The American Journal of Engineering and Technology
ISSN 2689-0984 | Open Access

TYPE Original Research
PAGE NO. 268-274
DOI 10.37547/tajet/Volume07Issue08-22

Data Security in Multi-Tenant Clusters

Megha Aggarwal
Software Development Engineer, Amazon AWS Seattle, WA, USA

Abstract: This article presents a comprehensive analysis of the set of threats that are characteristic of heterogeneous Kubernetes deployments. The work aims to systematize and examine these threats, as well as to develop an integrated security model suitable for practical implementation. The methodological foundation consisted of a rigorous literature review encompassing both academic papers and engineering reports from major cloud providers. Special attention was given to publications on container isolation, inter-pod network policy, secrets management, and data encryption protocols. Based on this analysis, a multi-layer threat map is presented, detailing the attack vectors at each layer. The proposed protective measures are integrated into a unified DevSecOps lifecycle framework and can be automated within CI/CD pipelines. The conclusions drawn and the model developed are intended for security engineers, DevOps teams, and cloud platform architects who need to design and maintain multi-tenant Kubernetes clusters with a guaranteed level of data protection.

Keywords: Kubernetes, data security, multi-tenancy, isolation, threat model, encryption, access control, CSI, Service Mesh, dynamic scaling.

OPEN ACCESS

SUBMITTED 29 July 2025
ACCEPTED 07 August 2025
PUBLISHED 21 August 2025
VOLUME Vol.07 Issue 08 2025

CITATION
Megha Aggarwal. (2025). Data Security in Multi-Tenant Clusters. The American Journal of Engineering and Technology, 7(8), 268–274. https://doi.org/10.37547/tajet/Volume07Issue08-22

Introduction

In recent years, the widespread adoption of cloud solutions and the transition to a microservices architecture have led to containerization becoming the foundational standard for packaging, deploying, and managing software components [1]. One of the most significant economic factors driving the demand for Kubernetes is the ability to organize multi-tenant clusters, where the resources of a single physical or virtual cluster can be reliably partitioned among multiple teams, projects, or external clients. This approach not only increases the utilization rate of compute and network resources but also substantially

reduces operational and infrastructure costs. At the same time, consolidating heterogeneous workloads within a unified cluster architecture creates new challenges in protecting data confidentiality and integrity. In a traditional virtualized environment, security boundaries are enforced at the hypervisor level. In contrast, in Kubernetes, isolation is implemented via Linux kernel mechanisms — namespaces and cgroups — which open additional vectors for potential attacks. The relevance of the topic is heightened by the growing number of incidents related to misconfigurations and vulnerabilities in container environments [2].

Despite extensive research on container security and network policies, a gap remains in the scientific analysis, particularly regarding threats targeting data in multi-tenant clusters, especially considering dynamic processes such as automatic scaling (autoscaling) that impact the data lifecycle and residual availability.

The **objective** of the study is to conduct a systematic review and analysis of data security threats in multi-tenant Kubernetes clusters.

The **novelty** of the study lies in the description of a multi-layered security model that ties potential threats at the container, pod, network communication, and persistent storage levels to the operational practices characteristic of dynamically scalable environments.

The **study hypothesiz**es that achieving reliable data protection in multi-tenant Kubernetes environments is only possible through a comprehensive approach combining:

1. Preventive measures — granular access control (RBAC), network segmentation (Network Policies), encryption of data at rest and in transit.

2. Detective mechanisms — continuous cluster state monitoring, activity auditing, and log analysis.

3. Automated response — orchestration of incident response actions through integration of SIEM/EDR systems and Kubernetes operators.

Materials and methods

In modern multi-tenant cloud clusters, the issue of security is increasingly considered through the lens of comprehensive reviews and comparative studies. Thus, the CNCF survey [1] records a rise in regulatory requirements and the broad adoption of cloud-native practices, noting increased interest in workload isolation and platform-level data encryption. The Palo Alto Networks report [2] focuses on emerging threats — from vulnerabilities in container runtimes to supply chain attacks — and proposes priorities for allocating security resources. A comprehensive systematization of attack and defense mechanisms for Docker containers is presented by Haq M. S. et al. [3], which analyzes more than 50 works on the subject and classifies security approaches at the runtime, kernel, and orchestrator levels.

