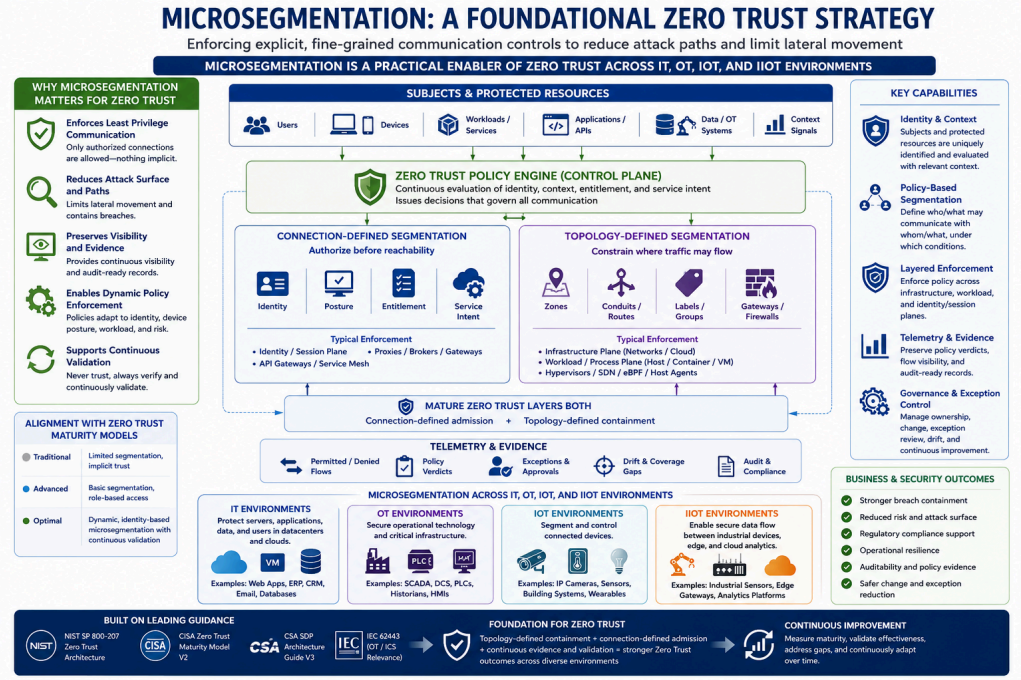
Zero Trust Microsegmentation Guidance

A Zero Trust architecture and operating model for least-privilege communication across modern IT, OT, cloud, edge, and agentic environments



Reviewers/Visitors:

- If you have a Google Account, please log in before commenting. Otherwise, please note your name and affiliation in the comment you leave.
- Use the Comments or Suggesting features on Google Docs to leave your feedback on the document. Suggestions will be written in and identified by your Google Account. To use the comments feature, highlight the phrase you would like to comment on, right-click, and select "Comment" (or Ctrl+Alt+M). Or, highlight the phrase, select "Insert" from the top menu, and select "Comment." The editing committee will review all suggestions and comments.

For more information about Google's Comments feature, please refer to http://support.google.com/docs/bin/answer.py?hl=en&answer=1216772&ctx=cb&src=cb&cbid=-rx63b0fx4x0v&cbrank=1

The permanent and official location for the Zero Trust Working Group is https://cloudsecurityalliance.org/research/working-groups/zero-trust/

Acknowledgments

The CSA Zero Trust Research Working Group

The scope of Zero Trust research and guidance necessarily includes cloud and on-premises environments, as well as mobile endpoints. It applies to the Internet of Things (IoT) and operational technology (OT).

The goals of the CSA Zero Trust (ZT) Working Group are to:

- Collaboratively develop and raise awareness of Zero Trust best practices as a modern necessity and a Hybrid or Cloud-appropriate approach to improving Information Security (InfoSec).
- Provide thought leadership and educate the industry on the strengths and weaknesses of different ZT approaches, enabling organizations to make informed decisions based on their specific business needs, risks, and priorities.
- Take a deliberate, product- and vendor-neutral approach to architecture and implementation for a mature Zero Trust implementation.
- Take a technically sound position on Zero Trust and make defensible recommendations while remaining product and vendor-neutral.
- The workstream for this document is ZT5 - Pillar: Networks/Environment, led by Philip Griffiths and Michael Roza.

Lead Authors

Philip Griffiths

Contributors

Michael Roza
Surendra Narang
Justin Bowen

CSA Global Staff

Erik Johnson

Reviewers

Mohit Bansal
Tushar Badlani
Kirtika Dommeti
Mallikarjunarao Sunke
Sree Ram R Venna
Debayan Basu
Mayur Agnihotri
Anthoni Rajanikanth Jasti
Amit Ziv
Santosh Kumar Puppala
Prashanth Reddy Pasham
Nihar Kalyanam
Meghana Parwate
Gaurav Vikram Singh
Ajay Nyayapathi
Aakashi Kapoor
Raj Annaladasu
Nishanth Sirikonda
Ting Yan
Suneet Malhotra
Nishank Soni
Priya Pandey

Table of Contents

Acknowledgments....4
- Lead Authors....4
- Contributors....4
- CSA Global Staff....4
- Reviewers....4

Table of Contents....5

Abstract....7
- Target Audience....7

Introduction....8
- Zero Trust Guiding Principles....8
- Importance of Zero Trust....9
- Microsegmentation as a Zero Trust Enabler....9
- Scope....10
  - Boundaries....10
  - Coverage....10
- How to Read This Document....11

Microsegmentation Defined....11
- Segmentation Models: Topology-Defined vs. Connection-Defined....13
- Microsegmentation and Zero Trust Architecture....14
  - Example: Containing a Common Lateral-Movement Path....15
  - Example: Limiting the Blast Radius of a Rogue Agent....16
- Core Objectives of Microsegmentation....16

Foundational Concepts....17
- Control Plane....18
  - Function and Data Flow....18
  - Core Components....19
  - Relationship to NIST Zero Trust Architecture....19
- Control Plane Resilience and Failure Behavior....20
- Enforcement Planes, Visibility, and Traffic Categorization....21
  - Visibility and Local Enforcement in Segmented Architectures....22
  - API Gateways and Application-Adjacent Controls....25
  - Operational Considerations: Underlay Dependence vs. Identity-First Policy....26
- Multi-Cloud, Hybrid, and Edge Considerations....26
- Outcome Buckets....28
- Threat and Failure Modes....30

AI-Accelerated Exploitation and Agentic Reachability 33
Implementation Prerequisites 34
Governance, Roles, and Responsibilities 35
Why Governance Matters for Microsegmentation 36
Core Roles in Microsegmentation Governance 36
Governance Best Practices 37
Segmentation Taxonomy (Macro/Micro/Nano) 38
What is Macro-Segmentation (Where)? 38
What is Micro-Segmentation (Who)? 38
What is Nano-Segmentation (What/How)? 39
How to Layer Macro, Micro, and Nano 39
The Maturity Model Lens 41
Call-Outs and Examples 41
Unified Policy Fabric for IT and OT 42
Why It's Needed 42
What It Is 42
Core Architecture Pattern 43
1. Intent Layer (Source of Truth) 43
2. Policy Compilation Layer ("Compile-to-Enforcement") 44
3. Enforcement Planes (Where Policy Actually Acts) 44
4. Telemetry and Verification Layer (Continuous Evidence) 44
How It Operates (Lifecycle Loop) 44
OT/ICS-Specific Design Requirements 45
Failure Behavior (Make It Explicit) 45
Governance and Evidence (Non-Negotiable) 46
Outcome 46
Operating Model: Outcome-Driven Microsegmentation in Iteration Loops 46
Technology and Vendor Evaluation (Checklist) 49
Conclusion and Future Outlook 52
Future Outlook 53
Useful References 56
Glossary 57
Zero Trust Glossary References 58

Abstract

Microsegmentation is a foundational Zero Trust strategy that strengthens security by enforcing explicit, fine-grained communication controls between systems, reducing attack paths, and limiting lateral movement. This paper explores the evolving role of microsegmentation as a practical enabler of Zero Trust across information technology (IT), operational technology (OT), Internet of Things (IoT), Industrial Internet of Things (IIoT), cloud-native, hybrid, and edge environments. With the growing complexity and interconnectivity of modern systems, particularly in critical infrastructure, the ability to apply fine-grained controls at the network, workload, application, process, identity, and session levels has become increasingly important.

Microsegmentation supports threat containment and breach isolation while also enhancing regulatory compliance and operational resilience. This paper provides a technical and strategic roadmap for implementing microsegmentation using multiple enforcement models, analyzes deployment trade-offs, and includes a maturity-model lens to guide adoption. The discussion is grounded in real-world applications across hybrid environments, including industrial control systems, dynamic cloud-native architectures, and distributed enterprise systems.

The paper also considers emerging agentic artificial intelligence (AI) environments. In these environments, agents, tools, application programming interfaces (APIs), models, and data services require explicit communication boundaries, governed egress, workflow-aware access constraints, and continuous validation to avoid recreating broad implicit trust paths.

Target Audience

- **Primary:** Business System and Process Owners, Chief Information Security Officers (CISOs), Security Architects, Security Engineers, Network Administrators, Network Engineers, Cloud Security Engineers, DevSecOps Engineers, OT/ICS Security Engineers, AI/ML Developers
- **Secondary:** Compliance Officers, Risk Managers, Security Analysts, IT Decision-Makers, Application Developers (with a security integration focus), IoT/IIoT Security Specialists and Developers, Security Analytics Engineers, and Observability Teams (involved in security analytics or network monitoring)

Introduction

As organizations adopt Zero Trust security strategies to adapt to modern threats and infrastructure complexity, microsegmentation has emerged as a core enforcement mechanism. Microsegmentation separates systems and workloads within environments into small, isolated segments, enabling granular control over communications between systems, workloads, and users. This segmentation limits lateral movement, prevents unauthorized access, and significantly reduces the impact of a breach. These defensive capabilities are especially vital in environments with sensitive or high-value assets, such as operational technology (OT), industrial control systems (ICS), and the rapidly expanding Internet of Things (IoT) and Industrial Internet of Things (IIoT) landscape.

These defensive capabilities are also becoming increasingly important for artificial intelligence and agentic artificial intelligence (AI) systems, where agents, models, tools, APIs, and data services may span multiple trust domains and where AI-assisted vulnerability discovery and exploitation can compress the time between exposure and impact.

Unlike traditional workloads, agentic systems may dynamically choose tools, invoke APIs, retrieve data, act on delegated user context, and create outbound egress paths at runtime; segmentation for these environments must therefore constrain permitted tool, model, API, and data paths, not only static workload-to-workload communication.

Zero Trust Guiding Principles

Together, the following Zero Trust principles shift security design away from implicit trust in network location and toward explicit validation of identity, context, policy, and permitted communication paths.

- **Never Trust, Always Verify:** Continuously authenticate and authorize users, devices, workloads, services, non-human identities, and agents, regardless of network location
- **Assume Breach:** Operate under the expectation that intrusions are inevitable and may have impacts, with segmentation and monitoring in place
- **Least Privilege:** Grant only the minimum access necessary to perform specific functions

Learn more about Zero Trust Guiding Principles.

Importance of Zero Trust

Zero Trust architectures address the evolving threat landscape, including insider threats, ransomware, and attacks on critical infrastructure. They support:

- The distributed nature of modern IT including remote work, cloud computing, and mobile-first operations
- Strong protection for sensitive data and mission-critical systems
- Enhanced threat visibility and response through fine-grained policy enforcement

Microsegmentation as a Zero Trust Enabler

Microsegmentation is essential for operationalizing Zero Trust across both IT and OT environments. As AI-assisted attack techniques reduce the time required to discover, weaponize, and exploit reachable services, microsegmentation helps shift defense from "detect and respond after exposure" toward preventing unnecessary reachability in the first place. It delivers:

- **Lateral Movement Prevention:** Critical for stopping the spread of malware or attackers across systems
- **Threat Containment:** Isolating compromised systems minimizes the impact of breaches, which is vital in critical infrastructure
- **Granular Control and Visibility-Preserving Enforcement:** Tailors policy to dynamic flows and context while preserving evidence of policy decisions, permitted flows, denied flows, and enforcement outcome
- **Operational Resilience:** Isolates disruptions and protects key assets in complex or legacy environments

This paper examines how microsegmentation facilitates Zero Trust through practical implementation strategies, technical architectures, and real-world use cases across IT, OT, IoT, IIoT, cloud, edge, and hybrid environments. It distinguishes between topology-defined and connection-defined segmentation models, explains how they can be layered, and shows how one segmentation intent can be enforced across workloads, hosts, cloud-native controls, gateways, session brokers, API boundaries, and OT/edge conduits, even in hybrid environments.

Scope

The focus of this guidance is to provide a deep dive into the specific role of microsegmentation within the broader Zero Trust framework, with a particular emphasis on its application in IT, IoT, IIoT, OT, and ICS environments—including conventional enterprise environments. It focuses on the controls that actually draw the lines between things: policies, identities, and enforcement points.

Software Defined Perimeter (SDP) with Zero Trust Network Access (ZTNA) is treated as the CSA-aligned reference pattern for connection-defined admission—authenticate and authorize the subject before permitting connectivity. Microsegmentation extends that model into a broader segmentation architecture by distinguishing connection-defined admission from topology-defined containment, and by explaining how both patterns can be governed, enforced, observed, and validated across mixed environments. In these patterns, the network underlay still transports traffic but identity, policy, and service-intent become the durable selectors for permissible communication rather than static IP topology alone.

Boundaries

Wide Area Network (WAN), Secure Access Security Edge (SASE), ZTNA, and SDP are referenced only when they help express or convey segmentation intent. Non-segmentation features (e.g., Secure Web Gateway (SWG) filtering, Cloud Access Security Broker (CASB) classifications, Software Defined Wide Area Network (SD-WAN) optimization) are explicitly out of scope here.

