Hardware-based trust and integrity verification

Federico Mancini
Federico.Mancini@ffi.no

High Integrity Day, Bergen 11.02.2014
Trust and assurance

We say that a system is **trusted** if it is relied upon to a certain extent to enforce a specified security policy.

To trust the system we need some **assurance** that the security mechanisms used to enforce the policy are correctly implemented and provided under all relevant circumstances (i.e., it is trustworthy).

**Common Criteria**

- System to evaluate (evaluation target)
- Protection profile (properties the system needs to fulfill)
- Evaluation process for assurance level (EAL)
EAL levels

- EAL1: Functionally Tested
- EAL2: Structurally Tested
- EAL3: Methodically Tested and Checked
- EAL4: Methodically Designed, Tested and Reviewed
- EAL5: Semi-formally Designed and Tested
- EAL6: Semi-formally Verified Design and Tested
- EAL7: Formally Verified Design and Tested

High-Assurance

PROBLEM

High-assurance levels are very difficult to obtain, very time consuming and expensive. The resulting system is an investment that must last many years to amortize the cost, and it is impossible to upgrade or modify because it would break the certification: a black box. Hence, it is often difficult to integrate in an existing infrastructure and manage it.
Trust = Certification + Integrity Protection

Given that we have a high-assurance system, how does a third party establish trust in it?

The certificate says that the system has been certified, but not the integrity in the meanwhile it has not been compromised, i.e., that its integrity is preserved.

We trust it because we rely on its integrity protection mechanisms, which are a requirement to achieve certification (temper-responsive and temper proof devices for instance, and formally verified software).
Integrity verification

Not all systems need to meet very high-assurance requirements, but maybe they need to have a specific configuration to be allowed to execute certain operations (e.g., a certain OS update with the last patches, or the latest BIOS version, or hardware with signed firmware)

How do we verify their configuration in a trustworthy manner without having to certify each machine every time the configuration changes?

We need an independent entity that can measure and report the system integrity in a trustworthy manner. Then only this part of the system would need to be certified.
Integrity measuring and reporting

How can I trust you?

You answer

Measure

I have been certified for high-assurance and this is what is running on this machine

Check

DB
The trusted platform module - TPM

- The **TPM (Trusted Platform Module)** is both a set of specifications and its implementation.
- The TPM is a *passive device* (it can only perform actions if asked to), soldered to the motherboard, that can be used to perform some cryptographic operations in a protected environment.

- **Main Goal**: increase trust in a platform
TPM main functionalities

• Better cryptographic services:
  – Hardware protected crypto operations
  – Hardware protected data encryption
  – Hardware protection against password guessing

• New functionalities:
  – Platform integrity protection (Trusted Boot)
  – Platform Attestation
  – Sealing
  – Anonimity

≈ Smart cards
Inside a TPM

- Endorsment Key (EK)
- Storage Root Key (SRK)
- Certificates
- Policies/AuthData
- RNG
- RSA Engine
- SHA-1 Engine
- Key Generation
- Program Code
- Exec Engine
- Opt-In
- PCR 1
- PCR 8
- PCR 16
- PCR 17
- PCR 21
- PCR 24

CRYPTO, NVRAM, OTHER
TPM - CRYPTO

- **RNG**: (True) Random Number Generator
- **SHA-1 Engine**: To compute hashes
- **RSA Engine**: For encryption, decryption and signing with asymmetric keys.
- **Key generator**: To generate RSA key pairs and symmetric keys.
Non-Volatile Memory

- **EK**: installed by manufacturer. Unique per TPM.
- **SRK**: created when user takes ownership of TPM.
- **Certificates**: Manufacturer, Conformance Entity, Validation Entity, Trusted Third Party
- **Policies/AuthData**: shared secret to access objects (authentication+authorization)
Platform Configuration Registers - PCR

- 20 bytes registers to store SHA-1 hashes.
- Cannot be written directly, only extended: \( PCR = SHA-1(Current\ value \ || \ new\ hash) \)
- 1-8 reserved. At least 16 must be present.
- They are always reset at boot time and only then.
What can you do with a TPM?

• Trusted Boot
• Secure Storage
• Remote Attestation
Trusted Boot

Each component involved in the boot process is measured, and the measurement stored both in the TPM PCRs and in a Log File.
PCR values can be used to verify the integrity of the log file

- Why should we trust the PCR values?
- What if a malware was installed that stored fake measurements?
- Who measured the system?
Where does the trust come from?

