Arm® Server Base Security Guide

Document number:	DEN 0086
Release Quality:	REL
Release Number:	1.0
Confidentiality	Non-Confidential
Date of Issue:	September 30, 2019

Contents

About this document

This document specifies requirements and guidance to secure the platform (firmware/hardware) of SBSA/SBBR class server systems.

Release Information

The change history table lists the changes that have been made to this document.

Date	Version	Confidentiality	Change
September 2019	Release 1	Non-Confidential	Initial Release

Arm® Server Base Security Guide

Arm Non-Confidential Document Licence ("Licence")

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

The Document consists solely of commercial items. You 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.

If any of the provisions contained in this Licence conflict with any of the provisions of any click-through or signed written agreement with Arm relating to the Document, then the click-through or signed written agreement prevails over and supersedes the conflicting provisions of this Licence. This Licence may be translated into other languages for convenience, and you agree 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 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

References

This document refers to the following documents.

Ref	Document Number	Title
[1]	NIST SP 800-193	**Platform Firmware Resiliency Guidelines**, NIST Special Publication 800-193, May 2018.
[2]	NIST SP 800-57	**Recommendation for Key Management**, NIST Special Publication 800-57, January 2016
[3]		TCG PC Client Platform Firmware Profile Specification, Family “2.0”, Level 00 Revision 1.04, June 3, 2019 Errata Version 1.0 for TCG PC Client Specific Platform Firmware Profile Specification Version 1.04. June 3, 2019
[4]		TCG EFI Protocol Specification, Family “2.0”, Version 1.0, Level 00 Revision 00.13, March 30, 2016
[5]		TCG ACPI Specification, Family “1.2” and “2.0”, August 18, 2017
[6]		TPM 2.0 Mobile Command Response Buffer Interface, Family “2.0”, Level 00 Revision 12, 16 December 2014
[7]	ARM DEN 0072	**PSA Trusted Boot and Firmware Update** (TBFU), 2019
[8]		Unified Extensible Firmware Interface Specification, Version 2.8, March 2019
[9]		TCG PC Client Platform Physical Presence Interface Specification, Family “1.2” and “2.0”, Version 1.30 Revision 00.52, July 28, 2015
[10]	ARM DEN 0022D	Arm® Power State Coordination Interface, Version 1.1, April 2017
[11]	NIST 800-90A NIST 800-90B NIST 800-90C	NIST Special Publication 800-90A, 800-90B, 800-90C define requirements for implementing random number generators. **SP 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators**. **SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation**. **SP 800-90C (DRAFT), Recommendation for Random Bit Generator (RBG) Constructions**.
[12]	ARM DEN 0034A	ARM® Debug and Trace Configuration and Usage Models
[13]		Protection Profile PC Client Specific TPM, Version 1.1. TPM Library Specification Family “2.0” Level 0 Revision 1.38. June 16, 2018
[14]		TPM 2.0 Mobile Reference Architecture, Family “2.0”, Level 00 Revision 142, 16 December 2014
[15]		TPM 2.0 Mobile Common Profile, Family “2.0”, Level 00 Revision 29, 27 August 2015

[16]	ARM DEN 0029B	**Arm® Server Base System Architecture** 6.0, 2019
[17]	ARM DEN 0044C	**Arm® Server Base Boot Requirements** 1.2, 2019
[18]	ARM DEN 0070A	**Arm® Firmware interfaces for mitigating cache speculation vulnerabilities** 1.3, May 2018
[19]		TCG PC Client Platform Reset Attack Mitigation Specification, Family “2.0”, Version 1.10 Revision 17, January 21, 2019
[20]	RFC 2119	IETF, RFC 2119, March 1997
[21]	NIST SP 800-107	**NIST Special Publication 800-107**, Recommendation for Applications Using Approved Hash Algorithms, Revision 1, August 2012
[22]		**Microsoft Project Cerberus**
[23]		**Google Titan**
[24]		**Amazon Web Services, Nitro System**, AWS re:Invent 2018

Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
Critical data	Critical data includes configuration settings and policies that need to be in a valid state for a device to maintain its security posture during boot and runtime. All other data is non-critical.
Normal world	The Non-secure privilege levels (Non-secure EL0/EL1/EL2) and resources, for example memory, registers, devices, that are not part of the Secure world.
PE	Processing Element. Processing Element, as defined in the Arm Architecture Reference Manual. Typically, a single hardware thread of a PE.
ROTPK	Root of Trust Public Key. A ROTPK is the public half of an asymmetric key pair at the root of a chain of trust that is used to authenticate firmware images or critical data.
Secure world	The environment that is provided by the Secure privilege levels in the Arm v8-A architecture (Secure EL0/EL1/EL2/EL3) and the resources, for example memory, registers, and devices, that are accessible exclusively from the Secure privilege levels.
Server Base system	A system that is compliant with the SBSA and SBBR standards.
TCG	Trusted Computing Group
TPM	Trusted Platform Module

Feedback

Arm welcomes feedback on its documentation.