Application-layer access-control design is also out of scope. This paper does not attempt to standardize or prescribe authorization models within applications, such as API authorization, role-based or attribute-based access control (RBAC/ABAC), object-level permissions, web application firewall (WAF) rules, or business-logic controls. However, application-adjacent signals are in scope where they improve segmentation accuracy, validation, or safety, including service identity, request metadata, dependency mapping, software bill of materials (SBOM), provenance signals, and observability/telemetry used to map flows and verify policy outcomes. For clarity, nano-segmentation may restrict whether a process, workload, container, or agent can reach an API or tool endpoint at all. Application authorization determines whether an authenticated principal may perform a specific business action once the request reaches the application. The former is in scope; the latter is out of scope except where application-adjacent signals improve segmentation policy, validation, or safety.

Coverage

Microsegmentation approaches include network, host, hypervisor, cloud-native, application-aware, container, gateway/conduit, workload-identity, and identity/session-based controls.

OT, IoT, and IIoT considerations include legacy constraints, limited agent support, passive-first visibility, zone/cell and conduit design, gateway-based enforcement, latency and safety requirements, staged rollout, rollback planning, governed exceptions, and continuous monitoring.

Future trends in microsegmentation for IoT, IIoT, OT, and emerging AI/agentic environments include agent-to-tool, service-to-service, model, API, data-store, and orchestration flows across cloud, edge, enterprise, partner, and operational domains.

How to Read This Document

This paper moves from why microsegmentation matters, to what outcomes it should deliver, to how it can be implemented and governed. It introduces common terminology, segmentation models, enforcement planes, threat and failure modes, governance requirements, and an outcome-driven operating model for IT, OT, cloud, edge, and emerging agentic AI environments.

The paper is organized around a simple architectural sequence: identify the protect surfaces and communication paths that matter; choose the appropriate segmentation model or combination of models; decide where enforcement can safely occur; define the policy, identity, telemetry, and failure-behavior requirements; and operate the result through governance, validation, exception management, and continuous improvement. This sequence is intended to help readers connect the strategy, architecture, and operating model rather than treating each section as a standalone topic.

Readers should refer to the glossary for definitions of recurring terms such as protect surface, blast radius, connection-defined segmentation, topology-defined segmentation, non-human identity, policy drift, exception half-life, and governed egress.

Microsegmentation Defined

Traditional segmentation often leaves broad implicit trust inside zones, relies on static topology, and struggles to keep pace with cloud, OT, software-as-a-service (SaaS), partner, and agentic AI dependencies. As environments become more distributed and interconnected, attackers can exploit unnecessary reachability, stale exceptions, shared services, and undocumented dependencies to move laterally or chain attacks. Microsegmentation addresses these gaps by reducing unnecessary reachability, aligning permitted communication to identity and context, and limiting blast radius when compromise occurs.

Microsegmentation is a Zero Trust enforcement outcome in which communication between subjects and protected resources is explicitly constrained by policy at a finer granularity than traditional network

segmentation. Subjects may include users, devices, workloads, services, processes, containers, agents, or non-human identities. Protected resources may include applications, services, APIs, data stores, management planes, OT systems, cloud workloads, model endpoints, tool gateways, or other systems.

Industry definitions commonly describe microsegmentation as the creation of smaller, secure zones within data centers and cloud environments to separate and secure workloads. This guidance extends that concept across hybrid IT, cloud-native, OT, IoT, IIoT, edge, and agentic AI environments, while recognizing that different environments require different enforcement patterns, such as host or workload enforcement, cloud-native policy, gateway or conduit enforcement, identity/session brokers, service mesh authorization, or passive-first OT visibility with choke-point enforcement.

Microsegmentation may define boundaries through topology-defined controls, connection-defined controls, workload or host controls, gateway/conduit controls, identity/session controls, process/runtime controls, or layered combinations of these models. IP addresses, subnets, VLANs, and network locations can be useful topology attributes, policy selectors, or enforcement details, but they should not be treated as durable identity primitives. Where feasible, policy should bind to stronger user, device, workload, service, certificate, or non-human identity signals, with IP and location used as contextual attributes rather than as the identity itself. The defining test is not the product category or enforcement mechanism, but whether only explicitly authorized communications are permitted, enforceable, observable, and governable over time.

These patterns differ partly because the Policy Enforcement Point (PEP) may sit at different layers, including network, host, workload, gateway, proxy, identity/session broker, service mesh, or application-adjacent enforcement. Different PEP placements provide different capabilities, visibility, latency characteristics, and operational constraints.

In topology-defined implementations, segmentation boundaries are expressed through network or workload placement, labels, routes, zones, security groups, network policies, firewalls, gateways, or enforcement proximity. In connection-defined implementations, the segmentation boundary is expressed as a connection-level policy decision before reachability is granted, using identity, posture, entitlement, service-intent, and contextual policy to determine whether a session may be established.

Microsegmentation should not be treated as synonymous with asset discovery, traffic visibility, ZTNA, SDP, SASE, firewalling, mutual transport layer security (mTLS), service mesh, or application authorization. These capabilities may support segmentation outcomes, but none of them is sufficient by itself unless it results in explicit, enforceable communication constraints.

Segmentation Models: Topology-Defined vs. Connection-Defined

Microsegmentation can be implemented using various models to define and enforce segmentation boundaries. Understanding these models early prevents confusion when evaluating technologies or mapping controls to Zero Trust architectures.

Topology-defined segmentation defines policy based on where traffic may flow within an enforcement domain. Boundaries may be expressed through zones, routes, VLANs/VXLANs, security groups, labels, tags, Kubernetes network policies, host controls, hypervisor controls, firewalls, gateways, or other enforcement points. This model is often strongest for deterministic containment, local east-west control, regulatory boundaries, and OT/ICS zones and conduits. Its main operational risks are policy sprawl, brittle IP dependencies, inconsistent labels, and drift across enforcement tools.

Connection-defined segmentation defines policy based on who or what may establish and maintain a session to a specific resource, and under what identity, posture, entitlement, service-intent, and contextual conditions. The segmentation boundary is expressed through a connection-level policy evaluation before reachability is granted, reducing exposure and limiting ambient discovery. These controls should also define re-evaluation triggers, such as posture change, credential revocation, threat-intelligence updates, policy change, anomalous behavior, or session age. Without re-evaluation, connection-defined admission can degrade into a static tunnel. This model is often strongest for user-to-application access, service-to-service access across clouds or administrative domains, vendor access, remote access, and agentic tool/API access. Its main operational risks are dependency on identity sources, broker or control-plane availability, bootstrap reachability, credential lifecycle, and incomplete local containment after a session is established.

Both approaches can support least-privilege outcomes and Zero Trust principles when they enforce explicit policy, produce evidence of enforcement, and are governed over time. They differ operationally in enforcement timing, dependency on network constructs, identity requirements, failure behavior, and scalability across distributed environments.

Dimension	Topology-Defined Segmentation	Connection-Defined Segmentation
Primary Question	Where may traffic flow?	Who or what may establish a session?
Common Selectors	Zones, routes, labels, tags, subnets, security groups, network policies, gateways IP/location attributes	User, device, workload, service, certificate, non-human identity, posture,

		entitlement, service intent, context, policy
Strong Fit	Deterministic containment, OT zones, local east-west control	Pre-reachability authorization, cross-cloud/service access, remote/vendor access, agent/tool/API access
Main Risk	Policy sprawl, brittle IP dependencies, drift across enforcement points	Identity-source dependence, credential lifecycle, broker/control-plane resilience
Best Use	Contain movement inside a domain	Prevent unnecessary reachability before a session exists
Mature Pattern	Layer with connection-defined controls	Layer with topology-defined containment

Table 1: Topology-Defined vs. Connection-Defined Segmentation

In this model, IP-based controls remain useful for containment and enforcement, but stronger Zero Trust policy should distinguish between topology attributes that describe where something is and identity attributes that establish what is connecting.

Bootstrap consideration. Connection-defined segmentation still requires controlled bootstrap paths for enrollment, identity verification, certificate or credential issuance, policy retrieval, DNS or service discovery where applicable, and control-channel establishment. These bootstrap paths should be explicitly designed, minimized, monitored, and protected. They should not become broad implicit trust paths.

In mature architectures, these models are not mutually exclusive. Connection-defined controls reduce exposure before a session exists, while topology-defined controls constrain movement if a system, credential, or session is compromised. This paper evaluates microsegmentation outcomes and enforcement patterns across both models without prescribing a single technical implementation approach.

Microsegmentation and Zero Trust Architecture

Microsegmentation reinforces Zero Trust principles by enabling enforcement of explicitly authorized communications within an environment, rather than relying on implicit trust based on network location. By restricting workload-to-workload, service-to-service, and system-to-system interactions to only those explicitly permitted, microsegmentation helps ensure that communications remain continuously

constrained and verifiable. This aligns with the Zero Trust assumption that no connection—internal or external—should be trusted by default.

Within a Zero Trust architecture, segmentation boundaries may be enforced using different models:

- **In topology-defined implementations**, policies constrain permitted flows between workloads, services, zones, gateways, or enforcement domains
- **In connection-defined implementations**, policy determines whether an authenticated and authorized subject may establish a session with a protected service before reachability is granted

In both models, policy intent must be translated into enforceable controls. The architectural risk is not that translation occurs, but that teams lose sight of where translation occurred and stop validating whether the enforced state still reflects the original Zero Trust intent. Where feasible, segmentation designs should preserve higher-level policy meaning—such as identity, entitlement, posture, and service intent—as close to enforcement as possible.

Topology-defined controls are not limited to infrastructure-level segmentation. They may also be expressed around endpoint groups, workload groups, service tiers, application environments, or system categories, even when the enforcement mechanism is implemented through infrastructure, host, cloud-native, or gateway controls.

Mature Zero Trust deployments frequently employ both approaches in combination. Connection-defined controls are often well suited to cross-domain, user-to-service, service-to-service, and agent-to-tool access because they preserve identity, entitlement, and service intent close to the policy model while reducing unnecessary reachability before a session exists. Topology-defined controls remain essential for deterministic containment, local guardrails, OT zones and conduits, and blast-radius reduction following compromise.

Microsegmentation supports Zero Trust objectives by enforcing least-privilege communication at a granular level that reflects real system interactions. This reduces lateral movement and blast radius, supports defense-in-depth, and can help organizations demonstrate isolation of sensitive or regulated environments such as payment card industry (PCI), personal health information (PHI), export-controlled, or OT systems.

Example: Containing a Common Lateral-Movement Path

A common attack path begins with an initial foothold on a workstation, server, or workload. In a flat or overly permissive environment, the attacker can enumerate reachable systems, pivot toward identity infrastructure such as Active Directory, compromise privileged credentials, and move broadly across the environment.

Microsegmentation disrupts this path by removing unnecessary reachability. Topology-defined controls can restrict which systems or zones may reach domain controllers, management planes, backup systems, CI/CD platforms, cloud control planes, OT engineering workstations, historians, identity providers, authorization services, and other high-value systems. Connection-defined controls further reduce exposure by requiring the subject to satisfy identity, posture, entitlement, and service-policy conditions before a session can be established.

The same principle applies to agentic AI environments: a compromised or over-permissioned agent, exposed tool, vulnerable API, or misused credential should not imply reachability to unrelated models, tools, data sources, or orchestration services.

Example: Limiting the Blast Radius of a Rogue Agent

An autonomous agent on a user workstation may discover reachable internal portals, shared storage, APIs, or tools, and attempt to use them even when they were not intended for the task. Microsegmentation constrains the agent's possible actions by limiting the workstation or agent runtime to pre-approved destinations; out-of-scope actions have no permitted path.

Core Objectives of Microsegmentation

Building on its role within Zero Trust architectures, microsegmentation is designed to achieve a specific set of security and operational outcomes.

Modern digital estates—hybrid, cloud-native, and legacy OT—often contain many implicit trust paths created by routing, legacy dependencies, shared services, administrative tools, and operational exceptions. Microsegmentation addresses this blast-radius problem by enforcing least-privilege connectivity at a scope far tighter than traditional VLANs or perimeter firewalls.

The core objectives of microsegmentation are:

- **Contain Lateral Movement:** Halt ransomware and post-exploit reconnaissance before they spread across environments
- **Reduce AI-Accelerated Attack Paths:** Limit the ability of automated or AI-assisted attackers to discover, reach, and chain vulnerable services, tools, APIs, identity systems, or management planes
- **Enforce Zero Trust Principles:** Apply granular policies to ensure "never trust, always verify"
- **Contextual Policy Enforcement:** Apply security based on identity, workload role, or OT safety level rather than IP topology

- **Visibility-Preserving Enforcement:** Avoid treating segmentation and observability as opposing goals. Effective microsegmentation should preserve evidence of who or what communicated with which service, under which policy, from which context, and with what enforcement outcome
- **Regulatory Ring-Fencing:** Demonstrably isolate PCI, PHI, or export-controlled data zones
- **Operational Resilience:** Limit the impact of inevitable compromises, maintenance activities, or cloud outages to the smallest possible cell
- **Reduce Trust Debt:** Remove or revalidate unnecessary connectivity that persists after its original business need has changed, expired, or become unclear. In agentic environments, trust debt can accumulate faster because agents may discover and chain new tool, API, model, and data paths faster than traditional human review cycles
- **Dynamic Adaptability:** Enable teams to build and tear down isolated environments on demand without ticket-driven network changes

When properly deployed, microsegmentation acts as an architectural circuit breaker for Zero Trust by requiring explicit authorization for new connections and preserving evidence of policy decisions, permitted flows, denied flows, and enforcement outcomes where applicable.

Foundational Concepts

Before we proceed, we need a shared vocabulary. The following three concepts—**Control and Enforcement Planes and Contextual Inputs (delivery tags, traffic categorization, and policy granularity tiers)**—form the scaffolding for every decision in the rest of this paper. Mastering this language enables you to translate abstract Zero Trust ambitions into concrete rule sets that you can deploy, monitor, and audit across IT and OT estates.

The diagram shown in Figure 1 illustrates how identity, posture, and telemetry drive continuous policy decisions and real-time enforcement across infrastructure, workload, and identity/session layers. It distinguishes between the **Control Plane**, where policy decisions and orchestration occur, and the **Enforcement Plane**, where those policies are actually applied to network and application traffic. Together, these layers establish the foundation for scalable, auditable segmentation across mixed environments.

