Zero Trust Architecture: Operational Reference

% Zero Trust Architecture: Operational Reference % Turtini LLC % (Generated from repository sources)

{ width=90% }

Zero Trust Architecture Reference Library

This repository is a living reference for designing, operating, and sustaining **Zero Trust Architectures (ZTA)** in regulated federal environments.

It focuses on **how Zero Trust actually works in practice** once software is deployed — including architecture, automation, ownership, and Day-2 operations — rather than abstract frameworks or product descriptions.

The materials here are written for senior architects, platform engineers, security leaders, and program owners responsible for long-term outcomes.

Why This Repository Exists

Many Zero Trust efforts stall not because of missing tools, but because of:

- unclear ownership
- fragmented authority
- manual enforcement
- contractor-dependent operations
- undocumented institutional knowledge

This repository exists to capture **operational patterns that survive audits, personnel changes, and budget cycles**.

It is intentionally practical, opinionated, and grounded in real federal constraints.

What This Repository Is

- A **reference library**, not a campaign
- A **working vocabulary** for architecture and governance discussions
- A **bridge** between policy intent and operational reality
- A **tool for internal alignment**, documentation, and continuity

The content here is meant to be:

- read
- discussed

- adapted
- cited internally

What This Repository Is Not

- Not a product catalog
- Not a compliance checklist
- Not a theoretical whitepaper
- Not vendor marketing

This material is focused on **operating systems and organizations at scale**.

Contents

How to Use This Library

This repository is commonly used to:

- Frame architecture reviews
- Support Zero Trust planning discussions
- Document current-state and target-state operations
- Reduce reliance on oral history and contractor knowledge
- Prepare for audits without creating parallel documentation

Content is modular and intended to be reused internally.

Stewardship and Philosophy

Zero Trust is not a project.
It is an **operating posture**.

Sustainable Zero Trust requires:

- automation as enforcement
- identity as the control plane
- platforms as shared infrastructure
- ownership that survives personnel changes

This repository reflects that philosophy.

About Turtini

This library is maintained by **Turtini LLC**, a small U.S.-based mission partner focused on successful technology implementations in regulated environments.

Turtini publishes open, versioned technical artifacts to support:

- informed evaluation
- peer review
- operational clarity
- long-term institutional resilience

More information:

- https://turtini.com/approach
- https://turtini.github.io

License

Published under the MIT License.

All materials are provided as reference and should be reviewed and adapted to each environment.

Achieving “Optimal” Maturity

How OpenShift and Ansible Fulfill CISA’s Zero Trust Pillars

Federal agencies are no longer asking *whether* they should adopt Zero Trust.
The question now is how to reach **“Optimal” maturity** under the CISA Zero Trust Maturity Model (ZTMM) without introducing operational fragility or excessive complexity.

At Turtini, we work in regulated federal environments where Zero Trust is not an abstract framework, but an operational mandate. This document explains how **Red Hat OpenShift** and **Ansible Automation Platform** are used together to move agencies beyond *Initial* or *Advanced* maturity toward **automated, identity-driven, and continuously verified systems**.

This is written for architects, operators, and security teams responsible for real platforms — not slideware.

Zero Trust Is an Operational State, Not a Product

The ZTMM defines maturity levels, but it does not prescribe tooling.
Reaching **Optimal** maturity requires more than deploying software — it requires **integration, automation, and governance across the full lifecycle of systems**.

In practice, agencies stall when:

- controls are enforced manually
- platforms are treated as app-specific dependencies
- security is bolted on after deployment
- operators lack repeatable patterns

The sections below map each CISA Zero Trust pillar to concrete platform capabilities and operational practices.

Pillar 1: Identity — The New Perimeter

At Optimal maturity, identity is continuously validated and centrally governed.

Platform Approach

- **PIV/CAC Integration**
  OpenShift integrates with federal identity providers to enforce phishing-resistant MFA for administrative and developer access.
- **Centralized Identity Lifecycle Management**
  Red Hat Identity Management (IdM) provides policy-driven identity control across hybrid environments, reducing identity sprawl and manual account management.

