



Base Boot Security Requirements 1.1

Document number: ARM DEN0107

Release Quality: REL

Confidentiality: Non-Confidential

Date of Issue: June 10, 2021

© Copyright Arm Limited 2021. All rights reserved.

Contents

| | |
|---------------------------------------|--------------------------------|
| About this document | iii |
| Release Information | iii |
| References | vi |
| Terms and abbreviations | vi |
| Potential for change | vii |
| Conventions | vii |
| Typographical conventions | vii |
| Numbers | vii |
| Status and anticipated changes | viii |
| Feedback | viii |
| Feedback on this book | viii |
| 1 | Introduction |
| | 9 |
| 2 | Security requirements |
| | 10 |
| 2.1 | Authenticated variables |
| | 10 |
| 2.2 | Secure boot |
| | 11 |
| 2.3 | Secure firmware update |
| | 12 |
| 2.4 | TPMs and measured boot |
| | 12 |
| 2.5 | Platform reset attacks |
| | 14 |

About this document

Release Information

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

| Date | Issue | Confidentiality | Change |
|---------------|-------|------------------|--|
| Oct 6, 2020 | 1.0 | Non-confidential | Initial release |
| June 10, 2021 | 1.1 | Non-confidential | <div>Updates<ul style="list-style-type: none">• Updates to sync with BBR now that it has the base firmware update requirements• Fix error in requirement numbering• Moved the db/dbx variable attributes requirements to the Secure Boot section• For measured boot, made creating the TPM event log an explicit requirement.• Removed reference to the deprecated <code>EFI_VARIABLE_AUTHENTICATED_WRITE_ACCESS</code> attribute• Minor cleanup/clarifications</div> |

Base Boot Security Requirements

Copyright ©2021 Arm Limited or its affiliates. All rights reserved. The copyright statement reflects the fact that some draft issues of this document have been released, to a limited circulation.

Arm Non-Confidential Document Licence (“Licence”)

This Licence is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“**Licensee**”) is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

- (i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;
- (ii) manufacture and have manufactured products which have been created under the licence granted in (i) above; and
- (iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANDING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PERMITTED 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.

Copyright © 2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.
110 Fulbourn Road, Cambridge, England CB1 9NJ.

Arm document reference: LES-PRE-21585 version 4.0

References

This document refers to the following documents.

| Ref | Document Number | Title |
|------|------------------------|---|
| [1] | DEN0044F | Arm® Base Boot Requirements |
| [2] | UEFI Specification 2.8 | Unified Extensible Firmware Interface Specification. Version 2.8 |
| [3] | DEN 0094A | Arm® Base System Architecture Version 0.8 |
| [4] | | TCG PC Client Platform Firmware Profile Specification, Family “2.0”, Level 00 Revision 1.04, June 3, 2019 |
| [5] | | TCG EFI Protocol Specification, Family “2.0”, Version 1.0, Revision 00.13, March 30, 2016 |
| [6] | | TCG ACPI Specification, Family “1.2” and “2.0”, Version 1.2, August 18, 2017 |
| [7] | | TCG PC Client Platform Physical Presence Interface Specification, Family “1.2” and “2.0”, Version 1.30, July 28, 2015 |
| [8] | | TCG PC Client Platform Reset Attack Mitigation Specification, Family “2.0”, Version 1.10, January 21, 2019 |
| [9] | PSA | Platform Security Architecture https://developer.arm.com/architectures/security-architectures/platform-security-architecture |
| [10] | PSA Certified | https://www.psacertified.org/ |

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. |
| Mutable | Changeable with respect to the platform of a system. |
| Normal world | The Non-secure privilege levels (Non-secure EL0, EL1, and EL2) and resources, for example memory, registers, and devices, that are not part of the Secure world. |
| Platform | The set of hardware and firmware mechanisms and services that an operating system and applications can rely on. |

| | |
|--------------|--|
| Secure world | The environment that is provided by the Secure privilege levels in the Arm v8-A architecture, S-EL0, S-EL1, S-EL2, EL3, and the resources, for example memory, registers, and devices, that are accessible exclusively from the Secure privilege levels. |
| TCG | Trusted Computing Group |
| TPM | Trusted Platform Module. A security module that is defined by TCG. |

Potential for change

The contents of this specification are subject to change.

Conventions

Typographical conventions

The typographical conventions are:

italic

Introduces special terminology, and denotes citations.

bold

Denotes signal names, and is used for terms in descriptive lists, where appropriate.

