AriaOS Resiliency Validation

Eight-Plus Days of Continuous Chaos Testing
Ubuntu 24 on Commodity Hardware
Failure-First Governance Under Network Isolation

Test Platform: HP G8 Laptop (5-Year-Old Class Device)
Test Period: 8+ Consecutive Days
December 26, 2025

Resilient Mind AI
SDVOSB (Pending Certification)

Contact: Joseph@ResilientMindAI.com
Technical Information: AriaOS.dev

Executive Summary

AriaOS is a failure-first execution and governance layer designed to preserve correct behavior when connectivity, cloud services, or upstream dependencies fail. This validation tested whether the system maintains governance, auditability, and deterministic state transitions under sustained failure injection.

Key Architectural Principle:

- Offline operation is the default assumption
- Cloud connectivity is optional, not required
- All decisions remain bounded by local policy and state machines
- Autonomous actions halt if audit integrity is compromised

Test Summary:

- **Duration:** 8+ consecutive days in chaos mode
- **Platform:** HP G8 laptop (5-year-old commodity hardware, Ubuntu 24)
- **Failure Injection:** Full network isolation, random process restarts, clock skew, storage pressure, dependency timeouts, identity provider unavailability, control plane loss, partial telemetry, event bus schema degradation
- **Observed Outcome:** System maintained operational continuity. No unrecoverable states entered. All degradations followed defined NORMAL → DEGRADED → RECOVERY → OFFLINE progression
- **Audit Integrity:** Hash-chained ledger remained intact. Post-test verification confirmed no tampering or data loss
- **Governance Enforcement:** HITL (Human-In-The-Loop) constraints remained active offline. No autonomous actions executed without approval

This is behavioral validation under failure, not performance benchmarking. The goal: Can the system explain what happened when things went wrong?

Purpose of This Validation

Traditional AI systems assume reliable infrastructure. When those assumptions break—network fails, cloud services are unreachable, identity providers are down—systems enter undefined states or fail catastrophically.

This validation tests whether AriaOS maintains governed, explainable operation when normal operating assumptions collapse. The question is not throughput or latency. The question is: when the network is gone and the control plane is unreachable, does the system degrade gracefully? Can it prove what it did?

For energy infrastructure, healthcare operations, public safety systems, and industrial control environments, this is not theoretical. Network failures, cloud outages, and air-gapped operation are operational reality.

Test Environment Overview

This testing was performed without vendor-provided hardware, cloud services, or external orchestration systems.

Hardware Configuration

Component	Specification
Device	HP G8 laptop (5-year-old class commodity hardware)
Rationale	Validates operation on aging, consumer-grade hardware without specialized accelerators
Deployment	Single-node installation (distributed mesh not tested)

Software Environment

Layer	Configuration
Operating System	Ubuntu 24 (Linux x86_64)
Network	Intermittent connectivity with extended full-isolation periods (hours to days)
Cloud Dependencies	None required for core operation
External Identity	Not required during operation (local authentication sufficient)
Execution Mode	Chaos testing enabled with continuous failure injection

Test Duration and Conditions

- **Total Runtime:** 8+ continuous days
- **Network Conditions:** Full network loss, intermittent reconnection, randomized failure injection
- **Failure Injection:** Continuous and randomized across all system layers
- **Recovery Testing:** Automatic recovery workflows exercised under failure conditions

What This Test Does NOT Validate

- Multi-node mesh coordination or distributed consensus
- High-availability clustering across multiple hosts
- Performance benchmarks (throughput, latency, requests/second)
- Security hardening or penetration resistance
- Production-scale deployments or load testing

Chaos & Failure Injection Methodology

Chaos testing injects random failures into running systems to validate resilience assumptions. Unlike controlled failure testing with scripted scenarios, chaos mode introduces unpredictable failure combinations.

All failures were synthetic but structured. Failure scenarios were predefined and repeatable. Outcomes were evaluated based on state correctness, not uptime alone.

