International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Applying Machine Learning for Network Security Monitoring in Kubernetes Environments

Matvii Horskyi*

Senior Software Engineer,Austin, TX, United States
Email:matvii.horskyi@gmail.com

Abstract

This article presents a systematic analysis of methodological approaches to applying machine learning for network security monitoring in Kubernetes environments, where the dynamic nature of containerized workloads, microservice architectures, and distributed observability significantly complicate attack detection and interpretation. The study is conducted as a review-and-analytical synthesis of peer-reviewed publications, summarizing architectural models, telemetry types, algorithmic approaches, and operational constraints without quantitative aggregation of results due to methodological heterogeneity of the sources. Particular attention is paid to the impact of Kubernetes network infrastructure on the properties of observed data, the role of non-identical and imbalanced distributions in distributed environments, and the limitations of classical centralized training schemes. The analysis shows that the prevailing practice of using ML-based intrusion detection systems is oriented toward isolated event detection and does not account for systemic telemetry distortions introduced by container network interfaces and the control plane. It is established that the greatest practical robustness is demonstrated by architectures based on hybrid data sources and federated learning algorithms, as well as two-tier schemes that separate anomaly detection from semantic interpretation. It is shown that the effectiveness of ML monitoring in Kubernetes is determined not so much by model complexity as by the degree of architectural integration into a managed security control loop encompassing observation, analysis, interpretation, and controlled response. The article will be useful for cybersecurity researchers, cloud platform architects, container security engineers, and specialists in operating distributed systems.

Keywords: Kubernetes; network security; machine learning; intrusion detection; federated learning; container networks; managed security control loop.

Received: *1/5/2026*
Accepted: *3/5/2026*
Published: *3/15/2026*

---

* *Corresponding author.*

1. Introduction

The integration of machine learning is becoming a necessary condition for network security monitoring in Kubernetes environments, which are distinguished by dynamic container workloads and microservice architecture. Since Kubernetes is oriented toward orchestration and scaling rather than protection, security relies on external analysis tools, while traditional signature-based and rule-oriented mechanisms are poorly adapted to the volatile network topology and the prevalence of internal traffic, rendering protection predominantly reactive.

The application of machine learning in Kubernetes network security is typically limited to isolated anomaly detection without inclusion in network policy management loops [4]. Such a modular approach retains analytical models in the role of a signal source and does not ensure a transition to managed defensive actions, while the absence of a universal architectural model is due to the dependence of solution effectiveness on the network layer, telemetry characteristics, and operational constraints regarding resources and latency.

The research problem lies in the absence of a holistic methodological framework that views machine learning for Kubernetes network security monitoring as a managed and architecturally embedded system. Unlike purely systematic literature reviews that focus on categorizing detection methods and reporting comparative metrics, this study adopts an expert analytical synthesis perspective. The analysis is oriented toward identifying architectural regularities and design patterns that determine the robustness and operational viability of machine learning–based security monitoring in production Kubernetes environments. Existing works focus predominantly on comparing models and detection metrics, whereas architectural limitations, the impact of network infrastructure, and the integration of detection results into response loops are analyzed in a fragmented and non-systemic manner.

The aim of this study is to structure approaches to the application of machine learning for network security monitoring in Kubernetes environments and to determine the conditions under which analytical mechanisms transition from isolated attack detection to an element of a managed and reproducible protection loop. To achieve this goal, the article addresses the following research tasks:

– identifying dominant architectural patterns of applying machine learning methods for network security monitoring in Kubernetes environments;

– analyzing the impact of Kubernetes network abstractions and control plane mechanisms on the quality and statistical properties of telemetry data used in machine learning tasks;

– comparing centralized, distributed, and federated monitoring architectures in terms of robustness to non-IID and unbalanced data;

– formulating architectural requirements for the integration of machine learning methods into a managed security loop comprising detection, interpretation, and response stages.

The research hypothesis is that the effectiveness of applying machine learning for Kubernetes network security is determined not so much by the complexity of the models used, but by the degree of their architectural alignment

with the platform, resistance to network telemetry distortions, and inclusion in the managed "observation – analysis – interpretation – action" cycle. This system construction reduces reliance on manual expertise and increases the resilience of security solutions under conditions of constant infrastructure and workload changes.

