World Journal of Advanced Engineering Technology and Sciences

eISSN: 2582-8266

Cross Ref DOI: 10.30574/wjaets

Journal homepage: https://wjaets.com/

WJAETS
WJAETS
World Journal of Advanced Engineering Technology and Sciences

(Review Article)

Check for updates

Securing kubernetes: An integrated approach to ai-driven threat detection and EBPF-based security monitoring

Nawazpasha Shaik *

Humana Inc, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(01), 1569-1576

Publication history: Received on 10 March 2025; revised on 14 April 2025; accepted on 16 April 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.1.0377

Abstract

This article presents a comprehensive framework for enhancing Kubernetes security through the integration of artificial intelligence-driven threat detection and extended Berkeley Packet Filter (EBPF) monitoring technologies. As organizations increasingly adopt containerized environments for mission-critical applications, traditional security approaches have proven insufficient against sophisticated attacks targeting the dynamic nature of Kubernetes orchestration. The article proposes a novel security architecture that combines machine learning models for real-time telemetry analysis with kernel-level visibility provided by EBPF instrumentation. The article approach enables automated anomaly detection across multi-cluster deployments while dynamically enforcing security policies aligned with zero trust principles. The proposed framework addresses critical security challenges including cryptojacking, privilege escalation, and unauthorized API access with minimal performance overhead. Experimental evaluations demonstrate the effectiveness of this integrated approach compared to conventional security methods, particularly in identifying emerging threats and reducing false positives. The article contributes significant advancements to cloud-native security practices and provides a foundation for future work in adaptive policy enforcement for containerized workloads.

Keywords: Kubernetes Security; Artificial Intelligence; Extended Berkeley Packet Filter (EBPF); Zero Trust Architecture; Cloud-Native Security

1. Introduction

1.1. Overview of Kubernetes Adoption and Its Role in Modern Cloud-Native Infrastructures

Kubernetes has emerged as the predominant orchestration platform for containerized applications, transforming how organizations deploy and manage cloud-native infrastructures. As Shu Sekigawa, Chikara Sasaki, et al. [1] highlight, the adoption of Kubernetes has accelerated across industries seeking to modernize their applications and achieve greater operational efficiency. This transition represents a fundamental shift in infrastructure management paradigms, enabling organizations to build scalable, resilient, and portable applications that can run consistently across diverse computing environments.

1.2. Analysis of Unique Security Challenges in Containerized Environments

Despite its benefits, Kubernetes introduces complex security challenges inherent to containerized environments. These challenges stem from the dynamic nature of container deployments, where workloads are ephemeral and distributed across multiple nodes. Yutian Yang, Wenbo Shen, et al. [2] identify several critical security concerns unique to container orchestration, including expanded attack surfaces, pod-to-pod communication vulnerabilities, and privileged container

* Corresponding author: Nawazpasha Shaik

risks. The multi-tenancy aspects of Kubernetes clusters further compound these challenges, as workloads from different applications or teams share underlying infrastructure and kernel resources.

1.3. Discussion of Limitations in Traditional Security Approaches for Dynamic Kubernetes Deployments

Traditional security approaches, originally designed for static infrastructure models, prove inadequate in addressing the security requirements of dynamic Kubernetes environments. Conventional perimeter-based security models fail to account for the ephemeral nature of containers and the constant state of flux within Kubernetes clusters [1]. Additionally, legacy security tools lack visibility into container internals and struggle to understand the complex network of relationships between Kubernetes objects. This gap between traditional security models and modern containerized environments creates significant blind spots that malicious actors can exploit.

1.4. Introduction to the Need for Advanced Threat Detection and Monitoring Capabilities

The evolving threat landscape necessitates advanced detection and monitoring capabilities specifically tailored to Kubernetes environments. As containerized workloads become increasingly critical to business operations, organizations require security mechanisms that can provide deep visibility into container runtime behaviors while adapting to rapidly changing deployments [2]. These advanced capabilities must operate at both the application and infrastructure layers, monitoring not only container activities but also the orchestration layer itself. The integration of artificial intelligence for threat detection and extended Berkeley Packet Filter (eBPF) for kernel-level monitoring represents a promising approach to addressing these challenges, offering the potential to detect sophisticated attacks while maintaining the performance benefits that drive Kubernetes adoption.

