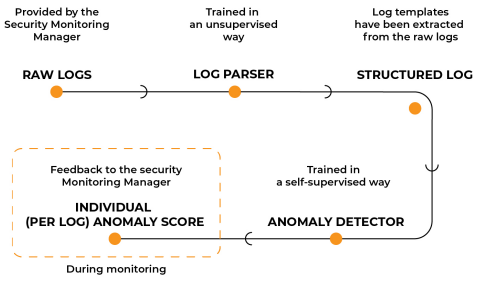

Figure 2: Workflow of log anomaly detection with LOMOS

Traditional log monitoring solutions, such as the one discussed in Section 4.3.1, are limited to rule-based (manual) analysis of time series data. In contrast, LOMOS makes use of state-of-the-art NLP methods in order 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 rather on behavioral model which is able to detect deviations that would represent any kind of abnormal situations, including potential security threats. The following relevant use cases are covered: (i) the insightful monitoring of application logs; (ii) the automatized alert system for any deviation from normal execution workflows; (iii) the easy root cause analysis via integration of logs from several components; and (iv) the identification of specific events, such as security incidents, performance drop and system failures. Additionally, LOMOS can be integrated with other systems as Grafana UX interface, Slack alerting system, Security information and event management system (SIEM) and extended and detection response tools (XDR).

5 USE-CASES AND APPLICATIONS

Table 1: Fields of security service application, IaC Scan Runner (IaC and Component check), VAT and LOMOS

	IaC check	IaC Component check	VAT	LOMOS
Public administration	✓	✓		✓
Transport infra	✓	✓		
Critical infrastructure	✓	✓		
Supply-chain Security			✓	✓
Connected Cars			✓	✓
Smart Agriculture			✓	✓
Health Devices				✓

The presented services have been already applied on the domains where users are evaluating them. The overall mapping between the domains of interest and presented tools is shown in Table 1. The Iac Scan Runner is currently evaluated by the Slovenian Public Administration Ministry that will use the tool to inspect their production services deployed on the state-internal network in the sense of IaC security and component – Ansible collection – verification. In the area of telecommunication infrastructure, the Ericsson will evaluate the usage of the IaC Scan Runner on the IaC code for network configuration and deploying edge applications. The Prodevelop, a maritime critical infrastructure manager, will evaluate the IaC Scan Runner for IaC supporting the migration of their application from private to the public cloud. All three mentioned domains are using the SAST tools through the PIACERE IDE developed over the Eclipse IDE framework. Our Ansible component check tool called Steampunk Spotter, which is included in IaC Scan Runner scan set, is also available as a commercial software for DevOps developers[11]. The overall chain of tools used for implementation of our approach is depicted in Figure 3.

On the one side, the VAT and LOMOS applications are in the evaluation in food chain producers, connected car producers and smart

[10] https://medina-project.eu/blog/tools-and-techniques-collecting-evidence-technical-and-organisational-measures, accessed on 20 January 2023

[11] https://steampunk.si/spotter/

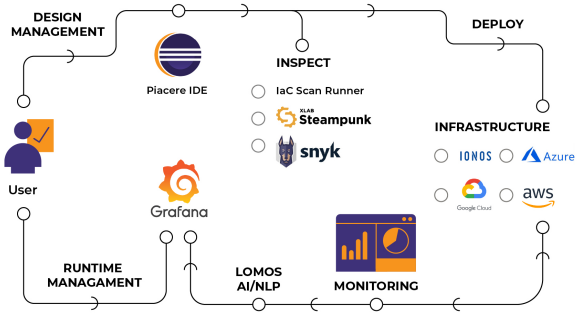

Figure 3: DevSecOps toolchain behind our approach: green - design-time, violet - deployment, blue - events, red - run-time

factory delivery chains – we address these fields with a common name *Supply-chain security* – where users use them for incident detection and prediction, as well as failure detection and prediction or root cause analysis, benefiting the efficiency of operations and reduced response times, while optimizing the usage of resources [12]. On the other side we are also using this secure approach in the context critical infrastructure as railway infrastructure management and energy provision to build the security Layer into the novel ICOS[13] platform, capturing and optimizing log streams, with integrated alerting capabilities. The LOMOS will be also applied in new initiatives in context of energy consumption prediction in case of extraordinary events, like pandemics, where historical daily, weekly and monthly cycles cannot be used as a reference. This topic is yet to be investigated in SUNRISE[14] project. In the future we also plan to evaluate LOMOS in health domain, where IT infrastructure is comprised of a large number of IoT devices that gather sensitive data. This is a topic of CYLCOMED project[15].

Finally, presented tools are not used only by the technicians that applying security measures to the applications but also by the auditors and standardisation bodies to evaluate the application compliance to the standard. This complex and though task is the goal of MEDINA[16] project, which uses presented VAT tools as an automated compliance check that eases the auditors work.