Operational Outcome

Identity becomes the primary enforcement point — not network location.
Access decisions are consistent, auditable, and centrally managed.

Pillar 2: Devices — Continuous Compliance and the “Golden Image”

Optimal device security requires complete inventory, baseline enforcement, and automation.

Platform Approach

- **STIG Enforcement via Automation**
  Ansible Automation Platform enforces DISA STIG-aligned configurations across RHEL and OpenShift nodes, ensuring drift is detected and corrected automatically.
- **Golden Image Strategy**
  RHEL Image Builder is used to create hardened base images so systems start compliant, rather than being remediated after deployment.

Operational Outcome

Compliance becomes continuous rather than episodic.
Audits rely on evidence, not screenshots.

Pillar 3: Networks — Micro-Segmentation by Default

Flat networks introduce lateral movement risk and complicate enforcement.

Platform Approach

- **Service Mesh with Mutual TLS (mTLS)**
  OpenShift Service Mesh encrypts service-to-service traffic by default and enables identity-based communication policies.
- **Granular Network Policy**
  Network access is restricted at the workload level, limiting blast radius and improving visibility.

Operational Outcome

Network trust is explicit, enforced, and observable — not assumed.

Pillar 4: Applications & Workloads — From Silos to First-Class Platforms

CISA's Optimal stage requires automated deployment, monitoring, and governance of workloads.

Platform Approach

- **OpenShift as a First-Class Platform**
  OpenShift is treated as a shared enterprise platform rather than a hidden dependency of individual applications.
- **Built-In Security and Visibility**
  Platform-native controls support workload scanning, policy enforcement, and runtime visibility.

The NASA Model

In our work with NASA, OpenShift was re-aligned from being a sub-component of application stacks (such as IBM Maximo) to a standalone enterprise platform.

This shift enabled:

- broader workload reuse
- standardized security governance
- reduced vendor lock-in
- clearer operational ownership

Operational Outcome

Workloads are governed consistently, regardless of application origin.

Pillar 5: Data — Automation and Visibility

At Optimal maturity, data protection is automated and continuously monitored.

Platform Approach

- **Encrypted Software-Defined Storage**
  Red Hat OpenShift Data Foundation (ODF) provides integrated encryption for data at rest.
- **Automated Audit Logging**
  Automation ensures access and system logs are continuously forwarded to agency SIEMs for real-time visibility.

Operational Outcome

Data protection becomes systemic rather than application-specific.

Workforce Enablement Is Non-Negotiable

Technology alone does not achieve Optimal maturity.

Agencies must be able to **operate** what they deploy.

Turtini develops RHCSA, RHCE, and OpenShift-focused labs and study materials built around federal security constraints. These materials are designed to strengthen internal Day-2 operational capability and reduce long-term contractor dependence.

Moving Toward Optimal

Optimal Zero Trust maturity is achieved when:

- controls are automated
- platforms are shared and governed
- identity drives access
- operators understand the system they run

The patterns described here are intentionally practical and repeatable.
They are designed to support real federal environments — not idealized ones.

About This Repository

This document is maintained as part of Turtini's public technical reference materials.

- Website: https://turtini.com
- Technical index: https://turtini.github.io

All content is provided as reference and should be reviewed and adapted to each environment.

License

Published under the MIT License.

Mapping Red Hat OpenShift and Ansible to the CISA Zero Trust Maturity Model

Purpose

This document provides a practical mapping between the **CISA Zero Trust Maturity Model (ZTMM)** and real-world implementations using **Red Hat OpenShift** and **Red Hat Ansible Automation Platform**.

It is intended for:

- Federal architects and engineers
- ISSOs and security teams
- Program and platform owners responsible for Zero Trust execution

The focus is not on theory, but on **how agencies move from Initial or Advanced maturity toward Optimal maturity through automation, identity-driven policy, and operational consistency**.