2. Current State of Kubernetes Security

2.1. Review of Common Kubernetes Security Vulnerabilities and Attack Vectors

Kubernetes environments face a wide array of security vulnerabilities that threat actors actively exploit. Chris Binnie and Rory McCune [3] categorize these attack vectors into several distinct domains, including misconfigurations in the Kubernetes API server, inadequately secured etcd databases, and compromised container images. The dynamic nature of Kubernetes deployments creates unique security challenges as the infrastructure continuously evolves with workloads being created, updated, and destroyed. External attackers frequently target exposed Kubernetes dashboards, insecurely configured kubelet endpoints, and vulnerable container runtimes. These entry points can provide attackers with initial access to cluster resources, potentially leading to privilege escalation and lateral movement throughout the infrastructure.

Table 1 Common Kubernetes Security Vulnerabilities and Attack Vectors [4]

Vulnerability Category	Example Attack Vectors	Potential Impact
API Server Misconfigurations	Exposed dashboards, insecure authentication	Unauthorized cluster access
Container Runtime Vulnerabilities	Container escape, shared kernel exploitation	Privilege escalation
Supply Chain Weaknesses	Compromised container images, malicious dependencies	Malware deployment
Network Security Gaps	Insufficient network policies, unencrypted communication	Lateral movement
Inadequate RBAC	Overly permissive service accounts, excessive privileges	Unauthorized access to sensitive resources
Secret Management Flaws	Unencrypted secrets, embedded credentials	Credential theft

2.2. Examination of Present Security Frameworks and Best Practices

Current Kubernetes security frameworks involve layered approaches that encompass infrastructure, cluster, container, and application-level security measures. Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, et al. [4] outline a comprehensive set of security practices they term the "XI Commandments of Kubernetes Security," addressing critical aspects from authentication and authorization to network policies and runtime protection. These frameworks advocate for implementing role-based access control (RBAC), namespace isolation, network segmentation through NetworkPolicies, and strict pod security contexts. Many organizations adopt compliance-oriented approaches aligned with industry standards such as CIS Kubernetes Benchmarks, which provide measurable security controls for hardening Kubernetes deployments against common threats.

2.3. Analysis of Security Gaps in Standard Kubernetes Deployments

Despite available security frameworks, standard Kubernetes deployments often contain significant security gaps. Binnie and McCune [3] identify several common oversights, including the default permissive security posture of many Kubernetes components, inadequate monitoring of container runtime activities, and insufficient validation of deployed artifacts. Many organizations struggle with implementing proper secret management, leading to credentials being inadvertently exposed through environment variables or configmaps. Additionally, complex RBAC configurations frequently result in overly permissive access policies that violate the principle of least privilege. These gaps create opportunities for attackers to compromise clusters through methods that bypass conventional security controls, particularly when considering the rapid pace of updates to both applications and the underlying Kubernetes platform.

2.4. Challenges in Securing Multi-Cluster and Hybrid Kubernetes Environments

Multi-cluster and hybrid Kubernetes environments present additional security challenges beyond those of single-cluster deployments. Shamim, Bhuiyan, et al. [4] highlight the complexities of maintaining consistent security policies across heterogeneous environments spanning multiple cloud providers and on-premises infrastructure. These distributed architectures introduce complex identity management concerns, as different environments may implement varying authentication mechanisms. Network security becomes particularly challenging in these scenarios, as inter-cluster communication must be secured without compromising application performance. Organizations operating hybrid environments also struggle with achieving unified visibility across diverse deployments, creating blind spots in security monitoring. The disparate nature of these environments complicates vulnerability management, as patches must be coordinated across multiple clusters with potentially different configurations and versions of Kubernetes components.

