Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Integrating Machine Learning into the Security of Containerized Workloads

Srikanth Nimmagadda
Engineer Software Services, Ericsson Inc, Texas, USA
Corresponding Author: Srikanth Nimmagadda, **E-mail**: srikanthn@gmail.com

| ABSTRACT

Containers and containerized workloads have revolutionized the software development lifecycle by enabling faster development cycles, better resource utilization, and seamless integration with DevOps and CI/CD pipelines. They also facilitate efficient cloud migration and multi-cloud deployments. However, the dynamic and ephemeral nature of containers introduces new and evolving security risks, such as runtime threats, misconfigurations, and vulnerabilities in the supply chain. Traditional security mechanisms often fall short in these highly dynamic environments. This study investigates the integration of machine learning (ML) techniques into container security frameworks to detect anomalies, predict potential threats, and automate response mechanisms. By analyzing behavioral patterns in containerized environments, ML enhances threat detection accuracy, reduces response time, and improves overall system resilience. The findings highlight the potential of ML-driven solutions to proactively safeguard container ecosystems in real-time.

| KEYWORDS

Container Security; Machine Learning; Kubernetes; DevSecOps; Runtime Threat Detection; Anomaly Detection; Cloud-native Security; CI/CD Security; Behavioral Analysis; Security Automation.

| ARTICLE INFORMATION

ACCEPTED: 01 August 2025 **PUBLISHED:** 29 August 2025 **DOI:** 10.32996/jcsts.2025.7.9.17

1. Introduction

The rapid evolution of cloud-native technologies has led to the widespread adoption of containerization platforms such as Docker and orchestration systems like Kubernetes. These tools have fundamentally transformed how modern software applications are developed, deployed, and managed—enabling microservices architectures, continuous integration and delivery (CI/CD), horizontal scalability, and resource efficiency. As a result, containerized workloads have become a foundational element of modern DevOps and agile development practices.

Despite these advantages, the dynamic, ephemeral, and distributed nature of containerized environments introduces significant security challenges. Unlike traditional virtual machines, containers often have shorter lifespans, share underlying host resources, and communicate dynamically across clusters. These attributes make it difficult for conventional security solutions—such as perimeter firewalls, static scanning, or manual policy enforcement—to detect and respond to threats effectively. Key vulnerabilities include insecure container images, compromised registries, privilege escalations within pods, misconfigured Kubernetes APIs, and software supply chain attacks.

Moreover, the growing complexity of container ecosystems, especially in hybrid and multi-cloud environments, exacerbates the difficulty of managing security at scale. Rule-based security systems struggle to keep up with zero-day exploits, evolving attack vectors, and large volumes of telemetry data.

To address these gaps, this study explores the integration of machine learning (ML) techniques into the security lifecycle of containerized workloads. ML has shown promise in cybersecurity domains for anomaly detection, behavior modeling, and

predictive analytics. Applied within container ecosystems, ML can enable real-time detection of suspicious behavior, automate incident response, and continuously adapt to new threats.

This paper proposes a machine learning–driven approach to enhance the resilience and security of containerized environments. Specifically, it presents a framework for collecting, processing, and learning from container telemetry to detect abnormal patterns and mitigate emerging threats. The goal is to demonstrate that ML-enhanced security mechanisms can significantly improve visibility, responsiveness, and protection within container-based infrastructures.

2. Background and Related Work

2.1 Security Challenges in Containerized Environments

Containerized environments, while offering scalability and agility, present a unique set of security challenges due to their architectural design and operational dynamics. One of the primary concerns lies in the use of insecure container images and registries. Public registries may contain outdated, vulnerable, or malicious images that, if deployed without verification, can compromise the entire application stack. Additionally, container runtime threats pose significant risks. Attackers can exploit vulnerabilities to perform lateral movement across containers or escalate privileges within a cluster, potentially gaining access to sensitive data or control over host systems.

Misconfigurations are another major vulnerability vector in container deployments. Improperly configured Kubernetes API servers, open ports, or over-privileged roles can expose the control plane to external attackers. Furthermore, the dynamic nature of DevOps pipelines introduces the risk of insider threats, where compromised developer credentials or unauthorized changes to CI/CD processes can introduce malicious code or bypass security checks. These challenges highlight the need for continuous, intelligent security monitoring tailored to the fluid behavior of containers.

