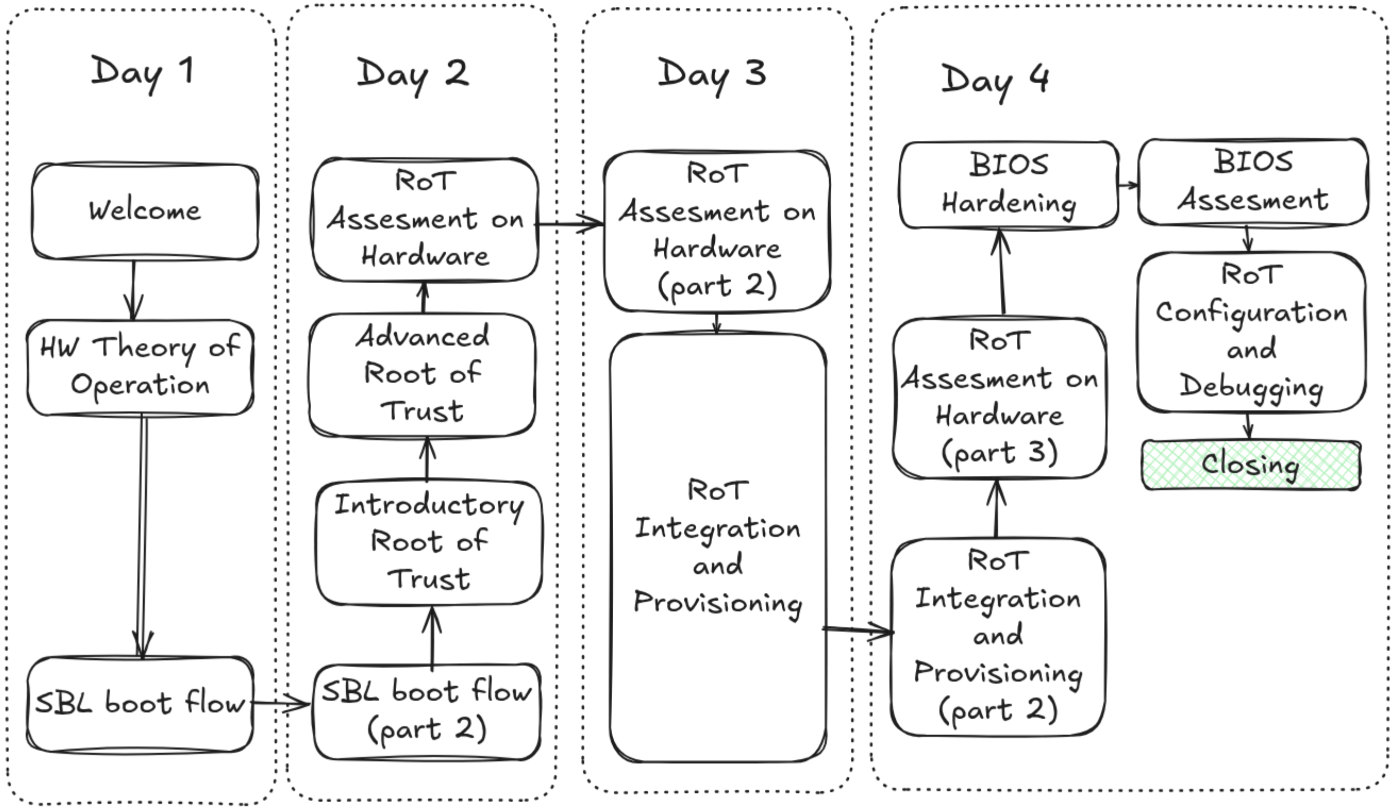
DS09SBL: Dasharo TrustRoot Training

DS09SBL Closing Remarks

Dasharo TrustRoot Training

Where we are in the course



Day 1
Welcome
HW Theory of Operation
SBL boot flow
Day 2
SBL boot flow (part 2)
Introductory Root of Trust
Advanced Root of Trust
RoT Assesment on Hardware
Day 3
RoT Assesment on Hardware (part 2)
RoT Integration and Provisioning
Day 4
RoT Integration and Provisioning (part 2)
RoT Assesment on Hardware (part 3)
BIOS Hardening
BIOS Assesment
RoT Configuration and Debugging
Closing

Goals of the Presentation

In this presentation, we aim to:

- Address remaining questions form day 2.
- Summarize plan vs execution.
- Feedback survey.
- Potential follow up.

