

IDSLab: A Low-Code Platform for End-to-End IDS Dataset Construction and Experimentation

Reda Morsli

École de Technologie Supérieure, University of Quebec
Montreal, Canada
reda.morsli.1@etsmtl.net

Hakima Ould-Slimane

University of Quebec at Trois-Rivières
Trois-Rivières, Canada
hakima.ould-slimane@uqtr.ca

Nadjia Kara

École de Technologie Supérieure, University of Quebec
Montreal, Canada
nadjia.kara@etsmtl.ca

Laaziz Lahlou

École de Technologie Supérieure, University of Quebec
Montreal, Canada
laaziz.lahlou@etsmtl.ca

Abstract

Developing machine learning-based intrusion detection systems (IDS) remains a fragmented and engineering-intensive process, often relying on ad-hoc pipelines for data collection, preprocessing, labeling, and experimentation. Moreover, most existing IDS datasets focus primarily on network-level data, despite growing evidence that combining heterogeneous sources can improve detection robustness. We present IDSLab, a low-code experimental platform designed to support data-centric IDS research and prototyping. IDSLab integrates environment management, adversary simulation, scalable multi-source data collection, and graph-based dataset construction within a single extensible framework, exposed through an interactive graphical interface. We describe the architecture and implementation of IDSLab and demonstrate its utility through an end-to-end use case involving dataset construction and ML-based DoS detection in a Kubernetes environment.

Keywords

Intrusion Detection Systems (IDS), Dataset Construction, Data Collection, Machine Learning

ACM Reference Format:

Reda Morsli, Nadjia Kara, Hakima Ould-Slimane, and Laaziz Lahlou. 2026. IDSLab: A Low-Code Platform for End-to-End IDS Dataset Construction and Experimentation. In *34th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE Companion '26)*, July 5–9, 2026, Montreal, QC, Canada. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3803437.3806411>

1 Introduction

Intrusion Detection Systems (IDS) play a critical role in protecting modern software systems against increasingly sophisticated cyber threats. With the widespread adoption of cloud computing, containerized microservices, and distributed infrastructures, the attack surface of applications has grown significantly, making security monitoring and anomaly detection more complex than ever [8, 13]. As a result, machine learning-based IDS approaches have

gained substantial attention due to their ability to model complex behaviors and adapt to evolving attack patterns [1, 20].

Recent studies have shown that leveraging *heterogeneous data sources*—such as network traffic, system calls, host metrics, and application logs—can significantly improve detection accuracy and robustness [27, 21, 9]. However, most datasets used in IDS research remain strongly biased toward network-level data, often relying exclusively on packet traces or flow features [5]. This narrow focus does not reflect the multi-layer observability signals used by modern IDS in production environments, limiting the realism and generalizability of experimental results [18].

Beyond data limitations, developing ML-based IDS remains a highly fragmented and engineering-intensive process. Researchers and practitioners must typically assemble custom pipelines for environment setup, data collection, labeling, feature extraction, dataset construction, model training, and deployment. These pipelines are often implemented using scripts, configuration files, and notebooks, making them brittle, difficult to maintain, and hard to reproduce [4, 23]. More importantly, this complexity creates a high entry barrier, preventing rapid experimentation and discouraging systematic exploration of design alternatives.

Another major challenge is the lack of *low-code* solutions that allow users to design, modify, and reproduce IDS experiments without extensive software engineering effort. Existing frameworks usually expose low-level APIs or require significant manual scripting, which hinders accessibility and limits adoption by non-specialist users. As a result, many research prototypes remain tightly coupled to specific environments, datasets, or attack scenarios, reducing transparency and comparability [2].

To address these challenges, we present IDSLab, a low-code experimental platform for building, analyzing, and iteratively refining ML-based intrusion detection systems. IDSLab focuses on the data-centric stages of the IDS lifecycle, including environment integration, adversary simulation, scalable and heterogeneous data collection, and graph-based dataset construction. These capabilities are exposed through an interactive graphical interface, allowing users to configure experiments, define preprocessing pipelines, and generate datasets without writing code. The system is designed to be extensible, with architectural support for future integration of model training and deployment components.