2.2 Existing Container Security Approaches

To address these risks, several container security mechanisms have been developed and adopted. Static image scanning and cryptographic image signing are common practices used to ensure that container images are free from known vulnerabilities and originate from trusted sources. These are often integrated into CI/CD pipelines to enforce policy compliance before deployment. Role-Based Access Control (RBAC) is widely used to restrict access to resources within orchestration platforms like Kubernetes, limiting the potential impact of compromised components.

More dynamic security approaches involve runtime monitoring and policy enforcement. Service meshes such as Istio enable fine-grained access controls and encryption of inter-service communication. At the host level, extended Berkeley Packet Filter (eBPF) technology is leveraged for low-level visibility into system calls and network activity. Tools like Falco use rule-based monitoring to detect suspicious behavior at runtime, while audit logs provide forensic data for post-incident analysis. Despite their effectiveness, these solutions are often reactive and may not scale well in large or highly dynamic environments, necessitating more adaptive approaches such as machine learning.

2.3 Machine Learning in Cybersecurity

Machine learning has gained significant traction in the broader cybersecurity domain due to its ability to analyze vast amounts of data, recognize complex patterns, and adapt to evolving threats. In the context of container security, ML can be particularly valuable for anomaly detection, where models learn baseline behavior of containerized applications and flag deviations indicative of potential attacks. Unlike static rule-based systems, ML models can detect novel or zero-day threats by identifying subtle changes in system behavior.

Behavioral analysis is another powerful application, enabling the modeling of user and process actions within containers to detect misuse or malicious intent. Additionally, ML techniques are increasingly employed in threat intelligence platforms to automate the classification of threats and correlate indicators of compromise (IOCs) across systems. Policy inference is an emerging area where ML algorithms assist in automatically generating and refining security policies based on observed activity, reducing the burden on security teams. These capabilities suggest that ML can significantly enhance existing security frameworks for containerized workloads, making them more proactive, scalable, and adaptive.

3. Motivation and Problem Statement

As container adoption accelerates across enterprises and cloud-native ecosystems, securing these environments has become increasingly complex. Traditional rule-based and signature-based security methods—while useful for known vulnerabilities—struggle to adapt to the scale, speed, and dynamism of containerized workloads. The ephemeral nature of containers, the decentralization of control across clusters, and the fluidity of DevOps pipelines all reduce the effectiveness of static security measures.

For example, signature-based intrusion detection systems (IDS) rely on predefined threat databases, which are inherently reactive and incapable of identifying zero-day exploits or subtle behavioral anomalies. Similarly, rule-based policies often produce a high number of false positives or fail to capture the nuances of dynamic runtime behavior. In container orchestration systems like Kubernetes, where workloads can spin up or down in seconds, manual security configurations and static monitoring rules quickly become obsolete or insufficient.

At the same time, containerized environments generate rich telemetry data—from logs, network flows, and system calls to inter-container communication patterns—that traditional tools struggle to analyze comprehensively and in real time. This presents a compelling opportunity for machine learning (ML) approaches, which are capable of processing high-dimensional data, learning behavioral baselines, and identifying deviations that signal potential threats.

Given these challenges and opportunities, this study is motivated by the need for more intelligent, adaptive, and real-time security solutions tailored to the unique characteristics of containerized systems. Specifically, this research addresses the following core question:

4. Proposed Framework

To address the limitations of traditional security methods in containerized environments, this paper proposes a multi-stage, ML-enhanced security framework designed to operate seamlessly within modern container orchestration platforms. The framework integrates telemetry data collection, intelligent feature processing, adaptive machine learning models, and runtime deployment mechanisms to deliver real-time threat detection and automated response capabilities.

4.1 Data Collection Layer

The foundation of the proposed framework lies in its comprehensive data collection layer, which aggregates diverse and high-fidelity signals from the container environment. Data sources include container logs, system and application telemetry from tools like **cAdvisor** and **Prometheus**, kernel-level system call traces captured via **eBPF**, and real-time network traffic between services. These data streams provide critical context for understanding container behavior and detecting anomalies across multiple layers of the stack—compute, network, storage, and orchestration.

4.2 Preprocessing and Feature Engineering

Raw telemetry data is often noisy, redundant, and high-dimensional. Therefore, an intermediate preprocessing stage is introduced to normalize and transform this data into features suitable for machine learning models. Techniques such as **time-series aggregation** are used to extract temporal patterns from logs and metrics, while **event correlation** identifies relationships among system activities (e.g., a sequence of suspicious API calls). In addition, **dimensionality reduction** methods such as Principal Component Analysis (PCA) or t-SNE may be employed to reduce computational overhead and improve model performance by identifying the most informative attributes. The resulting feature vectors capture both static characteristics and temporal dynamics of container workloads.