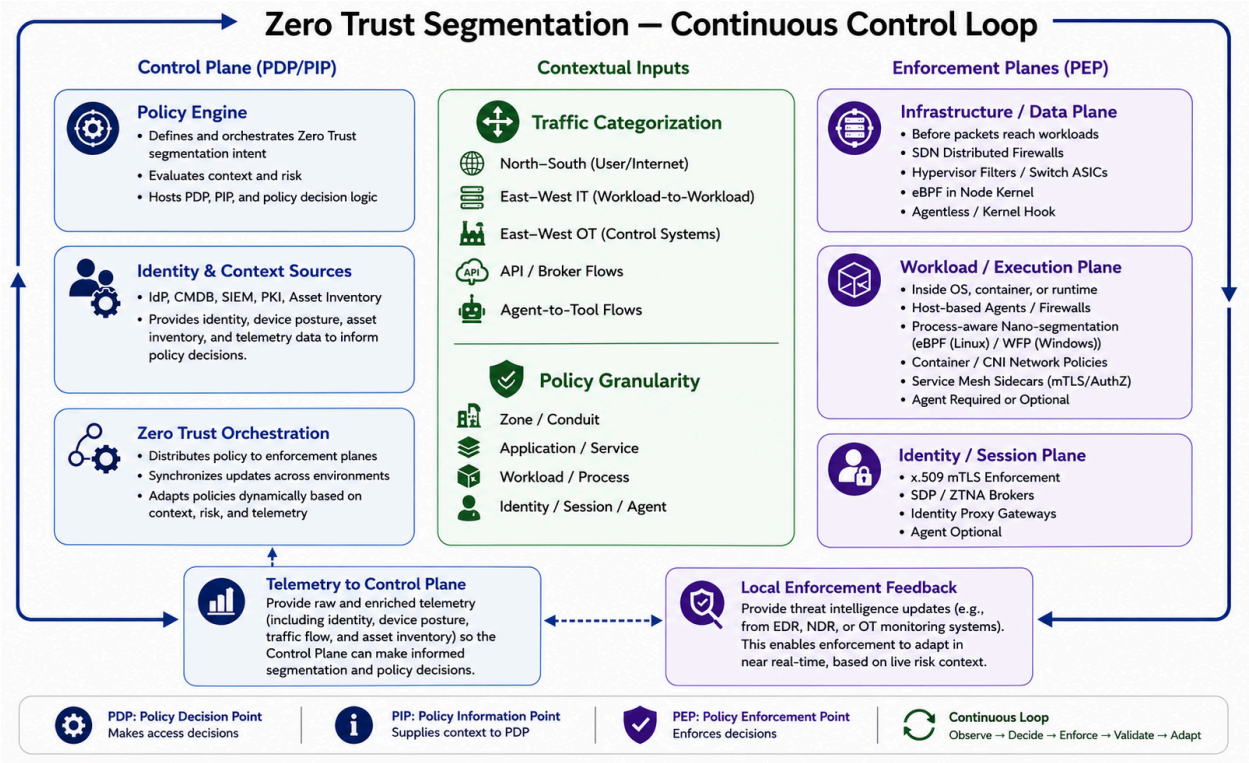


Figure 1: Zero Trust Segmentation Continuous Control Loop

Control Plane

A segmentation policy must originate somewhere; it does not live solely at enforcement points. The Control Plane is the source of truth and the coordination point for segmentation across the infrastructure, workload, and identity planes. It defines *what* to segment, *how* to segment it, and under *what conditions* segmentation boundaries can change.

It hosts the **Policy Decision Point (PDP)** and **Policy Information Point (PIP)** functions that translate Zero Trust intent into deployable segmentation rules. These functions determine which identities, devices, or processes are permitted to communicate, and when segmentation barriers should be dynamically tightened or relaxed based on the risk posture.

Function and Data Flow

Operating above the enforcement layers, the Control Plane continuously ingests identity, posture, and telemetry data from contextual sources to maintain accurate segmentation decisions. It ensures that segmentation policies evolve with context, such as new workloads appearing, device health changing, or

threat indicators emerging. Once evaluated, decisions are distributed downstream to enforcement planes for execution, synchronizing segmentation boundaries across infrastructure, workload, and session layers.

Core Components

- **Policy Engine/Administrator:** Defines Zero Trust segmentation intent and translates it into actionable, enforceable policy logic
- **Identity and Context Sources:** IdP, PKI, CMDB, SIEM, asset inventories, posture systems, and telemetry sources that provide contextual attributes of varying freshness and assurance that shape segmentation rules
- **Zero Trust Orchestration:** Synchronizes and pushes segmentation updates to enforcement planes, ensuring consistency across distributed IT and OT environments

Relationship to NIST Zero Trust Architecture

The terminology in this paper is intended to align with the logical architecture described in NIST SP 800-207 and related implementation guidance such as NIST SP 1800-35. The mapping is not one-to-one in every deployment, but the following alignment helps readers relate the paper's Control Plane and Enforcement Plane language to common Zero Trust terminology.

Paper Term	NIST SP 800-207 Alignment	Role in this Paper
Control Plane	Policy Engine, Policy Administrator, and policy information inputs	Coordinates segmentation intent, evaluates context, and distributes enforceable decisions across enforcement planes
Policy Engine/Administrator	Policy Engine and Policy Administrator	Defines and evaluates Zero Trust segmentation policy, then translates approved decisions into deployable enforcement actions
Identity and Context Sources	Enterprise data sources/policy information inputs	Provide identity, posture, asset, risk, telemetry, and environmental context used to shape policy decisions
Zero Trust Orchestration	Policy administration and policy distribution function	Synchronizes segmentation policy across infrastructure, workload, gateway, and identity/session enforcement points

Enforcement Planes	Policy Enforcement Points	Apply policy decisions to traffic, sessions, workloads, gateways, or local execution environments
Telemetry and Evidence Loop	Continuous diagnostics, activity logs, and policy feedback	Validates whether observed flows, policy decisions, exceptions, and enforcement outcomes remain aligned with intended Zero Trust policy

Table 2: Relationship to NIST Zero Trust Architecture

The Control Plane transforms Zero Trust segmentation from a static zoning exercise into a living, adaptive control system. It allows segmentation boundaries to respond automatically to changes in context, tightening isolation during anomalous behavior and relaxing constraints when trust is verified. Without this coordination layer, enforcement planes become isolated firewalls rather than an integrated, continuously adapting microsegmentation fabric.

Control Plane Resilience and Failure Behavior

Segmentation architectures must explicitly define how enforcement behaves when the control plane, policy source, identity source, telemetry source, or enforcement point is degraded or unavailable. This is especially important in OT, edge, disconnected, and high-availability environments, where availability and safety constraints may differ from those in conventional IT systems.

At a minimum, implementations should define:

- **Last-Known-Good Policy:** Enforcement points should be able to continue operating on a validated policy state when the control plane is temporarily unavailable
- **Fail-Open, Fail-Closed, and Fail-Safe Behavior:** The expected behavior should be documented by environment and flow type. IT administrative access may fail closed, while some OT safety or deterministic flows may require fail-safe continuity with compensating monitoring
- **Policy Rollback:** Corrupt or disruptive policy pushes should be reversible through a tested rollback mechanism
- **Stale Policy Indication:** Operators should be able to identify when enforcement points are operating on stale policy or stale context
- **Identity-Source Degradation:** When identity, posture, certificate, or context signals are unavailable, the system should not silently broaden access. Predefined compensating controls should apply
- **Identity-Source Compromise:** When identity signals appear valid but may be malicious because of Identity Provider (IdP), Certificate Authority (CA), token, or credential compromise,

compensating controls such as token binding, short-lived credentials, anomaly detection, privileged-flow monitoring, and topology-defined containment should be considered

- **Break-Glass Operation:** Emergency access should be time-bound, attributable, logged, and approved where practical, and reviewed after use
- **Disconnected or Edge Operation:** Environments with intermittent connectivity should define local policy cache, certificate lifetime, revocation behavior, telemetry buffering, and recovery procedures
- **Policy-Version Conflict and Split-Brain Behavior**: Distributed enforcement domains should define policy version precedence, conflict resolution, reconciliation behavior after partitions, and operator visibility into divergent policy states

These failure behaviors should be tested during implementation and reviewed periodically, especially for crown-jewel systems, regulated environments, and critical infrastructure. Rollback, break-glass, and recovery procedures should be validated as part of operational resilience and disaster-recovery planning, not inferred during an outage.

Enforcement Planes, Visibility, and Traffic Categorization

At a high level, each segmentation control resides on one of three logical "planes" that describe the proximity of the policy enforcement point to the packet and the workload. Thinking in planes prevents inappropriate comparisons between controls that operate at different layers, such as comparing a hypervisor firewall with an identity proxy. It lets architects layer complementary controls instead of pitting them against one another. It also captures operational feasibility, indicating whether a segmentation technique requires in-guest software, benefits from it, or can operate agentlessly. This helps teams identify infeasible options early, such as brownfield programmable logic controllers (PLCs) where an agent cannot be deployed.

Plane	Where Policy is Enforced	Example Controls	Deployment Model (Agent vs. Agentless?)
Infrastructure (Data Plane)	Before packets reach a workload	Software Defined Networking (SDN) Distributed Firewall (DFW), hypervisor filters, Extended Berkeley Packet Filter (eBPF) in node kernel	**Agentless** (switch Application-Specific Integrated Circuit (ASIC), hypervisor, kernel hook)

Workload (Execution Plane)	Inside the operating system (OS) or container data or management planes	Host agents, nano-seg to filter traffic by process ID (eBPF in Linux / WFP in Windows), CNI Network Policy, admission controllers	**Agent-required or optional** (whether deployed or pre-packaged)
Identity (Session Plane)	Post-authentication, post-authorization, pre-application	x. 509 mTLS, SDP/ZTNA broker, K8s Pod and Service OpenID Connect (OIDC)/roles enforcement	**Agent-optional** (depends)

Table 3: Enforcement Planes

Note: Latency, throughput, CPU, memory, policy-convergence, and failover impact vary by enforcement plane, implementation, traffic pattern, encryption/inspection requirements, and underlay conditions. Performance should be validated using representative flows and documented rollback thresholds, rather than assumed based on generic product claims.

Visibility and Local Enforcement in Segmented Architectures

Microsegmentation should be designed with visibility and enforcement together. As organizations reduce default reachability and increasingly encrypt east-west communications, they must ensure that segmentation does not unintentionally reduce security observability. Effective architectures preserve telemetry at the point of enforcement, including identity, source, destination, service, policy decision, session metadata, and, where available, workload, process, container, or socket context.

This does not mean that every flow must be decrypted or inspected centrally. Rather, visibility should be captured as close to the enforcement point as practicable, before traffic is encrypted, dropped, proxied, or forwarded. In some environments, this may come from host agents, kernel-level controls, container networking, service meshes, gateways, firewalls, or identity/session brokers. In OT and IoT environments, where endpoint agents may not be feasible, gateway-based telemetry, passive monitoring, and choke-point enforcement may provide the practical visibility layer.

Deeper packet inspection, where required, should be treated as a policy-driven inspection function rather than an assumption for all traffic. Organizations should decide which services, risks, or regulatory zones require inspection, packet capture, or external Intrusion Detection System (IDS) and Intrusion Prevention

System (IPS) integration, while preserving least-privilege segmentation and encryption for ordinary authorized flows.

The design goal is to avoid a false tradeoff between Zero Trust enforcement and operational visibility. Microsegmentation should produce evidence of who or what communicated with which service, under which policy, from which context, and with what enforcement outcome. This evidence supports monitoring, threat hunting, compliance validation, and incident response without requiring broad network reachability or weakening segmentation boundaries.

At a minimum, segmentation evidence should identify: subject identity, where available; source; destination; protected service or resource; enforcement point; policy decision; matched policy or rule; timestamp; session metadata; posture or context used in the decision; exception status; and policy version. Where feasible, evidence should also include context from workload, process, container, socket, API, tool, or model. This evidence should be exportable to operational monitoring, Security Information and Event Management (SIEM), Extended Detection and Response (XDR), Security Orchestration, Automation, and Response (SOAR), audit, and compliance workflows.

Segmentation telemetry can become operationally expensive if every event is retained at the same fidelity. Organizations should define retention tiers, sampling rules where appropriate, high-fidelity logging for crown-jewel and regulated flows, and lower-cost summaries for routine permitted flows. Dropped, denied, anomalous, exception-based, privileged, and policy-change-related events should normally receive higher retention priority.

This requirement becomes especially important for agentic AI workloads. Agents, tool runners, model gateways, and automation containers may initiate outbound connections, call APIs, retrieve data, invoke tools, and chain actions across multiple systems. Segmentation architectures should therefore capture local context where feasible, including process, container, socket, destination, service identity, and policy-verdict telemetry. The objective is not only to know that a workload has been communicated, but also to determine whether the specific agent process or container was permitted to access that specific model, tool, API, data source, or orchestration service under the governing policy.