CORRECT IMPLEMENTATION
ASSURANCE Lv 4
TEMPER PROOF
?
ROOT OF TRUST FOR MEASUREMENTS

CORRECT FILE MEASUREMENTS

TPM PROTECTION

PCR
324HIAS23408ADFI
INR86943UE83FOOQ
N356SDDW654SD
DS654SD97PHJD

OS

BIOS
MBR
BOOT LOADER

MEASURE
RUN

MEASURE
RUN

MEASURE
RUN

MEASURE
RUN
Root of Trust for Measurement

CORE ROOT OF TRUST FOR MEASUREMENT

CRTC

MEASURE

RUN

BIOS

MEASURE

RUN

MBR

MEASURE

RUN

BOOT LOADER

MEASURE

RUN

OS

PCRs

324HIA23408ADFI

INR89403UE83FOQ

N356SDDW654SD

DS654SD97PHJD
Root of Trust for Measurement

Guarantee that there is always a component that will measure the malware
Trusted boot and attestation

The computer has started and everything has been measured. This is going to happen no matter the measurements. So what now? Who verifies the integrity?

Prove to a third party the integrity of your platform: Remote Attestation.

Seal computer to current configuration so that it is unusable if someone tamper with it: BitLocker.
Attestation protocol: Root of trust for Reporting

SML

DB with valid configurations

1: RequestPlatformConfiguration()

2: GetEventLog()

3: GetSignedPCR()

SignedPCR()

4: SignPCRValue()

5: GetPlatformCredentials()

PlatformCredentials()

6: ValidatePlatformConfiguration()

Proof that PCR are genuine
Sealing/Binding

TRUSTED BOOT

NVRAM

TPM 2048 RSA KEY (Private)

PCR[1,2,3…]  
USER PASSWORD  
SYMMETRIC KEY

TPM 2048 RSA KEY

PCR[1,2,3…]  
USER PASSWORD  
SYMMETRIC KEY

DATA

USER DATA

ENCryption SOFTWARE
Problems

• We cannot say much about what happens «after» the OS takes control
• We need to maintain a potentially huge database of valid platform configurations
• We need an infrastructure parallel to PKI to manage TPM certificates
Protecting critical applications

- **Trusted Execution Environment (TEE) and Dynamic Root of Trust:** A secure and sanitized environment is created in hardware on the fly in order to run code securely, even if the system is compromised. TPM can be used to attest that code was securely run. Implementations:
  - Intel TXT
  - AMD-V
  - ARM TrustZone

- **Separation Kernel/MILS:** A secure separation kernel or hyper-visor is securely loaded with trusted boot, and different security domains are run in parallel. One domain is dedicated to TPM operations, so that the user or other processes cannot interfere.
1. Stop other processors
2. Mask all external events
3. Validate SINIT AC signature
4. Reset PCR 17-20 to a special value and extend 17 with SINIT hash
5. Unlock the chipset, load the SINIT AC and pass control to it

1. Test Hardware configuration
2. Initialize SMM handling
3. Enable DMA handling
4. Load and measure MLE
5. Store MLE hash in TPM
6. Pass control to MLE
Example of MLE: tboot and OSLO [8]

Static root of trust and trusted boot:
Must trust BIOS
Possible only at booting time

Dynamic root of trust:
BIOS out of trusted base, can be executed at any moment

TPM
- PCR 21
- PCR 18
- PCR 17
- PCR 7
- PCR 1
- PCR 0

S-CRTM → BIOS → BOOTLOADER

SENDER → INTEL TXT → tboot → XEN
SKINIT → AMD-V
OSLO → kernel
Problems with DRTM

• It is designed mainly as support to virtualization, i.e., to securely launch hypervisors like XEN.
• It has been already broken in various ways (together with XEN itself)
• We still need to trust the Hypervisor/Kernel/OS to execute our application securely at runtime.
ARM TrustZone

- Normal world
  - Normal world user mode
  - Normal world privileged modes
- Secure world
  - Secure world user mode
  - Secure world privileged modes
- Monitor mode

NORMAL WORLD MEMORY | SECURE WORLD MEMORY
Complete picture: TPM + INTEL TXT

- CRTM
- BIOS
- MBR
- BOOT LOADER
- OS
- TEE
- CRITICAL APP
- UNTRUST ED APPS
Complete picture: ARM TRUSTZONE
Conclusions

• TPM and DRTM can help greatly to increase trust in commodity PC and servers
• Huge growth in the last year (1 billion TPM deployed)
• Recently approved by NSA for use in public offices to improve security
• Tighter integration with commercial OSes (Windows 8, Chrome OS)
• Still a lot of problems to fix, but great potential