ARM Trusted Firmware
From Embedded to Enterprise

Dan Handley

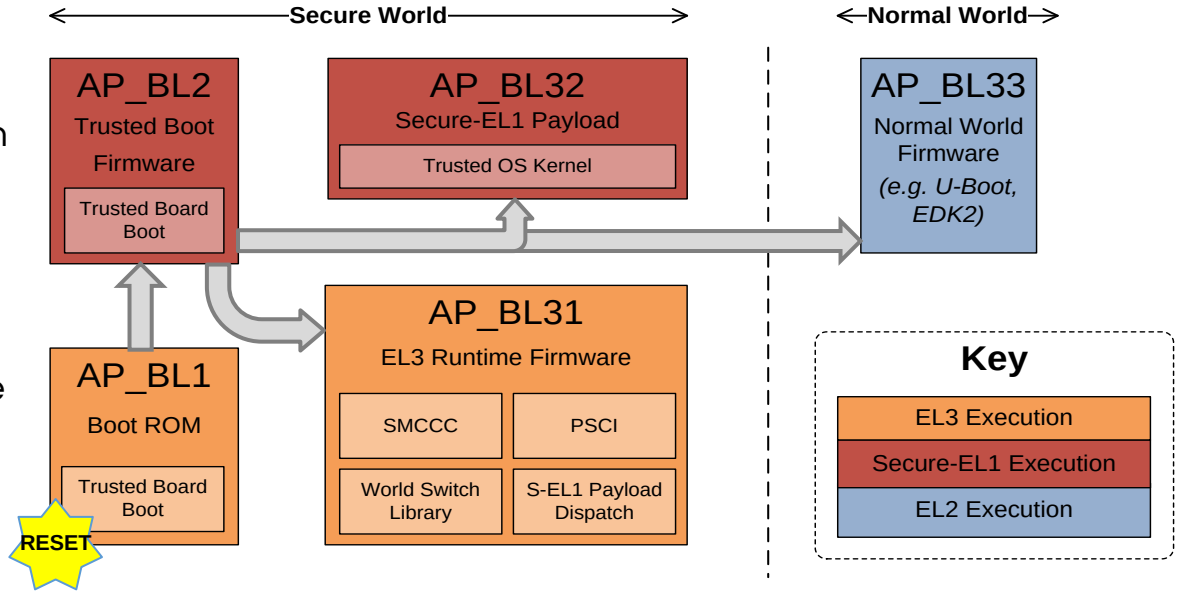
ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- Quick recap
- Project news
- Security hardening
- AArch32 support
- Other enhancements
  - Translation table library
  - RAM saving
  - Speed performance

ARM Trusted Firmware for AArch64 ARMv8-A

- Reference EL3 Runtime
  - Standard power control (PSCI)
  - Optional Trusted OS integration
- Trusted boot firmware
  - Optional
  - Compatible with other firmware
- Applicable to all segments
- Open Source at GitHub
  - BSD-3-clause license
  - Contributions welcome



1st level Boot Loader (BL1) loads 2nd level image

2nd level Boot Loader (BL2) loads other images

Loading

https://github.com/ARM-software/arm-trusted-firmware

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- Quick recap
- Project news
- Security hardening
- AArch32 support
- Other enhancements
  - Translation table library
  - RAM saving
  - Speed performance

TF inbound license model

- Before accepting code, ARM asks contributors to sign a CLA
  - Based on Apache license
  - Includes a list of authorized contributors (for companies)

- Has some benefits:
  - Corporate legal departments like them
  - Easier to change outbound license later
  - Signing once covers all ARM-maintained OSS projects (that use the CLA)

- But some serious drawbacks:
  - Can get stuck in corporate legal departments
  - Not popular with OSS developers
  - High barrier to entry
  - High admin overhead

We've ditched the CLA!

