TrenchBoot: Open DRTM implementation for AMD platforms

OSFC 2019

Piotr Król

3MDEB

About me

Piotr Król
Founder & Embedded Systems Consultant

- open-source firmware
- platform security
- trusted computing

@pietrushnic
piotr.krol@3mdeb.com
linkedin.com/in/krolpiotr
facebook.com/piotr.krol.756859

Agenda

- Introduction
- Terminology
- S-RTM vs D-RTM holy war
- TCG D-RTM
- Implementations
- Memory protection
- TrenchBoot and coreboot
- Demo
- Known issues
- Further improvements
- How to help

Goal

implement TrenchBoot support for AMD platforms

Motivation

- to provide open toolbox that can build reasonably secure AMD systems
- to improve trusted computing and platform security-fu

Terminology

Root of Trust (RoT) - component performing security-specific functions measurement/storage/reporting/verification/update

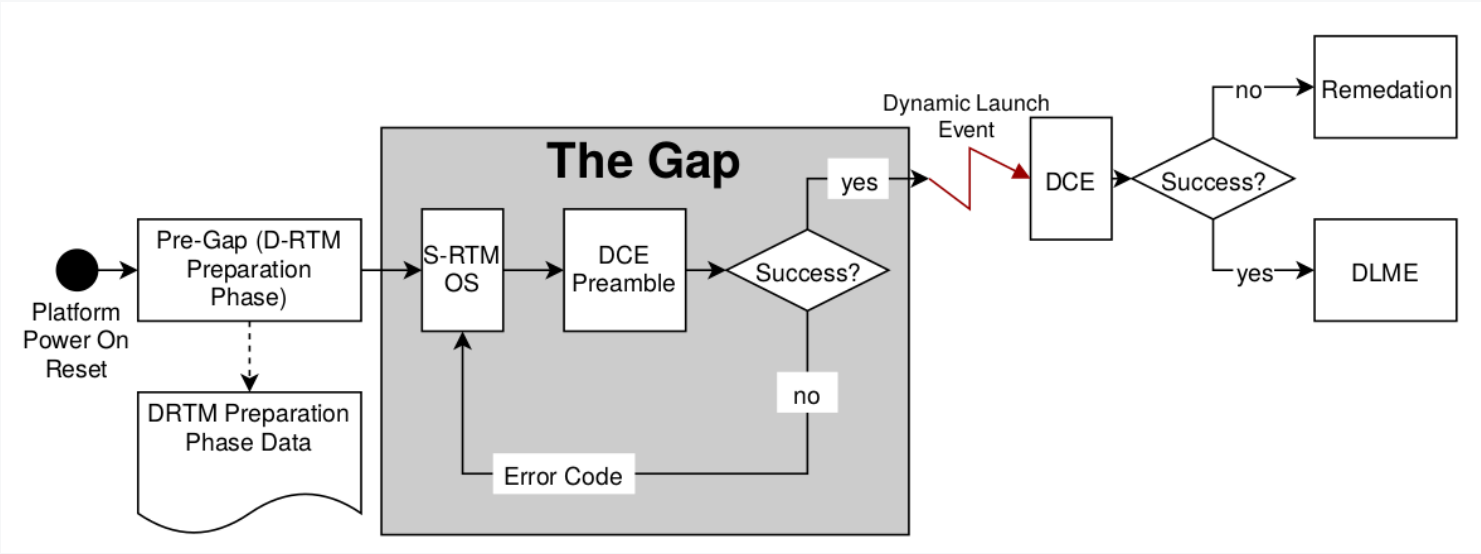
- RTM - An RoT that makes the initial integrity measurement, and adds it to a tamper-resistant log
- CRTM - the instructions executed by the platform when it acts as the RTM (Code, formerly Core)
- SRTM - provide chain of measurements for components that execute on the platform
- DRTM - platform-dependent function that initializes the state of the platform and provides a new instance of a RTM without rebooting the platform. The initial state establishes a minimal Trusted Computing Base.