The scope of the study is limited to the analysis of methodological, architectural, and operational aspects of applying machine learning for network security monitoring in Kubernetes environments. Primary attention is devoted to data sources, detection models, distributed learning, result interpretation, and integration with response mechanisms. Economic, regulatory, and sectoral factors are considered only insofar as they influence the practical realizability and scalability of machine learning-based network security systems in industrial Kubernetes clusters. The scientific contribution of this work lies in the architectural interpretation of existing approaches to applying machine learning methods for ensuring the security of Kubernetes environments. Unlike the majority of review works, which focus predominantly on comparing model quality metrics, the article proposes viewing ML security monitoring as a managed loop combining telemetry collection, anomaly detection, semantic event interpretation, and controlled response. Additionally, the significance of two-tier detection architectures and federated learning under non-IID data distribution conditions is substantiated.

2. Materials and Methods

This study is conducted in the format of a systematic literature review. In addition to the analysis of peer-reviewed publications, the architectural interpretation of the reviewed approaches is informed by practical experience in designing and operating production Kubernetes platforms. This includes distributed systems with active control planes, multi-layer observability, and network security monitoring under resource, latency, and stability constraints. This practical perspective is not treated as an empirical data source, but is used to guide architectural reasoning and synthesis. The review-and-synthesis approach without quantitative aggregation of results is adopted as the methodological basis, due to the high heterogeneity of architectures, datasets, and metrics in Kubernetes security research.

The publication search focused on research dedicated to the application of machine learning for attack detection, network security monitoring, and telemetry analysis in container and Kubernetes environments. The selection included peer-reviewed journal articles and conference proceedings reflecting modern approaches to ML-IDS, federated learning, and the analysis of system calls, network flows, and control logs.

The review included works satisfying the following criteria:

– focus on container, cloud, or Kubernetes environments;

– use of machine learning or deep learning methods for attack or anomaly detection tasks;

– presence of reproducible architectural or methodological descriptions;

– presentation of quantitative quality metrics or operational characteristics.

Studies limited only to traditional signature-based mechanisms or not accounting for the specifics of container infrastructure were excluded.Publication selection was carried out in several stages. Primary screening by title and abstract was followed by full-text analysis and a final check for compliance with inclusion criteria. For each article, data were extracted regarding the telemetry type, monitoring architecture, applied machine learning models, evaluation metrics, and operational constraints.

The final selection includes publications reflecting the main directions of machine learning application for Kubernetes security monitoring. The work of Araujo and Vieira [1] analyzes container system calls and demonstrates the advantages of graph data representation for increasing attack detection robustness. The study by Cohen and his colleagues [2] is dedicated to using language models to harden Kubernetes configurations based on logs and resource descriptions. The contribution of Dakić and his colleagues [3] lies in the quantitative assessment of the impact of container network interfaces on throughput and latency, which form network monitoring constraints. The review by Diana and his colleagues [4] is used as a methodological basis for classifying IDS and evaluation metrics. The work of Doriguzzi-Corin and his colleagues [5] examines federated learning under conditions of non-identical data distributions in Kubernetes clusters. A two-tier attack detection architecture using language models is presented by Kalafatidis and his colleagues [6]. A distributed HIDS architecture for Kubernetes is described in the study by Levy Rocha and his colleagues [7]. An analysis of observability and Kubernetes security compliance indicators is performed by Morić and his colleagues [8]. The review by Noor and his colleagues [9] systematizes machine learning and transfer learning methods for IDS and emphasizes the role of realistic datasets. The application of neural network models and kernel mechanisms for protecting microservice environments is shown in the work of Park and his colleagues [10].

The synthesis of results is performed as a comparative analysis of architectures and methodological approaches without direct comparison of absolute metrics between studies. Primary attention is focused on identifying general patterns, architectural limitations, and conditions for the applicability of machine learning for Kubernetes network security monitoring.

