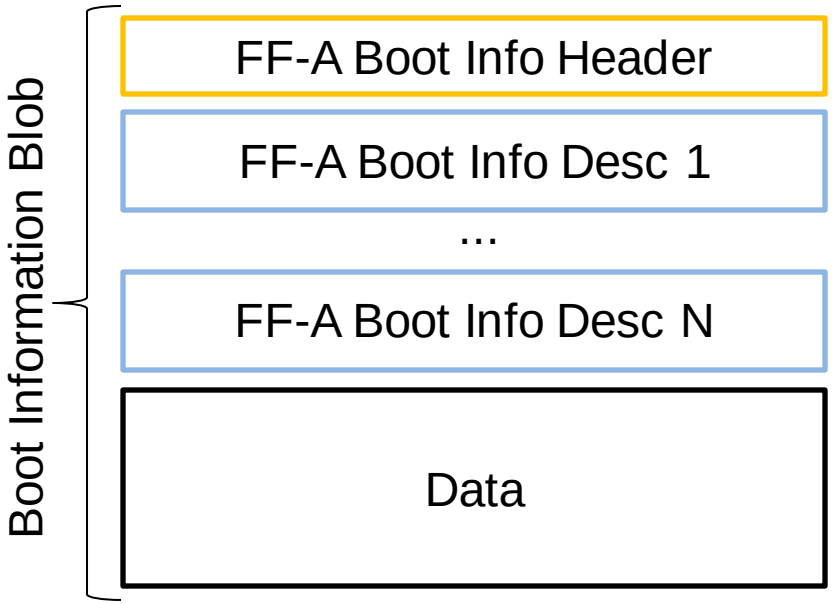
FF-A v1.1 boot protocol implementation (in Hafnium)

João Alves

FF-A v1.1 Boot Protocol Summary

- Boot information structures have been updated:
  - Boot information blob = boot info header + boot info descriptors + data.
- FDT boot info type to pass FF-A manifest to the partition.
- Memory management requirement to have the whole boot info blob into a continuous memory region:
  - Simplify memory mapping operation at S1.



Goals

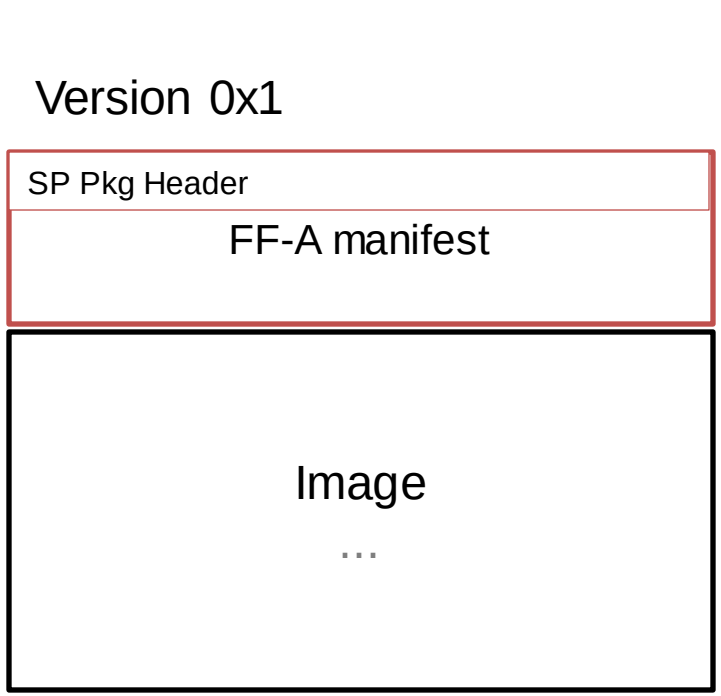
- SPMC interoperability (S-EL1 OPTEE and EL3 SPMC) by adopting the standard defined in FF-A v1.1 EAC0.
- Add ability to pass the SP manifest address using the FDT type.
- Allow for larger SP manifest sizes.
- Add ability to test SP with S1 translation granule larger than 4kb.

