International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Zero Trust Segmentation for Cloud-Native and AI Service Architectures: An Intelligent Policy Enforcement Framework to Minimize Lateral Movement

**Navaneeth Komirisetty**

**Abstract:** Cloud-native architectures break the concept of a perimeter, making lateral movement a focus of concern for distributed enterprise systems. AI services add to the attack surface via east-west traffic. As every workload, every pipeline, and every model-serving endpoint is a potential attack pivot point, layering zero-trust segmentation controls across identity, network, workload, and data planes provides a complementary strategy that restricts lateral movement in modern cloud-native and AI environments. The paper proposes a micro-segmentation model, disassociating policy decision and policy enforcement components in the context of securing data flow networks. The proposed model leverages workload identity, explicit allow-listing of communication patterns, and a service mesh to achieve micro-segmentation. Another AI-specific segmentation model addresses the introduction of the LLM tool chain, vector databases, and agentic services into a system's trust boundaries. This model adopts operational governance, evidence generation, and alignment with the NIST SP 800-207 and AI Risk Management Framework as early design requirements for the implementation and operation of zero trust segmentation in regulated and critical services contexts. It allows security architects and platform leaders to implement solutions through structured evidence generation.

***Keywords:*** *Micro-Segmentation, Zero Trust Architecture, Cloud-Native Security, Kubernetes Workload Identity, Service Segmentation With AI, Blocking Lateral Movement, Policy Enforcement, Service Mesh*

## 1. Introduction

### 1.1 The Lateral Movement Threat Surface in Distributed and AI-Enabled Systems

Most enterprise platforms are no longer restricted by a single network boundary, and containers need to communicate constantly across boundaries. The volume of east-west service-to-service traffic within a cluster or platform dwarfs the amount of north-south traffic that perimeter security was designed to monitor, and an opponent with a beachhead in this type of environment can move laterally between systems, using existing connectivity, from the initial compromise to high-value systems, without encountering internal trust boundaries.

In particular, AI applications commonly have a model-serving endpoint, a retrieval mechanism, one or more vector databases, a feature store, integrations with external tools, and sensitive datasets used to train or further fine-tune models. Such a large dependency graph introduces numerous new lateral movement pathways that would not have existed in more customary application architectures. For example, compromising a prompt handler, a poorly restricted tool integration, or an insecure vector store can lead to a chain of sensitive systems. Conversely, AI service components may not be familiar to, or closely monitored by, existing security operations in an organization.

*Sr.Cybersecurity Architect at American Express Travel Related Services inc., USA*

### 1.2 Why Perimeter-based Approaches Fall Short in Kubernetes Environments

Kubernetes was designed for agility, not security. Pods can reach each other by default, service accounts essentially have root access, and the pod boundaries are drawn for developer convenience, not with security in mind. Kubernetes security

concepts are often misunderstood. Organizations that implement perimeter controls on the edges of the cluster without controlling traffic between the containers effectively create a large flat network inside a hardened shell, where an attacker who is able to breach the perimeter using a vulnerable container image, ingress configuration, or stolen credential is able to access all internal services.

Kubernetes-native controls like network policies exist but are difficult to operate. Creating and maintaining a fine-grained allow-list of network rules requires active discipline. While service meshes provide richer identity-aware network controls, this architectural complexity has resulted in a common implementation gap in production environments between the theoretically available segmentation controls and the segmentation controls configured into the system.

### 1.3 Research Motivation and Problem Statement

While the industry has increased investments into ZTA and cloud-native security tooling, no existing work proposes an integrated architecture that incorporates Kubernetes workload segmentation and AI services' trust boundaries. NIST SP 800-207, which outlines concepts of a zero trust framework, lacks specific and detailed guidance for enforcing a ZTA in a container platform tailored for a specific domain. AI governance frameworks and standards like the NIST AI RMF acknowledge model risk but do not define model controls to limit exposure at the infrastructure layer, creating challenges for security architects building resilient AI infrastructure through effective segmentation.

The primary use case this article addresses is how to deconstruct and implement zero trust segmentation concepts to reduce risk from lateral movement in cloud-native architectures with AI services , and how to scale it to a governable and ready-for-evidence state.

### 1.4 Contributions of This Article

This article presents a layered zero trust segmentation model (identity plane, network plane, and workload & data plane) for Kubernetes and container platform environments. This article also presents micro-segmentation patterns in Kubernetes and cloud-native environments to separate policy decision from enforcement, supporting change-controlled and auditable management of trust boundaries inside workloads. Third, it provides the AI-specific segmentation model: model endpoints, vector stores, embedding pipelines, and agentic tool integrations. Fourth, it maps segmentation controls to OWASP LLM Top 10 threat categories, and it maps governance controls to NIST Cybersecurity Framework 2.0 and AI RMF control requirements. Finally, it also details operational lifecycle processes and evidence collection for maintaining segmentation in practice.

