INSTITUTE of INFORMATION SECURITY

Attestable Build Chain: Enabling Trust in Reproducible Builds

DSN 2026 Workshop - HAP 2026:
First Workshop on Holistic Architecture for Privacy in Clouds, Edges, and IoT
(2026.6.22)

Kenichiro Muto and Kuniyasu Suzaki

Institute of Information Security

Motivation and Contribution

- Software Supply Chain Trust
  - Software supply chain attacks increasingly target the build process.
    - Examples: SolarWinds, XcodeGhost, PyPI/npm repository compromises …
  - Reproducible Builds verify output equivalence, but not build-time execution.

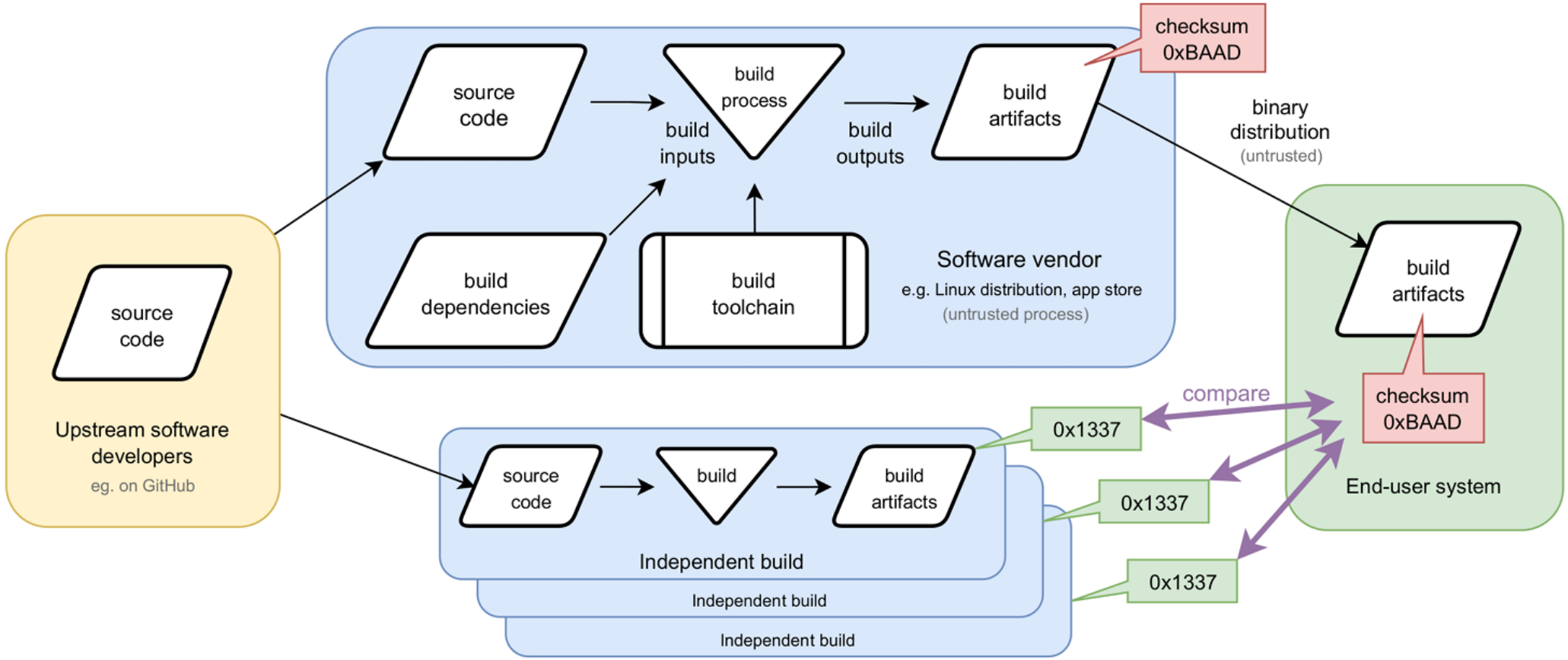
- Goal
  - Make build-time execution externally verifiable.

- **Our Contributions**
  - C1: Verifiable Build Execution
    - Two-phase attestation for both the build environment and build-time execution.
  - C2: Hardware-rooted Execution Evidence
    - Tamper-evident IMA evidence protected by vTPM and SEV-SNP attestation.
  - C3: Practical System and Evaluation
    - Prototype implementation on an SEV-SNP CVM and evaluation with OSS projects.

Background: Reproducible Builds

INSTITUTE of INFORMATION SECURITY

- Increase trust in software build outputs.
- Independently rebuilt artifacts should produce identical binaries.
- Enable third-party verification of software artifacts



Source: Chris Lamb and Stefano Zacchiroli, “Reproducible Builds: Increasing the Integrity of Software Supply Chains,”, IEEE Software, 2022.

Background: Software Bill of Materials (SBOM)

INSTITUTE of INFORMATION SECURITY

- SBOM (Software Bill of Material)
  - Provides transparency into software composition.
  - Includes information such as:
    - Dependencies
    - Licenses
    - Checksums (hash values)
  - Common SBOM Formats
    - SPDX
    - Cyclone DX
    - SWID
- **Focus of This Work**
  - Software supply chain integrity
  - Tamper detection using checksums

2.3 Mapping to Existing Formats

Table 1 (recreated from Table 1 in *Survey of Existing SBOM Formats*[11]) maps baseline component attributes across SPDX and SWID. More formats and their mappings are described in later in the *Survey* document.

