arm

Security Risk Analysis - Arm Firmware Framework for Arm A-profile

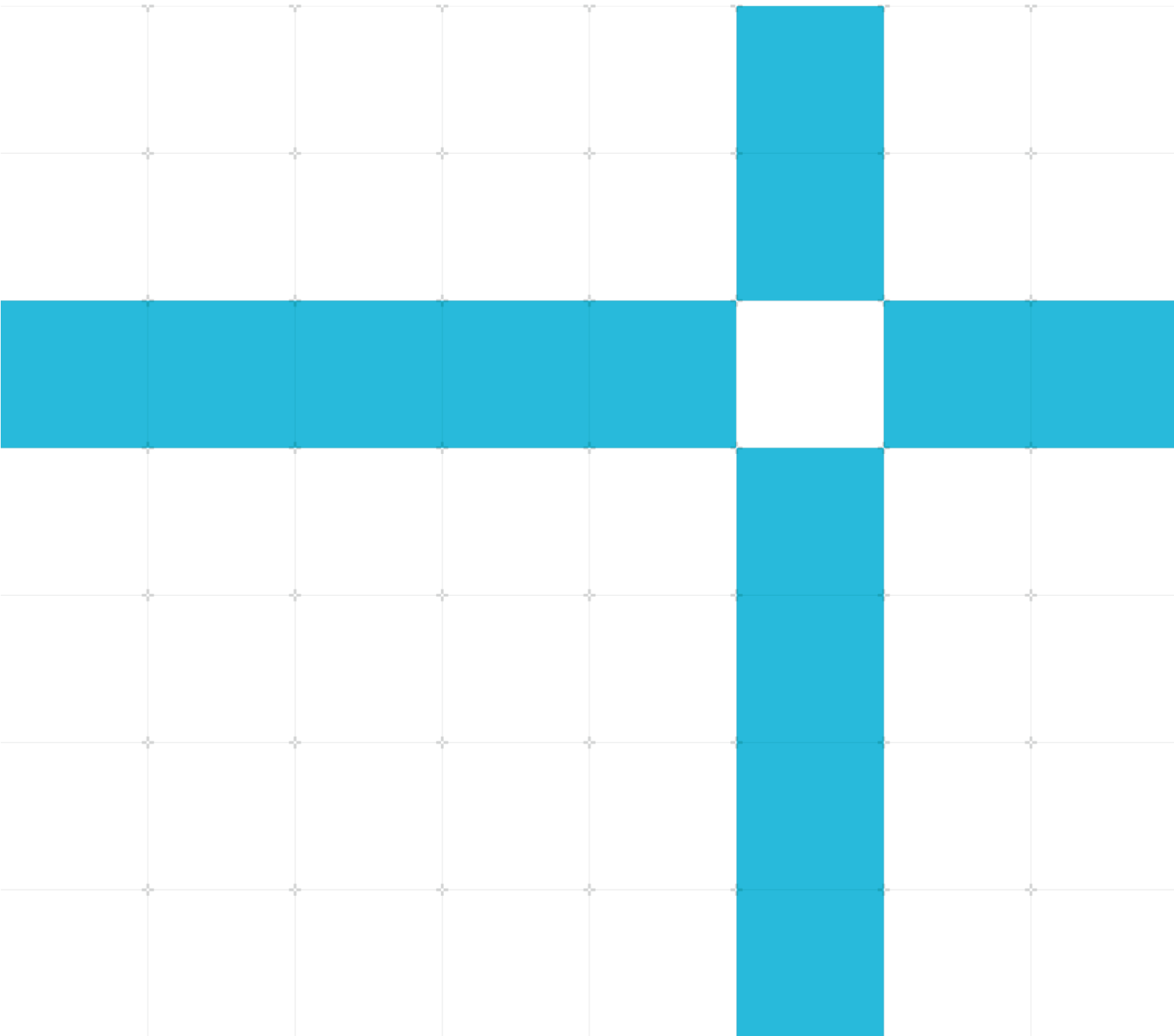Revision: [1.0]

Security Risk Assessment

Non-Confidential

Issue 01

[**ARM DDI 0619**]

Security Risk Analysis - Arm Firmware Framework for Arm A-profile

Security Risk Assessment

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
01	2022/Sep/28	Non-Confidential	ALP0 release. It tracks REL version of the Arm Firmware Framework for Arm A-profile v1.1.

Non-Confidential Proprietary Notice

analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE'S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

The Document consists solely of commercial items. Licensee shall be responsible for ensuring that any use, duplication or disclosure of the Document complies fully with any relevant export laws and regulations to assure that the Document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

This Licence may be translated into other languages for convenience, and Licensee agrees that if there is any conflict between the English version of this Licence and any translation, the terms of the English version of this Licence shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. No licence, express, implied or otherwise, is granted to Licensee under this Licence, to use the Arm trade marks in connection with the Document or any products based thereon. Visit Arm's website at https://www.arm.com/company/policies/trademarks for more information about Arm's trademarks.

The validity, construction and performance of this Licence shall be governed by English Law.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: LES-PRE-21585 version 4.0

Confidentiality Status

This document is Non-Confidential. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Product Status

The information in this document is Beta, that is for a product under development.

Web Address

https://developer.arm.com

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

If you find offensive language in this document, email **terms@arm.com**.

Contents

1 Introduction

1.1 Intended audience

This document is developed in conjunction with the development of the FF-A specification, and as such should be reviewed by those architecting the specification. Implementers of the architecture should refer to this document to ensure that the threats under scope are part of the implementation considerations. It is also expected that the document will be referred to by teams performing security validation to arrive at the scenarios, and by those reviewing the validation effort.

1.2 Conventions

The following subsections describe conventions used in Arm documents.

1.2.1 Abbreviations

This document uses the following abbreviations.

Table 1-1: Abbreviations and its meaning

Term	Meaning
ABI	Application Binary Interface
AMU	Activity Monitor Unit
API	Application Programming Interface
ASLR	Address Space Layout Randomization
CIA	Confidentiality, Integrity, and Availability
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
DMA	Dynamic Memory Access
DRAM	Dynamic Random Access Memory
DRM	Digital Rights Management, a systematic approach to copyright protection for digital media.
FF	Firmware Framework
FPU	Floating Point Unit

Term	Meaning
GIC	Generic Interrupt Controller
IP	Intellectual Property
OEM	Original Equipment Manufacturer
OS	Operating System
PE	Processing Element
PM	Partition Manager
PMU	Performance Monitor Unit
PSA	Platform Security Architecture
RAS	Reliability, Availability, and Serviceability
SiP	Silicon Partner
SME	Scalable Matrix Extensions
SMMU	System Memory Management Unit
SoC	System on Chip
SP	Secure Partition
SPCI	Secure Partition Client Interface: ABIs that partitions can invoke to communicate with other partitions.
SPM	Secure Partition Manager
SRA	Security Risk Analysis
SVE	Scalable Vector Extensions
TA	Trusted Application
TCB	Trusted Computing Base: set of all hardware, firmware, and/or software components responsible for enforcing a security policy
TEE	Trusted Execution Environment
TOS	Trusted Operating System
VM	Virtual Machine
VMSA	Virtual Memory System Architecture

1.2.2 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: **https://developer.arm.com/glossary**.

Non-Confidential

1.2.3 Typographical conventions

Convention	Use
italic	Introduces citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace` **bold**	Denotes language keywords when used outside example code.
`monospace` <u>underline</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
Note	This represents an important piece of information that needs your attention.
Tip	This represents a useful tip that might make it easier, better, or faster to perform a task.
Remember	This is a reminder of something important that relates to the information you are reading.

1.3 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

Table 1-2: Arm publications

Document name	Document ID	Licensee only
Arm Firmware Framework for Arm A-profile	DEN0077A	No
Arm® System Memory Management Unit Architecture specification versions 3.0, 3.1 and 3.2	ARM IHI 0070	No
Cache Speculation Side-channels whitepaper	https://developer.arm.com/documentation/102816/0205/	No
Arm Architecture Reference Manual for A-profile	DDI0487H.a	No

Table 1-3: Other publications

Document ID	Document name
SP 800-30	NIST Special Publication 800-30 Risk Management Guide for Information Technology Systems. (1) NIST and Aroms, E.

1.4 Feedback

Arm welcomes feedback on this product and its documentation.

1.4.1 Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name
- The product revision or version
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

1.4.2 Feedback on content

If you have comments on content, send an email to errata@arm.com and give:

- The title Security Risk Analysis - Arm Firmware Framework for Arm A-profile.

- The number [ARM DDI 0619].

- If viewing a PDF version of a document, the page number(s) to which your comments refer.

- If viewing online, the topic names to which your comments apply.

- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of this document when used with any other PDF reader.

---

2 Overview

This document describes and models the threats that affect theSecurity Risk Analysis - Arm Firmware Framework for Arm A-profile specification. The cost for removing defects increases rapidly through different phases of a typical product lifecycle like requirements, design, coding, test, and release phases. Also, the cost for fixing defects in architecture specifications is high as they impact implementations of the architecture. Identifying issues during the early phases can help minimize both cost and effort.

The security aspects of the Firmware Framework specification can be evaluated by performing Threat Model and Security Analysis (TMSA) of the Firmware Framework, along with the specification development. Threat modeling provides a systematic process to identify assets that are of interest to the attacker, their profile, the system vulnerabilities that attackers can exploit, and the countermeasures for mitigating these attacks. The threat models have been created using an English Language Protection Profile-style approach to enable ease of use and accessibility for engineers, regardless of their security knowledge or expertise.

3 About this assessment

3.1 Subject and scope

This document captures the security risk analysis of the Arm Firmware Framework for Arm A-profile (FF-A) specification. It lists the possible threats to the Firmware Framework in an environment where it aims to provide isolation of mutually mistrusting software components. For example, isolation of Normal world software from Secure world or isolation of Secure world software components from each other. The firmware framework which enables communication between isolated software components is also under consideration as it supports the primary security requirement of software isolation.

A simplified risk assessment of various threats implies that different sets of mitigations are put in place so FF-A can meet the stated security goals. The mitigations (hereafter called security objectives) form the basis for security functional requirements.

3.2 Methodology

This section explains the processes used to perform the SRA of the FF-A architecture.

