複数ビルド結果を用いた Nix Binary Cache の信頼性評価

野田蒼馬　N 高等学校二年

実装・検証コード (GitHub)

1. 背景

1.1. 目指す世界

「信頼する・しない」の二択から、 **証拠に基づく信頼のグラデーションを作ること** を中心に据える。

誰が配布したかではなく、 **どのような証拠があるか** で判断できるようにする。特定の巨大な Binary Cache だけに依存せず、多様な Cache を安心して利用できる Nix エコシステムを実現する。

盲目的な信頼，単一障害点の無い持続可能な Nix エコシステムの構築を目指す

1.2. Nix とは

Nix は、同じ入力から同じビルド結果を得ることを重視したパッケージ管理・ビルドシステムである。Nix を使用した NixOS では、OS の設定や導入するソフトウェアをコードとして管理でき、同じ設定から同等の環境を再構築できる。

この再現性を高めるため、Nix は外部環境の影響を抑えたサンドボックス内でビルドを行う。しかし、依存するソフトウェアも含めてビルドするため、処理に長い時間がかかり、ディスク容量も消費する。

そこで Nix では、あらかじめビルドされた成果物を取得できる **Binary Cache** が利用される。

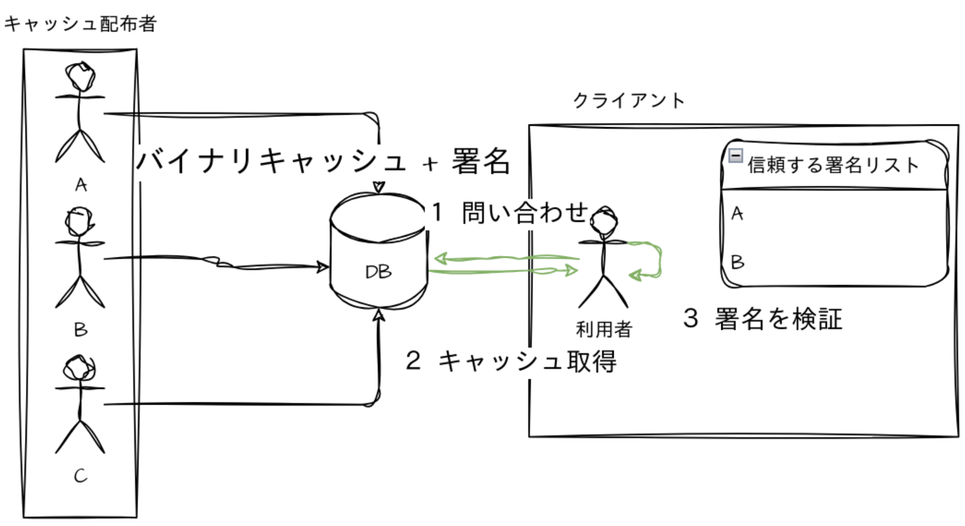
2. 現行の信頼モデルと課題

現行の信頼モデルには以下のような課題がある。

- 信頼判断が鍵単位の「信頼する・しない」に限られる
- 署名鍵が侵害されると、不正な成果物も正当と判断され得る
- 利用実績のある **大規模な Cache へ信頼が集中** する

Binary Cache は、あらかじめビルドされた成果物を署名と共にサーバーへ保存する仕組みである。利用者は自分でビルドせずに成果物をダウンロードできるため、ビルド時間を大幅に短縮できる。ただし、成果物の正当性は主に **Cache 運営者の署名鍵を信頼** して判断することになる。

