Review

Not peer-reviewed version

Kubernetes Rooms: A Comprehensive Review of Multi-Tenancy, Privacy, and Security Approaches

Ratana Soth * and Leangsiv Sok

Posted Date: 9 June 2026

doi: 10.20944/preprints202606.0625.v1

Keywords: Kubernetes; multi-tenancy; container isolation; privacy; confidential computing; eBPF; RBAC; network policy; runtime security; supply chain security; cloud-native computing

 doi:10.20944/preprints202606.0625.v1

Review

Kubernetes Rooms: A Comprehensive Review of Multi-Tenancy, Privacy, and Security Approaches

Ratana Soth * and Leangsiv Sok

Paragon International University, Phnom Penh, Cambodia
* Correspondence: rsoth@paragoniu.edu.kh

Abstract

Kubernetes has become one of the most important platforms for deploying and managing modern cloud-native applications. Its ability to automate container orchestration, scale services, and support distributed workloads has made it a central technology in enterprise cloud infrastructure. However, as more organizations place multiple users, teams, applications, and business units inside shared Kubernetes clusters, the challenges of multi-tenancy, privacy, and security become increasingly serious. Although Kubernetes provides native mechanisms such as namespaces, Role-Based Access Control (RBAC), network policies, resource quotas, and admission controllers, these mechanisms mainly provide logical separation. In highly regulated, hostile, or mutually distrusting environments, such soft boundaries may not be sufficient. Recent research has therefore explored stronger approaches, including virtual clusters, hardware-based Trusted Execution Environments (TEEs), eBPF-based runtime monitoring, service mesh encryption, formal policy verification, and automated misconfiguration detection. This paper presents a comprehensive review of Kubernetes multi-tenancy, privacy, and security research published between 2021 and 2026. The review is organized into three major pillars: multi-tenant isolation, privacy protection, and cluster security. For each pillar, this paper discusses the technical evolution of the field, summarizes representative studies, and compares the main approaches based on isolation strength, overhead, maturity, automation level, and practical limitations. The paper also identifies cross-cutting primitives, including eBPF, RBAC, network policy, admission control, and formal verification, that appear repeatedly across the three pillars. Finally, the paper discusses open research challenges and highlights future directions for building more secure, privacy-aware, and resource-efficient Kubernetes environments.

Keywords: Kubernetes; multi-tenancy; container isolation; privacy; confidential computing; eBPF; RBAC; network policy; runtime security; supply chain security; cloud-native computing

1. Introduction

Kubernetes has become the de facto orchestration platform for containerized workloads. In recent years, it has moved from being a cloud-native tool used mainly by early adopters to becoming a core infrastructure platform for enterprises, universities, service providers, and government systems. The Cloud Native Computing Foundation reported that Kubernetes adoption exceeded 96% among enterprise deployments as of 2024 [1]. This level of adoption shows that Kubernetes is no longer only a deployment technology. It has become part of the operational foundation of modern digital infrastructure.

As Kubernetes clusters grow larger and serve more workloads, organizations increasingly use shared clusters to reduce infrastructure cost, improve resource utilization, and simplify operations. This shared-cluster model naturally leads to multi-tenancy. In a multi-tenant Kubernetes environment, different users, applications, departments, or external customers may run workloads on the same underlying infrastructure. This improves efficiency, but it also introduces serious risks. Tenants may

 doi:10.20944/preprints202606.0625.v1

accidentally interfere with one another, consume excessive resources, access sensitive data, or exploit misconfigurations to move laterally across the cluster.

Kubernetes provides several built-in mechanisms to support isolation. Namespaces separate resources into logical scopes. RBAC defines which users or service accounts can perform specific actions on Kubernetes resources. Network policies restrict traffic between pods and services. Resource quotas and LimitRanges control how much CPU, memory, and storage a namespace can consume. Admission controllers can reject or modify workloads before they are accepted into the cluster. These mechanisms are useful and widely deployed, but they mainly enforce soft isolation. They do not fully separate the Kubernetes control plane, and they do not remove the risks created by misconfiguration, overly permissive access, shared kernel exposure, or vulnerable container images [2,3].

The privacy problem in Kubernetes is equally important. Many workloads process sensitive data, including personal information, business secrets, health data, financial records, research datasets, and machine learning training data. Privacy protection in Kubernetes must address three states of data: data at rest, data in transit, and data in use. Encryption at rest protects stored data such as Kubernetes Secrets and etcd records. Mutual TLS protects data moving between services. Trusted Execution Environments protect data while it is being processed in memory. However, none of these protections are fully automatic. They require correct configuration, operational discipline, and in many cases additional technologies beyond default Kubernetes [4–6].

Security research in Kubernetes has also expanded rapidly. The attack surface is broad and includes the container supply chain, Helm charts, Kubernetes manifests, the API server, RBAC permissions, etcd, the container runtime, the Linux kernel, and the pod network. Studies such as EPScan have shown that excessive RBAC permissions are common in real Kubernetes applications and can lead to exploitable privilege escalation [9]. At the same time, runtime security tools based on eBPF have become increasingly important because they allow low-overhead monitoring and enforcement at the kernel level [10,11].

The central argument of this paper is that Kubernetes multi-tenancy, privacy, and security should not be studied as separate problems. In practice, these three areas are deeply connected. A weak RBAC policy can break tenant isolation. A missing network policy can violate privacy assumptions. A compromised container can become a security threat to the entire shared cluster. Therefore, a complete understanding of Kubernetes in modern infrastructure requires a combined analysis of all three pillars.

The main contributions of this paper are as follows:

- It reviews Kubernetes multi-tenancy, privacy, and security research published recently.
- It organizes the literature into three major categories: multi-tenant isolation, privacy protection, and cluster security.
- It compares representative approaches based on technical strength, maturity, operational complexity, and limitations.
- It identifies cross-cutting primitives that appear across all three categories.
- It discusses open research challenges that remain unresolved in current Kubernetes research and practice.

The rest of the paper is organized as follows. Section II provides background on Kubernetes architecture and its native isolation, privacy, and security mechanisms. Section III explains the review methodology. Sections IV, V, and VI review multi-tenancy, privacy, and security respectively. Section VII presents cross-category synthesis. Section VIII discusses open research challenges. Section IX concludes the paper.

2. Background

2.1. Kubernetes Architecture

Kubernetes is an open-source platform for automating the deployment, scaling, and management of containerized applications. A Kubernetes cluster is normally divided into a control plane and a set of worker nodes. The control plane manages the desired state of the cluster. It includes the API server,

 doi:10.20944/preprints202606.0625.v1

etcd, scheduler, and controller manager. The API server is the main entry point for all administrative and workload-related actions. etcd stores the cluster state. The scheduler assigns pods to worker nodes. The controller manager continuously reconciles the actual state of the cluster with the desired state [2].