3.2.1 Architecture analysis

This document analyzes the FF-A architecture specification, and not any specific implementation. That is, the architecture is the set of behaviors that are exhibited by an FF-A implementation and the rules that an implementation must abide by, to be compliant with the architecture.

3.2.2 Iterative release

This document aligns with the project and is released at a regular cadence. Any advances to the design of the product should additionally be documented here.

3.2.3 Product definition

The purpose of defining the system is to obtain a detailed picture of the object in analysis, which aids in its robustness.

- Stage one is to document a high-level description of the system - its purpose, an overview of the design, and any other details that are relevant.

- Stage two is to gather inputs from market research. This includes any initial requirements, any constraints and assumptions for the project in that market. The focus of the research is purely from a security perspective, meaning further market research beyond security is out of scope of this document.

- The final stages are an in-depth documentation of the system. This includes defining the life cycle, components of the system, operational details, information flow, trust boundaries, stakeholders, assets, interacting entities and attack surfaces. The functional details of the system are typically provided by the architects of the project.

- The assets resident in the system and the groups of stakeholders who have a vested interest in these assets are captured. Stakeholders may not be interested in every aspect of an asset's protection; therefore each stakeholder-asset pair shall be given a set of properties. These properties are confidentiality, integrity, and availability, with which different interests in an asset can be represented. For instance, a stakeholder may only be concerned with the integrity of the asset, while confidentiality and availability are not relevant (for example, the asset could be some configuration data).

> **Note**
>
> Revenue and reputation, while still considered as assets, are not resident in the system and need not be documented in the list of assets.

3.2.4 Threat analysis

- Stage one in threat analysis is to define the adversarial models in use, which are derived from market requirements. The adversarial models are representative of the different levels of skill and resources that could be employed by an adversary. These are derived from the use cases, attack surfaces, and the adversaries that are expected to attempt to exploit the system.

- Stage two is to enumerate all identifiable threats to the system. This is performed by using the system definition to analyze the information flow over trust boundaries and to examine the attack surfaces. Threats are also documented with methods of exploitation.

- Stage three is to assess the risk associated with each threat. The risk ratings follow the five-level version of the Arm SRA methodology, which is derived from the *NIST SP 800-30 Rev 1* document. The likelihood and impact of each threat is determined in this stage. These threats are then evaluated on a 5-level scale, as defined in the following tables.

Table 3-1: Likelihood levels

Likelihood level	Description
Very Low (VL)	Threat is very unlikely to occur in practice, or mathematically near impossible
Low (L)	Threat can be exploited but would require significant resources or it is mathematically unlikely to occur.
Medium (M)	A motivated and well-equipped adversary can exploit the threat within the lifetime of the product based on the architecture.
High (H)	Threat is likely to be exploited within the lifetime of the product or architecture.
Very High (VH)	Threat will most certainly be exploited (for instance a zero-day)

Table 3-2: Impact levels

Level	Definition	Example Effects
Very Low (VL)	Has virtually no impact.	Probably none
Low (L)	The damage can easily be tolerated / absorbed.	There would be a CVE at most
Medium (M)	The damage has a noticeable effect, degrading some part of the functionality without completely degrading the use of the considered functionality.	There would be a CVE at most
High (H)	The damage has a severe effect, that is some functionality is suppressed or defeated.	Security analysist discusses this at length with papers, blog entries, and some partners would complain.
Very High (VH)	The damage has critical consequences, such as killing the feature, by affecting several of its security guarantees	Results in a big event where partners strongly complain, delay, or cancel deployment of the feature

Finally, these two measures are combined to obtain the overall risk of a threat, as shown in the following table. There are again five levels of risk, ranging from Very Low to Very High. For both likelihood and impact, when in doubt, always choose the higher value as a safety measure.

Figure 3-1 Risk assessment matrix based on likelihood and impact

Risk Level		Impact				
		VL	L	M	H	VH
Likelihood	VH	VL	M	H	VH	VH
	H	VL	L	M	H	VH
	M	VL	L	M	M	H
	L	VL	VL	L	L	M
	VL	VL	VL	VL	L	L

3.2.5 Mitigations

After risk assessment, threats are sorted in descending order of risk. This is the order in which they are addressed. In the mitigation process, each threat is assigned an action describing what shall be done in response to the risk. This is known as a mitigating action. The mitigating actions that can be taken are defined in the acronym CAST:

Table 3-3: Mitigating actions

Mitigation	Description
Control	• Put mitigations in place to reduce the likelihood and/or impact of a threat, thereby reducing the risk to an acceptable level. • Mitigations for SRA are listed as Security Objectives.
Accept	The threat is considered to be a low enough risk that mitigation is not necessary.
Suppress	Remove the feature or process that causes the threat.
Transfer	Transfer the responsibility of mitigating the threat to a more capable or suitable party.

Should a threat receive a Control action, one or more Security Objectives are created to address the risk, reducing it to an acceptable level. Likelihood and Impact is reassessed assuming that the mitigations are in place, resulting in a Residual Likelihood (this is the value that usually decreases), a Residual Impact (it is unlikely that this value decreases), and finally a Residual Risk.

A Residual risk of Very Low or Low is accepted. Residual risk values such as Medium or higher are normally further mitigated but in exceptional cases can be accepted, after a discussion with the architects. The rationale must be written down explicitly.

The analysis is completed when all Residual risk levels are accepted.

Note

A precise evaluation of the Residual Impact, Likelihood, and Risk is possible only for those threats that are mitigated by the architecture or transferred to a different sub-architecture of the Arm architecture. If the mitigation of a threat is transferred to an implementer or a user, Security objectives or Functional requirements are captured but it is the responsibility of the transferee to determine the appropriate course of action as per their own risk assessment. Therefore, the residual values reported here should be taken as indicators, and in some exceptional cases, they may even be omitted.

---

4 Architecture definition

4.1 Preliminary threat model and architecture description

4.1.1 Preliminary threat model

With the Secure world software stacks running in TrustZone becoming more complex, it is advisable to partition and sandbox them to increase the security posture of the environment that hosts various Secure world services. Such a mechanism allows running different TEEs side by side without compromising the confidentiality and integrity of each TEE, and at the same time limit their ability to compromise the whole system. For this purpose, Armv8.4 architecture introduced Virtualization extensions in the Secure state. The *Arm® SMMU v3.2 architecture* adds support for stage 2 translations for Secure streams to complement the Secure EL2 translation regime in an Armv8.4 PE.

However, a software architecture that leverages these architectural features is needed and is called the Arm Firmware Framework for Arm A-profile.

Note

The terms Framework, FF-A, and Firmware Framework are used interchangeably within this document.

4.1.2 Characterization of FF-A

Arm Firmware Framework for Arm A-profile enables an ecosystem of vendors to provide software components that can co-exist and run in distinct sandboxes, and can be generalized, reused, or ported across platforms to reduce the software development costs. This is achieved by:

- Leveraging the Virtualization Extensions to isolate software components from each other.

- Standardizing the communication between the isolated software components.

- Generalizing the interactions between a software image and privileged firmware.

This section provides a brief description of FF-A.

Figure 4-1: Software architecture
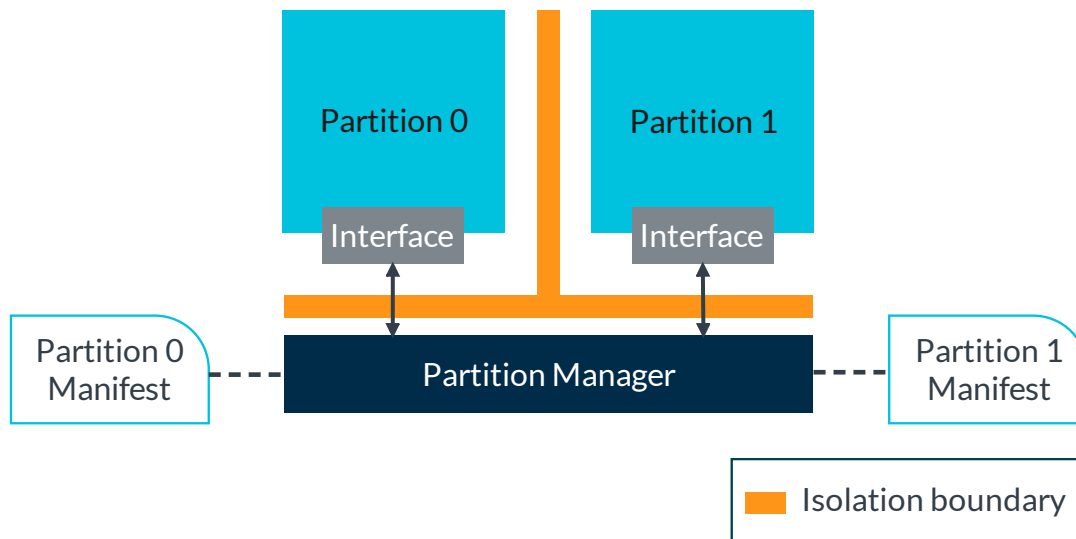
The following are the main components of FF-A:

- Isolation boundaries:

    - A logical isolation boundary that temporally isolates a software image from another, and a physical isolation boundary that spatially isolates a software image from another. For example, an S-EL0 partition is spatially isolated from another S-EL0 partition by S-EL1 or S-EL2 when Armv8.1 VHE is enabled. Two S-EL1 partitions are said to be spatially isolated when Virtualization extensions are implemented and enabled, else they are temporally isolated.

- Partitions:

    - A sand-boxed software image environment that implements one or more services accessible across the boundary only through well-defined interfaces. It is the VM or OS kernel (when virtualization extension is disabled or unavailable) in Normal world and Secure Partition in Secure world.

Note

The term partition is used when it is not required to distinguish between a logical and physical partition. The term endpoint is used interchangeably with the term partition.

- Partition manifest:

  - Describes the physical address space ranges and system resources like ownership of interrupts and memory a partition needs, identity of partition services to enable their discovery and other attributes of the partition that govern its runtime behavior.