Feedback on this document

If you have comments on the content of this document, send an e-mail to **errata@arm.com**. Give:

- The title (Arm® Server Base Security Guide).
- The number and release (DEN 0086 Alpha 1 REL 1.0).
- The page numbers to which your comments apply.
- The rule identifiers to which your comments apply, if applicable.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Introduction

1.1 Overview

A server system can be viewed as software layers (OS/hypervisor and applications) that run on a hardware platform (firmware, hardware, and devices). In this document, the term platform refers to the set of hardware and firmware components in a system on which an operating system and applications can run (see Figure 1). The platform provides a set of hardware and firmware mechanisms and services that an operating system and applications can rely on. The OS and application software layer provide the functional capabilities of the system that are available to users.

The platform forms the foundation of a system. This means that the integrity of the platform is essential to the overall integrity of the system. If any hardware or firmware component is compromised, the security of the entire system might be compromised.

Figure 1 shows an example of platform and software components that might typically be found in an Arm server system.

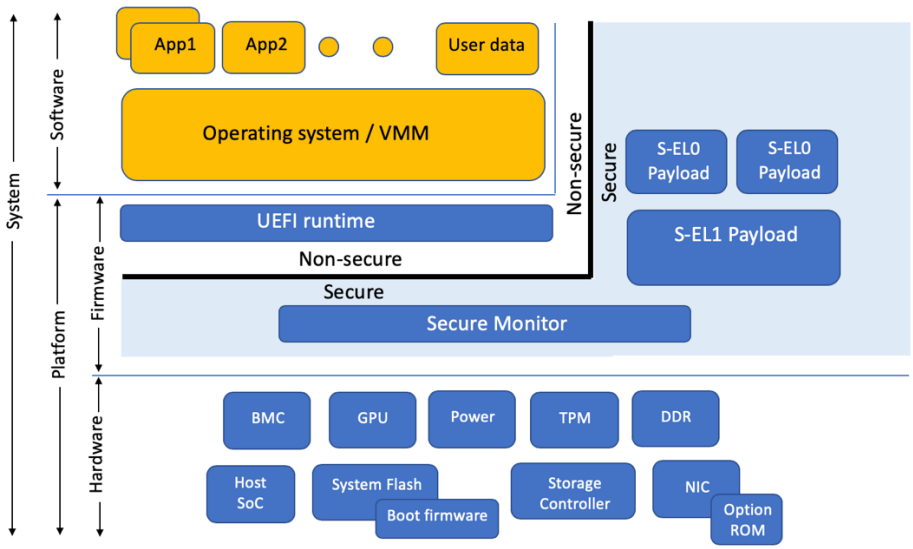

Figure 1

As shown in Figure 1, S-EL0 and S-EL1 payloads, for example secure partitions and trusted operating systems, are considered part of the platform.

The firmware in the platform extends beyond the host SoC firmware components. Other hardware components like the BMC, GPU, storage controller, and NIC might each have their own mutable firmware components that form a part of the platform. A compromise in the integrity of any of these platform components might affect the security of the whole system.

This document addresses the platform security of Arm v8 servers based on the Arm SBSA/SBBR standards [16][17].

Figure 1 shows the boundary between the Secure and Non-secure parts of the platform. In this document, Secure world refers to the environment that is provided by the Secure privilege levels in the Arm v8-A architecture (secure EL0/EL1/EL2/EL3) and the resources (for example memory, registers, devices) that are

accessible exclusively from the Secure privilege levels. The term Normal world refers to the Non-secure privilege levels (Non-secure EL0/EL1/EL2) and resources that are not part of the Secure world.

1.2 Scope

The design of a Secure product requires system-level threat modeling and analysis to determine the appropriate use of security features. The system design will also need to consider certification requirements of the target market.

This document provides requirements and guidance that support:

- Maintaining the integrity of the platform layer of a server based on SBSA and SBBR
- Securely attesting to the platform's state of integrity

This document focuses on protecting key assets in the platform, including:

- Mutable firmware components, critical data, and the corresponding non-volatile memory that contains them
- Secure memory, which could be Secure SRAM or partitioned DDR

This document specifies requirements and guidance for both firmware and hardware. Key areas of guidance that facilitate platform integrity include:

- Protection of firmware and critical data. Mutable firmware and critical data must be updatable, with updates being authorized and verified.
- Detection of corruption of firmware and critical data. All mutable firmware and critical data must be signed so that it can be verified during boot, forming a chain of trust rooted in an immutable hardware Root of Trust.
- First instruction integrity. The first mutable firmware that is executed on the host SoC or other system component must be authenticated before use by an immutable bootloader.
- TPM 2.0 integration and TPM-based measured boot, providing a means to securely attest to the state of the platform.
- UEFI security guidance
- Hardware requirements to facilitate implementing the firmware-based security requirements.
- Hardware requirements for secure memory.

This document addresses the requirements for deployed, production systems. It is not expected that development systems or debug builds of firmware meet the same security standards as production systems.