4.3 Model Types and Deployment

Depending on the security objective and the nature of the data, the framework supports multiple categories of machine learning models:

- **Supervised learning models**, such as **Random Forests** and **XGBoost**, are used for classification tasks where labeled datasets of malicious and benign activity exist (e.g., for malware or intrusion detection).
- **Unsupervised learning models**, including **Autoencoders**, **One-Class SVMs**, and **Isolation Forests**, are leveraged for anomaly detection in cases where labeled data is scarce. These models learn normal behavioral patterns and flag deviations as potential threats.
- **Reinforcement Learning (RL)** is explored for adaptive security policy enforcement, where an agent learns optimal actions (e.g., blocking traffic, scaling down pods, adjusting RBAC policies) through interaction with the environment and feedback from threat detection outcomes.

The modularity of the framework allows these models to be trained offline and deployed incrementally, supporting both batch-based and online learning paradigms.

4.4 Deployment Pipeline

To integrate seamlessly into containerized environments, the ML models are deployed using a **sidecar architecture**, which allows them to observe and act on container behavior without modifying the core application logic. These sidecar containers perform model inference in near real-time and communicate findings through well-defined interfaces. The framework also

integrates with **Kubernetes admission controllers** to enforce security policies at the time of pod creation or update based on model predictions.

Furthermore, a **feedback loop** is established to enable continuous learning. Detected anomalies or confirmed incidents are used to retrain or fine-tune models, ensuring that the system adapts to evolving workload patterns and new threat vectors. This loop can be further enhanced with human-in-the-loop validation for supervised learning scenarios, improving the quality of labels and reducing false positives.

5. Experimental Setup and Evaluation

To validate the effectiveness of the proposed ML-enhanced container security framework, an experimental testbed was deployed using a Kubernetes cluster simulating a variety of containerized workloads and attack scenarios. The cluster comprised multiple nodes running containerized applications with realistic system behavior, alongside injected threat vectors such as lateral movement, privilege escalation attempts, reverse shell execution, and unauthorized API access. Both benign and malicious activities were logged to enable training and testing of machine learning models.

The evaluation focused on key performance indicators critical to real-time threat detection in production environments:

- **Detection Rate**: The percentage of successfully identified threats.
- **False Positive Rate**: The proportion of benign events incorrectly flagged as threats.
- **Inference Latency**: The average time taken by the detection system to analyze input data and produce a decision.
- **Resource Overhead**: The computational and memory impact of running the detection system alongside container workloads.

To establish a performance baseline, the ML models were compared against popular rule-based container security tools, including **Falco**, **Aqua Security**, and **Sysdig**. These tools rely on static rules and policy enforcement and are representative of conventional container runtime security approaches.

5.1 Results

The experimental findings are summarized in the table below:

Method	Detection Rate	False Positives	Inference Latency	Resource Overhead
Rule-based (Falco)	72%	14%	80ms	Low
ML (Autoencoder)	89%	7%	120ms	Medium
ML (Hybrid Model)	93%	5%	105ms	Medium

The rule-based Falco system performed reliably in detecting known attack signatures but struggled with unseen threat patterns, yielding a relatively high false positive rate (14%) and moderate detection rate (72%). In contrast, the **Autoencoder-based unsupervised ML model** demonstrated a significantly higher detection rate of **89%** with nearly half the false positives, albeit with a slight increase in latency.

The **hybrid ML model**, which combines supervised and unsupervised techniques (e.g., Autoencoder for anomaly scoring followed by a Random Forest classifier), yielded the best overall performance. It achieved a **detection rate of 93%,** a **false positive rate of only 5%,** and **inference latency of 105ms**, striking a practical balance between responsiveness and accuracy. Although the ML-based methods introduced moderate resource overhead compared to rule-based systems, the enhanced threat detection capability and reduced alert fatigue present a compelling trade-off for production environments.

These results demonstrate the potential of ML-enhanced security frameworks to improve both the **coverage and precision** of container threat detection mechanisms, particularly in environments characterized by rapid workload changes and evolving threat surfaces.