### 1.5 Article Organization

Section 2 describes background and related work. Section 3 presents the layered segmentation framework. Section 4 addresses micro-segmentation of access for service-to-service authorization, while Section 5 explores segmentation considerations for AI. Section 6 outlines operational governance and evidence collection, while Section 7 includes evaluation and application guidance. Section 8 discusses potential future work.

## 2. Background and Related Work

### 2.1 Zero Trust Architecture: NIST SP 800-207 and SP 800-207A Foundations

The National Institute of Standards and Technology (NIST) defines zero trust as a collection of principles that eliminate the concept of trust that is based on network location, derived from NIST Special Publication (SP) 800-207. Per SP 800-207, access to resources is secured regardless of where the resource is located; the access is granted based on least privilege and per session; and the security posture of assets is continuously assessed [1]. These principles redefine the network as a transport medium, one signal among many in a broader authorization decision. This concept was first formalized by Kindervag in industry. His seminal work identified the limitations of perimeter-centric trust models and proposed treating all network traffic as untrusted by default [9].

NIST SP 800-207 A extends the core principles of this approach to cloud-native workloads, addressing topics such as containerized workloads, microservice architectures, and API-based communication [2]. It codifies as normative the use of workload identity as the principal mechanism for authorizing workloads running in containers. It also makes service-to-service authorization a first-class security requirement. The CISA Zero Trust Maturity Model builds on these principles, adding a more operational approach that provides a path to

progressively higher levels of implementation for each of the five pillars: identity, device, network, application, and data [8]. These three publications provide the normative foundation for the framework described in this article.

### 2.2 Existing Work on Kubernetes Network Segmentation

Common areas researched in Kubernetes security include namespace isolation, network policies, and a service mesh as an additional layer of segmentation. The CNCF Cloud Native Security Whitepaper describes security properties that can apply to a cloud-native workload, such as network segmentation, workload identity, and supply chain security, that can be mapped to Kubernetes resources [10]. Many studies have shown that default Kubernetes networking configurations do not provide sufficient internal network segmentation. Given the mixed adoption of network policy, studies have shown that service meshes such as Istio and Linkerd that provide mutual TLS (mTLS) and identity-aware authorization at the sidecar proxy layer can drastically reduce lateral movement in the presence of consistent policy [7].

Pod security standards and node pool isolation inform our work on workload segmentation, which applies specifically to privileged workloads with additional capabilities. These two areas of research informed our design of the workload plane in Section 3.

### 2.3 Security Challenges in AI Service Dependency Graphs

Architectures to deploy AI services have different security considerations than customary microservices, as retrieval-augmented generation (RAG) systems create dynamic data accesses for a model endpoint to query external knowledge bases at inference time. These forms of access are generally not considered for threat modeling. In the OWASP Top 10 for Large Language Model Applications, prompt injection, loss of sensitive data, excessive agency, and supply chain attacks all have a platform infrastructure component that goes beyond application layer controls alone [5].

The CISA joint guidance on the secure deployment of AI systems lists the protection of model artifacts, data pipelines, and inference-time access patterns as a best practice [6]; however, it does not provide a model with which to map it to specific controls to implement.

### 2.4 Gap Analysis: Lack of Unified AI-Aware Zero Trust Segmentation Models

Existing frameworks provide partial coverage of the problem space, including NIST SP 800-207 and 800-207A on zero trust for cloud-native platforms; AI RMF for model governance, risk, and management; OWASP LLM Top 10 on application layer risk; NIST SP 800-190 on container security; NIST SP 800-53 Rev. 5 on security and privacy controls for platform governance; and NIST Cybersecurity Framework 2.0 on an outcome-based governance model for continuous improvement. They do not cover platform segmentation from the AI service trust boundary design perspective. Security architects deploying AI workloads to Kubernetes clusters currently have to piece together recommendations from various sources. The below model provides a cohesive framework for Kubernetes security of AI workloads.

## 3. Proposed Framework: Layered Zero Trust Segmentation Model

### 3.1 Framework Overview and Design Principles

The Layered Zero Trust Segmentation Model (LZTSM) is a taxonomy of four planes of segmentation controls: identity, network, workload, and data. Each plane addresses a different type of trust decision, and defense-in-depth means that a breach of one plane does not compromise the security of other planes. It is based on five principles of ZTA, as defined by NIST SP 800-207: verify explicitly, use least-privilege access, assume breach, make policy visible and measurable, and generate evidence continuously.