3. Results

This section presents the results of an analytical generalization of quantitative indicators provided in the selected publications, rather than the results of an independent experimental study. All numerical values, quality metrics, and comparative characteristics are borrowed from primary sources and are used exclusively to identify architectural patterns, methodological trade-offs, and systemic limitations of applying machine learning methods for security monitoring in Kubernetes environments. The analysis of deep learning model effectiveness is performed using the BiLSTM architecture as an example, utilizing the 5G-NIDD dataset formed based on traffic from a real fifth-generation communication network. The choice of this dataset is due to the high variability of network flows, the presence of both normal and malicious traffic, and the absence of artificial simplification of attack scenarios, which allows experimental conditions to approximate Kubernetes operational environments, as emphasized by Noor and his colleagues [9]. This approach allows for assessing model behavior under high event density and overlapping activity types.

BiLSTM evaluation results show consistently high values, exceeding 97% across all metrics [9]. It is fundamentally important that the increase in precision is not accompanied by a decrease in detection recall, indicating balanced model behavior when classifying network events. In Kubernetes conditions, such a balance is critical, as excessive false positives quickly overload response loops, whereas missed attacks are capable of scaling through inter-service interactions, which is recorded in the study by Doriguzzi-Corin and his colleagues [5]. Figure 1 shows the quality metrics of the BiLSTM model when analyzing network traffic from the 5G-NIDD dataset, characterizing precision, recall, accuracy, and the integral quality score.

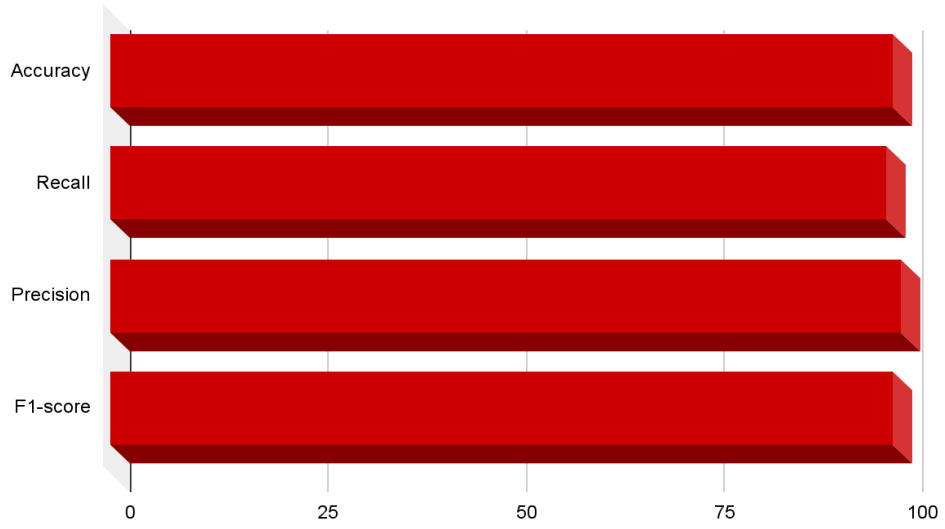

Figure 1: Quality metrics of the BiLSTM model on the 5G-NIDD dataset (Compiled by the author based on source: [5])

The set of values presented in Figure 1 captures a dense concentration of all key BiLSTM quality metrics in a narrow range of 97.9–99.6%, indicating the absence of a pronounced imbalance between detection precision and recall. A precision value of 99.62% reflects a minimal share of false positives, while a recall of 97.90% indicates the model's robust ability to identify malicious events without significant omissions, which is particularly critical when analyzing saturated network traffic. The proximity of the F1-score (98.75%) to accuracy (98.74%) testifies to the consistency of error distribution and the absence of bias toward a single class, which is characteristic of models resistant to data skew and complex attack patterns. Such a metric configuration shows that the model maintains stability while simultaneously accounting for competing requirements – reducing false positives and preserving high sensitivity.

When interpreting the obtained results, it is necessary to consider the dependence of detection quality on the form of input data representation. The study by Araujo and Vieira [1] shows that even highly effective models demonstrate degradation when using context-poor features. Consequently, the recorded BiLSTM indicators should be viewed as the result of a deep model operating with saturated and structured telemetry, which sets a benchmark for subsequent analysis of the applicability of similar approaches in Kubernetes environments with other data sources.

