Improving security of the FreeBSD boot process

Kornel Dulęba
Michał Stanek
Presentation plan

- Secure Boot 101
- Secure Boot implementation in UEFI
- FreeBSD veriexec and libsecureboot
- TPM overview
-Measured boot
- Strongswan with TPM
Presentation plan

- **Secure Boot 101**
  - Secure Boot implementation in UEFI
  - FreeBSD veriexec and libsecureboot
  - TPM overview
  - Measured boot
  - Strongswan with TPM
Secure Boot 101

- Purpose - allow only authenticated FW and OS to run
- Defense against rootkits, persistent malware, etc.
- Chain of Trust - each boot image verifies the next, and so on
- Pass execution to next boot image only it verifies OK
- First boot image is immutable (in ROM) - inherently trusted
Secure Boot 101

- First image (BootROM) is Root of Trust for the Secure Boot chain
- Root of Trust public key - needs to be protected from modification
- RoT key often burned in fuses, OTP, or ROM (or TPM)
Source: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf
Presentation plan

- Secure Boot 101
- **Secure Boot implementation in UEFI**
- FreeBSD veriexec and libsecureboot
- TPM overview
- Measured boot
- Strongswan with TPM
Secure Boot in UEFI

- UEFI uses Microsoft’s PE/COFF format for binaries
- PKCS#7 formatted signatures are embedded in the binary
- This format is supported by very few cryptographic libraries.
- The most common open source UEFI implementation - EDK2 is compiled with OpenSSL

Source: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf
Source: https://uefi.org/sites/default/files/resources/UEFI_Spec_2_7.pdf
Secure Boot in UEFI

Crucial UEFI variables:

- DB - Database of allowed certificates (for verification)
- DBx - Database of forbidden certificates
- PK - Platform Key, highest at key hierarchy
- KEK - Key Exchange Key, updates to KEK must be signed with PK
- DB/DBx updates must be signed with KEK or PK
- Possibility to whitelist/blacklist specific firmware hashes (no certs)
Hash image

Is digest in DBx?  
↓
NO

Fail

Is digest in DB?  
↓
NO

Do we trust the signer?  
↓
NO

Does our digest match the one in signature?  
↓
YES

Succeed

YES

NO
Presentation plan

- Secure Boot 101
- Secure Boot implementation in UEFI
- FreeBSD veriexec and libsecureboot
- TPM overview
- Measured boot
- Strongswan with TPM
Juniper created veriexec for Junos OS.

Available in FreeBSD HEAD since February’19.

It uses a manifest as a database of trusted components.

Prevents executing untrusted kernel, binaries, scripts

Integrity check hooks at execve and other critical points
Veriexec manifest

- Single file composed of entries in path + hash form.
- All of these are loaded into a metadata store, using path as key.
- When a file is loaded, search for its hash in the store.
- If an entry is found and corresponding hash doesn’t match - fail.
- There are different policies for loading kernel and other files (eg. config files) when no entry is found.
Verifying the manifest

- Broken chain of trust!
- How to verify the manifest itself?
Verifying the manifest

- Manifest file stored together with its signature
- Trusted public keys may be embedded in the loader
- But we could use UEFI trust anchors for manifest verification
- Loader has access to DB/DBX UEFI variables
- We picked BearSSL - lightweight crypto library to use in the loader
- Library with all the verification API - libsecureboot
- Still, embedded data may be used for systems without UEFI
Presentation plan

- Secure Boot 101
- Secure Boot implementation in UEFI
- FreeBSD veriexec and libsecureboot
- **TPM overview**
- Measured boot
- Strongswan with TPM
TPM 1.2 driver added in FreeBSD 8.2 (bsssd project)
TPM 2.0 driver added by Semihalf in Dec 2018
CRB and FIFO (TIS) modes supported
LPC bus only (no I2C/SPI support)
Tested with Infineon SLB9665 TPM
TPM overview

- Trusted Platform Module - a specification by TCG
- Versatile, low-cost HSM device
- Usually a dedicated hardware chip
- Ensures integrity (trustworthiness) of a platform

Features:
- Measured Boot
- Secure storage (with authorization)
- Secure key generation
- HW RNG
- Crypto operations (slow!) - RSA, ECC, AES, SHA, HMAC
TPM history

- **v1.1 (2003)** - now deprecated
- **v1.2 (2005-2009)**
  - anonymous attestation (DAA)
  - anti-hammering (prevent dictionary attacks)
  - limited crypto (SHA-1 only, RSA-2096, no ECC, AES optional)
- **v2.0 (2014-2018)**
  - algorithm agility (only max key/hash length defined)
  - Enhanced Authorization - complex object access rules
  - not backwards compatible!
Firmware TPM