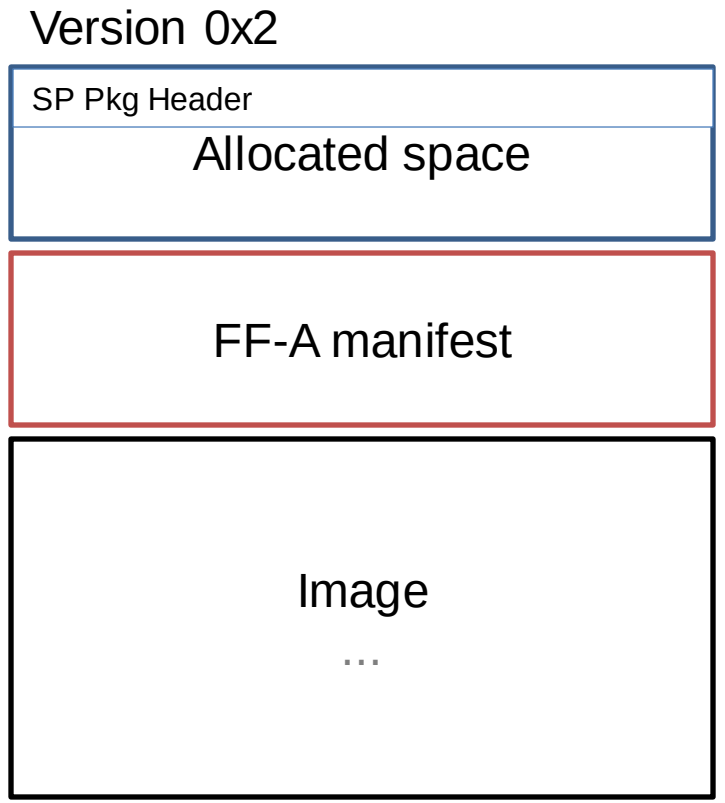
Solution Proposal

- Hafnium supports 1-to-1 memory mapping VA to PA.
- To avoid copying FF-A manifest from its loading location, allocate the memory for the boot Information in the SP Pkg:
  - SPTool change (bump package version to 0x2).
  - Hafnium's processing of the SP Pkg.
- Subscription to specific boot information in the SP's manifest.
- Patch stack available in link.

Secure Partition Package Update

- Hafnium supports both versions of the SP Pkg (patch).