- Partition manager:

  - A PM is responsible for creating and managing physical isolation boundary of a partition. The PM uses a partition manifest to assign physical address space ranges and system resources to a partition, initialize it as per the specified attributes and enable discovery of its services. The PM also implements FF-A ABIs to enable inter-partition communication for access to partition services. In the Secure world, this component is called the Secure Partition Manager (SPM). In the Normal world it is a Hypervisor (if the Virtualization Extension is enabled). The term PM is used when it is not necessary to distinguish between SPM and Hypervisor. The term FF-A component is used to collectively refer to partitions and PMs.

  - A PM also creates and manages isolation boundaries for DMA capable devices through an access control mechanism like an Arm SMMU or a vendor-specific System MPU in a system. For Arm SMMU, each transaction generated by an upstream device is associated with a StreamID that determines the Security state of the transaction and the stage 1 and/or stage 2 address translations that must be used for the transaction.

    In the Secure world, the SPMC programs SMMU to limit access to physical address space for DMA generated transactions using a Secure StreamID. In the Normal world, the Hypervisor programs SMMU to limit access to Non-secure physical address space for DMA generated transactions using a Non-secure Stream ID.

    A set of SMMU stage 2 translations maintained by a PM on behalf of a DMA capable device is called a Stream endpoint. Stream endpoints associated with a Secure StreamID are called Secure SEPIDs. Stream endpoints associated with a Non-secure StreamID are called Non-secure SEPIDs.

- Partition interfaces:

  - To keep EL3, S-EL2, and Hypervisor code generic, FF-A defines standard ABIs at the boundaries between the various software components. ABIs are needed between clients of secure services in the Normal world, and providers of services in secure partitions. They are also needed to ensure that no trusted OS specific drivers in the Normal world hypervisor and the SPM, and by consequence EL3 and Secure EL2 are required.

  - The ABIs cover features such as enabling message passing between endpoints, set up and discovery of a secure service in a partition, memory management between endpoints to manage access and ownership of memory, allocation of system resources like CPU cycles for FF-A components to do work, notification mechanism to enable Sender to inform

Receiver about an event with non-blocking semantics, and status reporting interfaces to report execution status of other FF-A ABIs.

- The framework enables synchronous (Direct) and asynchronous (Indirect) message passing between FF-A components. Messages can be transmitted through TX/RX buffers or registers or shared memory.

4.1.3 Deployment models

An example deployment scenario of FF-A architecture on an Arm A-profile configuration with S-EL2 is shown below.

Figure 4-2: FF-A deployment with S-EL2
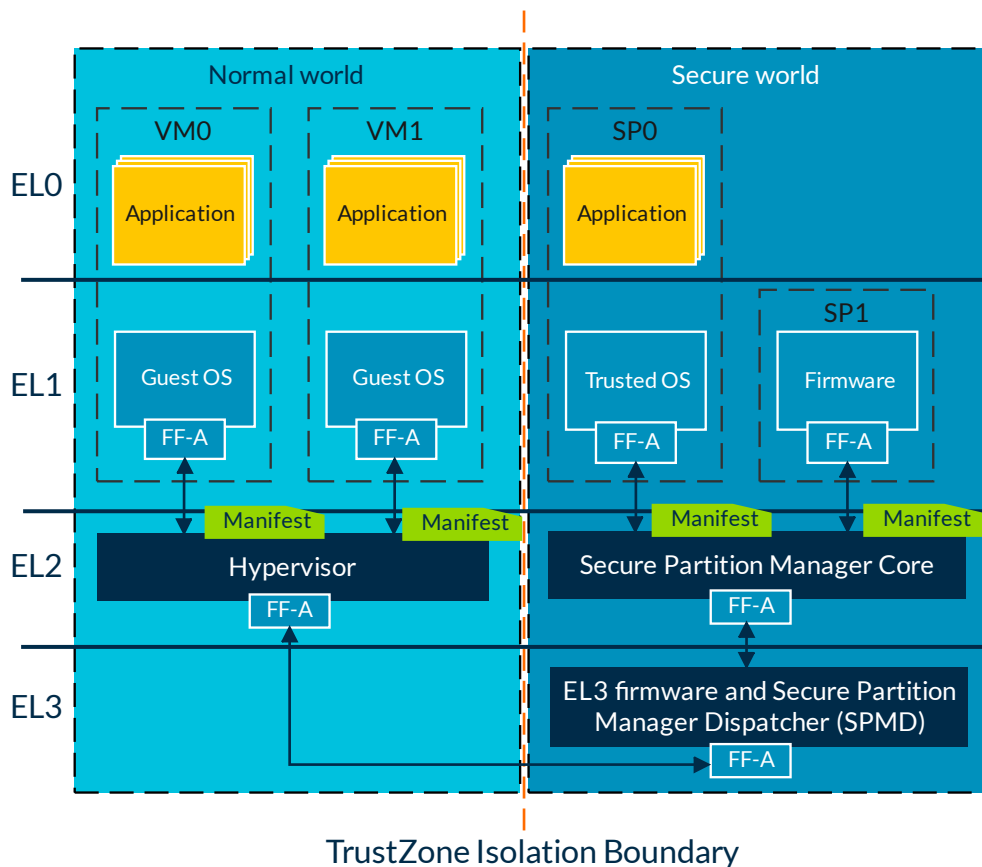
In the Normal world, both VM0 and VM1 implement an FF-A driver in EL1 to access each other's services and the services in S-Endpoints. The Hypervisor facilitates access to services in S-Endpoints from VM0 and VM1 by implementing an FF-A driver in EL2 and also access each other's services.

The following software images are deployed in the Secure world.

- A firmware image in EL3
- An SPM image in S-EL2
- A firmware image in S-EL1 (SP1)
- A Trusted OS image in S-EL1 (SP0)

SP0 and SP1 are physical partitions and could access each other's services using the SPM. The SPM is physically isolated from SP0 and SP1.

FF-A architecture can also be deployed on A-profile systems without S-EL2, with S-EL2 and Armv8.1 VHE. In **Figure 4-2: FF-A deployment with S-EL2**,

- When S-EL2 is unavailable or disabled, either the EL3 firmware image implements both SPMC and SPMD (referred to as SPM) or the SPMC is at S-EL1 with SPMD at EL3. SP0 and SP1 are temporally isolated logical partitions that could access each other's services through SPM. SPM is logically isolated from SP0 and SP1.
- When S-EL2 is available and enabled, Armv8.1 VHE can be additionally enabled to manage S-EL0 SPs. In this configuration, SPMC is at S-EL2 with SP0 firmware image (instead of a Trusted OS and its application) and SP1 at S-EL0 instead of S-EL1. SP0 and SP1 are physically isolated partitions that could access the services of each other through the SPMC at S-EL2. SPMC is also physically isolated from SP0 and SP1.

4.1.4 Life cycle, trust boundaries and information flow

4.1.4.1 Life cycle

A generalized life cycle diagram of the Firmware Framework is shown below.

Figure 4-3: Life cycle diagram of Firmware Framework
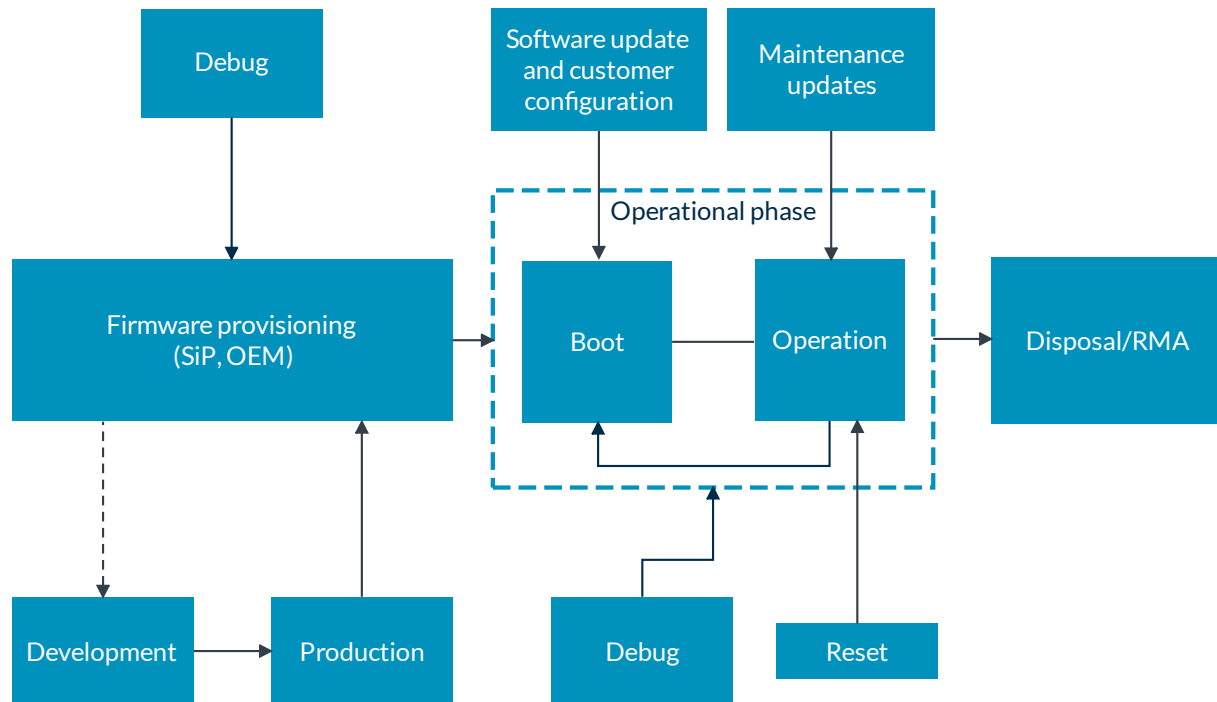
The different phases in the lifecycle are described in the following table. The scope of FF-A is limited to the Operational phase as shown in Figure 4-3: Life cycle diagram of Firmware Framework

Table 4-1: Firmware Framework lifecycle

Phase	Description

Firmware provisioning	SiP
	
	  SiP-owned firmware like SoC power management, CPU errata management, and so on.
	
	OEM
	
	  For example, injection of cryptographic assets. OEM-owned firmware like platform power management, DRAM configuration, and so on.
	
	• Hardware debug ports must be disabled or managed through an access control mechanism.