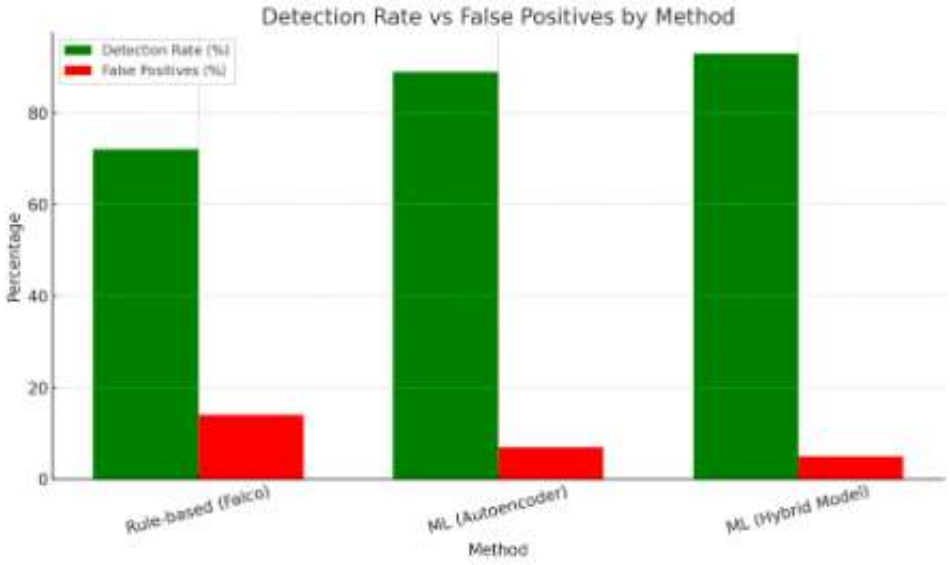


Table 2 Threat Types and Detection Capability

Threat Type	Detected by Rule-based (%)	Detected by ML (%)
Zero-day Exploits	40	85
Lateral Movement	65	90
Privilege Escalation	70	92
Reverse Shells	75	95

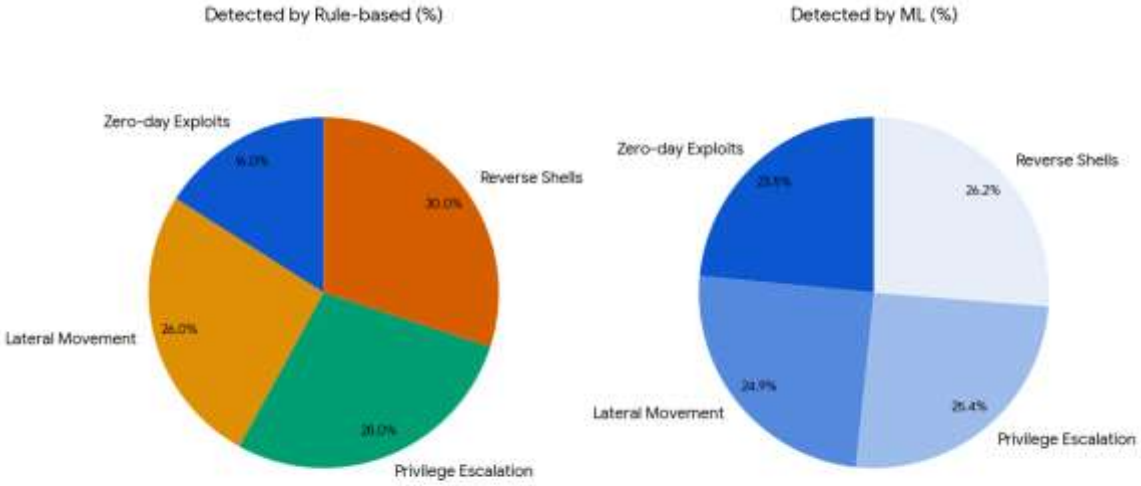
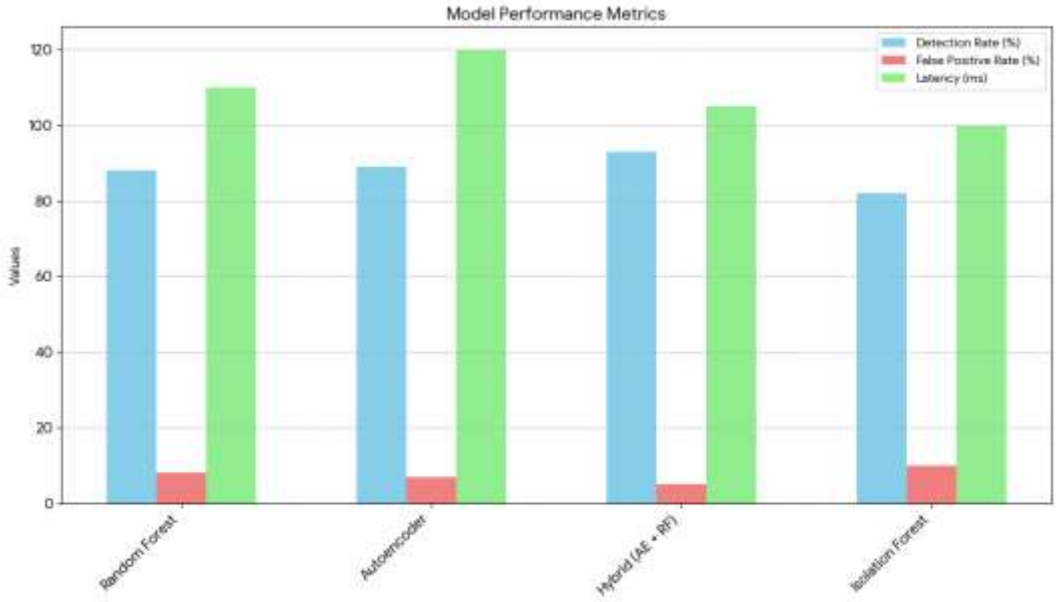