`monospace`

Used for assembler syntax descriptions, pseudocode, and source code examples.

Also used in the main text for instruction mnemonics and for references to other items appearing in assembler syntax descriptions, pseudocode, and source code examples.

SMALL CAPITALS

Used for some common terms such as IMPLEMENTATION DEFINED.

Also used for a few terms that have specific technical meanings, and are included in the Glossary.

Red text

Indicates an open issue.

Blue text

Indicates a link, which can be:

- A cross-reference to another location within the document.
- A URL, for example <http://infocenter.arm.com>.

Numbers

Numbers are normally written in decimal. Binary numbers are preceded by 0b, and hexadecimal numbers by 0x.

In both cases, the prefix and the associated value are written in a monospace font, for example 0xFFFF0000. To improve readability, long numbers can be written with an underscore separator between every four characters, for example 0xFFFF_0000_0000_0000. Ignore any underscores when interpreting the value of a number.

Status and anticipated changes

Alpha draft, revisions to be expected.

Rule identifiers are introduced when the document reaches REL quality.

Feedback

Arm welcomes feedback on its documentation.

Feedback on this book

If you have comments on the content of this book, send an email to <<feedback-address>>. Give:

- The title (<<title>>).
- The number and issue (<<Arm Document Unique ID>> <<version>> <<quality>> <<issue>>).
- 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

This document specifies security interface requirements and guidance for systems that are compliant with the Arm® Base Boot Requirements (BBR) [1] specification. BBR specifies the requirements for boot and runtime services that system software, like operating systems and hypervisors, can rely on. Meeting these requirements enables a suitably built operating system image to run on all compliant systems. BBR is based on industry firmware standards like UEFI and ACPI. The focus of BBR is standards-based boot and runtime services, and it does not address security.

This document identifies the platform requirements for BBR-based systems that enable standard, suitably built operating systems to seamlessly use standard security interfaces. These interfaces include the following security related functionality:

- UEFI authenticated variables
- UEFI secure boot
- UEFI secure firmware update using Update Capsules
- TPMs and measured boot

Compliance with this specification provides assurance that the security features in scope are implemented according to standards. However, compliance does not provide assurance that a platform is secure. In the process of architecting a system, system-level threat modeling should be performed to evaluate threats, risks, and mitigations. The Platform Security Architecture (PSA) [9] and the PSA Certified framework [10] do provide a comprehensive approach to platform security that is based on defined set of security goals. PSA provides architecture and requirements specifications for building secure platforms. This specification complements PSA. PSA Certified provides a measure of the robustness of an implementation, through an assessment process that is performed by a security certification laboratory.

2 Security requirements

In the following sections of this document, we describe the normative requirements in tables. These requirements are distinct from the supporting informative text. The informative text provides additional context to help clarify the rationale for each requirement.

Any system that is designed to conform to this specification must provide a complete implementation of the requirements that is specified in the following sections:

- Authenticated variables (section 2.1)
- Secure boot (section 2.2)
- Secure firmware update (section 2.3)

If the platform implements TPM-based measured boot, the implementation must comply with the requirements in the following section:

- TPMs and measured boot (section 2.4)

2.1 Authenticated variables

UEFI authenticated variables 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 are verified using digital signatures. The changes must be signed by an appropriate private key.

Authenticated variables must be protected from unauthorized modification. Arm recommends that the implementation of the protection of authenticated variables be considered as part of the platform threat model, which is beyond the scope of this specification.

Systems must implement support for UEFI authenticated variables as specified in the following table:

| ID | Requirement |
|-----------|---|
| R010_BBSR | Authenticated variables must be supported and be compliant with the following sections of the UEFI Specification [2]: Globally Defined Variables (section 3.3) Variable Services (section 8.2) |
| R040_BBSR | A minimum of 128 KB of non-volatile storage must be available for NV UEFI variables. There is no maximum non-volatile storage limit. |
| R050_BBSR | The maximum supported variable size must be at least 64 KB. |
| R060_BBSR | The platform must support EFI variables with any valid combination of the following UEFI variable attributes set: EFI_VARIABLE_NON_VOLATILE EFI_VARIABLE_BOOTSERVICE_ACCESS EFI_VARIABLE_RUNTIME_ACCESS EFI_VARIABLE_APPEND_WRITE |

| | |
|--|--|
| | EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS |
|--|--|

Table 1, UEFI Authenticated Variable Requirements

2.2 Secure boot

Secure boot is a process that cryptographically authenticates all code that runs on a system. Secure boot requires that all firmware is cryptographically signed. This enables the verification process to detect whether firmware components have been compromised or corrupted.