In this paper, we describe the design and implementation of IDSLab and demonstrate its practical value through an end-to-end



This work is licensed under a Creative Commons Attribution 4.0 International License.
FSE Companion '26, Montreal, QC, Canada
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2636-1/2026/07
<https://doi.org/10.1145/3803437.3806411>

use case involving dataset construction and ML-based DoS detection. Rather than focusing on achieving state-of-the-art detection performance, our goal is to showcase how IDSLab lowers the barrier to IDS experimentation, improves reproducibility, and enables systematic, data-diverse exploration of IDS design choices.

2 IDSLab

2.1 ML-based IDS Development Process

To contextualize the role of IDSLab, we summarize the typical development workflow of machine learning–based intrusion detection systems (IDS). This workflow is inherently iterative, as models must continuously adapt to evolving environments, workloads, and attack strategies [36, 20]. We conceptualize this process as a five-layer cycle (Figure 1), spanning from data generation to deployment and feedback.

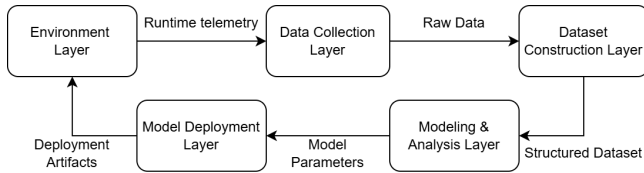


Figure 1: ML-based intrusion detection development cycle.

The cycle starts at the *environment layer*, which represents the operational context where security-relevant data is produced, such as cloud platforms, IoT ecosystems, or endpoint systems. The *data collection layer* ensures reliable, scalable ingestion of heterogeneous and potentially high-volume of telemetry streams, while routing them to the appropriate processing or storage services.

Next, the *dataset construction layer* transforms raw inputs into structured datasets through cleaning, labeling, aggregation, and feature extraction [1]. Transformed data can be archived, prepared for offline training, or forwarded to real-time analytics components. The *model training and evaluation layer* then extracts knowledge from these datasets using various learning paradigms, including supervised, unsupervised, and hybrid approaches [20]. For instance, anomaly-based IDS model benign behavior to detect deviations, whereas signature-based systems learn known attack patterns.

Finally, the *model deployment layer* closes the loop by integrating trained models into the target environment for real-time inference or experimental validation. This layer generates platform-specific artifacts and ensures consistency between training and inference pipelines [18]. Continuous deployment and monitoring enable feedback-driven refinement, completing the iterative IDS development cycle.

2.2 IDSLab Architecture & Implementation

IDSLab is designed as a cloud-native system following a multi-tier microservice architecture, illustrated in Figure 2. This design enables modularity, extensibility, and scalability, which are essential for supporting diverse experimental workflows in intrusion detection research [28]. In the following, we describe the main

architectural tiers and highlight key implementation choices. Additional technical details are available in the public repository linked in the availability section.

Client Tier. The client tier provides the primary interface between users and the system. It is designed to support multiple interaction modalities, including graphical applications, command-line interfaces, software development kits, and AI-agent clients based on the Model Context Protocol (MCP) [31]. Each client issues high-level commands that are processed by the backend.

We implemented a cross-platform graphical client using the Godot Engine [17], chosen for its flexible UI toolkit and native support for interactive workflows. The client communicates with the backend via WebSocket, enabling real-time feedback on long-running operations.

Backend Tier. The backend tier orchestrates the execution of client commands and coordinates the system’s internal components. As shown in Figure 2, incoming requests are received by an API server and dispatched to dedicated controllers, each responsible for a specific functional domain (e.g., data management, environment interaction, or pipeline execution). Controllers interact with storage services and environment interfaces to persist data, retrieve artifacts, or trigger actions in the target environment.

