Content

- Platform Security Architecture (PSA) Firmware Framework
- What is in Secure Partition Runtime Library?
- Placement of the Secure Partition Runtime Library
- Secure Partition Runtime Library Design
- Project Status
Platform Security Architecture (PSA) Firmware Framework

What is Secure Partition?
- Unit of isolation
- Execution environment for RoT service:
  - Provide access to resources
  - Protect code and data
  - Mechanisms to interact with other components

Isolation level 1 - Protect SPE from NSPE
Isolation level 2 - Protect PSA RoT from App RoT
Isolation level 3 - Protect each Secure partition from other Secure partition
What is in Secure Partition Runtime Library?

- C Runtime:
  - printf/assert
  - malloc/free/realloc
  - memcmp/memcpy/memmove/memset

- PSA APIs:
  - Client APIs
  - Secure partition APIs
  - RoT service APIs

- Miscellaneous APIs:
  - Boot data API
  - Others

- Secure partition launcher:
  - Initial necessary data for secure partition before entering main entry

- Secure Partition

- But all these APIs MUST be implemented with security consideration.

- What is requested in PSA FF:
  - Only Code is executable
  - Only private data is writable
  - Private data must be isolated
Placement of the Secure Partition Runtime Library

Option 1: Per-Partition Library

- Image size increases much.
- Hard to put into a single loaded image.
Placement of the Secure Partition Runtime Library

Option 2: All by Supervisor Call

- SPM consume more execution time.
- Library execution cannot be preempted.
- Library code runs under unnecessary privileged level.
Placement of the Secure Partition Runtime Library

Option 3: Partition Shared Library

- Fast and efficient.
- Dedicated MPU region needs to be reserved for library.
- A well-balanced implementation.
- THIS IS IT.
Launch Partitions with Secure Partition Runtime Library

1. Jump to lib::sprtmain
2. Runtime metadata
3. Init Heap
4. Heap manager instance is placed in Runtime Metadata.
5. Jump to SP::main
6. Programming error
Function call to Secure Partition Runtime Library

1. Function call
2. Get runtime metadata
3. Access SP memory
4. Function back

Function with Implied Parameters Passing:

void *malloc(size_t sz)
void *malloc_impl(size_t sz, void *p_inst)
Supervisor Call of Secure Partition Runtime Library

1. Function call
2. Supervisor call
3. Access SP memory
4. Supervisor back
5. Function back
Project Status

- Secure Partition Runtime Library Design had been public: [https://git.trustedfirmware.org/trusted-firmware-m.git/tree/docs/design_documents/tfm_secure_partition_runtime_library.rst](https://git.trustedfirmware.org/trusted-firmware-m.git/tree/docs/design_documents/tfm_secure_partition_runtime_library.rst)

- Secure Partition Runtime Library APIs status
  - PSA FF APIs – 60%
  - C Runtime – 30%
  - Secure Partition launcher – 10%
  - Miscellaneous APIs – 30%

- TF-M Source code: [https://git.trustedfirmware.org/trusted-firmware-m.git/](https://git.trustedfirmware.org/trusted-firmware-m.git/)
arm

Q & A
Thank You
Danke
Merci
谢谢
ありがとうございます
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
תודה