3. AI-driven Threat Detection for Kubernetes

3.1. Theoretical Foundations of Machine Learning for Security Anomaly Detection

Machine learning provides powerful theoretical frameworks for identifying anomalous behaviors in complex environments like Kubernetes clusters. Ali Bou Nassif, Manar Abu Talib, et al. [5] outline how anomaly detection algorithms establish baseline patterns of normal system behavior, enabling the identification of deviations that may indicate security threats. These approaches leverage statistical methods, distance-based techniques, density-based models, and neural network architectures to distinguish between normal and abnormal activities. Within Kubernetes environments, machine learning models can analyze multidimensional data from container metrics, API server logs, network traffic, and system calls to establish normal behavioral patterns specific to each cluster's workload characteristics. The theoretical foundation of these techniques involves feature extraction from high-dimensional data spaces, dimensionality reduction for computational efficiency, and the establishment of decision boundaries that separate normal operations from potential security incidents.

3.2. Implementation Frameworks for AI-based Kubernetes Telemetry Analysis

Implementing AI-based security for Kubernetes requires robust frameworks capable of collecting, processing, and analyzing telemetry data at scale. Vasco Samuel Carvalho, Maria João Polidoro, et al. [6] describe architectures that capture and analyze multiple data streams including kube-audit logs, container runtime metrics, network flows, and host-level telemetry. These implementation frameworks typically employ streaming data pipelines that ingest telemetry data, preprocess and normalize it for model consumption, and feed it into trained models for real-time analysis. Effective implementations for Kubernetes environments must address the ephemeral nature of containers, tracking entities across restarts while maintaining context about their expected behaviors. Many frameworks employ ensemble approaches that combine multiple detection algorithms specialized for different types of threats, allowing for more robust identification of security incidents across diverse attack vectors while minimizing false positives that could lead to alert fatigue.

3.3. Real-time Detection Capabilities for Advanced Threats

AI-driven security solutions offer capabilities for detecting sophisticated threats targeting Kubernetes environments that traditional rule-based systems often miss. Nassif, Abu Talib, et al. [5] highlight how machine learning approaches can identify subtle indicators of attacks like cryptojacking, where containers are compromised to mine cryptocurrency using cluster resources. These models analyze resource utilization patterns, network communication behaviors, and process execution sequences to detect anomalous activities indicative of threats. For privilege escalation attacks, AI systems monitor for unusual permission changes, suspicious API calls, and abnormal access to sensitive resources within the cluster. The real-time nature of these detection systems enables rapid response to emerging threats, with many implementations providing automated containment capabilities that can isolate suspicious pods or restrict their network access to prevent lateral movement across the cluster infrastructure.

3.4. Comparative Analysis of Supervised vs. Unsupervised Learning Approaches for Kubernetes Security

The choice between supervised and unsupervised learning approaches significantly impacts the effectiveness of AI-driven Kubernetes security solutions. Carvalho, Polidoro, et al. [6] compare these approaches, noting that supervised learning requires labeled datasets of both normal and malicious activities specific to Kubernetes environments. While supervised methods can achieve high accuracy for known attack patterns, they struggle with zero-day threats and novel attack techniques. Conversely, unsupervised learning approaches can detect previously unknown anomalies by identifying deviations from established normal patterns without requiring labeled attack data. However, these methods typically generate more false positives and require additional contextual analysis to determine whether detected anomalies represent genuine security threats. Many mature Kubernetes security implementations employ hybrid approaches that combine supervised models for detecting known attack signatures with unsupervised techniques that identify suspicious behavioral deviations, providing defense-in-depth against both established and emerging threats to containerized environments.

