Image-Based Linux and TPMs

🔐

Measured Boot, Protecting Secrets and You

Lennart Poettering, LSG, MSFT

What’s an Image-Based Linux?

A system where the vendor data (binaries, libraries, data) are deployed via **coarse** file system images, instead of **fine-grained** packages.

Various benefits (discussed elsewhere)

Here we’ll focus on their relevance for building **trusted systems** with them.

DDIs FTW!

What’s a TPM?

A security module (either discrete chip, or in firmware, or in software), a “cryptographic co-processor”. (Typically slow, though.)

Standardized, widely available

Conceptually similar to SmartCards or FIDO keys

Supposedly temper-proof, can store at least one key securely.

That key cannot be extracted.

But you can ask the TPM to encrypt or decrypt data with it. In particular other keys (“wrapping”).

What’s a TPM? (Part 2)

May enforce policy under which conditions to allow decryption of data. Examples:

- PIN is provided
- System is in a specific state (e.g. runs specific firmware, boot loader, software; is in a specific boot phase; uses a specific disk encryption volume key, …) via *PCRs*
- It’s a specific time
- …

What’s a TPM? (part 3)

TPM also can do many other things (RNG, NV storage, Dictionary Attack protection…)

PCR Policies

Most interesting type of access policy for encrypted TPM objects: PCR policies

PCR = Platform Configuration Registers, of which there are 24 on a typical TPM.

Can be read at any time

Typically initially zero

Can only be updated from firmware/boot loader/OS with the following operation:

```
PCR[n]’ = SHA256(PCR[n] || data)
```

This operation is called “extension” of the PCR (or “measurement” of data)

How are PCRs used?

Firmware/Boot Loader/OS always measures the *next* code it runs right before running it. ("Measured Boot")

Final PCR[n] value is hence result of all code in the boot chain of code, if you change any element in the chain, final result will be different. And any code that runs can only manipulate subsequent PCR[n] values, but not the "past" (including its own).

If you know the elements of the chain ahead of time, you can pre-calculate the PCR values.

Use Case: allow decrypting keys via the TPM only if the system is in a very specific state, by binding decryption to a known-good PCR value (either specified literally, or indirectly by provided signature of PCR value matching a specified public key). Thus enforce that it runs the right software, is in the right phase of the boot, has the right identity info (i.e. `/etc/machine-id`, FS identity, root FS volume key, …)

Measured Boot and Image-Based Linux

Image-Based OSes have the benefit that images can be measures as a whole, and are deterministically the same everywhere. Package-based OSes are systematically different, as every individual install will differ in packages, timestamps, inode numbers, …

If measurements of OS image are deterministic and always the same, the resulting hashes are stable, and can thus be signed by the OS vendor ahead of time.

TPM Infrastructure for Image-Based OS in systemd

`systemd-stub` (UEFI PE stub glued in front of a Linux kernel that runs in UEFI mode, and then passes control to the actual kernel to switch into Linux mode) will measure its own components into PCR 11. Also measures configuration parameters (kernel cmdline) into PCR 12.

`systemd-pcrphase@.service` measures short strings indicating boot phases into PCR 11 at various stages of the boot process

`systemd-pcrfs@.service` measures FS identity (UUID, …) into PCR 15 on mount

`systemd-pcrmachine.service` measures system identity (/etc/machine-id) into PCR 15 during early boot

`systemd-cryptsetup@.service` measures volume key into PCR 15 during boot

Future: `systemd-veritysetup@.service` measures Verity root hash into PCR 11 during boot

TPM Infrastructure (continued)

`systemd-cryptsetup@.service` can unlock LUKS volumes with keys encrypted by TPM (“enrolled”), bound to PCR policies.

`systemd-creds` and the service manager can unlock “credentials” (i.e. secrets, private keys, passwords, …) with PCR policies

`systemd-measure` can pre-calculate expected PCR values for UKIs (and sign them)

systemd’s `ukify` may be used to build UKIs that carry PCR signatures, and `systemd-stub`

Net Result

PCR[11] → The OS images are measured into (OS UKI and OS DDI)

PCR[12] → The local configuration for the kernel is measured into

PCR[15] → The identity of the local system is measured to.

Usage: via `systemd-cryptenroll` lock disk against TPM, so that:

- May be unlocked only with physical access to the TPM, i.e. local hardware
- May be unlocked only if booted with properly signed OS UKIs and OS DDIs by vendor
- May be unlocked only if configuration matches what user picked earlier
- May only be unlocked during specific boot phase (e.g. from initrd)
- May only be unlocked if a user PIN is supplied
- Will be rate-limited if attempted too often.

Outlook

PCRs have other uses:

- Remote attestation: e.g. system can prove to you that it is in order, before you log in. i.e. that it runs the right unmodified software that you picked, and is the right system, and is properly booted up and so on.
- Configuration images that can be overlaid on top of /etc/ and can only be decrypted on specific hardware if it runs the right software, in the right boot phase, within a specific timeframe
- Credentials (short bits of information) that carry secrets/credentials in a secure way that can only be unlocked locally.

Measured Boot vs. SecureBoot

SecureBoot implements a cryptographic “allowlist” of code that may run on PCs. SecureBoot is typically very imprecise (i.e. the allowlist is huge). Its effect is on the future and has a strong active implications (i.e. blocks boot if need be to continue). Primarily covers code.

Measured Boot implements a cryptographic “allowlist” of conditions that must be met before local secrets can be unlocked. It’s usually very focussed on specific state/software/identity, its immediate effect is passive/history-focussed, has weak implications (i.e. only restricts access to specific secrets, but any code may run and can talk to it and try their luck). Can cover code, configuration, identity, …

Best used in combination.

That’s all!