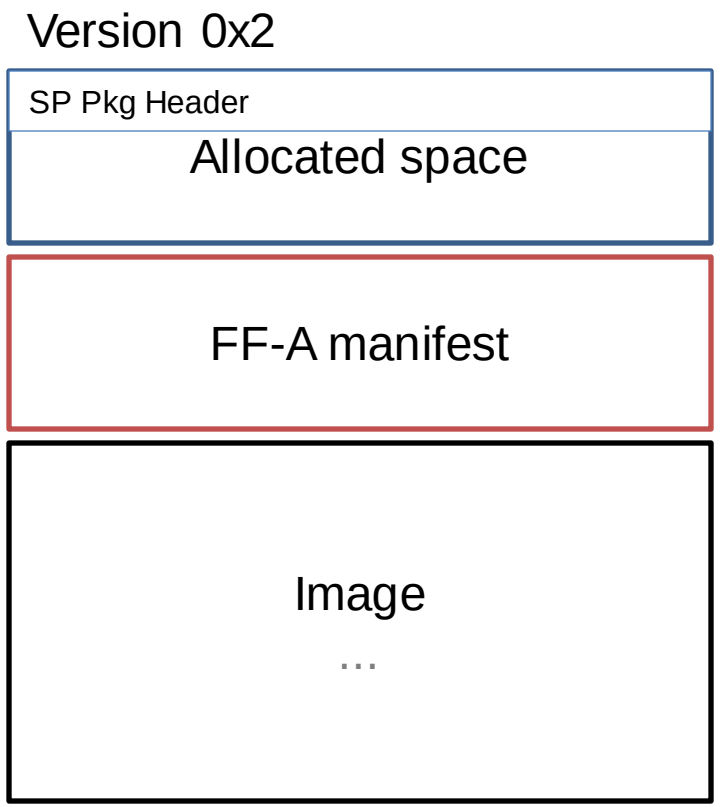




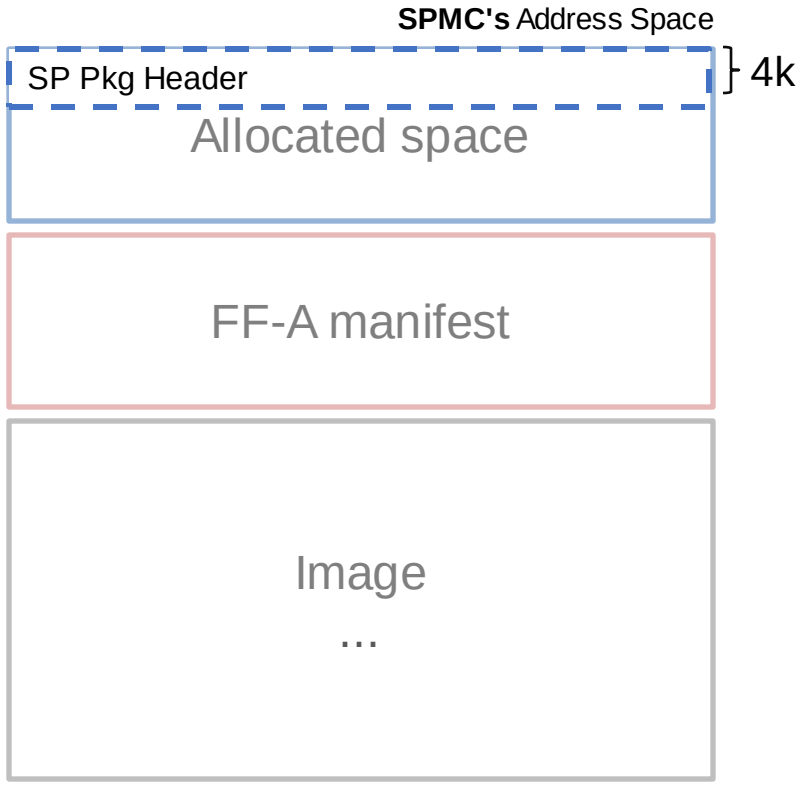
SPTool update

- Manifest and Image offset specified through the arguments to the tool
  - Default values: *0x1000* and *0x4000* respectively.
  - Means to the increase the size of the allocated space for the boot info descriptors.
  - Allow for packaging SP's with translation granules of 16Kb or 64Kb.
- Tool updated to python.

```
joaalv01@e124102:~/Workspace/trusted-firmware-a$ ./tools/sptool/sptool.py --help
usage: sptool.py [-h] -i I [--pm-offset PM_OFFSET] [--img-offset IMG_OFFSET]
                 -o O [-v]

optional arguments:
  -h, --help            show this help message and exit
  -i I                  path to partition's image and manifest separated by a
                        colon.
  --pm-offset PM_OFFSET
                        set partitition manifest offset.
  --img-offset IMG_OFFSET
                        set partition image offset.
  -o O                  set output file path.
  -v                    print package information.
```