Table 2 Common Kubernetes Security Vulnerabilities and Attack Vectors [4]

Learning Approach	Strengths	Limitations	Suitable Use Cases
Supervised Learning	High accuracy for known attacks, Lower false positives	Requires labeled datasets	Identifying known attack patterns
Unsupervised Learning	Detection of zero-day threats	Higher false positive rates	Resource abuse detection
Semi-supervised Learning	Balanced approach	Model drift over time	Baseline deviation detection
Reinforcement Learning	Adaptive policy enforcement	Complex implementation	Dynamic policy adjustment

4. Advanced eBPF Security Monitoring

4.1. Technical Foundations of eBPF and Its Evolution for Security Applications

Extended Berkeley Packet Filter (eBPF) represents a revolutionary technology that enables programmable access to various kernel subsystems without requiring kernel modifications or module loading. Simon Sundberg and Anna Brunstrom [7] describe how eBPF has evolved from its origins as a simple packet filtering mechanism to a sophisticated technology capable of attaching programs to various kernel hooks, including system calls, function entry and exit points, network events, and kernel tracepoints. This evolution has made eBPF particularly valuable for security monitoring applications, as it allows security tools to access low-level system information with minimal overhead. The eBPF architecture includes a verification engine that ensures programs cannot crash or compromise the kernel, making it ideal for security applications that require both safety and performance. The technology provides capabilities for introspecting running applications, monitoring network traffic, and observing system call patterns—all essential components for comprehensive security monitoring in containerized environments.

4.2. Kernel-level Visibility Mechanisms and Implementation in Kubernetes Contexts

In Kubernetes environments, eBPF provides unprecedented visibility into container activities at the kernel level. Songi Gwak, Thien-Phuc Doan, et al. [8] explain how eBPF programs can be attached to various kernel hooks to monitor container behaviors across namespaces, providing visibility that traditional container-focused monitoring tools cannot achieve. Within Kubernetes contexts, eBPF implementations typically deploy agents on each worker node to collect telemetry data about pod activities, network communications, and file system accesses. These agents leverage eBPF maps to efficiently share data between kernel and user space, enabling the correlation of events across containers and pods. The kernel-level visibility allows security tools to detect container escape attempts, privilege escalation, and unauthorized access to resources by monitoring system calls and process activities. This comprehensive visibility is particularly valuable in multi-tenant Kubernetes clusters where workloads from different trust domains run on shared infrastructure.

Table 3 eBPF Security Monitoring Capabilities in Kubernetes [7, 8]

Monitoring Domain	eBPF Capabilities	Security Applications
Process Execution	System call monitoring	Container escape detection
Network Activity	Packet inspection, Connection tracking	Lateral movement detection
File System Access	File operation tracing	Malware detection
Resource Utilization	Performance counters	Cryptojacking detection
Container Lifecycle	Container creation/deletion events	Unauthorized container deployment

4.3. Performance Analysis of eBPF-based Monitoring Compared to Traditional Approaches

eBPF-based security monitoring offers significant performance advantages over traditional security approaches. Sundberg and Brunstrom [7] highlight the efficiency of eBPF programs, which execute directly in the kernel context without requiring context switches or data copying between kernel and user space. This architectural advantage results in substantially lower overhead compared to traditional monitoring approaches that rely on system call interception or kernel modules. In Kubernetes environments, where performance and resource efficiency are critical, eBPF monitoring tools introduce minimal CPU and memory overhead while providing comprehensive visibility. The performance characteristics are particularly important for production workloads where security monitoring must not significantly impact application responsiveness or resource utilization. eBPF's ability to filter and aggregate data within the kernel further reduces the overhead by minimizing the amount of data that must be transferred to user-space analysis engines.

4.4. Case Studies of eBPF Security Implementations in Production Kubernetes Environments

Several organizations have successfully implemented eBPF-based security monitoring in production Kubernetes environments. Gwak, Doan, et al. [8] document implementations that leverage eBPF to detect and prevent security threats in containerized applications. These case studies demonstrate how eBPF-based tools can identify abnormal process executions within containers, detect unauthorized network connections, and monitor sensitive file accesses—all without significant performance impact. Some implementations focus on specific threat vectors, such as container escape attempts or privilege escalation, while others provide comprehensive security monitoring across multiple dimensions. The deployments typically integrate with existing security information and event management (SIEM) systems, enabling security teams to incorporate eBPF-generated insights into their established workflows. These case studies consistently report improvements in threat detection capabilities, particularly for sophisticated attacks that traditional container security tools fail to identify, while maintaining acceptable performance levels for production workloads.