- Move to inbound = outbound model, retaining BSD-3-clause license
- All contributions accepted under DCO with "Signed-off-by" in commit message
  - Developer Certificate of Origin
  - See https://github.com/ARM-software/arm-trusted-firmware/contributing.md
- Benefits:
  - Same as Linux and increasing number of other projects
  - More community friendly
  - No admin overhead
  - (Hopefully) means more contributions
- What's not to like :o)

TF releases

- v1.2 created at start of year
- v1.3 coming very soon

Future:

- More regular release cadence
  - Requires test automation improvements
- Removal of deprecated APIs
  - Dual v1.4/v2.0 release

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- Quick recap
- Project news
- Security hardening
- AArch32 support
- Other enhancements
  - Translation table library
  - RAM saving
  - Speed performance

TF security incident handling

- Need to handle security vulnerabilities differently from other bugs
- Committed to public disclosure of all vulnerabilities
  - But want to disclose to partners under NDA first
  - Security advisories to be advertised on GitHub issue tracker
- New mailing list for reporting potential vulnerabilities
  - trusted-firmware-security@arm.com
  - For other bugs, continue using the issue tracker
- None found yet!
- https://github.com/ARM-software/arm-trusted-firmware/wiki/ARM-Trusted-Firmware-Security-Centre

Security hardening - completed

- Process improvements
- Independent code audit

- Use of Coverity Online

- Improved error checking / handling

- Enable SCR.SIF bit
- NV counter support in authentication module
- (Optional) separate mappings for code and RO/XN data

Security hardening - future

- Independent white box testing
- Improved secure coding/design guidelines

- Reduce what is mapped in
- Constrain mappings at runtime
- Put page tables in ROM

- More use of tools
  - Static analyzers, stack protectors, fuzzers, ...

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- Quick recap
- Project news
- Security hardening
- AArch32 support
- Other enhancements
  - Translation table library
  - RAM saving
  - Speed performance

AArch32 TF

Challenges

- Many existing AArch32 systems
  - Hard to standardize back into this space
- No EL3 / S-EL1 separation
  - Trusted OS / Secure FW tightly bound
  - SW separation is hard
- Trusted OS already have TF features
  - World switching
  - SMC routing
  - Interrupt handling
- Full AArch32 TF may not be adopted