The framework is not a point product or a single technology but an architectural pattern that can be implemented using some combination of Kubernetes-native primitives, service mesh functionality, and a cloud provider's identity services. Organizations are free to adopt the layers of the framework incrementally, starting with identity and network controls, followed by workload enforcement, and finally, data plane enforcement.

### 3.2 Layer 1—Identity Plane—Workload Identity and Continuous Verification

The fundamental unit in this model is the identity plane. Each workload (pod, pipeline, batch job, or AI inference server) must have a verifiable identity that is used for making all authorization decisions.

In Kubernetes, this is implemented with service accounts bound to cryptographically verifiable tokens and automatic rotation of those tokens by the platform. For portability, the workload identity provider SPIFFE/SPIRE is used, which generates cryptographic identities not based on the network address of the workload but based on attestation from the workload platform that the workload is running on [13]. Identity models are traceable to authentication research in distributed systems, where it was established that principals must cryptographically attach to their asserted identity prior to being authorized to perform operations [12].

Continuous verification is the practice of ensuring that an identity cannot be implicitly trusted. Token expiration, certificate rotation, and session-level authorization checks can be used to ensure a compromised credential cannot be used for long-term access. The model-serving endpoints, data pipeline workers, and retrieval agents that run the AI workloads all require their own workload identity with scoped permissions to their dependencies. Systemic use of shared service accounts across the AI components breaks the segmentation model via implicit trust.

### 3.3 Layer 2: Network Plane: Namespace-Level and Pod-Level Isolation

Network plane isolation happens at the namespace and pod levels. At the namespace level, Kubernetes Network Policies can restrict pod communication between namespaces unless traffic is explicitly permitted. This creates coarse boundaries between functional namespaces, such as namespaces for AI inference, data processing, or more general-purpose application-level namespaces.

Pod-level isolation controls network traffic at the namespace level, specifying which endpoints a pod can talk to and on which port. When combined with a default-deny network policy applied at the namespace level and explicit allow rules for legitimate network actions, an allow-list model can be constructed to limit opportunistic lateral movement. This layer is independent of application-layer identity, operating at the network level and providing a baseline level of enforcement even when controls for higher-order activities fail.

### 3.4 Layer 3 - Workload Plane: Containment of controls in a privileged container and node pools

The workload plane is comprised of the containers themselves. Privileged containers, host path mounts, and containers running as root should be considered high-risk workloads and should be subject to a stricter set of controls. Pod Security Admission is a namespace-level admission controller that limits the pod specifications that can run within a given namespace. All workload specifications are validated against organizational policy before any pods are scheduled.

You have the option of running your sensitive workloads on a separate node pool. For instance, if you are running AI inference workloads on sensitive data, do not run them on the same node pool as your application workloads. This ensures that the impact of a node compromise is limited and allows you to control your network policy at the node level rather than the pod level, using Kubernetes taints and tolerations to enforce that separation.

### 3.5 Layer 4 - Data Plane: Scoped Access to Sensitive Stores and Sources of AI Knowledge

The data plane controls access to persistent data stores such as databases and secret stores and AI-specific data stores such as vector databases and feature stores. Data plane access is controlled by a combination of workload identity-based authorization, access control at the level of the database, and secrets management tooling such as Vault or cloud-native secrets managers. Workloads should be given access only to the resources they need and read or write access at the minimum level of permission necessary to function.

Other sources of knowledge that should be singled out are vector databases containing dense vector representations of confidential documents of the organization, feature stores containing enterprise business intelligence derived from customer data, and fine-tuning datasets containing sensitive information. The data sources behind the network boundary should follow a permission model that is clear and that associates the identity of the workloads with the actions that they can perform. The data layer should provide access logs that can be used to investigate incidents/information.

