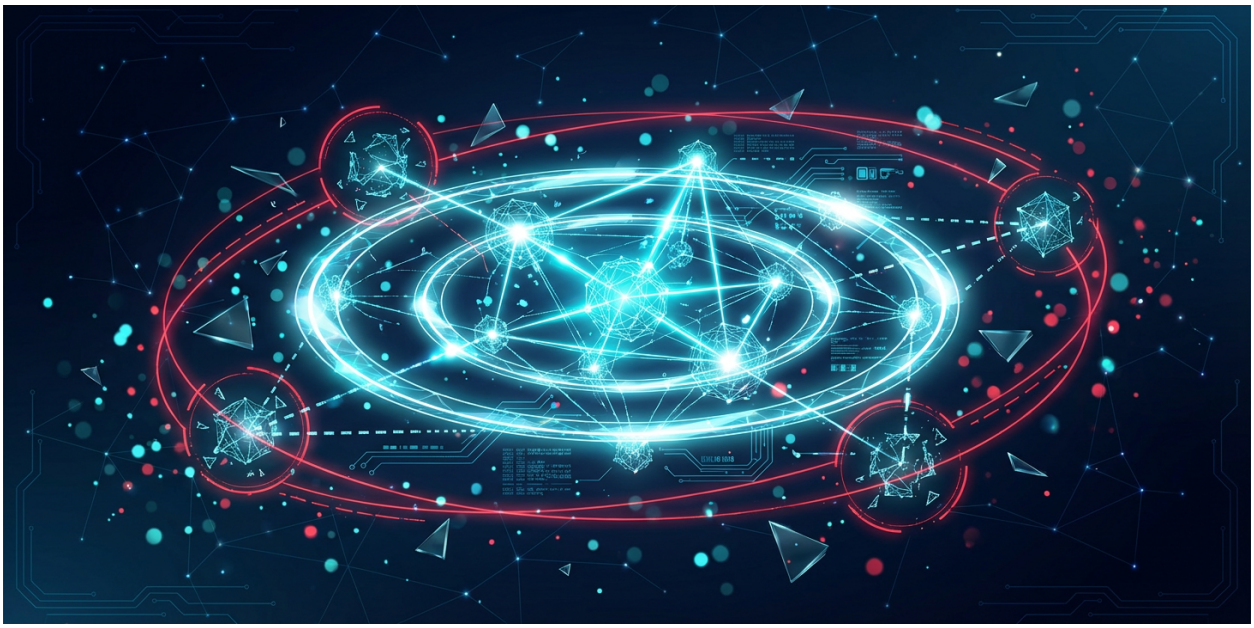

WHITEPAPER

Zero-Trust Architecture on the Disconnected Edge

Applying NIST 800-207 zero-trust ideas when identity, policy, and enforcement must work offline

April 2026

nova8 Technologies

nova8.io

Table of Contents

1. Executive Summary

Zero-trust architecture, as defined by NIST SP 800-207, is built on three core ideas: continuous verification of every request, least-privilege access for every actor, and the assumption that the network is always hostile. The reference architecture describes a model where access decisions are made dynamically by a Policy Decision Point (PDP) based on real-time signals including device posture, user identity, resource sensitivity, and behavioral analytics.

This model works well in enterprise environments with persistent, high-bandwidth connectivity to identity providers, policy engines, and telemetry backends. At the disconnected edge, these assumptions break. Tactical networks, remote industrial sites, maritime platforms, and contested environments cannot guarantee continuous connectivity to centralized services. When the policy engine is unreachable, access decisions either fail-open (destroying the zero-trust property) or fail-closed (halting mission-critical workloads). Neither outcome is acceptable.

This whitepaper examines which zero-trust principles survive at the disconnected edge and how they can be enforced locally. It proposes a three-domain trust model (device, runtime, workload) where each boundary is independently verifiable from local state, describes which security controls work without network connectivity, and addresses the policy reconciliation challenge when connectivity returns. The paper draws on NIST SP 800-207, NIST SP 800-123, and publicly available specifications from the UAPI Group and OCI community. It is intended for security architects, infrastructure leads, and program managers evaluating zero-trust approaches for edge deployments.

2. Why Cloud-Only Zero Trust Fails at the Edge

NIST SP 800-207 defines zero-trust architecture through three core ideas: continuous verification of every request, least-privilege access for every actor, and the assumption that the network is always hostile. The reference architecture describes a Policy Decision Point (PDP) and Policy Enforcement Point (PEP) model where access decisions are made dynamically based on real-time signals: device posture, user identity, resource sensitivity, and behavioral analytics.