Worker nodes run the actual application workloads. Each worker node normally includes the kubelet, kube-proxy, and a container runtime such as containerd or CRI-O. The kubelet communicates with the control plane and ensures that pods are running as expected. The container runtime pulls images and starts containers. The network layer connects pods across nodes and allows services to communicate.

Kubernetes follows a declarative model. Users describe what they want the system to run, and Kubernetes attempts to make the actual cluster match that description. This model is powerful, but it also means that any mistake in configuration can be automatically propagated across the cluster. A poorly written manifest, excessive RBAC permission, or insecure admission policy can therefore create significant risk.

2.2. Native Isolation Mechanisms

Kubernetes provides several mechanisms for isolation. Namespaces are the most common logical isolation mechanism. They allow teams or tenants to separate resources such as pods, services, ConfigMaps, and Secrets. However, namespaces do not create hard security boundaries. They do not isolate the underlying kernel, control plane, or physical infrastructure.

RBAC controls access to Kubernetes API resources. It defines which subjects can perform which actions on which resources. For example, one service account may be allowed to read pods but not delete deployments. RBAC is essential for access control, but it is also one of the most common sources of misconfiguration. Overly broad roles, wildcard permissions, and excessive service account privileges can create paths for privilege escalation [3,9].

Network policies define how pods can communicate with one another. In a secure multi-tenant cluster, each namespace should normally start with a default-deny policy, and only required traffic should be explicitly allowed. However, network policies depend on the CNI plugin, and they are only effective when written correctly. A permissive or incomplete policy can still allow lateral movement between tenants [17,20].

Admission controllers add another layer of control. Tools such as OPA/Gatekeeper and Kyverno can enforce security and compliance rules before workloads are admitted into the cluster. For example, they can reject privileged containers, enforce image signing, require resource limits, or block workloads that violate tenant policies. Admission control is powerful, but it is mainly a pre-deployment mechanism. It must be combined with runtime monitoring to detect violations after workloads start running.

2.3. Multi-Tenancy Models

Kubernetes multi-tenancy can be understood through two broad models: soft multi-tenancy and hard multi-tenancy.

Soft multi-tenancy uses a shared Kubernetes control plane. Tenants are separated using namespaces, RBAC, network policies, quotas, and admission controllers. This model is efficient and relatively easy to operate. It is widely used in organizations where tenants are trusted or semi-trusted, such as internal teams within the same company. However, it provides limited protection against hostile tenants or severe misconfigurations.

Hard multi-tenancy provides stronger isolation by separating tenants at the control-plane, runtime, or infrastructure level. This can be achieved through dedicated clusters, virtual clusters, lightweight control planes, sandboxed runtimes, or VM-based isolation. Hard multi-tenancy provides stronger boundaries but usually introduces higher resource overhead and greater operational complexity [13,14,24].

 doi:10.20944/preprints202606.0625.v1

Recent research has attempted to combine the efficiency of soft multi-tenancy with the stronger guarantees of hard multi-tenancy. Virtual cluster systems, such as vCluster, KubeFlex, and Pyramid, are examples of this direction. They attempt to provide each tenant with a more isolated Kubernetes experience without requiring a full physical cluster for every tenant.

2.4. Privacy Protection Layers

Privacy protection in Kubernetes must consider data at rest, data in transit, and data in use.

Data at rest includes stored data such as Kubernetes Secrets, ConfigMaps, persistent volumes, and etcd records. Kubernetes Secrets are base64-encoded by default, but base64 encoding is not encryption. For stronger protection, Kubernetes must be configured to use encryption at rest, preferably with envelope encryption and an external Key Management Service [35].

Data in transit refers to information moving between pods, services, nodes, and external systems. Service meshes such as Istio, Linkerd, and Cilium can enforce mTLS between services. This provides encryption and mutual authentication, helping prevent interception and impersonation [15,34].

Data in use refers to data being processed in memory. This is the most difficult state to protect. Trusted Execution Environments, including Intel SGX, AMD SEV, ARM CCA, and Kata-based confidential containers, attempt to protect workloads even when the host or cloud provider is not fully trusted [4,31,40].

2.5. Kubernetes Security Surface

The Kubernetes security surface spans multiple layers. At the supply chain layer, threats include malicious container images, vulnerable dependencies, unsigned artifacts, and insecure Helm charts. At the control plane layer, threats include exposed API servers, weak RBAC, insecure etcd access, and excessive service account permissions. At the runtime layer, threats include container escape, syscall abuse, privilege escalation, and shared-kernel exploitation. At the network layer, threats include pod spoofing, lateral movement, service impersonation, and policy bypass [7,16,17].

This layered attack surface explains why no single security mechanism is enough. Kubernetes security requires defense in depth. Static scanning, admission control, runtime monitoring, network enforcement, access control, and formal verification all play different roles.

3. Review Methodology

This paper conducts a focused literature review of Kubernetes research published recently between 2021 and 2026. The review focuses on studies related to multi-tenancy, privacy, and security in Kubernetes and cloud-native environments.

The main sources consulted include IEEE Xplore, ACM Digital Library, ScienceDirect/Elsevier, Springer Link, arXiv, Semantic Scholar, CNCF technical materials, and stable institutional technical reports. Search terms combined “Kubernetes” with technical keywords such as “multi-tenancy,” “namespace isolation,” “virtual cluster,” “confidential computing,” “federated learning privacy,” “RBAC,” “eBPF,” “runtime security,” “supply chain,” “misconfiguration,” and “intrusion detection.”

The inclusion criteria were as follows:

1. The work was published recently between 2021 and 2026.
2. The work focused directly on Kubernetes, containers, or cloud-native infrastructure.
3. The work contributed to multi-tenant isolation, privacy protection, or cluster security.
4. The work was published in a recognized academic venue, technical repository, or stable institutional source.
5. The work included a technical contribution, evaluation, architectural analysis, or systematic discussion.

The final corpus was organized into three categories: multi-tenancy, privacy, and security. Each category was reviewed through a technical narrative, a literature table, and a comparative analysis. The

 doi:10.20944/preprints202606.0625.v1

paper also evaluates technology maturity using Technology Readiness Level to distinguish production-ready mechanisms from early-stage research prototypes.

4. Multi-Tenancy in Kubernetes

4.1. Technical Narrative

Early Kubernetes multi-tenancy research showed that namespaces alone cannot provide sufficient isolation. A namespace can separate objects logically, but it does not isolate the control plane, the kernel, or the network by itself. Therefore, practical multi-tenancy requires several controls to work together: namespaces, RBAC, network policies, quotas, admission control, and runtime restrictions.