Trusted Computing Base (TCB) self-defending platform security mechanism (HW, SW, FW)

- TCG Glossary v1.1 rev 1.0 and TCG D-RTM Architecture v1.0.0

S-RTM vs D-RTM holy war

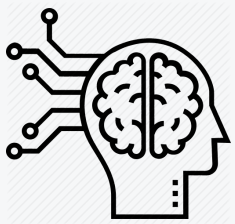
- *Static Root of Trust for Measurement (S-RTM)*
  - requires platform reset to establish trusted state
  - hard to update (re-evaluation of PCR values)
  - vendor specific (NXP HAB, Intel Secure Boot/Boot Guard, AMD HVB) requires proprietary tools and NDAs
  - well-established in IBV environment
  - there are open implementations based on coreboot and vboot (Chromebook, PC Engines)
- *Dynamic Root of Trust for Measurement (D-RTM)*
  - works even in compromised environment
  - avoid platform reboot to establish secure state
  - avoid problem with secure re-evaluation of PCR values
  - small enough to pass common criteria certification
  - open implementation coming to GRUB ad Linux kernel



- DRTM start when Dynamic Launch event call executes
- DL Event controls PCRs 17-22, those are initialized with value -1
- DL Event change PCRs value to 0 and immediately extends with DCE hash
- Any attempt to reset TPM will set PCR[17] to -1 (TPM reset attack immunity)

TCG D-RTM Architecture v1.0.0

Implementations



- Intel Trusted Execution Technology (TXT)
  - out of scope of this presentation and associated paper
  - uses, so far, proprietary ACM BIOS module
  - leverages `GETSEC[SENTER]` and other leaf function
- AMD Secure Startup
  - it is possible to create fully open-source implementation
  - leverages `SKINIT` instruction introduced with the AMD-V extension
- Qualcomm - rumours about IP core based on info from Xen Summit
- Both Intel and AMD implementations are compatible with TCG specification
  - competing implementation and TCG specification cause terminology mess

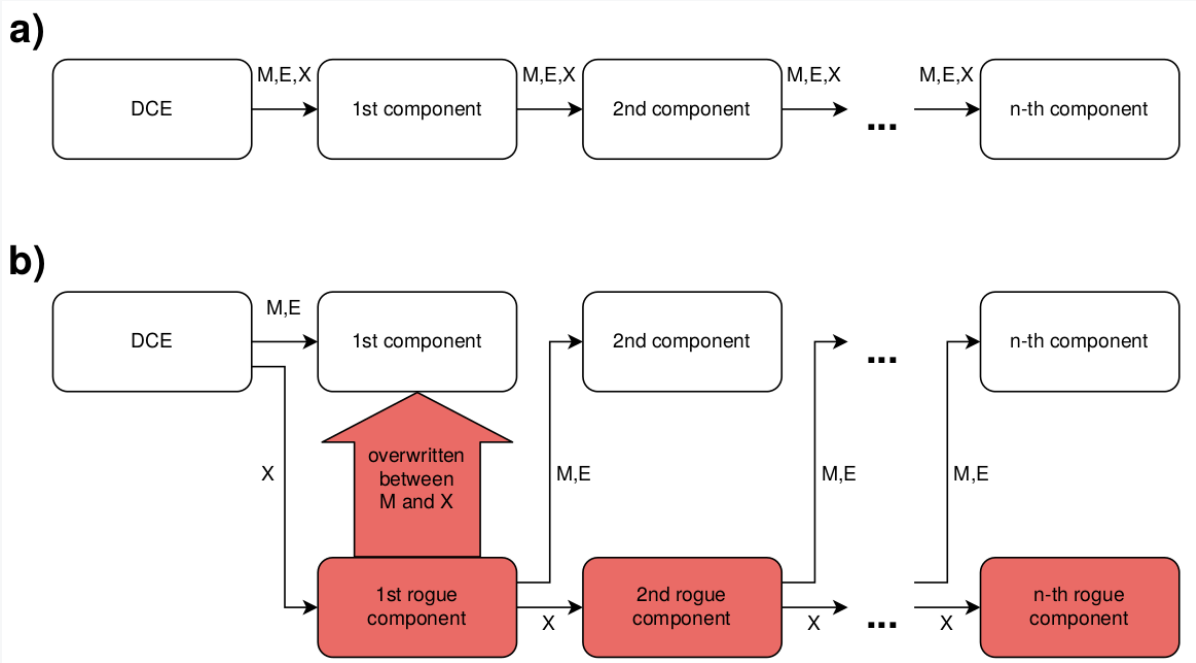
Terminology untangled

TCG	AMD	TrenchBoot
The Gap	non-trusted mode	N/A
Dynamic Launch (DL) Event	SKINIT instruction	secure launch
DMA protections	Device Exclusion Vector (DEV)	N/A[1]
DRTM Configuration Environment (DCE)	Secure Loader Block (SLB)	Landing Zone (LZ) code + data
N/A[2]	Secure Launch (SL) Image	Landing Zone (LZ) code
Dynamically Launched Measured Environment (DLME)	Secure Kernel (SK)	kernel
DCE Preamble	N/A	slaunch module