Architectural deployment patterns form the foundation for multitenancy, as they define the boundaries of isolation and performance impact. Berenberg, A., & Calder, B. [4] in classification identify four main archetypes — from full virtual machines to serverless containers — and assess their trade-offs between elasticity and security. Superbo G. [5], examining a local 5G deployment, tests strict Kubernetes multitenancy mechanisms (NetworkPolicy, cgroup, SELinux), demonstrating that only a combined approach provides an acceptable level of isolation with minimal overhead. Shethiya et al. [9] explain how increasing isolation through namespace and resource constraints affects throughput and latency, key metrics for latency-sensitive applications.

Turning to practical methods for enhancing security, Morić Z., Dakić V., Čavala T. [6] propose a framework for hardening the Kubernetes control plane and workloads, incorporating checks for compliance with CIS Benchmarks and the implementation of OPA/Gatekeeper policy. Dos Santos R. F. [7] implements Zero Trust principles in the cluster by configuring mTLS for all services, implementing strict RBAC, and performing dynamic context validation of API server requests.

For continuous monitoring and rapid response, observability and anomaly-detection tools are essential. Nutalapati P. [8] describes the use of the Istio service mesh for centralized metric collection, call tracing, and enforcement of network policies via Envoy filters, simplifying audit and forensics in multi-cluster scenarios. Kosińska J., & Tobiasz M. [10] demonstrate how machine learning methods (clustering, autoencoders, decision trees) detect atypical behavior patterns in a Kubernetes cluster, issuing alerts before a full-scale attack unfolds [10].

Thus, two main contradictions emerge in the literature: first, the reports [1, 2] and SoK [3] emphasize the risks of supply chain and container runtime, whereas

architecture researchers [4, 5, 9] are compelled to sacrifice security for performance. Second, there is a stark contrast in approaches to security enhancement: some authors rely on static compliance [6], others on adaptive Zero Trust [7]. At the same time, there is a lack of research on the dynamic scaling of secure multitenancy under peak loads and on inter-cluster coordination of security policies. Issues of protection against data leakage channels (side channels), automated remediation responses, and integration of ML-detection tools into the CI/CD pipeline are also underdeveloped.

Results and Discussion

Following a comprehensive analysis of contemporary scientific and industry literature, an integrated conceptual framework for data protection in multi-tenant Kubernetes clusters has been developed. The concept of a secure multi-tenant platform for Kubernetes encompasses a formalized threat model, multi-layer isolation and access-control mechanisms, and a regulatory–procedural operational framework that guarantees the required level of protection throughout the entire application life-cycle.

The development of protective measures begins with the construction of a reliable threat model. For a multi-tenant Kubernetes cluster, a stratified approach comprising the node, container, and cluster layers is appropriate. At the node layer, compromise of the operating system of a compute node is critical: an adversary who gains root access effectively controls all containers, their data, and the network traffic hosted on the given physical or virtual host.

The container (Pod) layer exposes attack scenarios such as container escape through vulnerabilities in the OS kernel or runtime, exploitation of side channels through shared hardware resources (CPU cache, DRAM controller) to extract confidential data of other tenants, as well as abuse of privileged mode and mounting of critical host directories, which creates conditions for privilege escalation.

The cluster layer is vulnerable to network-traffic interception — both passive (eavesdropping) and active (tampering) — in East-West flows between Pods and North-South flows when accessing external services or storage subsystems. Unauthorized access to kube-api-server allows an attacker to create malicious workloads, extract the contents of Secret objects, and modify the configurations of other tenants' resources. Finally, misconfiguration of CSI drivers and the dynamic volume-provisioning mechanism can result in one tenant mounting another tenant's Persistent Volume, thereby gaining unauthorized access to data [3, 4].