Phase	Description
Operational phase	Boot • Refers to the stage in the system boot where system initialization is not complete. • Platform services may not be accessed using standard FF-A communication interfaces. • Configuration and isolation of software components may not be fully complete. Runtime • Refers to the post-initialization phase where applications can access platform services or those implemented in other software components. • Uses standard FF-A communication interfaces to access platform services or functionality in other software components. • Configuration and isolation of software components is complete. • System reboot for maintenance-like feature updates, bug fixes, and so on.
Disposal/RMA phase	If a major fault develops during the operational phase, a return to the OEM is required to triage issues. The OEM may need higher privileges to run diagnostics than that available in the operation phase. It should be possible to re-provision a decommissioned platform without compromising the earlier provisioned secrets.

4.1.4.2 Trust boundaries and information flow

The following diagram shows the information flow and trust boundaries in an FF-A based system that uses TX and RX buffers for message exchange.

Figure 4-4: Information flow and trust boundaries in an FF-A based system
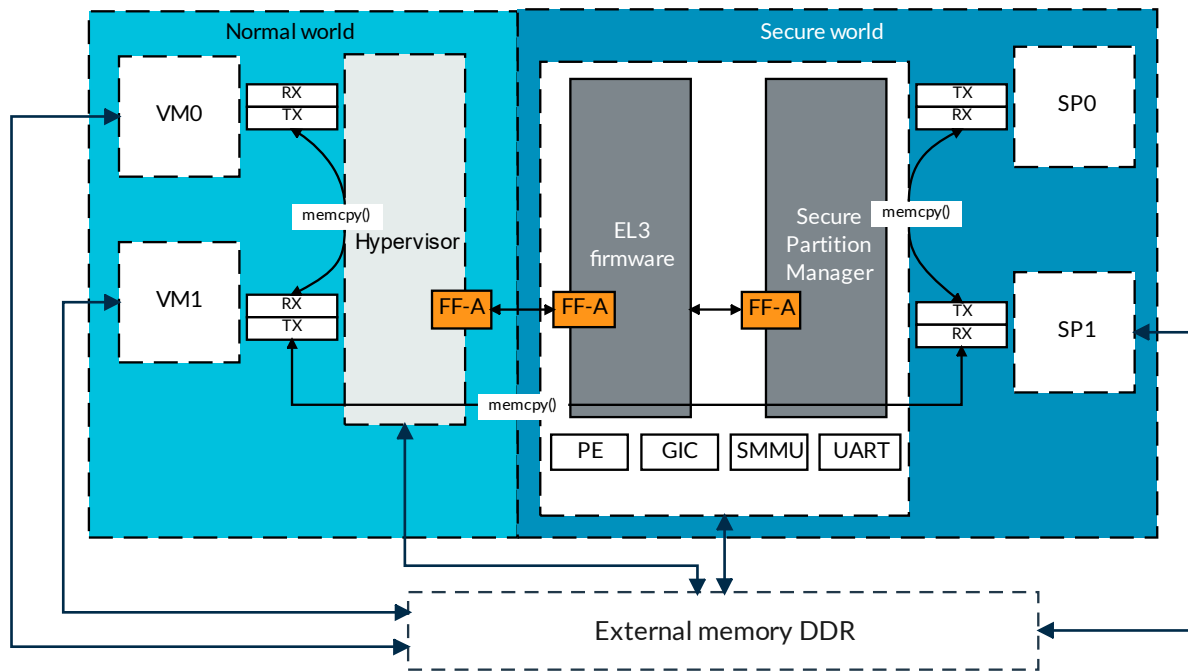
The different aspects of the data flow and trust boundaries are described below:

- The EL3 firmware and SPM are completely trusted in this model. Also, hardware IPs like PE, SMMU, and GIC are also trusted. External RAM is outside this trust boundary.

- Interactions within the Secure world:

  o The EL3 firmware and SPM do not trust SPs, hence the necessity for an additional trust boundary around EL3 firmware and SPM.

  o SPs trust EL3 firmware and SPM.

  o SPs are in different trust boundaries from each other.

- Interactions between the Secure world and the Normal world:

  o The Secure world and the Normal world are in different trust boundaries. The Normal world is untrusted.

  o Components in the Normal world are isolated from the components in the Secure world using Arm TrustZone Security Extensions implemented by the CPU and system architecture.

  o The Hypervisor trusts EL3 firmware and SPM. It also trusts the Trusted OS when S-EL2 is not implemented or disabled.

- Interactions within the Normal world:

    o   The Hypervisor does not trust VMs.

    o   VMs trust Hypervisor, EL3 firmware, and SPM. They also trust Trusted OS when S-EL2 is not implemented or disabled.

    o   VMs are in a different trust boundary from each other.

4.1.5 Attack surface

For FF-A, the attack surface is any FF-A component, that is, endpoints like VMs and SPs, the Normal world Hypervisor, and the SPM. The attack vectors include:

- ABIs

- Registers used for input or output parameters in ABIs, that is, general-purpose registers, FPU or Neon registers

- Private register state of an FF-A component

- Memory that is either private to an FF-A component or shared between two or more FF-A components

4.1.6 Assumptions, constraints, and interacting entities

1. FF-A assumes that VMs and SPs are statically provisioned in the system such that they are loaded and initialized during system boot.

2. It is assumed that firmware provisioning phase happens in a secure environment.

3. As the scope of FF-A is only at Operational phase, security risk analysis in Firmware provisioning and Disposal/RMA phases is outside the scope of this document.

4. The firmware images are assumed to run from either ROM or on-chip trusted SRAM so tampering or probing with off-chip memory is not possible.

5. It is assumed that Secure boot is enabled and so an adversary cannot boot arbitrary images that are not approved by system providers.

6. Measured boot might or might not be enabled. Neither the threats nor the associated mitigations are in scope.

7. Some systems running FF-A may require protections against some physical attacks. Examples include:

    - Passive non-invasive attacks like side-channel attacks that include differential power analysis, electromagnetic, photonic, acoustic, or other similar emissions measured by hardware probing

- Semi-invasive attacks in which systems are subjected to fault injection through optical techniques

A security risk assessment of the system is needed to evaluate the protection against these attacks. Mitigations are dependent on the system implementation and are considered outside the scope of this analysis.

8. Advanced or invasive physical attacks in which systems are unpackaged and subjected to microprobing are outside the scope of analysis.

9. The list of interacting entities which are outside the scope of analysis include:

- Software and interfaces such as memory allocation schemes and conduits for message passing that are used by the FF-A.

- Services that run inside the partitions. While the services that rely on the chosen message passing method are not themselves under review, the message payload exchange method might impact the security requirements of the services and additional mitigations might need to be implemented.

4.2 Stakeholders and Assets

4.2.1 Stakeholders

A stakeholder is a person or group with a sense of property and volition that places value in the system. The following stakeholders are identified in systems which use FF-A to meet the requirements described in the **Preliminary threat model and architecture description** section:

Table 4-2: Stakeholder and its rationale

Stakeholder	Rationale
SiP (Silicon Provider)	To protect their firmware from being misused by other software components in the system.
System integrator. For example, OEM.	To protect their components (mostly Secure world software) from being abused by other software components in the system.
Secure service vendors	To protect their content from being abused by other components in the system. For example, a mobile wallet or payment service that processes credit card information of a customer.
Normal world Hypervisor or OS vendor	Same as the OEM and when this may not always be the case, this document differentiates between the two. They have an interest in protecting their components (mostly Normal world software) from being abused by other software components in the system.
Tenants	Same as the system owner, but there is a chance they may not be, such as in the cloud environment. This document does not differentiate between the two. The end users of the system want to protect their private data while accessing access services in the hosting environment.

Stakeholder	Rationale
Arm	Any damage to a system using FF-A can have implications on Arm's reputation and revenue.

4.2.2 Assets

Assets include:

- Internal state of partition manager like SPM and Hypervisor. Stakeholders for this asset include Arm, OEM, and System integrator.

- Internal state of partition like SP in Secure world or VM in Normal world. Stakeholders for this asset include OEM, SiP, Trusted OS, and Secure service vendors.

- Scheduling cycles. Stakeholders for this asset include tenants or device users, Trusted OS, and Secure service vendors.

Note

Information like cryptographic keys, certificates, audit logs, attestation data (boot stats, image signature), vendor-specific measurement, key/credential storage, Hardware RoT data like Hardware Unique Key (HUK), Root of Trust Public Key (ROTPK) which is the public key half of an asymmetric key pair, and so on are auxiliary assets that, if compromised, might lead to breach of the goal of protecting the confidentiality, integrity or availability property of the main assets. Similarly, information exchange between endpoints and shared memory resources like TX and RX buffers.

Also, not all stakeholders are interested in every aspect of an asset. For example, a stakeholder might require that the integrity of an asset be upheld but does not need that the asset be confidential.

4.3 Security goals

The security goals of this analysis can be formulated together with the *baseline impact* of breaching them. The actual impact of a threat to a security goal may be modified by the analyst with respect to the baseline impact depending on threat-specific aspects such as scalability, ability to mitigate, and/or bandwidth of exfiltration, rate of corruption, and effectiveness of access-denial.

SG 1. *An adversary must not be able to prevent Secure Partitions to run or run at the intended performance.*

Baseline impact of breaching this goal: **Medium.**

SG 2. *An adversary must not be able to breach the confidentiality of the code or the data of a target partition.*

Baseline impact of breaching this goal: **High.**

SG 3. *An adversary must not be able to breach the integrity of the code or the data of a target partition.*

Baseline impact of breaching this goal: **Medium.**

SG 4. *An adversary must not be able to breach the confidentiality of the code or the data of the partition manager. This includes hardware keys.*

Baseline impact of breaching this goal: **Very High.**

SG 5. *An adversary must not be able to breach the integrity of the code or the data of the partition manager. This includes hardware keys.*

Baseline impact of breaching this goal: **High.**

The Framework also goes beyond the stated goals to ensure that the interfaces can be used to standardize communication:

- In the absence of the Virtualization Extensions in the Secure world. This provides a migration path for existing Secure world software images to a system that implements the Virtualization Extension in the Secure state.

- Between VMs managed by a Hypervisor in the Normal world. The Virtualization Extensions in the Secure state mirrors its counterpart in the Non-secure state. Hence, a Hypervisor could use the firmware framework interfaces to enable communication between the VMs it manages.