The following list identifies threats that are out of scope for this document:

- Threats to the Normal world operating system or hypervisor and application software stacks. The platform assumes that the Normal world OS and software stacks are not trusted.
- CPU side channel attacks, including differential power analysis attacks, timing attacks, and speculative execution attacks

  See **Arm Security Updates** for detailed information on the mitigation of speculative execution attacks, including Arm v8.5 mechanisms to address these attacks such as new barriers and memory disambiguation control. The SBSA and SBBR standards specify hardware and firmware requirements around the v8.5 security extensions. See also the Arm document *Firmware interfaces for mitigating cache speculation vulnerabilities*[18].

- Laboratory attacks in which devices are unpackaged and probed
- Power and clock glitch attacks which cause faults such as instruction skipping, malformed data in reads/writes, or instruction decoding errors
- BMC security. This document specifies requirements for protection, detection of corruption, and resiliency of firmware and would apply to all system components including BMCs. However, this document does not address BMC security in a comprehensive manner.
- Supply chain attacks. While the guidance in this document does provide mitigations against some potential attacks in a supply chain (for example firmware tampering), it does not directly address supply chain security.
- A Trusted Platform Module (TPM) is a security module that is commonly found on server systems. This document does not consider attacks against a TPM module. TCG has published a Common Criteria protection profile that describes threats and security requirements for a TPM: *Protection Profile PC Client Specific TPM* [13].

1.3 Relationship to Existing Standards and Security Architectures

A key goal of this document is to consolidate the essential security requirements for building a secure server platform in a single place. This document relies on existing security standards and provides references to the applicable sections of existing standards. Guidance is provided where clarifications might be needed to map the existing standards to Arm Server Base systems. Here are key standards that are referenced in this document:

- *NIST Platform Resiliency Guidelines (SP 800-193)* [1] defines guidelines for promoting resiliency in platform firmware. The SP 800-193 principles of protection, detection, and recovery are fundamental and assumed in this document.
- *Arm® Platform Security Architecture Trusted Boot and Firmware Update (TBFU)* [7] provides detailed guidance on protecting platform firmware through a trusted boot process and a secure firmware update process. TBFU addresses a wide range of Arm-based systems from IoT endpoints to servers. This document references key TBFU sections and provides additional details and clarifications to apply TBFU requirements and guidance to Server Base systems.
- *Systems based on SBBR assume firmware based on the Unified Extensible Firmware Interface Specification* [8]. The security requirements for a UEFI firmware implementation such as Secure Boot, Update Capsules, and Authenticated Variables are enumerated and clarified in this document where necessary.
- Trusted Computing Group specifications are referenced for systems that implement a TPM security module, and guidance is provided in this document for implementing a measured boot flow on an Arm Server Base system.

Cloud service providers have developed new platform Root of Trust security architectures that address authentication and attestation of server system components. See for example Microsoft's Project Cerberus [22], Google's Titan [23], and the AWS Nitro System [24]. The requirements and guidance in this document are complementary and compatible with these architectures.

1.4 Key Words

This document uses the key words must and should in accordance with *IETF, RFC 2119* [20].

Guidance points that are described as must are numbered, italicized, and highlighted in blue. These statements identify the key requirements to achieve the goal of maintaining the integrity of a server platform.

1.5 Audience

This document is targeted to suppliers that design and build the hardware and firmware layers of Arm servers, including silicon providers, OEM/ODMs, and BIOS vendors. The guidance is aimed at product managers who specify requirements, and hardware designers and firmware developers who implement those requirements. This document does not replace the need for component-level and system-level threat modeling to consider the key assets that must be protected, the potential attacks that could compromise those assets, and appropriate mitigations against those attacks.

A given requirement specified in this document may only affect one part of the supply chain, such as a silicon provider or firmware provider. However, it is advised that any supplier involved in the creation of Arm servers or server components be familiar with the broader set of platform security requirements. This is because there are interactions and dependencies between components, for example between firmware components and hardware mechanisms.

2 Platform Security Guidance

2.1 Firmware and Critical Data Protection

A secure firmware update process ensures that only authorized changes are permitted to the firmware in a system. This process ensures that firmware components remain in a state of integrity and addresses the protection requirement of NIST SP 800-193 [1].

Firmware updates could be in-band (controlled by an OS) or out-of-band (for example controlled by a BMC that is trusted to perform updates to the system flash).