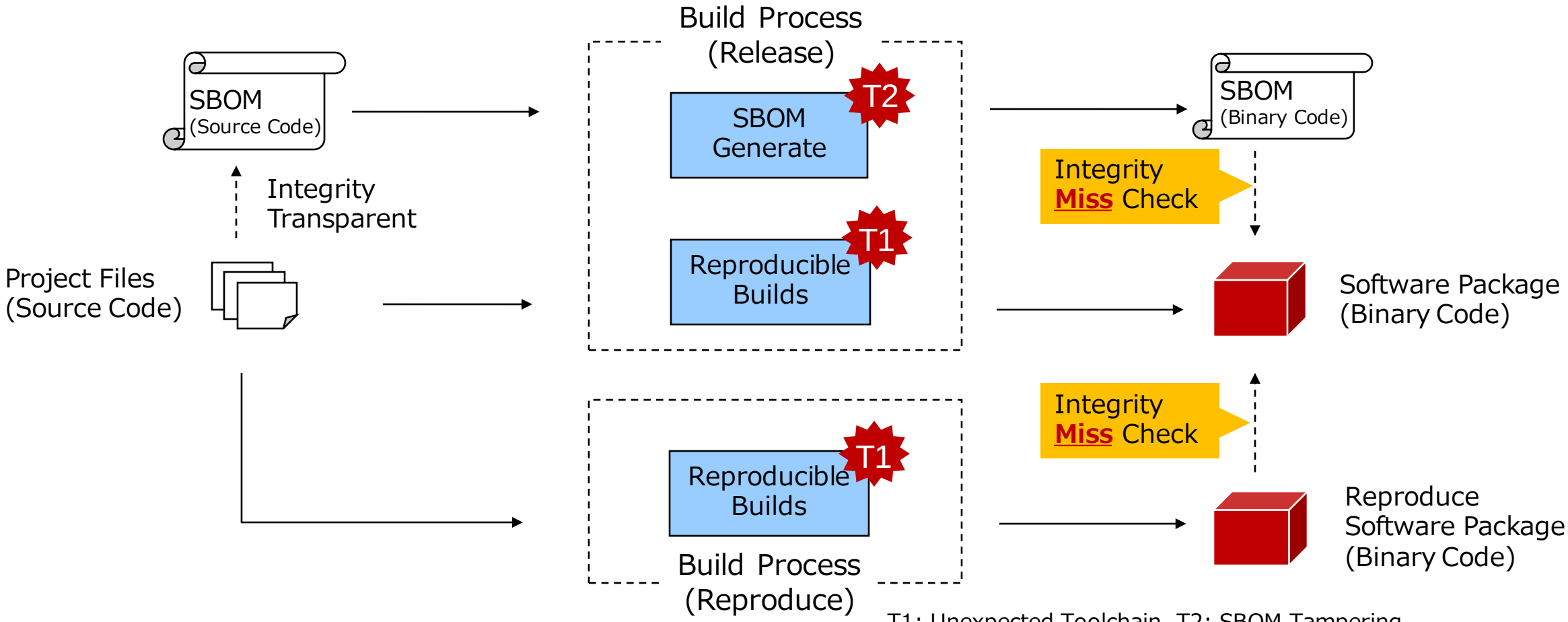
Baseline	SPDX	SWID
Supplier Name	(3.5) PackageSupplier:	<Entity> @role (softwareCreator/publisher), @name
Component Name	(3.1) PackageName:	<softwareIdentity> @name
Unique Identifier	(3.2) SPDXID:	<softwareIdentity> @tagID
Version String	(3.3) PackageVersion:	<softwareIdentity> @version
Component Hash	(3.10) PackageChecksum:	<Payload>/../<File> @[hash-algorithm]:hash
Relationship	(7.1) Relationship: CONTAINS	<Link> @rel, @href
Author Name	(2.8) Creator:	<Entity> @role (tagCreator), @name

Table 1: Mapping baseline component information to existing formats

Figure adapted from: NTIA, "Framing Software Component Transparency:Establishing a Common Software Bill of Materials (SBOM)," Framing Working Group, Nov. 2019.

Threat Model

- Even when Reproducible Builds and SBOMs are available,
  - the actual build-time execution is not observable,
  - and software consumers cannot verify how artifacts were produced.



T1: Unexpected Toolchain, T2: SBOM Tampering

Research Question

- Research Gap
  - Current approaches verify outputs and software components,
  - but cannot verify the actual build-time execution.

- **Research Question**
  - RQ1. Build Process Visibility
    - Can we observe the actual files and toolchain components used during a build?
  - RQ2. Integrity of Build Evidence
    - Can we make build evidence tamper-evident?
  - RQ3. Third-Party Verification
    - Can a third party verify the build process?