Most backend components are implemented in Python. DuckDB [11] and MinIO [22] are used for structured data storage and object storage, respectively. In its current version, IDSLab supports Kubernetes-based environments. Integration with Kubernetes is achieved using the official Python client [19] and Helm CLI [16], enabling programmatic deployment, configuration, and lifecycle management of in-cluster components.

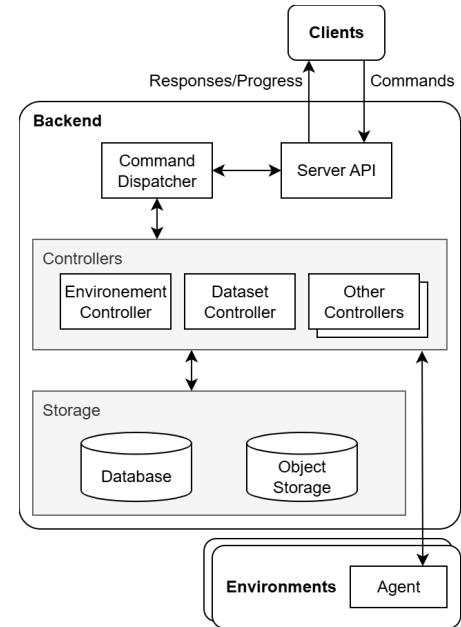


Figure 2: IDSLab backend architecture.

Environment Tier. The environment tier abstracts the physical or virtual infrastructure from which data is collected and on which models are deployed. This abstraction enables IDSLab to support heterogeneous targets, including workstations, virtual machines, on-premise clusters, and cloud platforms. While the current implementation focuses on Kubernetes clusters, the architecture is designed to accommodate additional environment types with minimal changes.

To ensure seamless integration with heterogeneous environments, IDSLab deploys platform-specific *agents* within the target infrastructure. These agents are responsible for collecting system events and metrics and forwarding them to processing services. In Kubernetes environments, the agent is implemented as a DaemonSet running on each node. It integrates Tracee [3] to monitor system calls, network activity, and file operations, as well as cAdvisor [7] and Prometheus [29] for resource metrics and time-series data collection. Fluent Bit [14] is used as a lightweight data pipeline for buffering, filtering, and forwarding the collected telemetry to the appropriate services.

2.3 IDSLab Features

Environment Integration. IDSLab integrates with Kubernetes clusters, after which it can deploy, configure, and monitor in-cluster agent components. Building on this connection, IDSLab also supports controlled adversary simulation: users can define targets (e.g., endpoints or networks) and execute parameterized attack operations selected from an extensible catalog exposed through the user interface.

Diverse Data Collection. IDSLab deploys a *collector* agent within the target infrastructure to gather data and forward it to user-selected repositories. Collectors can be configured with *filters* that specify which data to capture (e.g., selected syscalls, network events, or resource metrics) and the conditions events must satisfy (e.g., process name or container identifier), as illustrated in Figure 3a. IDSLab supports all Tracee event categories, as well as cAdvisor metrics with Kubernetes-specific attribute filtering. Collected data and processed dataset branches can be stored in S3-compatible object storage through reusable repositories.

Graph-Based Dataset Preprocessing Pipelines. Dataset construction is a central objective of IDSLab. The system provides a graph-based editor in which users compose preprocessing pipelines from modular nodes, each implementing a specific transformation (e.g., input parsing, feature extraction, labeling). Figure 3b shows the dataset graph editor integrated into the client. This representation enables flexible pipeline composition, rapid experimentation, and the generation of multiple dataset variants from the same raw sources. Graphs can be shared among users together with the underlying data, supporting reproducibility and collaborative experimentation.