5. Zero Trust Implementation with AI-driven Policy Enforcement

5.1. Zero Trust Architecture Principles for Kubernetes Environments

Zero Trust Architecture (ZTA) represents a paradigm shift in security approaches, moving away from perimeter-based models toward continuous verification of every access request regardless of source. Naeem Firdous Syed, Syed W. Shah, et al. [9] outline core Zero Trust principles including the elimination of implicit trust, continuous validation, and least-privilege access controls—all particularly relevant for Kubernetes environments where workloads are dynamic and distributed. In Kubernetes contexts, Zero Trust principles translate to treating each pod, service, and component as potentially compromised, requiring explicit authentication and authorization for all interactions. Daniel D'Silva and Dayanand D. Ambawade [10] describe how Kubernetes implementations of Zero Trust architectures leverage service mesh, admission controllers, and security policies to enforce fine-grained access controls at multiple layers of the stack. These implementations establish trust boundaries between namespaces and enforce strict workload identity verification through mechanisms like mutual TLS authentication and cryptographic attestation of container images and configurations.

5.2. Dynamic Policy Adjustment Based on Real-time Threat Intelligence

AI-driven systems enable dynamic policy adjustments based on real-time threat intelligence, allowing Kubernetes security postures to evolve in response to changing conditions. Syed, Shah, et al. [9] explain how machine learning models analyze telemetry data to identify potential threats and automatically modify security policies to mitigate risks without human intervention. These dynamic policy systems use reinforcement learning approaches to optimize security configurations based on observed attack patterns and system behaviors. In Kubernetes environments, this capability manifests as automated adjustments to NetworkPolicies, PodSecurityPolicies, and RBAC configurations in response to detected anomalies or emerging threats. D'Silva and Ambawade [10] describe implementations where policy enforcement points continuously receive updated security directives based on AI analysis of cluster-wide telemetry data, enabling rapid response to emerging threats while maintaining operational continuity for legitimate workloads.

5.3. Implementation Frameworks for Automated Workload Identity Validation

Automated workload identity validation represents a critical component of Zero Trust architectures in Kubernetes environments. Syed, Shah, et al. [9] describe how cryptographic attestation mechanisms establish workload identity based on multiple factors including deployment source, configuration attributes, runtime behaviors, and service account credentials. These frameworks leverage Kubernetes admission controllers to validate workload identities at deployment time and continue monitoring for potential identity compromise through runtime behavioral analysis. D'Silva and Ambawade [10] outline implementations that use service meshes to enforce mutual TLS authentication between services, ensuring that each communication is authenticated and authorized based on workload identity rather than network location. Advanced implementations incorporate behavioral profiles for each workload type, allowing AI systems to detect when containers deviate from expected behavior patterns—potentially indicating compromise. These workload identity frameworks typically integrate with secrets management systems, certificate authorities, and policy engines to create comprehensive identity validation pipelines.

5.4. Challenges and Solutions for Policy Enforcement Across Multi-Cluster Deployments

Implementing Zero Trust with AI-driven policy enforcement across multi-cluster Kubernetes deployments presents significant technical challenges. Syed, Shah, et al. [9] identify several complexities including cross-cluster identity federation, consistent policy distribution, and heterogeneous security capabilities between clusters in different environments. In multi-cluster scenarios, security policies must be synchronized while respecting cluster-specific constraints and accommodating workload migration between environments. D'Silva and Ambawade [10] describe architectural approaches for addressing these challenges, including federated policy management systems that synchronize security directives across clusters while accounting for local variations in capabilities. These implementations typically establish central policy repositories with distributed enforcement points, enabling consistent security postures across heterogeneous environments. For handling cross-cluster communication, solutions include establishing secure gateways that apply consistent authentication and authorization controls at cluster boundaries, coupled with end-to-end encryption for all traffic. AI systems play crucial roles in these complex environments by identifying policy inconsistencies, detecting cross-cluster attack paths, and recommending policy adjustments to maintain security parity across the distributed infrastructure.