Zero Trust as an Operational Model

CISA's ZTMM makes a critical distinction:

- *Initial and Advanced* maturity stages are often tool-driven
- *Optimal* maturity is **automation-driven and continuously enforced**

Across all pillars, reaching Optimal maturity requires:

- Policy-as-code
- Identity-centric access controls
- Continuous validation and enforcement
- Reduction of manual, exception-based processes

OpenShift and Ansible provide the control plane and automation fabric required to operate Zero Trust at scale.

Pillar 1: Identity

CISA Focus

- Phishing-resistant MFA
- Centralized identity governance
- Continuous validation of users and services

OpenShift & Ansible Implementation

- **PIV/CAC integration** with OpenShift authentication
- **Role-based access control (RBAC)** enforced consistently across clusters
- **Red Hat Identity Management (IdM)** for centralized identity lifecycle management
- **Automated policy enforcement** using Ansible to eliminate manual access drift

Outcome

Identity becomes the security boundary — not network location — with policies enforced consistently across hybrid environments.

Pillar 2: Devices

CISA Focus

- Asset inventory
- Continuous compliance
- Elimination of unmanaged or unknown devices

OpenShift & Ansible Implementation

- **STIG enforcement via Ansible Automation Platform**
- **RHEL Image Builder** used to create hardened “golden images”
- Automated remediation when configuration drift is detected
- Consistent baselines across on-prem, cloud, and edge environments

Outcome

Devices start compliant and remain compliant without relying on periodic, manual audits.

Pillar 3: Networks

CISA Focus

- Micro-segmentation
- Encrypted communications
- Restricted lateral movement

OpenShift & Ansible Implementation

- **OpenShift Service Mesh (Istio)** for mutual TLS (mTLS)
- Identity-aware network policies at the workload level
- Encryption in transit enforced by default
- Reduced blast radius through service-to-service authorization

Outcome

Network trust is eliminated. Every request is authenticated, authorized, and encrypted.

Pillar 4: Applications & Workloads

CISA Focus

- Secure-by-default deployments
- Continuous monitoring
- Standardized governance

OpenShift & Ansible Implementation

- OpenShift operated as a **first-class enterprise platform**, not an application-specific dependency
- Automated build, deploy, and policy enforcement pipelines
- Consistent workload governance across environments
- Security controls integrated into Day-1 and Day-2 operations

Outcome

Applications are deployed and operated within a governed platform rather than managed as isolated exceptions.

Pillar 5: Data

CISA Focus

- Encryption at rest and in transit
- Access visibility
- Auditability

OpenShift & Ansible Implementation

- **OpenShift Data Foundation (ODF)** with built-in encryption
- Automated storage policy enforcement
- Centralized audit logging
- Integration with agency SIEM platforms for continuous visibility

Outcome

Data protections are enforced automatically, with access patterns visible and auditable in real time.

Workforce Enablement: The Missing Pillar

While not formally listed as a ZTMM pillar, **workforce readiness** determines whether Zero Trust succeeds or stalls.

Turtini Approach

- Hands-on RHCSA, RHCE, and OpenShift (DO180 / EX280) labs
- Federal-specific scenarios aligned with STIG and Zero Trust constraints
- Emphasis on verification habits and operational discipline
- Reduced reliance on external contractors for Day-2 operations

Outcome

Zero Trust becomes sustainable because internal teams can operate and evolve the platform.

Summary

Reaching **Optimal Zero Trust maturity** requires more than deploying tools.

It requires:

- Automation over documentation
- Platforms over point solutions
- Policy enforced continuously, not reviewed periodically

Red Hat OpenShift and Ansible provide the technical foundation.
Turtini focuses on making that foundation **operable, auditable, and sustainable** in federal environments.

Related Artifacts

- `zero-trust-cisa-mapping.md`
- `ztmm-mapping.png`
- https://turtini.com/approach
- https://turtini.github.io

Implementation Considerations for Zero Trust on OpenShift

Purpose

This document captures practical considerations, constraints, and lessons learned when implementing Zero Trust architectures using **Red Hat OpenShift** and **Ansible Automation Platform** in regulated federal environments.