6 CONCLUSION AND FUTURE WORK

This paper has shown how it is possible to cover both the design and run-time aspects of trustworthiness within DevOps workflows, relying on integration of various tools for static code analysis and anomaly detection, on the other side. For static analysis, we focus on the Ansible case study, leveraging the capabilities of automatic code correction when issues were detected. When it comes to run-time analysis, we see the value of the complementarity of a static approach, based on rule matching using deployed agents, and a dynamic approach based on machine learning methods on text.

Regarding the SAST-related future works, we will be working an automated procedure which enables to extend the existing design-time analysis tool with new IaC checks, which is highly beneficial when it comes to saving time and effort. We are also further developing our DAST approach, VAT, in relation to a Wazuh-based SIEM in the context of the management of trustworthy evidence on various levels, ensuring the trustworthiness of evidence across the lifecycle[17]. The run-time security approach discussed in this paper is also being applied and further developed in the context of cybersecurity of supply chains and failure detection and prediction as discussed in Section 5. Finally, it is planned to incorporate the proposed approach in context of platform engineering[18] - a novel paradigm fusing software development, security and operations, promoting collaboration aided by shared platforms that provide self-service capabilities and automated infrastructure management.

7 ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 and Horizon EU research and innovation programmes under Grant Agreements No. 101000162 (PIACERE), 952644 (FISHY), 952633 (MEDINA), 101070177 (ICOS), 101073821 (SUNRISE) and 101095542 (CYLCOMED)

REFERENCES

[1] Juncal Alonso, Christophe Joubert, Leire Orue-Echevarria, Matteo Pradella, and Daniel Vladušic. 2021. Programming trustworthy Infrastructure As Code in a Secure Framework. In *First SWForum workshop on Trustworthy Software and Open Source 2021*. 1–8.
[2] Juncal Alonso, Radosław Piliszek, and Matija Cankar. 2023. Embracing IaC Through the DevSecOps Philosophy: Concepts, Challenges, and a Reference Framework. *IEEE Software* 40, 1 (2023), 56–62. https://doi.org/10.1109/MS.2022.3212194
[3] Song Chen and Hai Liao. 2022. BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. *Applied Artificial Intelligence* 36, 1 (2022), 2145642.
[4] Dario Di Nucci, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Andrea De Lucia. 2018. Detecting code smells using machine learning techniques: Are we there yet?. In *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 612–621. https://doi.org/10.1109/SANER.2018.8330266
[5] Amir FarzadT and Aaron Gulliver. 2020. Unsupervised log message anomaly detection. *ICT Express* 6, 3 (2020), 229–237.
[6] Kerim Goztepe. 2012. Designing Fuzzy Rule Based Expert System for Cyber Security. *International Journal of Information Security Science* 1 (01 2012), 13–19.
[7] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log Anomaly Detection via BERT. https://doi.org/10.48550/ARXIV.2103.04475
[8] Jingquan Jin and Xin Lin. 2022. Web Log Analysis and Security Assessment Method Based on Data Mining. *Computational Intelligence and Neuroscience* 2022 (08 2022), 1–9. https://doi.org/10.1155/2022/8485014
[9] Sean Kauffman. 2022. Log Analysis and System Monitoring with nfer. *Science of Computer Programming* 225 (11 2022), 102909. https://doi.org/10.1016/j.scico.2022.102909
[10] Lukas Layer, Daniel Abercrombie, Hamed Bakhshiansohi, Jennifer Adelman-McCarthy, Sharad Agarwal, Andres Hernandez, Weinan Si, and Jean-Roch Vlimant. 2020. Automatic log analysis with NLP for the CMS workflow handling. *EPJ Web of Conferences* 245 (01 2020), 03006. https://doi.org/10.1051/epjconf/202024503006
[11] Nenad Petrovic, Matija Cankar, and Anže Luzar. 2022. Automated Approach to IaC Code Inspection Using Python-Based DevSecOps Tool. 1–4. https://doi.org/10.1109/TELFOR56187.2022.9983681
[12] Jina Kim Yukyung Lee and Pilsung Kang. 2021. LAnoBERT: System Log Anomaly Detection based on BERT Masked Language Model. *arXiv preprint* 211109564 (2021).

[12] https://fishy-project.eu/blog/importance-early-detection-vulnerabilities-and-attacks-healthy-supply-chain
[13] https://www.icos-project.eu/
[14] https://sunrise-europe.eu/
[15] https://www.cylcomed.eu/
[16] https://medina-project.eu/
[17] https://medina-project.eu/blog/tools-and-techniques-collecting-evidence-technical-and-organisational-measures, accessed on 20 January 2023
[18] https://www.honeycomb.io/blog/future-ops-platform-engineering, accessed on 20 January 2023