- fTPM - TPM implemented in firmware
- Must run in TEE to make sense (ARM TrustZone, SGX)
- Used in millions of mobile devices with TrustZone
- Much faster than discrete TPM - runs on main CPU
- fTPM also in Intel ME, AMD PSP (check your BIOS)
TPM use cases

- Not just for enterprise!
- Remote attestation - proof of platform/boot integrity
  - (somewhat) proves system is rootkit-free
- 2FA, smart card (GPG) - sign with key embedded in TPM
  - private key never leaves the TPM
- IPSEC VPN hardening - sign IKE payloads with TPM
- MS Bitlocker / LUKS key storage (no GELI support yet...)
  - anti-hammering - TPM locks down on failed attempts
- Securely store root certificates/keys (prevent modification)
- HWRNG entropy for the OS (early boot, embedded systems)
TPM authorization

- Enhanced Authorization in TPM 2.0 allows complex rules
- Each NVRAM object has separate access rules
- Combine multiple rules with AND/OR
- Authorization policies:
  - Password
  - PIN
  - HMAC
  - PCR state (platform/boot integrity)
  - Physical presence (press key, assert pin, access BIOS)
  - Counters, time limits
TPM caveats

- Anonymity concerns - mostly fixed with TPM 1.2 attestation (DAA)
- DRM concerns - Trusted Computing in general (SGX, Intel ME)
- Discrete TPMs are slow
- Different pinout/pin pitch configurations
- Complex, hard to read spec - 2 versions
- Poor SW support, especially for 2.0
- Hard to use correctly:
  - Bus encryption optional (need PSK)
  - ACPI reset vulnerabilities (PCRs cleared)
  - Need to update TPM FW manually (do it!)
Presentation plan

- Secure Boot 101
- Secure Boot implementation in UEFI
- FreeBSD veriexec and libsecureboot
- TPM overview
- Measured boot
- Strongswan with TPM
Measured Boot

- Machine state represented by PCRs - Platform Configuration Registers containing cryptographic hashes.
- PCRs can be updated ("Extend" operation) by supplying another hash, but no direct modification is allowed.
- newPCR = HASH(oldPCR || dataToExtend)
- PCRs can only be reset by hardware reset (important)

Measured Boot

- Use PCRs to measure critical components, and if the resulting hashes are incorrect take appropriate action.
- Inconvenient for management - updating measured part of the system forces a change in the verification software.
- On the other hand, Secure Boot only requires user to sign the updated component.
RTM

Firmware

Boot Loader

OS

Applications

PCR0

PCR1

....

PCR24
On each measurement UEFI updates an event log with object names (file paths) and digests used for the Extend operation.

One can later compare the log entries against a database of expected values.

Software can replay the extend operations and confirm log authenticity against signed PCR values. (Quote operation)
● Currently FreeBSD can’t extend PCRs on its own.
● UEFI measures every binary before passing execution to it - boot1.efi and loader.efi are included in measurements already.
● Loader could be extended to measure kernel and modules too
Presentation plan

- Secure Boot 101
- Secure Boot implementation in UEFI
- FreeBSD veriexec and libsecureboot
- TPM overview
- Measured Boot
- **Strongswan with TPM**
Strongswan

- Strongswan is an open source multiplatform IPSEC implementation.
- Tunnels can be established using Internet Key Exchange (IKE) protocol.
- Authentication can be based on certificates or PSK. (Pre-shared key)
- In this case authentication payload is signed with private part of the key bound to certificate.
Strongswan - IKE

Signing request with proper authentication

Encrypted digest

private key

TPM
Prerequisites

- TPM 2.0 FreeBSD driver
- IBM TSS, a userspace library that can “talk” to the TPM.
- Only a small, one-line patch is needed to make it build on FreeBSD.
- Our patch has not yet been merged on IBM TSS Sourceforge.
- Strongswan patched to work with IBM TSS - pull request is up on Github.
Strongswan

- Strongswan can use private keys stored inside a TPM.
- That key is bound with a certificate to be used during IKE.
- Access protected with a passphrase, either be stored in clear text in configuration file or prompted for.
- Private keys are not leaked even if machine is compromised.
- A discrete TPM is slow, on Infineon SLB9655 signing takes ~0.15s using RSA2048 key.
An excerpt from swanctl config file that links a private key from TPM with a certificate.
Acknowledgements

- Stormshield - initiators and sponsors of entire research and development
- Simon J. Gerraty (sjg@) and Juniper - for fruitful and vivid cooperation around the libsecureboot and Veriexec
Questions