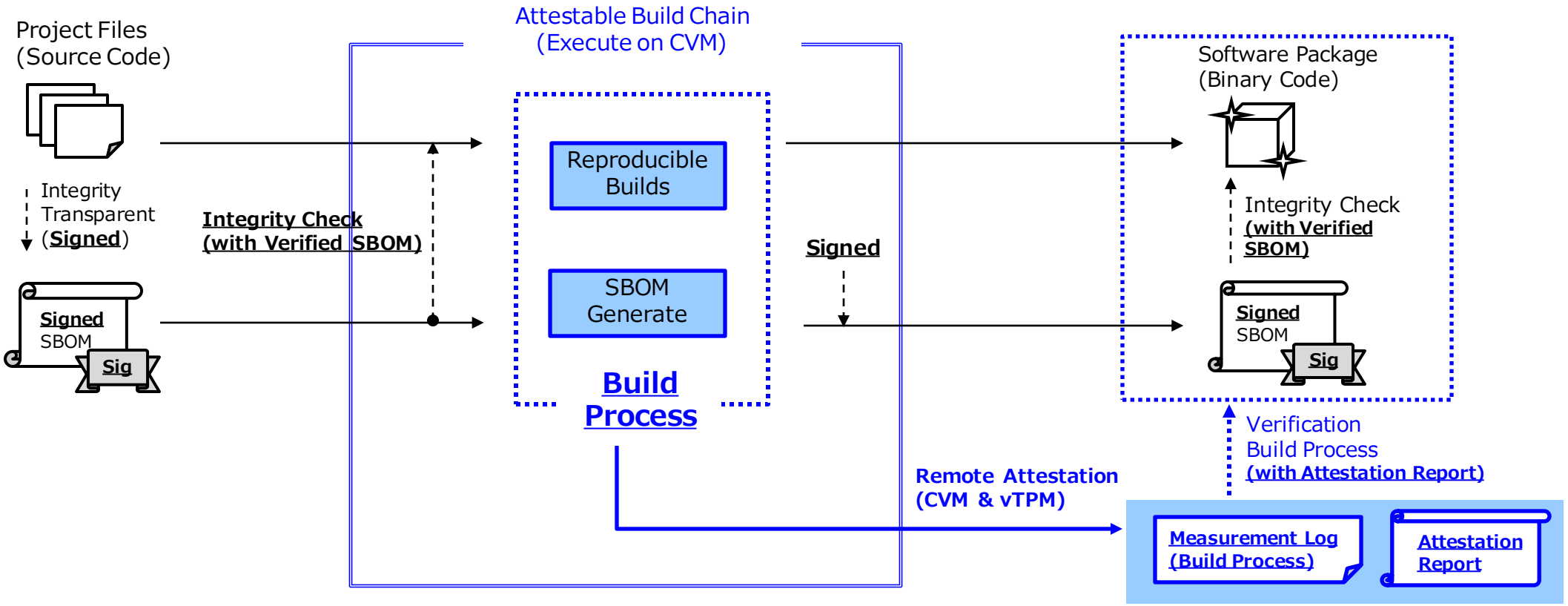
(Approach and Assumptions)

- The build environment is deployed in a CVM.
- Trusted reference values are securely provisioned to the verifier (e.g., via DDC).
- Side-channel attacks and compromised trust anchors are out of scope.

Key Idea: Attestable Build Chain

INSTITUTE of INFORMATION SECURITY

- Extend verification from build outputs to build execution.
  - Input/Output Integrity: Verify signed SBOMs and artifacts.
  - **Execution Integrity: Make build-time execution externally verifiable.**

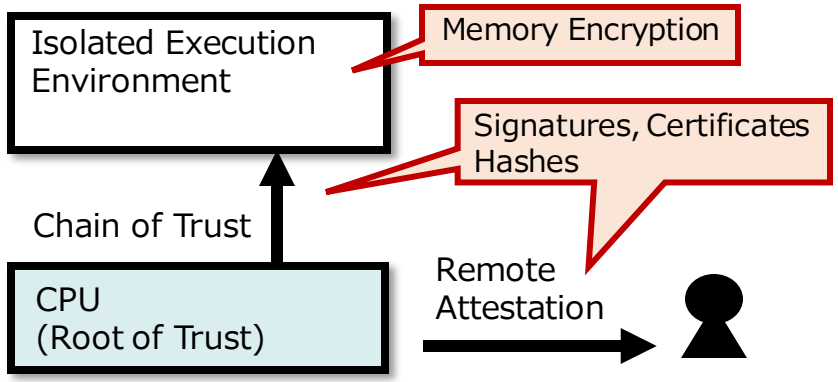


Key Technologies: Remote Attestation with CVM

INSTITUTE of INFORMATION SECURITY

Confidential VM (CVM)

- A VM-based Trusted Execution Environment (TEE) increasingly adopted in cloud environments (e.g., AMD SEV-SNP).
- Protects the confidentiality and integrity of the execution environment through hardware-assisted isolation and memory encryption.