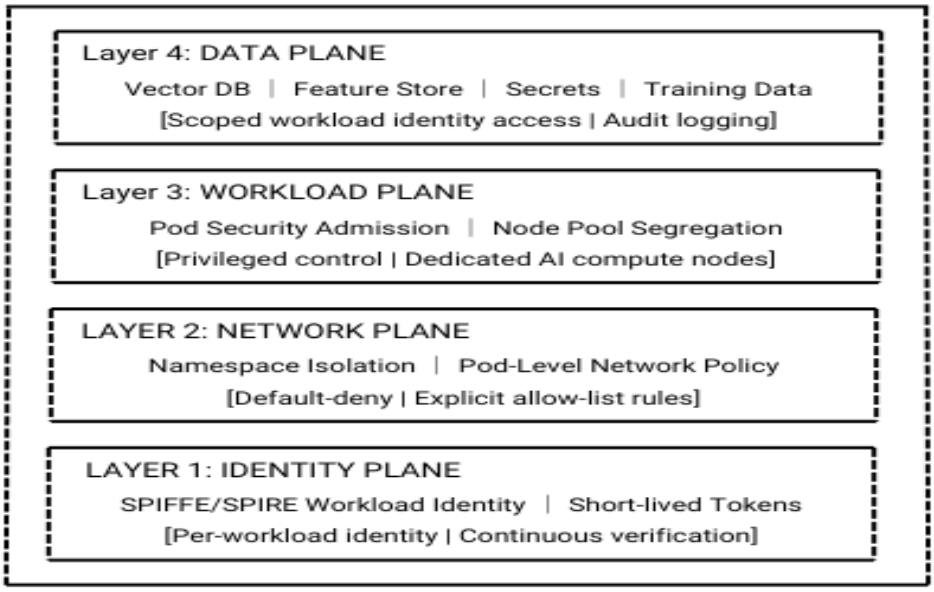


**Figure 1: Layered zero-trust segmentation architecture showing defense-in-depth across four enforcement planes.**

## 4. Micro-Segmentation Patterns for Service-to-Service Authorization

### 4.1 Separating Policy Decision from Policy Enforcement

One prescriptive design principle for scalable micro-segmentation is decoupling policy decision points (PDPs) from policy enforcement points (PEPs). PDPs compare the authorization request to the policy store and return a permit or deny decision. When a service-to-service request is made, PEPs intercept and forward requests to the PDP , and pass through traffic if authorized.

Separating policy management from enforcement enables centralized policy authoring and versioning, policy enforcement near the workloads, and separating policy testing from application logic testing. Policy behavior can be verified under simulation mode without changing how it is enforced, and enforcement can be tested without executing the application logic. In Kubernetes environments, it is often deployed with the Open Policy Agent (OPA) as the PDP while the policy is enforced inside the network policy layer, ingress controller, or sidecar proxy in a service mesh.

### 4.2 Explicit Allow-List Communication Patterns

Service-to-service communication is denied unless explicitly allowed with allow-list rules. Allow-list rules specify the identity of the source service, the destination identity, the port and protocol, and the operational context that provides the authorization. This allow-list model means that the teams will declare their service dependencies at the time of deployment, and thus the architecture is visible and documentable.

As allow-list management grows in scale to hundreds of services, group communication patterns into different categories and create templates for policy. Include frontend-to-backend requests, data reads like database queries, and event-driven messaging. Define them once and create instances per service. Non-standard patterns, such as application services directly accessing AI model endpoints, are documented exceptions and require formal approval.

### 4.3 Service mesh and identity-aware proxy integration

Service meshes provide the most complete implementation of IAM micro-segmentation by deploying sidecar proxies to each workload and enabling mTLS for all service-to-service communication. mTLS is based on TLS 1.3, which provides mutual authentication and forward secrecy. TLS 1.3 can be used to optimize performance in containerized environments where workloads are invoked frequently and the handshake time can

become a bottleneck [11]. The sidecar handles all inbound and outbound traffic, validates the calling workload's certificate, applies authorization policy, and forwards requests, thereby enforcing transport layer identity. In cases where an application does not implement proper access control, the policy provides an additional layer of defense independent of the application.

Mesh enforcement is useful for services serving AI functions, since the requests and responses for underlying functions of these services are generally opaque to customary observability tooling. The mesh can enable observability of all service interactions and flag anything abnormal, such as a model endpoint querying a service it should not , or a tool integration triggering an anomalous traffic pattern.

**4.4 Policy Versioning and Change Control Workflow**

Network and authorization policies (e.g., firewalls, IAM) should be version controlled and undergo the same change management process as application code. Policy changes that would modify trust boundaries (for example, introducing a new communication path, modifying a permission from allow to deny or vice versa, or granting an additional workload identity access to an existing sensitive store) should always go through security or platform governance.

For example, the service team may request a policy change and justification, followed by security validating that the change is consistent with the principle of least privilege, followed by approval and merging the change into the service, followed by automatic deployment of the policy to the enforcement layer, and finally, validation that the expected communication channel is working and the unexpected channel is blocked.