Within the outlined threat model, the security system must satisfy four fundamental properties, traditionally condensed into the acronym CIA-A. Confidentiality requires that the information flows of one tenant remain inaccessible to others, both at rest and in transit; isolation must be end-to-end, encompassing storage, network, applications, and logs. Integrity implies that any modification of artefacts (data, manifests, configurations) is permissible only after subject authentication and is recorded in immutable logs, forming a verifiable chain of trust. Availability presupposes platform resilience to peak loads and deliberate sabotage (for example, DoS) while ensuring linear horizontal scalability of compute and storage subsystems. Manageability and auditability are implemented through centralized event-correlation services, long-term log retention, and automated response, which together create an evidential basis for subsequent incident investigation and regulatory compliance.

The security of a multi-tenant platform is achieved by composing complementary mechanisms whose combined effectiveness is compared in Table 1. The foundation of logical segmentation is formed by namespaces, which group all resources (Pod, Service, Secret) by tenant and block cross-tenant interaction at the object level; the role-based model (RBAC), analysed in detail by Singh and Kumar [5], makes it possible to define atomic privileges of the form ⟨action, resource, namespace⟩ and thereby strictly enforce the principle of least privilege.

For the declarative description of high-level policies, the Open Policy Agent, together with Gatekeeper, is employed, as the legacy Pod Security Policies have been withdrawn from operation, and their successor, Pod Security Admission, offers only a fixed set of rigidities. Experimental studies by Williams [6] demonstrate how Rego rules, for example, prohibit running images from untrusted registries or executing containers as the root user.

Kubernetes network policies implement a deny-by-default model: Pod-to-Pod East-West traffic is permitted only when an explicit rule is present. However, as Chen

points out [7], control at L3/L4 provides neither encryption nor mutual authentication. These tasks are solved by a Service Mesh (Istio, Linkerd, etc.), where a proxy sidecar is automatically injected into each Pod, activating mTLS, verifying certificates, and making authorization decisions at Layer 7; a topological and functional description of this architecture is provided by Zhang [8].

Data isolation at rest is achieved through CSI drivers. Best practice recommends allocating a dedicated StorageClass for each tenant and binding a PVC to the corresponding namespace, which prevents cross-tenant access [9]. Modern drivers support encryption with individual keys managed by an external KMS, thereby minimizing the risk of both inadvertent leaks and insider attacks

Table 1. Comparison of isolation mechanisms in Kubernetes [5, 6, 7, 9]

Mechanism	Isolation level	Granularity	Primary purpose	Limitations
Namespaces	Logical (API)	At the object level	Grouping of tenant resources	Does not provide network isolation or host-level isolation
RBAC	API access control	User/Group → Action → Resource	Restriction of rights to manage Kubernetes objects	Does not control processes inside the container or network traffic
Network Policies	Network (L3/L4)	Pod → Port/IP	Isolation of East–West traffic	Does not encrypt traffic, does not operate at L7, depends on CNI plugin
OPA/Gatekeeper	Configuration control	Arbitrary rules (Rego)	Enforcement of security best practices	Requires expertise in the Rego language, may introduce delays in the API
Service Mesh	Network (L4/L7)	Service → HTTP method/path	End-to-end encryption (mTLS), L7 authorization	High complexity of deployment and operation, overhead

eliable isolation must be supplemented by cryptographic data protection and mature operational processes. The integrated architecture encompassing these components is shown in Figure 1.