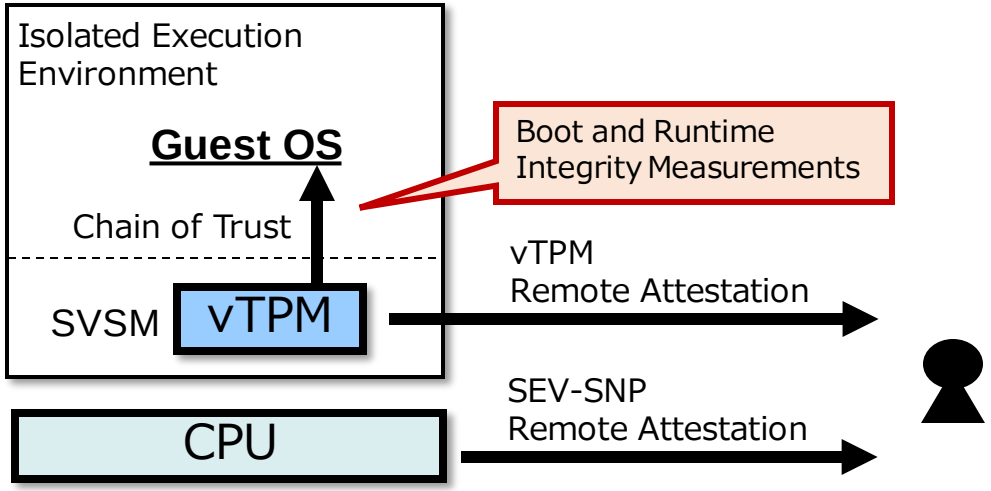


Fundamental Properties of a Confidential VM

Virtual TPM inside a CVM

- A virtual TPM (vTPM) is deployed inside the CVM. The vTPM is protected inside the CVM by Secure VM Service Module(SVSM).
- Remote attestation enables external verification of the integrity of the CVM and its vTPM state.

We extend CVM and vTPM remote attestation from environment verification to build execution verification.

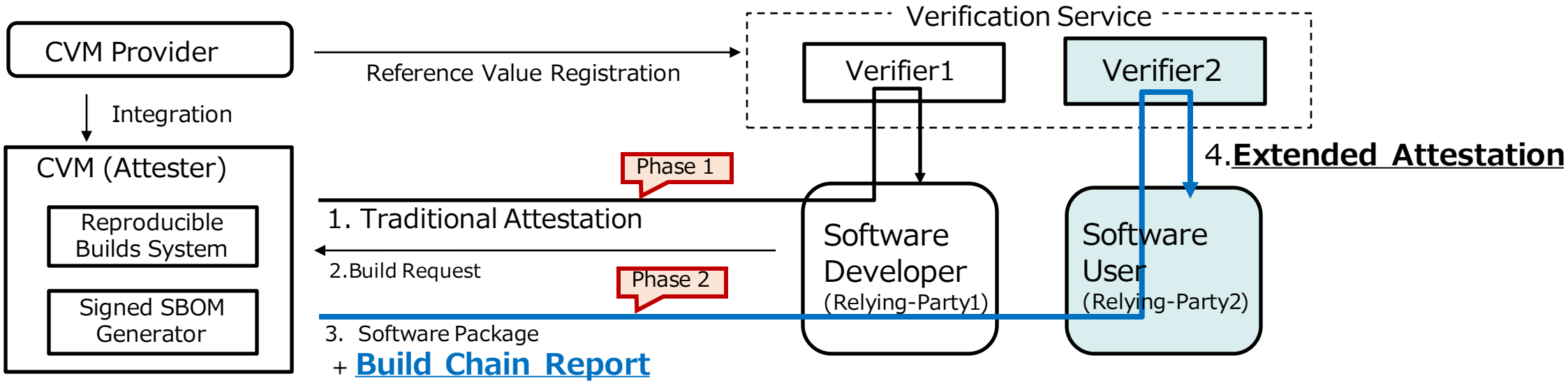


SVSM-based vTPM Architecture in a CVM
(Concept adapted from Narayanan et al., ACSAC 2023.)

System Design Overview

INSTITUTE of INFORMATION SECURITY

- System Roles
  - Build Server (Attester)
    - Executes reproducible builds inside a CVM. Generates a signed SBOM and build execution evidence.
    - Build execution evidence is packaged as a Build Chain Report.
  - Developer / User (Relying Party)
    - Requests builds and receives software artifacts.
  - Verifier
    - Verifies the build environment and build-time execution against registered reference values.