Day 2 questions

Question

TPM init fails after reboot into freshly flashed Dasharo (SBL+UEFI) v0.9.0, what's the reason?

Flashing ME is to blame. A power cycle is needed for ME to initialize properly and for TPM to start working. Possibly the global reset requested by FSP already does this.

Question

Boot time optimization could be subject of future course.

Boot time will be long on debug binaries. What can be optimized is disabling security: measured boot, verified boot, and BTG. In addition, one should enable the fast boot option in SBL in the Python BoardConfig scripts. Unfortunately, one must expect that display won't work in firmware in such a setup.

Question

Will release version also show stage2 verification failure?

Yes, but no detailed hex dumps will be presented just short information about security violation.

Question

For sbl release build OSloader, can you enter the shell? Is `mm` there – if so it's a security risk.

OS Loader has its own shell Yes, there is `mm`. Currently, in the UEFI Payload we also have Shell, we haven't disabled it yet. In fact, in the UEFI payload we don't even have UEFI Secure Boot, because of #1485, which explain lack of SMM support.

Question

What is the real priority of UART debug versus stitch data and other settings? One of the students noticed that if you set different UARTs in different places where it’s possible to configure, it’s unclear what logic decides which config is respected

Everything is taken from platform data (aka STITCH_DATA), if the proper marker is found at reset vector+7 bytes. If it’s there, the debug port number is taken from it. If there’s no marker, the port is taken from PCD, which uses the value `DEBUG_PORT_NUMBER` from Python BoardConfig.py scripts.

```c
VOID
EarlyPlatformDataCheck (
  VOID
)
{
  STITCH_DATA          *StitchData;

  // Stitching process might pass some platform specific data.
  StitchData = (STITCH_DATA *)(UINTN)(0xFFFFFFF4);

  if (StitchData->Marker != 0xAA) {
    // set default as Debug UART
    // PlatformID will be deferred to be detected
    SetDebugPort (PcdGet8 (PcdDebugPortNumber));
  } else {
    SetDebugPort  (StitchData->DebugUart);
    SetPlatformId (StitchData->PlatformId);
  }
}
```

BootloaderCorePkg.dsc:

```
gPlatformCommonLibTokenSpaceGuid.PcdDebugPortNumber      | $(DEBUG_PORT_NUMBER)
```

Question

Students had bad experiences with SBL capsule update and are asking for support and a demo of how to do it “professionally”, question whether this is related to our SMI issues on MTL?

There is no connection with SMI on MTL. SBL doesn’t support SMM at all and doesn’t install SMI handlers: #1513, #1494.

However, capsule update worked flawlessly for us on ODROID (without Boot Guard, of course). Redundancy and a two-step update process are provided to minimize/eliminate risk of power failure, etc.

We would need to hear more about those issues. Updates using capsule when Boot Guard is enabled may be worth including in the training. Unfortunately, it wasn’t planned in the agenda. There’s also the limitation with the CSME update driver, which can only be compiled under Windows, in order for capsules to also update ME.