The following list specifies requirements and guidance for firmware updates for the host SoC and other system components that boot firmware:

- *R010_SBSG: All mutable firmware in a system **must** be updatable.* This includes firmware for components within the host SoC such as application processors and auxiliary controllers, and peripherals and board-level devices such as peripherals, adapters, and BMCs.
- *R020_SBSG: The firmware protection guidance in NIST SP 800-193 **must** be followed—all updates to firmware components in the platform **must** be authorized and authenticated by a Root of Trust for Protection, and the authentication process **must** not be bypassable.* As defined by NIST, authenticated means that the source and integrity of a firmware component can be verified, and authorization is the permission granted to apply an update. See the informational section Update Process in TBFU [7].
- *R030_SBSG: The firmware update mechanism **must** implement a firmware rollback protection scheme in accordance with TBFU to prevent an older, insecure version of firmware from being installed.* TBFU provides flexibility in how rollback protection is implemented and when rollback counters are increased. See the section Anti-rollback Protection in TBFU [7].
- *R040_SBSG: On UEFI-based systems, OS-controlled updates of mutable host SoC firmware **must** be implemented using Update Capsules.* OS-controlled updates of firmware for other board level components should be implemented using UEFI update capsules.
- *R050_SBSG: On UEFI-based systems, support for Update Capsules **must** be implemented as described in the UEFI Specification section Update Capsule.* See section 8.5.3, Update Capsule, in the UEFI Specification [8]. Capsules might be delivered through a file within the EFI system partition as described in section 8.5.5, Delivery of Capsules via file on Mass Storage device, in the UEFI Specification [8].

- *R060_SBSG: In conjunction with support for Update Capsules, systems **must** implement support for the Firmware Management Protocol, Firmware Management Protocol defined updates, and the EFI System Resource Table (ESRT) as defined in the UEFI Specification section Firmware Update and Reporting.* See section 23, Firmware Updating and Reporting in the UEFI Specification [8].

As defined by NIST SP 800-193, critical data includes configuration settings and policies that need to be in a valid state for a system to maintain its security posture during boot and runtime. All other data is non-critical.

*R070_SBSG: The critical data protection guidance in NIST SP 800-193 **must** be followed. All updates to critical data **must** be authorized and authenticated.*

UEFI authenticated variables are critical data and provide a means for a platform owner to control the setting of critical UEFI settings, like variables that affect UEFI Secure Boot. Changes to authenticated variables must be signed by an appropriate private key. The following requirements are applicable to UEFI authenticate variables:

- *R080_SBSG: Authenticated variables **must** be supported and be compliant with the Globally Defined Variables and Variable Services sections of the UEFI Specification.* See sections 3.3, Globally Defined Variables, and 8.2, Variable Services, in the UEFI Specification [8].
- *R090_SBSG: Non-volatile variables **must** be stored in a persistent location that is not writable by the Normal world. An example is a Secure world only accessible SPI flash.*
- *R100_SBSG: The db/dbx signature database variables EFI_IMAGE_SECURITY_DATABASE and EFI_IMAGE_SECURITY_DATABASE1 **must** be created to include the EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS attribute to prevent rollback.*

2.2 Detection of Corruption of Platform Firmware and Critical Data

Trusted Boot is a process that cryptographically authenticates all code that runs on a system. This process addresses the detection requirement of NIST SP 800-193 [1]. Trusted Boot requires that all firmware be cryptographically signed. This enables the verification process to detect whether firmware components have been compromised or corrupted.

For the host SoC and other system components that contain mutable firmware, Trusted Boot begins in an immutable bootloader component such as a boot ROM which loads the first mutable firmware image. Before transferring control to the loaded image, the image's integrity and authenticity is verified using digital signatures. The boot process continues with each component in the boot chain performing integrity and verification of the next component before it is executed or used. This forms a chain of trust anchored in the immutable bootloader and continuing through all code that is executed up to the runtime environment, for example OS. For UEFI systems, UEFI Secure Boot is part of the Trusted Boot process.

This document requires that all firmware be signed and verified during boot and recommends that all software executed up through and including the OS/hypervisor be signed and verified. This includes secondary bootloaders, ramdisks, and OS/hypervisor kernels. If verification fails, the boot process must halt or a recovery process can be initiated.

The public key at the root of a chain of trust used for authentication is referred to in this document as a Root of Trust Public Key (ROTPK). See TBFU [7] for further details on firmware signing and ROTPKs.

The immutable bootloader and ROTPK form the Root of Trust for Detection in NIST SP 800-193 terms.

Trusted Boot Guidance

The following list specifies requirements and guidance for Trusted Boot for the host SoC and other system components that boot firmware:

- *R110_SBSG: All mutable firmware components **must** be signed in accordance with the requirements in TBFU.* For the host SoC, components include firmware up to and including the Normal world bootloader and any Trusted operating systems and Trusted applications in the Secure world. See the Cryptographic Algorithms and Image Manifests sections in TBFU for guidance on image signing, such as acceptable cryptographic algorithms.
- *R115_SBSG: All mutable firmware components executed from reset **must** be authenticated in accordance with the requirements in TBFU in order to detect corruption of firmware.* For the host SoC this includes firmware up to and including the Normal world bootloader and any Trusted operating systems and Trusted applications in the Secure world.
- *R120_SBSG: If authentication fails, the boot process **must** halt, or a recovery process can be initiated.* See the Image verification and Trusted Boot Flow sections in TBFU.
- *R130_SBSG: The immutable bootloader **must** authenticate the first mutable firmware component using an immutable ROTPK.* See the section PRoT immutable bootloader requirements in TBFU.
- The detection mechanism should be capable of logging events and creating notifications of corruption, for example to a BMC. See guidance in NIST SP 800-193.
- *R140_SBSG: The critical data detection guidance in NIST SP 800-193 **must** be followed. Integrity checks on critical data **must** be performed prior to use.* A recommended approach would be to sign all critical data.
- The Trusted Boot flow can be divided into two phases. First, the Secure world Trusted Boot flow that executes Secure world components. Second, the Normal world Trusted Boot flow that executes normal world component, for example UEFI and grub. *R150_SBSG: When Trusted Boot is enabled on a system it **must** not be possible to disable or bypass Secure world Trusted Boot.*
- *R160_SBSG: It **must** not be possible for a user to bypass Secure world Trusted Boot failures.* A physically present user override is not permitted for images that fail signature verification.

