The Core Operational Question

“Can You Prove What Is Running?”

Meta-Governance
Featuring Argus Runtime Assurance
Proof. Not Promises.

Executive Summary

Modern cybersecurity, governance, and compliance programs are built around a foundational assumption:

> Organizations know what is operating inside their environments.

Increasingly, this assumption is false.

Modern software-defined systems are:

- continuously changing,
- dependency-driven,
- cloud-connected,
- dynamically orchestrated,
- and increasingly autonomous.

Containers rebuild automatically.
Dependencies mutate silently.
AI services update dynamically.
Infrastructure changes continuously.
Runtime environments drift operationally.

Yet many organizations still govern these systems using:

- spreadsheets,
- static inventories,
- periodic scans,

- manual attestations,
- and point-in-time audits.

This creates a dangerous operational gap between:

- intended systems,
  and:
- actual running systems.

The most important operational question in modern governance is no longer:

> “Was the system approved?”

The real question is:

“Can you prove what is running right now?”

This paper explores why:

- runtime truth,
- operational visibility,
- and continuously verifiable configuration state

have become foundational requirements for modern assurance.

The paper introduces:

- Configuration-First Governance,
- continuous runtime validation,
- cryptographic provenance,
- and the role of the Meta-Governance runtime assurance platform component:

Argus

Argus operationalizes runtime assurance by continuously collecting:

- runtime snapshots,
- executing process evidence,

- dependency observations,
- operational fingerprints,
- and cryptographically verifiable runtime state.

The future of software assurance depends not on trusting deployed systems.

It depends on continuously proving operational reality.

The Runtime Assurance Crisis

Historically, organizations governed systems that:

- changed slowly,
- deployed infrequently,
- operated on stable infrastructure,
- and remained relatively static over time.

Modern systems no longer behave this way.

Today's environments include:

- CI/CD pipelines,
- ephemeral infrastructure,
- dynamic orchestration,
- open-source dependency ecosystems,
- AI-enabled systems,
- distributed APIs,
- and autonomous operational workflows.

Runtime environments continuously evolve.

Yet governance processes often remain:

- document-centric,
- audit-centric,
- and operationally disconnected.

This creates a growing inability to answer a deceptively simple question:

> "What is actually running?"

The Runtime Reality Problem

Most organizations possess:

- architecture diagrams,
- approved baselines,
- vulnerability reports,
- SBOMs,
- compliance packages,
- and signed deployments.

But these artifacts often describe:

- intended state,
  rather than:
- operational state.

Modern runtime environments drift continuously through:

- dependency updates,
- orchestration changes,
- injected libraries,
- sidecar services,
- hot patches,
- container rebuilds,
- infrastructure mutation,
- and unauthorized operational changes.

Without runtime validation:
organizations govern assumptions rather than reality.

The Core Operational Question

Modern assurance ultimately reduces to one critical question:

“Can you prove what is running?”

Not:

- what was deployed,
- what was approved,
- or what the architecture diagram claims exists.

But:

- what is operationally executing,
- right now,
- across actual runtime infrastructure.

This includes:

- running processes,
- loaded libraries,
- active dependencies,
- runtime containers,
- orchestration state,
- infrastructure relationships,
- and live execution behavior.

Without this capability:
governance becomes:

- retrospective,
- assumption-based,
- and operationally incomplete.

Introducing Argus Runtime Assurance

Argus is the runtime operational assurance component within the Meta-Governance ecosystem.

Its purpose is straightforward:

Continuously observe and prove operational runtime reality.

Argus bridges the assurance gap between:

- intended configuration,
  and:
- actual execution.

Rather than relying exclusively on:

- deployment artifacts,
- SBOM exports,
- or CI/CD metadata,

Argus validates:

- live runtime conditions.

What Argus Collects

Argus captures runtime operational evidence including:

- executing processes,
- loaded binaries,
- active services,
- dependency observations,
- runtime libraries,
- host-level telemetry,
- container/process relationships,
- and execution fingerprints.

This creates:

- continuously refreshable runtime evidence snapshots.

These snapshots become:

- operational truth anchors.

Runtime Snapshots as Evidence

Traditional governance often treats runtime as:

- transient,
- difficult to capture,
- or operationally opaque.

Argus changes this model.

Argus creates:

- timestamped runtime snapshots,
- operational fingerprints,
- and cryptographically anchored evidence objects.

These snapshots can be:

- compared,
- validated,
- diffed,
- scored,
- and historically preserved.

This transforms runtime execution itself into:

- continuously measurable evidence.