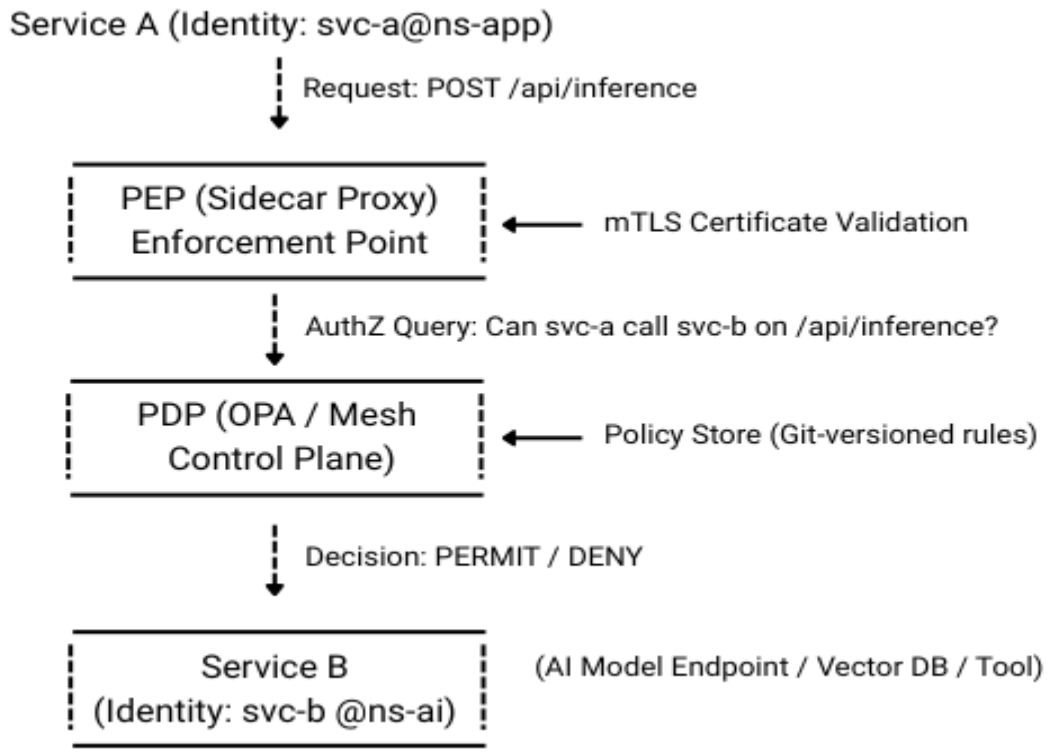


**Figure 2: Micro-segmentation pattern separating policy decision (PDP) from policy enforcement (PEP) for east-west traffic authorization.**

| Mechanism | Granularity | Identity-Aware | mTLS Support | Observability | Complexity |
|---|---|---|---|---|---|
| Kubernetes Network Policy | Pod / Namespace | No | No | Limited | Low |
| Service Mesh (Istio/Linkerd) | Service / Workload | Yes | Yes | High | Medium-High |
| Host-Based Firewall (eBPF) | Node / Pod | Partial | No | Medium | Medium |
| Identity-Aware Proxy (IAP) | Application / API | Yes | Yes | High | Medium |

**Table 1: Comparison of enforcement mechanisms across key segmentation criteria.**

## 5. AI-Specific Segmentation: Model Endpoints, Vector Stores, and Tool Integrations

### 5.1 Threat Model for AI Service Component Chains

An AI service component chain may consist of the following components: an API gateway or frontend service. A model-serving endpoint (potentially calling an external LLM API or self-hosting a model); a retrieval layer (such as an external vector database or document store); optional tool integration that allows the model to perform actions on outside tools via API calls; and logging and monitoring infrastructure. Each component of a chain may be a target.

Adversaries' attack vectors include prompt injection via retrieved documents, vector store queries for information disclosure, tool abuse, or pivoting from a compromised model endpoint to data stores the model has access to. Segregation does not eliminate these, but it raises the cost of exploitation by forcing adversaries to overcome multiple independent control mechanisms in a specific workflow that the model is always expected to follow.

### 5.2 Securing model-serving endpoints as critical infrastructure

Model-serving endpoints should be treated as any other piece of critical infrastructure in the platform security model since they may process sensitive prompts, have access to internal data sources, or make operationally important decisions. Model-serving endpoints should be accessible on the network only to a network policy allowlist. Each caller of the model-serving endpoint should authenticate to the model with a verifiable workload identity. Furthermore, model-serving endpoints should maintain access logs with the identity, timestamp, and metadata of every inference request.

External model API calls (e.g., calls to hosted LLM providers) must go through an egress proxy that enforces an allow-list of outbound destinations and logs outbound traffic to lower the risk of an attacker with access to model API credentials from setting up arbitrary external connections.

### 5.3 Access Controls for Vector Database and Embedding Store

The risk of data leakage in vector databases is not obvious. Even if it appears the vector database contains only mathematical vectors, the content of the document is encoded in the vector. An attacker can retrieve sensitive organizational information that they should not have access to by querying a vector store with a created embedding. Secret data should be likewise protected in vector stores, just like in the document stores they are based on.

Access to vector databases should be limited to workload identities that have an explicit documented need to access the store to read or update data, with read and write access being separate. In addition, workloads that are allowed to write to the store, such as data ingestion pipelines, should be separated from workloads that are allowed to read the store. This is achieved by namespace isolation between ingestion and inference components , and database access control per identity.