Failure Injection Layers

Network Layer Failures

- **Full Network Loss:** Complete disconnection for extended periods (hours to days)
- **DNS Failure:** Name resolution unavailable while routes remain
- **Network Partition:** Partial connectivity with routing failures
- **Dependency Unavailability:** Upstream services reachable but unresponsive

Process and Service Failures

- **Randomized Process Failure:** Critical subsystems terminated during active operations
- **Service Restarts:** Forced restart of components under load
- **Agent Crashes:** Heartbeat loss triggering dead agent detection
- **Control Plane Loss:** Orchestration and supervision layers made unavailable

Infrastructure Failures

- **Clock Skew:** System time drift without NTP synchronization
- **Storage Pressure:** Disk space exhaustion and I/O contention
- **Storage Degradation:** Intermittent read/write failures
- **Configuration Drift:** Unexpected configuration state changes

Application Layer Failures

- **Event Bus Schema Degradation:** Malformed events with missing required fields
- **Identity Provider Unavailability:** Authentication services offline
- **Partial Telemetry:** Incomplete observability and monitoring data
- **Certificate Failures:** TLS handshake failures and certificate expiry

> **CONTINUOUS FAILURE INJECTION**
>
> Failures were injected randomly and continuously throughout the 8+ day period. Recovery mechanisms were not given "clean" periods to stabilize. This tests whether recovery logic exhausts resources, whether degraded states accumulate into cascading failures, and whether audit logs corrupt under sustained chaos.

Preset Failure Scenario Catalog

Generic chaos testing (kill processes, drop packets) validates basic resilience. Industry-specific failure presets validate whether the system handles domain-realistic operational failure modes.

Each preset encodes:

- **Industry Context:** Which operational domain this failure affects
- **Failure Scenario:** Specific technical failure (network isolation, ledger desync, identity outage)
- **Execution Posture:** How the system should respond (OFFLINE_DETERMINISTIC, AUDIT_HEAVY, DEGRADED_CONSTRAINED)

This catalog is extensible and intended to be adapted by third parties.

Preset Categories and Representative Scenarios

Category	Example Scenario	Industry	Execution Posture
Legacy	Healthcare EHR System Outage	Healthcare	OFFLINE_DETERMINISTIC
	Energy Grid Islanding Event	Energy	OFFLINE_DETERMINISTIC
	Finance Trading Platform Halt	Finance	OFFLINE_DETERMINISTIC
Security & Core IT	Identity Provider Outage (SSO/IAM Failure)	Enterprise IT	OFFLINE_DETERMINISTIC
	SIEM Blindness Event (Log Ingestion Loss)	Security Ops	OFFLINE_DETERMINISTIC
	Zero Trust Policy Engine Offline	Security Ops	OFFLINE_DETERMINISTIC
Infrastructure & IT Ops	Data Center Network Partition	Enterprise IT	OFFLINE_DETERMINISTIC
	VM Orchestration Control Plane Loss	Enterprise IT	OFFLINE_DETERMINISTIC
Cloud & Hybrid	Cloud Control Plane Unreachable	Cloud Ops	OFFLINE_DETERMINISTIC
	Secrets Manager Lockout	Cloud Ops	OFFLINE_DETERMINISTIC
Enterprise Platforms	ERP Transaction Ledger Desynchronization	Enterprise	OFFLINE_DETERMINISTIC
	HRIS Payroll Execution Lock	Enterprise	OFFLINE_DETERMINISTIC
Public Sector & Regulated	Classified Network Air Gap Event	Government	AUDIT_HEAVY
	Continuity of Operations (COOP) Trigger	Government	DEGRADED_CONSTRAINED
	Public Safety CAD System Isolation	Public Safety	OFFLINE_DETERMINISTIC
OT Security Adjacent	IT-OT Boundary Collapse	Industrial	OFFLINE_DETERMINISTIC
	Safety Interlock Override Loss	Industrial	OFFLINE_DETERMINISTIC
	Historian Data Gap	Industrial	OFFLINE_DETERMINISTIC

Total Preset Catalog: 31 scenarios across 7 categories (Healthcare, Energy, Finance, Public Sector, Industrial/OT, Security Operations, Enterprise IT)

> **WHY PRESET-DRIVEN TESTING MATTERS**
>
> Generic process crashes prove basic resilience. Industry presets prove the system handles operational realities: EHR outages during patient surge, grid islanding during storms, CAD isolation during 911 volume spikes, SIEM blindness during security incidents. This is the difference between "it recovered from a crash" and "it maintained governance during a COOP event."