It is intended to supplement architectural mappings by addressing:

- Operational realities
- Organizational friction points
- Common failure modes
- Design decisions that materially affect outcomes

This is not a checklist. It is guidance drawn from implementation experience.

Zero Trust Is an Operating Model, Not a Project

One of the most common failure modes in Zero Trust initiatives is treating Zero Trust as a discrete delivery milestone.

Zero Trust is not “implemented” and completed.
It is **operated continuously**.

Implication:

- Architecture decisions must prioritize repeatability
- Controls must be enforceable by automation
- Teams must be structured for sustained Day-2 operations

Platforms that cannot be operated consistently at scale will not reach Optimal maturity.

Identity Integration Requires Early Alignment

Common Pitfall

Identity systems are often treated as a downstream integration.

Reality

Identity becomes the control plane for:

- Cluster access
- API authorization
- Service-to-service communication
- Automation credentials

Considerations

- Align early with enterprise IdM, PIV/CAC, and directory owners
- Avoid local identity silos inside platforms
- Design RBAC models that match organizational roles, not individuals

Delayed identity integration creates long-term technical debt that is difficult to unwind.

Automation Must Be Authoritative

Common Pitfall

Automation is introduced as an optional convenience.

Reality

In Optimal maturity, **automation is the enforcement mechanism**.

Considerations

- Manual changes should be considered drift by default
- Automation must be trusted to remediate non-compliance
- Exceptions should be explicit, time-bound, and auditable

If humans remain the primary enforcement mechanism, Zero Trust will stall at Advanced maturity.

Platform Scope Matters More Than Tool Selection

Common Pitfall

OpenShift is deployed to support a single application or vendor workload.

Reality

Zero Trust benefits emerge when platforms are shared, governed, and standardized.

Considerations

- Treat OpenShift as an enterprise platform, not middleware
- Avoid application-specific clusters when possible
- Standardize patterns across environments to reduce cognitive load

Siloed platforms lead to inconsistent security posture and fragmented governance.

Day-2 Operations Are Where Zero Trust Succeeds or Fails

Common Pitfall

Operational ownership is unclear once platforms are live.

Reality

Zero Trust requires continuous validation, monitoring, and adjustment.

Considerations

- Clearly define platform ownership versus application ownership
- Invest in verification habits, not just deployment automation
- Ensure operational teams can explain and validate system state

If Day-2 ownership is outsourced or under-resourced, Zero Trust controls erode over time.

Logging and Visibility Are Not Optional

Common Pitfall

Logs are treated as troubleshooting artifacts rather than security signals.

Reality

Auditability and visibility are core Zero Trust requirements.

Considerations

- Centralize logs early
- Ensure identity context is preserved in logs
- Validate that security teams can query and interpret platform telemetry

Controls that cannot be observed cannot be trusted.

Workforce Enablement Is a Dependency

Common Pitfall

Training is deferred until after platform delivery.

Reality

Zero Trust architectures require skilled operators to remain effective.

Considerations

- Align training with the actual platform implementation
- Use hands-on labs that reflect production constraints
- Reduce reliance on external contractors for routine operations

Zero Trust maturity is limited by the least prepared operational role.

Incremental Progress Is Normal — Drift Is Not

Reality

Most agencies will operate across multiple maturity stages simultaneously.

Considerations

- Accept phased progress across pillars
- Avoid introducing unmanaged exceptions
- Track drift explicitly and remediate intentionally

Zero Trust is resilient when deviation is visible and reversible.

Summary

Successful Zero Trust implementations share common characteristics:

- Identity-driven control planes
- Automation as the source of truth
- Platforms designed for long-term operation
- Teams equipped to validate and sustain controls

OpenShift and Ansible enable these patterns — but architecture alone is insufficient.

Zero Trust succeeds when **technology, process, and ownership are aligned**.

Related Documents

- `mappings/cisa-pillars.md`
- `zero-trust-cisa-mapping.md`
- `ztmm-mapping.png`
- https://turtini.com/approach

Common Zero Trust Anti-Patterns

Purpose

This document describes recurring anti-patterns observed in Zero Trust initiatives across regulated environments.