3 Evaluation

To demonstrate the practicality and end-to-end capabilities of IDSLab, we conducted an experiment that covers the main stages of the ML-based IDS workflow: (1) collecting syscall data from a microservice application under both benign and attack conditions,

Table 1: Extracted behavioural features

Feature	Description
<i>args num</i>	Number of arguments
<i>return value</i>	Syscall return value
<i>time since prev global s</i>	Time since the previous event in the whole dataset
<i>time since prev same container s</i>	Time since previous event for the same container
<i>rc1 container</i>	Rolling count of events in the last 1 second for that container
<i>rc5 container</i>	Rolling count of events in the last 5 second for that container

(2) constructing a labeled DoS detection dataset, and (3) training and evaluating a machine learning classifier to distinguish between normal and malicious behavior.

We deployed a local single-node K3s cluster [30] inside a virtual machine and hosted Google’s Bank of Anthos microservice application [15] as the target workload. This environment was integrated into IDSLab by importing the cluster configuration file and registering the application’s frontend endpoint. This setup allowed IDSLab to automatically deploy its monitoring agents and interact with the environment through its standard interfaces.

After connecting IDSLab to the environment, dataset construction required two main steps: starting a collector and assembling a preprocessing pipeline using the graph-based editor. During data collection, we launched a Slowloris DoS attack [35] against the application’s frontend endpoint to simulate adversarial behavior. The collector was configured using a filter (Figure 3a) to capture TCP network events. While IDSLab supports heterogeneous telemetry sources, this use case focuses on network events to demonstrate end-to-end dataset construction for DoS detection.

Once the raw capture files were available, we constructed a preprocessing graph (Figure 3b) to transform them into a structured dataset. Events were labeled based on the originating process name, and a set of behavioral features was extracted. Table 1 summarizes the selected features. The final dataset contains 11,786 samples, including 9,236 benign events and 2,550 DoS events.

To validate the usability of the generated dataset, we trained a binary classifier to distinguish between benign and DoS events. We used the XGBoost library [34] to train the model, with an 80/20 split between training and test sets. The resulting performance metrics are reported in Table 2.

It is important to emphasize that this experiment is not intended to establish state-of-the-art detection performance. Rather, its objective is to demonstrate the correctness, consistency, and practical usability of the datasets produced by IDSLab. The high F1-scores achieved for both classes (98.21% for benign, 93.54% for DoS) indicate that the generated labels and features are coherent and informative, confirming that IDSLab-produced datasets can directly support standard ML-based IDS workflows such as training, evaluation, and comparative analysis.

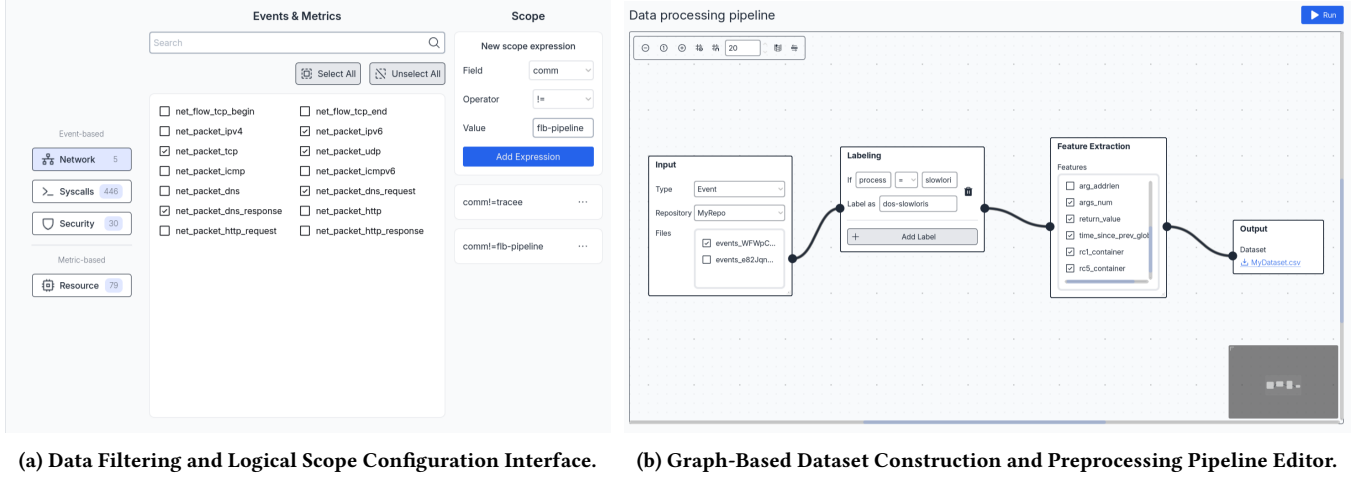


Figure 3: IDSLab Interactive Interfaces for Dataset Configuration and Construction.