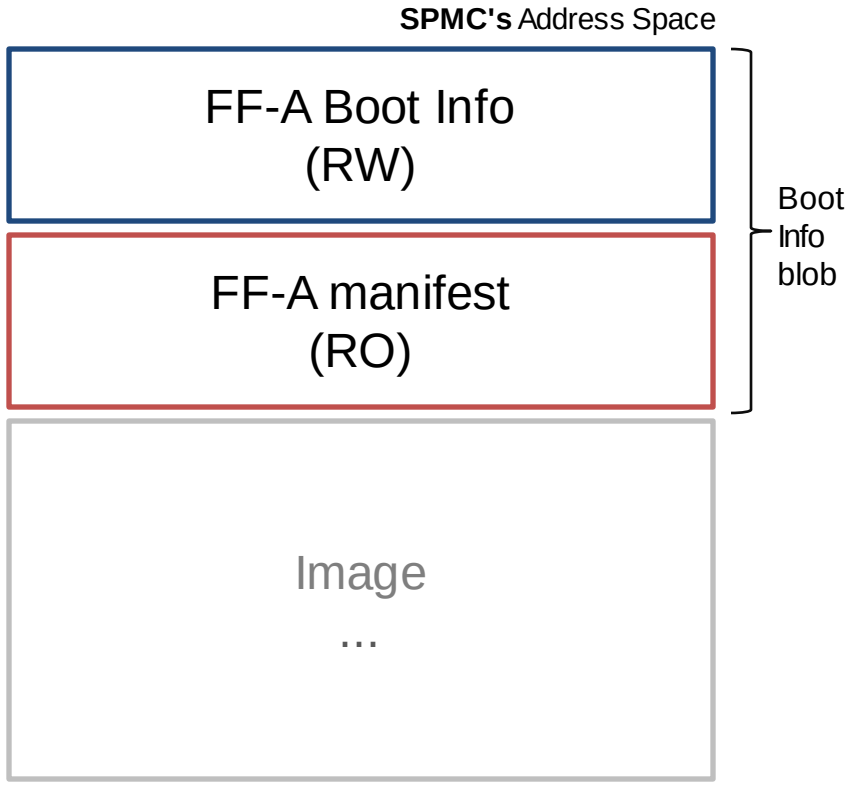
Hafnium update



- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP Pkg header.

```
/**
 *  Header for a SP Partition Package.
 */
struct sp_pkg_header {
        /** Magic used to identify a SP package. Value is "SPKG". */
        uint32_t magic;
        /** Version number of the header. */
        uint32_t version;
        /** Offset in bytes to the partition manifests. */
        uint32_t pm_offset;
        /** Size in bytes of the partition manifest. */
        uint32_t pm_size;
        /** Offset in bytes to the base address of the partition binary. */
        uint32_t img_offset;
        /** Size in bytes of the partition binary. */
        uint32_t img_size;
};
```