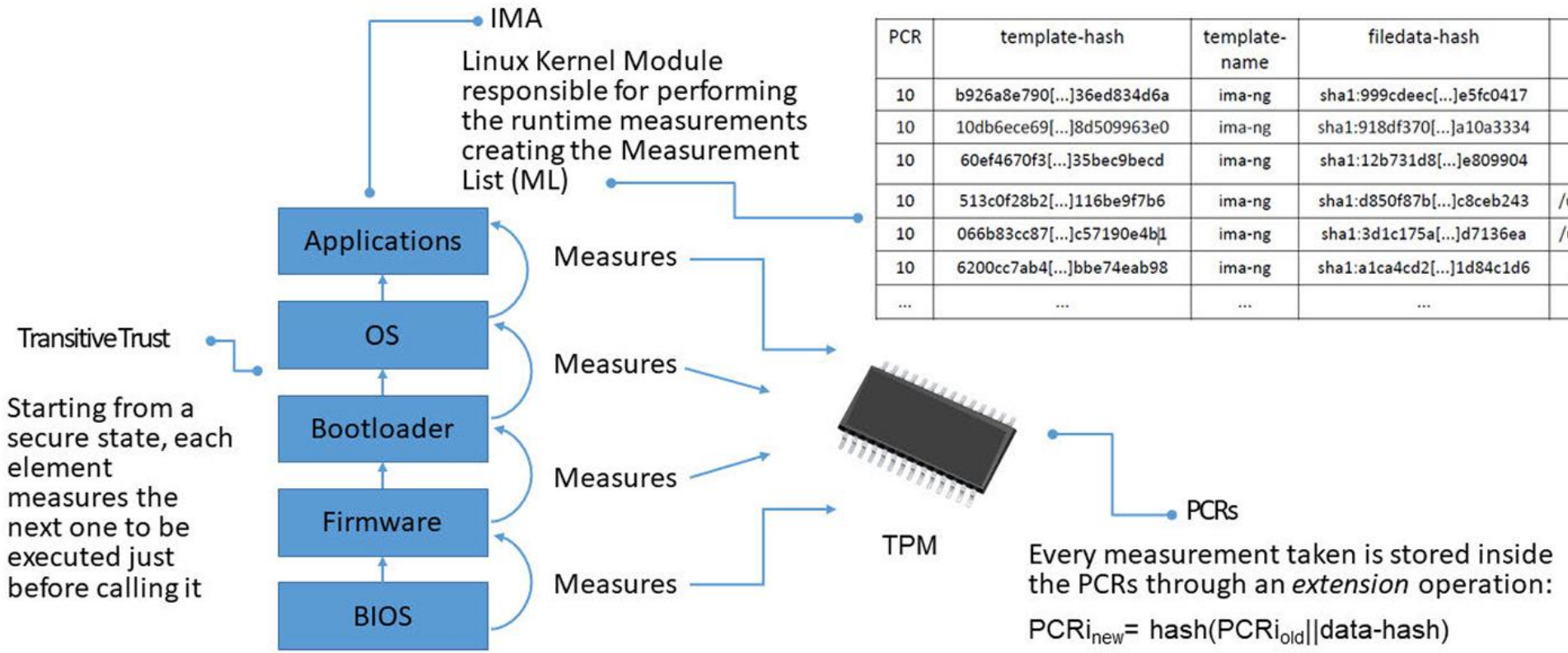


Two-Phase Attestation Workflow (Phase 1 verifies the build environment, while Phase 2 verifies build-time execution.)

Point1: Execution Evidence Collection

INSTITUTE of INFORMATION SECURITY

- We focus on Linux IMA measurements and TPM-based remote attestation.
- IMA records runtime measurements and binds them to TPM PCRs.
- We apply this mechanism to generate verifiable build execution evidence.



PCR	template-hash	template-name	filedata-hash	filename-hint
10	b926a8e790[...]36ed834d6a	ima-ng	sha1:999cdeec[...]e5fc0417	boot_aggregate
10	10db6ece69[...]8d509963e0	ima-ng	sha1:918df370[...]a10a3334	/usr/bin/kmod
10	60ef4670f3[...]35bec9becd	ima-ng	sha1:12b731d8[...]e809904	/usr/lib/.../ld-2.31.so
10	513c0f28b2[...]116be9f7b6	ima-ng	sha1:d850f87b[...]c8ceb243	/usr/lib/.../liblzma.so.5.2.4
10	066b83cc87[...]c57190e4b1	ima-ng	sha1:3d1c175a[...]d7136ea	/usr/lib/.../libcrypto.so.1.1
10	6200cc7ab4[...]bbe74eab98	ima-ng	sha1:a1ca4cd2[...]1d84c1d6	/usr/lib/.../libc-2-31-so
...	...	...	...	...

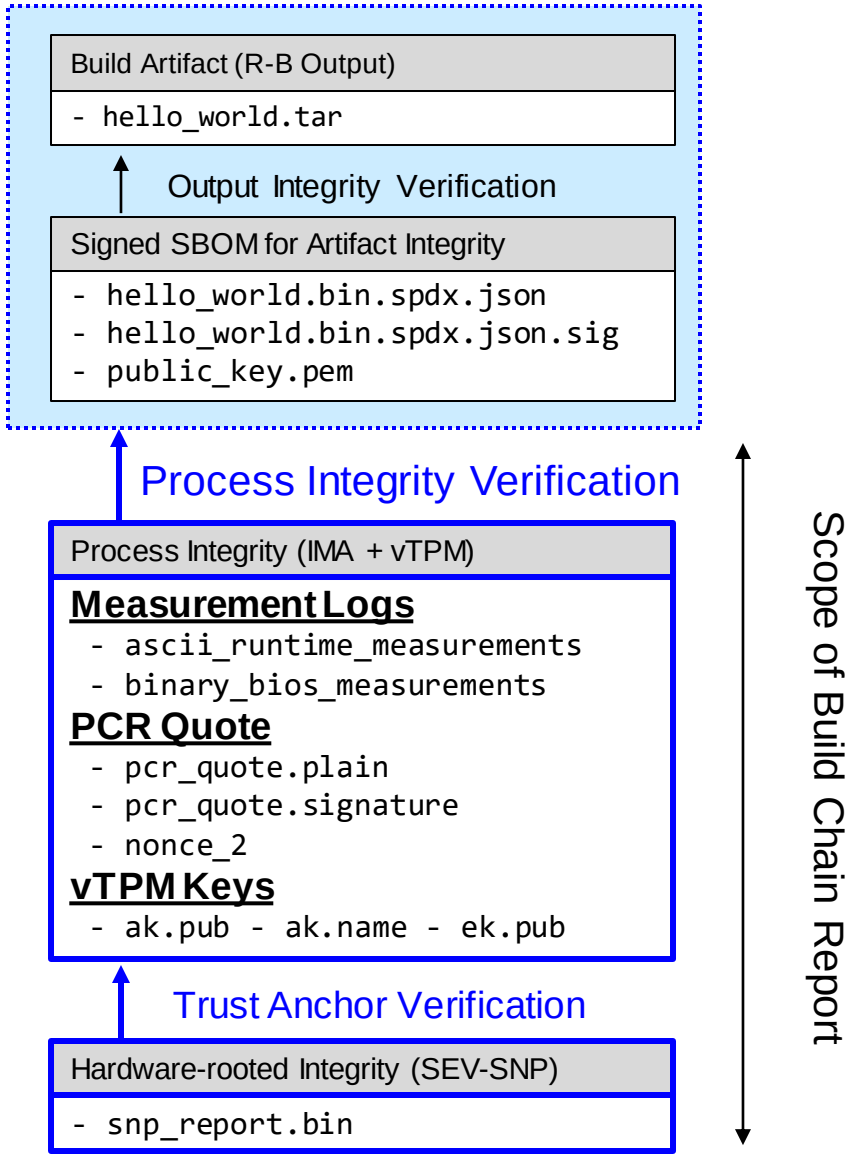
FIGURE 2. Software integrity measurement components (exploiting IMA and TPM).