Table 2: XGBoost classifier test results

Class	Precision (%)	Recall (%)	F1-score (%)	Samples
Benign	98.27	98.16	98.21	1848
DoS	93.36	93.73	93.54	510

4 Related Work

Network-only IDS dataset generation. Several dataset-generation frameworks focus on *network* intrusion detection systems (NIDS) and primarily produce packet traces or flow records, which limits their ability to capture host-level signals (e.g., system calls, logs, and resource metrics) commonly used in modern IDS deployments [27, 21, 9]. ID2T (*Intrusion Detection Dataset Toolkit*) creates labeled NIDS datasets by injecting synthetic attacks into background PCAP traffic, enabling controlled evaluation of detection methods [10]. NEDAGEN generates NIDS datasets by building configurable synthetic networks (scalable to large container counts), executing benign and ATT&CK-aligned malicious activities, and exporting captures in PCAP/JSON for downstream dataset construction [32]. ConCap supports reproducible NIDS experimentation by setting up an isolated environment that produces automatically labeled packets/NetFlows and can be shared via a single configuration file [33]. Farasat et al. propose a Kubernetes-based testbed leveraging eBPF/XDP to generate network traffic datasets for ML-based NIDS experiments with strong ground-truth guarantees [12].

Multi-source dataset collection. CREME proposes an automated toolchain for IDS dataset generation integrating configuration, attack reproduction, data collection, feature extraction, labeling, and evaluation across multiple data sources [6]. However, its predefined testbed and fixed scenarios make transferring the setup to new environments or research questions labor-intensive. In contrast, IDSLab’s low-code design decouples environment integration, data-collection configuration, and preprocessing pipelines, reducing the effort required to port experiments across scenarios.

5 Conclusion

In this paper, we presented IDSLab, a low-code experimental platform designed to support data-centric research and prototyping of machine learning-based intrusion detection systems. IDSLab addresses key challenges in current IDS workflows, including fragmented tooling, limited dataset diversity, and poor reproducibility, by integrating environment management, adversary simulation, scalable multi-source data collection, and graph-based dataset construction within a single extensible framework.

Through an end-to-end use case involving DoS attack simulation, dataset generation, and model training, we demonstrated how IDSLab simplifies the early stages of the IDS development lifecycle. Our evaluation illustrates the practical usability of the tool and its potential to support systematic, reproducible, and heterogeneous data-driven experimentation.

IDSLab is under active development. Future work will focus on integrating native support for model training, evaluation, and deployment, extending environment support beyond Kubernetes, and enriching the library of preprocessing and feature extraction components. We also plan to support additional attack scenarios, data modalities, and collaborative workflows. By lowering the barrier to IDS experimentation, we aim to foster more transparent, realistic, and reproducible security research.

Availability

IDSLab source code, installation instructions, and documentation are publicly available online [25]. A screencast demonstrating the platform’s end-to-end workflow is available at [26]. The dataset used in the evaluation, including the raw captures, labeled samples, and ML notebooks, is also publicly accessible to facilitate reproducibility and further experimentation [24].

Acknowledgment

This work was supported by funding from the Innovation for Defence Excellence and Security (IDEaS) program from the Department of National Defence (DND), Canada.