- **Traffic Categorization:** Segmentation should focus on the flows that matter most. By first classifying traffic—north-south, east-west IT, and east-west OT—you determine where microsegmentation enforcement points should be placed and which protocol quirks, latency budgets, and safety constraints each control must adhere to. Segmenting *all* flows equally is impractical. Classify traffic first:
    - **North-South:** Ingress/egress for users, partners, and the Internet
    - **East-West (IT):** Lateral workload-to-workload, service-to-service, API-to-API, and broker-mediated communication inside data centers, cloud virtual private clouds (VPCs)/virtual networks (VNETs), Kubernetes clusters, SaaS integrations, and event-streaming platforms such as message queues, Kafka-style brokers, or cloud event hubs

    - **East-West (OT):** Deterministic or latency-sensitive control flows between PLCs, Human-Machine Interfaces (HMIs), historians, controllers, gateways, sensors, and IoT devices. Mapping these lets architects place the *right* plane at the *right* choke-point. Unlike IT east-west traffic, many OT control flows, and some IoT operational flows, may not tolerate packet drops, jitter, or added latency from inline controls. Above the cell, however, where traffic is less time-sensitive, inline enforcement, gateway mediation, or brokered access may become more feasible
- **Policy Granularity:** Granularity defines the blast radius you are willing to tolerate. Coarse zones are fast to stand up but leave room for lateral movement. Per-process or per-session rules shrink the attack surface to the minimum, but they add operational overhead. The table below illustrates this trade-off across five representative tiers, allowing teams to select the optimal balance for their risk appetite and resources

Level	Example	Typical Use Case	Level of Segmentation
Zone/VLAN	Production vs. Development	Quick initial segmentation	Network Segmentation
Application Tier	Web → App → DB	Compliance boundary	Macrosegmentation
Workload	vm-app-123	Ransomware containment	Microsegmentation
Process/Service	nginx → auth-svc	High-assurance OT or PCI CDE	Microsegmentation
User/Session	Contractor laptop	Bring your own device (BYOD) least-privilege	Microsegmentation
Process Instance/Transaction	eBPF filtering of control-loop process → historian query	Deterministic OT flow protection, per-process Zero Trust enforcement	Nano-segmentation

Table 4: Enforcement Level Use Cases

ZTNA/SDP and SASE in the Segmentation Stack

Security architectures often conflate microsegmentation with ZTNA, SDP, or SASE capabilities. While these technologies may contribute to segmentation outcomes, they operate at different layers of the control stack.

- **ZTNA and Software-Defined Perimeter (SDP)** technologies primarily implement connection-defined segmentation at the identity and session plane. As described in CSA's SDP Architecture Guide v3, SDP follows an authenticate/authorize-before-connect model in which access to protected resources is brokered only after identity, context, and policy conditions are satisfied. In this paper, that pattern is treated as connection-defined admission control: its primary function is session admission, exposure reduction, and minimizing unnecessary reachability before a permitted connection exists
- **Topology-defined microsegmentation**, in contrast, typically operates at the infrastructure or workload-enforcement plane, constraining which workloads or systems can communicate once connectivity is established
- **SASE architectures** integrate multiple edge security capabilities, such as secure web gateways, CASB, firewall-as-a-service, and SD-WAN. While SASE may transport or enforce segmentation policies at network edges, it does not itself define segmentation boundaries unless explicitly configured to do so. In this context, SASE should be understood as a transport and edge-enforcement layer rather than as the primary definition of trust boundaries

Within a layered Zero Trust architecture:

- Connection-defined controls reduce pre-authentication exposure and govern session establishment
- Topology-defined controls constrain lateral movement and enforce containment within domains
- Edge and transport layers may carry or enforce segmentation intent, but do not replace identity- or workload-based policy definition

Understanding these distinctions helps organizations avoid architectural ambiguity and select complementary controls rather than overlapping tools.

API Gateways and Application-Adjacent Controls

API gateways, service gateways, and application delivery controls may contribute to segmentation outcomes when they enforce explicit communication policy between subjects and protected APIs or services. However, API authorization, Web Application Firewall (WAF) policies, object-level permissions, and business-logic controls are not the same as microsegmentation. They should be treated as complementary controls that can strengthen segmentation when integrated with identity, service-intent, policy enforcement, and telemetry.

Operational Considerations: Underlay Dependence vs. Identity-First Policy

In many environments, implementing least-privilege connectivity using topology-defined controls requires coordinated changes to routing, firewall policies, security groups, name resolution, and sometimes private links. This coordination introduces delay and encourages compensating behaviors such as broadly permitted rules or long-lived exceptions. Connection-defined approaches (e.g., identity- and policy-gated session establishment) can reduce this friction by making identity and intent the durable selectors, decreasing reliance on static addressing and reducing the operational cost of change across clouds, sites, and administrative domains.

These operational properties do not replace the need for topology-defined containment controls. Rather, they influence where least privilege is most effectively applied (session admission vs. flow constraints) and how consistently it can be applied across distributed environments. This often favors connection-defined controls for cross-domain, user-to-service, service-to-service, and agent-to-tool access, because they preserve identity, entitlement, and service intent closer to the policy model. Topology-defined controls remain essential for deterministic containment, local guardrails, OT zones and conduits, and blast-radius reduction after compromise.

Multi-Cloud, Hybrid, and Edge Considerations

Multi-cloud, hybrid, and edge environments make segmentation harder because enforcement primitives differ across platforms. A single business intent may need to be expressed through cloud security groups, Kubernetes network policies, host firewalls, service mesh authorization, SD-WAN or SASE policy, identity/session brokers, OT gateways, and traditional firewalls. Each system may use different labels, identity models, policy syntax, logging formats, and change-control processes.

Segmentation programs should therefore normalize policy intent around durable selectors such as identity, role, label, environment, data sensitivity, service ownership, and protect surface. Where policy must be compiled into platform-specific controls, teams should validate that the enforced state still matches the intended trust boundary.

Special attention is required for cross-cloud identity and label consistency, partial enforcement, edge and disconnected operations, serverless and ephemeral workloads, and administrative domain boundaries. Missing enforcement should be tracked as a coverage gap rather than hidden by architectural diagrams.

The following quick cues summarize common use cases and indicate when topology-defined, connection-defined, or hybrid segmentation models are typically the best fit, along with key tradeoffs and common layering patterns.

Use Case	Primary Fit	Why it Fits	Tradeoffs/ Constraints	Common Pairing
Remote User → Private Application	Connection-defined	Pre-reachability authorization reduces the exposure of private apps	Depends on strong identity/posture signals; broker/service availability becomes critical	Topology-defined controls inside the app enclave to constrain east-west movement
Service-to-Service Across Clouds/Domains	Connection-defined or hybrid	Reduces reliance on IP/routing symmetry across domains; improves auditability of "who talked to what."	Requires workload identity lifecycle (PKI/certs); legacy protocols may need gateways	Local topology-defined guardrails (SG/NSG/host policies) per domain
Intra-Domain East-West (Data Center/VPC) Ransomware Containment	Topology-defined	Deterministic containment boundaries; broad coverage within an enforcement domain	Policy sprawl, if tied to static addressing, needs label hygiene for durability	Add connection-defined controls for privileged/admin access paths where appropriate
OT/ICS Zone/Cell Protection with Deterministic Flows	Topology-defined	Often agentless; works with legacy/embedded endpoints; stable choke points	Limited identity context at endpoints; risk of "keep-plant-running" exceptions	Connection-defined brokering for vendor/remote access into OT, as well as IT/OT convergence or connections across cells + strict monitoring/recording
Kubernetes/Ephemeral Microservices	Hybrid	Topology-defined (Container Network Interface(CNI)) scales for east-west; connection-defined (service identity) binds "who" to sessions	Operational complexity (identity issuance/sidecars); label hygiene required	Start with topology allow-lists; add service identity/mTLS for high-value namespaces

Regulatory Ring-Fencing (PCI/PHI/Controlled Data)	Hybrid	Hard boundaries + strong auditability; reduces scope creep	Over-reliance on one layer can leave gaps (admission without containment, or containment without identity assurance)	Topology-defined boundaries + connection-defined access governance + continuous monitoring

Table 5: Use Cases by Segmentation Model

As a design principle, avoid treating topology-defined and connection-defined segmentation as competing models. Topology-defined controls are often strongest for deterministic containment inside an enforcement domain, especially where agentless or infrastructure-level controls are required. Connection-defined controls are often strongest where reachability should not exist until identity, posture, entitlement, and service-intent have been evaluated. Mature Zero Trust architectures layer both connection-defined controls, which reduce exposure before a session exists, and topology-defined controls, which contain movement if a system, credential, or session is compromised.

Outcome Buckets

Outcome buckets translate the broad goal of "segment everything" into practical protection aims. Instead of starting with products or enforcement mechanisms, teams can begin with the risk they are trying to reduce: ransomware containment, regulatory ring-fencing, OT safety isolation, DevOps dynamism, user/device least privilege, or agentic workload containment.

Each bucket suggests likely enforcement planes and tradeoffs, but the final design should be validated against the organization's latency, safety, operational, compliance, and governance requirements. Buckets are not mutually exclusive: a regulated payment application may also be a crown-jewel workload, run in Kubernetes, expose APIs, rely on shared identity services, and include agentic automation.

#	Outcome	Typical Drivers	Primary Enforcement Plane(s)	Deployment Model	Predominant Traffic Direction
1	**Blast-Radius Containment**	Crown-jewel IT apps, protecting against ransomware or privilege escalation	Infrastructure (Hypervisor DFW) and host kernel (eBPF/XDP),	Agentless at infrastructure; agent-optional at workload	Mostly east-west

		footholds in IT and converged Demilitarized Zones (DMZs); reducing the number of systems, teams, suppliers, or business functions affected by an incident	optionally workload (host firewall)		
2	**Regulatory Ring-Fencing**	Datacenter and cloud workloads; PCI DSS scope, PHI enclave, export-controlled IP	Infrastructure (SDN fabric, kernel eBPF cloud security group (SG)/network security group (NSG); identity/session (mTLS, ZTNA proxy)	Predominantly agentless; client agent optional for posture	Mix of north-south and east-west
3	**Safety-Critical OT Isolation**	PLC/Supervisory Control and Data Acquisition (SCADA) zones, deterministic traffic in IEC 62443 levels 1–3, as well as supporting IT/OT convergence and Security Risk Assessment (SRA) into the zone/cell	Infrastructure (SDN L2/3 or eBPF kernel); machine-identity gateways or certificate-based overlays where endpoints cannot support TLS or agents directly	Agentless (agents seldom viable on PLCs)	Mix, though mostly deterministic, east-west and north-south
4	**Dynamic DevOps Isolation**	Ephemeral micro-services, Kubernetes, serverless functions	Infrastructure (container CNI eBPF); workload (service-mesh sidecar), cloud SG/NSG	Infrastructure layer agentless; sidecar or node level is an agent	Mostly east-west
5	**User/Device Least Privilege**	BYOD, contractor laptops, and Augmented Reality (AR)/Virtual Reality (VR) devices at the edge (unmanaged or high-risk endpoints)	Identity/session (SDP/ZTNA broker); infrastructure (cloud SG)	Agent-optional—enhances posture checks	Predominantly north-south

6	**Agentic Workload Containment**	AI agents, tool runners, model gateways, automation containers, Model Context Protocol (MCP)/tool servers, AI-enabled workflows	Workload/Execution Plane and Identity/Session Plane; optionally, Infrastructure Plane for local guardrails	Container/node agent, kernel enforcement, sidecar, gateway, or brokered access, depending on environment	Primarily egress and service-to-service; often cross-domain

Table 6: Outcome Buckets

Threat and Failure Modes

Threat actors routinely exploit implicit trust and undocumented connectivity to move laterally, escalate privileges, and expand blast radius. Microsegmentation is intended to constrain these paths, but the control can be weakened by design gaps, incomplete coverage, policy drift, or misalignment with operational and safety requirements. This section highlights common threats and failure modes observed in real deployments and provides mitigation strategies to maintain effective segmentation outcomes over time.

Some failure modes are more prevalent in topology-defined implementations (e.g., policy sprawl tied to network constructs), while others are more prevalent in connection-defined implementations (e.g., dependency on identity signals and session infrastructure). Hybrid architectures reduce dependence on a single control but require stronger governance to prevent cross-layer drift. For this reason, segmentation programs should produce evidence that policy intent, enforcement state, observed flows, exceptions, and ownership remain aligned across all enforcement planes.

Failure Mode	How it Manifests	Security/Operational Impact	Mitigations/Compensating Controls
Incomplete or Ineffective Enforcement	Segmentation is designed or visualized but not enforced consistently across all systems	Lateral movement remains possible despite apparent segmentation coverage	Identify enforcement gaps explicitly; apply compensating controls, architectural isolation, or alternative enforcement mechanisms

Overly Broad or Static Policy Definitions	Policies rely on coarse rules, static IPs, or permanent exceptions	Implicit trust paths persist and are exploitable by attackers	Use role-, function-, and identity-aligned policies; regularly review and refine policies as environments change
Policy Sprawl and Operational Complexity	Large numbers of overlapping or undocumented rules accumulate over time	Increased misconfiguration risk and slower containment during incidents	Standardize policy models, document intent, and enforce governance and lifecycle management
Dependency Blind Spots and Application Fragility	Legitimate but undocumented dependencies are blocked or bypassed	Application outages, OT safety impacts, or permanent policy exceptions	Perform dependency discovery before enforcement; use staged or monitor-first deployment
Identity and Context Gaps at Enforcement Points	Enforcement points lack reliable identity or contextual signals	Policies fall back on coarse indicators, reducing precision	Strengthen identity primitives where possible; apply policy-based access controls, such as RBAC/ABAC, and contextual or network-level compensating controls
Visibility Gaps After Segmentation or Encryption	Traffic is blocked, proxied, tunneled, or encrypted without sufficient local telemetry or policy-decision logging	Security teams lose evidence needed for monitoring, threat hunting, audit, or incident response	Capture telemetry at enforcement points; log identity, service, policy decision, session metadata, and local workload/process context where available; use policy-driven inspection for higher-risk flows
Misalignment with Operational or Safety Requirements	Security controls conflict with availability or safety constraints	Segmentation is weakened or disabled during incidents or maintenance	Design segmentation jointly with security, engineering, and operations teams
Assumed Trust in Upstream or Adjacent Controls	Reliance on perimeter controls (VPN, ZTNA, firewalls) reduces internal enforcement	Rapid lateral movement once an attacker gains internal access	Treat internal segmentation as mandatory; do not relax policies based solely on upstream controls

Policy-as-Code Without Policy-as-Intent	Rules are codified but the business purpose, owner, protect surface, and rationale are lost	Teams cannot determine whether later exceptions or changes violate the original segmentation intent	Store policy rationale, owner, protect surface, approval history, and expiry/review requirements with policy code
Shared Infrastructure Services as Lateral Movement Pivots	Domain Name System (DNS), Network Time Protocol (NTP), backup, monitoring, logging, patching, artifact repositories, package registries, continuous integration/continuous delivery (CI/CD) systems, signing services, software provenance systems, secrets stores, or build systems are broadly reachable across segments	Attackers compromise or impersonate shared services to cross segmentation boundaries	Treat shared services as protect surfaces; use dedicated identities, scoped allow-lists, privileged monitoring, and separate administrative paths
DNS as an Implicit Trust Path	DNS is broadly allowed across segments; recursive resolvers, split-horizon zones, DNS over HTTPS (DoH), or tunneling paths bypass intended controls	Attackers use DNS for discovery, command-and-control, exfiltration, or policy bypass despite apparent segmentation	Use dedicated DNS zones, split-horizon DNS, resolver allow-lists, query filtering, DoH controls, DNS telemetry, and policy-aligned resolver paths
Credential-Based Movement or Identity-System Compromise	Attackers use valid credentials, service accounts, tokens, certificates, or compromised identity infrastructure to access systems that appear authorized	Segmentation based only on valid identity signals may permit malicious activity if identity assurance, posture, or behavior is weak	Protect IdPs, CAs, secrets stores, and authorization services as crown jewels; use short-lived credentials, revocation, posture/context checks, anomaly detection, and topology-defined containment
Coverage Drift	New workloads, containers, cloud resources, APIs, functions, or	New implicit trust paths appear even though the original protect surface was segmented	Continuously compare inventory, deployment metadata, observed flows, and policy coverage; report

	applications are deployed without expected segmentation controls		uncovered assets as segmentation gaps

Table 7: Segmentation and Failure Modes

Many of these failure modes are preventable through staged deployment, policy-as-code practices, continuous validation, and governance that treats segmentation as a lifecycle control rather than a one-time project.