Figure reproduced from: Diana Gratiela Berbecaru, et al. "Mitigating Software Integrity Attacks With Trusted Computing in a Time Distribution Network", IEEE Access, vol. 11, pp. 50510–50527, 2023.

Point2: Build Chain Report

INSTITUTE of INFORMATION SECURITY

- A Build Chain Report packages all evidence required for third-party verification.
- It binds together:
  - Build artifacts and signed SBOMs
  - IMA measurement logs and TPM quotes
  - Hardware-rooted attestation evidence (SEV-SNP)
- Verifiers can validate:
  - Output integrity
  - Build-process integrity
  - Trust-anchor integrity
- This enables third parties to verify build-time execution without rerunning the build.



Prototype Implementation

- Key Implementation Features
  - Bazel-based reproducible builds
  - Customized Linux IMA measurements
    - File accesses + repeated measurements
  - SEV-SNP and vTPM attestation
    - COCONUT-SVSM-based CVM
  - Build Chain Report generation and verification
- Prototype Scope
  - An end-to-end prototype integrating build execution evidence, hardware-rooted attestation, and third-party verification.

Table. Implementation Environment

Component (Role)	Machine Spec with OS and Tools (Version)
Build Client (Relying-Party_1&2)	AMD Ryzen 3 7330U Processor Virtual Box (7.1.4) - 2CPU (2295.6 MHz × 2), Memory 2048 MB Ubuntu Server (24.04.2 LTS) Kernel (6.8.0-59-generic) Python3 (3.12.3), Requests(2.32.3) OpenSSL (3.0.13 30 Jan 2024) tpm2-tools (5.6-1build4)
Build Server (Attester)	AMD EPYC 7313P Processor qemu (v8.2.0-81-g260df2b36b-dirty) - 4CPU (3000.0 MHz × 4), Memory 8192MB Ubuntu Server (24.04 LTS) Kernel (6.8.0-coconut-svsm-sevsnp+) *customized Python3 (3.12.3), Flask (3.1.0) OpenSSL (3.0.13 30 Jan 2024) VirTEE snpguest (0.5.1) tpm2-tools (5.7-12-gbd832d3f) Bazel (7.6.1)
Verification Server (Verifier_1&2)	AMD Ryzen 3 7330U Processor Virtual Box (7.1.4) - 2CPU(2295.6 MHz × 2), Memory 2048 MB Ubuntu Server (24.04.2 LTS) Kernel (6.8.0-59-generic) Python3 (3.12.3), Flask (3.1.0), PyJWT (2.10.1) OpenSSL (3.0.13 30 Jan 2024) VirTEE snpguest (0.8.3) tpm2-tools (5.6-1build4)

Evaluation：OSS Applicability

- Applied the prototype to four real-world OSS projects.
  - Covered multiple languages and build systems (C++, Rust, Go / Bazel, Cargo).
  - Successfully generated artifacts, signed SBOMs, and build execution evidence.
  - Required minor BUILD file modifications to support Bazel-based builds.

