Supply Chain Defense by Default

SBOMs, SLSA, and Provenance in CI/CD — a practical guide to making secure delivery pipelines the default, not the exception.

Bhaskar Bharat Sawant

Senior IEEE Member

Southeastern Michigan Section (SEM) Section

Lead engineer & Senior .NET Developer

The Problem: Supply Chain Attacks

Traditional Security Concerns

- Web application vulnerabilities
- Server misconfigurations
- Insider threats

Modern Attack Vectors

- Open-source libraries
- Package registries
- Build servers
- CI/CD runners
- Artifact repositories

Why This Matters Now

Recent high-profile incidents underscore the critical importance of supply chain security.

Incident	What Happened	Impact
SolarWinds	Build environment hacked	18,000 companies compromised
Log4j	Vulnerable dependency triggered global response	Weeks of emergency patching
XZ Backdoor (2024)	Trusted maintainer account compromised to inject backdoor	Nearly entered Linux distros globally

Trust Is Not Assured

Years of usage don't guarantee dependency safety

Pipelines Are Targets

Build systems are strategic entry points

Automation ≠ Transparency

Automated processes need verification

What Supply Chain Defense by Default Means

Defense by default integrates security directly into CI/CD systems, making them the policy engine and gatekeeper, rather than relying on manual checks.

Automated

Security checks run automatically on every build.

Enforced

Policies are consistently applied across all pipelines.

Repeatable

Consistent inputs yield consistent security outcomes.

Observable

All security decisions are logged and auditable.

Software Bill of Materials (SBOM)

An SBOM is the ingredient list of your software, providing transparency into components, versions, and origins.

Why SBOMs Matter

- Rapid vulnerability response
- Efficient component identification
- Regulatory compliance & transparency

Every Build

SBOM generated automatically

Every Artifact

Matching SBOM attached

Every Deployment

Stored with metadata

Provenance & Attestations

Provenance answers critical questions about artifact origin: Who built this? How was it built? Where was it built? With what inputs? Attestations are the signed proofs that the build followed expected processes.

Eliminates Shadow Builds

No unauthorized or undocumented build processes

Prevents Arbitrary Pipelines

Human-triggered builds must follow standards

Blocks Unverified Artifacts

No deployment without proof of trusted origin

If the artifact cannot prove it came from your trusted pipeline, then it should not be allowed to deploy.

SLSA Framework

SLSA (Supply Chain Levels for Software Artifacts) provides an incremental maturity roadmap for secure software delivery.

SLSA Level 1

Build steps tracked and logged

SLSA Level 2

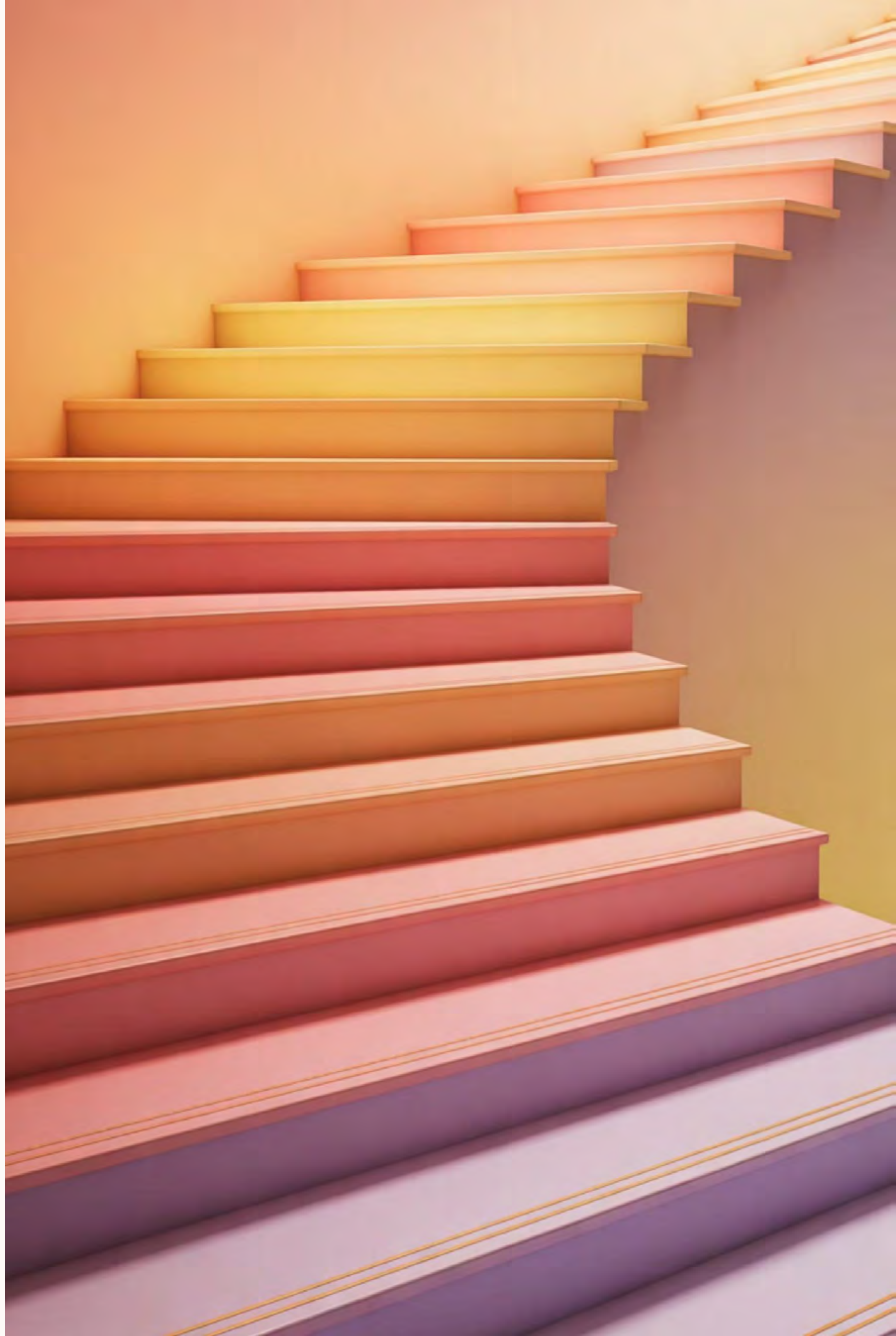
Build is scripted and version-controlled