[1] uses the same name as vendor
[2] not distinguishable part of DCE

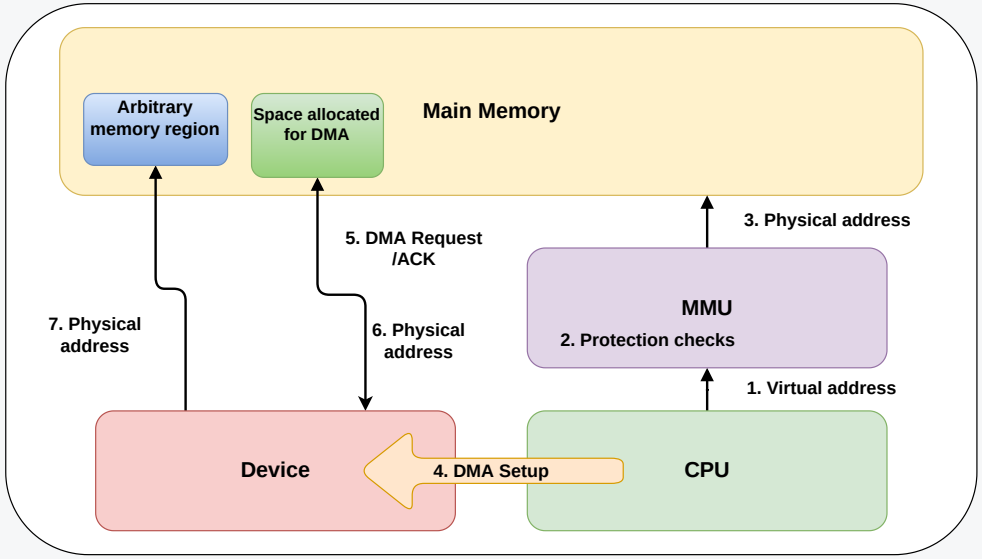
- More to that ARM has its own terminology
- TrenchBoot community have to work to establish correct names
- No problem for people working with D-RTM long time
- Huge communication issue for newbies

Memory protection



- DRTM architecture must ensure that memory containing code cannot be changed by anything outside of TCB, 2 most concerning mechanisms are:
  - DMA
  - SMM

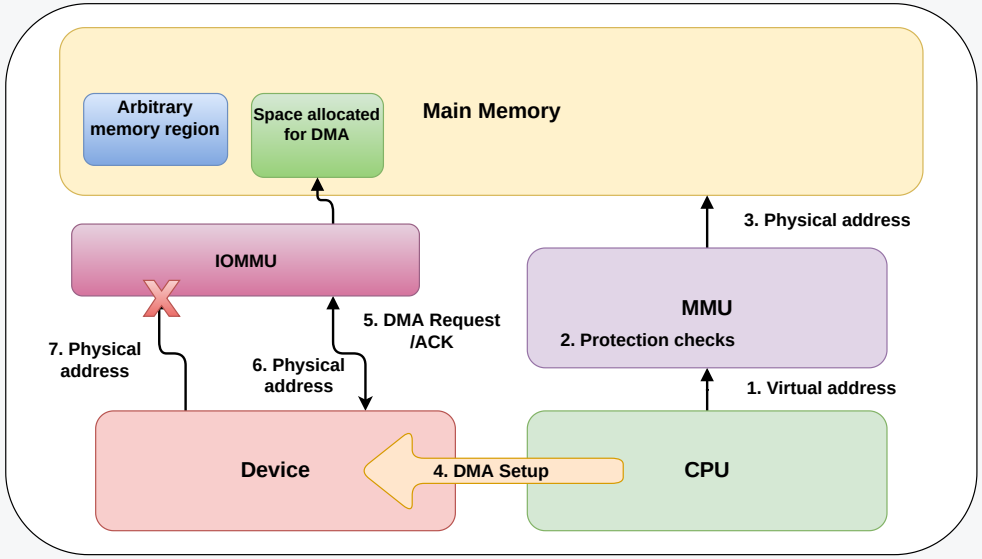
DMA



- DMA access can be blocked by: **exclusion** or **translation**
- Intel VT-d implements **exclusion** using DMA Protected Range (DPR) and Protected Memory Regions (PMR)
- AMD uses Device Exclusion Vector (DEV)
    - continuous array of bits in physical memory (each 4Kbyte)