Kim et al. evaluated major CNI plugins, including Cilium, Calico, WeaveNet, Kube-router, and Antrea. Their study showed that Cilium can achieve high throughput under L3/L4 policy enforcement, making eBPF-based networking practical for performance-sensitive Kubernetes environments [18]. Budigiri et al. similarly showed that Kubernetes network policies can support low-overhead isolation in 5G and edge environments when implemented with efficient datapaths [19].

From 2022 onward, research increasingly focused on cross-layer orchestration. Bufalino et al. showed that coordinated policy orchestration across Kubernetes pods and OpenStack virtual machines can significantly reduce the network attack surface compared with permissive default configurations [20]. Other work proposed hierarchical namespaces and intent-based multi-domain policies to improve isolation across distributed and edge environments [21,22].

After 2024, the field moved toward stronger isolation models. KubeAegis introduced a unified policy management framework that integrates KubeArmor, network policies, RBAC, and OPA admission control into a single management approach [23]. IBM Research proposed KubeFlex, which combines OVN-Kubernetes, KubeVirt, and lightweight K3s control planes to provide stronger tenant isolation [24]. Pyramid introduced a pluggable Kubernetes architecture for stronger control-plane and data-plane isolation while reducing the cost of cluster-per-tenant deployment [14].

A recurring issue in multi-tenancy research is lateral movement. Even when namespaces exist, a compromised pod may still communicate with another tenant's pods if network policies are missing or poorly configured. Bufalino et al. demonstrated that permissive network policies and default-allow communication can create exploitable paths across tenant boundaries [25]. This shows that multi-tenancy is not achieved simply by creating namespaces. It requires correct, consistent, and continuously verified policies.

4.2. Multi-Tenancy Literature

4.3. Comparative Analysis

The literature shows a clear movement from basic namespace-based isolation toward stronger and more integrated approaches. Namespace, RBAC, and NetworkPolicy remain the foundation of production Kubernetes multi-tenancy because they are mature and low-overhead. However, they are not enough for hostile environments. Virtual clusters, pluggable control planes, and unified policy frameworks provide stronger isolation, but they are still less mature and often more complex to operate.

 doi:10.20944/preprints202606.0625.v1

Table 1. Multi-Tenancy Isolation Techniques in Kubernetes

Label	Authors / Year	Venue	Technique	Key Result	Ref.
MT1	Kim et al., 2024	IEEE Access	eBPF CNI; Cilium; Calico	Cilium achieved high throughput under L3/L4 policy enforcement	[18]
MT2	Budigiri et al., 2022	CNCF / ACM	eBPF CNI; network policy	Low-overhead Kubernetes isolation for 5G deployments	[19]
MT3	Bufalino et al., 2022	KU Leuven	Cross-layer policy orchestration	Significant attack surface reduction	[20]
MT4	Anon., 2024	IEEE Access	Hierarchical namespace; virtual cloud	Dynamic edge cloud abstraction through namespace hierarchies	[21]
MT5	IRIS Polito, 2024	IEEE NetSoft	Intent-based multi-domain policy	Cross-cluster network isolation orchestration	[22]
MT6	Kim and Lee, 2024	IEEE Access	KubeAegis; KubeArmor; OPA	Unified security policy management framework	[23]
MT7	IBM Research, 2026	IBM Technical Report	KubeFlex; OVN-Kubernetes; KubeVirt; K3s	Three-dimensional tenant isolation model	[24]
MT8	Bufalino et al., 2025	arXiv	Network misconfiguration analysis	Lateral movement caused by permissive network policies	[25]
MT9	Qin et al., 2026	ACM EuroSys	Pluggable Kubernetes; virtual control plane	Pyramid improved resource efficiency compared with cluster-per-tenant isolation	[14]
MT10	Anon., 2025	ACM	RBAC; NetworkPolicy; sandboxed runtime	Hostile multi-tenancy framework evaluation	[26]
MT11	Her et al., 2024	IEEE Access	eBPF; LSM; syscall filtering	Dynamic syscall filtering for containers	[27]
MT12	Parra-Ullauri et al., 2022	Aalto University	Hard multi-tenancy; 5G; RBAC	Hard tenancy study in local 5G Kubernetes deployment	[28]
MT13	Anon., 2025	IEEE Access	Mixed-runtime pod networking	Runtime flexibility for edge Kubernetes with isolation preservation	[29]
MT14	Cesarano and Natella, 2025	arXiv	API request filtering; attack surface	Workload-tailored API-level hardening	[30]
MT15	Parra-Ullauri et al., 2023	IEEE INFOCOM Workshops	Link-layer isolation; IPsec operator	Federated learning networking with tenant isolation	[5]

Table 2. Comparative Analysis of Multi-Tenancy Isolation Approaches

Approach	Isolation	Overhead	Complexity	TRL	Work
Namespace + RBAC + NetworkPolicy	Medium	Low	Low	8–9	[18,19]
Cross-layer policy orchestration	Medium–High	Low	Medium	6–7	[20,22]
Unified policy framework	High	Low–Medium	Medium	6–7	[23]
Virtual cluster	High	Medium	Medium–High	6–7	[13,24]
Pluggable Kubernetes	Very High	Medium	High	4–5	[14]
eBPF dynamic syscall filtering	Medium	Very Low	Medium	7–8	[27]
Hostile multi-tenancy framework	Experimental	Variable	High	3–4	[26]

