Provenance Is Not Entropy (White Paper)

A Governance Perspective on Trust in Software-Dominant Systems

---

Executive Summary

Modern software governance relies heavily on precise, verifiable artifacts such as cryptographic hashes, digital signatures, build manifests, and point-in-time Software Bills of Materials (SBOMs). These mechanisms are essential for validating integrity at a specific moment and remain foundational to secure system development.

However, integrity at a moment in time is not equivalent to provenance.

As software systems become increasingly dynamic, interconnected, and long-lived, stakeholders are frequently asked to explain not only *what* a system is, but *how it came to be*. This paper clarifies the distinction between snapshot-based verification and provenance as a historical, contextual record of evolution.

The central position is simple:

> **Provenance is not randomness, entropy, or snapshot integrity.**
> **Provenance is recorded evolution over time.**

This paper articulates that position in a manner applicable across software governance, assurance, audit, and lifecycle management contexts, without reliance on any single standard or framework.

Contact: Jeff Heisler jrheisler7@gmail.com

Non-Normative Positioning and Scope

This document is **informational and interpretive in nature**. It does not propose new requirements, modify existing standards, or prescribe specific implementation approaches.

The purpose of this paper is to clarify terminology and concepts commonly used in software governance, assurance, and lifecycle management, with particular attention to the distinction between:

- Snapshot-based integrity verification, and
- Provenance as a historical record of system evolution.

The perspectives presented here are intended to be **technology-agnostic and framework-neutral**, and may be applied in conjunction with a wide range of existing standards, policies, and organizational practices.

Nothing in this document should be interpreted as:

- A critique of any specific standard or governing body
- A proposal to revise normative language
- A replacement for established configuration, assurance, or audit processes

Instead, this paper seeks to:

- Articulate a conceptual model for understanding provenance in software-dominant systems
- Support clearer communication across engineering, governance, audit, and security disciplines
- Provide a common vocabulary for discussing lifecycle evidence, traceability, and accountability

Any references to practices such as hashing, signatures, SBOMs, or integrity checks are descriptive, not evaluative. These mechanisms remain essential and are assumed to be part of a well-governed software environment.

This document should be read as a **position paper**, offered to stimulate discussion and shared understanding, rather than as a normative specification.

---

1. Context: Software as Infrastructure

Software now functions as infrastructure across safety-critical, regulated, and mission-dependent domains. These systems are characterized by:

- Continuous change rather than discrete releases
- Automated build and integration pipelines
- Deep dependency trees and transitive inheritance
- Long operational lifespans with evolving threat exposure

In this environment, governance depends not only on correctness, but on **explainability**.

When incidents occur — whether failures, vulnerabilities, or audits — the questions that matter are rarely limited to integrity alone.

---

2. Clarifying Key Terms

2.1 Randomness and Entropy (Engineering Tools)

Randomness and entropy are widely and correctly used in engineering practice, including:

- Fuzz testing and chaos engineering
- Randomized inputs for robustness testing
- Cryptographic key material and nonces

These techniques are essential for discovering unknown failure modes and strengthening security properties.

This paper does **not** challenge the value of randomness in engineering or security.

Instead, it distinguishes between **discovery mechanisms** and **governance evidence**.

2.2 Snapshot Integrity Artifacts

Common governance artifacts include:

- Cryptographic hashes and signatures
- Build manifests and release descriptors
- SBOMs generated at a point in time
- Compliance or verification reports

These artifacts answer an important question:

> *“Is this artifact identical to what was previously recorded or approved?”*

They are:

- Precise
- Verifiable
- Reproducible

They are also, by design, **memoryless**.

2.3 Provenance

For the purposes of this paper, provenance is defined as:

> **A time-ordered record of how a system’s configuration, structure, and dependencies evolved, including the context and sequence of changes.**

Provenance captures:

- Lineage
- Causality
- Dependency inheritance
- Change impact over time

It answers a different class of questions than snapshot integrity.

---

3. The Limits of Snapshot-Based Trust

Snapshot artifacts excel at answering:

> *“Has this artifact changed?”*

They do not answer:

- How did this component enter the system?
- What dependencies influenced its behavior?
- What changed first — and what followed?
- What was known at each decision point?

When systems fail, or when accountability is required, these questions become central.

Snapshot evidence alone cannot reconstruct sequence, causality, or context.

---

4. Provenance as Recorded Evolution

Provenance should be understood as an **evolutionary record**, not a static attribute.

In a provenance-aware system:

- Structural changes are observable over time
- Dependencies behave as inherited configuration, not isolated references
- Vulnerabilities propagate along historical paths
- Remediation actions have traceable downstream effects
- Time is treated as a first-class dimension

This approach does not replace existing artifacts.
It **connects them**.

---

5. Governance Implications

Governance requires more than correctness at inspection points. It requires confidence that:

- Changes occurred in an understandable sequence
- Decisions were made with available knowledge at the time
- Impacts can be reconstructed after the fact

Provenance enables:

- Root cause analysis
- Incident reconstruction
- Audit defensibility
- Lifecycle accountability

These outcomes emerge naturally from continuity of record, not from additional reporting layers.

---

6. Making Deception Expensive

Static artifacts can be:

- Re-generated
- Curated
- Selectively disclosed

By contrast, continuous provenance introduces natural friction against misrepresentation:

- Gaps become visible

- Inconsistencies surface across time
- Silence becomes notable
- Absence itself requires explanation

This does not eliminate risk, but it materially raises the cost of concealment.

---

7. Relationship to Existing Practices

The perspective presented here is **complementary**, not disruptive, to existing governance mechanisms, including:

- Integrity verification
- Baseline definition
- Change approval processes
- Compliance reporting

Snapshot artifacts remain necessary.
Provenance provides the **historical coherence** that allows those artifacts to be interpreted meaningfully.

---

8. Principle Statement

The governing principle can be stated succinctly:

> **Provenance is not randomness.**
> **Provenance is recorded evolution.**
> **Trust emerges from history, not snapshots alone.**

---

9. Conclusion

As software systems grow in complexity and longevity, trust increasingly depends on the ability to explain how systems came to be — not merely to verify that they match a recorded state.

Randomness remains essential for discovery.
Integrity checks remain essential for validation.

But **governance depends on memory**.

Provenance is that memory.