This model implicitly assumes that the PDP is continuously reachable. Identity tokens are short-lived and require renewal. Policy updates propagate in near-real-time. Telemetry flows to a central analytics engine that informs risk scoring. When connectivity is reliable, these assumptions hold. At the disconnected edge (tactical networks, remote industrial sites, maritime platforms, or any environment where bandwidth is contested or intermittent) they break.

The failure mode is not graceful degradation but binary collapse: when the PDP is unreachable, access decisions either fail-open (destroying the zero-trust property) or fail-closed (halting mission-critical workloads). Neither outcome is acceptable for edge systems that must continue operating through connectivity loss.

The root cause is architectural: cloud-centric zero-trust designs centralize both decision-making and the state required for decisions. When the connection to that centralized state is severed, the local system lacks the information needed to make correct access decisions. The solution is not to cache cloud decisions locally (caches go stale) but to design a trust model where the local system carries sufficient state to make its own enforcement decisions.

- Short-lived tokens become unusable when the issuing identity provider is offline.
- Real-time policy evaluation requires a reachable policy engine; fail-open or fail-closed are both unacceptable.
- Centralized telemetry and risk scoring assume continuous data flow that disconnected environments cannot guarantee.
- Session re-authentication at cloud cadence is incompatible with high-latency or denied-link conditions.

3. Local Trust Boundaries: Device, Runtime, and Workload

A disconnected zero-trust model must decompose the system into trust domains that are independently verifiable from local state. The three natural boundaries on an edge device are the device itself (hardware identity and boot integrity), the container runtime (workload admission and isolation), and the individual workload (application identity and data access).

Each of these boundaries must carry its own verification roots, its own policy, and its own enforcement mechanisms. The critical property is independence: a failure or compromise at one boundary should not automatically propagate to the others, and each boundary should be verifiable without requiring network access (see Figure 1).

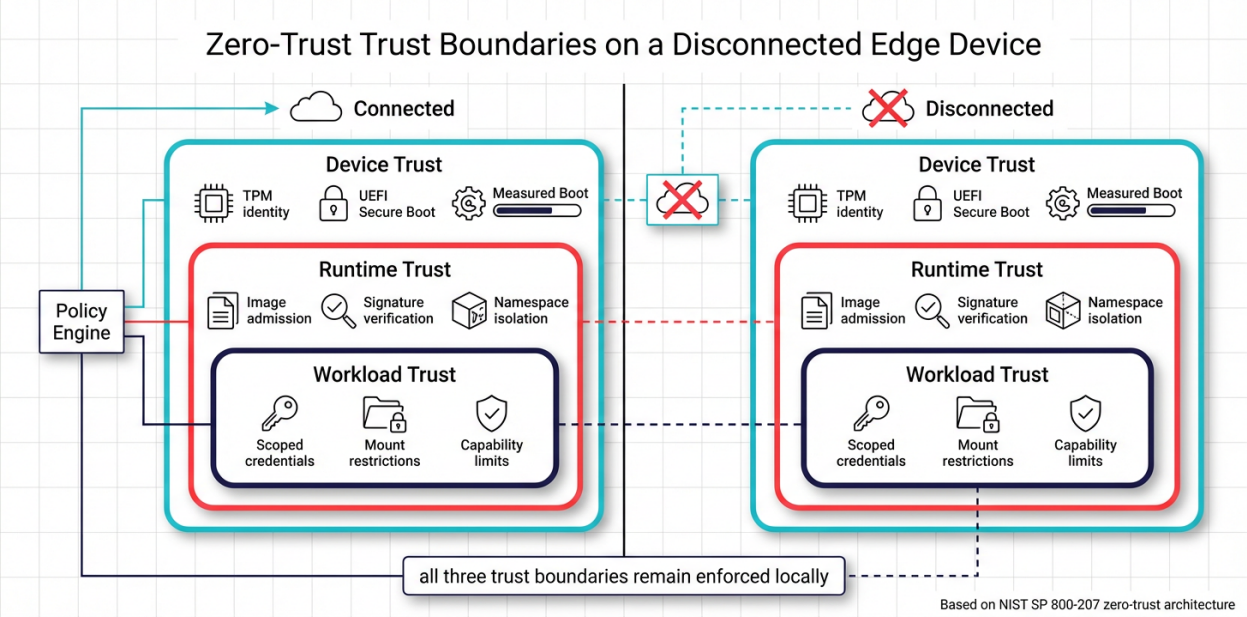

Figure 1: Zero-trust trust boundaries on a disconnected edge device, showing device, runtime, and workload domains.