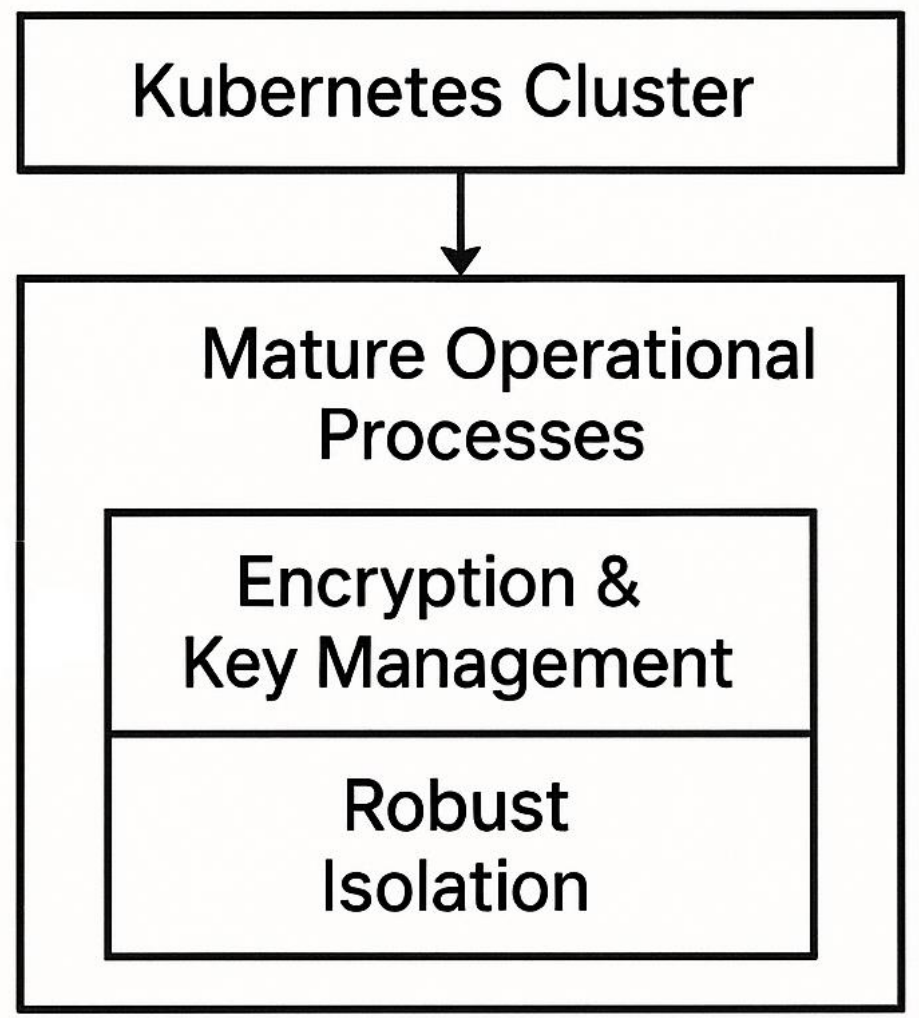

Figure 1. Integrated data security architecture in Kubernetes [8, 10].

Cryptographic data protection in modern distributed systems is divided into two mutually complementary directions: ensuring confidentiality during transmission and storage. Intra-cluster network traffic is protected by end-to-end encryption through mutual authentication of TLS channels (mTLS), implemented at the Service Mesh layer, which prevents unauthorized subjects from accessing inter-service communications. When interacting with external resources—such as databases, caching systems, or object stores—a traditional unidirectional TLS channel is employed, which is sufficient to maintain the required level of trust between the parties.

Data safety «at rest» is achieved by block-level encryption of the contents of persistent volumes using dm-crypt or equivalent mechanisms, thereby minimizing the risk of compromise in the event of physical access to storage media. Secret configuration information in etcd is additionally protected at the API server level using EncryptionConfiguration, in which an external cryptographic provider integrated with an external key management system (KMS) is specified.

Cryptographic keys are allocated in isolation for each tenant, stored in hardware- or software-protected containers, and rotated according to the established life-cycle policy. Thus, even a successful attack on etcd, without the simultaneous compromise of the KMS, does not allow an adversary to decrypt the concealed secrets.

It should be emphasized that the built-in Kubernetes Secrets mechanism is regarded only as a primary layer of protection and must not be treated as the sole means of ensuring the confidentiality of critical data.

The changes compared with the original bullet-point presentation are dictated by the requirement of a scientific-didactic style: the key provisions are integrated into a coherent text, which enhances the logical continuity of the argumentation and underscores the interdependence of the protection mechanisms both «in transit» and «at rest». Additionally, the abandonment of markers eliminates visual fragmentation, placing emphasis on the semantic links

between the components described. For production environments, a centralized store (for example, HashiCorp Vault) is preferable, which:

1. dynamically generates credentials on request,
2. records each access in a detailed audit,
3. supports a variety of authentication mechanisms,
4. enforces a consistent least privilege policy.