図 1　署名鍵を基準とする現行の信頼モデル



注： A〜C は Cache 配布者。利用者は取得後、署名者が信頼リストに含まれるかを検証する。

3. 提案するアプローチ

Binary Cache の成果物を署名鍵だけに頼らず評価するために、 **複数のコンピュータでビルド** を実行し、その結果をサーバーに記録する。

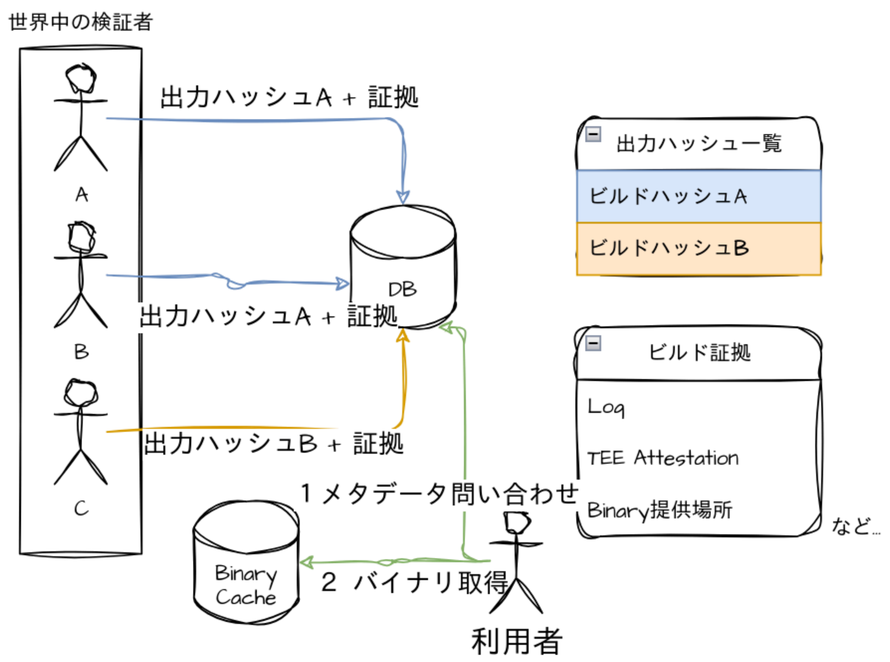
現状は以下の手法を組み合わせることを考えている。

- TEE(Trusted Execution Environment)による **実行証明**
  - 入力・ビルド環境・出力を結びつける
- Commit-Reveal による **後追い提出の抑止**
  - Commit = SHA256(ビルド証拠 && Salt)

現行方式との比較

課題	現行	提案	説明
信頼判断が鍵単位の**「信頼する・しない」に限られてしまう**	×	◎	ビルド証拠によって**信頼にグラデーション**ができる
署名鍵が侵害されると、不正な成果物も**正当と判断され得る**	×	○	署名鍵以外の証拠を加え、**署名鍵だけへの依存を軽減**する
利用実績のある**大規模な Cache へ信頼が集中**する	△	△	ローカルで証明を検証可能なのでサーバーは比較的分散できるが、根本的な問題は解決しない

図 2　複数のビルド証拠を用いる提案モデル



注： A〜C は独立した検証者。出力ハッシュと実行証拠を蓄積し、利用者が取得前に評価する。

4. 入力と成果物の一致を検証する

利用者が確かめたいのは、 **指定した入力に対応する成果物** である。たとえば Google Chrome を導入しようとしたのに、Chromium へすり替えられるのは困る。

本研究はマルウェア判定ではなく、 **指定した入力から得られるはずの成果物との一致** を検証する。マルウェア検出は VirusTotal などの専用サービスに任せる方が効率的で現実的な可能性が高い。

5. 脅威モデル

現状は以下を想定している。

- Sybil Attack
  - 複数のビルダーを装い、一致数と信頼スコアを水増しする。
- TEE 侵害・Attestation 偽装
  - TEE の脆弱性を悪用し、不正な成果物を正当と偽る。
- 証拠のリプレイ
  - 証拠を再利用・複製し、独立した証拠数を水増しする。

参考文献

[1] A. Hoese, “Trustix: Distributed Trust and Reproducibility Tracking for Binary Caches,” Tweag, 2020.
[2] J. Malka and A. Engelen, “Lila: Decentralized Build Reproducibility Monitoring for the Functional Package Management Model,” Proc. MSR ’26, 2026. doi: 10.1145/3793302.3793328.
[3] D. Hugenroth, M. Lins, R. Mayrhofer, and A. R. Beresford, “Attestable Builds: Compiling Verifiable Binaries on Untrusted Systems Using Trusted Execution Environments,” Proc. ACM CCS ’25, pp. 4514-4528, 2025. doi: 10.1145/3719027.3765128.