Federated learning is examined under conditions characteristic of Kubernetes clusters, where data are distributed unevenly among nodes and possess pronounced heterogeneity. Such a configuration reflects a practical situation where each node or namespace observes a limited and specific set of network events, as formalized in the study by Doriguzzi-Corin and his colleagues [5]. Under these conditions, the algorithm's ability to maintain detection quality is determined by the local accuracy of models and the stability of the aggregation procedure.

A comparison of federated learning algorithms shows that baseline parameter aggregation methods are sensitive to rare attacks and data skew. When training on local samples containing a limited set of scenarios, such algorithms demonstrate a decline in quality indicators during global model formation, which corresponds to the general limitations of intrusion detection systems highlighted by Diana and his colleagues [4]. An additional factor is the influence of the Kubernetes network infrastructure, where telemetry distortions and flow variability intensify the divergence of local updates, as follows from the analysis of container network interfaces presented by Dakić and his colleagues [3]. Table 1 presents numerical indicators and comparative characteristics of federated learning algorithms, reflecting differences in the level of network costs and attack detection quality under conditions of non-identical and unbalanced data.

Table 1: Comparative summary of FL algorithms for IDS in Kubernetes (Compiled by the author based on the source: [5])

FL Algorithm	**Network Overhead (MB)**	**Test F1 (mean)**	**Characteristics**
FedAvg	34.59	0.897	Baseline, poor under non-IID
FedProx	34.59	0.815	Stability increase, weak generalization
SCAFFOLD	60.25	0.893	Drift correction, high overhead
FedALA	34.59	0.814	Strong locally, weak globally
DAFL	71.98	0.855	Client filtering, costly communication
FedSBS	72.06	0.877	Information gain based selection
FLAD	52.18	0.984	Robust generalization, out-of-distribution

The data presented in Table 1 capture substantial differences in the ability of algorithms to maintain detection quality when transitioning from local training to a global model. The FLAD algorithm demonstrates a stable F1-score value, reflecting the preservation of balance between attack misses and false positives when aggregating structurally heterogeneous data. At the same time, FedAvg and its modifications are characterized by noticeable quality degradation in scenarios where individual clients train on narrow and rare attack classes.

The increase in generalization robustness in FLAD is accompanied by increased network costs, which is of

fundamental importance for Kubernetes environments where federated learning service traffic competes with application workloads. This trade-off between detection quality and resource costs correlates with the architectural features of distributed monitoring systems described by Levy Rocha and his colleagues [7] and with cluster observability and manageability requirements recorded by Morić and his colleagues [8].

Thus, the comparative analysis of federated algorithms shows that in Kubernetes conditions, the determining factor becomes the method's ability to support stable generalization under non-identical and unbalanced data, which is directly reflected in the differences in F1-score values between the considered approaches.

4. Discussion

The Kubernetes network infrastructure forms a specific observation environment in which traffic characteristics inherently differ from physical network parameters. The use of container network interfaces leads to systemic changes in throughput, latency, and resource consumption, which directly affect the properties of data entering machine learning models. The study by Dakić and his colleagues [3] quantitatively shows that container network interfaces introduce significant distortions that cannot be interpreted as noise or measurement error, as they are of a stable and reproducible nature. From an architectural perspective, Kubernetes network interfaces should therefore be treated as a systemic source of telemetry transformation rather than as incidental noise.

The obtained results indicate that the effectiveness of machine learning–based monitoring in Kubernetes environments cannot be interpreted solely through model performance metrics. Unlike traditional intrusion detection studies that emphasize classification accuracy, the analyzed works collectively demonstrate that operational robustness emerges from architectural alignment between analytical components and platform observability mechanisms. This shifts the analytical focus from isolated model optimization toward system-level integration, suggesting that ML-based monitoring should be conceptualized as a continuous control process embedded into infrastructure governance rather than as an independent analytical module.

An analysis of the presented characteristics shows that container network interfaces lead to a substantial reduction in throughput and a noticeable increase in latency, the magnitude of which varies significantly depending on the tuning profile and transmission parameters [3]. Moreover, the use of solutions based on extended kernel mechanisms is accompanied by a noticeable increase in computational load on nodes, limiting available resources for analytical components. This effect is fundamental for network monitoring systems, since machine learning algorithms begin to work not with the original network behavior of applications, but with its transformed form depending on the chosen network layer implementation. Figure 2 shows the impact of Kubernetes network interfaces on throughput, data transmission latency, and cluster node computational load.