System State & Health Results

Observed Runtime Characteristics

Metric	Observed Value
Total Runtime	8+ consecutive days under chaos conditions
Core System Health	Maintained operational state throughout test period
Sustained Degraded States	Zero (degraded states triggered recovery, did not persist)
Unrecoverable Failures	Zero (no manual intervention required)
Recovery Loop Failures	Zero (recovery workflows completed or timed out gracefully)
Audit Ledger Integrity	Verified intact post-test (hash chain unbroken)

State Transition Logic

The system implements threshold-based state transitions:

- **NORMAL:** Baseline operational state
- **DEGRADED:** Triggered when error count exceeds 5 in 60-second window OR 10+ consecutive warnings
- **RECOVERY:** Active recovery workflows engaged (5-minute timeout before OFFLINE)
- **OFFLINE:** Subsystem non-functional after recovery timeout

Critical Design Decision: Warnings do not automatically degrade state. Only error patterns exceeding thresholds trigger transitions. This prevents alert fatigue and false escalations.

Observed Degradation and Recovery Patterns

Multiple agent crashes occurred during testing (tool, memory, reasoning agents). Observed sequence:

1. Agent heartbeat degraded at 15-18 seconds (WARNING level logged)

2. Agent marked DEAD at 30+ seconds if heartbeat not restored (ERROR level)
3. Dead letter queue entry created (operation not silently dropped)
4. Recovery workflow triggered automatically (where configured)
5. State transition logged to audit ledger with timestamp and reason

Event bus schema violations (missing `session_id` fields) were handled by auto-generating identifiers and logging schema warnings rather than dropping events. Zero data loss from schema degradation.

> **KEY FINDING**
>
> All state transitions followed defined patterns. No undefined behavior observed. Governance remained enforceable during network isolation. Audit trail integrity verified post-test. The system explained what happened.

Governance & Determinism Model

Explicit State Machines

AriaOS does not rely on implicit behavior or emergent properties. State transitions are explicit, threshold-based, and auditable:

- Thresholds defined: error counts, time windows, consecutive warnings
- Transitions logged: every NORMAL → DEGRADED transition recorded with reason
- Recovery timeouts enforced: RECOVERY → OFFLINE after 5-minute timeout
- No silent degradation: all state changes emit audit events

Bounded Execution Authority

The system operates under explicit authority bounds:

- **Default Mode:** HITL (Human-In-The-Loop) - no autonomous actions without approval
- **Partial Autonomy:** Requires profile enablement + policy definition + risk assessment
- **Full Autonomy:** Disabled system-wide (not permitted)

Intent → Decision → Execution Separation

The system maintains explicit separation between intent, approval, and execution:

```
1. INTENT GENERATION
   ↓
   System declares: "I want to perform action X on subsystem Y"
   Intent ID generated, risk level assessed

2. DECISION EVALUATION
   ↓
   Policy engine validates against active governance rules
   Checks: Is autonomy permitted? Is action in allowed scope?

3. APPROVAL
```

```
   ↓
   HITL Mode: Human must approve
   Partial Autonomy: Policy approves if within scope, else human approval
   Full Autonomy: Blocked

4. EXECUTION
   ↓
   Action executes with cryptographic receipt generation
   Receipt hash: SHA-256(intent_id, action, approver, timestamp, policy_id)
```

This separation enables post-incident review to answer: What did the system want to do? What was approved? What actually executed?

Cloud as Strategist, Node as Executor

When cloud connectivity is available:

- **Cloud Role:** Strategy, optimization, long-term planning (when reachable)
- **Node Role:** Execution, governance enforcement, audit logging (always accountable)

When cloud connectivity is lost:

- Node continues governed execution with local policy
- Audit events queue locally and sync when connectivity restores
- HITL workflows function offline (local approval mechanisms)
- State transitions remain deterministic without external coordination

The node never becomes "unbound" when cloud is unavailable. Governance constraints tighten under isolation (revert to HITL if partial autonomy relies on cloud-based policy updates).