 doi:10.20944/preprints202606.0625.v1

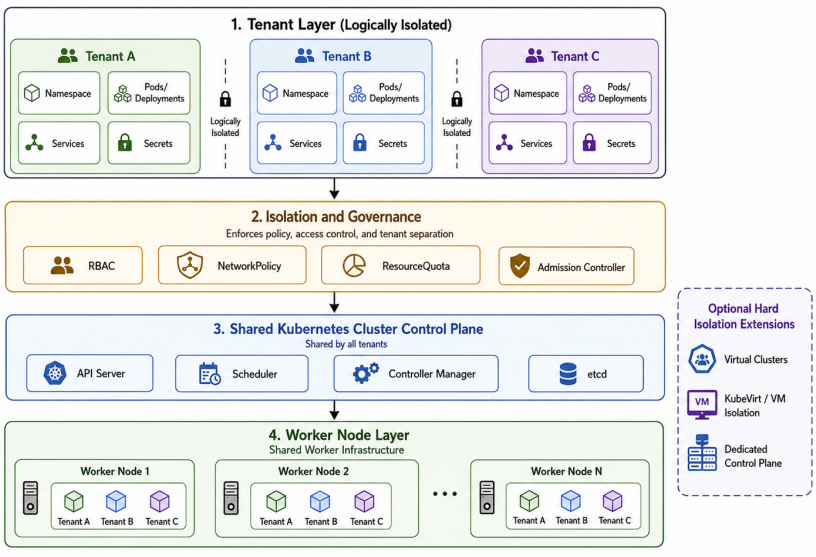

Figure 1. Sample Kubernetes multi-tenancy architecture. The architecture shows three logically isolated tenants running inside a shared Kubernetes cluster. Each tenant has its own namespace, pods, services, and Secrets. The isolation and governance layer applies RBAC, NetworkPolicy, ResourceQuota, and admission control to enforce tenant separation. The shared control plane contains the API server, scheduler, controller manager, and etcd, while worker nodes host workloads from multiple tenants. Optional hard isolation extensions, such as virtual clusters, KubeVirt/VM isolation, and dedicated control planes, can be added when stronger tenant separation is required.

5. Privacy in Kubernetes

5.1. Technical Narrative

Privacy protection in Kubernetes must be considered across the full lifecycle of data. Data may be stored in etcd or persistent volumes, transferred between services, or processed inside containers. Each state requires different protection mechanisms.

Confidential computing has become one of the most important research directions for protecting data in use. The CNCF Confidential Containers project integrates TEEs with the Kubernetes runtime layer, allowing workloads to run inside hardware-protected environments [4]. This reduces the need to fully trust the host operating system, cloud provider, or cluster administrator. However, TEE-based Kubernetes deployment remains technically complex and has not yet reached the same maturity as conventional Kubernetes security controls.

CRISP addressed a specific weakness in TEE-based systems: rollback attacks. Even when memory is protected, an attacker may try to revert persistent state to an earlier version. CRISP introduced rollback protection for confidential cloud-native workloads and showed how such protection can be applied to Kubernetes-based environments [31]. Other research has surveyed SGX, SEV, and ARM CCA as part of a broader movement toward confidential distributed cloud computing [32].

Federated learning is another important privacy-related area. Kubernetes is attractive for federated learning because it can orchestrate distributed clients and workloads. However, the default Kubernetes network model can violate federated learning privacy assumptions if pods can communicate freely. Parra-Ullauri et al. proposed link-layer isolation and IPsec-based protection to improve federated learning privacy in Kubernetes [5]. FedOps extended this direction by providing a Kubernetes-based platform for managing federated learning operations under heterogeneous conditions [33].

For data in transit, service mesh mTLS is the most mature approach. It encrypts communication between services and verifies service identities. This is important in multi-tenant environments because tenant workloads may share the same network fabric [15,34]. For data at rest, etcd encryption and KMS-backed Secret protection are essential. However, these controls are not always enabled by default, and many real-world Kubernetes deployments still contain weak Secret management practices [35,36].

 doi:10.20944/preprints202606.0625.v1

Compliance with GDPR, HIPAA, and similar regulations requires more than enabling encryption. A compliant Kubernetes environment also needs audit logging, least-privilege access, network segmentation, secure Secret management, and clear retention policies. The literature shows that these controls are often missing or inconsistently configured in real deployments [35,36].

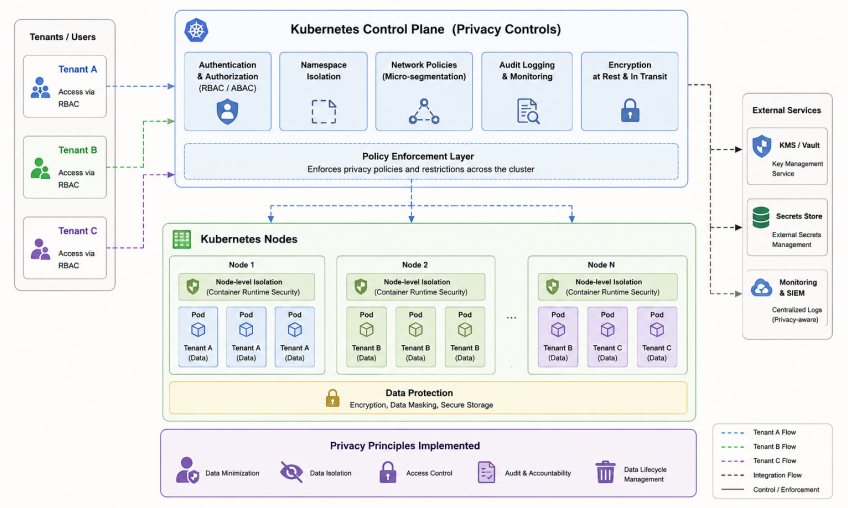

Figure 2. Kubernetes privacy architecture. The architecture shows privacy protection across tenants, the Kubernetes control plane, worker nodes, external services, and monitoring systems. Tenants access workloads through RBAC-controlled identities. Privacy controls include authentication and authorization, namespace isolation, network policies, audit logging, encryption at rest and in transit, and policy enforcement. Worker nodes run tenant pods with node-level isolation and runtime controls. External services, such as KMS/Vault, Secrets stores, and monitoring systems, support secure key management, privacy-aware logging, and compliance reporting. The architecture also highlights privacy principles such as data minimization, data isolation, access control, audit accountability, and data lifecycle management.

5.2. Privacy Literature

Table 3. Privacy Protection Techniques in Kubernetes

Label	Authors / Year	Venue	Technique	Key Result	Ref.
PR1	CNCF Confidential Containers, 2021	CNCF	TEE; SGX; SEV; ARM CCA; CRI	Integration of TEEs with Kubernetes runtime	[4]
PR2	Hartono et al., 2024	IEEE CLOUD	SGX; rollback protection; CRISP	Rollback prevention for confidential cloud-native workloads	[31]
PR3	Anon., 2025	IEEE Access	TEE survey; SGX; SEV; ARM CCA	Survey of privacy and security in distributed cloud computing	[32]
PR4	Parra-Ullauri et al., 2023	IEEE INFOCOM Workshops	Link-layer isolation; IPsec	Privacy-preserving networking for Kubernetes-based federated learning	[5]
PR5	Cheng et al., 2024	IEEE Access	FedOps; federated learning lifecycle	Heterogeneity-aware federated learning operations	[33]
PR6	Anon., 2024	IEEE Access	Remote attestation; edge Kubernetes; mTLS	Attestation-backed trusted edge node enrollment	[34]
PR7	Anon., 2025	IEEE Access	DevSecOps; Secrets; KMS; GDPR	Kubernetes Secret and compliance control analysis	[35]
PR8	Rahman et al., 2023	ACM TOSEM	Static analysis; manifests	Empirical study of Kubernetes misconfigurations	[36]
PR9	Syed et al., 2025	CMC	eBPF; IP spoofing prevention	eBPF-based pod IP spoofing detection	[17]
PR10	Anon., 2024	arXiv	mTLS; service mesh benchmarking	Comparison of service mesh mTLS overhead	[15]
PR11	Kim et al., 2026	CMC	eBPF; hybrid runtime; ML	Hybrid runtime detection using flow and syscall signals	[37]
PR12	Anon., 2025	IEEE Access	eBPF; AI anomaly detection	Adaptive runtime anomaly detection	[11]
PR13	Anon., 2024	IEEE Access	CNI; mTLS; network isolation	Network privacy in multi-tenant Kubernetes	[38]
PR14	Rahman et al., 2025	ACM FSE	Dynamic application security testing	Runtime compliance gap analysis	[39]
PR15	Anon., 2026	arXiv	ARM CCA; pipeline confidentiality	Confidential computing for cloud-native pipelines	[40]