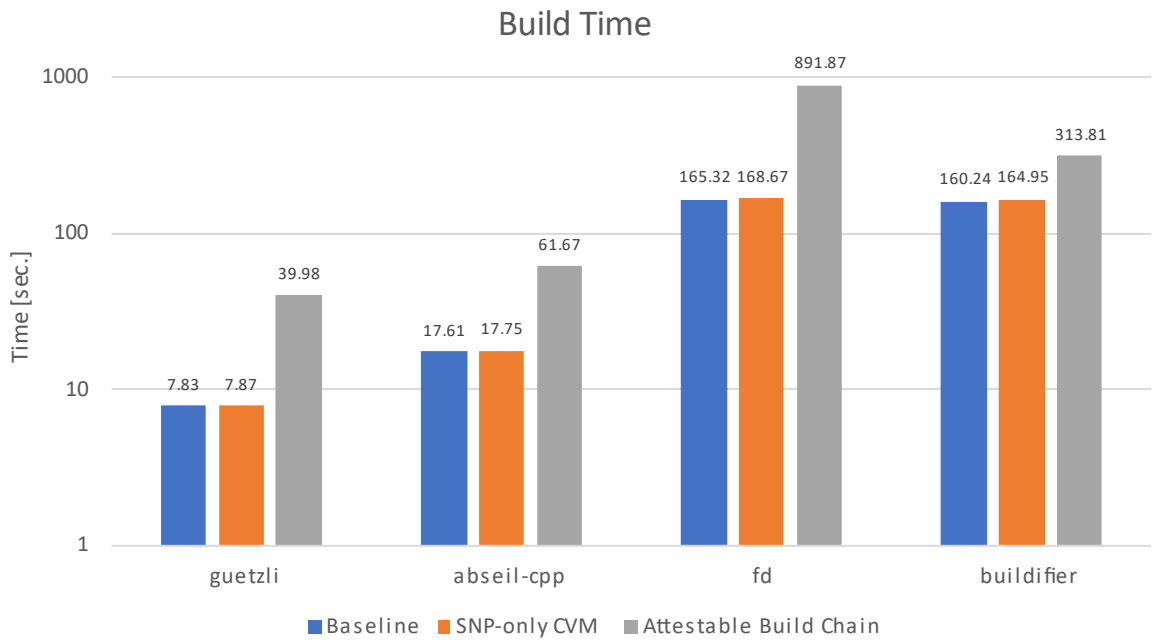
Table. Build Targets (Four OSS projects)

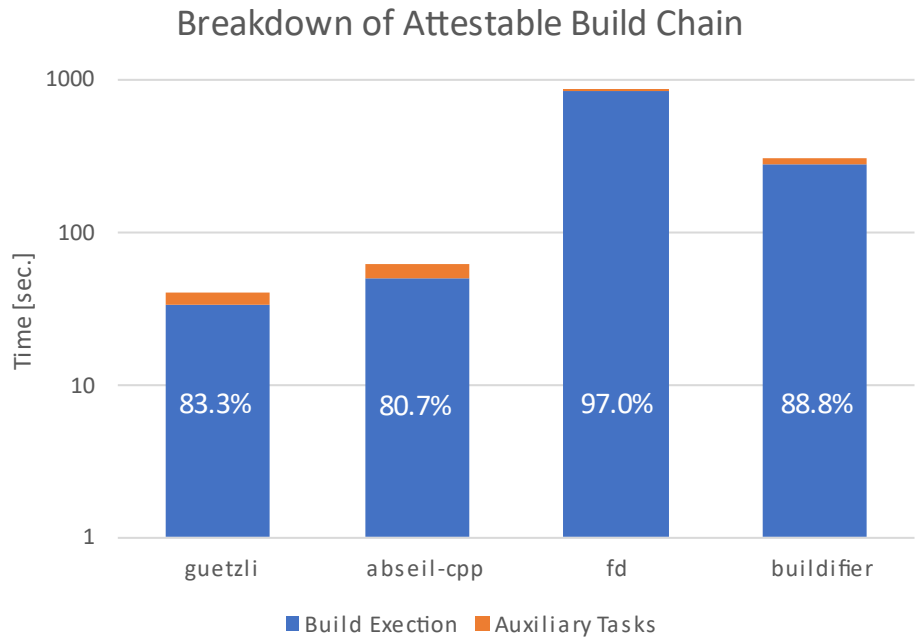
	guetzli	abseil-cpp	fd	buildifier
Language	C++	C++	Rust	Go
Original Build System	Bazel	Bazel	Cargo	Bazel
Project Files	88	1534	60	329

Evaluation : Performance

- Compared build times across three environments
  - SNP-disabled VM (Baseline), SNP-only CVM, and SNP-enabled CVM with IMA (Attestable Build Chain)