6. Integration Frameworks for AI and eBPF Security Solutions

6.1. Reference Architecture for Combined AI and eBPF Security Monitoring

The integration of artificial intelligence with eBPF-based monitoring represents a powerful approach for comprehensive Kubernetes security. Alex Mathew [11] proposes a reference architecture that combines eBPF's deep system visibility with AI's analytical capabilities, creating multi-layered defense systems for containerized environments. This architecture typically consists of eBPF agents deployed on each node in the Kubernetes cluster, collecting low-level telemetry data including system calls, network flows, and process activities. The collected data flows into a centralized analytics platform where machine learning models process and analyze it for security anomalies. Maximilian Bachl, Joachim Fabini, et al. [12] describe how these architectures implement feedback loops between the AI analytics layer and eBPF agents, enabling adaptive monitoring based on detected threats. The reference architecture includes components for data collection, preprocessing, feature extraction, anomaly detection, classification, and response orchestration. This architectural approach enables security teams to detect sophisticated attacks targeting Kubernetes clusters by correlating low-level system behaviors with higher-level application patterns and cluster-wide activities.

6.2. Data Flow Models for Real-time Security Telemetry Processing

Effective real-time security telemetry processing requires sophisticated data flow models that balance completeness with efficiency. Mathew [11] outlines data flow architectures that leverage eBPF's ability to filter and aggregate data in the kernel, reducing the volume of information that must be transferred to user space for AI analysis. These models typically employ stream processing frameworks that handle continuous data flows from distributed eBPF agents across the Kubernetes cluster. Bachl, Fabini, et al. [12] describe implementations that process telemetry data through multiple stages, including normalization, enrichment with contextual information, feature extraction, and batch aggregation for model training. Advanced implementations employ time-series analysis techniques to identify temporal patterns in security events that might indicate coordinated attacks. The data flow models must address the challenge of maintaining context across ephemeral container lifecycles, ensuring that security analytics can track entities despite the dynamic nature of Kubernetes environments. These systems typically employ distributed streaming platforms that provide scalability, fault tolerance, and low-latency processing capabilities essential for real-time threat detection.

6.3. Deployment Considerations for Enterprise Kubernetes Environments

Deploying integrated AI and eBPF security solutions in enterprise Kubernetes environments presents unique challenges that must be addressed through careful planning. Mathew [11] emphasizes the importance of implementing these solutions with minimal disruption to existing workloads and infrastructure. Deployment architectures typically involve DaemonSets to ensure eBPF agents run on every node, with careful resource allocation to prevent performance impacts on production applications. Enterprise implementations must address compatibility issues across diverse Kubernetes distributions, kernel versions, and container runtimes. Bachl, Fabini, et al. [12] highlight considerations around regulatory compliance, data sovereignty, and privacy implications when deploying these monitoring solutions, particularly in multi-tenant environments where workloads may have different security requirements. Integration with existing security operations workflows, including SIEM systems and incident response procedures, represents another critical deployment consideration. Enterprise implementations typically include phased rollout strategies, beginning with monitoring mode before progressing to enforcement actions, allowing security teams to validate the solution's effectiveness while minimizing operational risks.

6.4. Operational Challenges and Performance Optimization Techniques

Operating integrated AI and eBPF security solutions at scale presents several challenges requiring specific optimization techniques. Mathew [11] identifies key operational challenges including managing the resource footprint of eBPF programs, handling the computational requirements of real-time machine learning inference, and maintaining performance during security incidents. Performance optimization techniques include selective instrumentation that focuses eBPF monitoring on high-risk system calls and activities rather than comprehensive tracing. Bachl, Fabini, et al. [12] describe optimizations for the AI components, including model quantization, feature selection to reduce dimensionality, and edge inferencing that distributes analytical workloads across the cluster. Operational challenges also include managing false positives that can lead to alert fatigue, requiring continuous model training and threshold adjustments based on feedback from security analysts. Advanced implementations employ adaptive sampling techniques that increase monitoring granularity for suspicious workloads while reducing overhead for trusted applications. These optimizations ensure that security monitoring doesn't significantly impact application performance or resource availability—a critical requirement for production Kubernetes environments where security solutions must operate efficiently alongside business-critical workloads.