References

- [1] Qazi Omair Ahmed. 2024. Machine learning for intrusion detection in cloud environments: a comparative study. *Journal of Artificial Intelligence General science (JAIGS) ISSN*, 3006–4023.
- [2] Oluwasegun Apejoye, Nemitari Ajenka, Jun He, and Xiaoqi Ma. 2025. Critical review of network intrusion detection benchmark datasets for practical iot security. *Computer Networks and Communications*, 182–208.
- [3] [SW] Aqua Security and Tracee Contributors, Tracee 2025. Aqua Security. URL: <https://aquasecurity.github.io/tracee/latest/>, vcs: <https://github.com/aquasecurity/tracee>.
- [4] Rabia Basry, Shahid Ameer, Amina Aslam, Wasim Akram, Muhammad Suleman Shahzad, and Junaid Hamza. 2025. Cyber defense through machine intelligence: evolving perspectives on intrusion detection and prevention. *Journal for Current Sign*, 3, 4, 77–92.
- [5] Philipp Bönninghausen, Rafael Uetz, and Martin Henze. 2024. Introducing a comprehensive, continuous, and collaborative survey of intrusion detection datasets. In *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*, 34–40.
- [6] Huu-Khoi Bui, Ying-Dar Lin, Ren-Hung Hwang, Po-Ching Lin, Van-Linh Nguyen, and Yuan-Cheng Lai. 2021. Creme: a toolchain of automatic dataset collection for machine learning in intrusion detection. *Journal of Network and Computer Applications*, 193, 103212.
- [7] [SW], cAdvisor (Container Advisor) 2025. Google. URL: <https://github.com/google/cadvisor/blob/master/docs/storage/prometheus.md>, vcs: <https://github.com/google/cadvisor>.
- [8] Victor Chang, Lewis Golightly, Paolo Modesti, Qianwen Ariel Xu, Le Minh Thao Doan, Karl Hall, Sreeja Boddu, and Anna Kobusińska. 2022. A survey on intrusion detection systems for fog and cloud computing. *Future Internet*, 14, 3, 89.
- [9] Zhiyan Chen, Murat Simsek, Burak Kantarci, Mehran Bagheri, and Petar Djukic. 2024. Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier. *Computer Networks*, 250, 110576.
- [10] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Aidmar Wainakh, Max Mühlhäuser, and Simin Nadjm-Tehrani. 2021. On generating network traffic datasets with synthetic attacks for intrusion detection. *ACM Transactions on Privacy and Security (TOPS)*, 24, 2, 1–39.
- [11] [SW], DuckDB 2025. DuckDB Labs. URL: <https://duckdb.org/>, vcs: <https://github.com/duckdb/duckdb>.
- [12] Talaya Farasat, JongWon Kim, and Joachim Posegga. 2024. Advancing network security: a comprehensive testbed and dataset for machine learning-based intrusion detection. *arXiv preprint arXiv:2410.18332*.
- [13] Jose Flora and Nuno Antunes. 2024. Evaluating intrusion detection for microservice applications: benchmark, dataset, and case studies. *Journal of Systems and Software*, 216, 112142.
- [14] [SW], Fluent Bit 2025. CNCF (Cloud Native Computing Foundation). URL: <https://fluentbit.io/>, vcs: <https://github.com/fluent/fluent-bit>.
- [15] [SW] GoogleCloudPlatform Contributors, Bank of Anthos 2025. GoogleCloudPlatform / Google Cloud. vcs: <https://github.com/GoogleCloudPlatform/bank-of-anthos>.
- [16] [SW], Helm: The Kubernetes Package Manager 2025. Cloud Native Computing Foundation (CNCF). URL: <https://helm.sh/>, vcs: <https://github.com/helm/helm>.
- [17] [SW] Godot contributors Juan Linietsky Ariel Manzur, Godot Engine 2025. Godot Engine Community. URL: <https://godotengine.org/>, vcs: <https://github.com/godotengine/godot>.