The open source Trusted Firmware-A project demonstrates an implementation of a Trusted Boot flow that meets the requirements of this document. See **https://www.trustedfirmware.org/**.

Root of Trust Public Key Guidance

The following are requirements for ROTPKs:

- *R170_SBSG: The ROTPK or hash of the ROTPK used by an immutable bootloader must be entirely embedded in on-chip, immutable, non-volatile storage.*
- *R180_SBSG: An ROTPK key size appropriate for a security strength of 128-bits as recommended by NIST SP 800-57 [2] **must** be used.* Geographical cryptographic requirements may apply to the keys used. If elliptic-curve or RSA-based keys are used, the following requirements apply:
    - *R190_SBSG: An elliptic-curve-based ROTPK **must** be at least 256 bits in size.*
    - *R200_SBSG: An RSA-based ROTPK **must** be at least 3072 bits in size.*
- To reduce the ROTPK storage footprint, it is permitted to store a cryptographic hash of the key. See Cryptographic Algorithms in TBFU [7] for specific algorithm guidance. The public key can then be stored in external non-volatile memory. When required, the key must be retrieved from external memory before it is used, and successfully compared with the stored hash by trusted hardware or software. This approach is known as hash locking. In order to minimize the amount of immutable non-volatile storage

that is required, truncation of the hash value can provide sufficient security strength. NIST SP 100-107 [21] provides a detailed description of the requirements and security properties of truncated digests. For example, a common truncation mode is for the leftmost 128 bits from a SHA-256 digest to be used.

UEFI Secure Boot

Secure Boot is a feature that is described in the UEFI specification [8] to detect and prevent unauthorized EFI drivers, option ROMs, or programs from being executed during boot. The following are requirements for UEFI Secure Boot:

- *R210_SBSG: Systems **must** implement UEFI Secure Boot to prevent unauthorized EFI drivers, option ROMs, or programs from being executed during boot.*
- *R220_SBSG: To support UEFI Secure Boot the system firmware **must** be compliant with the Runtime Services Rules and Restrictions, Variable Services, and Secure Boot and Driver Signing sections of the UEFI Specification.* See:
    - See section 8.1, Runtime Services Rules and Restrictions, in the UEFI Specification [8]
    - See section 8.2, Variable Services, in the UEFI Specification [8]
    - See section 32, Secure Boot and Driver Signing, in the UEFI Specification [8]
- *R230_SBSG: UEFI drivers, option ROMs, and UEFI executables **must** be signed in accordance with Secure Boot and Driver Signing in the UEFI specification.* See section 31, Boot and Driver Signing, in the UEFI Specification [8].
- *R240_SBSG: Firmware **must** implement the Secure Boot variable as documented in the section Globally Defined Variables of the UEFI Specification.* See section 3.3, Globally Defined Variables, in the UEFI Specification [8].
- *R250_SBSG: UEFI components that are not required to boot the platform **must** not be signed by a production certificate in the authorized signatures database (db).* UEFI applications could potentially weaken the security of Secure Boot. This includes applications like UEFI shells and utilities for manufacturing, test, and debug. The platform administrator must disable Secure Boot to run these unsigned utilities.
- *R260_SBSG: If authentication of a component fails, the boot process **must** halt, or a recovery process can be initiated.*
- *R270_SBSG: It **must** not be possible for a user to bypass UEFI Secure Boot failures.* A physically present user override is not permitted for images that fail signature verification.
- The dbx signature database variables EFI_IMAGE_SECURITY_DATABASE1 should be pre-populated with updated revocations that are published by the **UEFI Forum**.

2.3 Protection of Secure Storage and Memory

As described in the Overview in this document, Arm systems have a critical security boundary separating the Secure world and Normal world. Two key assets that are affected by that boundary include Secure non-volatile storage and Secure memory.

Secure Storage

Secure storage in this document refers to non-volatile memory in the system, for example the system flash, that holds firmware images, critical data, and UEFI variables that affect the security posture of the system platform.

*R280_SBSG: In order to prevent unauthorized access and tampering, secure storage **must** not be accessible by the Normal world.*

Secure Memory

Secure memory is volatile memory that belongs to the Secure world and is not accessible by the Normal world. A system should have sufficient secure memory for the use cases and markets that the system addresses. Secure memory could include an on-chip SRAM and off-chip DDR that is partitioned into Secure and Non-secure regions.