AI-Accelerated Exploitation and Agentic Reachability

AI changes the economics and tempo of exploitation. AI-assisted tools can accelerate vulnerability discovery, exploit generation, reconnaissance, exposure testing, and chaining of reachable services. In this environment, any service, API, tool, management plane, identity system, model endpoint, data store, or orchestration service that is unnecessarily reachable becomes a more urgent risk. Relevant supporting guidance includes NIST AI RMF, MITRE ATLAS, and OWASP LLM/agentic AI security guidance.

Agentic AI introduces an additional segmentation challenge because agents may operate across models, tools, APIs, data stores, SaaS platforms, cloud services, internal systems, and operational environments. These flows often cross multiple trust zones and administrative domains. If reachability is granted broadly, a compromised, manipulated, or over-permissioned agent may discover tools, invoke unauthorized services, chain actions, exfiltrate data, or pivot to systems outside its intended scope.

Microsegmentation helps contain these risks by limiting which agents, workloads, services, and tools can communicate; enforcing least-privilege connectivity; reducing ambient discovery; and producing telemetry that supports policy validation and investigation. For higher-risk AI and agentic systems, the segmentation policy should explicitly define which agents may access which tools, APIs, data sources, model endpoints, orchestration services, and supporting services, and under what conditions of identity, posture, user delegation, task, environment, workflow, monitoring, and approval.

Agentic risk tiers. Agentic workloads should be segmented by risk. Higher-risk agents include those that can act autonomously, invoke external or internal tools, access sensitive data, act on behalf of users, write to systems of record, trigger operational changes, or reach across trust domains. Lower-risk agents may be limited to read-only retrieval, constrained model access, or isolated test environments. Segmentation depth should increase with autonomy, tool power, data sensitivity, and cross-domain reachability, and action reversibility. Read-only or reversible actions may require different controls than

externally impactful or irreversible actions, which may require step-up authorization, approval, or stronger runtime and segmentation controls. Risk tiering may also consider action reversibility. Read-only or reversible actions may require different controls than externally impactful or irreversible actions, which may require step-up authorization, approval, stronger segmentation, or additional runtime evidence. For agentic workloads, valid identity and posture are not sufficient if behavior is compromised by prompt injection, poisoned memory, malicious tool output, or model/tool supply-chain compromise. Higher-risk actions may require runtime policy gates, action-class checks, step-up approval, or independent evidence that the action is consistent with the approved workflow.

Example: Constrained agent-to-tool access. An enterprise coding or support agent may require access to a source repository, ticketing system, package registry, model gateway, and approved internal documentation. It should not automatically have reachability to production databases, identity infrastructure, CI/CD signing systems, secrets stores, or unrelated internal APIs. A segmentation policy can restrict the agent runtime to approved destinations, require workload identity or short-lived credentials, route tool access through a governed gateway, log policy verdicts, and, by default, deny unapproved egress.

Protocols such as MCP can serve as examples of agent-to-tool interaction, but the segmentation problem is broader than any single protocol. The same principles apply to agent-to-agent, model-to-service, tool-to-API, automation-to-database, and workflow-to-workflow communication.

Organizations should maintain an approved inventory of tools, APIs, models, data sources, and services that agents are allowed to access. This inventory should be treated as a managed lifecycle surface, not a static list. It should be tied to owners, permitted workflows, credential scope, logging requirements, review cadence, and detection of newly observed agent activity that is unsanctioned, unapproved, or outside permitted workflows. Review triggers should include model, prompt, tool, permission, workflow, or deployment changes. Denials should distinguish likely policy violations from stale inventory or governance drift requiring owner review.

Implementation Prerequisites

Implementing microsegmentation in OT and hybrid IT environments requires groundwork beyond conventional asset discovery. Organizations must accurately classify and map digital and physical assets, including PLCs, SCADA systems, and unmanaged field devices, while evaluating communication dependencies and segmentation boundaries. Technology stack selection must account for limited upgrade paths and real-time system constraints typical in ICS. Furthermore, successful implementation depends on telemetry-driven monitoring and centralized policy management that can span across IT, OT, and cloud workloads to support resilient Zero Trust enforcement.

- **Data Discovery and Classification:** Identify critical assets and map dependencies to inform segmentation policies
- **Policy Definition and Management:** Create and enforce rules that align with security and operational requirements
- **Identity, Label, Ownership, and Criticality Hygiene:** Establish reliable identifiers, labels, asset owners, service roles, business criticality, and data sensitivity so policies can be expressed in durable terms rather than brittle IP-based rules
- **Crown-Jewel Identity and Secrets Systems:** Treat IdPs, CAs, secrets stores, signing systems, token services, privileged access systems, and certificate or key-management services as protect surfaces with heightened segmentation, monitoring, and governance. Topology-defined containment should be used as a compensating control when identity or credential systems are compromised or cannot provide sufficient assurance
- **Technology Stack Selection:** Evaluate microsegmentation solutions (e.g., network-based, host-based) based on scalability, compatibility, and existing architecture
- **Continuous Monitoring and Adjustment:** Use monitoring tools to refine segmentation policies and respond to emerging threats
- **Performance Characterization Plan:** Define latency, jitter, throughput, packet-loss, error-rate, CPU, memory, policy-convergence, and failover expectations for each protect surface and outcome bucket. Test using representative production-like flows before enforcement. Document baseline behavior, expected overhead, safety constraints, service level objectives (SLOs), rollback conditions, test ownership, and retained evidence. In OT and IoT environments, include deterministic flow requirements, protocol sensitivity, change-window restrictions, and safety validation.

Note: Performance should not be represented as a universal property of "microsegmentation." It depends on enforcement placement, traffic type, encryption, packet inspection, host load, control-plane interaction, policy complexity, and underlay conditions. Vendor or implementation benchmarks may be useful during procurement, but implementation teams should validate performance against their own flows and safety requirements.

Governance, Roles, and Responsibilities

Technical enforcement is only one part of a successful microsegmentation strategy. Equally important is governance: establishing clear ownership, accountability, and processes to define, approve, and maintain segmentation policies. Without this, microsegmentation often suffers from "no one owns east-west," where responsibility for lateral traffic is split among networking, security, and operations teams.

Why Governance Matters for Microsegmentation

- **Policy Sprawl:** Without centralized oversight, segmentation policies accumulate inconsistently across tools and teams
- **Accountability Gaps:** Security may define policy intent, but NetOps or CloudOps may control the enforcement plane, leading to conflicts or inaction
- **Compliance Risk:** Regulators expect documented ownership of controls (e.g., PCI DSS, HIPAA, GDPR Article 25)
- **Zero Trust Alignment:** NIST SP 800-207 emphasizes the need for defined roles in the Policy Engine (PE), Policy Administrator (PA), and PEP. Microsegmentation serves as PEP and must integrate with governance

Core Roles in Microsegmentation Governance

A clear Responsible, Accountable, Consulted, Informed (RACI) model ensures that microsegmentation does not drift or get abandoned:

- **Policy Author (Responsible):** Security Architects and Engineers
  - Define segmentation boundaries around protect surfaces, including data, applications, assets, and services
  - Translate Zero Trust principles into enforceable rules
- **Policy Approver (Accountable):** CISO or Delegated Governance Committee
  - Validate that proposed policies align with business risk and compliance requirements
  - Own exceptions and break-glass approvals, including time-bound approval, expiry or review dates, compensating controls, and periodic revalidation of long-lived exceptions
- **Policy Implementer (Responsible):** Network, SecOps, or CloudOps teams
  - Deploy segmentation rules on enforcement planes (infrastructure, workload, session)
  - Maintain day-to-day changes and respond to incidents
- **Policy Reviewer (Consulted/Informed):** Compliance Officers and Auditors
  - Ensure that microsegmentation controls meet regulatory and framework requirements (e.g., PCI DSS, HIPAA, GDPR, CSA CCM)
  - Validate that policies are consistently documented, logged, and monitored

Governance Best Practices

- **Protect Surface and Segmentation Ownership Register:** Maintain a register of protect surfaces, business owners, technical owners, enforcement owners, communication-path or trust-boundary owners, criticality, required SLOs, approved communication paths, exception owners, and escalation paths

- **Change Management Integration:** Treat segmentation policy as code where feasible, with version control, policy rationale, peer review, simulation or dry-run, canary deployment, staged rollout, rollback criteria, and automated drift detection. Route change requests through established DevSecOps or ITIL processes before enforcement
- **Exception Register:** Maintain an auditable exception register with owner, business justification, affected protect surface, approved scope, expiry or review date, compensating controls, and evidence of review. Temporary exceptions should expire by default. Long-lived exceptions may be necessary, especially in OT, legacy, or regulated environments, but they should be explicitly approved, periodically revalidated, and monitored to prevent them from becoming permanent, unbound trust paths
- **Metrics and Accountability:** Assign measurable KPIs to role owners, such as percentage of workloads segmented, mean time to isolate compromised assets, compliance scope reduction, exception half-life, coverage drift, and change lead time
- **Minimum Implementation Artifacts:** Maintain a protect-surface register, communication-path or trust-boundary register, RACI, exception register, policy workflow, coverage dashboard, and evidence export process. These artifacts should be sufficient for security operations, audit, compliance, and incident response without requiring each team to recreate governance from scratch
- **Coverage Dashboard:** Track segmented assets, unsegmented assets, known exceptions, policy violations, newly observed dependencies, stale policies, enforcement gaps, and segmentation coverage percentage, such as segmented assets or protect surfaces divided by discovered assets or protect surfaces
- **Cross-Team Governance Board and Decision Authority:** Establish a microsegmentation steering group with representatives from SecOps, NetOps, CloudOps, application teams, and OT teams. The governance model should also define who has the authority to resolve policy conflicts when teams disagree
- **Review and Update Cycle:** At least annually, and after significant architecture changes, major incidents, mergers and acquisitions, divestitures, cloud migrations, OT modernization, new agentic AI workflows, or material changes to identity architecture, re-evaluate governance roles, ownership assignments, policy effectiveness, and exception status

Segmentation Taxonomy (Macro/Micro/Nano)

This section establishes a common taxonomy for segmentation. It starts with historical context (traditional segmentation = macro), then expands into micro- and nano-segmentation. It closes with a simple layering guide, a quick reference, and a maturity-model lens to help readers decide what to apply, when, and where.

Macro-, micro-, and nano-segmentation describe relative granularity and enforcement proximity, not mutually exclusive product categories. A single architecture may use macro boundaries for zones, microsegmentation for workload or service communication, and nano-segmentation for process, container, API, or transaction-level constraints. The appropriate layer depends on risk, feasibility, operational maturity, and the consequences of enforcement failure.

What is Macro-Segmentation (Where)?

- **Definition:** Macro-segmentation divides an estate into broad, well-defined zones and conduits using network-centric identifiers (e.g., IP ranges, subnets/VLANs, routed tiers, perimeter firewalls). It primarily answers the question of where traffic is allowed between large areas of the environment
- **Strengths:** Simple, well-understood; compatible with legacy and OT; implemented in existing network infrastructure; creates clear security boundaries
- **Limitations:** Coarse and topology-bound; assumes zone-level trust; brittle in cloud/ephemeral contexts; cannot express "who" or runtime intent
- **Typical Controls:** Layer-3 firewalls, Access Control Lists (ACLs), Virtual Routing and Forwarding (VRFs), VLANs, traditional DMZ patterns, cloud SG/NSG at VPC/subnet boundaries
- **Outcome Fit:** Compliance demarcation, environment separation (prod/dev/test), north-south control, blast-radius reduction at coarse granularity

What is Micro-Segmentation (Who)?

- **Definition:** Micro-segmentation enforces least-privilege communication across workloads, applications, services, users, devices, and non-human identities by using durable attributes such as labels, tags, certificates, workload identity, posture, role, service-intent, or contextual policy. It primarily answers which subjects may communicate with which protected resources, under what

conditions, and through which enforcement points. It should reduce reliance on static IP addressing where feasible but may still compile policy into IP-, route-, gateway-, host-, or cloud-native controls when required by the environment.