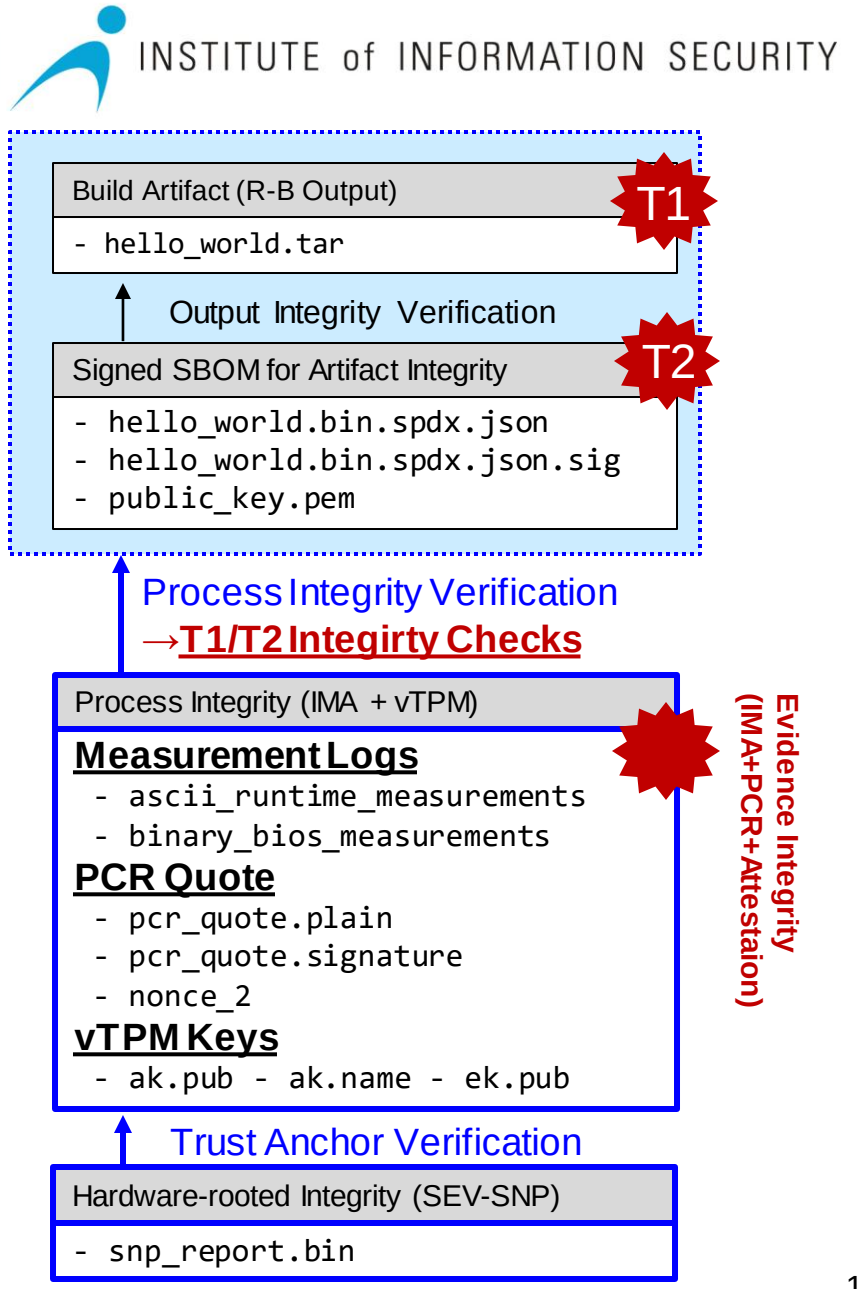
- Attestable Build Chain introduces additional build-time overhead.
  - Most time is still spent on actual build execution (80–97%),while the additional overhead mainly comes from extended IMA measurements for repeated and file-access monitoring.
  - The approach is better suited to release builds than frequent development builds.





Evaluation : Security

- Goal
  - Verify whether T1/T2 attacks lead to verification failures during Build Chain Report verification.
- T1: Untrusted Toolchain Execution
  - Compare hashes of executed binaries in the IMA log with registered reference values.
- T2: SBOM Tampering
  - Compare declared artifact/dependency hashes in the SBOM with file-access measurements in the IMA log.
- Evidence Integrity
  - IMA measurements are extended into vTPM PCRs and bound to SEV-SNP attestation.
- T1/T2 resulted in verification failures due to inconsistencies in execution evidence.



Related Work

- Existing approaches focus on specific aspects of software supply-chain trust:
  - Drexel et al. (Sicherheit 2024)
    - Verifies output equivalence, not actual build-time execution.
  - in-toto (USENIX Security 2019)
    - Verifies declared workflow metadata, not system-level execution.
  - Revelio (Middleware 2023)
    - Verifies environment integrity, not build-process integrity.
  - Attestable Builds (CCS 2025)
    - Uses user-level build reports, not system-level execution evidence.

Table. Comparison with Related Work *

Related Work	C1	C2	C3	C4	C5
Drexel et al. [46]	○	○	◐	◐	○
in-toto [47]	○	○	◐	●	○
Revelio [48]	●	●	○	○	○
Attestable Builds [20]	●	●	◐	○	●
Attestable Build Chain	●	●	●	●	●

●: Supported ◐ : Partially Supported ○: Not supported

C1: Execution Env. Protection **C2**: Root of Trust & RA
C3: Build Process Integrity **C4**: Provenance Transparency
C5: Verify w/o Rebuild

* Table excerpted from our paper.

Attestable Build Chain provides externally verifiable, system-level build execution evidence using Linux IMA, anchored in vTPM and SEV-SNP.

Discussion

- RQ1: Build Process Visibility
  - Actual file accesses and toolchain execution become externally observable via Linux IMA.
- RQ2: Integrity of Build Evidence
  - Kernel-level measurements (IMA) can be anchored to hardware-rooted attestation (SEV-SNP + vTPM).
- RQ3: Verification without Rebuilding
  - The Build Chain Report enables third-party verification of build-time execution without rerunning the build.

Future Directions and Takeaway

- Future Directions
  - Support additional build systems beyond Bazel.
  - Reduce noise in IMA measurements through refined policies, contextual information, and smaller TCBs.
  - Integrate with existing supply-chain ecosystems (e.g., SPDX, CycloneDX, OpenSSF S2C2F, MITRE SoT, Binary Transparency).
- Takeaway
  - Attestable Build Chain makes build-time execution externally verifiable; future work will improve evidence clarity and ecosystem integration.

Conclusion

- Reproducible Builds verify what was produced,
  but not how it was produced.

- Attestable Build Chain extends verification
  from artifact reproducibility to build-time execution.

- Our main Contributions
  1. Build-time execution visibility via Linux IMA
  2. Hardware-rooted evidence integrity via vTPM and SEV-SNP
  3. Prototype implementation and evaluation with real-world OSS projects