CI/CDパイプラインの実行来歴に基づく完全性検証手法
—PBOMとin-totoフレームワークの統合アプローチ—

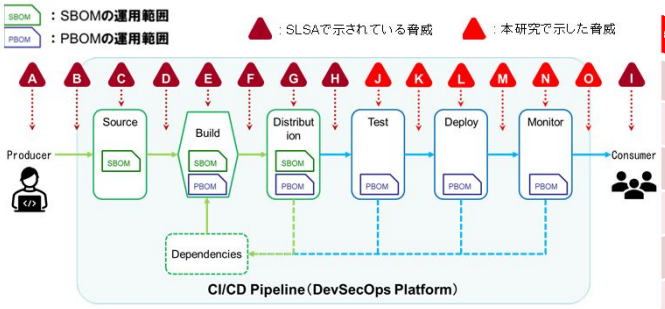
Proposal and Implementation of a Completeness Verification Method Based on CI/CD Pipeline Execution Provenance: An Approach Integrating PBOM and the in-toto Framework

吉村 隼哉・システム分科会・情報セキュリティ大学院大学

Abstract: Attacks targeting software supply chains (SSCs) are becoming more serious. With the spread of DevSecOps, a variety of tools have been introduced to streamline and automate operations, and CI/CD Pipelines have become more complex and bloated, increasing the threat of SSCs. In this research, we concretize and implement the concept of a Pipeline Bill of Materials (PBOM) for managing the components of CI/CD Pipelines. Furthermore, we integrate the in-toto framework with PBOM to develop an integrity verification method based on the execution provenance of CI/CD Pipelines and demonstrate its effectiveness.

1. 研究背景

・DevSecOpsの普及によりCI/CDパイプラインは複雑化・肥大化しているが，パイプラインの構成要素を管理する手法は確立されていない
・既存フレームワークは開発生成物およびビルドフェーズの対策に集中し，CI/CDパイプライン全体のセキュリティ対策には限界がある



[1] SLSA(Supply-chain Levels for Software Artifacts) を元に筆者が作成した脅威モデル

CI/CDパイプラインにおける脅威を整理

脅威	フェーズ	脅威の概要	代表事例
J	Test	Test フェーズにおいて実行される試験・解析ツールや，その依存関係が侵害される脅威.	**・npm (Shai Hulud) ・Codecov**
K	Test / Deploy	Test フェーズから Deploy フェーズへ引き渡される成果物が，意図しない差し替えや，誤った選択がされることにより，不正な成果物が利用される脅威.	・Package Confusion ・Passwordstate
L	Deploy	コンテナ実行環境やオーケストレーション基盤が侵害されることで，意図しない挙動や不正処理が実行される脅威.	・Docker ・Kubernetes
M	Deploy / Monitor	Deploy フェーズとMonitor フェーズの連携不備や設定不整合により，実行中のシステムが正しく監視されず，障害・侵害・異常が検知されない脅威.	・Sidecar Injection
N	Monitor	監視・ログ収集・可観測性ツール自体に脆弱性や設定不備が存在することで機密情報の漏洩，不正な操作，監視データの改ざんが発生する脅威.	**・SolarWinds**
O	Pipeline as Code	CI/CDパイプライン定義（Workflow/Job/Runner/Secrets/権限設定 等）の改ざん・誤設定・運用不備により，不正実行や機密情報流出が発生する脅威.	・GitHub Actions ・GhostAction

2. 提案手法

■PBOMとin-totoフレームワークを統合することで，CI/CDパイプライン構成要素の管理と実行プロセスの検証を実現

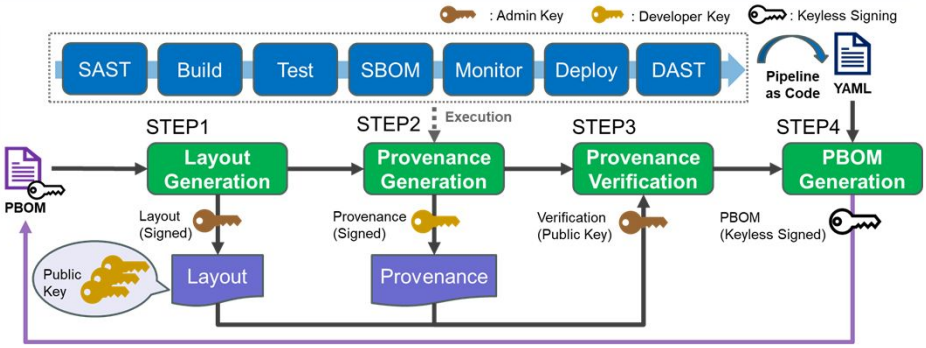
- **PBOM（Pipeline Bill of Materials）の具体的要素を提案し，CI/CDパイプラインで使用されるツールの自動管理を実現**
- **in-totoによる来歴検証の期待値としてPBOMを活用することで，実行プロセスの完全性検証における品質向上および自動化を実現**

■手順

【 STEP1 】:PBOMを基に期待値（Layout）生成
【 STEP2 】:実行来歴（Provenance）生成
【 STEP3 】:来歴検証（Layout-Provenance）
【 STEP4 】:検証されたCI/CDのPBOMを生成

※初期導入時はSTEP0として手動生成を実施

**DevSecOpsのように
頻繁に実行される開発環境で
高い効果を発揮**

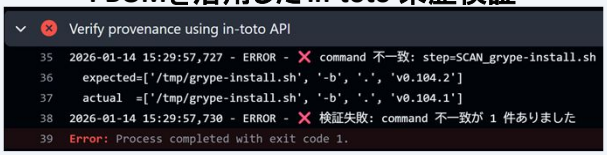


3. 実装と評価

— PBOM生成 —

- ✓ **YAMLファイルを解析し，CycloneDX形式で出力可能**
- ✓ **CycloneDXの拡張要素を活用し，PBOMの独自要素を実現**
- ✓ **脆弱性管理基盤との連携が可能（Dependency-Track にインポートし，脆弱性検知を確認）**

— PBOMを活用した in-toto 来歴検証 —



- ✓ **PBOMで管理されたバージョンと異なるバージョン指定（コマンド改ざん）が検知されることを確認**
- ✓ **実行ユーザー検証や実行プロセス欠如の検知にも対応**

4. 結論

- PBOMは，CI/CDパイプラインに含まれるツールを管理し，来歴検証や脆弱性管理に活用可能である.
- PBOMとin-totoを統合した来歴検証は，CI/CDパイプラインで発生した意図しない実行プロセスを検出可能である. また，自動化による検証コスト削減・検証品質向上を実現可能である.

5. 今後の展望

【検証範囲拡大】: CI/CDパイプライン外プロセスへの検証範囲拡大検討
【PBOMの一般化検討】: PBOMの運用ルールや外部共有方法の検討
【再帰検証の実現】: 複数組織を横断した再帰検証の実現に向けた検討
【他手法との統合】: 他手法（例 Reproducible Builds ）との統合検討