- **Strengths:** Portable across data center, cloud, containers; aligns to business intent (labels/roles); resists IP churn; enables Zero Trust policy (default-deny + explicit allow by identity)
- **Limitations:** Requires label hygiene and/or PKI/identity lifecycle; may need host/agent or platform integration; retrofit can be hard for legacy OT endpoints
- **Typical Controls:** Identity-first overlay network, host-based firewalls/agents, hypervisor DFW, Kubernetes NetworkPolicy/CNI, service identity (SPIFFE/SPIRE), mutual TLS, policy driven by tags/attributes
- **Outcome Fit:** Lateral-movement containment, ring-fencing regulated data/workloads, multi-cloud east-west control, DevOps isolation

What is Nano-Segmentation (What/How)?

- **Definition:** Nano-segmentation constrains communication within and between processes/services at runtime (per binary, per process identifier (PID), per container, per API/verb). It answers what is being talked about and how (e.g., protocol, process, call), thereby enforcing execution-time intent
- **Strengths:** Smallest blast radius; blocks "living-off-the-land" pivots; precise OT/PCI high-assurance controls; complements micro-segmentation within-workload guardrails
- **Limitations:** Requires deep OS/runtime hooks or sidecars or equivalent runtime controls; increases maintenance and resource requirements; has higher operational sensitivity to change such as patches and redeployments; and is rarely feasible on embedded or legacy OT
- **Typical Controls:** eBPF/WFP-based process filters, application-aware policies, service-mesh AuthZ + mTLS, API-level allow-lists
- **Outcome Fit:** Crown-jewel protection, high-sensitivity data/service tiers, strong Remote Desktop Protocol (RDP)/Server Message Block (SMB)/API containment, regulated transactions

Nano-segmentation is operationally sensitive. Policy changes must be integrated with CI/CD, patching, release management, policy-as-code testing, and runtime attestation where applicable. Poorly governed nano-segmentation can block legitimate application changes, create excessive alerts, or drive teams toward broad exceptions.

How to Layer Macro, Micro, and Nano

Think defense-in-depth by plane and granularity. Use macro to draw the big map, micro to name the actors, and nano to constrain their lines.

1. **Start with Macro (Where):** Establish default-deny between zones; carve DMZ/OT cells/VPCs; keep shared services in dedicated hubs. Target: quick wins (e.g., prod vs. dev; internet-facing vs. internal; OT L2/3 cells).
2. **Add Micro (Who):** Express policy in identities/labels (e.g., app role, environment, data sensitivity). Replace IP allow-lists with role-to-role rules (e.g., `web: prod → auth-svc:443 if env=prod & posture=healthy).
3. **Apply Nano to Crown Jewels (What/How):** Constrain processes/APIs (e.g., only nginx userland can egress 443 to auth-svc; only db-migrators can reach postgres:5432). Tie to runtime signals (e.g., image hash, signing, attestation) where available.
4. **Infrastructure Plane:** Macro boundaries and coarse east-west default-deny; hypervisor, SDN, or eBPF enforcement at the node. Workload plane: micro/nano allow-lists on hosts and containers. Identity/session plane: ZTNA/SDP and mTLS bind users and services to sessions.
5. **Operate as Code:** Labels as source-of-truth; simulate before enforce; measure coverage and drift; iterate by protect surface.

Segmentation layers do not replace credential hygiene. If an attacker obtains a valid API key, OAuth token, service account key, certificate, or OIDC token, they may authenticate directly to a target service unless the identity/session plane includes credential lifecycle controls, short-lived credentials, revocation, posture/context checks, and anomaly detection.

Quick reference (cheat sheet) decision hints:

- If you can only do one thing quickly: Macro (default-deny between zones)
- To stop ransomware spread across east-west: Micro (identity-based allow-lists)
- For your most sensitive flows: Nano (process/API constraints)

Readers focused on compliance, audit, or risk management can treat macro, micro, and nano as levels of segmentation granularity. The specific technical mechanisms referenced in the examples are defined in the glossary and should be interpreted as implementation options rather than required products or architectures.

Layer	Primary Question	Examples	Best For	Caveats
Macro	Where may zones talk?	VLANs, subnets, VRFs, L3 firewalls, cloud SG/NSG	Environment separation; compliance scoping; coarse blast radius	Coarse, IP-bound; implicit trust inside zones
Micro	Who may communicate?	Labels/tags, workload identity (certs, SPIFFE), hypervisor/host policies, K8s NetworkPolicy	Lateral-movement containment; role-to-role policy; hybrid consistency	Requires label hygiene/PKI; retrofit on legacy

Nano	What/how at runtime?	eBPF/WFP process rules, service-mesh mTLS+AuthZ, API allow-lists	Crown jewels; PCI/PHI; high-assurance tiers	Operationally sensitive; agent/sidecar needs

Table 8: How to Layer Macro, Micro, and Nano (Cheat Sheet)

The Maturity Model Lens

Use the following maturity model to phase adoption:

Maturity Stage	Primary Focus	Typical Evidence
Initial → Managed	Establish macro boundaries, basic flow visibility, and one pilot protect surface	Zone map, initial flow inventory, pilot policy, named owner
Managed → Defined	Standardize labels, identities, policy-as-code, and repeatable exception governance	Protect-surface register, policy library, exception register, simulation/canary evidence
Defined → Optimized	Expand identity/context-aware segmentation, continuous verification, and selective nano-segmentation	Coverage dashboard, drift reports, runtime telemetry, exception burn-down, and audit evidence

Table 9: Maturity Model

- **Metrics that Matter:** % of workloads under identity-based policy; mean time to isolate; reduction in permitted east-west paths; % crown-jewel processes under nano; audit scope reduction for regulated zones
- **Governance Pointers:** Treat segmentation as a shared service with clear RACI; manage policy drift; review exceptions quarterly; publish a protect-surface register and coverage dashboard

Call-Outs and Examples

- **OT/ICS:** Prefer macro at cell/zone boundaries (agentless), add brokered identities/overlays for inter-zone conduits as well as SRA/IT/OT convergence; reserve nano for Windows/servers next to PLCs
- **Kubernetes:** Macro via cluster/VPC boundaries; micro via NetworkPolicy, CNI policy, label-based rules, and identity-driven overlays; nano via service-mesh or overlay authorization and

process-aware node/container controls. Kubernetes NetworkPolicy is typically L3/L4 label-based policy and should not be treated as verified workload identity by itself. Where identity assurance is required, use workload identity mechanisms such as certificates, service identities, SPIFFE/SPIRE-style patterns, service mesh authorization, or equivalent identity-aware enforcement

- **Legacy Apps:** Macro via Virtual Local Area Network (VLAN)/ACL; micro through hypervisor DFW + tags; emulate identity with proxies/overlays; apply nano where hosts support modern agents

Unified Policy Fabric for IT and OT

Why It's Needed

Most organizations do not lack segmentation mechanisms, they suffer from fragmented **segmentation intent**. The same "allowed business interaction" often has to be re-expressed in multiple places (e.g., firewalls, cloud security groups, CNIs, host controls, service mesh policies, and remote access brokers). Over time, these independent rule sets **drift**, exceptions become permanent, and operators lose confidence that the enforced reality matches the intended trust boundaries. In IT, this becomes toil and risk; in OT, it becomes a safety concern because uncertainty drives overly permissive rules and "keep the plant running" bypasses.

A unified policy fabric pattern reduces this fragmentation by establishing a single source of segmentation intent, distributed enforcement across multiple enforcement planes, and continuous verification that outcomes remain aligned with policy intent.

What It Is

A **unified policy fabric** is a *single, declarative policy model* that describes **who/what may talk to whom, under which conditions**, expressed in durable selectors (e.g., identity, labels, roles, safety level, environment), and **compiled** into the enforcement mechanisms an organization already uses across IT and OT.

It does **not** require a single monolithic product. It is an architectural pattern that treats the segmentation policy as a portable intent layer, with multiple enforcement implementations.

Multi-cloud and hybrid compilation challenge. A unified policy fabric does not remove the fact that cloud providers, Kubernetes distributions, identity systems, service meshes, firewalls, and OT gateways use different policy primitives, label models, identity formats, and logging semantics. The fabric should

either provide an abstraction layer (e.g., identity-first overlay networks) that compiles intent into local controls or explicitly document accepted per-platform divergence. In both cases, the enforced state should be reconciled against intended policy.

Translation gap. The risk is not that segmentation intent must be translated into different enforcement mechanisms; every hybrid architecture requires translation. The risk is that teams forget where translation occurred, fail to document accepted divergence, and stop reconciling the enforced state back to the original policy intent. Where two approaches can achieve the same security objective, architects should prefer the one that preserves the higher-level policy meaning with the least translation, provided it also satisfies availability, safety, performance, and governance requirements.

Core Architecture Pattern

1. Intent Layer (Source of Truth)

A canonical model of segmentation intent, authored around **protect surfaces** (Data, Applications, Assets, and Services (DAAS)) and mapped to outcome buckets. Policies are expressed in durable terms:

- **Subjects:** Workload/service identity, device identity, user role, zone/cell role
- **Objects:** Services, protect surfaces, conduits, data sensitivity classes
- **Conditions:** Posture, time, environment, safety constraints, change window, break-glass state
- **Actions:** Allow/deny, require authentication, require mTLS, require gateway mediation, log/alert, rate-limit (where feasible)

Policy inputs do not all carry the same level of freshness or assurance. Posture signals may be near real-time, while configuration management database (CMDB) ownership, application labels, or data classifications may be stale. Where ownership, identity attributes, labels, posture, and inventory systems conflict, the policy process should apply documented precedence rules, flag low-confidence inputs, and avoid silently expanding access.

SBOM and provenance signals may also inform segmentation policy when they change the risk of a workload, dependency, or software supply-chain path. For example, if an SBOM reveals a critical vulnerability in a component used by a workload, or provenance signals indicate an unsigned artifact, untrusted build pipeline, or questionable source, the affected workload may be placed into a more restrictive segmentation policy until it is patched, approved, isolated, or protected through compensating controls.

2. Policy Compilation Layer ("Compile-to-Enforcement")

A translation step that renders the same intent into the controls you already own. This is where the fabric prevents "policy sprawl" from turning into human copy/paste across tools.

3. Enforcement Planes (Where Policy Actually Acts)

- **Infrastructure Plane:** Agentless enforcement where possible (hypervisor DFW, network ACLs, cloud SG/NSG equivalents, OT choke points and conduits, gateway policy)
- **Workload Plane:** Host/container enforcement where supported (host firewall primitives, kernel-level controls, process/service-aware controls, node or sidecar patterns)
- **Session Plane:** Authorization-before-reachability for user/device/workload access to services (brokered access, mTLS, identity-aware proxies), especially across domains and for remote/vendor access into OT

4. Telemetry and Verification Layer (Continuous Evidence)

A unified fabric is incomplete without verification. Telemetry binds "what we intended" to "what is happening," enabling:

- **Flow Validation:** Observed flows vs. intended allow-list
- **Drift and Coverage Detection:** Unauthorized paths, new dependencies, missing enforcement coverage, newly deployed unsegmented assets, stale policy inputs, and resources without assigned owners
- **Outcome Measurement:** Blast-radius reduction, scope reduction, operational lead time, exception burn-down

Business justification should also be periodically revalidated. A policy may still be technically correct while the original business reason for allowing the flow has expired.

How It Operates (Lifecycle Loop)

A unified policy fabric runs as a continuous control loop rather than a one-time firewall project:

1. **Observe:** Passively capture flows; enrich with owners, tiers, labels, and risk (active discovery is constrained in OT)
2. **Model:** Propose segmentation intent per protect surface and outcome bucket
3. **Simulate:** Compute what would break; identify undocumented dependencies and missing owners
4. **Stage:** Canary enforcement in a controlled slice (time-bound and reversible)
5. **Enforce:** Promote to steady state with explicit allowlists and governed exceptions

6. **Verify:** Continuously reconcile observed vs. intended; alert on drift; retire exceptions

This loop is the mechanism that enables segmentation at scale and keeps it safe in mixed IT/OT estates.

OT/ICS-Specific Design Requirements

A unified fabric must explicitly encode OT realities instead of assuming IT defaults. Designs should align where appropriate to IEC 62443 zones and conduits, while recognizing that the Purdue Model is a useful reference architecture for industrial environments rather than a complete security or segmentation model.

- **Safety-by-Environment Modes:** IT often prefers *fail-closed*; OT may require *fail-safe* patterns for certain deterministic flows, with explicit, audited exceptions and compensating monitoring
- **Conduit-First Enforcement:** Where endpoints cannot run agents (PLCs, embedded), enforce at **zone/cell boundaries** and **gateways**, not on the endpoint
- **Protocol and Latency Budgets:** Plane selection is determined by jitter/latency tolerance and safety impact, not feature richness. Encryption, packet inspection, IDS/IPS forwarding, and full-fidelity telemetry can add overhead or operational complexity. In deterministic OT environments, these functions should be placed to avoid disrupting safety-critical control loops and validated during approved change windows
- **Governed Break-Glass:** Break-glass is out-of-band, time-bound, attributable, and triggers heightened logging/monitoring

Failure Behavior (Make It Explicit)

A practical policy fabric must state "what happens when things go wrong," because outages drive bypasses:

- **If the intent/control plane is unavailable,** enforcement points continue with last-known-good policy; changes are paused; operators see explicit "stale policy" status
- **If telemetry is incomplete,** the system does not silently widen access; it flags uncertainty and requires explicit operator action for exceptions
- **If identity signals are missing,** policies fall back to pre-defined compensating controls (e.g., topology containment + gateway mediation), not broad allow rules
- **If prerequisite checks fail or return insufficient confidence,** downstream policy decisions should not treat the chain as approved. The flow should be denied, held, escalated to a higher approval class, or routed to manual review or compensating controls until the prerequisite failure is resolved
- **If policy sources conflict,** the system should apply documented precedence rules and flag the conflict for review rather than silently choosing the broadest access

- **If labels or ownership metadata are stale,** the system should mark the affected policy as lower confidence and require review before expansion
- **If a new resource appears without segmentation coverage,** it should be reported as a coverage gap and placed into a default policy state appropriate to the environment
- **If identity signals appear valid but are compromised,** compensating controls such as topology containment, token binding, short credential lifetimes, privileged-flow monitoring, and anomaly detection should limit the blast radius

Governance and Evidence (Non-Negotiable)

A unified policy fabric is only credible if it produces audit-ready evidence and assigns ownership:

- Protect surface register with owners, SLOs, criticality, and approved communication paths
- Policy change history, approvals, rationale, simulation results, and canary results
- Current-state policy export and observed-flow-to-policy reconciliation
- Exception register with owner, scope, expiry or review date, rationale, and compensating controls
- Continuous verification reports covering drift, coverage gaps, stale inputs, newly observed dependencies, and unresolved conflicts
- Evidence exports suitable for security operations, audit, compliance, and incident response

Outcome

A unified policy fabric enables **one place to author segmentation intent** and **many places to enforce it consistently**, while keeping operations sustainable:

- Fewer duplicated policies across disparate tools
- Reduced change lead time and exception sprawl
- Stronger assurance that segmentation outcomes hold across IT, cloud-native, and OT/ICS environments
- Continuous, explainable evidence that trust boundaries are real—not just diagrammed

Operating Model: Outcome-Driven Microsegmentation in Iteration Loops

Visibility is the lever. Insufficient visibility increases implementation risk, lengthens deployment timelines, and raises the likelihood of operational disruption. Treat visibility as the control that makes policy design, rollout, and audits cheaper and safer.