 doi:10.20944/preprints202606.0625.v1

5.3. Comparative Analysis

Privacy approaches in Kubernetes can be divided into hardware-enforced and software-enforced mechanisms. Hardware-enforced approaches, especially TEEs, provide stronger protection for data in use, but they are still emerging in Kubernetes production environments. Software-enforced approaches, such as mTLS, KMS encryption, and static manifest analysis, are more mature and easier to deploy, but they depend heavily on correct configuration and trust in the software stack.

Table 4. Comparative Analysis of Privacy Protection Approaches

Approach	Data State	Fit	TRL	Key Limitation	Work
TEE	In use	High	4–6	Performance overhead and kernel trust gap	[4,31]
ARM CCA realm	In use	High	3–4	Emerging hardware only	[40]
mTLS service mesh	In transit	High	7–8	Does not protect data in use	[15,34]
etcd KMS encryption	At rest	High	7–8	Requires explicit setup	[35]
eBPF + ML anomaly detection	Runtime behavior	Medium	6–7	Evasion and model drift risk	[11,37]
FL link-layer isolation	FL privacy	Medium	5–6	Specific to FL scenarios	[5,33]
Static manifest analysis	Design-time configuration	Medium	7–8	Cannot detect runtime violations	[36,39]

6. Security in Kubernetes

6.1. Technical Narrative

Kubernetes security research has developed rapidly because the platform has a large and complex attack surface. A secure Kubernetes deployment must protect the supply chain, control plane, runtime, and network fabric at the same time.

A full-stack vulnerability analysis published in Computers & Security showed that no single tool can cover all Kubernetes attack surfaces [7]. This is an important finding because it confirms that Kubernetes security cannot depend on one scanner, one policy engine, or one runtime monitor. Instead, security must be layered across the entire platform.

Chained escape attacks demonstrate the danger of weak defense-in-depth. An attacker may begin with a small misconfiguration, such as excessive RBAC permission, then use a runtime vulnerability to escape a container, and finally escalate privileges on the host or control plane. Luo and Zou showed that container-to-cluster compromise can happen through chained vulnerabilities, making layered protection necessary [16].

RBAC remains one of the most critical areas of Kubernetes security. EPScan introduced automated detection of excessive RBAC permissions using pod-oriented program analysis. Its findings showed that excessive permissions are widespread in real Kubernetes applications [9]. KubeKeeper extended this concern to Kubernetes Secrets, showing that overly broad Secret access can create serious privacy and security risks [41].

Research on implicit permissions further showed that not all privilege escalation paths are visible in explicit RBAC rules. Some permissions arise through controller behavior, default service accounts, aggregated roles, and indirect Kubernetes mechanisms [42]. This makes manual RBAC auditing difficult. Formal verification approaches attempt to solve this by modeling access control policies mathematically and checking them before deployment [3].

Runtime security has increasingly moved toward eBPF. eBPF allows security tools to observe kernel-level events, network flows, system calls, and runtime behavior with relatively low overhead. This makes it useful for intrusion detection, anomaly detection, syscall filtering, and network enforcement [10,11]. However, eBPF also introduces a dual-use problem. If attackers gain enough privileges, they may use eBPF itself for stealth, persistence, or kernel-level attack behavior [46].

Supply chain and configuration security are also major concerns. Vulnerable images, insecure Helm charts, weak manifests, and exposed API servers can create serious risks before a workload even starts running. Bui et al. identified several vulnerability classes in real Kubernetes deployments, including exposed APIs, excessive permissions, unencrypted etcd endpoints, and weak network policies [47]. GenKubeSec explored the use of large language models to detect, localize, explain, and

 doi:10.20944/preprints202606.0625.v1

remediate Kubernetes misconfigurations [48]. While promising, LLM-based security analysis still requires caution because of hallucination risk and potential evasion.

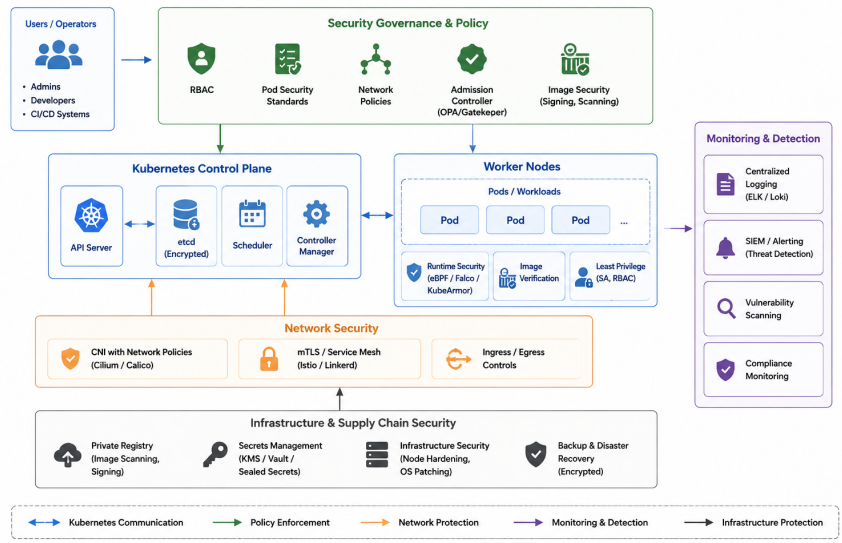

Figure 3. Kubernetes security architecture. The architecture shows a simplified security model for Kubernetes. Users, operators, developers, and CI/CD systems interact with security governance and policy mechanisms, including RBAC, Pod Security Standards, network policies, admission controllers, and image security controls. The Kubernetes control plane protects shared components such as the API server, etcd, scheduler, and controller manager. Worker nodes host pods and workloads with runtime security controls, image verification, and least-privilege service accounts. Network security is enforced through CNI-based network policies, service mesh mTLS, and ingress/egress controls. Infrastructure and supply chain security support private registries, Secrets management, node hardening, patching, and encrypted backup. Monitoring and detection systems provide centralized logging, SIEM alerting, vulnerability scanning, and compliance monitoring.