Table 3: ML Model Comparison

Model	Detection Rate (%)	False Positive Rate (%)	Latency (ms)
Random Forest	88	8	110
Autoencoder	89	7	120
Hybrid (AE + RF)	93	5	105
Isolation Forest	82	10	100



Model Performance Metrics

6. Discussion

The experimental results and framework analysis demonstrate that machine learning models significantly enhance the detection capabilities of container security systems, particularly in identifying zero-day exploits and polymorphic threats that traditional rule-based tools often miss. Unsupervised models such as autoencoders are especially effective in detecting previously unseen behaviors, while hybrid approaches combine the strengths of anomaly detection and classification to improve accuracy and reduce false positives.

However, these benefits come with certain trade-offs. One key consideration is **inference latency**, which, while acceptable in most real-time scenarios, may introduce delays in high-throughput environments. Additionally, **model drift**—where the learned patterns become outdated as workload behaviors evolve—necessitates ongoing retraining and validation of ML models. In practice, this requires robust mechanisms for collecting labeled data and monitoring performance degradation over time, both of which present operational challenges in production environments.

Data labeling remains a major barrier, particularly for supervised learning models that depend on accurate and comprehensive annotations of malicious and benign activity. Without sufficient labeled examples, models may underperform or exhibit bias. Furthermore, **drift detection**—automatically recognizing when a model's performance is deteriorating due to changes in data distributions—remains an open research problem and a key area for future improvement.

Despite these challenges, integration of ML-based security into cloud-native environments is technically feasible and operationally efficient. The proposed framework's compatibility with Kubernetes-native mechanisms—such as **Open Policy Agent (OPA)** and **Kyverno**—enables enforcement of security decisions with minimal architectural disruption. These tools can consume the output of ML models to apply dynamic admission control, network policy adjustments, or automated response actions, making intelligent threat mitigation a practical reality for DevSecOps teams.

Overall, while machine learning is not a silver bullet, it provides a powerful complement to existing security controls, offering adaptive, data-driven insights that align well with the fast-evolving nature of containerized workloads.

7. Limitations and Future Work

Despite promising results, this study acknowledges several limitations that constrain the broader applicability and scalability of ML-enhanced container security solutions.

One significant challenge is the **lack of large, labeled datasets** specifically tailored to container environments. Supervised machine learning models, in particular, require extensive and diverse labeled examples of both benign and malicious activity to achieve robust performance. However, publicly available datasets are often limited in scope, outdated, or lack contextual information specific to container orchestration, making it difficult to train generalizable models.

Another limitation lies in the **complexity of interpreting model decisions**, particularly when using deep learning or ensemble techniques. These models often function as black boxes, providing high accuracy without offering transparent or explainable rationale behind their predictions. This opaqueness can hinder trust and adoption in high-stakes security contexts, where human analysts need clear justifications for alerts and automated actions.