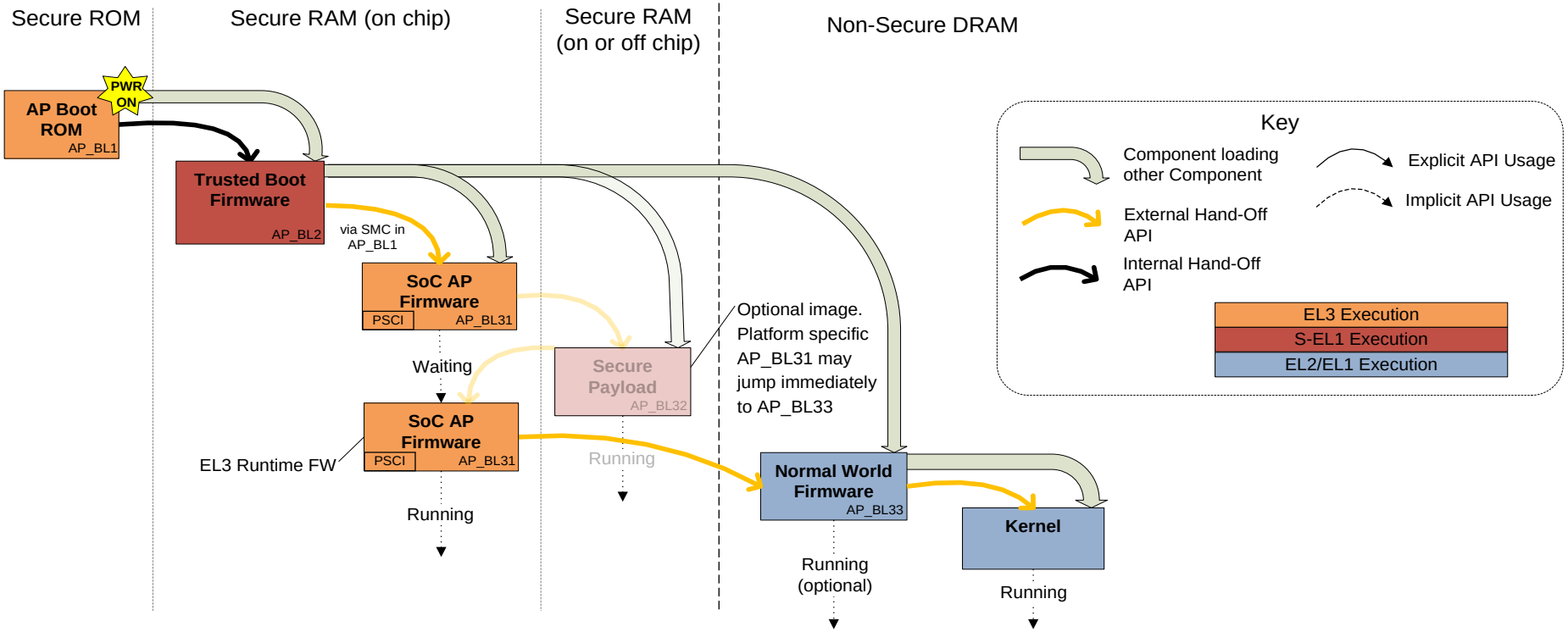
Opportunities

- New systems still being produced
  - AArch32-only ARMv8-A and ARMv7-A
- PSCI equally applicable for AArch32
- No reference secure world boot FW
- Current solutions either
  - Hack secure FW into normal world FW, or
  - Are specific to a Rich OS / Trusted OS, or
  - Are proprietary