- For cases where DDR is partitioned, the partitioning can be achieved by fixed logic or by a configurable mechanism. *R290_SBSG: For configurable partitioning of Secure memory the configuration mechanism **must** not be accessible from the Normal world.*
- DDR is memory external to the SoC and is thus subject to attacks on the external memory bus. For cases where DDR is partitioned, this document still recommends some amount of on-chip secure SRAM that can be used to hold sensitive assets.
- *R300_SBSG: For a system with configurable partitioning of Secure memory any region **must** be scrubbed before it can be reallocated from Secure world to Normal world or from Normal world to Secure world.* The reason for this requirement is:
    - A Normal world region might have Non-secure code residing in it.
    - A Secure world region might have secrets residing in it.

In systems where Secure memory comes from partitioned DDR, a potential threat is a row hammer attack where the attacker running Normal world software takes advantage of disturbance errors in DDR to cause bit flips in Secure memory. This could lead to a variety of exploits.

- If DDR is partitioned into Secure and Non-secure regions, the memory controller should implement ECC protection for robustness and to mitigate against row hammer attacks against Secure memory.

DMA Attacks

If an attacker compromises the firmware of a DMA capable I/O device, the compromised device might be able to access system memory during the boot process. This could allow the attacker to interfere with the system's Trusted Boot process or corrupt memory through time-of-check-time-of-use (TOCTOU) attacks against system firmware components.

A System MMU can provide protection against DMA attacks by defaulting to a deny access policy. Firmware can selectively create SMMU mappings and enable DMA for specific devices that are needed to boot the system.

The following requirements are applicable to System MMUs:

- *R310_SBSG: All master-capable devices that can be controlled by PE software **must** be behind an SMMU, except as noted here.* These types of devices include: PCI devices, USB bus controllers, SPI controllers, I2C controllers network devices, disk controllers, and accelerators. In this context, control of a device means actions like initialization, operation, interrupt handling, and managing errors.
    - This requirement does not apply to Trusted master-capable devices which manage resources that are not intended to be accessible or controlled by PE software. Examples include system controllers or power management controllers. While a PE might interact with these types of devices, they operate autonomously and not under control of PE software. And, from a security perspective they are Trusted.
    - The requirement does not apply to SMMUs, interrupt controllers, or debug access ports which are master-capable, but are explicitly forbidden from being behind an SMMU by design.

- If a system implements secure DMA masters, an SMMU implementation should support two security states as specified by the System Memory Management Unit Architecture Specification so that secure devices cannot bypass the SMMU. Whether the implementation supports two security states is indicated in SMMU_IDR0.SES.
- For SMMU v3, SMMU_GBPA[ABORT] should reset to a value of 1 to implement a default "deny" policy on reset. If it is not possible to have SMMU_GBPA[ABORT] reset to a value of 1, firmware should configure SMMU_GBPA[ABORT] to have a value of 1 as early as possible during the boot process, preferably in the immutable bootloader of the SoC.
- *R320_SBSG: For operating system-controlled devices, any SMMU mappings that are created by firmware for booting the system **must** be unmapped prior to handoff to the operating system.*

Platform Reset Attacks

A platform reset attack happens when an attacker with physical presence causes a system to be unexpectedly rebooted without a clean shutdown of the operating system, and then makes the system boot on an alternate boot device such as a USB drive or DVD into an OS that is under the control of the attacker. The reboot could be caused by actions like resetting the system, power cycling the system, or causing a kernel crash. A key to the attack succeeding is that that volatile system memory can retain its contents across the attacker-forced reboot. This can happen even when cycling the system's power. After booting into the attacker-controlled OS, the attacker can then scan memory to identify secrets like disk encryption keys.

A mitigation against this attack is defined in *TCG PC Client Platform Reset Attack Mitigation Specification* v1.10 [19], where a firmware mechanism is defined enabling an OS or bootloader to set a non-volatile memory overwrite flag (MemoryOverwriteRequestControl) telling the firmware to clear memory prior to booting an operating system. The TCG specification also defines a protection flag (MemoryOverwriteRequestControlLock) for the memory overwrite flag, preventing it from being modified.

To mitigate against platform reset attacks, UEFI-based systems should implement the mitigation specified in *TCG PC Client Platform Reset Attack Mitigation Specification* v1.10 [19].

2.4 TPM 2.0 Integration and Measured Boot

A TPM is a security module that provides a number of foundational building blocks for platform security, including:

- Secure storage for boot measurements, forming the basis for a system to be able to perform secure attestation and implement security policies by sealing TPM objects to PCR values.
- Endorsement key provides a unique, unclonable identity bound to hardware
- Key storage and management
- Secure cryptography where keys are never brought into the clear
- Key generation
- True random number generator

The threat model for a system will dictate whether a system needs a TPM or security module with equivalent functionality. There might be geography specific requirements for security modules, like TPCM/TCM devices in China.

SBSA specifies that, if a system implements a TPM, it must be compliant with version 2.0 of the TCG specifications.