Table 2 describes the characteristics of observability, audit, and response in data protection.

Table 2. Observability, audit, and data protection response [9, 10].

Component	Objective	Tools
Metrics	Monitoring of the cluster core and applications status	Prometheus (+ Alertmanager)
Logs	Centralization of node, pod, and Kubernetes API events	Unified ELK/EFK stack
Behavioral analysis	Response to deviations (spawn shell, writing to /etc, etc.)	Falco
Correlation and SOAR	Aggregation of telemetry streams and automation of playbooks	SIEM platform with reactions (for example, pod isolation via NetworkPolicy)

Automatic creation and deletion of pods (HPA, VPA, Cluster Autoscaler) may lead to data leakage if a PVC is deleted faster than the underlying volume is physically sanitized. To prevent another tenant from recovering residual fragments:

1. Crypto-shredding. Use CSI drivers/storage arrays that support destruction of the encryption key; data becomes cryptographically inaccessible immediately after PVC deletion.
2. Forced zeroing. Configure automatic overwriting of the volume with zeros or random blocks before returning it to the pool.
3. StorageClass → reclaimPolicy: Delete. Destroy the physical volume, including the PVC, to ensure the sanitization process has been completed correctly [10].

A complex combining multilayer encryption, mature secret management, full-format observability with automated response, and secure volume lifecycle handling forms a resilient data protection architecture for highly dynamic multitenant Kubernetes environments.

Conclusion

The analysis performed enabled a comprehensive examination of the data protection challenges associated with the shared operation of Kubernetes clusters and the development of an integrated concept for their assurance. It was found that the application of individual security tools—whether RBAC or network policy configuration—by itself does not provide adequate protection against modern, often multifaceted attacks. The initial hypothesis has been confirmed: reliable data protection in multi-tenant environments requires a holistic approach that combines preventive, detective, and corrective mechanisms.

Prospects for further research are associated with the implementation of confidential computing technologies for hardware isolation of workloads and the development of formal methods for verification of security policies in rapidly evolving Kubernetes clusters.

References

1. Cloud Native Computing Foundation. (2024). *CNCF annual survey 2023: The state of cloud native development*. Retrieved from https://www.cncf.io/reports/cncf-annual-survey-2023/ (date accessed: 17.05.2025).
2. Palo Alto Networks. (2024). *2024 state of cloud native security report* [Report]. Retrieved from https://www.paloaltonetworks.com/resources/res

earch/state-of-cloud-native-security-2024 (date accessed: 20.05.2025).

3. Haq, M. S., et al. (2024). *SoK: A comprehensive analysis and evaluation of Docker container attack and defense mechanisms*. In *2024 IEEE Symposium on Security and Privacy (SP)* (pp. 4573–4590). IEEE. https://doi.org/10.1109/SP54263.2024.00268

4. Berenberg, A., & Calder, B. (2022). *Deployment archetypes for cloud applications*. *ACM Computing Surveys*, *55*(3), 1–48. https://doi.org/10.1145/3498336

5. Superbo, G. (2022). *Hard multi-tenancy Kubernetes approaches in a local 5G deployment: Testing and evaluation of the available solutions,* 43 – 60.

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

7. Dos Santos, R. F. (2025). *Applying zero trust to Kubernetes clusters*. *ARIS2 – Advanced Research on Information Systems Security*, *5*(1), 57–71. https://doi.org/10.56394/aris2.v5i1.58

8. Nutalapati, P. (2021). *Service mesh in Kubernetes: Implementing Istio for enhanced observability and security*. *Journal of Scientific and Engineering Research*, *8*(11), 200–206.

9. Shethiya, A. S. (2024). *Ensuring optimal performance in secure multi-tenant cloud deployments*. *Spectrum of Research*, *4*(2), 1–7.

10. Kosińska, J., & Tobiasz, M. (2022). *Detection of cluster anomalies with ML techniques*. *IEEE Access*, *10*, 110742–110753. https://doi.org/10.1109/ACCESS.2022.3216080.