AYI NEDJIMI Consultants

Supply Chain Security : SBOM, SLSA et Sigstore en 2026

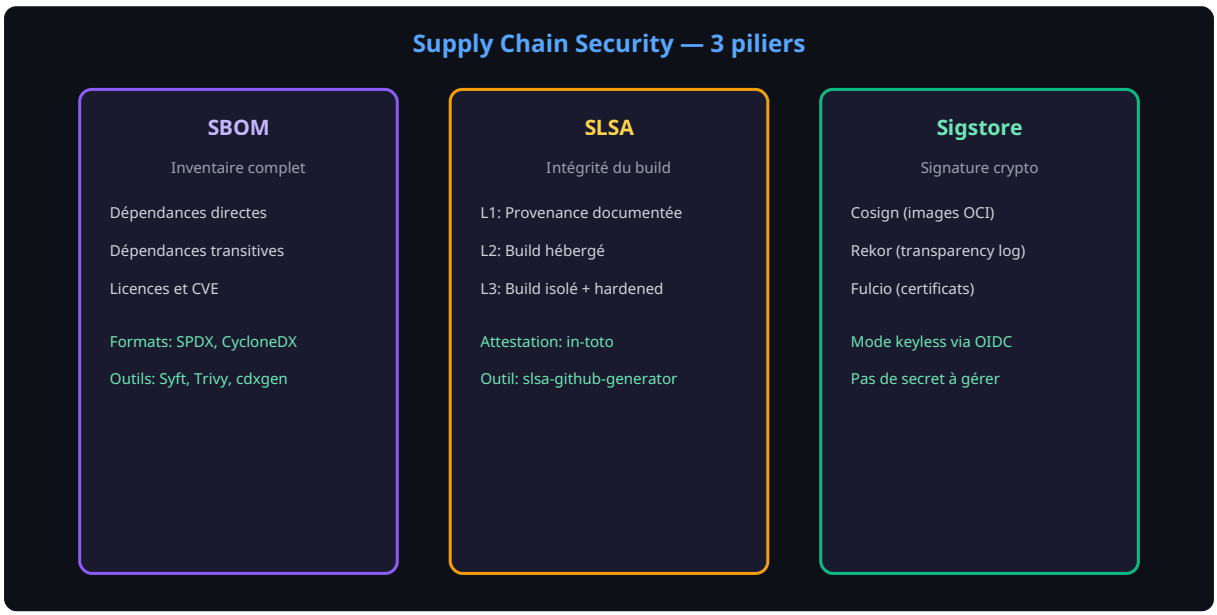
Catégorie : DevSecOps Lecture : 4 min Publié le : 12/03/2026 Auteur : Ayi NEDJIMI

Sécurisez votre chaîne logicielle avec SBOM, SLSA et Sigstore. Guide pratique pour protéger vos dépendances et garantir l'intégrité de vos builds.

L'attaque SolarWinds en 2020, la compromission de CodeCov en 2021, le cauchemar Log4Shell fin 2021, la backdoor XZ Utils en 2024 — la supply chain logicielle est devenue le vecteur d'attaque favori des groupes les plus sophistiqués. Et pour cause : compromettre une dépendance utilisée par des milliers de projets offre un effet de levier incomparable. Vous pouvez verrouiller votre code, durcir vos serveurs, former vos développeurs — si une bibliothèque tierce embarque une backdoor, tout s'effondre. La réponse de l'industrie se structure autour de trois piliers complémentaires. Le SBOM (Software Bill of Materials) dresse l'inventaire exact de ce que contient votre logiciel. SLSA (Supply-chain Levels for Software Artifacts) certifie que votre processus de build est intègre. Sigstore fournit la cryptographie pour signer et vérifier vos artefacts sans gérer de clés. Ce guide vous montre comment déployer ces trois briques concrètement dans vos pipelines existants.

Points clés à retenir

- Un **SBOM** complet au format SPDX ou CycloneDX est exigé par la réglementation US et bientôt par le Cyber Resilience Act européen
- **SLSA** niveau 3 garantit un build isolé, reproductible et vérifiable — c'est la cible pour les applications critiques
- **Sigstore** avec Cosign permet la signature keyless d'images OCI, éliminant le problème de gestion des clés
- La combinaison SBOM + SLSA + Sigstore forme une défense en profondeur contre les attaques supply chain



SBOM : savoir exactement ce que contient votre logiciel

Un **SBOM** est l'équivalent numérique de la liste d'ingrédients sur un produit alimentaire. Pour chaque composant de votre application, il référence le nom, la version, la licence et l'origine. Deux formats dominent : **SPDX** (standardisé ISO/IEC 5962) et **CycloneDX** (maintenu par l'OWASP). En pratique, CycloneDX est plus adapté à la sécurité car il inclut nativement les VEX (Vulnerability Exploitability eXchange). Mon conseil : partez sur CycloneDX si votre objectif premier est la sécurité.

Pour générer un SBOM, **Syft** (d'Anchore) est la référence open-source. Une commande suffit : `syft packages dir:. -o cyclonedx-json`. Ce SBOM peut ensuite être analysé par **Grype** pour détecter les CVE. L'Executive Order 14028 américain exige un SBOM pour tout logiciel vendu au gouvernement fédéral. En Europe, le Cyber Resilience Act imposera une obligation similaire dès 2027. Intégrez la génération de SBOM dans votre pipeline CI/CD sécurisé dès maintenant.