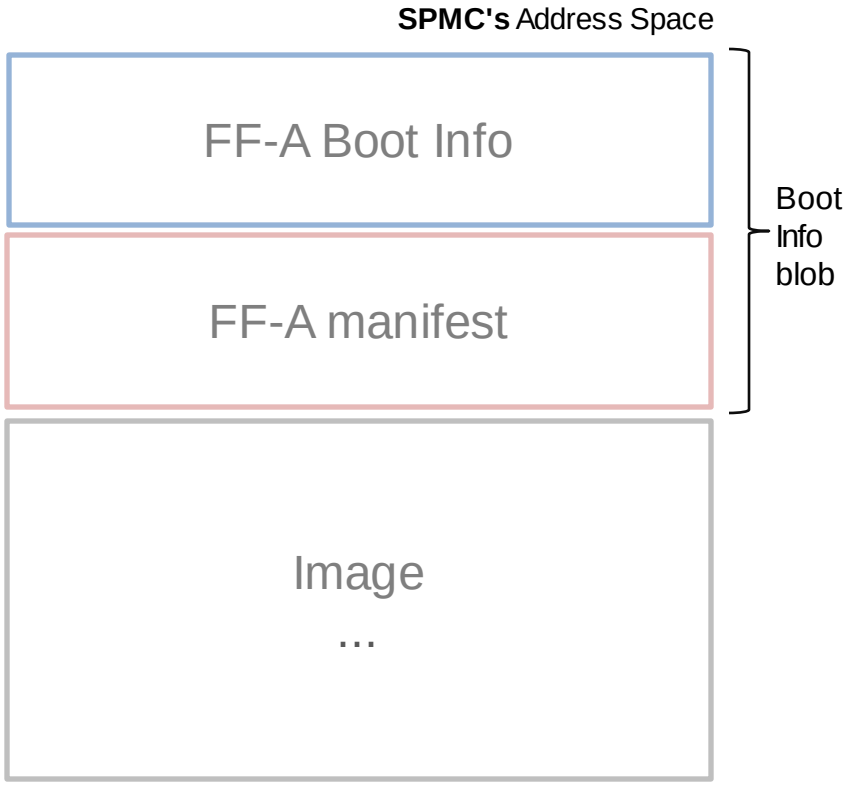
Hafnium update



- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP package header.
  2. Map FF-A Boot Information (RW) and FF-A manifest (RO) sections, into S-EL1 S1 translation.
  3. Parse FF-A manifest and populate FF-A boot info descriptors.
  4. Write the address of the boot info blob to the register specified in the 'gp-register-num' field.

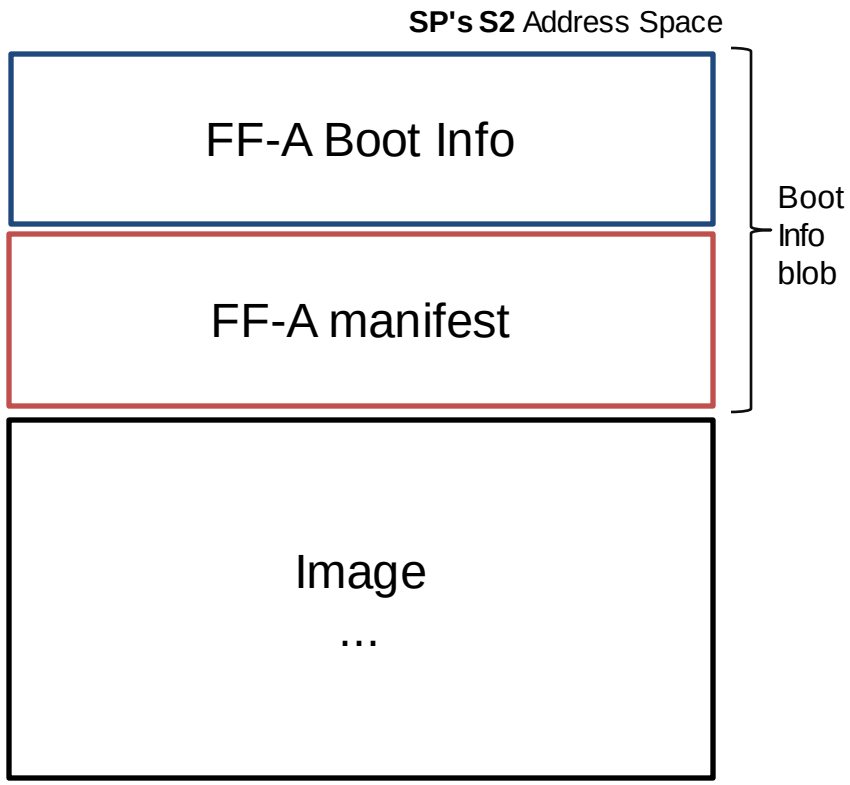
```
        gp-register-num = <0>;

        /* Boot Info */
        boot-info {
                compatible = "arm,ffa-manifest-boot-info";
                ffa_manifest;
        };
```

Hafnium update



- Per SP initialization:
  1. Map 4k (lowest translation granule) to read SP package header.
  2. Map FF-A Boot Info (RW) and FF-A manifest (RO) sections.
  3. Parse FF-A manifest and populate FF-A boot info descriptors.
  4. Write the address of the boot info blob to the register specified in the 'gp-register-num' field.
- SPMC unmaps from EL-2 Stage-1

