Google

Transparent UEFI

Confidential VM

dionnaglaze@

May 2024

Agenda

01 Baseline context

02 Current circumstances

03 Opaque signed reference values

04 Concrete road to transparency

05 Q&A

Google Compute Engine

Confidential VMs with

- AMD SEV-SNP (preview)
- Intel TDX (preview)

Confidential Space

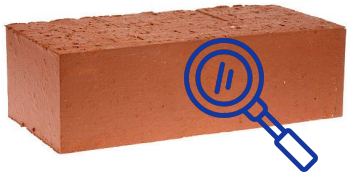
- Runs a single (measured) container.
- Attestation verification service for vTPM quote
- Secure Key Release for Split Trust Encryption Tool

Anyone

Google Cloud

Remote Attestation Diagram

UEFI
e8a4c56ceaa0b589112869f54a61e98eea155302d25bf0a1e3380dabfad480b19
2997d6036ef80c341c6f5e5a90046d2 👍🏼



Transparency is an **endorsement** of the reference value.

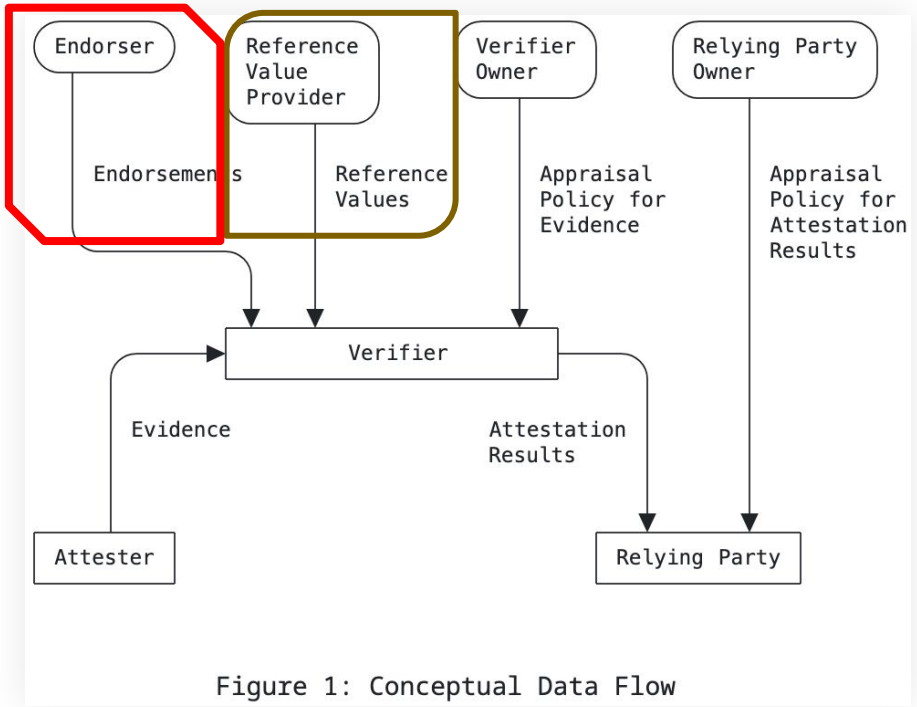


Figure 1: Conceptual Data Flow

Where are we now?

e8a4c56ceaa0b589112869f54a61e98eea155302d25bf0a1e3380dabfad480b192997d6036ef80c341c6f5e5a90046d2
Without the “👍”

- Sources in the google3 monorepo
- Submodules are all internally vendored
- Built with internal build infrastructure, not the upstream stuart_build tooling
- Toolchain is the internal build of clang managed by a toolchain team.
- SLSA L3 builds only Google can verify.
- The firmware you get at instance creation is whatever is currently deployed.

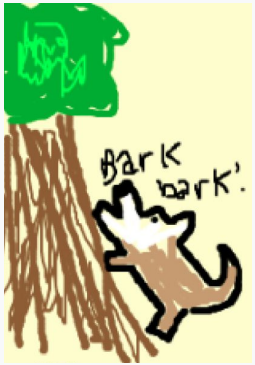
Very soon™ though, you’ll see a 👍 rooted to pki.goog/cloud_integrity/GCE-cc-tcb-root_1.crt

Format of the thumb isn’t CoRIM yet:
https://github.com/google/gce-tcb-verifier/blob/main/proto/endorsement.proto

Totally uncommitted roadmap

What I personally want to see happen, and might be able to make happen.