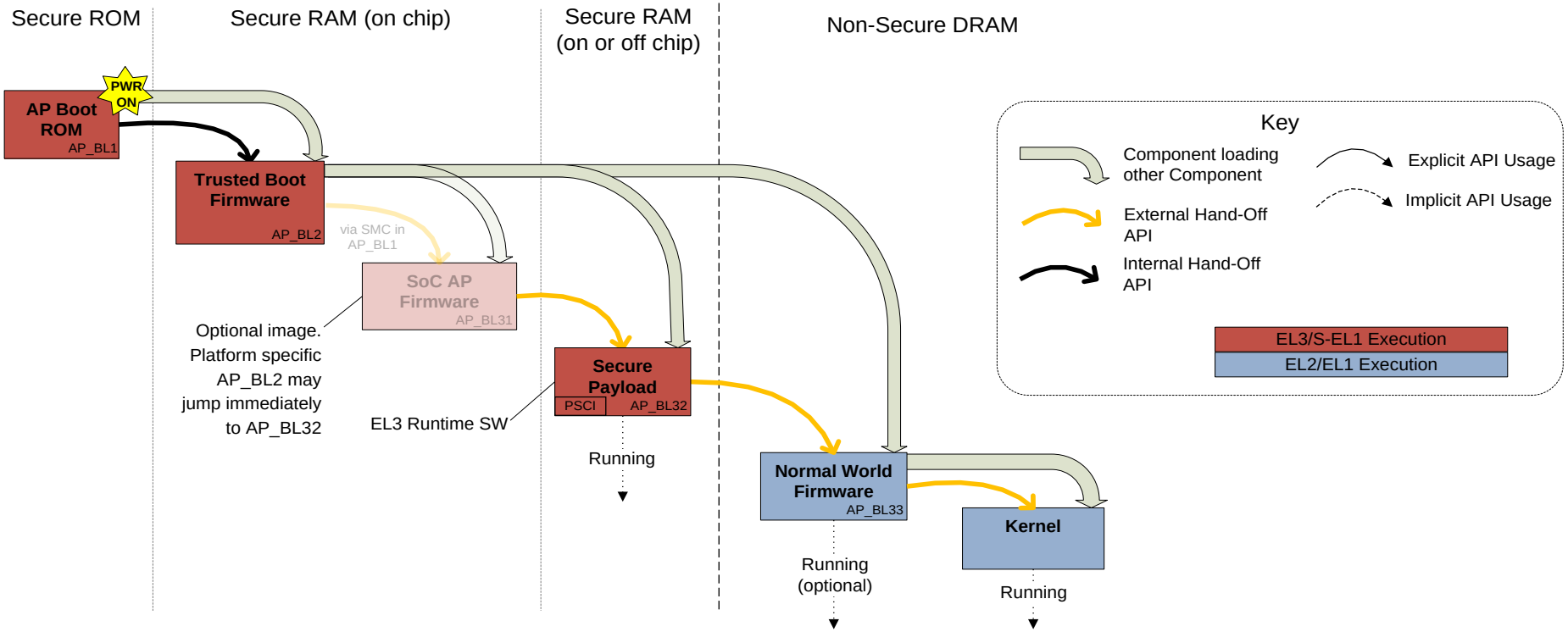
Initial AArch32 TF solution

- PSCI library
  - Reference integration into AArch32 EL3 Runtime SW
- Full port of BL1/BL2 to AArch32
- CPU library for Cortex-A32
- Enhance ARM FVP port to support Cortex-A32 variant
- No explicit ARMv7-A support
  - Focus on AArch32 ARMv8-A, which gets most of the way there

AArch64 boot flow



AArch32 boot flow



TF PSCI library

- Common in-source library for AArch64 and AArch32
  - Uses existing platform porting interface
- Integration interface provided for EL3 Runtime SW
  - Implemented by generic BL31 in AArch64
  - Must be integrated into each BL32 for AArch32
  - SP_MIN provided to demonstrate minimal AArch32 EL3 Runtime SW with PSCI
- Pulls in other TF library code
  - Context management, CPU ops, bakery/spin locks, ...
- Requires implementations of other utility functions
  - assert, panic, memcpy, printf, cache management, ...
  - EL3 Runtime SW can use TF implementations or its own

New image loading code: LOAD_IMAGE_V2

- BL2 no longer has hardcoded awareness of specific BL3x images
  - Allows more data driven approach to image loading and execution
  - Optional static descriptor mechanism provided
- No longer uses esoteric top/bottom loading behaviour
- AArch32 BL1/BL2 require LOAD_IMAGE_V2
  - TBBR not fully enabled yet
- Also applicable for AArch64, but not by default yet
- Unlocks other image loading use-cases

AArch32 and image loading future

In progress:

- Fully enable TBBR for LOAD_IMAGE_V2 generally and AArch32 in particular
- Use LOAD_IMAGE_V2 in ARM platforms for AArch64 (as well as AArch32)

Collaborate to:

- Integrate PSCI library into AArch32 OP-TEE
- Enable OP-TEE paging using LOAD_IMAGE_V2
  - Both AArch64 and AArch32

Future:

- ARMv7-A support
- Enable on Juno (requires SCP support)

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Agenda

- Quick recap
- Project news
- Security hardening
- AArch32 support
- Other enhancements
  - Translation table library
  - RAM saving
  - Speed performance

Translation table library enhancements

Completed:

- Uncached memory type
- 4 level page table support

In progress:

- Support dynamic changes to translation tables
- Support allocated virtual addresses

Future:

- Standardize BL image memory mappings
- More flexibility in memory types and attributes

RAM savings - future

- Use non-identity mappings to reduce page table size
- Use 'tiny' GCC memory model
- Put cold boot and runtime code/data in separate sections

Speed performance

Completed:

- PSCI STATs implementation
- Performance Measurement Framework (PMF)

In Progress:

- Runtime instrumentation in BL31
- Analysis of key PSCI use-cases, especially suspend to power down
- Enable platforms that support hardware-assisted coherency

Possible future:

- Use DC ZVA instruction to zero memory
- Reduce the amount of cache flushing during image load/auth

Other enhancements - completed

- New IP support - Cortex-A32, Cortex-A35, Cortex-A73, DMC-500

New Platforms:

- Xilinx Zynq
- Rockchip 3368, 3399
- QEMU
- HiKey (coming soon)

- More build system flexibility (e.g. for Windows hosts)

- Ongoing test / automation improvements

Other enhancements - future

- Dynamic configuration support
  - Images to load/execute
  - Chain of Trust
  - Dual AArch64/AArch32 kernel support
- Position Independent Executable (PIE) support
- RAS enhancements
  - Software Delegated Exception Interface (SDEI)
  - Rework of exception handling framework
- Anything else?

Thank You

#LAS16

For further information: www.linaro.org

LAS16 keynotes and videos on: connect.linaro.org