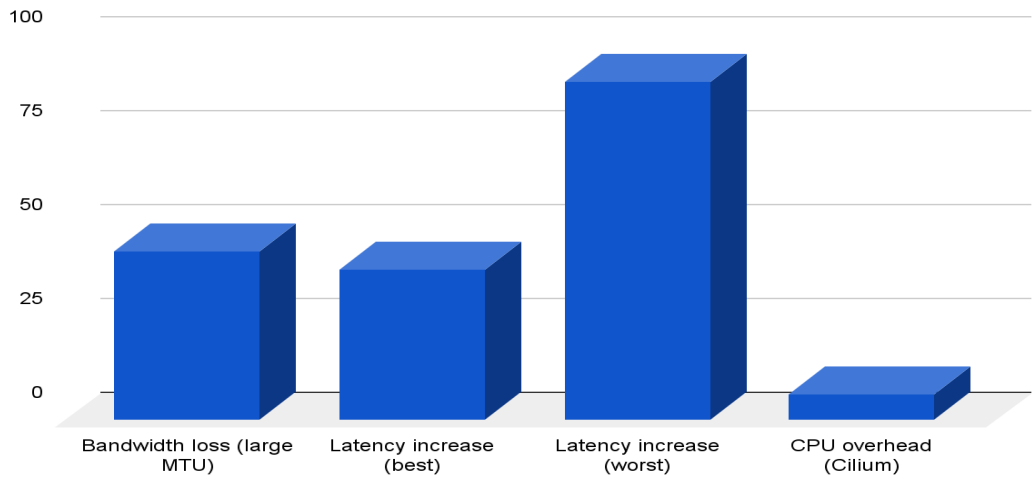

Figure 2: Impact of Kubernetes network interfaces on network telemetry and resource utilization (Compiled by the author based on source: [3])

The numerical values presented in Figure 2 show a pronounced and uneven impact of Kubernetes network interfaces on traffic characteristics and resource utilization. Throughput losses at a level of around 45% when using increased packet sizes indicate a substantial reduction in data transmission efficiency compared to a physical network, which changes the distribution of flows observed by the monitoring system. The increase in latency demonstrates an even wider range. Under optimal conditions, the increase is around 40%, whereas in unfavorable scenarios it reaches 90%, meaning nearly a twofold distortion of network interaction timing characteristics. Against this background, the computational load on the processor, recorded at a level of about 7% for solutions based on extended kernel mechanisms, appears quantitatively smaller; however, it is this load that forms constant background resource consumption affecting the available node computational capacity. The aggregate of these values indicates that input data for machine learning models are formed under conditions of systemically biased throughput and latency characteristics, while computational load acts as an additional constraint imposed by the Kubernetes network layer on the monitoring process.From the perspective of ML monitoring design, this means that models are trained and applied in conditions where telemetry changes may be caused not by an attack, but by network implementation features. Park and his colleagues [10] indicate that such distortions increase the risk of false positives and complicate the interpretation of network anomalies in microservice architectures. As a result, the boundary between malicious activity and infrastructural effects becomes less defined.Complexity is associated with the distributed nature of observability. Cohen and his colleagues [2] show that security and compliance indicators in Kubernetes are formed based on aggregated signals from different system levels. In such a configuration, the influence of the container network interface extends to all subsequent analysis stages, including event correlation and behavioral model construction. This creates a situation in which the Kubernetes network layer effectively becomes an active participant in forming training data, rather than a neutral traffic transmission channel.Consequently, the analysis of the Kubernetes network infrastructure shows that container network interfaces set rigid conditions for applying machine learning in network monitoring tasks, since models work with telemetry that is already transformed and limited by infrastructural solutions, rather than with "pure" network characteristics.