### 5.4 Tool Integration Authorization and Scope Restriction for Agentic Services

Agentic AI systems, or AI systems that can perform actions in other systems based on the model's output,

are considered the highest-risk type of AI service component. Tool integrations may enable an agentic service to read and write email, update records in enterprise systems, execute code, or call APIs. However, in cases where such integrations are not well scoped, a compromised prompt could have downstream effects.

Tool authorization should follow the principle of minimal agency, where each tool should be given the narrowest permission needed to do its job over the authorized resources that the agent can act on. Tool usage should be checked out-of-band, either via an authorization system that ensures that the action is legitimate or through human-in-the-loop approval for specific tools in high-stakes scenarios. For any segmentation of the network, no tool integration components should be allowed to open a connection to systems outside of what they declare.

| OWASP LLM Risk | Segmentation Mitigation |
|---|---|
| LLM01: Prompt Injection | Isolate the retrieval layer; restrict data sources accessible to the model. |
| LLM02: Sensitive Information Disclosure | Data plane access controls; vector store authorization |
| LLM06: Excessive Agency | Tool integration scope restriction: minimal-agency authorization |
| LLM08: Vector and Embedding Weaknesses | Separate ingestion and inference identities; store isolation |
| LLM09: Misinformation | Model endpoint integrity controls; artifact access restrictions |
| LLM10: Unbounded Consumption | Egress proxy rate limiting; network-level resource controls |

**Table 2: Mapping OWASP LLM Top 10 Risks to Segmentation Mitigations [5]**

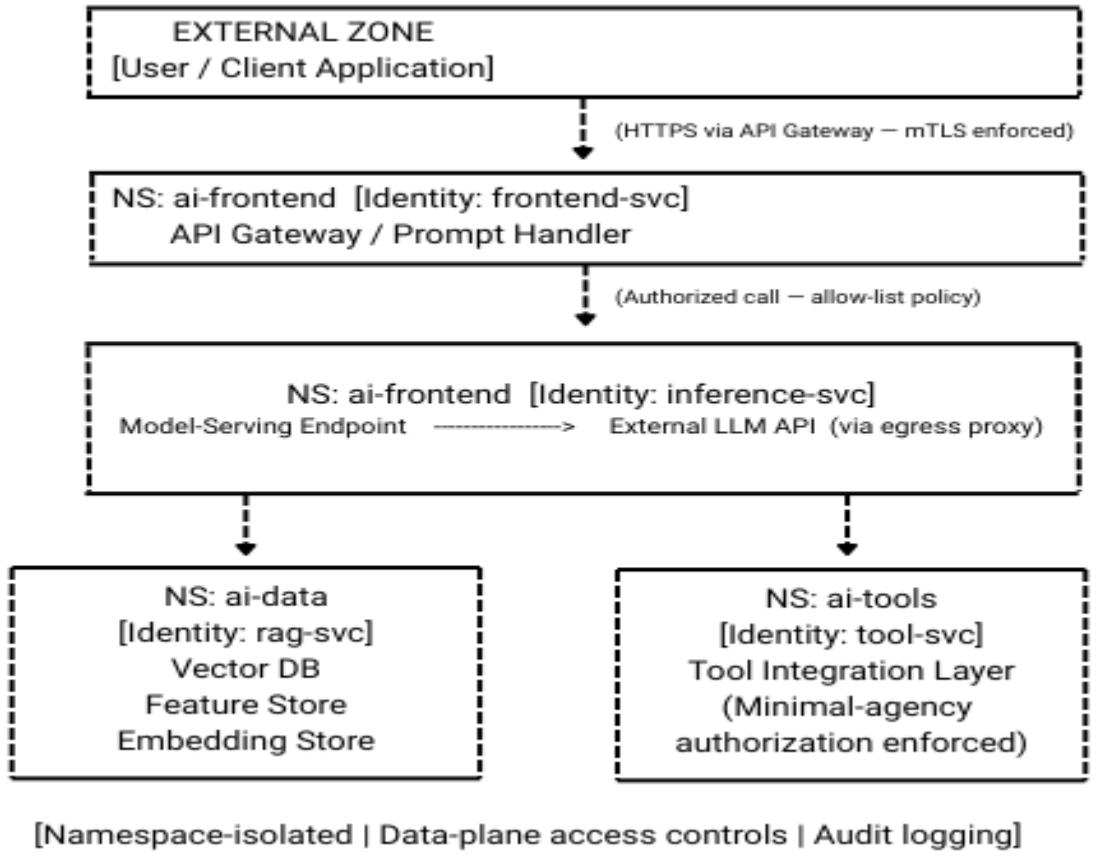


**Figure 3: AI Service Segmentation Topology with Trust Boundaries**

## 6. Operational Governance and Evidence Collection

### 6.1 Policy Lifecycle: Onboarding, Change Approval, and Exception Management

Segmentation can degrade when safeguards for architectural changes are not in place. For example, deploying new services without validating network policy, authorizing new integrations without proper evaluation, or never re-evaluating exceptions made for urgent operational tasks. A sustainable segmentation program requires a policy lifecycle in areas such as onboarding, change approval, periodic review, and exception management.

