SFO15-200: TEE kernel driver

Jens Wiklander
Introduction

● A TEE (Trusted Execution Environment) is a Trusted OS running in some secure environment

● There exists a number of TEE implementations, each with their own out of tree kernel driver
Secure world

- Implementation discussed here based on ARM TrustZone

- Could be based on other technologies, for instance
  - Virtualization
  - Separate secure coprocessor
TEE Software components

- **User space**
  - TEE client library
  - tee-supplicant

- **Kernel driver**
  - TEE subsystem
  - TEE driver

- **Trusted OS**
  - The TEE itself, running in secure world
User space - TEE client library

- The user space API provided by the kernel has the building blocks needed to implement a full client API
  - For instance GlobalPlatform TEE Client API 1.0 which we’re using for OP-TEE
User space - tee-suppliant

● An optional helper daemon for Trusted OS
  ○ Similar daemons has been implemented for other TEE's
  ○ Can provide
    ■ file system access
    ■ access to shared resources
User space API

- Modeled after GlobalPlatform TEE Client API
  - open(/dev/teeX) - TEEC_InitializeContext()
  - close(fd from above) - TEEC_FinalizeContext()
  - ioctl(OPEN_SESSION) - TEEC_OpenSession()
  - ioctl(INVOKE) - TEEC_InvokeCommand()
  - ioctl(CLOSE_SESSION) - TEEC_CloseSession()
  - ioctl(SHM_ALLOC), mmap() - TEEC_AllocateSharedMemory()
Kernel driver - TEE subsystem

- Provides a generic API towards user space in `<uapi/linux/tee.h>`
- Provides an API towards the TEE drivers in `<linux/tee_drv.h>` which:
  - Handles registration of the TEE driver and its callbacks
  - Manages shared memory between user space, kernel and Trusted OS
Kernel driver - TEE driver

- Implements a driver for a Trusted OS
- Handles communication with secure world
  - How requests and responses are passed and received
  - Helps secure world with certain tasks and may forward some to tee-supplicant
  - These tasks could be sleep, wait for event, file system access, etc
Shared memory 1

- Shared memory between Linux user space and TEE is a must for bandwidth intensive applications
- Currently using the model required by OP-TEE
  - reserved region of physically contiguous memory
- Model can be extended when needed for other TEEs
Shared memory 2

- An allocated chunk of shared memory is represented by a `struct tee_shm` in the TEE subsystem and drivers

- To the rest of the kernel as a `struct dma_buf`

```c
struct tee_shm {
    struct list_head list_node;
    struct tee_device *teedev;
    phys_addr_t paddr;
    void *kaddr;
    size_t size;
    struct dma_buf *dmabuf;
    u32 flags;
};
```
Shared memory 3

- User space can \texttt{mmap()} a file descriptor connected to the \texttt{struct tee_shm}.

- Secure world uses a TEE specific representation
  - OP-TEE uses physical address and length
OP-TEE driver 1

- Implements two devices
  - Client device
  - Supplicant device
- Each device is described by a `struct tee_desc`

```c
struct tee_desc {
    const char *name;
    const struct tee_driver_ops *ops;
    struct module *owner;
    u32 flags;
};
```
OP-TEE driver 2

- Uses OP-TEE message protocol as secure world interface
- Enters secure world from clients task
- Remote Procedure Calls (RPC) to the supplicant
  - Rendez-vous with mutex and completions
  - Temporarily shares memory with the supplicant process
OP-TEE driver 3

- Shared memory between secure and nonsecure world has to have compatible cache settings in both worlds
  - On ARM systems that’s: Normal cached memory (write-back), shareable for SMP systems and not shareable for UP systems
Adding a new TEE driver

- The interface to secure world defines what the driver needs to handle, for instance
  - RPC: is a new supplicant needed?
  - Shared memory: is the current model enough or does it need to be extended?
  - What happens when an IRQ is received while in secure mode?
Status

- The latest patch set is V5 https://lwn.net/Articles/655018/
- The general interest at the mailing lists is low as this is a narrow field
- Please help reviewing, especially the internals of the “tee: generic TEE subsystem” patch