6.2. Security Literature

Table 5. Security Techniques in Kubernetes

Label	Authors / Year	Venue	Technique	Key Result	Ref.
SE1	Anon., 2023	Computers & Security	Full-stack vulnerability classification	No single tool covers all Kubernetes attack surfaces	[7]
SE2	Luo and Zou, 2025	IEEE Access	Chained escape attack model	Container-to-cluster compromise through chained vulnerabilities	[16]
SE3	Anon., 2025	ACM Computing Surveys	Container security taxonomy	Broad taxonomy of exploits and defenses	[8]
SE4	Zhang et al., 2025	IEEE S&P	Pod-oriented RBAC analysis	EPScan detects exploitable excessive permissions	[9]
SE5	Anon., 2025	IEEE SERVICES	Secret protection; least privilege	KubeKeeper detects excessive permissions on Secrets	[41]
SE6	Anon., 2025	ACM ISSTA	Implicit permission graph	Hidden privilege escalation paths outside explicit RBAC	[42]
SE7	Anon., 2026	IEEE TSE	Helm chart static analysis	Automated RBAC and PSS violation detection	[43]
SE8	Anon., 2025	IEEE Access	Formal verification; SMT solver	Pre-deployment RBAC and ABAC verification	[3]
SE9	Anon., 2025	IEEE Access	eBPF security survey	eBPF security foundations and deployment patterns	[10]
SE10	Anon., 2025	IEEE Access	DeSFAM; eBPF; AI anomaly detection	Adaptive runtime enforcement and anomaly detection	[11]
SE11	Anon., 2025	IEEE Access	eBPF semantic DDoS detection	Kernel tracepoint-based DDoS detection	[44]
SE12	Anon., 2025	IEEE TDSC	Hela; BPF-LSM; syscall restriction	Reduced kernel attack surface	[45]
SE13	Anon., 2024	IEEE ISCC	eBPF-Sec	Defense strategy against malicious eBPF use	[46]
SE14	Bui et al., 2024	IEEE Access	Misconfiguration scanning	Real-world container misconfiguration analysis	[47]
SE15	Anon., 2024	arXiv	LLM-based misconfiguration detection	GenKubeSec detects and explains Kubernetes misconfigurations	[48]

 doi:10.20944/preprints202606.0625.v1

6.3. Comparative Analysis

Kubernetes security research is clearly moving toward automation. Manual review is no longer sufficient for large clusters with many workloads, frequent deployments, and complex policy rules. Automated RBAC analysis, formal verification, Helm chart scanning, runtime monitoring, and LLM-based misconfiguration detection are all attempts to make Kubernetes security more scalable.

However, automation does not remove the need for careful design. Static tools may miss runtime attacks. Runtime tools may miss supply chain problems. LLM tools may generate incorrect explanations. Formal verification may be mathematically strong but difficult to integrate into production workflows. Therefore, the strongest approach remains layered defense.

Table 6. Comparative Analysis of Security Approaches

Approach	Attack Surface	Automation	TRL	Limitation	Work
Full-stack vulnerability scanning	All four layers	Semi-automated	7–8	Tool fragmentation	[7,47]
RBAC formal verification	Control plane	Automated	4–5	Research prototype maturity	[3,42]
LLM misconfiguration detection	Supply chain and control plane	Automated	4–5	Hallucination and evasion risk	[48]
EPScan pod program analysis	Control plane	Automated	5–6	Requires complementary analysis	[9]
eBPF runtime enforcement	Runtime and network	Automated	7–8	Dual-use risk	[10,11]
eBPF semantic DDoS detection	Network	Automated	6–7	Focused on application-layer DDoS	[44]
Helm chart static analysis	Supply chain	Automated	6–7	Limited to chart-level analysis	[43]
Chained attack defense-in-depth	All four layers	Manual + tooling	5–6	Coordination complexity	[16]

7. Cross-Category Synthesis

The most important finding from this review is that Kubernetes multi-tenancy, privacy, and security are not separate technical problems. They overlap at several layers of the system. The same primitive may support isolation, privacy, and security at the same time.

eBPF is the clearest example. In multi-tenancy, eBPF can enforce network policy and support efficient CNI behavior. In privacy, it can monitor network flows and prevent IP spoofing. In security, it can detect suspicious runtime behavior, trace system calls, and enforce policies at the kernel level [17,18,44]. This makes eBPF one of the most important primitives for future Kubernetes security architectures.

RBAC is another cross-cutting primitive. It affects tenant isolation because it controls what tenants can do. It affects privacy because it controls who can access Secrets and sensitive resources. It affects security because excessive RBAC permissions are a major path to privilege escalation [9,41,42]. Therefore, RBAC should not be treated as a simple access-control configuration. It should be treated as a core security and privacy mechanism.

Network policy also appears across all three pillars. It supports tenant separation, protects data flows, and limits lateral movement after compromise. However, the literature repeatedly shows that network policy must be correct, complete, and enforced by a capable CNI plugin. A weak or permissive network policy can create a false sense of security [20,25,47].

Admission control plays a preventive role. It can block insecure workloads before they enter the cluster. It can enforce image provenance, deny privileged containers, require resource limits, and ensure that tenant policies are followed. However, admission control alone cannot detect runtime drift. It must be combined with monitoring and continuous verification.

 doi:10.20944/preprints202606.0625.v1

Table 7. Research Volume and Venue Distribution by Category

Metric	Multi-Tenancy	Privacy	Security
Total papers reviewed	15	15	15
IEEE journals/conferences	10	10	12
ACM venues	2	2	4
Elsevier	0	0	1
arXiv technical papers	2	2	1
Technical report / thesis	1	1	0
Peak publication year	2024–2025	2023–2025	2024–2025

Table 8. Technology Readiness Level Assessment

Technology	Category	TRL	Rationale
Namespace + RBAC + NetworkPolicy	Multi-tenancy	8–9	Widely used in production Kubernetes
Unified policy framework	Multi-tenancy	6–7	Validated in research and early enterprise settings
Virtual cluster	Multi-tenancy	5–6	Promising but not fully mature for hostile tenancy
mTLS service mesh	Privacy	7–8	Mature and widely deployed
etcd KMS encryption	Privacy	7–8	Available but requires explicit configuration
Confidential Containers	Privacy	4–6	Technically promising but still maturing
eBPF runtime security	Security	7–8	Increasingly adopted in production environments
RBAC formal verification	Security	3–5	Strong research value but prototype maturity
LLM misconfiguration detection	Security	4–5	Promising but not yet production-proven