Preset Failure Scenarios: Industry Coverage

Healthcare

Scenario: EHR System Outage During Peak Hours

Cloud-based electronic health record system loses connectivity during patient surge. Clinical staff require offline access to records and decision support.

Expected Behavior: Local governance maintains HITL for clinical suggestions. Audit trail preserved for HIPAA compliance review.

Energy

Scenario: Grid Islanding Event

Power grid control system forced into islanded operation due to upstream grid failures. System must maintain local control without external coordination.

Expected Behavior: Offline-first execution with local policy enforcement. All control actions audited for NERC CIP compliance.

Public Sector

Scenario: Continuity of Operations (COOP) Trigger

Government facility activates COOP plan, relocating operations to alternate site with degraded connectivity and limited infrastructure.

Expected Behavior: DEGRADED_CONSTRAINED execution posture. Enhanced audit logging. HITL enforced. Mission-critical functions continue with reduced capability.

Public Safety

Scenario: CAD System Isolation

Computer-aided dispatch system isolated from network during 911 call surge. Dispatch operations must continue with local data.

Expected Behavior: Offline operation with local database. No cloud-dependent features. All dispatch decisions logged for post-incident review.

Industrial / OT-Adjacent

Scenario: IT-OT Boundary Collapse

Network segmentation failure exposing industrial control systems to IT network. Safety systems must isolate or fail-safe.

Expected Behavior: Isolation triggers. Safety interlock verification. No control actions without explicit safety confirmation. Audit-heavy logging for incident investigation.

Scenario: Safety Interlock Override Loss

Safety interlock systems unable to communicate override status to control logic.

Expected Behavior: Fail-safe behavior. Control actions blocked until interlock status confirmed. Manual override required with audit trail.

Security Operations

Scenario: SIEM Blindness Event

Security information and event management system blind due to log ingestion pipeline failure.

Expected Behavior: Security operations continue with degraded visibility. Local logging preserved. Event replay enabled when SIEM restores.

Enterprise IT

Scenario: Control Plane Loss (VM Orchestration Unavailable)

Virtualization management control plane unavailable. Existing VMs continue but no new provisioning or reconfiguration possible.

Expected Behavior: Graceful degradation. Existing operations continue. New operations queue or defer until control plane restores.

Governance Enforcement Under Failure

Human-In-The-Loop as Default

AriaOS enforces HITL by architectural default. Autonomous execution is not permitted unless:

1. Operational profile explicitly enables autonomy
2. Active policy defines allowed action scope with risk thresholds
3. Audit ledger integrity verified (blocks autonomy if chain broken)

During chaos testing, HITL mode remained active. No autonomous actions executed. All decisions required explicit approval, including during network isolation periods.

Profile-Gated Autonomy

Autonomy levels (L0-L4):

- **L0 (Manual):** No autonomous actions - pure tool mode
- **L1 (Assisted):** Suggestions only, human executes
- **L2 (Supervised):** Autonomous execution with human oversight
- **L3 (Delegated):** Autonomous within mission parameters
- **L4 (Mission-Grade):** Full autonomous mission execution (requires explicit whitelisting)

Chaos test ran at L1 (Assisted). System generated suggestions but did not execute autonomously.

No Autonomous Behavior Without Declared Intent

If autonomy were enabled (not tested), every action would require:

1. **Intent Declaration:** System declares `intent_id`, action, subsystem, risk_level
2. **Policy Check:** Governance engine validates against active policy
3. **Approval:** Human (HITL) or policy (if risk below threshold)
4. **Receipt Generation:** SHA-256 hash of decision record created

5. **Audit Logging:** Intent, approval, execution all logged separately

If audit ledger integrity fails (hash chain broken), system immediately:

- Transitions to DEGRADED state
- Reverts to HITL mode (blocks autonomous actions)
- Returns 423 Locked for configuration changes
- Requires manual integrity verification before resuming

> **GOVERNANCE UNDER NETWORK ISOLATION**
>
> During extended network loss periods, governance layer continued functioning with local policy enforcement, offline HITL workflows, and queued audit events. No governance bypass occurred when cloud was unreachable.