However, visibility alone is not segmentation; it becomes a Zero Trust control only when observed dependencies are translated into an explicit, enforceable, continuously governed communication policy.

Principles:

- **Assess readiness and tier criticality.** Before enforcement, perform a gap assessment for crown jewels and critical systems, including current visibility, known dependencies, ownership, RACI, logging and monitoring coverage, rollback readiness, and the appropriate segmentation tier or model
- **Passive first.** Capture flows and enrich them with owners, tiers, labels, and risk. Use active discovery sparingly in OT/ICS
- **Define protect surfaces as business assets, not IPs.** Assign each a named owner, RACI, criticality tier, and SLO covering availability, security, logging, and monitoring expectations
- **Tighten in loops.** Observe → hypothesize → simulate → canary → enforce → monitor → retire exceptions. Define canary scope, success criteria, rollback triggers, and responsible approvers before enforcement begins
- **Canary and rollback criteria.** Before moving from visibility or simulation into enforcement, teams should define the affected protect surface, enforcement plane, expected flows, test duration, acceptable latency/error thresholds, business owner approval, rollback authority, and rollback trigger. Examples of rollback triggers include blocked critical flows, unacceptable latency or jitter, failed health checks, increased application error rates, OT safety concerns, or loss of required monitoring evidence
- **Measure outcomes.** Track % workloads with known flows; % protect surfaces with explicit allow-lists; mean time to isolate (MTTI); exception half-life; change lead time

Step hooks (what changes by phase):

- **Design:** Choose enforcement planes by latency and safety, not fashion. In OT cells, agentless first; nano only where hosts can bear it
- **Policy:** Express intent in labels/identity; treat IPs as an implementation detail
- **Operations:** Continuous verification + drift alerts; regular exception burn-down; audit-ready evidence by default

Many readers (and CSA companion documents) expect microsegmentation guidance to be organized around the familiar **Zero Trust five-step implementation flow**: protect surface → flows → architecture → policy → operations. The table below keeps that recognizable structure, but maps each step to the **outcome-bucket** approach used in this paper, so teams can start from the business risk they're trying to reduce (e.g., ransomware containment, regulatory ring-fencing, OT safety isolation, DevOps dynamism, user/device least privilege) while still producing the standard artifacts and evidence expected in Zero Trust programs. This approach aligns with CSA's guidance on Defining the Zero Trust Protect Surface, mapping transaction flows as part of the broader CSA Zero Trust canon.

Short examples help show how the model applies. For ransomware containment, teams may begin with a crown-jewel application and its administrative paths, map the transaction flows, and enforce explicit allow-lists before expanding to adjacent services. For OT segmentation, teams may begin with a zone/cell conduit and apply gateway or choke-point enforcement with safety-aware rollback. For agentic AI tool access, teams may define an approved tool inventory, deny unapproved egress by default, and monitor policy verdicts and newly observed dependencies.

Agentic workload containment (Bucket 6) should be treated consistently across all five implementation steps. Because agents may discover tools, invoke APIs, retrieve data, and act across trust domains, teams should define agent protect surfaces, map agent-to-tool and agent-to-data flows, choose both identity/session and local runtime enforcement, create explicit allow-lists for permitted tool use, and continuously monitor policy verdicts, egress attempts, and newly observed dependencies. Agentic denials should be triaged as policy violation, stale inventory, or newly discovered dependency. Incident routing should use the agent, tool, or workflow owner recorded in the approved inventory and RACI. Baselines should be revalidated after model, prompt, tool, permission, deployment, or workflow changes because normal agent behavior can shift as those inputs change.

CSA Five Steps (Expected Framing)	What it Means in an Outcome-Bucket Microsegmentation Program	Output/Evidence (Keep it Light, Auditable)	Outcome-Bucket Hooks (How the Step Changes by Bucket)
1. Define the Protect Surface	Pick the business assets (DAAS) you will segment around; assign owners; decide required granularity (macro/micro/nano), and determine which enforcement planes are viable.	Protect-surface register; owners/RACI; SLOs; "must-allow" dependencies (initial hypothesis).	B1 (Blast-radius): start with crown-jewel apps/admin paths. B2 (Regulatory): define scope boundary (PCI/PHI/export). B3 (OT): zone/cell protect surfaces + conduits. B4 (DevOps): namespaces/services. B5 (User/device): brokered access to specific apps. B6: define agents, tools, model endpoints, data sources, orchestration services, and tool gateways as protect surfaces; assign owners for agent behavior and permitted tool access.
2. Map the Transaction Flows	Establish ground truth flows, enriched with owners/tags/identities; prioritize passive visibility; confirm which flows are truly required.	Flow inventory (who/what→what); dependency map; "unknown/unowned flows" list.	B3 (OT): passive-first + minimal active probing. B4 (DevOps): time-bound maps (ephemeral). B1/B2: focus east-west + privileged mgmt flows. B6: map agent-to-tool, agent-to-model,

			agent-to-data, agent-to-API, and agent-to-orchestrator flows; identify unauthorized egress paths and unmanaged tool discovery.
3. Build a Zero Trust Architecture	Choose enforcement placement based on latency/safety and operability: infra/workload/session planes; decide where topology-defined vs. connection-defined policies carry the intent.	Reference architecture; plane selection rationale; failure-mode assumptions; rollout model (simulate→canary→enforce).	B3 (OT): agentless/gateway-first, deterministic choke points. B4: CNI/mesh + service identity where needed. B5: session plane first, then local containment inside the enclave. B6: combine identity/session controls for approved tool and service access with local workload/container enforcement to prevent unauthorized egress or tool invocation.
4. Create and Enforce Policy	Translate intent into explicit allow-lists (labels/identity first; IPs as implementation detail); enforce iteratively; manage exceptions with expiry.	Policy library; simulation report; canary results; exception register with "half-life."	B2: audit-ready ring-fence rules. B1: fast containment controls. B4: automation/GitOps required. B3: safety exceptions are explicit and governed. B6: express explicit allow-lists for which agents may reach which tools, APIs, models, data stores, and orchestration services; govern exceptions with expiry.
5. Monitor and Maintain	Continuous verification: drift detection, "new dependency" alerts, and outcome metrics (containment + operational speed).	KPIs: % surfaces with explicit allow-lists, MTTI, change lead time, exception burn-down, and drift logs.	All buckets: measure outcomes; B3: safety validation + change windows; B4: continuous policy validation against deploy events; B2: scope control evidence for audits. B6: monitor agent egress, tool invocation paths, policy verdicts, and newly observed dependencies; alert on agent behavior that attempts to reach unapproved services.

Table 10: CSA Five-Step Implementation Framing

Technology and Vendor Evaluation (Checklist)

Technology selection for microsegmentation is frequently derailed by vendor-driven terminology and "feature checklists" that obscure what matters: which segmentation outcomes you must achieve, where policy will be enforced (planes), how intent is expressed (topology-defined vs. connection-defined), and how safely the program can be operated over time. This checklist is therefore designed to be evidence-led and outcome-first, helping teams evaluate whether a solution can (1) deliver the required controls in IT, cloud-native, and OT/ICS constraints, (2) reduce operational friction and policy sprawl rather than adding another silo, and (3) support a safe lifecycle (visibility → simulation → staged rollout → continuous verification) with clear failure-mode behavior, auditability, and exception governance.

The checklist below should be interpreted in order of priority. Baseline requirements are capabilities that materially affect the solution's ability to deliver segmentation outcomes safely. Advanced requirements improve scale, automation, and assurance. Context-specific requirements, especially OT, disconnected edge, serverless, and agentic AI support, should be weighted according to the environment being protected.

- ☐ **Policy Model and Expressiveness**
  - Can policy be authored in **durable selectors** (identity/labels/posture), not just IPs?
  - Supports **topology-defined**, **connection-defined**, and **hybrid layering** without duplicating policy intent?
    - **Evidence:** Sample policies; mapping from intent → enforcement artifacts; demo changing IPs without policy rewrite
    - **Red Flags:** "We do identity" but only for users; workloads, services, processes, and non-human identities remain IP/port-based; policy is described as identity-aware but cannot bind enforcement to a durable workload or service identity
- ☐ **Discovery, Visibility, and Dependency Mapping**
  - Passive-first flow collection? Enrichment (e.g., owners, app tiers, labels, risk)?
  - Can it show **"who talked to what"** with time bounds and explain policy decisions?
    - **Evidence:** Flow maps; dependency reports; "unknown flow" handling; OT-safe discovery modes
    - **Red Flags:** Requires active scanning across the board; cannot attribute flows to owners/services
- ☐ **Policy Lifecycle and Safety (Simulate → Stage → Enforce)**
  - Supports **simulation/permissive mode**, canary rollout, and rapid rollback?
  - Exception management with **expiry ("exception half-life")** and approvals?
    - **Evidence:** Pre-enforcement impact report; change logs; rollback demo; exception aging dashboard
    - **Red Flags:** Enforcement is binary; rollback is manual; exceptions never expire
- ☐ **Enforcement Coverage and Placement**
  - Which enforcement planes are supported (infra/workload/session)?
  - Agentless options for OT/legacy? Gateways for "can't-run-an-agent" endpoints?

        - **Evidence:** Coverage matrix by environment (VMs/K8s/bare metal/cloud/OT)
        - **Red Flags:** "Works everywhere," when in truth, broad claim coverage depends heavily on agent deployment and runtime realities and limitations (can they be deployed at all?)

- ☐ **Identity and Key Lifecycle**
    - Workload identity support (mTLS/service identity), credential rotation, revocation, and audit?
    - Integration with IdP, PKI, and device posture sources?
        - **Evidence:** Rotation policy; cert lifecycle workflows; revocation blast radius test
        - **Red Flags:** Long-lived shared secrets; unclear key ownership
- ☐ **OT/ICS Safety and Operational Constraints**
    - Determinism/latency: Can enforcement be placed to respect safety constraints?
    - Explicit modes for fail-safe vs. fail-closed, with documented exceptions?
        - **Evidence:** Performance characterization plan; OT reference deployments; safety exception workflows
        - **Red Flags:** Assumes IT-style controls fit field/cell layers
- ☐ **Scale, Resilience, and Failure Behavior**
    - High availability (HA) control plane, degraded-mode behavior, and "what happens when telemetry is missing?"
        - **Evidence:** Failure-mode table; chaos testing results; capacity benchmarks
        - **Red Flags:** "Controller down = everything down" without clear mitigations
- ☐ **Integration and Operations**
    - SIEM/XDR/SOAR hooks; infrastructure as code (IaC)/policy as code; CMDB; ticketing; evidence exports for audit
        - **Evidence:** APIs; Terraform/GitOps examples; audit report outputs
        - **Red Flags:** GUI-only ops; limited exportability
- ☐ **Time-to-Value and Adoption**
    - Onboarding time for the first protect surface; operator workflow quality, and training burden
        - **Evidence:** Pilot plan; "first 30 days" runbook; sample KPIs
        - **Red Flags:** Requires professional services for routine policy changes

Conclusion and Future Outlook

Microsegmentation operationalizes Zero Trust by enforcing least-privilege communication within and across environments, reducing lateral movement, and limiting blast radius when compromise occurs. As shown throughout this guide, microsegmentation is best treated as an outcome-driven discipline rather than a single technology: the same security objectives can be achieved through different enforcement planes (infrastructure, workload, identity/session) and through different segmentation models (topology-defined, connection-defined, or layered combinations of both).

Successful programs share common characteristics: they define protect surfaces and transaction flows, express policy intent using durable attributes (identity, role, labels, and context), and validate enforcement continuously to prevent drift. In IT and cloud-native estates, policy-as-code practices, simulation/permissive modes, and staged enforcement reduce operational risk while improving consistency. In OT and ICS environments, where agent deployment and protocol constraints are common, segmentation must be engineered in close collaboration with safety and operations teams, favoring predictable enforcement points and explicitly governed exceptions.

A critical theme across real deployments is that microsegmentation is as much an operational control as a security control. If least-privilege connectivity is difficult to change, organizations tend to accumulate broad rules, permanent exceptions, and policy drift. Mature programs therefore treat operability, consistency, and evidence as success criteria alongside containment: segmentation must be repeatable, auditable, and practical to evolve as systems change.

Key takeaways for organizations considering microsegmentation include:

- Microsegmentation is a core Zero Trust enforcement mechanism that reduces blast radius and constrains lateral movement across IT, cloud, and OT/ICS environments
- Topology-defined and connection-defined models are complementary; mature designs layer both to achieve exposure reduction and deterministic containment
- Governance is essential: without ownership, lifecycle management, and continuous validation, segmentation degrades into policy sprawl, exceptions, and the reintroduction of implicit trust
- The most effective implementations are measurable, evidence-driven, and iteratively expanded by protect surface and outcome buckets

Strategically, microsegmentation is evolving from a network segmentation practice into a broader Zero Trust discipline for governing communication itself. As enterprises adopt distributed cloud, edge, OT, IoT, and agentic AI environments, trust boundaries can no longer be defined only by network location or static topology. Identity- and policy-defined connectivity, governed AI communication, continuous validation, and evidence-driven enforcement are reshaping how organizations reduce attack paths and contain compromise across modern digital ecosystems.