These are not theoretical mistakes. They are patterns that consistently slow progress, introduce hidden risk, or stall organizations at **Initial** or **Advanced** maturity levels under the CISA Zero Trust Maturity Model (ZTMM).

Identifying and avoiding these early materially improves outcomes.

Anti-Pattern 1: Zero Trust as a Compliance Checkbox

Description

Zero Trust is framed as a documentation or audit exercise rather than an operating model.

Symptoms

- Heavy focus on policy artifacts
- Minimal change to system behavior
- Manual enforcement remains the norm

Why It Fails

Compliance without enforcement does not change risk posture.

Zero Trust requires systems that **continuously enforce intent**, not static attestations.

Anti-Pattern 2: Tool Acquisition Before Architecture

Description

Products are purchased before defining how they will be operated together.

Symptoms

- Disconnected security tools
- Overlapping capabilities
- Unclear ownership boundaries

Why It Fails

Zero Trust maturity depends more on **integration and operating discipline** than on individual tools.

Architecture must precede acquisition.

Anti-Pattern 3: Treating Identity as an Afterthought

Description

Identity integration is deferred until after platform deployment.

Symptoms

- Local accounts proliferate
- RBAC models drift from organizational structure
- Emergency access becomes permanent

Why It Fails

Identity is the control plane for Zero Trust.

Retroactively aligning identity is costly, disruptive, and often incomplete.

Anti-Pattern 4: Manual Exceptions Become Permanent

Description

Temporary workarounds are introduced to meet delivery timelines.

Symptoms

- “Just for now” permissions
- Untracked firewall exceptions
- Hard-coded credentials

Why It Fails

Unmanaged exceptions accumulate faster than they are removed.

Zero Trust architectures degrade quietly when drift is not explicit and reversible.

Anti-Pattern 5: Platform Siloing by Application

Description

Platforms are deployed per application or vendor rather than as shared infrastructure.

Symptoms

- Duplicate clusters
- Inconsistent security posture
- Fragmented operational practices

Why It Fails

Zero Trust benefits emerge through **standardization and reuse**.

Siloed platforms prevent consistent enforcement and visibility.

Anti-Pattern 6: Automation as Optional Convenience

Description

Automation is introduced but not treated as authoritative.

Symptoms

- Manual changes override automation
- Configuration drift is tolerated
- Automation lacks enforcement authority

Why It Fails

Zero Trust cannot scale with human enforcement.

Automation must be trusted to **detect and correct drift**, not just accelerate delivery.

Anti-Pattern 7: Logging Without Context

Description

Logs are collected but lack identity, workload, or policy context.

Symptoms

- Logs are difficult to interpret
- Security teams cannot correlate events
- Audit findings require manual reconstruction

Why It Fails

Visibility without context does not support trust decisions.

Logs must explain **who did what, where, and why**.

Anti-Pattern 8: Day-2 Operations Are Under-Resourced

Description

Operational ownership is unclear or deprioritized after go-live.

Symptoms

- Platform teams are thinly staffed
- Knowledge lives with external contractors
- Validation habits erode over time

Why It Fails

Zero Trust is sustained through continuous operation.

Under-resourced Day-2 teams cannot maintain enforcement integrity.

Anti-Pattern 9: Training Detached From Reality

Description

Training content does not reflect production constraints.

Symptoms

- Labs ignore STIG requirements
- Examples differ from deployed architecture
- Operators lack confidence in real environments

Why It Fails

Operators trained in abstraction struggle under real constraints.

Training must mirror the environment it prepares people to run.

Anti-Pattern 10: Chasing “Optimal” Without Foundations

Description

Organizations attempt to implement advanced controls before establishing baselines.

Symptoms

- Complex architectures with fragile foundations
- Inconsistent enforcement
- Frequent rollback of advanced features

Why It Fails

Zero Trust maturity is incremental.

Skipping foundational controls introduces instability rather than progress.