This decomposition follows the principle of defense in depth applied to the trust model itself. Rather than a single perimeter (the network) or a single trust anchor (the cloud PDP), the system has multiple independent trust boundaries, each with its own verification mechanism and failure domain.

4. Device Trust: Hardware-Rooted Identity and Boot Integrity

Device trust starts at boot. UEFI Secure Boot, measured boot with TPM-backed attestation, and signed boot artifacts establish a hardware-rooted chain of trust that does not depend on any network service. The device either boots a verified image or it does not boot. This binary property is the foundation of the local trust model.

Hardware-rooted identity means the device's cryptographic identity is bound to a hardware component (typically a TPM or hardware security element) that cannot be cloned or extracted through software. This identity persists across reboots, image updates, and connectivity changes. It provides a stable anchor for all other trust decisions on the device.

Measured Boot and Remote Attestation

Measured boot extends the trust chain beyond the initial boot image. Each stage of the boot process records measurements (cryptographic hashes) of the next stage into TPM Platform Configuration Registers (PCRs). The resulting PCR values provide a tamper-evident record of the complete boot chain. When connectivity is available, these measurements can be presented to a remote attestation service for verification. When connectivity is unavailable, the measurements still provide local integrity evidence.

The combination of Secure Boot (which prevents execution of unsigned code) and measured boot (which records what code was executed) creates a device trust foundation that is entirely local. No network service is required for the device to verify its own boot integrity or to refuse to boot from tampered media.

Boot Media Substitution Prevention

UEFI Secure Boot, when properly configured with enrolled keys and disabled fallback boot paths, prevents the device from executing unsigned boot media. This is a critical physical security control for unattended edge devices. Without Secure Boot, an attacker with physical access can boot the device from a USB drive or network source, bypassing all host security controls.

The key enrollment must be properly scoped: only keys authorized by the platform operator should be enrolled, and the default keys shipped by hardware manufacturers should be evaluated and, where appropriate, replaced. The UEFI firmware should also disable alternative boot paths (legacy BIOS boot, PXE boot from untrusted networks) that could bypass the Secure Boot verification chain.

5. Runtime Trust: Workload Admission and Isolation

Runtime trust separates management functions from application workloads. By running distinct runtime instances for different trust domains, a compromise in one workload cannot reach management APIs or other workloads through a shared runtime socket. Each runtime instance enforces its own admission policy, image signature requirements, and resource constraints.

Split-Runtime Architecture

The split-runtime pattern runs separate container runtime instances for management and workload domains. The management runtime handles platform services (updates, telemetry, health monitoring) with appropriate privileges. The workload runtime handles application containers with more restrictive permissions. Because the runtime instances are separate processes with separate storage and separate API sockets, a compromise of the workload runtime cannot reach the management runtime's API or data.

This separation is enforced at the service manager level. Each runtime instance runs as an independent service with its own sandboxing profile, resource limits, and filesystem views. The isolation does not depend on the container runtime's internal security mechanisms alone; it is reinforced by the host's process isolation and service management infrastructure.

Image Admission Policy

Workload admission policy determines which container images are allowed to run on the device. For zero-trust compliance, this policy should enforce image signature verification: only images signed by authorized keys should be admitted. The verification keys and the policy definition should be embedded in the device's current system image so that admission decisions do not depend on network access.

The OCI image specification includes provisions for image signing through conventions such as cosign and sigstore. Verification can be performed entirely on-device using locally stored public keys and signature metadata. This makes image admission a locally enforceable zero-trust control that does not degrade during disconnection.

6. Locally Enforceable Policy: What Works Without Cloud Reach

Not all zero-trust controls require real-time cloud connectivity. Several critical enforcement mechanisms can operate entirely from local state, and a well-designed edge platform should maximize these local controls as the baseline security posture.

Controls That Work Offline

Image admission policy (verifying that only signed, authorized container images can run) is locally enforceable if signature verification keys and the allow-list are embedded in the device's current image. Namespace and cgroup isolation, seccomp profiles, Linux Security Module (LSM) policies, and capability restrictions are all kernel-enforced and do not require any network service. These mechanisms provide workload isolation, resource limits, and syscall filtering that remain effective regardless of connectivity state.