Compute Engine is a different team.
Virtual Firmware is a different team.
Supply chain integrity is a different team.
Assured Open Source is a different team.
Non-monorepo build service is a different team.



Transparency sliding scale

Project Oak has Transparent Releases:

- Known sources
- Known toolchain container
- Non-repudiable with Sigstore
- Uses GitHub actions

Very similar to “Why should I trust your code?” article in CACM.

This is my baseline. Transparency is necessary but not sufficient for security and trustworthiness.

Transparency Goals and Assured Open Source

- New policy: all open source binaries must be released through Assured Open Source (AOSS).
- AOSS has a managed Git host for quality assurance (e.g., large scale changes).
- Toolchain containers
  - Debian packages assured internally.
  - periodically updated
  - periodically analyzed

- Build service will use a simple toolchain container from Debian packages only, and run offline.
- Binaries are kept internal until qualified, and then published with attestations and reference values.
- **Transparency endorsement is the Verified Summary Attestation** (VSA) of the build.
  - A second VSA for the same artifact from another organization may be desired.
  - Proof of inclusion in append-only log may be desired (e.g., Sigstore).

One UEFI, Many Hurdles

The team that manages our EDK2 tree has the capacity for **ONE** UEFI.

Compute Engine	⇒	Separately release UEFI, run publish step in tooling	👍🏽
Virtual Firmware	⇒	Develop in Git, not monorepo	Convinced
Supply chain integrity	⇒	Attestation formats, ownership of toolchain	Hands off
Assured Open Source	⇒	Releasing process, ownership of toolchain	Defined
Non-monorepo build service	⇒	Onboard with new tool for building and prod-signing off-tree build.	👍🏽

~~Open Sourcing~~ Publishing UEFI

No contributions accepted, no capacity for community engagement.

Google uses BoringSSL, not OpenSSL.

Authenticode (PKCS#7) has been paravirtualized, but needs to be moved back to VM.

BoringSSL integration and GooglePkg/ ought to get pushed upstream with build config.

Internal Git testing hooks for running Compute Engine tests will require heavy lifting.

Security Embargos

We may have to patch internally and not push the changes publicly due to embargo agreements.

We'll still have signed reference values, but endorsed transparency wouldn't be guaranteed.

Should be rare, but can happen. Sorry.

Ulysses pact using time-locked encryption of the patches? Risky untested legal theory.

Transparent *Process*?

Even with transparent releases, you still get whatever firmware that's deployed.

Pro: Google keeps firmware updated.
Con: No ability to vet changed firmware before you land on it.

Pro: Google keeps PCR0 long term stable.
Con: PCR0 doesn't reflect the firmware bits, just firmware version.

Instead,

- offer choice of releases within the last year to pin in an instance
- releases published to a pub/sub topic
- forewarning allows us to have a standards-compliant PCR0: subscribers can check new versions and update TPM policies before updating

Reproducible Builds

Build attestations become less important with reproducible builds.
You just check everything yourself.

Note: Upstream EDK2 doesn't have a reproducible build.

We might be able to manage it for UEFI.
Maybe whole distributions can become reproducible.
Beyond that, reproducible builds are highly atypical.

What can we do?

SLSA Ln: attested builds

Intel researcher Marcela Melara has a proposal for SLSA to assure builds ran in a TEE.

General builds may be non-reproducible, but if the build service is trustworthy we don't care.

To trust the build service, we need

- Reproducible UEFI
- Reproducible distro
- Reproducible build launcher.
    - Measures inputs and outputs to PCRs for offline attestation verification.

The last is where there is maybe secret sauce, but maybe could be published.

(if the build service is not reproducible, then you get turtles all the way down)

Summary: Transparency is many things

- Source to binary unbroken chain
  - public sources, binaries, and SLSA provenances for offline auditing
  - watchdogs produce VSAs for attestation verifiers to make fast decisions.
  - SLSA L3 with higher on the horizon
- Continuous delivery with forewarning
  - Pub/sub connects security policy makers to the supply chain.