Dynamic secure firmware configuration

Dan Handley (Arm)
Recap

• BUD17 had a session to discuss possible secure world use of kernel Device Tree (DT)
  ○ Like the kernel, it’s desirable to have a single set of firmware run on multiple devices

• General agreement this would be good
  ○ At least for some platforms / use-cases

• Concerns this might not satisfy everyone’s needs
  ○ Performance / RAM footprint / complexity concerns on highly constrained devices
  ○ Still want some (lightweight) firmware configurability on platforms that don’t use (and don’t want to use) DT
Use-cases

1. Dynamic configuration of secure firmware features
   ○ e.g. enable/disable Trusted Board Boot (TBB), log level, normal world entrypoint info, …
   ○ e.g. Load address/size of images to load/authenticate (also requires PIE support)

2. Dynamic firmware configuration using hardware configuration
   ○ e.g. PSCI topology, debug console, secure watchdog, device base addresses, …

3. Dynamic configuration of a specific firmware component
   ○ e.g. DDR training parameters, TZC / NIC security policy, …

4. Centralized static firmware configuration
   ○ i.e. use same config information in 1-3 as input into build process

5. Modification of hardware configuration as seen by other software
   ○ e.g. probed runtime memory, secure memory reservation, kernel boot arguments, …
Observations

● Some config may be shared by various firmware components
  ○ e.g. hardware device info
  ○ DT is a good fit here

● Other config may be used by a single component, or contain “secret” information, or not relate to hardware
  ○ DT may or may not be a good fit; does not necessarily need to be in the kernel DT

● With TBB enabled, any dynamic config must be authenticated
  ○ Using an appropriate certificate

● Platforms need a lot of flexibility
  ○ Whether dynamic config feature is enabled or not, on a per component basis
  ○ Even with dynamic config feature, each option should still be statically configurable
  ○ The config file format to use

● Implies need for single “hw_config” plus a “fw_config” per component
Proposed Arm TF based boot flow (complex case)

- **Secure ROM**
- **Secure RAM (on chip)**
- **Secure RAM (on or off chip)**
- **Non-Secure DRAM**

Key:
- Component loading other Component
- External Hand-Off API
- Internal Hand-Off API

- **AP Boot ROM**
- **BL1**
  - **Secure Boot**
  - **Firmware**

- **BL2**
  - **SoC AP Firmware**
  - **tos_fw_config**
  - **PSCI**
  - **BL31**

- **BL32**
  - **Trusted OS**
  - **PSCI**

- **BL33**
  - **Normal World Firmware**
  - **PSCI**

- **Kernel**

- **hw_config**
  - **PWR ON**
  - **tb_fw_config**
  - **soc_fw_config**
  - **hw_config**
  - **External Hand-Off API**
  - **Internal Hand-Off API**
  - **Component loading other Component**

- **Secure ROM**
- **Secure RAM (on chip)**
- **Secure RAM (on or off chip)**

- **Copy from on-chip memory to final location. If not used by BL1, BL2 can load instead**

- **Used/modified by on-chip ROM/RAM firmware. If not used by BL1, BL2 can load instead**

- **Ptr to hw_config and relevant fw_config passed during each image hand-off**

- **Non-Secure Boot**
  - **Firmware**

- **External Hand-Off API**

- **Internal Hand-Off API**

- **Component loading other Component**

- **Used/modified by off-chip firmware and kernel**

- **Copy from on-chip memory to final location. If not used by BL1, BL2 can load instead**

- **Used/modified by on-chip ROM/RAM firmware. If not used by BL1, BL2 can load instead**

- **Copy from on-chip memory to final location. If not used by BL1, BL2 can load instead**

- **Run**
  - **PWR ON**
  - **BL1**
  - **BL2**
  - **BL31**
  - **BL32**
  - **BL33**
  - **Kernel**
TF config file format

- FIP tool and BL2 load/auth code are agnostic to format (just blobs)
  - Assuming file format can be easily detected by code that uses configs
  - e.g. by having a magic number at start like Device Tree

- Propose Device Tree as default format for all config files
  - Add support in generic make files to build DTS files and add configs to FIP
  - Add library functions to parse DTBs and populate dynamic config global data
  - Consider pre-build tools to convert DTS into code to enable centralized static config

- Various options if particular platform or BL does not want to use DT:
  - Use bl2_plat_handle_post_image_load() to convert into format that BL expects
  - Get platform make file to invoke tool to compile config using required format
  - Get platform make file to add blob in required format directly to FIP

- As other formats emerge, can move support from platform to generic code
TF hand-off interface

- On entry to BL31, conventionally use:
  - X0 for bl_params_t (list of images to execute)
  - X1 for pass platform specific data