Previous studies predominantly examine intrusion detection systems from an algorithmic perspective, focusing on improving classification accuracy and reducing false positives [4,9]. While these works provide important insights into model performance, they often treat monitoring components as isolated analytical tools. Research by Doriguzzi-Corin and his colleagues [5] and Levy Rocha and his colleagues [7] highlights the importance of distributed monitoring architectures, emphasizing scalability and decentralized analysis. Building upon these findings, the present study extends prior research by demonstrating that architectural integration across observability layers represents a primary determinant of robustness in Kubernetes environments. In contrast to approaches centered on detection performance alone, this work interprets machine learning monitoring as part of a managed security loop linking observation, interpretation, and controlled response. The analysis of architectural approaches shows that the isolated application of attack detection mechanisms does not correspond to the multi-level structure of Kubernetes. Network events, container behavior, and control plane operations form interconnected signals requiring coordinated interpretation. Limiting analysis to a single layer leads to a loss of context and complicates the distinction between infrastructural effects and attack signs.In such a configuration, machine learning should be viewed as an analytical component within a distributed observability architecture, rather than as an autonomous protection mechanism. Within this framing, ML-based security monitoring can be modeled as a managed control loop comprising four tightly coupled stages: observation, analysis, interpretation, and action. Detection models operate at the analysis stage, while semantic interpretation mechanisms and response tools ensure that analytical outputs are transformed into controlled and auditable security actions. Detection effectiveness is determined by the degree of integration of analysis results into control and response loops, and the consistency of data from different system levels. This indicates that in industrial Kubernetes environments, architectural integration of machine learning components with observability pipelines and response mechanisms is more critical than incremental improvements in isolated model quality metrics. High detection accuracy without architectural alignment may increase operational risk by overloading response loops or generating non-actionable alerts. The network layer captures activity consequences, the execution layer reflects local behavioral deviations, and the control plane sets the context of ongoing changes.

Two-tier integration schemes of analytical models and language mechanisms demonstrate practical significance. In such architectures, the primary tier performs lightweight deviation detection with minimal latency, while the second tier is used for the semantic interpretation of a limited set of events. This allows for reducing computational costs and increasing interpretability without losing system manageability.Schemes in which detection results are used to form configuration changes and access policies are gaining development. This approach links event analysis with managed actions and moves detection systems into the active management loop. This study has several limitations. First, the analysis relies on experimental results reported in previously published works rather than independent empirical validation conducted in large-scale industrial Kubernetes clusters. Second, heterogeneity of datasets, evaluation metrics, and experimental configurations across reviewed studies prevented quantitative aggregation and direct statistical comparison of results. Third, the rapid evolution of Kubernetes networking mechanisms and cloud-native security tooling may alter some architectural constraints discussed in this work over time. Finally, organizational deployment factors, including operational policies, security maturity levels, and human decision-making processes, were considered only indirectly, which may influence real-world applicability of the proposed architectural interpretations.Thus, the architectural integration of machine learning

mechanisms and language models in Kubernetes forms a system in which deviation detection, interpretation, and managed impact turn out to be linked into a single loop. Taken together, the findings of this study contribute to the existing body of research by reframing machine learning–based security monitoring in Kubernetes environments from an algorithm-centered perspective toward an architectural and governance-oriented paradigm. The analysis demonstrates that detection performance alone does not determine operational effectiveness; instead, robustness emerges from the coordinated interaction between telemetry generation, distributed learning mechanisms, semantic interpretation, and controlled response processes. By synthesizing insights from intrusion detection, federated learning, and cloud-native observability research, this work provides an integrative conceptual foundation for designing ML-driven security systems as managed socio-technical infrastructures rather than standalone analytical tools. This perspective helps explain inconsistencies observed in prior studies and offers a unifying framework for future research on scalable security monitoring in distributed cloud environments.

5. Conclusion

The conducted study has shown that the effectiveness of applying machine learning for network security monitoring in Kubernetes environments is determined not so much by the choice of individual models or algorithms, as by the architectural consistency of analytical components with the platform infrastructure. The key factor is the system's ability to account for the multi-level nature of Kubernetes, including the network layer, container execution layer, and control plane, as well as systemic telemetry distortions formed by container network interfaces.The analysis demonstrated that isolated approaches to attack detection oriented exclusively toward network traffic or local events do not ensure robustness in a dynamic and distributed environment. Architectures combining hybrid data sources, distributed and federated learning schemes, and two-tier mechanisms separating deviation detection and interpretation show the greatest practical applicability. Such separation allows for reducing the load on infrastructure and increasing the manageability of analytical loops.A substantial result is the identification of the role of the Kubernetes network infrastructure as an active factor in forming data for model training and application. Container network interfaces are not a neutral traffic transmission channel but set constraints and biases that directly influence detection quality metrics and anomaly interpretation. Ignoring this circumstance leads to inflated expectations from models and a reduction in their operational reliability.