7. Conclusion

The integration of AI-driven threat detection and eBPF-based security monitoring represents a significant advancement in Kubernetes security, addressing the unique challenges posed by containerized environments. This article has examined how machine learning approaches can analyze complex telemetry data to identify anomalous behaviors that traditional security tools might miss, while eBPF provides unprecedented kernel-level visibility without compromising performance. The combination of these technologies enables a comprehensive Zero Trust implementation that continuously validates workload identities and dynamically adjusts security policies based on real-time threat intelligence. However, successful implementation requires careful consideration of architectural design, deployment strategies, and performance optimization techniques to ensure effective security without disrupting application functionality. As organizations continue to adopt Kubernetes for mission-critical workloads, these advanced security approaches will become increasingly essential for protecting containerized applications against sophisticated threats while maintaining the operational benefits that drive container adoption. Future research should focus on improving the accuracy of AI detection models, reducing false positives, enhancing cross-cluster policy synchronization, and developing standardized integration frameworks that security teams can implement regardless of their specific Kubernetes distribution or infrastructure environment.

References

[1] Shu Sekigawa, Chikara Sasaki, et al., "Toward a Cloud-Native Telecom Infrastructure: Analysis of Kubernetes Adaptation," IEEE Xplore, 12 January 2023. https://ieeexplore.ieee.org/document/10008579/authors#authors

[2] Yutian Yang, Wenbo Shen, et al., "Security Challenges in the Container Cloud," IEEE Conference Publication, 14 April 2022. https://ieeexplore.ieee.org/abstract/document/9750244/authors#authors

[3] Chris Binnie, Rory McCune, "Kubernetes External Attacks," IEEE Xplore (Part of Cloud Native Security), 2021. https://ieeexplore.ieee.org/abstract/document/9932401

[4] Md Shazibul Islam Shamim, Farzana Ahamed Bhuiyan, et al., "XI Commandments of Kubernetes Security: A Systemization of Knowledge Related to Kubernetes Security Practices," IEEE SecDev, 2020. https://secdev.ieee.org/wp-content/uploads/2020/11/s4-02-shamim.pdf

[5] Ali Bou Nassif, Manar Abu Talib, et al., "Machine Learning for Anomaly Detection: A Systematic Review," IEEE Access, 24 May 2021. https://ieeexplore.ieee.org/document/9439459/metrics#metrics

[6] Vasco Samuel Carvalho, Maria João Polidoro, et al., "OwlSight: Platform for Real-Time Detection and Visualization of Cyber Threats," IEEE Conference Publication, 04 July 2016. https://ieeexplore.ieee.org/abstract/document/7502265

[7] Simon Sundberg, Anna Brunstrom, "Efficient Continuous Latency Monitoring with eBPF," Lecture Notes in Computer Science (LNCS), March 10, 2023. https://link.springer.com/chapter/10.1007/978-3-031-28486-1_9

[8] Songi Gwak, Thien-Phuc Doan, et al., "Container Instrumentation and Enforcement System for Runtime Security of Kubernetes Platform with eBPF," Intelligent Automation & Soft Computing, June 21, 2023. https://www.techscience.com/iasc/v37n2/53245

[9] Naeem Firdous Syed, Syed W. Shah, et al., "Zero Trust Architecture (ZTA): A Comprehensive Survey," IEEE Access, June 3, 2022. https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?arnumber=9773102

[10] Daniel D'Silva, Dayanand D. Ambawade, "Building a Zero Trust Architecture Using Kubernetes," IEEE Conference Publication, 10 May 2021. https://ieeexplore.ieee.org/abstract/document/9418203/keywords#keywords

[11] Alex Mathew, "AI Cyber Defense and eBPF," World Journal of Advanced Research and Reviews, April 29, 2024. https://wjarr.com/sites/default/files/WJARR-2024-1305.pdf

[12] Maximilian Bachl, Joachim Fabini, et al., "A Flow-Based IDS Using Machine Learning in eBPF," arXiv.org, February 2021. https://arxiv.org/pdf/2102.09980