Table 9. Cross-Cutting Primitive Overlap Matrix

Primitive	Multi-Tenancy	Privacy	Security	Tools
eBPF enforcement	Yes	Yes	Yes	Cilium / Tetragon
RBAC policy	Yes	Yes	Yes	EPScan / KubeKeeper
Network policy	Yes	Yes	Yes	Cilium / Calico
Admission controller	Yes	Yes	Yes	OPA/Gatekeeper / Kyverno
TEE runtime	Limited	Yes	Yes	Confidential Containers
Formal verification	Yes	Limited	Yes	SMT / Z3

8. Open Research Challenges

8.1. Formal Privacy Verification for Tenant Boundaries

Virtual cluster systems and pluggable Kubernetes architectures provide stronger separation than basic namespaces, but the literature still lacks formal proof that these boundaries fully prevent cross-tenant leakage. Current systems may isolate APIs and resources, but questions remain about shared storage, shared nodes, scheduling side channels, metadata leakage, and kernel-level interactions.

Future research should develop formal models that can verify tenant boundaries across the control plane, data plane, runtime, and storage layer. This is especially important for hostile multi-tenancy, where tenants may actively attempt to observe, infer, or attack one another [3,14,26].

 doi:10.20944/preprints202606.0625.v1

8.2. Privacy-Preserving Audit Logging

Audit logging is necessary for accountability and compliance, but it can also expose sensitive metadata. Kubernetes audit logs may reveal namespace names, resource names, request patterns, timestamps, user agents, and IP addresses. Even if the actual workload data is encrypted, audit metadata can still reveal relationships between services, operational behavior, and tenant activity.

A major open problem is how to design audit logging that supports investigation and compliance while minimizing metadata leakage. This challenge is especially difficult because regulators require traceability, while privacy goals require minimization.

8.3. Adversarially Robust Intrusion Detection in Shared Clusters

Many eBPF-based and ML-based detection systems perform well in controlled environments, but shared multi-tenant clusters introduce adversarial conditions. A malicious tenant may intentionally generate traffic or system behavior designed to evade detection. The attacker may also observe system responses and adjust behavior over time.

Future intrusion detection systems must therefore be robust against adaptive tenants. This requires adversarially trained models, tenant-aware baselines, stronger telemetry isolation, and continuous model validation [11,37].

8.4. Unified Policy Framework Across All Three Pillars

Current Kubernetes policy enforcement is fragmented. Multi-tenancy is handled through namespaces, quotas, network policies, and virtual clusters. Privacy is handled through mTLS, KMS encryption, TEEs, and audit logging. Security is handled through RBAC tools, scanners, admission controllers, and runtime monitors. These tools often use different policy languages, different assumptions, and different enforcement points.

A unified framework that reasons about isolation, privacy, and security together would be highly valuable. Such a framework should understand how policies interact and should be able to detect conflicts, gaps, and violations across the full Kubernetes stack. KubeAegis is a step in this direction, but it does not yet fully integrate privacy mechanisms such as TEE attestation, audit metadata protection, or differential privacy [23].

9. Conclusions

This paper reviewed Kubernetes multi-tenancy, privacy, and security research from 2021 to 2026. The review shows that Kubernetes has become a powerful platform for shared cloud-native infrastructure, but its default mechanisms are not sufficient for all multi-tenant, privacy-sensitive, or adversarial environments.

In multi-tenancy, the field has moved from namespace-based soft isolation toward virtual clusters, pluggable architectures, and unified policy frameworks. Namespaces, RBAC, and NetworkPolicy remain essential, but they must be combined with stronger controls for hostile or regulated environments.

In privacy, the literature shows two major directions. Software-based controls such as mTLS, KMS-backed etcd encryption, and static manifest analysis are more mature and easier to deploy. Hardware-based approaches such as TEEs and confidential containers provide stronger data-in-use protection but remain less mature in large-scale Kubernetes production environments.

In security, the field is moving toward automation. RBAC analysis, formal verification, Helm chart scanning, eBPF runtime enforcement, and LLM-based misconfiguration detection all show that manual security auditing is no longer sufficient for modern Kubernetes environments. However, each mechanism has limitations, and production security still requires layered defense.

The major future research opportunities lie at the intersection of the three pillars. Formal tenant boundary verification, privacy-preserving audit logging, adversarially robust intrusion detection, and unified cross-pillar policy enforcement remain open and important challenges. These challenges

 doi:10.20944/preprints202606.0625.v1

are especially relevant for massive multi-tenant Kubernetes environments where isolation, privacy, security, resource consumption, computation time, and energy efficiency must be optimized together.