> **Note**
>
> While the availability of service running inside a partition is important in some market segments, FF-A does not provide comprehensive mechanisms to meet this requirement.

4.4 Adversarial model

Adversarial models are descriptions of the adversaries that the FF-A architecture is intended to defend against, grouped into classes. The models are derived from the market requirements and then mapped to one or more of the four adversarial models identified by Arm:

AM 0. *The adversary is only capable of accessing data that requires neither physical access to a system containing an implementation of the architecture nor the ability to run software on it. This*

adversary is intercepting or providing data or requests to the target system through a network or other remote connection.

For instance, the adversary can:

- *Read any input and output to the target through external devices*
- *Provide, forge, replay or modify such inputs and outputs*
- *Perform timings on the observable operations done by the target machine, either for normal operation or as a response to arranged inputs (such as timing attacks on web servers)*

AM 1. *The adversary can mount attacks from software running on a target device implementing the feature. This type of Adversary can run software on the target.*

For instance, the adversary can:

- *Attempt software exploitation by running software on the target,*
- *Exploit access to any memory mapped configuration, monitoring, debug registers,*
- *Mount any side channel analysis that relies on software-exposed built-in hardware features to perform physical unit measurements and time measurements, and*
- *Perform software-induced glitching of resources such as Rowhammer, Raspberry or crashing the CPU by running resource-intensive tasks.*

AM 2. *The adversary can mount passive hardware attacks that do not require physical breaching of the chips.*

This type of adversary, as well as the following ones, has access to a system (such as a computer, a smartphone, and others) containing implementations of the target feature.
In particular such adversarial models include access to debug ports, signals, communication buses or ports and the memory bus, for instance through insertion of malicious hardware including chip interposers. The malicious hardware serves to eavesdrop transactions, communications, signals.

It also includes other simple, reversible modifications to the system.
The following are some examples:

- *Side channel analysis that requires measurement devices (including any leakage source such as electromagnetic emissions, power consumption, photonics emission, acoustic channels, and others),*
- *Plugging malicious hardware in the system without its modification,*
- *Gaining access to the internals of the target system and interposing the SoC or memory for the purposes of reading, blocking, replaying, and injecting transactions, and*
- *Replacing or adding chips on the motherboard.*

AM 3. *The adversary can mount active hardware attacks and fault injection attacks that do not require physical breaching of the chips. The adversary has the same type of access as in model* **AM 2,** *where the malicious hardware serves not only to eavesdrop transactions, communications, signals but also to block, inject, corrupt or replay them.*

The modifications to the system described in model **AM 2** *apply to this AM as well, but they may be irreversible.*

AM 4. *The adversary performs invasive SoC attacks.*

Threats that can be mounted by this type include, for instance:

- *Chip decapsulation through laser or chemical etching, followed by microphotography to reverse engineer the chip*
- *Use a focused ion beam microscope to perform gate-level modification*

AM 4 is out of scope of FF-A.

Mitigations against threats that can be mounted by adversaries **AM 1**, **AM 2**, and **AM 3** are not necessarily provided by the architecture and are transferred to implementers and system integrators. It is up to them whether to accept or control (mitigate) these threats.

SW agents under control of the adversary include any component running in the NS world (including FF-A components) or endpoints in Secure world, such as software running in S-EL0 and S-EL1 exception levels.

---

5 Threats and mitigations

This section details threats to FF-A and provides examples of attacks. They are derived by applying STRIDE analysis on each element of the Trust boundaries and Information flow diagram as described in **Life cycle, trust boundaries and information flow** section.

Note

Threats associated with the payload of partition messages are not listed as this is specific to a service or partition implementation. For example, a legitimate, more privileged borrower called confused deputy, is tricked by a malicious endpoint to read/modify a memory region (privilege escalation) that it neither owns nor shares with the owner of the memory region, or a malicious endpoint uses payload of a partition message to exploit vulnerabilities in bound checks in the receiver endpoint and mount ROP style attacks.

5.1 Spoofing message sender or receiver

Description: A malicious endpoint impersonates the identity of a message sender or receiver to gain unauthorized access or interfere with its execution environment. For example, a VM may spoof the identity of an SP in its direct message request/response to a targeted SP. For DMA capable devices, a dependent device that is upstream of an access control mechanism like Arm SMMU may spoof the identity of an independent device (through its proxy endpoint manifest) to cause memory management operations to update stage 2 translation tables in SMMU without the consent of its proxy endpoint (privilege escalation).

Unmitigated Impact	High
Unmitigated Likelihood	Very High
Unmitigated Risk	Very High
Mitigating Action	The threat is controlled by endpoint identification field in the manifest, endpoint identity validation responsibilities of the relayer as described in the architecture. The threat must also be mitigated by transferring to the implementations of FF-A or system integrators by requiring them to perform endpoint identity checks in SPM for transactions originating from Normal world as it is untrusted.
Residual Impact	High
Residual Likelihood	Very Low
Residual Risk	Low

5.2 Tampering with the shared memory between FF-A components

Description: A malicious endpoint may tamper with the shared memory while it is being accessed and processed by another FF-A component. For example, a malicious endpoint or Hypervisor may alter message payload while the SPM reads and processes the same portion of data or metadata more than once from a shared memory region like TX/RX buffer. It is also possible that the RX buffer of a VM may be tampered due to resource contention. For example, the Hypervisor transmits a message response from VM0 to VM1 and the SPMC transmits a message from SP0 to VM1 simultaneously.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture. Implementations of FF-A are required to read transmission data only once, and if necessary, make a local copy before processing or if the data needs to be referenced multiple times. The architecture also provides mechanisms to define and manage ownership (for example, FFA_RX_ACQUIRE and FFA_RX_RELEASE ABIs), state, and access that must be followed by the producers and consumers of TX/RX buffers.
	The threat is also transferred to FF-A implementations when there are multiple producers, or consumers of the TX/RX buffers, or multiple instances of the same producer or consumer. For example, the SPM might use spinlock mechanism to protect TX/RX buffers from concurrent accesses of its own instances running on different PEs.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.3 Tampering with the firmware images

Description: Firmware images and sensitive configuration like manifest data on the system may be subjected to unauthorized version rollback to exploit a version with a legacy bug.

Unmitigated Impact	Very High
Unmitigated Likelihood	High
Unmitigated Risk	Very High
Mitigating Action	The threat is transferred to system integrators and silicon vendors. Preventing rollback to a known version with security vulnerabilities is essential. However, rollback

for recovery purposes may be allowed if authorized. Silicon vendors must implement anti-rollback protection, for example using non-volatile counters.

Residual Impact	Very High
Residual Likelihood	Very Low
Residual Risk	Low

5.4 Tampering with owned state or private state of other FF-A components

An endpoint may tamper with its own state or the state of another endpoint. Following are some threats and their descriptions:

5.4.1 Out of order FF-A messages, improper setup and initialization of partitions, and incorrect interrupt management

Description:

- A malicious endpoint may invoke FF-A messages out of order to cause unexpected state changes to the SPMC or another endpoint like SP.
- An endpoint sends FF-A messages to another endpoint that is not yet initialized by the framework and unable to process messages. For example, an SP sends a direct request to the normal world early while the normal world is not booted yet.
- Spurious injection of interrupts or incorrect routing may force a partition into an inconsistent state, potentially leading to data corruption and further, as of yet UNKNOWN, consequences.
  - o An example of a potential issue arising from enacting this threat is related to affinity of interrupts. Interrupts are typically configured affine to a specific CPU, and can rely on local IRQ masking to provide mutual exclusion for data structures used by the IRQ handler. Injecting the interrupt to a different CPU permits the IRQ handler to unexpectedly race with other code, potentially causing data corruption.
  - o Also, a malicious Normal world endpoint may repeatedly interrupt long running memory management transactions in an attempt to corrupt the state of the SPMC that is processing these transactions.
  - o It is also possible for a malicious Normal world endpoint to generate repeated interrupts targeting an SP performing managed exit in an attempt to corrupt the state of SP or cause DoS by forcing SP to perform recursive managed exits.

Unmitigated Impact	High
Unmitigated Likelihood	Medium

Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture. It provides guidance on partition setup and initialization, partition run-time states and state transitions, run-time models and interrupt management. FF-A ensures that any execution state like registers, memory, ownership and access attributes associated with every memory region in physical address space between ABI calls are consistently maintained and only valid state transitions will be permitted. For interrupts, FF-A ensures that direct injection of virtual or physical interrupts into partitions does not happen. It also requires interrupts to be masked while a managed exit is in progress.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.4.2 Software-induced fault injection on sensitive data structures

Description: Sensitive data structures like state machine that tracks ownership/access of memory regions in memory management transactions, metadata of partitions like manifests, runtime state of SPM and other sensitive data should be protected from unauthorized access and modification.

Since all these data structures are stored in Secure memory by SPM, and SPs do not have mappings to them and VMs in Normal world cannot access Secure memory due to TrustZone architecture, the threat model considered here is DRAM corruption using RowHammer family of attacks. RowHammer allows a user-level program to reliably and consistently induce disturbance errors in vulnerable DRAM modules.

Initially deemed to be a threat only to system integrity and availability, research has shown that RowHammer disturbance errors can be leveraged to mount attacks that leak data from other processes, thus directly violating confidentiality (RAMBleed attack).

Unmitigated Impact	High
Unmitigated Likelihood	High
Unmitigated Risk	High
Mitigating Action	This threat is transferred to silicon vendors, system integrators, and implementations of FF-A. Integrity protection and encryption with temporal freshness is required on any system without tamper-resistant memory. However, this is outside the scope of FF-A as it depends on the underlying hardware.
	Considering the above constraint, performing simple integrity checks (for example, CRC, cryptographic hash functions) of sensitive data structures in software is highly recommended. The exact list of these sensitive data structures and the details of integrity checks may need to be determined by the implementations of FF-A and system integrators.

Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.4.3 Altering counters/clocks used in SPs

Description: A compromised SP may try to stop/alter the passage of time as seen by a victim SP if it has access to CNTControlBase region.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to implementations of FF-A which must ensure that system counter accesses be only performed from EL3.
Residual Impact	Medium
Residual Likelihood	Low
Residual Risk	Low

5.4.4 Exploiting power management operations

Description: A malicious Non-secure endpoint may issue power management operations like turning off a CPU, suspend CPU execution, and so on to cause unexpected changes to an execution context running on the targeted PE. When the power management operation is invoked on the targeted PE, it is possible that the execution context is pinned to this PE, or this PE is the last one in the system to be powered off.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the Trusted OS vendors. SPs must be robust enough to cope with the power down of the PE.
Residual Impact	Medium
Residual Likelihood	Medium
Residual Risk	Medium

5.4.5 Misuse of FFA_MEM_PERM_GET and FFA_MEM_PERM_SET ABIs

Description: Usage of FFA_MEM_PERM_GET and FFA_MEM_PERM_SET ABIs by an S-EL0 SP beyond its initialization may allow an attacker to alter MMU mappings and compromise the S-EL0 partition.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture by requiring the callee to invoke FFA_ERROR if these ABI are invoked by an SP post its initialization or at any time on a secondary PE.
Residual Impact	Medium
Residual Likelihood	Very Low
Residual Risk	Very Low

5.4.6 Hardware fault injection attacks

Description: Active non-invasive attacks like fault injection attacks that involve power, clock, temperature, and energy glitches can cause faults such as instruction skipping, data corruption while reading from or writing to memory or instruction decode errors. An adversary with physical access to the system can mount such attacks during normal execution flow, for example during validation of firmware image against a signature, bypass authentication checks and execute arbitrary code on the system.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	This threat is transferred to silicon vendors and system integrators. They must implement physical tamper detection and prevention mechanism.
	However, implementations of FF-A can use software techniques, such as adding redundant checks when performing conditional branches that are security sensitive, to harden against such attacks.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.5 Information leakage through audit logs

Description: Sensitive information like audit logs that include confidential information like crash reports with details about PE state or stack dumps might be present in the system to triage issues during firmware development phase as described in the **Life cycle** section. An adversary might use this information along with any other logging mechanisms that leak sensitive data of firmware or partitions to develop a working exploit if they are present in production releases.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the implementations of FF-A. Ensure sensitive logging information is removed from production releases and that the release versions are a lot less verbose than development builds.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.6 Invalid or incorrect arguments to FF-A ABIs

Description: A malicious endpoint may exploit memory corruption vulnerability when accessing system resources in an attempt to execute arbitrary code, tamper with the state of the partition to change execution flow or leak sensitive information. FF-A ABIs take a number of input arguments and return responses to various transactions. Memory corruption vulnerabilities may arise due to lack of boundary checks, memory overflow errors like integer overflow, buffer overflow, incorrect array bound checks, and incorrect error responses. For example:

- A malicious sender might provide a pointer to memory that it does not own or have access to, causing the receiver to read data from this location and return it to the sender. For memory management transactions, the malicious endpoint can provide a buffer length greater than the supplied input buffer value or page count greater than the supplied pages, causing the receiver to read past the end of the memory, thereby potentially reading data that does not belong to the sender.
- Transmission of a memory transaction descriptor through dynamic buffer requires a Hypervisor or SPM or both to make changes to their translation tables. A malicious endpoint might exploit this mechanism to tamper with the translation system managed by the relayer by specifying incorrect memory granularity or alignment requirements.
- A malicious sender might not use the same shared resource while transmitting the message payload to cause unexpected changes in the partition manager state machine that tracks the

transaction or the state of the receiver. For example, transmitting some fragments of the memory transaction descriptor through Tx buffer and others through dynamic memory. Otherwise, using Tx buffer to transfer payload in FFA_MEM_RETRIEVE_REQ and dynamic memory to transfer payload in FFA_MEM_RETRIEVE_RESP, and so on.

- An S-EL0 adversary might supply unaligned address to FFA_MEM_PERM_GET ABI to extract access permissions of S-EL1/S-EL2 (when VHE is enabled) and track its execution flow.

Unmitigated Impact	High
Unmitigated Likelihood	High
Unmitigated Risk	High
Mitigating Action	The threat is controlled by the architecture. It requires all input parameters of an ABI be validated to ensure that they conform to the correct format, type, field values, size, alignment, and length associated with the call being made. It must do this before any processing occurs. It also ensures that the ABIs are designed using the smallest possible input to reduce potential attack surface for malformed data within that input.
	SPM must sanitize any inputs coming from outside its TCB.
	If input parameters to an ABI correspond to memory locations, pointers or handles to memory regions or buffers, the FF-A shall ensure that they are appropriately sized and that the ownership, access attributes, security state of the memory region permit the call to make forward progress. It must do this before any processing occurs.
	FF-A shall generate an error if any input parameter validation fails. Implementations of FF-A must implement support for handling all errors that can be returned on completion of ABIs.
Residual Impact	High
Residual Likelihood	Very Low
Residual Risk	Low

5.7 Exploiting debug and trace logic to compromise confidentiality and integrity of FF-A components

Description: An adversary may read sensitive data and execute arbitrary code through the external debug and trace interface. Arm PEs include hardware-assisted debug and trace features that can be controlled without the need for software running in the system. If left enabled without authentication, this feature may be used by an attacker to inspect and modify register and memory state, allowing the attacker to read sensitive data and execute arbitrary code.

Unmitigated Impact	High
Unmitigated Likelihood	Very High

Unmitigated Risk	Very High
Mitigating Action	The threat is transferred to system integrators. Either the debug and trace capabilities shall be disabled in production systems or debug authentication mechanisms shall be implemented and enabled.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.8 Leaking sensitive data of the partition manager (relayer) due to improper message processing

Description: If the payload by a sender is modified by the relayer before it is dispatched to the intended endpoint, the sender can see the modified payload in its TX buffer. For example, when a VM sends a message to an SP, the Hypervisor must convert IPAs to PAs before forwarding the message to SPM. This can leak PAs to the sender if the relayer edits the payload in the sender's TX buffer.

Unmitigated Impact	Very High
Unmitigated Likelihood	Medium
Unmitigated Risk	High
Mitigating Action	The threat is controlled by the architecture. The partition manager or the relayer is required to make a local copy of the payload before processing such messages.
Residual Impact	Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.9 Leaking sensitive data of an endpoint

5.9.1 Exploiting direct message response

Description: A malicious endpoint may trick a victim endpoint to reveal its internal state through the direct message response. The victim SP or SPMC replies to a partition message by a direct message response with information which may reveal its internal state, for example, a response outside the allowed error encodings.

Unmitigated Impact	High

Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture. It expects implementations of FF-A, especially SPMC, to sanitize responses from SPs to FF-A message requests. The threat is also transferred to the Trusted OS vendors as they are expected to harden their code to ensure sensitive state is not leaked through the direct message response.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.9.2 Probing the communication between FF-A components

Description: A malicious agent with physical access may use non-invasive methods to probe the external memory bus and extract the traffic between an SP and the SPMC or among SPs when shared buffers are held in external memory.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the silicon vendors or system integrators to protect the confidentiality of DRAM contents.
Residual Impact	High
Residual Likelihood	Medium
Residual Risk	Medium

5.9.3 Transient execution attacks

Description: A malicious endpoint may attempt to reveal the state of SPMC or SP by the use of software-based cache side-channel attack techniques as described in *Cache Speculation Side-channels whitepaper*.

Also, an attacker may exploit the lazy implementation of context switching in FF-A implementations, where the FPU context is not saved until a different process (can be a different TA under a TOS, different TOS under S-EL2 SPMC or a Security state switch through EL3) performs an FPU operation that reads or writes an FP register. When the FP operation occurs, an exception is raised for TOS/S-EL2 SPMC/EL3 to perform the context save. However, during the time until the instruction is retired, the attacker might speculatively read FPU register data from the context of a previous process. This is

not a problem if the underlying hardware implements architectural measures to limit the effects of speculative execution-based side-channel attacks.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the silicon vendors, implementations of FF-A, and Trusted OS vendors.
	Do not allow Lazy FPU (potentially applicable to SVE and SME state) in FF-A implementations, especially if the underlying hardware has not implemented cache speculation side-channel related security features introduced in Arm v8.5. The SPMC may be hardened further with software mitigations, for example speculation barriers, for the cases not covered in hardware. For non-hardened cores, the usage of techniques such a kernel page table isolation can help mitigating Meltdown type of attacks.
	In case lazy stacking of any internal CPU state is implemented, architectural measures in hardware and software must be implemented to limit the effects of speculative execution-based side-channels. If these are exposed through ABI, their use is required in the system.
Residual Impact	Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.9.4 Cache-based side-channels

Description: An attacker in Normal world or from a compromised SP can leak sensitive information like code flow, memory accesses of a victim SP through cache timing attacks like Prime and Probe, Flush and Reload (on shared memory between attacker and victim), Evict and Reload, Evict and Time, and so on.

Unmitigated Impact	Medium
Unmitigated Likelihood	High
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the silicon vendors, system integrators, Trusted OS vendors, and implementations of FF-A.
	Requires implementations of FF-A to use constant-time and uniform algorithms with no secret-dependent memory access patterns.
	Specifically for software implementations on the Arm architecture, the Armv8.4-DIT Data Independent Timing feature as described in *Arm Architecture Reference Manual for A-profile* should be used. In both hardware and software implementations, constant

time and uniform implementations are necessary but not sufficient to mitigate against all types of timing attacks. Effective cache-based side channel mitigations must be implemented in hardware of the CPU microarchitecture.

Residual Impact	Medium
Residual Likelihood	Medium
Residual Risk	Medium

5.9.5 Interrupt-based side-channels

Description: A Normal world attacker may repeatedly interrupt the execution of a security-critical routine in an SP to infer its secrets. For example, by measuring the latency of a carefully timed interrupt, a malicious endpoint can infer instruction-granular execution state from an SP.