Day	Topic	Status	Notes
Day 1	Welcome and introduction	✅ Covered	As planned
	Hardware Theory of Operation	✅ Covered	As planned
	- Demo: Flashing and recovering the BIOS	✅ Covered	As planned
	Introduction to the x86 and Slim Bootloader boot flow	✅ Covered	As planned
	- Generic boot flow of UEFI firmware	✅ Covered	As planned
	- Boot flow of other firmware projects: U-boot, coreboot and Slim Bootloader	✅ Covered	As planned
	- Demo: Running Slim Bootloader on the hardware and observing the boot flow	✅ Covered	As planned
	Introduction to the x86 and Slim Bootloader boot flow	✅ Covered	As planned
	Q&A	✅ Extended	Extra 0.5 hours, free of charge

Day	Topic	Status	Notes
Day 1	Root of Trust and Chain of Trust Technologies	🔄 Shifted	Delivered on Day 2 instead
	- Root of Trust: Taxonomy and Division	🔄 Shifted	Delivered on Day 2 instead
	- Chain of Trust Technologies and Taxonomy	🔄 Shifted	Delivered on Day 2 instead
	- Measured Boot and Verified Boot	🔄 Shifted	Delivered on Day 2 instead
	- Demo: Investigating and breaking chain of trust in Slim Bootloader	🔄 Shifted	Delivered on Day 2 instead
Day 2	- Static vs Dynamic Root of Trust	✅ Covered	As planned
	- Example implementations of Root of Trust	✅ Covered	As planned
	- Intel FIT table and Authenticated Code Modules for Intel Boot Guard and TXT	✅ Covered	As planned

Day	Topic	Status	Notes
Day 2	Root of Trust assessment on Hardware	✅ Covered	As planned
	- Demo: Assessing platform security capabilities on proprietary BIOS and Slim Bootloader with Intel ME tools	🔄 Shifted	Delivered on Day 3 instead
	- Demo: Verifying Boot Guard state and configuration	🔄 Shifted	Delivered on Day 3 instead
	Integration and provisioning of Root of Trust	🔄 Shifted	Delivered on Day 3 instead
	- Demo: Provisioning and enabling Boot Guard in Slim Bootloader using	🔄 Shifted	Delivered on Day 3 instead
	- Demo: Verifying Boot Guard operation in Slim Bootloader	🔄 Shifted	Delivered on Day 4 instead
	Q&A+Day 1 material	✅ Extended	Extra 1 hour, free of charge

Day	Topic	Status	Notes
Day 3	Root of Trust assessment on Hardware	🔄 Shifted	Delivered on Day 4 instead
	- Intel ME Configuration and FPF fuses	🔄 Shifted	Delivered on Day 4 instead
	BIOS hardening and extra security mechanisms	🔄 Shifted	Delivered on Day 4 instead
	- BIOS hardening and extra security mechanisms	🔄 Shifted	Delivered on Day 4 instead
	- Chipset-based SPI protection and security (PRR, SMI BWP, etc.)	🔄 Shifted	Delivered on Day 4 instead
	BIOS security Assessment	🔄 Shifted	Delivered on Day 4 instead
	- CHIPSEC	🔄 Shifted	Delivered on Day 4 instead
	- fwupd HSI	🔄 Shifted	Delivered on Day 4 instead
	Q&A+Delays	✅ Extended	Extra 1.3 hours, free of charge

Day	Topic	Status	Notes
Day 4	Advanced Intel Boot Guard configuration and debugging	✅ Covered	As planned
	- Demo: Breaking Boot Guard Root of Trust and recovery	✅ Covered	As planned
	- Demo: Boot Guard debugging with Boot Guard status registers	✅ Covered	As planned
	- Demo: ME Boot Guard configuration and its impact on boot process	✅ Covered	As planned
	- End of Manufacturing and fusing process	✅ Covered	As planned
	- Q&A: final questions from attendees	✅ Covered	As planned
–	Bonus Topic: Slim Bootloader Measured Boot deep dive	➕ Added	Not in agenda, free of charge
	Q&A+Delays	✅ Extended	Extra X hours, free of charge

Feedback surve

Follow up

- Zarhus Provisioning Box
- Slim Bootloader Capsule Updates
- Slim Bootloader Boot Time Optimization
- Post-Qantum Crypto in Firmware
- Intel FSP
- Simics

Thank you