References

1. Cloud Native Computing Foundation, "Annual Survey 2024: Cloud Native Computing Foundation Report," CNCF, 2024.
2. The Linux Foundation, "Kubernetes Documentation: Concepts — Cluster Architecture," Kubernetes Documentation, 2024. [Online]. Available: https://kubernetes.io/docs/concepts/
3. "Formal Verification for Preventing Misconfigured Access Policies in Kubernetes," *IEEE Access*, vol. 13, 2025, doi: 10.1109/ACCESS.2025.11122676.
4. CNCF Confidential Containers Project, "Confidential Containers: Overview and Architecture," CNCF GitHub, 2021.
5. J. Parra-Ullauri, A. Bravalheri, A. Ramírez, X. Wu, R. Nejabati, and D. Simeonidou, "Privacy Preservation in Kubernetes-Based Federated Learning: A Networking Approach," in *Proc. IEEE INFOCOM Workshops*, 2023, doi: 10.1109/INFOCOMWKSHPS57453.2023.10225876.
6. ARMO, "Kubernetes Compliance Under GDPR," ARMO, 2025. [Online]. Available: https://www.armosec.io
7. "Full-Stack Vulnerability Analysis of the Cloud-Native Platform," *Computers & Security*, vol. 133, 2023, doi: 10.1016/j.cose.2023.103173.
8. "A Container Security Survey: Exploits, Attacks, and Defenses," *ACM Computing Surveys*, 2025, doi: 10.1145/3715001.
9. Y. Zhang, L. Zhang, Z. Zhang, G. Hong, Y. Zhang, and M. Yang, "EPScan: Automated Detection of Excessive RBAC Permissions in Kubernetes Applications," in *Proc. IEEE Symposium on Security and Privacy*, pp. 3199–3217, 2025, doi: 10.1109/SP61157.2025.00011.
10. "An In-Depth Analysis of eBPF-Based System Security Tools in Cloud-Native Environments," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.11146725.
11. "DeSFAM: An Adaptive eBPF and AI-Driven Framework for Securing Containerized Applications," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.11095719.
12. "Securing the Shared Kernel: Exploring Kernel Isolation and Mitigation Techniques," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.10769445.
13. CNCF and Loft Labs, "Solving Kubernetes Multi-Tenancy Challenges with vCluster," CNCF Blog, 2025.
14. Y. Qin et al., "Pyramid: A Secure, Resource-Efficient, and Pluggable Kubernetes System for Multi-Tenancy," in *Proc. ACM European Conference on Computer Systems*, 2026, doi: 10.1145/3767295.3769324.
15. "Performance Comparison of Service Mesh Frameworks: The mTLS Test Case," arXiv:2411.02267, 2024.
16. Y. Luo and B. Zou, "From Container to Cluster: Chained Escape Attacks in Kubernetes and Orchestration Platforms," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.11223725.
17. H. J. Syed et al., "Preventing IP Spoofing in Kubernetes Using eBPF," *Computers, Materials & Continua*, 2025, doi: 10.32604/cmc.2025.062628.
18. D. Kim et al., "Exploring Security Enhancements in Kubernetes CNI: A Deep Dive into Cilium, Calico, WeaveNet, Kube-Router, and Antrea," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.10896680.
19. S. Budigiri et al., "Network Policies in Kubernetes: Performance Evaluation and Security Assessment," in *Proc. CNCF KubeCon / ACM*, 2022.
20. J. Bufalino et al., "Elastic Cross-Layer Orchestration of Network Policies in the Cloud," KU Leuven / Lirias, 2022.
21. "A Hierarchical Namespace Approach for Multi-Tenancy in Distributed Clouds," *IEEE Access*, vol. 12, pp. 32597–32617, 2024, doi: 10.1109/ACCESS.2024.3297296.
22. IRIS Polito, "An Intent-Based Solution for Network Isolation in Kubernetes Multi-Domain Environments," in *Proc. IEEE Conference on Network Softwarization*, 2024.
23. B. Kim and S. Lee, "KubeAegis: A Unified Security Policy Management Framework for Containerized Environments," *IEEE Access*, vol. 12, pp. 160636–160652, 2024, doi: 10.1109/ACCESS.2024.3486772.
24. IBM Research, "Three Shades of Isolation: A Multi-Tenancy Fortress Architecture," IBM Research Technical Report, 2026.
25. J. Bufalino et al., "Defending Kubernetes Clusters Against Lateral Movement via Network Misconfigurations," arXiv:2506.21134, 2025.
26. "Towards Enabling Hostile Multi-Tenancy in Kubernetes," in *Proc. ACM Conference*, 2025, doi: 10.1145/3731599.3767357.

 doi:10.20944/preprints202606.0625.v1

27. J. Her et al., "KubeRosy: A Dynamic System Call Filtering Framework for Containers," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3486772.
28. J. Parra-Ullauri et al., "Hard Multi-Tenancy Kubernetes Approaches in a Local 5G Deployment," M.S. thesis, Aalto University, 2022.
29. "Mixed-Runtime Pod Networking for Kubernetes-Based Edge Deployments," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.11257535.
30. C. Cesarano and R. Natella, "KubeFence: Security Hardening of the Kubernetes Attack Surface," arXiv:2504.11126, 2025.
31. A. P. P. Hartono, A. Brito, and C. Fetzer, "CRISP: Confidentiality, Rollback, and Integrity Storage Protection for Confidential Cloud-Native Computing," in *Proc. IEEE CLOUD*, 2024, doi: 10.1109/CLOUD62652.2024.00026.
32. "A Survey on Privacy and Security in Distributed Cloud Computing," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.10963898.
33. Y. Cheng et al., "FedOps: A Platform of Federated Learning Operations with Heterogeneity Management," *IEEE Access*, vol. 12, pp. 4301–4314, 2024, doi: 10.1109/ACCESS.2024.3349691.
34. "Trusting the Cloud-Native Edge: Remotely Attested Kubernetes Worker Nodes," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.10637515.
35. "Extensive Review of Threat Models for DevSecOps Environments in Kubernetes," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.10909533.
36. A. Rahman, S. Islam, and D. Bose, "Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study," *ACM Transactions on Software Engineering and Methodology*, 2023, doi: 10.1145/3579639.
37. S. Kim et al., "Hybrid Runtime Detection of Malicious Containers Using eBPF," *Computers, Materials & Continua*, 2026, doi: 10.32604/cmc.2025.074871.
38. "Exploring Security Enhancements in Kubernetes CNI for Multi-Tenant Data Privacy," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.10380563.
39. A. Rahman et al., "Dynamic Application Security Testing for Kubernetes Deployment," in *Proc. ACM International Conference on the Foundations of Software Engineering*, 2025, doi: 10.1145/3729364.
40. "Mica: Confidential Computing Architecture on ARM CCA for Cloud-Native Pipelines," arXiv:2603.03403, 2026.
41. "KubeKeeper: Protecting Kubernetes Secrets Against Excessive Permissions," in *Proc. IEEE SERVICES*, 2025, doi: 10.1109/SERVICES63567.2025.11129402.
42. "Understanding and Exploiting Implicit Permissions in Kubernetes," in *Proc. ACM International Symposium on Software Testing and Analysis*, 2025, doi: 10.1145/3719027.3765106.
43. "Automated Analysis of Security Policy Violations in Helm Charts," *IEEE Transactions on Software Engineering*, 2026, doi: 10.1109/TSE.2026.11223885.
44. "eBPF-Based Runtime Detection of Semantic DDoS Attacks in Linux Containerized Environments," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.11179973.
45. "Hela: A System Call Restriction Framework for Protecting the Entire Container Lifecycle," *IEEE Transactions on Dependable and Secure Computing*, 2025, doi: 10.1109/TDSC.2025.11456734.
46. "eBPF-Sec: A Defensive Framework Against eBPF Attacks on Containers," in *Proc. IEEE Symposium on Computers and Communications*, 2024, doi: 10.1109/ISCC61673.2024.10733575.
47. T. Bui et al., "A Study on Misconfigured Container Components in the Wild," *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.10788674.
48. "GenKubeSec: LLM-Based Kubernetes Misconfiguration Detection, Localization, Reasoning, and Remediation," arXiv:2405.19954, 2024.