Service manager sandboxing (restricting filesystem access, network access, device access, and privilege escalation for each service) is configured at deployment time and enforced by the kernel. These sandboxing profiles are part of the system image and do not depend on any external service for enforcement.

Controls That Require Connectivity

The controls that genuinely require connectivity include certificate revocation checks (CRL or OCSP), real-time threat intelligence feeds, centralized audit log shipping, and policy updates from the central management plane. These should be designed as enhancements to the local baseline, not as prerequisites for safe operation. When the network is available, they add depth. When it is not, the local posture holds.

The key design principle is that the device must be safe by default before the network returns. No connectivity-dependent control should be the sole enforcement mechanism for a security property. If a control matters enough to enforce, it must have a locally enforceable equivalent or fallback.

- Image signature verification is locally enforceable with embedded verification keys.
- Kernel isolation (namespaces, cgroups, seccomp, LSM) operates without any network dependency.
- Capability restrictions and mount policies enforce least privilege from the service manager.
- Network-dependent controls (CRL checks, threat feeds, audit shipping) should enhance, not replace, local enforcement.

7. Policy Reconciliation When Connectivity Returns

Disconnected operation creates policy drift: the central control plane may have issued updated policies, revoked credentials, or changed workload configurations while the device was offline. When connectivity returns, the system must reconcile these changes without overriding locally enforced security decisions or creating a window of reduced protection.

Deterministic Reconciliation

Reconciliation should be deterministic and conflict-aware. If the central policy revokes a workload that is currently running on a disconnected device, the reconciliation process must apply that revocation when connectivity returns, but it should do so through the same controlled rollout mechanism used for any other change, not as an uncoordinated immediate kill. Abrupt state changes during reconnection create their own operational risks (see Figure 2).

The reconciliation algorithm must be deterministic: the same disconnected state plus the same pending policy changes must produce the same outcome every time. Non-deterministic reconciliation creates audit uncertainty and makes it impossible to predict the system's behavior during reconnection.

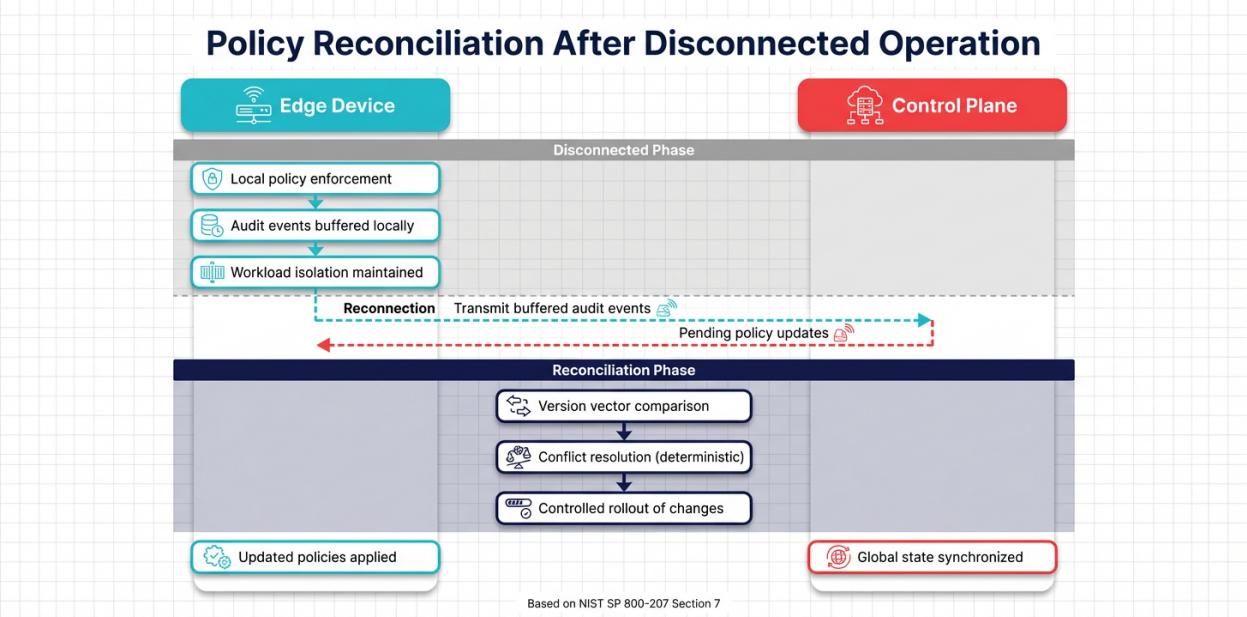

Figure 2: Policy reconciliation flow after disconnected operation, from local enforcement through deterministic reconnection.