Unmitigated Impact	Medium
Unmitigated Likelihood	Low (There is extra latency incurred to enter and exit SP at each interrupt and an interrupt must be injected precisely at the point after entry to SP and at the point the first instruction begins executing to effectively time the length of that instruction.)
Unmitigated Risk	Low
Mitigating Action	The threat is transferred to the Trusted OS vendors and implementations of FF-A. Requires implementations of FF-A to use constant-time and uniform algorithms with no secret-dependent memory access patterns, and making interrupt requests constant time.
Residual Impact	Medium
Residual Likelihood	Low
Residual Risk	Low

5.9.6 Using architectural counters for side channels

Description: An adversary can use AMU, PMU, or other statistics to further increase the accuracy of side-channel attacks. Such statistics include fine-grained counters which significantly aid in timing-reliant side-channel efforts. For example, a Non-secure world attacker may read AMU and PMU counters through a memory mapped interface to infer execution flow and memory access patterns of a Secure world partition. While this is not a threat in itself, it boosts the efficiency of other threats.

Unmitigated Impact	Medium
Unmitigated Likelihood	Very High
Unmitigated Risk	High

Mitigating Action	The threat is transferred to the implementations of FF-A. Implementations of FF-A to use features in the base hardware architecture (for example, MDCR_EL3.EPMAD as described in *Arm Architecture Reference Manual for A-profile*) and save/restore of the state of these registers on context switch to limit access to these counters from Non-secure state or from other SPs.
	For AMU, implementations of FF-A may either disable the leaky AMU monitors on SP execution or the SP must accept that the leaky counters are visible through the memory-mapped interface (for example, from a compromised Hypervisor).
	When PMUv3 is supported in the underlying hardware implementation, secure word event counting is enabled when external debug is enabled. In such cases, system integrator has to ensure external debug is disabled in production devices or proper debug authentication is in place.
	Note that performance counters can still be used by NS-EL2 to increase the accuracy of other side-channel attacks, for instance, by using them, instead of more noisy time measurements, to detect own cache misses or branch mispredictions caused by an SP.
Residual Impact	Medium
Residual Likelihood	Very Low
Residual Risk	Very Low

5.9.7 Timing side-channel attack to break Kernel ASLR

Description: An S-EL0 adversary tries to get or set the memory attributes of kernel address space using FFA_MEM_PERM_GET or FFA_MEM_PERM_SET ABI and measures the timing of the error response to break the kernel address space layout randomization (ASLR), an OS security feature against software vulnerability exploitation, allowing the adversary to map the base address of kernel (S-EL1 or S-EL2 when VHE is enabled) modules and drivers thereby enabling the exploitation of memory corruption bugs in kernel address space.

For example, an implementation of the ABI may use address translation instructions or soft translation table walk to determine whether the caller is allowed to access the memory region the address (supplied to the ABI) lies in. The time taken to complete (TLB hit vs miss or cache hit vs miss of soft translation table walk) is an indication whether the supplied address belonging to S-EL1 or S-EL2 (when VHE is enabled) is accessed by the privileged software. In addition to this, the EL0 adversary may monitor TLB hit/miss associated with the address translation process using PMU and deduce whether the address is accessed by privilege software or not.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium

Mitigating Action	The threat is transferred to the implementations of FF-A. If the access is prohibited, the ABI returns an error response, and should take the same time to send to the caller, whether the address translation is taken from various caching structures like TLB and caches or not, to mitigate attacks that use error timing.
Residual Impact	High
Residual Likelihood	Low
Residual Risk	Low

5.10 Denial of service attacks

5.10.1 Denying an endpoint to raise service requests and denying a service to other endpoints

Description: A malicious endpoint may flood SPMC with multiple requests targeting a service within a receiving endpoint to keep it busy from raising its service requests. Similarly, the malicious endpoint may target a service within a receiving endpoint to keep it busy from serving requests from other endpoints. Multiple endpoints may collude to mount a Distributed Denial of Service (DDoS) attack by flooding a service with malicious messages to cause exhaustion of resources used in the communication framework.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the implementations of FF-A. Implementations shall ensure that their resources are not exhausted or over-utilized by an endpoint or a few endpoints, to guarantee Reliability and Availability of Services. This can be achieved by:

- Providing fair access of CPU cycles to receivers for servicing requests.
- Providing mitigations for resource contentions like shared memory, TX/RX buffers for message transmission.
- Providing arbitration among multiple requests targeted to specific endpoints.
- Managing resources like memory used for stack or heap in the partition manager to handle multiple message requests.

Residual Impact	Medium
Residual Likelihood	Low
Residual Risk	Low

5.10.2 Starvation on TX/RX buffers due to long running operations

Description: A long running memory management transaction may keep an endpoint's Tx/Rx buffers busy and the endpoint or the partition manager may not tolerate the latency in acquiring access to these buffers.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture through an optional feature that allows endpoints to perform memory management transactions through dynamic buffers.
Residual Impact	Low
Residual Likelihood	Low
Residual Risk	Very Low

5.10.3 Delaying an endpoint progress due to long running operations

Description: A legitimate endpoint may not be able to tolerate latency to service a pending interrupt due to long running memory management transactions. Some of these transactions involve a relayer to map and unmap large memory regions in translation tables and issue necessary TLB maintenance operations to synchronize these changes.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture by providing an optional feature to time slice long running memory management transactions to enable the calling partition to make forward progress.
Residual Impact	Medium
Residual Likelihood	Low
Residual Risk	Low

5.10.4 DoS attacks due to fatal error in endpoints

Description: In a memory management transaction, if a borrower of a memory region terminates due to fatal error condition, it may block the lender from making progress as the memory that is being shared or lent to is not released.

Also, in direct messaging, if a sender crashes after sending a request, the receiver may respond through FFA_MSG_SEND_DIRECT_RESP repeatedly unless it is made aware of this situation. This in turn can keep the receiver busy and deny servicing requests from other endpoints.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture by ensuring a resilient state even in case of failures. When a VM or an SP terminates due to fatal error, FF-A shall require that the ownership and access of their memory regions are transferred to Hypervisor or SPM. Similarly, if the Hypervisor terminates due to fatal error, the ownership and access of its memory regions are transferred to SPM.
	The framework defines "ABORTED" error encoding to indicate to the caller of FFA_MSG_SEND_DIRECT_RESP ABI that the receiver endpoint ran into an unexpected error and has aborted. This enables the caller to make forward progress with the help of its partition manager.
Residual Impact	Low
Residual Likelihood	Low
Residual Risk	Very Low

5.10.5 Deadlocks due to call chain loop

Description: For example, the scenario is possible when VM0 sends a request to VM1, VM1 sends a request to VM2 to process the VM0 request and VM2 needs a service implemented in VM0 to complete the VM1 request. Similarly, a call chain loop is also possible amongst SPs in Secure world.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture by providing guidance on detection and prevention of call chain loops that can cause deadlocks in the system.
Residual Impact	Very Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.10.6 Using interrupts to mount DoS attacks on SPs

Description: A Normal world endpoint interrupts a Secure world service to mount DoS attack by:

- Repeatedly preempting its application threads,

- diverting CPU cycles allocated to an SP execution context by its scheduler to process requests from another endpoint, and

- denying a SP from receiving critical events like an OS issuing a power state transition event.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture through Managed exit.
	During preemption, Managed exit ensures that an SP:
	• manages its internal state before relinquishing control
	• receives critical events on time
	• uses CPU cycles to process the intended request
Residual Impact	Very Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.10.7 DoS attacks on Normal world endpoints by an SP

Description: A compromised SP performing a managed exit may indefinitely delay delivery of critical Non-secure interrupts to Normal world and cause DoS attacks in the system.

Many subsystems of an OS rely on timer interrupts to function correctly. For example, periodic timer interrupts used for scheduling are not expected to be delayed for many scheduling intervals. Timers can also be used to signal execution of periodic integrity checks which, when masked, could limit the effectiveness of the check or prevent it entirely.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the implementations of FF-A.
	A Normal world interrupt that occurs during a Secure world execution must be serviced as early as possible. The SPMC can impose an IMPLEMENTATION DEFINED timeout within which an SP must complete the managed exit. The SPMC can take an IMPLEMENTATION DEFINED action if the timeout expires before the managed exit is completed.
Residual Impact	Medium

Residual Likelihood	Medium
Residual Risk	Medium

5.10.8 Using RAS error injection to DoS Normal world

Description: A compromised SP could use RAS error injection to constantly raise RAS exceptions on the Normal world. This can cause DoS in Normal world which could be a problem for the system owner.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the implementations of FF-A. RAS control registers should not be made available to SPs. FF-A implementers (through SPMC) must ensure that memory mapped RAS registers are not exposed to SPs. Whenever possible, any access to RAS System registers from an SP should be trapped to SPMC using the features defined in the base hardware architecture.
Residual Impact	Medium
Residual Likelihood	Low
Residual Risk	Low

5.10.9 DoS attacks due to resource exhaustion in partition manager

Description: For example, in a time-sliced memory management operation, a malicious endpoint may issue new memory management requests instead of resuming an earlier paused operation in an attempt to cause exhaustion of memory in the partition manager and deny services to other endpoints. Multiple malicious endpoints may collude to mount DDoS attack using this technique. In a scenario where the transaction descriptor is transmitted in fragments, a malicious endpoint may initiate new memory management transactions instead of transmitting fragments of an earlier transaction to cause exhaustion of memory used to track FF-A messages in the partition manager. Also, multiple malicious endpoints may collude to mount DDoS attack using this technique.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the implementations of FF-A. After IMPLEMENTATION DEFINED transactions that are outstanding from each endpoint, implementations of

FF-A must deny new requests from endpoints to ensure it does not run out of memory.

Residual Impact	Medium
Residual Likelihood	Medium
Residual Risk	Medium

5.10.10 DoS attacks on SP or VM due to unsuccessful notification configuration phase in Normal world scheduler

Description: For example, a VM (sender) wants to send a notification to an SP (receiver) or an SP (sender) wants to send a notification to another SP (receiver). As part of the notification configuration, the receiver endpoint driver in Normal world (receiver's) scheduler needs to register a call back (during FF-A driver initialization phase in Normal world scheduler) to run receiver for a pending notification.

The SPM is unaware if this FF-A driver initialization in Normal world (receiver's) scheduler fails. For the above scenario, this may result in a sender indefinitely waiting for crucial communication from the receiver or the receiver is unable to service any request at all.