This document recommends that a TPM be implemented as a Secure device, such as a TPM connected to a Secure-world SPI controller. A Secure world-owned TPM provides several potential benefits:

- Enables the TPM to be used by the Secure world for use cases that need it.

- Provides potential flexibility for implementing policies that restrict what the Normal world is able to do to the TPM.

Measured Boot

One capability a TPM enables is measured boot. In measured boot, hashes of all firmware and critical data are securely recorded in platform configuration registers (PCRs) in the TPM during boot. This process enables a system to later attest to its state and enforce security policies.

As per TCG guidance, the intent of PCR[0] is to represent a consistent view of the system platform between boot cycles. PCR[0] contains the cumulative measurement of the firmware components that are provided by the system platform manufacturer. This allows attestation and TPM sealed storage policies to be defined based on PCR[0], which represents the less changeable platform manufacturer-provided firmware components. The components measured into PCR[0] are typically code, but could be data, such as an ACPI image in flash.

This section specifies requirements and guidance when implementing support for a TPM 2.0.

*R340_SBSG: Mutable firmware components and critical data **must** be measured into PCR[0] through PCR[7] during boot following the PCR usage guidelines in the PC Client Platform Firmware Profile Specification [3]* and the *Errata for TCG PC Client Specific Platform Firmware Profile Specification* [3]. The following below summarizes PCR usage:

PCR	Description
0	Pre-UEFI firmware, Secure world software, boot manager, and signed data
1	Platform configuration and data, for example SMBIOS tables
2	UEFI drivers and applications that are not in a firmware volume, for example option ROMs
3	UEFI drivers and applications configuration and data
4	Boot manager code and boot attempts
5	Boot manager configuration and data
6	Host platform manufacturer specific
7	Secure boot policy

This document recommends that all software executed up through and including the OS/hypervisor be measured. This includes secondary bootloaders, ramdisks, and OS/hypervisor kernels. Critical data like Secure boot configuration and the kernel command line should be measured.

Arm systems will have a number of firmware components that execute on the host SoC in the Secure world prior to running Normal world boot firmware such as UEFI. The following requirements specify guidance for how these Secure world components are measured:

- *R350_SBSG: All mutable, Secure world firmware components that run on the host SoC **must** be measured into PCR[0].* This includes any Secure world runtime firmware, like SEL1 and SEL0 payloads. This also includes mutable firmware for auxiliary controllers that play a role in SoC initialization, firmware image loading, power management, clock management, or other SoC management functions.

- *R360_SBSG: Any signed critical data **must** be measured into PCR[0].*
- *R364_SBSG: Secure boot policy information **must** be measured into PCR[7] as described in PCR[7] - Secure Boot Policy Measurements in the PC Client Platform Firmware Profile Specification [3]. In addition, PCR[7] should indicate if the system is in a mode, like a debugging or manufacturing mode, that may compromise firmware security.*
- *R365_SBSG: All measurements that are made into TPM PCRs **must** be made with a SHA-256 or stronger hashing algorithm.*
- The immutable bootloader that anchors the chain of trust does not need to be measured, and the version of the immutable bootloader does not need to be logged (event EV_S_CRTM_VERSION).
- It is acceptable for the first mutable firmware component to measure itself because it has been verified by the immutable bootloader and is then trusted.
- As per TCG specifications, firmware components that are measured into PCR[0] must be logged in the event log using the event type EV_POST_CODE. The following table contains recommended strings for the event data that are used for Secure world firmware components. In the following table, %d represents a platform appropriate integer value and %s represents a platform appropriate string for the component.

EV_POST_CODE event data	Component description
SYS_CTRL_%d	For firmware for any kind of auxiliary controller in the SoC
BL_%d	For any bootloader component on the application processor. For example BL2 in Trusted Firmware-A would be “BL_2”.
SECURE_RT_EL0_%s	Secure EL0 runtime component
SECURE_RT_EL1_%s	Secure EL1 runtime component
SECURE_RT_EL2	Secure EL2 runtime component
SECURE_RT_EL3	EL3 runtime component. For example BL31 in Trusted Firmware-A nomenclature.

It is permissible to implement the TPM 2.0 device driver as part of a Secure EL0 runtime component, providing an SMC-based TPM 2.0 service to the Normal world for access to the TPM. If this approach is followed:

- A command response buffer interface should be used as defined in TPM 2.0 Mobile Command Response Buffer Interface [6].
- The SMC start method should be advertised as defined in the TCG ACPI Specification [5].
- Since TPM functionality is not available during the early phases of boot because the Secure EL0 runtime has not yet been started, it is permissible to record measurements in Secure memory for later replay into the TPM after the Secure EL0 runtime has been started.