SP initialization



- SP boot with S2 translation update.

```
INFO: Arm SMMUv3 initialized
INFO: Loading VM id 0x8001: ffa_secure_partition.
VERBOSE: VM has 0 physical interrupts defined in manifest.
VERBOSE: stream_count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x6280000.
INFO: Hafnium initialisation completed
VERBOSE: plat_psci_cpu_resume: cpu mpidr 0x0 ON
[hftest_ctrl:get_command_line]
VM 8001: [hftest] SP boot info (6280000):
VM 8001: [hftest]   Signature: ffa
VM 8001: [hftest]   Version: 10001
VM 8001: [hftest]   Blob Size: 4833
VM 8001: [hftest]   Descriptor Size: 32
VM 8001: [hftest]   Descriptor Count: 1
VM 8001: [hftest]     Type: 0
VM 8001: [hftest]     Flags:
VM 8001: [hftest]       Name Format: 0
VM 8001: [hftest]       Content Format: 0
VM 8001: [hftest]     Size: 737
VM 8001: [hftest]     Value: 6281000
VM 8001: [hftest] FINISHED
[hftest_ctrl:finished]
```

```
INFO: Loading VM id 0x8001: ffa_secure_partition.
VERBOSE: VM has 0 physical interrupts defined in manifest.
VERBOSE: stream_count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x6280000.
INFO: Hafnium initialisation completed
VERBOSE: plat_psci_cpu_resume: cpu mpidr 0x0 ON
[hftest_ctrl:get_command_line]
VM 8001: [hftest] FF-A Manifest Address: 6281000
VM 8001: [hftest] FF-A Version: 10001
VM 8001: [hftest] FINISHED
[hftest_ctrl:finished]
```

TF-A build integration

- Script sp_mk_generator.py parses sp_layout.json file and generates 'sp_gen.mk'.
- The patch extends the script to generate a make rule for an SP package, for each SP defined in the sp_layout.json.

```
SP_PKGS += /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-primary.pkg

/home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-primary.pkg:
	$(Q)echo Generating /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-primary.pkg
	$(Q)$(PYTHON) $(SPTOOL) -i /home/joaalv01/Workspace/tf-a-tests/build/fvp/debug/cactus.bin:/home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/fdts/cactus.dtb \
	                        --pm-offset 4096 --img-offset 8192 \
	                        -o /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-primary.pkg

SP_PKGS += /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-secondary.pkg

/home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-secondary.pkg:
	$(Q)echo Generating /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-secondary.pkg
	$(Q)$(PYTHON) $(SPTOOL) -i /home/joaalv01/Workspace/tf-a-tests/build/fvp/debug/cactus.bin:/home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/fdts/cactus-secondary.dtb
	                        -o /home/joaalv01/Workspace/trusted-firmware-a/build/fvp/debug/cactus-secondary.pkg
```

```
        "cactus-primary" : {
                "image": {
                        "file": "cactus.bin",
                        "offset":"0x2000"
                },
                "pm": {
                        "file": "cactus.dts",
                        "offset":"0x1000"
                },

                "owner": "SiP"
        },

        "cactus-secondary" : {
                "image": "cactus.bin",
                "pm": "cactus-secondary.dts",
                "owner": "Plat"
        },
```

FF-A manifest configuration

- Adjust the 'entrypoint-offset' to either match the image offset in the SP Pkg.
  - For version *0x2* of the SP Pkg this can be the default value taken by the sptool.py (0x4000), if no arguments are provided.
- Configure 'gp-register-num' to the register the SP expects to receive the address of the boot info blob.
- Define 'boot-info' node and list the 'ffa_manifest'.

```
        compatible = "arm,ffa-manifest-1.0";

        /* Properties */
        description = "Base-1";
        ffa-version = <0x00010001>; /* 31:16 - Major, 15:0 - Minor */
        uuid = <0x1e67b5b4 0xe14f904a 0x13fb1fb8 0xcbdae1da>;
        id = <1>;
        auxiliary-id = <0xae>;
        stream-endpoint-ids = <0 1 2 3>;
        execution-ctx-count = <8>;
        exception-level = <2>; /* S-EL1 */
        execution-state = <0>; /* AARCH64 */
        load-address = <0x7000000>;
        entrypoint-offset = <0x00002000>;
        xlat-granule = <0>; /* 4KiB */
        boot-order = <0>;
        messaging-method = <3>; /* Direct messaging only */
        managed-exit; /* Managed exit is supported */
        notification-support; /* Support receipt of notifications. */
        run-time-model = <0>; /* Run to completion */

        /* Boot protocol */
        gp-register-num = <0>;

        boot-info {
                compatible = "arm,ffa-manifest-boot-info";
                ffa_manifest;
        };
```

Limitations

- SPMC to SP boot information passing only.
- Parallel to firmware Handoff protocol.
- HOB and implementation defined boot information types not yet in use.
- SP Pkg impacts configuration of 'entrypoint-offset' in SP manifest and could be automated.