Audit-First Failure Handling

Hash-Chained Event Ledger

Every system action generates an immutable audit event. Events are cryptographically chained:

```
Event[N]:
  timestamp: 2025-12-26T16:47:08.880
  subsystem: "recovery"
  level: "ERROR"
  message: "Agent reasoning marked DEAD"
  prev_hash: sha256(Event[N-1])
  event_hash: sha256(timestamp + subsystem + level + message + prev_hash)

Event[N+1]:
  prev_hash: event_hash(Event[N])  ← Chains to previous event
```

Modifications to historical events break the chain. Verification recomputes expected hashes and detects mismatches immediately.

Ledger Verification Strategy

- **Boot-Time Verification:** Full chain verification before system becomes operational
- **Continuous Verification:** Periodic integrity checks during operation
- **On-Demand Verification:** API endpoint for manual verification requests

During chaos testing, continuous verification ran throughout the 8+ day period. No integrity failures detected.

Event Categorization for Noise Reduction

The ledger distinguishes:

- **INFO:** Operational events, no state change
- **WARNING:** Potential issues, tracked but do not trigger state transitions

- **STATE_TRANSITION:** Explicit state changes (NORMAL → DEGRADED, etc.)
- **ERROR:** Failures that may trigger state transitions if thresholds exceeded

Operators can filter to STATE_TRANSITION events only and see the exact degradation sequence without noise from routine warnings.

Post-test analysis: Thousands of WARNING events logged (agent degraded, DLQ items, schema warnings). Only a subset triggered ERROR-level events. Only ERROR patterns exceeding thresholds triggered state transitions. This is threshold-based governance, not reactionary.

Reproducibility & Third-Party Validation

This work is published to enable independent replication and comparison across environments.

Hardware Agnostic Testing Approach

The testing approach does not require:

- Specialized accelerators or GPUs
- Vendor-provided hardware or cloud credits
- Proprietary infrastructure or tooling
- External orchestration systems

Any vendor, research lab, or third party can replicate these tests on commodity hardware running Linux. Preset failure scenarios are portable across environments.

Replication Procedure

1. Install AriaOS on Ubuntu 24 or compatible Linux distribution
2. Enable chaos mode in kernel configuration (`chaos_enabled: true`)
3. Load preset failure scenario catalog (31 scenarios across 7 categories)
4. Configure failure injection parameters (process restart probability, network failure frequency)
5. Enable full audit logging with hash-chained ledger
6. Start system and run for minimum 8-day period
7. Monitor state transition logs for proper DEGRADED → DEAD progression
8. Verify ledger integrity post-test (hash chain validation)
9. Confirm HITL constraints remained active throughout test
10. Review audit logs for autonomous action attempts (should be zero without approval)

Comparison Basis for Third-Party Validation

Results should be compared based on:

- **Behavior Under Failure:** Did the system maintain governance? Were state transitions explainable?
- **Audit Trail Completeness:** Can every decision be traced? Is the ledger tamper-evident?
- **Offline Continuity:** Did governance work without network? Did audit logging continue?
- **Threshold Correctness:** Were warnings distinguished from errors? Did only patterns trigger transitions?

Do not compare on: uptime percentage, latency, throughput, or resource utilization. Those are performance metrics. This is behavioral validation.

> **INVITATION FOR INDEPENDENT VALIDATION**
>
> Any organization can replicate this testing methodology with their own systems. Preset scenarios are published. Chaos injection parameters are documented. The goal is comparable behavioral validation across platforms, not vendor competition.