Summary

Organizations that stall in Zero Trust maturity usually do so because of **operational patterns**, not missing technology.

Avoiding these anti-patterns enables:

- Sustained enforcement
- Clear ownership
- Predictable progress across pillars

Zero Trust succeeds when architecture, automation, and operations reinforce each other.

Related Documents

- `mappings/cisa-pillars.md`
- `notes/implementation-considerations.md`
- `zero-trust-cisa-mapping.md`
- https://turtini.com/approach

Day-2 Operating Model

Purpose

This document describes a practical Day-2 operating model for Zero Trust environments built on Red Hat OpenShift, RHEL, and Ansible.

Zero Trust maturity is not achieved at deployment. It is sustained through **repeatable operations**, **clear ownership**, and **continuous validation**.

Day-2 is where most Zero Trust initiatives succeed or quietly fail.

What Day-2 Means in a Zero Trust Context

Day-2 operations encompass everything that happens **after** initial deployment:

- Configuration drift management
- Access lifecycle enforcement
- Patch and image lifecycle
- Policy validation
- Incident response and recovery
- Audit readiness

In Zero Trust, Day-2 is not “maintenance.”
It is **active enforcement of intent**.

Core Day-2 Principles

1. Automation Is the Source of Truth

Manual changes are permitted only when:

- They are documented
- They are temporary
- They are reconciled back into automation

If automation and reality diverge, automation must win.

---

2. Drift Is Expected — Silence Is Not

Drift will occur. That is normal.

What is unacceptable is:

- Drift without detection
- Drift without alerting
- Drift without correction

Drift should be:

- Visible
- Measurable
- Reversible

---

3. Validation Is a Habit, Not an Event

Validation activities must be:

- Scheduled
- Repeatable
- Low-friction

Examples:

- Regular access reviews
- Image compliance checks
- Policy conformance scans
- Log completeness validation

Audit readiness is an **output**, not a separate activity.

Recommended Day-2 Roles

A healthy operating model separates responsibilities without fragmenting ownership.

Platform Engineering

- Maintains OpenShift and RHEL baselines
- Owns cluster lifecycle
- Enforces platform-wide policies

Automation Owners

- Maintain Ansible roles and pipelines
- Encode compliance and recovery logic
- Prevent configuration drift

Security Stakeholders

- Define policy intent
- Review enforcement outcomes
- Validate audit artifacts

Application Teams

- Operate within guardrails
- Consume platform services
- Do not own platform security controls

Day-2 Failure Modes

Common failure signals include:

- “We’ll automate that later”
- Persistent emergency access
- Manual fixes during incidents
- Knowledge concentrated in contractors

These indicate an operating model that cannot scale.

Success Indicators

A healthy Day-2 Zero Trust model shows:

- Reduced manual intervention over time
- Fewer audit findings per cycle
- Faster recovery with less variance
- Confidence in enforcement, not fear of change

Summary

Zero Trust maturity is sustained by **operations discipline**, not tooling.

Organizations that invest early in Day-2 clarity move faster, recover better, and audit cleaner.

Related Documents

- `notes/organizational-alignment.md`
- `notes/ownership-models.md`
- `notes/common-antipatterns.md`

Ownership Models

Purpose

This document outlines ownership models that support sustained Zero Trust enforcement in complex, regulated environments.

Clear ownership reduces risk, accelerates decision-making, and prevents silent degradation.

Why Ownership Matters in Zero Trust

Zero Trust introduces:

- Continuous enforcement
- Identity-driven controls
- Automated correction

Without explicit ownership:

- Exceptions linger
- Drift becomes invisible
- Accountability diffuses

Ownership Is Not Control

Ownership defines:

- Who decides
- Who enforces
- Who validates

It does **not** mean:

- Centralized micromanagement
- Blocking delivery
- Replacing collaboration

Recommended Ownership Domains

Platform Ownership

Owns:

- OpenShift clusters
- RHEL baselines
- Core platform services

Accountable for:

- Stability
- Enforcement integrity
- Lifecycle management

Automation Ownership

Owns:

- Ansible roles
- Pipelines
- Drift remediation logic

Accountable for:

- Repeatability
- Auditability
- Recovery consistency

Identity Ownership

Owns:

- Identity sources
- Access models
- Role definitions

Accountable for:

- Least privilege
- Lifecycle enforcement
- De-provisioning accuracy

Security Ownership

Owns:

- Policy intent
- Risk acceptance
- Audit coordination

Accountable for:

- Clear requirements
- Validated outcomes
- Exception governance

Anti-Patterns in Ownership

Shared Ownership Without Authority

Everyone is responsible, no one can decide.