Future Outlook

Microsegmentation will continue to evolve as environments become more distributed, identity-centric, and automation-driven. Several trends are likely to shape how organizations implement segmentation outcomes over the next few years:

- **Increased adoption of connection-defined controls for cross-domain consistency.** As organizations operate across multiple clouds, sites, edge environments, and administrative domains, reliance on static addressing, routing symmetry, and ticket-driven changes to firewalls or security groups becomes a limiting factor. Connection-defined approaches, in which authorization precedes reachability and identity serves as a durable selector, are increasingly used to express least-privilege intent across these boundaries. This shift is driven as much by operational feasibility as by security: reducing duplicate policy changes, shortening approval cycles, and minimizing brittle IP dependencies can make least-privilege connectivity easier to sustain. Topology-defined controls will remain useful for local containment and deterministic enforcement, especially in OT/ICS and intra-domain hard boundaries, but connection-defined controls are likely to play a larger role in expressing segmentation intent across heterogeneous and fast-changing environments
- **Compliance evidence for connection-defined controls.** Where segmentation is enforced through connection-defined models, audit evidence should not rely only on firewall rules or network diagrams. Evidence should include identity binding, entitlement policy, policy-decision records, session establishment logs, certificate or credential lifecycle records, posture/context inputs, denied-session records, and proof that unauthorized subjects lacked reachability to the protected service
- **Stronger workload identity, attestation, and short-lived credentials.** Workload-to-workload segmentation will increasingly rely on stronger machine identity primitives, including certificate-based identity, workload identity frameworks, signed artifacts, runtime verification, hardware-backed identity where appropriate, and improved key lifecycle management. As these mechanisms mature, segmentation policy can be tied more directly to service identity, posture, deployment metadata, and attestation signals, improving auditability and reducing reliance on network identifiers. Short-lived credentials and continuous verification patterns also help reduce blast radius and accelerate response when compromise is suspected
- **Microsegmentation for agentic workloads and governed egress.** As AI systems become more agentic, segmentation must extend beyond traditional workload-to-workload communication into the runtime behavior of agents, containers, tools, and model access paths. Agentic workloads are often designed to discover, call, and chain tools across APIs, data stores, SaaS services, and internal systems, which makes unrestricted egress and ambient service reachability increasingly risky. Future segmentation architectures will need to combine identity-defined service access with local runtime containment, ensuring that an agent or container can reach only explicitly approved destinations, such as authorized model gateways,

tool gateways, DNS, health checks, or named internal services. Where feasible, local enforcement should capture process-, socket-, container-, destination-, and policy-verdict telemetry to help organizations distinguish permitted agent activity from unauthorized egress attempts. This is especially important as AI compresses the timelines for exploitation and misuse: if an exposed service or tool path is reachable, automated discovery and exploitation can outpace human approval, firewall changes, or patch cycles. Microsegmentation, therefore, becomes a foundational prerequisite for safe agentic AI adoption, not merely a compensating control after model-, prompt-, or application-layer defenses fail

- **Serverless and ephemeral workloads.** Serverless functions, short-lived containers, jobs, and automated agents require segmentation models that can bind policy to workload identity, deployment metadata, signed artifacts, runtime context, and, where available, attestation signals. Static network location is often insufficient for these ephemeral environments, so policy may need to be embedded into the application path, service invocation path, identity/session layer, or gateway pattern
- **Policy unification and "compile-to-enforcement" architectures.** A common operational challenge is drift across multiple enforcement systems (firewalls, cloud security groups, CNIs, service meshes, and session brokers). "Unified policy fabric" approaches—where intent is authored once and compiled into multiple enforcement points—will become more prevalent, enabling consistent outcomes across planes while reducing duplicated rule management
- **Automation, simulation, and continuous validation as defaults (telemetry-driven).** Mature microsegmentation programs will increasingly rely on continuous validation rather than periodic audits. Policy changes will be tested via simulation and staged rollout, and then continuously verified against observed behavior to detect drift, overly permissive exceptions, and enforcement gaps. This shift will be enabled by richer telemetry and analytics, including identity- and flow-level observability, workload and process telemetry where feasible, and higher-fidelity signals from kernel-level instrumentation (e.g., eBPF-based collection on supported platforms). AI/ML-assisted approaches may accelerate baselining, dependency inference, and anomaly detection but should be treated as decision support rather than a replacement for explicit policy intent and governance. Integration with SIEM/XDR platforms will increasingly serve as the operational feedback loop, correlating segmentation decisions, identity context, and runtime detections, to validate containment outcomes and to trigger response actions (e.g., tightening policies, isolating segments, or elevating verification requirements). This will increasingly include policy simulation, mirrored environments, and digital twins that allow teams to test proposed segmentation changes before deployment, especially in dynamic, OT, and AI-assisted environments where policy changes can have safety, availability, or cascading dependency impacts
- **Operator-augmenting tooling.** Sustained segmentation will depend on tooling that assists policy authoring, evidence review, role separation, onboarding, denial triage, and decision support
- **OT/ICS segmentation maturity via gateways and identity-aware conduits.** For OT environments, progress will remain constrained by endpoint limitations and deterministic traffic requirements. Growth is expected in segmentation patterns that preserve safety (e.g., zone/cell

boundaries, monitored conduits, gateway-based identity mediation) while improving visibility and governance. Incremental modernization, particularly around machine identity at gateways and secure remote access, will drive meaningful improvements without requiring invasive endpoint changes

- **Toward identity- and policy-defined connectivity as a default property.** As Zero Trust architectures mature, segmentation outcomes will increasingly be treated as a foundational connectivity property: identities, services, and policy determine permissible connections; enforcement is layered across session, workload, and infrastructure planes; and verification is continuous. This direction strengthens least-privilege by design while retaining the containment and safety guarantees provided by topology-defined controls when operationally necessary. This model also supports emerging domains such as tool-mediated and agentic AI systems, where agents, models, tools, APIs, and data paths may span multiple trust domains without requiring broad implicit reachability

Future CSA work may further separate identity-defined reachability as its own architectural pattern, building on SDP's authenticate/authorize-before-connect model and this paper's distinction between connection-defined admission and topology-defined containment.

Useful References

1. **Existing Guidance and Regulations:**
   a. CISA Zero Trust Maturity Model (2023)
   b. CISA, Secure Connectivity Principles for Operational Technology (OT), (2026)
   c. CISA, Microsegmentation in Zero Trust Part One: Introduction and Planning,(2025)
   d. NIST, IoT Security Maturity Model: NIST Cybersecurity Framework 1.1 Mappings, (2024)
   e. The NIST Cybersecurity Framework (CSF) 2.0 (2024)
   f. NIST SP 800-82 Rev. 3: Guide to OT Security, (2023)
   g. NIST SP 800-213, IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements, (2021)
   h. NIST IR 8259A, IoT Device Cybersecurity Capability Core Baseline, (2020)
   i. NIST Cybersecurity for IoT Program (2020)
   j. Applying Zero Trust Principles to Enterprise Mobility (2022)
   k. CISA Guidance on Edge Devices (2025)
   l. CISA Securing the Internet of Things, (2021)
   m. NIST Special Publication 800-207: Zero Trust Architecture (2020)
   n. CISA/NSA, Implement Network Segmentation and Encryption in Cloud Environments (2024)
   o. Australian Cyber Security Center (ACSC), Implementing Network Segmentation and Segregation (2025)
   p. CISA/NSA/ASD, Securing Edge Devices: Multi-Agency Guidance (2025)
   q. NIST Special Publication 1800-35: Implementing a Zero Trust Architecture (2025)
   r. NIST Special Publication 800-215: Guide to a Secure Enterprise Network Landscape (2022)
   s. ISA/IEC 62443 Standards, Zero Trust Outcomes Using (2024)
   t. ISO/IEC 27001 and Implementing a Zero Trust Security Model (2022)
   u. NIST SP 800-53 Rev. 5: AC-4, SC-7
   v. NIST SP 1800-35: The most current ZTA practice guide
   w. MITRE D3FEND: Specifically Network Isolation (D3-NI)
   x. OWASP Top 10 for LLM & Agentic Applications: ASI01/ASI02
   y. NIST AI RMF: Govern/Map functions for the agentic workloads section
2. **CSA Resources**
   a. Cloud Security Alliance (CSA), SDP Architecture Guide v3, (2026)
   a. CSA Zero Trust Guidance for IoT (2025)
   b. CSA Zero Trust Advancement Center (ZTAC) Resource Hub CSA (2026)
   c. CSA Map the Transaction Flows for Zero Trust, (2024)
   d. CSA Zero Trust Guidance for Critical Infrastructure (2024)
   e. CSA Zero Trust Guiding Principles v1.1, (2024)
   f. CSA Defining the Zero Trust Protect Surface, (2024)
   g. CSA Zero Trust Implementation Primer - the 5-step process in Draft (2025)
   h. Enabling Zero Trust for Cellular Networks (2026)

i. Zero Trust Guidance for Achieving Operational Resilience (2026)

3. **Zero Trust References**
   a. NSTAC Report to the President on Zero Trust and Trusted Identity Management, (2022)
   b. Department of Defense (DoD) Zero Trust Reference Architecture Version 2.0 (2022)
   c. DoD Zero Trust Capability Execution Roadmap (COA 1), (2023)
   d. DoD Zero Trust Strategy (2022)
4. **Industry Microsegmentation and Zero Trust**
   a. NetFoundry What Is Overlay Networking and Zero Trust Microsegmentation
   b. Gartner, Market Guide for Microsegmentation, (2025)
   c. Illumio Blog 3 Takeaways from the 2025 Gartner® Market Guide for Network Security Microsegmentation, (2025)
   d. Akamai, Microsegmentation Moves the Zero Trust Needle for Commerce, (2024)
   e. Elisity, Microsegmentation and Zero Trust: A Powerful Security Duo, (2023)
   f. Akamai, A Blueprint for Building a Zero Trust Architecture
   g. Kong, What Role Does Microsegmentation Play in Zero Trust Security?, (2024)
   h. Akamai, Exploring Key Use Cases for Microsegmentation, (2023)
   i. GigaOm, Microsegmentation: Implementing Zero Trust at the Network Level, (2024)
   j. Zero Trust Network Architecture and Microsegmentation (2023)
   k. HCLtech, Securing Your Data Center with NSX-T Microsegmentation, (2024)
   l. Ericom, The Role of Microsegmentation in Zero Trust Security, (2023)
   m. Ericom, Microsegmentation Is Key To Zero Trust Security. Here's Why (2023)
   n. Cyber.gov.au, Implementing Network Segmentation and Segregation, (2021)
   o. The Forrester Wave™: Microsegmentation Solutions, Q3 2024, (2024)
   p. Packet Protector, Ep. 79, PP079: Rethinking the Architecture of Microsegmentation, 2025

Glossary

- **Microsegmentation:** Micro-segmentation enforces least-privilege communication across workloads, applications, services, users, devices, and non-human identities by using durable attributes such as labels, tags, certificates, workload identity, posture, role, service-intent, or contextual policy. It primarily answers which subjects may communicate with which protected resources, under what conditions, and through which enforcement points. It should reduce reliance on static IP addressing where feasible but may still compile policy into IP-, route-, gateway-, host-, or cloud-native controls when required by the environment.
- **Macrosegmentation:** Macro-segmentation divides an estate into broad, well-defined zones and conduits using network-centric identifiers (e.g., IP ranges, subnets/VLANs, routed tiers, perimeter firewalls). It primarily answers the question of where traffic is allowed between large areas of the environment
- **Nano-segementation:** Nano-segmentation constrains communication within and between processes/services at runtime (per binary, per process identifier (PID), per container, per

API/verb). It answers what is being talked about and how (e.g., protocol, process, call), thereby enforcing execution-time intent

- **Topology-Defined Segmentation:** Segmentation in which policy defines where traffic may flow within or between enforcement domains, using constructs such as zones, routes, labels, tags, security groups, network policies, firewalls, gateways, or workload placement
- **Connection-Defined Segmentation:** Segmentation in which policy defines who or what may establish a session to a specific resource, and under what identity, posture, entitlement, service-intent, and contextual conditions
- **Protect Surface:** The data, application, asset, service, system, workflow, or operational function that the segmentation policy is intended to protect
- **Blast Radius:** The scope of systems, data, services, or operations that could be affected by a compromise, misconfiguration, or failure
- **Exception Half-Life:** A governance metric describing how long segmentation exceptions remain open before expiry, review, reduction, or removal
- **Policy Drift:** A condition in which the enforced segmentation state no longer matches intended policy because of environmental change, manual modification, stale metadata, unmanaged exceptions, or inconsistent enforcement
- **Coverage Drift:** A condition in which new or changed assets, workloads, services, or flows are not covered by the expected segmentation controls
- **Segmentation Entropy:** The tendency for environments to become less segmented over time as exceptions, unmanaged dependencies, and uncovered assets accumulate unless governance actively counteracts drift
- **Agentic Workload:** An AI-enabled workload, agent, tool runner, automation process, or orchestration component capable of taking actions, invoking tools, retrieving data, calling APIs, or interacting with other systems
- **Gateway/Conduit Enforcement:** Segmentation enforced at a boundary, gateway, broker, conduit, industrial firewall, or choke point, commonly used where endpoints cannot support agents or direct identity-based enforcement
- **Governed Egress:** Outbound communication that is explicitly authorized, constrained by policy, monitored, and reviewed rather than permitted broadly by default
- **Last-Known-Good Policy:** A previously validated policy state retained by an enforcement point for continued operation when the control plane or policy source is unavailable

Zero Trust Glossary References

- CSA Glossary (main/primary)
- On2IT ZT Glossary - Zero Trust Dictionary (John Kindervag)
- CSA SDP Glossary (Software Defined Perimeter)