image: "How to enable IOMMU in coreboot", Piotr Król

DMA



- Translation is a part of IOMMU (AMD) or VT-d DMA remapping (Intel)
- Generally more complicated then **exclusion**, care must be taken to not redirect to range that is or will be TCB
- D-RTM typically use **exclusion**

image: "How to enable IOMMU in coreboot", Piotr Król

SMM

The S-RTM SHALL place the SMM code in SMRAM and lock it. The DCE SHALL consider SMM to be "correct by construction." The SMM implementation SHALL successfully maintain its code and data integrity indefinitely.

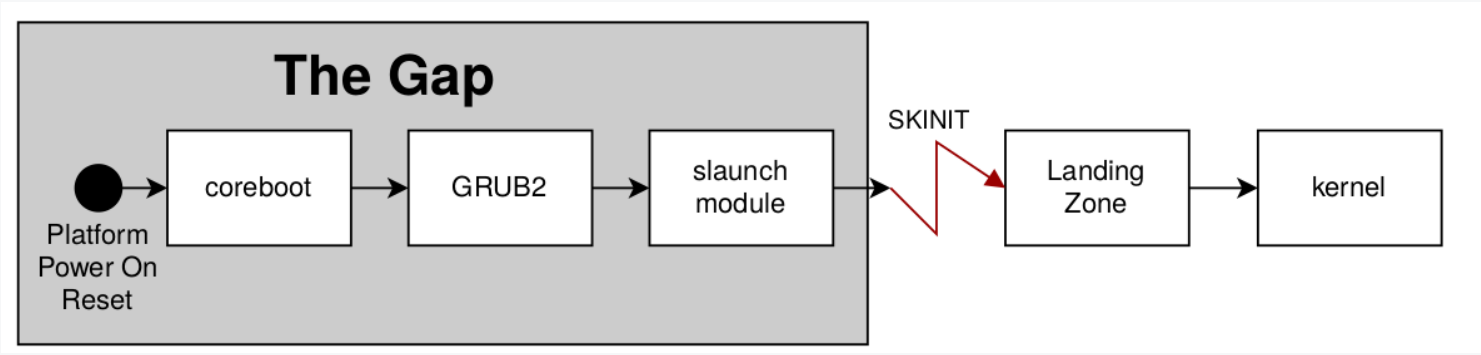
- One of the most problematics areas of D-RTM
- IBV should ensure safety of the SMM, can we relay on that?
- SMM cannot be validated after lock
- SMM code in open-source firmware can be audited, SRTM can measure it and hash can be verified by DCE
- Even if SRTM helps, ring-0 can change SMRAM content, what was proven couple times
- The only viable path on horizon is SMI Transfer Monitor (STM)
    - AMD version of it would be interesting topic

TCG D-RTM Architecture v1.0.0

Other D-RTM implementations

- OSLO
- Flicker
- Soft Cards
- GE implementation
    - there was try to merge it in tboot

TrenchBoot and coreboot




```
linux path/to/bzImage (...)
slaunch skinit
slaunch_module (cbfsdisk)/lz
boot
```

- load Linux kernel
- hook into `boot` command so loaded kernel will be executed using D-RTM by Landing Zone code
- load Landing Zone code to memory
- execute AMD Secure Startup

Demo time

Known issues

- At the time of writing SKINIT cannot correctly extend PCR17
  - we work with AMD to solve that issue
  - hard to debug
  - can be problem of LPC TPM that we use or TPM code instead of CPU

Future improvements

- Address SMM/SMI problems
  - STM implementation for AMD
- Add ACPI tables validation
- Add code to create DRT (DRTM Resource Table) ACPI table
- Support different OSes and hypervisors
- Support different payloads e.g. iPXE, SeaBIOS
- Support OE/Yocto for building components

Where to go from here?

- Still WIP, but whole code is accessible on GitHub
  - we will publish blog post and documentation to present how to start
- Source code is spread across various repositories and projects, we trying to gather everything under TrenchBoot GitHub account
- There is Google Groups mailing list for discussion

Q&A