Limitations & Next Validation Steps

What This Validation Demonstrates

- Offline-first operation maintained governance during network isolation
- Threshold-based state transitions prevented false escalations
- Hash-chained audit ledger remained intact under sustained chaos
- HITL constraints enforced throughout test, including offline periods
- Event bus schema violations handled without data loss
- Agent crashes followed defined degradation patterns (WARNING → ERROR → DLQ)
- 8+ day runtime on commodity hardware without resource exhaustion

Explicit Limitations

WHAT THIS DOES NOT PROVE

- **Not Production-Ready Proof:** Synthetic failures are not real-world incidents. Chaos testing does not imply production readiness
- **Not a Certification:** This is not formal certification, accreditation, or compliance validation from any standards body or regulatory authority
- **Not a Performance Benchmark:** No claims about throughput, latency, or performance relative to other systems
- **Not Security Accreditation:** Testing focused on resilience and governance, not security hardening or penetration resistance
- **Not Multi-Node Validation:** Single-node test does not validate distributed consensus, partition tolerance, or mesh coordination
- **Not Hardware Qualification:** Testing on one device does not qualify the system for all hardware configurations
- **Not Domain Compliance Proof:** Does not prove compliance with HIPAA, NERC CIP, FedRAMP, or other regulatory frameworks

Ongoing and Future Validation

Hardware Diversity Testing:

- Edge accelerators (Jetson-class or similar SoC platforms)
- Ruggedized industrial hardware
- Resource-constrained embedded devices
- Multi-node mesh configurations

Domain-Specific Compliance:

- Energy: NERC CIP audit mapping, FERC compliance requirements
- Healthcare: HIPAA controls audit, FDA medical device validation
- Finance: SOX audit trail validation, regulatory reporting
- Public Sector: FedRAMP control mapping, FISMA compliance

Security Hardening:

- Penetration testing and vulnerability assessment
- Cryptographic strength validation
- Access control enforcement testing
- Supply chain security review

Implications for Regulated and Critical Environments

Energy Infrastructure

Grid control, distributed energy resources, and substation automation require offline-first operation. Cloud connectivity cannot be assumed during storms, cyber events, or grid disturbances.

Relevance of Observed Behavior:

- Local governance without external coordination validated
- Grid islanding scenario tested with offline decision-making
- Audit trails maintained for NERC CIP / FERC review
- State transitions explainable for post-incident analysis

Healthcare Operations

EHR systems, medical devices, clinical decision support must function during network outages and cloud disruptions.

Relevance of Observed Behavior:

- HITL enforced (no silent AI clinical suggestions)
- Offline operation validated during EHR connectivity loss
- Audit trail supports HIPAA / FDA compliance review
- Deterministic failure modes enable clinical risk assessment

Public Safety and Emergency Services

Dispatch, emergency communications, and incident management systems must handle COOP events and air-gapped operation.

Relevance of Observed Behavior:

- Air-gapped execution validated (no external dependencies)
- COOP scenario tested (continuity without cloud)
- Audit-heavy logging maintained (all actions verifiable)

- Evidence integrity preserved (hash chain prevents tampering)

Industrial and OT-Adjacent

Manufacturing, process control, and safety systems require fail-safe behavior and explainable degradation.

Relevance of Observed Behavior:

- IT/OT boundary failures handled with isolation
- Safety interlock scenarios validated (fail-safe behavior)
- Historian data gaps logged (not silently ignored)
- Sensor blind spots triggered appropriate degradation

> **FOUNDATION FOR COMPLIANCE VALIDATION**
>
> Chaos testing does not prove compliance. It demonstrates that the architecture can support compliance requirements: audit trails are tamper-evident, governance is enforceable offline, decisions are traceable, and state transitions are explainable. Domain-specific compliance mapping is a separate validation phase.

Appendix A: Glossary

Offline-First:

Architecture where systems operate without network connectivity as the default assumption. Cloud services are optional enhancements. Core functionality must work air-gapped.

Governance Layer:

Policy enforcement system that validates actions against defined rules. Ensures autonomous systems cannot execute outside approved scope. Functions locally without external coordination.

Chaos Testing:

Deliberate injection of random failures to validate resilience assumptions. Differs from scripted testing by introducing unpredictable failure combinations.

HITL (Human-In-The-Loop):

Operating mode where all autonomous actions require explicit human approval. Default mode in AriaOS. Partial autonomy requires profile enablement and policy definition.

Tamper-Evident Ledger:

Hash-chained event log where each entry includes the hash of the previous entry. Modifications to historical events break the chain and are detectable during verification.

State Transition:

Explicit change in system operational state (NORMAL → DEGRADED → RECOVERY → OFFLINE). Transitions require threshold evaluation. Single warnings do not trigger transitions.