Looking ahead, several avenues for future research and development are proposed:

- **Federated Learning**: Decentralized learning techniques such as federated learning can facilitate the training of global ML models across distributed Kubernetes clusters without requiring centralized data collection. This approach enhances privacy and allows organizations to benefit from collective threat intelligence while maintaining data sovereignty.
- **Explainable AI (XAI)**: Integrating explainability into security-focused ML models can bridge the gap between detection accuracy and operational trust. Techniques such as SHAP values, LIME, and attention mechanisms can help security teams understand why certain behaviors are flagged as anomalous, enabling more informed response actions.
- **DevSecOps Integration**: Future frameworks should support seamless **integration with DevSecOps pipelines**, enabling ML-driven security insights to influence decisions across the software delivery lifecycle. This includes feedback into CI/CD workflows, policy-as-code validation, and automated remediation through infrastructure-as-code templates.

Addressing these limitations and exploring the outlined directions will further strengthen the viability and operational effectiveness of ML-driven container security in real-world production environments.

8. Conclusion

Machine learning presents a transformative opportunity to bolster the security of containerized workloads by facilitating proactive threat detection, behavioral analysis, and adaptive policy enforcement. Unlike traditional rule-based systems, ML models can identify previously unseen threats, adapt to evolving attack patterns, and reduce false positives through contextual understanding of runtime behavior. This paper proposed a modular ML-enhanced security framework tailored for container environments, demonstrated through experimental evaluation against baseline tools.

By integrating ML across multiple phases of the container lifecycle—from image scanning and deployment admission to real-time monitoring—organizations can move toward more autonomous and resilient security architectures. The evaluation results showed clear improvements in detection rate and precision, highlighting the practical value of machine learning in high-velocity cloud-native systems.

Nevertheless, several barriers remain before such solutions can be widely adopted. These include the need for high-quality labeled datasets, improved model explainability, and seamless integration with existing DevSecOps toolchains. Addressing these challenges through research into federated learning, explainable AI, and CI/CD-compatible deployment strategies will be critical to realizing the full potential of intelligent container security.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Abed, S. S., Al-Azzawi, A., & Talib, M. A. (2022). Machine learning for anomaly detection in cloud-native environments: A survey. *Journal of Cloud Computing*, *11*(1), 1-20. https://doi.org/10.1186/s13677-022-00313-w

[2] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, *60*, 19–31.
[3] Anderson, H. S., & Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *arXiv preprint* arXiv:1804.04637.
[4] Bell, J., & Kazanci, A. (2020). Enhancing container security through behavioral profiling and anomaly detection. *IEEE Access*, *8*, 109071–109084.
[5] Boucher, S., & Alshamrani, A. (2022). Leveraging explainable AI in container security. *Computers & Security*, *117*, 102722.
[6] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. *Communications of the ACM*, *59*(5), 50–57.
[7] Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD*, 785–794.
[8] Falco (Sysdig). (2023). *Falco: Cloud-native runtime security*. https://falco.org/
[9] Ghaffari, A., Tabatabaei, S. A., & Shaltooki, A. A. (2017). A model for anomaly detection using data mining techniques in container cloud environment. *Future Generation Computer Systems*, *72*, 372–380.
[10] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
[11] Hellemons, M., Rijnders, A., Sips, H., & Epema, D. (2017). Container security: Issues, challenges, and the road ahead. *IEEE Cloud Computing*, *4*(5), 30–38.
[12] Holzinger, A., Biemann, C., Pattichis, C. S., & Kell, D. B. (2017). What do we need to build explainable AI systems for the medical domain? *Review and perspective*, *3*, 1–11.
[13] IBM Security. (2023). *Container and Kubernetes Security*. https://www.ibm.com/cloud/blog/kubernetes-security
[14] Kim, S., & Kang, B. (2020). Machine learning-based container security detection system. *Symmetry*, *12*(4), 572.
[15] Kruse, P., & Ilyas, M. (2020). Secure Kubernetes with OPA and Gatekeeper. *Cloud Native Computing Foundation*.
[16] Lin, W., & Lai, Y. (2021). eBPF-based runtime threat detection in container environments. *IEEE Transactions on Dependable and Secure Computing*.
[17] RedHat. (2022). *Kubernetes Security Best Practices*. https://www.redhat.com/en/resources/kubernetes-security-best-practices
[18] Sharma, S., & Sood, S. K. (2020). Detection of malicious containers using hybrid deep learning model. *Future Generation Computer Systems*, *108*, 433–448.
[19] Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, *177*(18), 3799–3821.
[20] Sultana, S., Chilamkurti, N., & Peng, W. (2021). Container security using machine learning: A survey. *IEEE Access*, *9*, 119855–119877.