- [18] Sivaprasad Yerneni Khaga and Aarthi M Reddy. 2025. Multi-modal deep learning for robust cloud intrusion detection: fusion of logs, metrics, and traces.
- [19] [SW], Kubernetes Python Client 2025. Cloud Native Computing Foundation (CNCF). vcs: <https://github.com/kubernetes-client/python>.
- [20] Avinash Kumar and Jairo A Gutierrez. 2025. Impact of machine learning on intrusion detection systems for the protection of critical infrastructure. *Information*, 16, 7, 515.
- [21] Ying-Dar Lin, Shin-Yi Yang, Didik Sudyana, Fietyata Yudha, Yuan-Cheng Lai, and Ren-Hung Hwang. 2024. Two-stage multi-datasource machine learning for attack technique and lifecycle detection. *Computers & Security*, 142, 103859.
- [22] [SW], MinIO 2025. MinIO, Inc. URL: <https://min.io/>, vcs: <https://github.com/minio/minio>.
- [23] Vincent Zibi Mohale and Ibidun Christiana Obagbuwa. 2025. Evaluating machine learning-based intrusion detection systems with explainable ai: enhancing transparency and interpretability. *Frontiers in Computer Science*, 7, 1520741.
- [24] [SW] Reda Morsli, Experiment dataset 2026. URL: <https://kaggle.com/datasets/deacafbe9bef40e508fd655dc31672507cc4b4a6ccbf43b98447f970ee62e980>.
- [25] [SW] Reda Morsli, IDSLab 2026. vcs: <https://github.com/RedaMorsli/IDSLab>.
- [26] Reda Morsli. 2026. Idslab screencast. (2026). <https://youtu.be/IKUNkGjKVPY>.
- [27] Reda Morsli, Nadja Kara, Hakima Ould-Slimane, and Laaziz Lahlou. 2025. Multidimensional intrusion detection system for containerized environments. In *2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*. IEEE, 546–554.
- [28] Oyekunle Claudius Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, Chidimma Francisca Azubuko, et al. 2024. Microservices architecture in cloud-native applications: design patterns and scalability. *International Journal of Advanced Research and Interdisciplinary Scientific Endeavours*, 1, 2, 92–106.
- [29] [SW] Prometheus Authors, Prometheus 2026. Cloud Native Computing Foundation (CNCF). URL: <https://prometheus.io/>, vcs: <https://github.com/prometheus/prometheus>.
- [30] [SW] Rancher Labs and k3s Contributors, k3s 2025. Rancher / SUSE. URL: <https://k3s.io/>, vcs: <https://github.com/k3s-io/k3s>.
- [31] [SW] The Linux Foundation and contributors, Model Context Protocol (MCP) Specification version 2025-11-25, Nov. 25, 2025. Model Context Protocol Project. URL: <https://modelcontextprotocol.io/specification/2025-11-25>, vcs: <https://github.com/modelcontextprotocol/modelcontextprotocol>.
- [32] [SW] Dennis van Wijk, Jeroen van Saane, and COSSAS Contributors, NEDAGEN: A Network Traffic Dataset Generator for Network-based Intrusion Detection Systems Jan. 2026. COSSAS Project. URL: <https://github.com/COSSAS/nedagen>, vcs: <https://github.com/COSSAS/nedagen>.
- [33] Miel Verkerken, Laurens D'hooge, Bruno Volckaert, Filip De Turck, and Giovanni Apruzzese. 2025. Concap: practical network traffic generation for flow-based intrusion detection systems. *arXiv preprint arXiv:2509.16038*.
- [34] [SW], XGBoost 2025. DMLC (Distributed (Deep) Machine Learning Community). URL: <https://xgboost.ai/>, vcs: <https://github.com/dmlc/xgboost>.
- [35] Gokberk Yaltirakli. 2015. Slowloris. *github.com*. <https://github.com/gkbrk/slowloris>.
- [36] Chaoyu Zhang, Ning Wang, Y Thomas Hou, and Wenjing Lou. 2025. Machine learning-based intrusion detection systems: capabilities, methodologies, and open research challenges. *AI for Cybersecurity: Research and Practice*, 67–108.