Configuration-First Governance

Configuration-First Governance treats:

- configuration state
  as:

- the primary assurance object.

Not PDFs.
Not spreadsheets.
Not architecture diagrams.

Actual runtime configuration determines:

- what software executes,
- what dependencies are active,
- what infrastructure communicates,
- and what risks are operationally present.

Argus operationalizes this philosophy by continuously validating:

- operational runtime state.

Runtime Drift Detection

Drift is inevitable.

Modern systems continuously evolve through:

- orchestration behavior,
- infrastructure mutation,
- dynamic dependency resolution,
- hot patches,
- and operational scaling.

Argus enables runtime drift detection by comparing:

- approved baselines,
  against:
- actual runtime observations.

This includes:

- unexpected processes,
- unauthorized binaries,
- dependency divergence,

- runtime library mismatches,
- and operational anomalies.

This transforms governance from:

- static approval,
  to:
- continuously measurable operational integrity.

Runtime SBOM Validation

SBOMs provide critical visibility into software composition.

Standards such as:

- CycloneDX
- and Software Package Data Exchange

help establish:

- dependency inventories,
- supplier visibility,
- and software composition baselines.

However:
static SBOMs alone do not prove runtime integrity.

Argus extends SBOM assurance into runtime operations by enabling comparison between:

- approved SBOM state,
  and:
- observed runtime execution.

This creates:

- operational SBOM validation.

Cryptographic Provenance

Runtime assurance requires:

- verifiable lineage,
- tamper-evident evidence,
- and provable integrity.

Argus integrates into Meta-Governance provenance systems by:

- attaching runtime snapshots to evidence timelines,
- cryptographically hashing observations,
- and preserving immutable operational history.

This enables:

- historical runtime validation,
- drift chronology,
- and evidence continuity over time.

Continuous Evidence Generation

Traditional governance often depends on:

- manually assembled evidence,
- exported reports,
- and audit documentation.

Argus enables:

- continuously generated runtime evidence.

Evidence becomes:

- machine-derived,
- operationally attached,
- continuously refreshed,
- and cryptographically verifiable.

This shifts governance from:

- administrative reporting,
  to:
- operational assurance.

---

cATO and Runtime Assurance

Continuous Authorization to Operate (cATO) initiatives increasingly recognize that:

- deployment approval alone is insufficient.

True cATO requires:

- continuously measurable operational integrity.

Argus supports cATO by enabling agencies to:

- continuously validate runtime execution,
- detect drift,
- generate live evidence,
- and maintain continuously refreshed operational trust.

Without runtime validation:
continuous authorization becomes:

- accelerated paperwork,
  rather than:
- operational assurance.

Air-Gapped and Sovereign Operations

Argus is particularly valuable in:

- aerospace,
- defense,

- tactical edge,
- industrial control systems,
- and sovereign operational environments.

These systems often require:

- disconnected operation,
- local evidence generation,
- cryptographic independence,
- and operational survivability.

Argus supports:

- local-first runtime assurance,
  allowing organizations to continuously validate operational state:
- even in disconnected or air-gapped environments.

Executive Visibility

Executives increasingly require answers to operational questions such as:

- What is actually running?
- What changed?
- What drifted?
- What dependencies are operationally active?
- Which systems remain trustworthy?

Traditional governance rarely provides continuous answers.

Argus enables:

- live operational visibility,
- continuously measurable trust,
- and evidence-backed operational awareness.

The Strategic Shift

The software industry is undergoing a profound transition:

From:

- deployment-centric assurance,

To:

- runtime-centric assurance.

Organizations can no longer assume:

- approved systems remain trustworthy,
- deployed systems remain unchanged,
- or signed systems remain operationally compliant.

Operational truth must become:

- continuously observable,
- continuously measurable,
- and continuously provable.

Argus operationalizes this capability.

Conclusion

Modern software-defined systems evolve too rapidly for:

- static inventories,
- periodic audits,
- and assumption-based governance.

The central operational question of modern assurance is no longer:

> “Was the system approved?”

It is:

> “Can you continuously prove what is running?”

Configuration-First Governance provides a path toward answering this question through:

- runtime validation,
- continuous evidence generation,
- SBOM intelligence,
- operational drift detection,
- and cryptographic provenance.

Argus extends this model directly into runtime operations by continuously observing:

- actual execution,
- actual dependencies,
- and actual operational state.

The future of software assurance depends not on trusting deployed systems.

It depends on continuously proving operational reality.

Meta-Governance

Argus Runtime Assurance

Proof. Not Promises.