The practical significance of the obtained results lies in the possibility of using the identified architectural principles when designing security systems for Kubernetes clusters. For industrial Kubernetes clusters, the results emphasize the necessity of treating machine learning–based security monitoring as an architectural subsystem rather than as an auxiliary detection tool. Practical deployments benefit most from solutions that explicitly account for telemetry distortions introduced by container networking, support distributed or federated learning under non-IID conditions, and integrate detection outputs into managed response workflows aligned with operational constraints. Priority shifts from optimizing individual metrics to building a managed security loop in which machine learning, interpretation tools, and configuration change mechanisms function as a coordinated system.Thus, the effectiveness of ML-based network security monitoring in Kubernetes is ensured by the coordinated development of architecture, analytical mechanisms, and management processes, which allows for maintaining the resilience and adaptability of protection systems under conditions of high dynamics and uncertainty.

References

[1]. Araujo, I., & Vieira, M. (2025). Enhancing intrusion detection in containerized services: Assessing machine learning models and an advanced representation for system call data. Computers & Security, 154, 104438. https://doi.org/10.1016/j.cose.2025.104438

[2]. Cohen, O. S., Malul, E., Meidan, Y., Mimran, D., Elovici, Y., & Shabtai, A. (2025). KubeGuard: LLM-assisted Kubernetes hardening via configuration files and runtime logs analysis. arXiv. https://doi.org/10.48550/arXiv.2509.04191

[3]. Dakić, V., Redžepagić, J., Bašić, M., & Žgrablić, L. (2024). Performance and latency efficiency evaluation of Kubernetes container network interfaces for built-in and custom tuned profiles. Electronics, 13(19), 3972. https://doi.org/10.3390/electronics13193972

[4]. Diana, L., Dini, P., & Paolini, D. (2025). Overview on intrusion detection systems for computers networking security. Computers, 14(3), 87. https://doi.org/10.3390/computers14030087

[5]. Doriguzzi-Corin, R., Sabel, P., Cretti, S., & Ranise, S. (2025). Federated learning in the wild: A comparative study for cybersecurity under non-IID and unbalanced settings. arXiv. https://doi.org/10.48550/arXiv.2509.17836

[6]. Kalafatidis, S., Papageorgopoulos, N., Kartakoullis, A., & Ledakis, G. (2025). LLM-enhanced intrusion detection for containerized applications: A two-tier strategy for SDN and Kubernetes environments. In F. Skopik, V. Naessens, & B. De Sutter (Eds.), Availability, reliability and security. ARES 2025. Lecture Notes in Computer Science (Vol. 15998). Springer. https://doi.org/10.1007/978-3-032-00642-4_4

[7]. Levy Rocha, S., Lopes de Mendonca, F. L., Staciarini Puttini, R., Rabelo Nunes, R., & Amvame Nze, G. D. (2023). DCIDS—Distributed container IDS. Applied Sciences, 13(16), 9301. https://doi.org/10.3390/app13169301

[8]. Morić, Z., Dakić, V., & Čavala, T. (2025). Security hardening and compliance assessment of Kubernetes control plane and workloads. Journal of Cybersecurity and Privacy, 5(2), 30. https://doi.org/10.3390/jcp5020030

[9]. Noor, K., Imoize, A. L., Li, C.-T., & Weng, C.-Y. (2025). A review of machine learning and transfer learning strategies for intrusion detection systems in 5G and beyond. Mathematics, 13(7), 1088. https://doi.org/10.3390/math13071088

[10]. Park, H., El Azzaoui, A., & Park, J. H. (2025). AIDS-based cyber threat detection framework for secure cloud-native microservices. Electronics, 14(2), 229. https://doi.org/10.3390/electronics14020229