- Propose instead:
  - X0 for bl_params_t (no change)
  - X1 for hw_config
  - X2 for soc_fw_config

- For entry to BL2, BL32 and BL33, propose:
  - X0 for hw_config
  - X1 for target BL fw_config

- Need to add these args to blX_early_platform_setup()
  - and maybe blX_platform_setup()

- These args are optional and still just a convention (at least to start with)
  - Arm platforms can provide reference but other platforms need to copy pattern
Configuration contents

- **hw_config** contains:
  - Everything kernel DT does already (plus any new secure device nodes)

- **tbb_fw_config** (shared by BL1/BL2) might contain:
  - Initial or final load address/size of hw_config
  - Address/size info needed to load BL3X images, certificates, configs
  - BL3X entrypoint info needed to generate bl_params_t
  - The Chain of Trust (certificate structure)
  - BL1/BL2 feature enablement: TBB on/off, etc...

- **Other fw_configs** can contain whatever the specific component needs
  - Even a subset of hw_config if platform doesn’t wish to use hw_config
TBBR Chain of Trust

- Root of Trust Public Key (ROTPK) or hash of it in OTP/eFuse
  - Trusted Boot Firmware Certificate
    - Hash of TB_FW (BL2)
  - Trusted Key Certificate
    - Trusted World Public Key
      - SoC Firmware Key Certificate
        - SoC Public Key
          - SoC Firmware Content Certificate
            - Hash of SoC_FW (BL31)
    - Normal World Public Key
      - Normal World Firmware Key Certificate
        - Normal World Content Public Key
          - Normal World Firmware Content Certificate
            - Hash of NT_FW (BL33)
  - Trusted OS Firmware Key Certificate
    - Trusted OS Public Key
      - Trusted OS Firmware Content Certificate
        - Hash of TOS_FW (BL32)
  - Normal World Firmware Key Certificate
    - Normal World Content Public Key
      - Normal World Firmware Content Certificate
        - Hash of NT_FW (BL33)
TBBR Chain of Trust

Root of Trust Public Key (ROTPK) or hash of it in OTP/eFuse

- Trusted Boot Firmware Certificate
  - Hash of HW_Config
  - Hash of TB_FW_Config
  - Hash of TB_FW (BL2)

- SoC Firmware Key Certificate
- SoC Public Key
- SoC Firmware Content Certificate
  - Hash of SoC_FW_Config
  - Hash of SoC_FW (BL31)

- Trusted Key Certificate
  - Trusted World Public Key
  - Normal World Public Key

- Trusted Boot Firmware Certificate
- Hash of TB_FW (BL2)

- Trusted OS Firmware Key Certificate
- Trusted OS Public Key
- Trusted OS Firmware Content Certificate
  - Hash of TOS_FW_Config
  - Hash of TOS_FW (BL32)

- Normal World Firmware Key Certificate
  - Normal World Content Public Key
  - Normal World Firmware Content Certificate
    - Hash of NT_FW_Config
    - Hash of NT_FW (BL33)
Open issues

• The TF build system is already creaking at the seams
  ○ The additional defines and targets could tip it over the edge
  ○ Is now the right time to consider an overhaul, e.g. leverage Zephyr work?

• Some config may be required by BL before MMU is enabled
  ○ e.g. console information to enable MMU setup debug output
  ○ e.g. load info for SCP or other firmware that needs to be loaded before platform setup

• Parsing at this time may be very slow

• Assume for now that platforms that care about this either
  ○ Configure statically
  ○ Defer config parsing to platform setup (after MMU is enabled), or
  ○ Use own config format
Other implications

- TF could end up being the de-facto kernel DT upstream for platforms that use TF BL1/BL2
TF implementation plan

● Phase 1 – basic infrastructure
  ○ Add support in build system for DT-based configs
  ○ Add minimal tbb-fw-config for reference platforms containing hw-config final address/size
  ○ Update ref platforms to compile configs and add them to FIP
  ○ Update ref platforms to load/auth configs and pass hw_config to BL33
  ○ Enable new boot flow in ref platform releases

● Phase 2 – community enablement
  ○ Add simple options to tbb-fw-config (e.g. TBBR on/off)
  ○ Enable soc-fw-config (e.g. RAS config)
  ○ Enable nt-fw-config for EDK2 (e.g. secure memory reservation)
  ○ Enable tos-fw-config for OP-TEE and others (e.g. OP-TEE header info)
  ○ Enable other config formats if needed

● Phase 3 – advanced features
  ○ Add image load/entrypoint info to tbb-fw-config and enable PIE
  ○ Use/modify hw-config in secure firmware
  ○ Evaluate new build system and tools for code generation
Thank You

#SFO17
BUD17 keynotes and videos on: connect.linaro.org
For further information: www.linaro.org