Thank You
Danke
Gracias
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה

SP parsing manifest

<HF_ROOT>/test/hftest/hftest.py --out_partitions out/reference/secure_aem_v8a_fvp_vm_clang --log out/reference/kokoro_log --spmc out/reference/secure_aem_v8a_fvp_clang/hafnium.bin --driver=fvp --partitions_json test/vmapi/ffa_secure_partition_only/ffa_secure_partition_only_test.json --suite ffa_boot_info –test parse_fdt

```
/**
 * Validate that SP can access its own FF-A manifest.
 */
TEST(ffa_boot_info, parse_fdt)
{
	struct ffa_boot_info_header* boot_info_header = get_boot_info_header();
	struct ffa_boot_info_desc* fdt_info;
	struct fdt fdt;
	struct fdt_node root;
	void* fdt_ptr;
	size_t fdt_len;
	uint64_t ffa_version;

	fdt_info = get_boot_info_desc(boot_info_header, FFA_BOOT_INFO_TYPE_STD,
	                              FFA_BOOT_INFO_TYPE_ID_FDT);

	ASSERT_TRUE(fdt_info != NULL);

	HFTEST_LOG("FF-A Manifest Address: %x", fdt_info->content);
	// NOLINTNEXTLINE(performance-no-int-to-ptr)
	fdt_ptr = (void*)fdt_info->content;

	ASSERT_TRUE(fdt_size_from_header(fdt_ptr, &fdt_len));
	ASSERT_TRUE(fdt_init_from_ptr(&fdt, fdt_ptr, fdt_len));
	EXPECT_TRUE(fdt_find_node(&fdt, "/", &root));

	EXPECT_TRUE(fdt_is_compatible(&root, "arm,ffa-manifest-1.0"));
	EXPECT_TRUE(fdt_read_number(&root, "ffa-version", &ffa_version));
	HFTEST_LOG("FF-A Version: %x", ffa_version);
	ASSERT_EQ(ffa_version, MAKE_FFA_VERSION(1, 1));
}
```

```
INFO: Loading VM id 0x8001: ffa_secure_partition.
VERBOSE: VM has 0 physical interrupts defined in manifest.
VERBOSE: stream_count of upstream peripheral device: 0
INFO: Loaded with 8 vCPUs, entry at 0x6280000.
INFO: Hafnium initialisation completed
VERBOSE: plat_psci_cpu_resume: cpu mpidr 0x0 ON
[hftest_ctrl:get_command_line]
VM 8001: [hftest] FF-A Manifest Address: 6281000
VM 8001: [hftest] FF-A Version: 10001
VM 8001: [hftest] FINISHED
[hftest_ctrl:finished]
```

Build

- TF-A

```
make CROSS_COMPILE=aarch64-none-elf-  \
SPD=spmd \
CTX_INCLUDE_EL2_REGS=1 \
ARM_ARCH_MINOR=5  \
BRANCH_PROTECTION=1  \
CTX_INCLUDE_PAUTH_REGS=1 \
PLAT=fvp DEBUG=1 \
BL33=../tf-a-tests/build/fvp/debug/tftf.bin  \
BL32=/home/joaalv01/Workspace/hafnium/out/reference/secure_aem_v8a_fvp_clang/hafnium.bin  \
SP_LAYOUT_FILE=../tf-a-tests/build/fvp/debug/sp_layout.json  \
 all fip
```

- TF-A-Tests

```
make CROSS_COMPILE=aarch64-none-elf-  \
PLAT=fvp \
DEBUG=1 \
TESTS=spm
```