Additional TPM-related requirements are listed here:

- *R370_SBSG: A TPM 2.0 device **must** be advertised via ACPI tables as specified in the TCG ACPI Specification [5], so that it can be discovered by an operating system.*

- The only locality that is supported on Arm systems is locality 0. DRTM (Dynamic Root of Trust for Measurement) and the associated dynamic localities 1 – 4 are not supported at this time.
- *R380_SBSG: UEFI firmware **must** implement the EFI_TCG2_PROTOCOL as defined in the TCG EFI Protocol Specification Family 2.0 [4].*
- As described in TCG PC Client Platform Physical Presence Interface Specification [9], physical presence is a form of authorization to perform certain TPM functions. This authorization normally comes from the platform operator. There are two methods that are defined by TCG to provide physical presence authorization in a PC Client environment: the command method and the hardware method.

  *R390_SBSG: If the command method is implemented by firmware in a platform, it **must** follow the requirements in TCG PC Client Platform Physical Presence Interface Specification [9].*
- A TPM 2.0 can be implemented as a firmware component in a protected environment like the Secure world. TCG has published a number of specifications providing guidance for an firmware TPM implementation: *TPM 2.0 Mobile Reference Architecture* [14], *TPM 2.0 Mobile Common Profile* [15], and *TPM 2.0 Mobile Command Response Buffer Interface* [6]. The threats faced by a firmware TPM are different than a physical TPM and threat modelling should be done to evaluate potential threats and mitigations.

2.5 Firmware recovery

In order to be resilient, a system needs a way to restore the platform to a state of integrity if corruption of firmware or critical data is detected. Without resiliency a successful attack against the platform might leave a system inoperable resulting in a denial of service.

*R400_SBSG: A system **must** provide the ability to recover from corrupted firmware or critical data for the host SoC and other components with mutable firmware, following the guidance in NIST SP 800-193 [1].*

2.6 Entropy source

In order to support key and nonce generation, a system should have a hardware entropy source, a true random number generator, that meets the requirements that are specified in the NIST SP 800-90 series of specifications [11].

A processor ISA implementation based on a TRNG, like the random number instructions defined for Arm architecture v8.5-A, or one implemented in a Secure element, like a certified TPM 2.0 can satisfy this requirement.

2.7 System Lifecycle Management

A system should have a mechanism for managing its security lifecycle. This mechanism governs the behavior of the system, both hardware and firmware, in each stage of the lifecycle, protecting any security assets that are introduced into the system and reducing the risk of IP theft and reverse engineering.

As a minimum, a system should provide a lifecycle control mechanism in which:

- The lifecycle state is held in, or derivable from, the state that is kept in on-chip, immutable non-volatile storage.
- All lifecycle state transitions are restricted to a designated set of states in which there is at least a designated initial state from which all systems start, a designated deployed state which mandates the

use of the system's security features, and a designated terminal state from which no further transitions are allowed. An example of a terminal state would be a product's end-of-life.

- A transition into the terminal state addresses how to deal with any secret and private cryptographic keys in the system.
- Booting, debugging and scan access are governed by a Secure lifecycle policy.

2.8 Identity

If confidential data is stored in an external storage device such as a system flash, the data must be protected by encryption using an on-chip key. This is to prevent attacks against the storage device itself, such as the use of physical probes or removing the storage device. A unique, unclonable, on-chip key is referred to as a Hardware Unique Key (HUK) in TBFU [7].

If a system requires an HUK:

- *R410_SBSG: The HUK **must** be entirely embedded in on-chip, immutable non-volatile storage such as OTP.*
- *R420_SBSG: The HUK **must** have at least 128 bits of entropy.*
- *R430_SBSG: The HUK **must** be accessible by the Secure world only.*

A system may require a unique, unclonable identity that is bound to the system's hardware. The identity can be used to verify the authenticity of the system. An example is remote attestation, where a remote challenger authenticates the identity of a remote system in order to implement security policies. An HUK or the endorsement key from a TPM can provide this unique hardware-bound identity.

2.9 Additional Guidelines

2.9.1 Memory Attributes

*R440_SBSG: For all firmware components, all memory that is marked executable in page table entries **must** not be marked writable.*

2.9.2 Secure Instruction Fetch

Secure instruction fetch, SCR_EL3[SIF], is a mechanism in the Arm architecture and should be set to prevent Secure world from executing Non-secure code. This setting can prevent attacks where the Secure world is made to branch to attacker-controlled code in the Normal world.

2.9.3 Interrupt Controller Configuration

SBSA requires a GIC v3 interrupt controller that supports Secure and Non-secure interrupts.

- *R450_SBSG: Interrupts for secure devices **must** be configured as secure in the GIC.*
- *R460_SBSG: GICD_CTLR[DS] (disable security) **must** be set 0.*

2.9.4 Warm Boot

If the device supports a fast boot path during CPU hotplug and secondary CPU boot operations, a flag or register should exist to enable distinguishing between cold and fast boot paths. This register or flag must be programmable only by the Secure world and must be automatically reset after a cold boot from the point of view of CPU firmware. For information about CPU hotplug and secondary boot see section 4.3, CPU hotplug and secondary CPU boot, in the PSCI specification [10].

2.9.5 Development/Manufacturing Backdoors

*R470_SBSG: Development or manufacturing backdoors **must** be disabled in a production system.*