Service onboarding templates should require teams to declare inbound and outbound dependencies, the workload identity to run the service, and the data sources the service must access (e.g., hosts for HTTP connections and other hosted services). These can be used to create a baseline network policy, which is applied to block undeclared traffic. Changes to trust boundaries for trusted patterns should be lightweight. For non-trusted or high-risk patterns, changes should undergo security review, be temporary with clear reasons, be recorded in a central register, and be revisited periodically.

### 6.2 Continuous Monitoring and Audit Log Design

Segmentation without monitoring has strong structural protection , but may be difficult to monitor. Monitoring network flows, policy violations, and workload identity usage provides security operations with visibility to detect violations, to investigate incidents, and to verify controls are effective. Signals include rejected network connections (which may indicate lateral movement), unexpected policy denials at the service mesh (which may indicate misconfigurations or unauthorized use of services), anomalous data store access patterns (which may indicate potential data exfiltration), and unusual egress traffic from namespaces that host AI inference workloads.

Audit logs should contain elements that are applicable to investigation use cases, such as the workload's identity, the requested resource, the policy decision, the timestamp, and enough context to reconstruct the authorization path. Logs are forwarded to an external security information and event management (SIEM) system not controlled by the monitored workloads so a compromised workload cannot delete any evidence of its activity.

| Governance Outcome | Evidence Source | Framework Reference |
|---|---|---|
| Access is limited to authorized identities | Workload identity issuance logs; authorization policy records | NIST CSF PR.AA-05 |
| Network communications are monitored | Network flow logs; mesh telemetry | NIST CSF DE.CM-01 |
| Segmentation controls are tested | Automated policy validation results; penetration test reports | NIST CSF PR.PS-04 |
| AI data access is governed | Vector store access logs; data plane audit records | NIST AI RMF GOVERN 1.2 |
| Exceptions are tracked and reviewed | Exception register; approval records | NIST CSF GV.OC-01 |
| Policy changes are change-controlled | Version control history; change approval records | NIST CSF PR.PS-01 |

**Table 3: Evidence Mapping to NIST CSF 2.0 and AI RMF Governance Outcomes**

## 7. Evaluation and Discussion

### 7.1 Framework Validation Against NIST SP 800-207 Principles

The layered zero trust segmentation architecture can be mapped to the seven tenets of zero trust architecture defined in NIST SP 800-207 [1]. For the identity plane, the tenet that all resources must be accessed securely and that access is granted on a per-session basis with least privilege is applicable. The network plane enforces that all communication is encrypted, regardless of network location. The workload plane enforces authentication and security posture assessment against workloads before granting access to any resources. The data plane enforces access only to those data sources needed to fulfill the workload's role using resource-level access control. Dynamic security posture monitoring is also enabled at every layer. Policy versioning and change control workflow operationalizes the requirement that the enterprise security posture is collected and measured.

The AI-specific segmentation model extends this validation to AI RMF GOVERN and MANAGE outcomes, with controls and evidence to assess model infrastructure security, data access governance, and tool integration authorization.

### 7.2 Applicability in Different Deployment Scenarios

Though the architecture fundamentally focuses on Kubernetes-based production environments, you can easily apply several principles to other cloud-native environments. For example, organizations can use cloud-provider workload identity federation in managed K8s environments to implement the identity plane, cloud VPC network policy for the network plane, and cloud-native IAM and database access control for the data plane. The service mesh layer itself is also cloud-agnostic, since the major service mesh products are portable between clouds and on-premises Kubernetes clusters.

For early-stage organizations, a gradual adoption of the framework is possible. We should give priority to controls in the identity and network planes, as these provide the largest risk reduction for effort. As the platform matures, you should adopt workload and data plane controls as important components of a defense-in-depth strategy.

### 7.3 Limitations and Future Research Directions

The patterns are designed to work with a Kubernetes-based container platform, but organizations that run on virtual machines, serverless, or hybrid environments can adapt the patterns accordingly, specifically in the network and workload planes. The framework assumes the existence (or availability for implementation) of a workload identity infrastructure, which will be more challenging for organizations using legacy service-based authentication schemes.