Dead Letter Queue (DLQ):

Holding area for operations that could not complete. Enables retry workflows. Prevents silent failures.

Receipt Hash:

SHA-256 cryptographic hash of a decision record (action, approver, policy, timestamp). Provides unforgeable proof of authorization.

Preset Failure Scenario:

Industry-specific failure pattern (EHR outage, grid islanding, CAD isolation, SIEM blindness) encoded as test configuration. Enables realistic failure modeling.

Execution Posture:

How the system should respond to specific failure scenarios. Examples: OFFLINE_DETERMINISTIC (maintain determinism without connectivity), AUDIT_HEAVY (enhanced logging for regulated environments), DEGRADED_CONSTRAINED (reduced capability with strict bounds).

Air-Gapped Execution:

Zero external connectivity. No internet, no cloud, no external services. Governance and audit must function entirely locally.

Intent Declaration:

Explicit statement of what the system wants to do before action is taken. Includes intent_id, action, subsystem, risk_level. Enables separation of intent from execution.

Appendix B: Test Configuration

Failure Types Exercised

Layer	Failure Modes
Network	Full isolation, DNS failure, dependency timeouts, network partitions, failover without DNS
Process/Service	Random termination, agent crashes, heartbeat loss, control plane unavailable, service degradation
Infrastructure	Clock skew, storage pressure, configuration drift, storage degradation, backup failures
Application	Event schema violations, identity provider outages, certificate failures, partial telemetry, secrets lockout
Security	SIEM blindness, EDR visibility loss, zero trust policy engine offline, CA expiry

Preset Scenario Summary

Category	Scenario Count	Industries Covered
Legacy	5	Healthcare, Energy, Finance, Manufacturing, Logistics
Security & Core IT	6	Enterprise IT, Security Operations
Infrastructure & IT Ops	6	Enterprise IT, Data Center Operations
Cloud & Hybrid	5	Cloud Operations, Hybrid Infrastructure
Enterprise Platforms	5	ERP, CRM, HRIS, BI, MDM
Public Sector & Regulated	5	Government, Public Safety, Elections, Courts
OT Security Adjacent	4	Industrial Control, Manufacturing

Total: 31 preset failure scenarios across 7 categories

All scenarios are synthetic but operationally realistic. They encode failure patterns observed in real deployments but executed in controlled test environment.

Conclusion

Eight-plus days of chaos testing on Ubuntu 24 demonstrated that AriaOS maintains governed, auditable, deterministic operation under sustained failure injection on commodity hardware.

The system operated without vendor-provided hardware, cloud services, or external orchestration. Governance constraints remained active during network isolation. Audit trails remained intact. State transitions remained explainable.

What Was Validated

- Offline-first execution posture (governance works without connectivity)
- Threshold-based state transitions (warnings don't trigger escalations)
- Audit-first failure handling (all transitions logged and hash-chained)
- HITL enforcement under failure (no autonomous bypass when cloud unavailable)
- Deterministic degradation (no undefined behavior observed)

What Remains to Be Validated

- Multi-node mesh coordination under network partitions
- Edge accelerator platforms (GPU/SoC deployment targets)
- Domain-specific regulatory compliance (NERC CIP, HIPAA, FedRAMP)
- Security hardening and penetration testing
- Production-scale load and performance characteristics

Applicability to Critical Environments

For energy infrastructure, healthcare operations, public safety systems, and industrial control: offline-first governance, tamper-evident audit trails, and HITL enforcement provide a foundation for deployment in failure-prone environments.

This validation does not replace domain-specific compliance validation, security review, or production readiness assessment. It demonstrates that the architecture can support those requirements.

CORE THESIS VALIDATED

AriaOS preserves correct, governed, explainable behavior when connectivity, cloud services, and upstream dependencies fail. The system does not require external coordination to maintain governance. Audit trails remain tamper-evident under chaos. This is foundational for deployment where "it worked in the cloud lab" is insufficient.

This document presents observed test results from an 8+ day chaos validation.
It makes no claims beyond what was directly measured and logged during the test period.
Results are published to enable independent replication and third-party validation.