BUD17-400 Secure Data Path with OP-TEE

Mark Gregotski, Director LHG
Overview

- Move to ION-based Secure data path Memory Allocator (SMA)
- OP-TEE/SDP extensions of the GlobalPlatform TEE APIs
- ION SMA implementation for Android and Linux-based media framework solutions
- Focus on the Android Media Framework
- Secure buffer reference using dma-buf
Migration to ION-based SDP Memory Allocator

- OP-TEE integration with ION is being driven by the Security Working Group (SWG)
  - Extensive work done by Etienne Carriere to define the OP-TEE/SDP integration

- LHG plan is to use ION SMA as the underlying secure buffer mechanism for both Android and Linux-based secure media framework implementation

- The Security Working Group has proposed OP-TEE/SDP extensions to the GlobalPlatform APIs

- SWG has proposed extensions to the ION memory pools (“heaps”) types
Highlights of Secure Data Path Support in OP-TEE (1)

- A set of OP-TEE/SDP extensions extend the GlobalPlatform APIs

- One important extension allows non-secure REE to allocate secure memory
  - A Trusted Application can access secure memory references provided as invocation parameters

- REE is responsible for the allocation of Secure Data Path (SDP) buffers

- TEE is responsible for providing a TA clear & safe memory references to both non-secure shared memory and SDP secure memory buffers

- SDP memory buffers are secure; Client App cannot access the buffer

Source: SDP Support in OP-TEE - Etienne Carriere
Highlights of SDP Support in OP-TEE (2)

● OP-TEE/SDP extensions provided for:
  ○ TEE Client APIs to register a SDP memory buffer into the TEE framework
  ○ OP-TEE Linux Driver: creates a ‘shared memory’ instance of each memory reference used as invocation parameters
  ○ OP-TEE Core: extension to GP TEE specification for a TA to be ‘SDP aware’
  ○ Trusted Application: extension allows TA to be invoked with a SDP memory buffer as invocation parameters; virtual memory range (base addr, size) passed as an argument to the TA entry point

● SDP Aware Trusted Applications
  ○ For a TA to be invoked with SDP memory reference parameters, the TA requires support from the OP-TEE core
    ■ OP-TEE/SDP extension allows a TA to check memory reference as being either non-secure (shared) or secure
SDP Memory Allocator (SMA)

- SMA mechanism assume non-secure world manipulates the secure memory reference in clear form (physical base addr + byte size)
  - ION implements the Secure Shared Memory Allocator agent in non-secure world

- ION Memory Allocator framework registers specific memory pools (“heaps”) and gets allocation support from both kernel- and user-space

- The defined ION secure heap type allocation algorithm
  - None of the natively supported ION heaps offer the service expected by a secure memory allocator
  - Need to allocate a physical region of memory the allocator cannot access
  - Borrowed the concept of a “secure heap” originally created by Allwinner Technology
  - Renamed to “unmapped heap” since it is basically an “unmapped memory allocator” driver for ION
SDP Memory References in Non-secure World

- OP-TEE chose the Linux kernel native “dma-buf” framework to reference SDP memory buffers:
  - Linux user-space, SDP buffer referred to by a dma-buf file descriptor
  - Linux kernel-space, SDP buffer referred to by a dma-buf handle

- OP-TEE/SDP extension of the TEE Client API provides an API to allow any “dma-buf” file descriptor to be registered as a “shared memory instance” in order to invoke a TA through the generic TEE Client API
Android Media Framework Hardening

- Look at how Media and DRM processes are defined in Android 7.0
- Reconcile the Android native buffer handle implementation with the ION-SMA framework using dma-buf file descriptors and with IPC
- With MediaDrm and MediaCrypto stacks in the new mediadrmserver process, buffers are allocated differently
- Vendors must update the secure buffer handles so they can be transported across binder when MediaCodec invokes a decrypt operation on MediaCrypto

In Android 7.0 and higher, buffer allocation in mediaserver. [https://source.android.com/devices/media/framework-hardening.html](https://source.android.com/devices/media/framework-hardening.html)
Secure Buffers using Native Handles

- In Android 7.0, the OMX:: allocateBuffer() must return a pointer to a native_handle struct
  - Contains File Descriptors (FDs) and additional data
- A new OMX extension (OMX.google.android.index.allocateNativeHandle) can be queried for this support and an OMX_SetParameter call that notifies the OMX implementation it should use native handles
- SoC vendors who use FDs to represent secure buffers can populate the FD in the native_handle
- Use dma-buf FDs in the native_handle struct
The proposal for OP-TEE/SDP with ION. MediaServer allocates secure buffer initially and passes the ION FDs to MediaDrmServer and MediaCodecService via Native Handle via Binder. In secure world the decryption and decoding are performed by the respective TAs. Output to display, e.g. HDMI output, protected by HDCP.
Thank You

#BUD17

For further information: www.linaro.org
BUD17 keynotes and videos on: connect.linaro.org