Unmitigated Impact	Medium
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is transferred to the system integrators. Ensuring a successful initialization of the receiver endpoint driver in Normal world scheduler is vital to enable notification signaling between endpoints.
Residual Impact	Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.10.11 DoS attacks due to flooding of notifications to a target endpoint

Description: A malicious endpoint might attempt to flood another endpoint with notifications and affect its operation. The intent of the malicious endpoint could be to interfere with both the receiver's and/or primary endpoint execution, as they can both be preempted by the Notification Pending Interrupt and Schedule Receiver Interrupt respectively.

Unmitigated Impact	Low
Unmitigated Likelihood	Medium

Unmitigated Risk	Low
Mitigating Action	The threat is transferred to the implementations of FF-A. However, the impact is low due to the architecture definition:
	Notifications are not queued, one that has been signaled needs to be retrieved by the receiver, until it can be sent again. This limits the number of spurious interrupts that can be targeted to an endpoint.
Residual Impact	Low
Residual Likelihood	Low
Residual Risk	Low

5.11 Privilege escalation attacks

5.11.1 Elevation of access permissions, memory type, and attributes through memory management interfaces

Description:

- A malicious endpoint could elevate access permissions, memory type and attributes beyond what is granted to it by the owner of the memory region. For example, this scenario is possible if a malicious borrower of a memory region tries to elevate access permissions, memory type or attribute by modifying its stage 1 translation tables or by requesting the Hypervisor or SPM to provide privileges beyond what is granted to it by the owner.

- A malicious lender may try to elevate access permission or memory type or attributes of its own memory region through memory management interfaces. For example, the scenario is possible if a lender tries to lend/share a memory region it owns to borrowers with higher privileges, and later attempts to reclaim the memory region to get higher privileges than what it had in the first place.

Unmitigated Impact	High
Unmitigated Likelihood	Low
Unmitigated Risk	Low
Mitigating Action	The threat is controlled by the architecture by ensuring that access permissions, memory type, and attributes set by the owner of a memory region to all borrowers are enforced. The architecture also ensures that the owner of a memory region is not able to elevate their privileges through memory management interfaces.
Residual Impact	Very Low
Residual Likelihood	Very Low

Residual Risk	Very Low

5.11.2 Memory management transactions between VMs and SPs that involve Secure physical memory

Description: An SP tries to share or lend access or donate ownership of Secure memory to an NS endpoint. The scenario is possible if SP is induced into giving access to Secure memory to an NS endpoint that is, it is able to bypass the TrustZone address space controls for this memory region. The scenario is also possible if a malicious OS kernel attempts to alter globally observable state of Secure memory through a memory management transaction.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture. SPM must sanitize any inputs coming from outside its TCB. If input parameters to an ABI correspond to memory locations, pointers or handles to memory regions or buffers, the FF-A shall ensure that they are appropriately sized and that the ownership, access attributes, security state of the memory region permit the call to make forward progress. It must do this before any processing occurs. For example, FF-A shall ensure that Secure partitions cannot perform any memory management transactions involving Secure memory with NS endpoints.
Residual Impact	Very Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.11.3 Exploiting transient execution state to elevate privileges

Description:

- Where multiple fragments are needed to complete message payload transmission through memory management interfaces, implementations of FF-A may process some fragments and may expose ownership and access attributes of partial memory regions before the final fragment is processed successfully. A malicious endpoint could exploit this transient state to gain privileges even if the last fragment is not processed successfully.

- If a partition manager exposes ownership and access attributes of partially processed memory regions in a time sliced memory management transaction, a malicious endpoint may gain unauthorized access to such regions even if the overall transaction fails.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium
Mitigating Action	The threat is controlled by the architecture. It requires that any globally observable state associated with the memory region described by the memory transaction descriptor be made visible to all endpoints only after successful completion of a memory management transaction.
Residual Impact	Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.11.4 Privilege escalation through FFA_MEM_DONATE

Description: The scenario is possible if a memory region is donated from one partition to another and back again with the original owner requesting the memory region to be mapped with greater privileges than it originally had.

Unmitigated Impact	High
Unmitigated Likelihood	High
Unmitigated Risk	High
Mitigating Action	The threat is controlled by the architecture. It expects the relayer to enforce the permissions in a partition manifest to act as a threshold of the permissions the partition is permitted to have during runtime.
Residual Impact	Very Low
Residual Likelihood	Very Low
Residual Risk	Very Low

5.11.5 Unauthorized retrieval of memory request

Description: The scenario is possible if a malicious owner tries to retrieve the request of a memory region that it had already lent access to a borrower, to cause elevation of privileges.

Unmitigated Impact	High
Unmitigated Likelihood	Medium
Unmitigated Risk	Medium

Mitigating Action	The threat is controlled by the architecture by forbidding the owner to never retrieve a memory region after it has been lent unless the borrower has relinquished access first.
Residual Impact	Low
Residual Likelihood	Very Low
Residual Risk	Very Low

Non-Confidential

6 Mitigation summary

6.1 Architectural mitigations

6.1.1 FF-A architecture

The following are some of the mitigations defined in the FF-A architecture that an implementer can use:

1. Use partition manifest and partition ID to mitigate 5.1.
2. Use buffer synchronization (management of buffer states and ownership) and buffer-based message transmission aspects to mitigate 5.2.
3. Use partition setup and initialization, partition runtime models, and interrupt management to mitigate 5.4.1.
4. Use execution context initialization protocols to mitigate 5.4.5.
5. Use sanitization of inputs to FF-A ABIs to mitigate 5.6.
6. Use message processing to mitigate 5.8.
7. Use sanitization of responses to FF-A ABI requests to mitigate 5.9.1.
8. Use dynamic buffer-based transmission of memory transaction descriptors to mitigate 5.10.2.
9. Use time slicing of memory management operations to mitigate 5.10.3.
10. Use memory region ownership and access concepts to mitigate 5.10.4.
11. Use guidance in partition runtime models and interrupt management to mitigate 5.10.5.
12. Use managed exit to mitigate 5.10.6.
13. Use memory region attributes precedence concepts to mitigate 5.11.1.
14. Use relayer responsibilities (sanitization of inputs to FF-A ABIs) to mitigate 5.11.2.
15. Use global observability state concepts defined for transmission of transactions descriptors in fragment or time slices to mitigate 5.11.3.
16. Use memory access permissions in partition manifest to mitigate 5.11.4.
17. Use memory retrieve request usage constraints to mitigate 5.11.5.

6.1.2 Other architectures

None.

6.2 Implementation-level mitigations (Security Objectives)

The mitigating actions listed in this section are meant for implementations of FF-A, Silicon vendors, and System integrators.

1. Implementations of FF-A or system integrators must provide mechanisms in SPMC to sanitize inputs from FF-A components outside the TCB of a partition, like Normal world ID of an endpoint in a direct message request or response, to mitigate 5.1.
2. Implementations of FF-A must provide mechanisms to protect the integrity of TX/RX buffers from concurrent accesses by the producers or consumers of these buffers to mitigate 5.2.
3. Silicon vendors must provide mechanisms to protect firmware from (unauthorized) rollback to mitigate 5.3.
4. In the absence of tamper-resistant memory, implementations of FF-A and system vendors must determine sensitive data structures and protect their integrity in memory. For example, using cryptographic hashes to mitigate 5.4.2.
5. Implementations of FF-A must ensure system counter accesses are only performed from EL3 to mitigate 5.4.3.
6. Silicon providers, system integrators, and implementations of FF-A must provide physical tamper detection and prevention mechanisms to mitigate 5.4.6.
7. Implementations of FF-A must ensure sensitive logging information like crash reports and high-verbosity logging are not available in release builds to mitigate 5.4.6.
8. System integrators must provide mechanisms to prevent unauthorized access to debug and trace ports to mitigate 5.7.
9. Silicon vendors or system integrators must provide mechanisms to protect the confidentiality of DRAM contents to mitigate 5.9.2
10. To mitigate 5.9.3, silicon vendors must implement cache timing side-channel mitigations defined in the Arm architecture for A-profile. Implementations of FF-A might be hardened further with software mitigations for cases not covered in hardware.
11. System integrators and implementations of FF-A must use constant-time and uniform algorithms with no secret-dependent memory access patterns to mitigate 5.9.4. Effective cache-based side-channel mitigations must be implemented in the hardware by silicon vendors.
12. Implementations of FF-A must use constant-time and uniform algorithms with no secret-dependent memory access patterns and making interrupt requests constant time to mitigate 5.9.5.
13. Implementations of FF-A must provide access restrictions to PMU and AMU counters from Non-secure world or SPs, and system integrators must either disable external debug in production devices or provide debug authentication mechanisms to mitigate 5.9.6.
14. Implementations of FF-A must ensure that ABIs return response constantly to mitigate 5.9.7.

15. Implementations of FF-A must ensure that their resources like memory are not exhausted or over-utilized by an endpoint or a few endpoints, to guarantee Reliability and Availability of Services to mitigate 5.10.1.
16. Implementations of FF-A must provide mechanisms to ensure delivery of critical Non-secure world interrupts that arrive during Secure world execution to mitigate 5.10.7.
17. Implementations of FF-A must restrict SPs to access RAS system registers to mitigate 5.10.8.
18. Implementations of FF-A must provide mechanisms to ensure it does not run out of system resources like memory to mitigate 5.10.9.
19. System integrators must ensure the receiver endpoint driver in Normal world scheduler is properly initialized to mitigate 5.10.10.
20. Implementations of FF-A must provide mechanisms to ensure endpoints are not frequently interrupted due to notifications to mitigate 5.10.11.

6.3 User-level mitigations (Remediations)

The mitigations listed in this section are meant for Trusted OS vendors.

1. Trusted OS vendors shall ensure SPs are robust enough to cope with the power down of the PE to mitigate 5.4.4.
2. Trusted OS vendors shall harden their code to ensure sensitive information is not leaked through message response to mitigate 5.9.1.
3. Trusted OS vendors shall implement techniques, for example kernel page table isolation, if the underlying hardware is not hardened with cache speculation side-channel related security features introduced in Armv8.5 to mitigate meltdown-type attacks listed in 5.9.3.
4. Trusted OS vendors shall implement constant-time and uniform algorithms with no secret dependent memory access patterns to mitigate 5.9.4.

Appendix A   Revisions

This describes the technical changes between released issues of this document.

Table A-1: Issue 01

Change	Location
First release	-