Contractor-Owned Enforcement

Knowledge leaves, controls decay.

Application-Owned Security Controls

Inconsistent enforcement and duplicated effort.

Effective Ownership Characteristics

- Explicit decision rights
- Documented escalation paths
- Time-bound exceptions
- Automation-backed enforcement

Ownership should be **boring**, predictable, and resilient to personnel changes.

Transitioning to Clear Ownership

Steps that work:

1. Map current responsibilities
2. Identify decision bottlenecks
3. Assign enforcement authority
4. Encode rules into automation
5. Review quarterly

This is organizational refactoring — it should be iterative.

Summary

Zero Trust architectures fail most often due to ownership ambiguity.

Clear ownership:

- Reduces friction
- Improves audit outcomes
- Enables sustained maturity

Ownership is an architectural decision.

Related Documents

- `notes/organizational-alignment.md`
- `notes/day-2-operating-model.md`
- `notes/common-antipatterns.md`

Organizational Alignment

Purpose

This document describes the organizational alignment required to sustain Zero Trust architectures beyond initial deployment.

Zero Trust is not blocked by technology. It is blocked by **unclear decision rights, misaligned incentives, and fragmented ownership**.

Zero Trust Is a Shared System

Zero Trust architectures cut across:

- Security
- Infrastructure
- Application delivery
- Compliance
- Procurement

Without alignment, each group optimizes locally — and the system degrades globally.

Common Misalignments

Security Owns Policy, Not Enforcement

Security teams define intent but lack authority over implementation.

Result:

- Policies that look correct
- Systems that behave differently

Platform Teams Own Uptime, Not Risk

Infrastructure teams are rewarded for stability, not enforcement.

Result:

- Exceptions accumulate
- Controls weaken quietly

Application Teams Are Held Accountable Without Authority

Teams are responsible for outcomes but cannot influence platform controls.

Result:

- Workarounds
- Shadow processes
- Loss of trust in the platform

Required Alignment Principles

1. Policy Intent Must Be Operationally Enforceable

If a policy cannot be automated, validated, and audited:

- It is incomplete
- It will decay

Policy authors must collaborate with platform and automation owners.

---

2. Enforcement Authority Must Be Explicit

Someone must be empowered to say:

- “This is not compliant”
- “This change must be reverted”
- “This exception has expired”

Ambiguity erodes Zero Trust faster than disagreement.

---

3. Incentives Must Reward Compliance, Not Workarounds

Teams should not be punished for:

- Slower delivery due to guardrails
- Enforced security controls

They should be rewarded for:

- Operating within standards
- Reducing exceptions over time

Recommended Governance Model

Strategic Governance

- Sets Zero Trust goals
- Prioritizes maturity milestones
- Resolves cross-team conflicts

Operational Governance

- Owns enforcement mechanisms
- Reviews drift and exceptions
- Maintains baselines

Execution Teams

- Build and operate within guardrails
- Provide feedback on friction
- Do not bypass controls

Warning Signs of Misalignment

- Repeated exception requests
- Security reviews after deployment
- Disputes over “who owns this”
- Platform teams acting as gatekeepers

These indicate structural issues, not individual failures.

Summary

Zero Trust succeeds when:

- Authority matches responsibility
- Policy is enforceable by design
- Teams share outcomes, not blame

Alignment is not optional — it is architectural.

Related Documents

- `notes/ownership-models.md`
- `notes/day-2-operating-model.md`
- `notes/common-antipatterns.md`