Future work should further automate policy generation from service dependency declarations to reduce the burden of managing allow-list rules at scale. Additionally, AI observability tools and network-level segmentation monitoring could be more effectively integrated to improve detection of AI-specific attack patterns. As agentic AI systems become more advanced and take on higher-consequence actions, it may be more important to formally define minimal-agency authorization, potentially drawing from the access control literature.

## Conclusion

Zero trust segmentation is often used as a foundational architectural control in cloud-native and AI-enabled systems: organizations knock down the customary network perimeter with containerization, rendering perimeter defenses moot in protecting against the lateral movement attack surface created by exposing increasingly capable AI via service-accessible APIs. The Layered Zero Trust Segmentation Model is described, with control mappings across the identity, network, workload, and data (I, N, W, and DA) segments. Micro-segmentation was achieved by separating policy decisions from policy enforcement and applying segmentation principles to the special problem of trust in component chains that are part of an AI service.

The AI segmentation model captures the LLM application dependency graphs (model serving endpoints, retrieval layers, vector stores, and tool integrations) that map identified risks to the platform's controls and processes, treating these as design requirements rather than an operational afterthought. The model allows organizations to achieve segmentation at scale while generating the artifacts needed for cybersecurity assurance and

complying with emerging AI governance frameworks.

Layered segmentation and evidence-ready processes will remain a foundation element of trustable cloud-native AI systems as machine identities proliferate, agentic capabilities advance, and AI is brought into increasingly sensitive public service roles. This section has provided a long-term, highly durable architectural foundation with a set of trustable design principles and approaches that will constrain blast radius, improve observability, and earn the trust of the organizations and communities that depend on these systems.

## References


[1] Scott Rose et al., "Special Publication 800-207, Zero Trust Architecture," NIST, 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf

[2] Ramaswamy Chandramouli, Zack Butcher , "Special Publication 800-207A, Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments," NIST, Gaithersburg, MD, USA, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207A.pdf

[3] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST, Gaithersburg, MD, USA, 2023. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf

[4] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile (NIST AI 600-1)," NIST, Gaithersburg, MD, USA, 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[5] OWASP, "OWASP Top 10 for Large Language Model Applications 2025," OWASP Foundation, 2025. [Online]. Available: https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-v2025.pdf

[6] Cybersecurity and Infrastructure Security Agency et al., "Deploying AI Systems Securely: Best Practices for Deploying Secure and Resilient AI Systems," CISA, 2024. [Online]. Available: https://media.defense.gov/2024/Apr/15/2003439257/-1/-1/0/CSI-DEPLOYING-AI-SYSTEMS-SECURELY.PDF

[7] Murugiah Souppaya, John Morello, Karen Scarfone, "Special Publication 800-190, Application Container Security Guide," NIST, Gaithersburg, MD, USA, 2017. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf

[8] Cybersecurity and Infrastructure Security Agency, "Zero Trust Maturity Model," Version 2.0, CISA, Washington, DC, USA, Apr. 2023. [Online]. Available: https://share.google/Cpflc2KflsC7Uh3Yh

[9] John Kindervag, "No More Chewy Centers: Introducing the Zero Trust Model of Information Security," Forrester Research, Cambridge, MA, USA, Tech. Rep., 2010. [Online]. Available: https://media.paloaltonetworks.com/documents/Forrester-No-More-Chewy-Centers.pdf

[10] Brandon Kriege et al., "Cloud Native Security Whitepaper," Version 2, CNCF, San Francisco, CA, USA, 2022. [Online]. Available: https://www.cncf.io/wp-content/uploads/2022/06/CNCF_cloud-native-security-whitepaper-May2022-v2.pdf

[11] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, RFC 8446, Aug. 2018. doi: 10.17487/RFC8446. [Online]. Available: https://doi.org/10.17487/RFC8446

[12] Butler Lampson, et al., "Authentication in Distributed Systems: Theory and Practice," ACM Transactions on Computer Systems, vol. 10, no. 4, pp. 265–310, Nov. 1992. Available: https://pages.cs.wisc.edu/~remzi/Classes/739/Spring2003/Papers/theory-practice.pdf

[13] Cloud Native Computing Foundation, "SPIFFE and SPIRE: Secure Production Identity Framework for Everyone," CNCF, San Francisco, CA, USA, 2023. [Online]. Available: https://spiffe.io/docs/latest/spiffe-about/overview/

[14] JOINT TASK FORCE, "Special Publication 800-53 Rev. 5, Security and Privacy Controls for Information Systems and Organizations," NIST,

2020. doi: 10.6028/NIST.SP.800-53r5. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf

[15] National Institute of Standards and Technology, "Cybersecurity Framework 2.0," NIST, Gaithersburg, MD, USA, Feb. 2024. doi: 10.6028/NIST.CSWP.29. [Online]. Available: https://doi.org/10.6028/NIST.CSWP.29