SLSA : certifier l'intégrité de votre processus de build

Le SBOM vous dit ce qu'il y a dans votre logiciel. **SLSA** (prononcez "salsa") garantit que le processus de construction n'a pas été altéré. Développé par Google, ce framework définit trois niveaux de maturité. Le niveau 1 produit une provenance documentée. Le niveau 2 utilise un service de build hébergé avec des logs inaltérables. Le niveau 3 isole et durcit le build : pas d'accès réseau pendant la compilation, environnement éphémère, reproductibilité vérifiable.

Sur GitHub Actions, atteindre SLSA 3 se fait avec le `slsa-github-generator`. La provenance générée est une attestation **in-toto** vérifiable par n'importe quel consommateur de votre artefact. C'est exactement ce qui manquait lors de l'attaque SolarWinds : personne ne pouvait vérifier que le binaire distribué correspondait au code source audité. Pour comprendre les attaques sur les pipelines, notre article sur les attaques CI/CD et GitOps détaille les vecteurs exploités.

Sigstore : signer sans gérer de clés

La signature cryptographique d'artefacts existe depuis longtemps (GPG, clés PEM), mais la gestion des clés est un cauchemar opérationnel. **Sigstore** résout ce problème avec le mode keyless : votre identité CI (l'OIDC token de GitHub Actions) sert de preuve. **Fulcio** émet un certificat éphémère, **Cosign** signe l'artefact, et **Rekor** enregistre la signature dans un log de transparence immuable.

```
# Signer une image OCI (mode keyless)
cosign sign ghcr.io/myorg/myapp@sha256:abc123

# Vérifier la signature
cosign verify   --certificate-identity https://github.com/myorg/myapp/.github/workflows/
build.yml@refs/heads/main   --certificate-oidc-issuer https://
token.actions.githubusercontent.com   ghcr.io/myorg/myapp@sha256:abc123
```

En production, configurez un admission controller Kubernetes (comme Kyverno) pour rejeter les images non signées. Pour les politiques Kubernetes, notre article sur Policy as Code avec OPA et Kyverno vous guide dans cette mise en place.

Workflow complet de défense supply chain

Les trois piliers fonctionnent en synergie. Voici le workflow recommandé pour un projet conteneurisé :

1. **Build** — Compiler dans un environnement isolé (SLSA 3). Générer le SBOM avec Syft.
2. **Scan** — Analyser le SBOM avec Grype ou Dependency-Track. Gate bloquante sur les CVE critiques.
3. **Sign** — Signer l'image avec Cosign keyless. Attacher le SBOM comme attestation OCI.
4. **Store** — Pousser l'image signée et le SBOM sur un registry OCI compatible (Harbor, GHCR, ECR).
5. **Verify** — L'admission controller Kubernetes vérifie signature et SBOM avant admission.
6. **Monitor** — Dependency-Track surveille les nouvelles CVE qui affectent vos SBOM existants.

Ce workflow ajoute 2-3 minutes à votre pipeline. C'est dérisoire comparé au coût d'une compromission. Selon le rapport Sonatype State of the Software Supply Chain 2025, les attaques supply chain ont augmenté de 245% en un an.

Surveiller les dépendances en continu

Générer un SBOM au moment du build ne suffit pas. Une CVE découverte trois mois après votre release affecte vos artefacts déjà déployés. **OWASP Dependency-Track** résout ce problème : vous uploadez vos SBOM, et la plateforme surveille en continu les nouvelles vulnérabilités. Elle interroge la base NVD du NIST, OSV.dev et les advisories GitHub.

Quand une nouvelle CVE est publiée, vous recevez une alerte avec la liste exacte des projets affectés et des versions à mettre à jour. C'est ce mécanisme qui a permis aux organisations équipées de réagir en quelques heures lors de Log4Shell. Pour la gestion des secrets liés à ces dépendances, consultez notre guide sur le secrets management cloud.

Sources et références : OWASP DevSecOps · NIST

Questions fréquentes sur la supply chain security

Mon projet utilise peu de dépendances, suis-je vraiment concerné ?

Oui. Même 10 dépendances directes génèrent souvent 200 dépendances transitives ou plus. Une seule d'entre elles suffit pour compromettre votre application. L'attaque event-stream de 2018 visait une dépendance de troisième niveau utilisée par des milliers de projets sans que personne ne le sache. Le SBOM révèle cette réalité cachée.

SPDX ou CycloneDX, lequel choisir ?

Pour la sécurité, CycloneDX. Il supporte nativement les VEX et les attestations de build. Pour la conformité licences dans un contexte juridique, SPDX qui est une norme ISO. Si vous devez n'en choisir qu'un, CycloneDX couvre 90% des besoins DevSecOps courants.

Sigstore fonctionne-t-il en environnement air-gapped ?

Pas en mode keyless, qui nécessite une connexion à Fulcio et Rekor. En air-gapped, vous pouvez déployer votre propre instance Sigstore (Fulcio + Rekor) on-premise, ou revenir à un mode clé classique avec Cosign et une paire de clés gérée manuellement.

FAQ

Qu'est-ce que Supply Chain Security ?

Supply Chain Security désigne l'ensemble des concepts, techniques et méthodologies abordés dans cet article. Les fondamentaux sont détaillés dans les premières sections du guide.

Pourquoi supply chain security sbom slsa est-il important ?

La maîtrise de supply chain security sbom slsa est devenue essentielle pour les équipes de sécurité. Les enjeux et le contexte opérationnel sont développés tout au long de l'article.

Ayi NEDJIMI Consultants — Expert cybersécurité offensive & intelligence artificielle

ayinedjimi-consultants.fr · ayi@ayinedjimi-consultants.fr