Audit Event Continuity

Audit evidence continuity is equally important. Events that occurred during disconnected operation (workload starts, policy enforcement decisions, health check results, any break-glass access) must be buffered locally and transmitted when connectivity is restored. The central control plane should be able to reconstruct a complete timeline of device behavior across the disconnection gap.

Policy version vectors or timestamps should prevent stale updates from overwriting more recent local state. If a device applied a locally cached policy update during disconnection that is more recent than a pending central update, the reconciliation process must recognize the conflict and resolve it deterministically rather than blindly applying the central version.

8. Design Implications for Platform Selection

Teams evaluating edge platforms for zero-trust compliance should assess which trust properties are locally enforceable and which degrade or disappear under disconnection. A platform that achieves zero-trust properties only when connected to its cloud control plane is not a zero-trust edge platform; it is a cloud-managed platform with an edge presence.

Key Evaluation Questions

- Does the device establish identity from hardware roots without network assistance?
- Does workload admission enforce image signing policy locally, not only when the cloud is reachable?
- Does isolation hold when the central policy engine is unreachable?
- Is reconciliation deterministic and auditable?
- Can the platform demonstrate to assessors (per NIST SP 800-207 Section 7) that zero-trust properties are maintained during realistic operational scenarios?
- Which zero-trust properties degrade under disconnection, and is the degradation documented and acceptable?
- Is device identity hardware-rooted and independent of network-issued tokens?
- Can the platform produce audit evidence continuity across disconnection gaps?

The answers to these questions distinguish platforms that provide genuine zero-trust properties at the edge from platforms that provide zero-trust marketing with cloud-dependent enforcement. For deployments in contested, disconnected, or intermittently connected environments, the distinction is operationally critical.

9. How nova8 Technologies Aligns with These Principles

nova8 Technologies applies publicly documented zero-trust principles, including NIST SP 800-207 trust boundaries, hardware-rooted device identity, locally enforceable policy, and deterministic reconciliation, in its edge platform architecture. This paper describes the public architectural pattern and evaluation criteria, not product implementation details.

The platform architecture is designed to maintain zero-trust properties during disconnected operation by embedding trust verification, admission policy, and isolation enforcement in the deployed system image rather than depending on continuous connectivity to a cloud control plane.

nova8 Technologies is aligning infrastructure and development practices with CMMC Level 2 expectations. The company holds U.S. Provisional Patent Applications 63/897,352, 63/897,609, 63/903,132, 63/903,161, 63/903,164, and 63/903,168 related to edge computing, security architecture, and operational resilience innovations.

For more information, visit nova8.io or contact the team at contact@nova8.io.

10. References

1. NIST, "SP 800-207: Zero Trust Architecture," August 2020, csrc.nist.gov/pubs/sp/800/207/final

2. NIST, "SP 800-123: Guide to General Server Security," July 2008, csrc.nist.gov/pubs/sp/800/123/final

3. NIST, "SP 800-190: Application Container Security Guide," September 2017, csrc.nist.gov/pubs/sp/800/190/final

4. NSA, "Commercial National Security Algorithm Suite 2.0," September 2022, media.defense.gov

5. CISA, "Zero Trust Maturity Model," 2023, cisa.gov/zero-trust-maturity-model

6. UAPI Group, "Unified Kernel Images (UKI) Specification," uapi-group.org/specifications/specs/unified_kernel_image/

7. systemd, "Automatic Boot Assessment," systemd.io/AUTOMATIC_BOOT_ASSESSMENT/

8. OCI, "Open Container Initiative Image Format Specification," opencontainers.org

9. NIST, "FIPS 203: ML-KEM Standard," August 2024, csrc.nist.gov/pubs/fips/203/final

10. NIST, "FIPS 204: ML-DSA Standard," August 2024, csrc.nist.gov/pubs/fips/204/final

11. Trusted Computing Group, "TPM 2.0 Library Specification," trustedcomputinggroup.org

12. UEFI Forum, "UEFI Specification," uefi.org/specifications