SLSA Level 3

Provenance and signing enforced

SLSA Level 4

Hermetic and reproducible builds



Pipeline Security Controls

A hardened CI/CD pipeline implements multiple layers of security controls that work together to ensure security is enforced by the system, not by people.

Keyless Signing

Prevents private key theft by eliminating long-lived credentials

Ephemeral Build Runners

Short-lived runners limit lateral movement and reduce attack surface

Immutable Artifact Storage

Ensures artifacts cannot be modified after creation

Deploy-Time Verification

Ensures only authorized components reach production

Hermetic & Reproducible Builds

Hermetic Builds

- No network access during build
- Dependencies pre-fetched, pinned, verified
- Build has no external side effects

Reproducible Builds

- Same input always produces same output
- Bit-for-bit identical artifacts
- Verifiable by independent parties

→ Prevents Dependency Hijacking

Pre-verified dependencies can't be swapped during build

→ Stops Supply Chain Poisoning

No external influence during artifact creation

→ Blocks Build-Time Tampering

Sealed environment prevents interference

> Hermetic builds turn your pipeline into a sealed factory rather than an open workshop.

Dependency Hygiene

Security teams often focus on vulnerabilities, but the real danger is: "Where did this dependency actually come from?" Treating dependencies with the same rigor as production code is essential.

1

Use Dependency Allowlists

Maintain approved registries and sources

2

Pin Every Version

Exact versions, not ranges or latest tags

3

Scan Before Adoption

All new additions validated before use

4

Rotate Maintainers

Distribute trust across multiple reviewers

5

Treat Updates as Change Management

Dependency updates require review and approval

Automation is helpful — but automated updates must never bypass validation.

Real-World Example: Before vs. After

A team lacked artifact verification in their Kubernetes deployments. Implementing supply chain defense transformed their security posture.

Before Implementation

- Manual local rebuilds
- Unsigned artifacts in registry
- No SBOM, no vulnerability insight
- 5 days to locate Log4j impacts

Risk: Malicious builds deployed unnoticed

After Implementation

- Keyless signing enforced by CI
- Auto-generated SBOM with image
- Admission controller required provenance
- SBOM in security dashboard

Impact: XZ-style alerts identified in minutes

5

Days Before

Time to locate vulnerable services

<5

Minutes After

Time to identify impacted systems

0%

Velocity Impact

No deployment slowdown

> We did not slow deployment. We made it safer — by default.

Deployment Enforcement

Automated security enforcement at deployment is critical, ensuring requirements are met where risk is highest.

1

Provenance Check

Valid provenance

2

Attestation Verification

Valid attestations

3

Signature Validation

Signature verified

✗ No Provenance

Deployment blocked automatically

✗ Invalid Attestation

Deployment blocked automatically

✗ Failed Signature

Deployment blocked automatically

Automated enforcement is the only scalable security.

30–60 Day Implementation Roadmap

Here's a practical roadmap to evolve delivery without disruption.

Weeks 1–2

SBOM + artifact metadata storage

Outcome: Visibility

Weeks 3–4

Provenance signing + verification in staging

Outcome: Traceability

Weeks 5–8

Enforce signature verification in production

Outcome: Trusted delivery

The Cultural Shift

Supply chain security is not just a tooling problem — it's a mindset shift.

From: Trusting Pipelines

To: Proving Trust

From: Hoping for Safety

To: Knowing Shipments

From: Security Slowdown

To: Security Speed-Up

Transformation occurs when all teams view supply chain defense as a shared advantage, not an obstacle.

Key Takeaways

Software supply chain: a cybersecurity battlefield

Attackers target build systems & dependencies.

Secure software by securing its build systems

Pipeline security = application security.

SBOMs, provenance, attestation, SLSA, & verification are practical

Implementable solutions, not future ideas.

Security shouldn't rely on heroics. It must be the default outcome of how we build.

Thank you

Bhaskar Bharat Sawant

Lead Engineer at Cornerstone Building Brands