Secure boot begins in an immutable bootloader component, for example a boot ROM, that loads the first mutable firmware image. Before transferring control to the loaded image, the integrity and authenticity of the image is verified using digital signatures. The boot process continues with each image in the boot chain performing integrity and verification of the next image, before that image is executed or used. This process forms a chain of trust that is anchored in the immutable bootloader and continues through all code that is executed up to the runtime environment, for example the OS. UEFI Secure Boot is defined by the UEFI Specification [2].

Systems must implement support for UEFI Secure Boot as specified in the following table.

| ID | Requirement |
|-----------|--|
| R070_BBSR | System firmware must implement UEFI Secure Boot to prevent unauthorized EFI drivers, option ROMs, or programs from being executed during boot. |
| R080_BBSR | To support UEFI Secure Boot, the system firmware must be compliant with the following sections of the UEFI Specification: Runtime Services Rules and Restrictions (section 8.1) Variable Services (section 8.2) Secure Boot and Driver Signing (section 32) |
| R020_BBSR | To prevent rollback, the db signature database variable EFI_IMAGE_SECURITY_DATABASE must be created to include the EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS attribute. |
| R030_BBSR | To prevent rollback, the dbx signature database variable EFI_IMAGE_SECURITY_DATABASE1 must be created to include the EFI_VARIABLE_TIME_BASED_AUTHENTICATED_WRITE_ACCESS attribute to prevent rollback. |
| R090_BBSR | All UEFI images, for example drivers, applications, boot loaders, that are not contained in the system firmware image must have their signature verified in accordance with Secure Boot UEFI Image Validation in the UEFI Specification section 32.5. |
| R100_BBSR | System firmware must implement the Secure Boot variable as documented in Globally Defined Variables in the UEFI Specification section 3.3. |
| R110_BBSR | If authentication of a component fails, that component must not continue to load or execute. |
| R120_BBSR | 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. |

2.3 Secure firmware update

A secure firmware update process ensures that only authorized changes are permitted to the firmware in a system. This process ensures that firmware components maintain their integrity.

The Update Capsule architecture is defined by the UEFI Specification. This architecture provides a flexible mechanism to deliver and apply updates for system firmware components, for example Trusted Firmware-A or UEFI, or firmware for I/O devices in the system.

The Base Boot Requirements (BBR) [1] specification requires that in-band system firmware updates be implemented using UEFI Update Capsules. The following requirements must be implemented:

| ID | Requirement |
|-----------|---|
| R130_BBSR | In-band firmware updates must be implemented in accordance with the requirements in BBR: <ul style="list-style-type: none">• Firmware must implement UEFI update capsules (UEFI specification section 8.5.3)• Firmware must implement the Firmware Management Protocol Data Capsule Structure format (UEFI specification section 23.3)• Firmware must implement an ESRT that describes firmware updated in-band (UEFI specification section 23.4) |
| R140_BBSR | Capsule payloads for updating system firmware must be digitally signed. |
| R150_BBSR | Before updates to system firmware are applied, images must be verified using digital signatures. |

Note: Prior to the call to UpdateCapsule(), operating systems need to clean the cache by VA to Point of Coherency using the DC CVAC instruction on each ScatterGatherList element that is passed. This is needed only when UpdateCapsule() is called after ExitBootServices(). This requirement will be clarified in a future version of the UEFI specification.

Note: Capsules might be delivered through a file within the EFI system partition, as described in Delivery of Capsules via file on Mass Storage device in the UEFI Specification section 8.5.5.

2.4 TPMs and measured boot

This specification does not mandate use of a TPM. However, if a TPM is implemented, the requirements in this section must be followed.

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

- Platform Configuration Registers (PCRs) that securely store boot measurements and form the basis for a system to be able to perform secure attestation. PCRs provide one mechanism to implement security policies by sealing TPM objects to PCR values.
- Endorsement key that provides a unique, unclonable identity that is bound to hardware
- Key storage and management
- Secure cryptography in which 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 a security module with equivalent functionality. Some security models might include geography-specific requirements.

Note: The Arm Base System Architecture specification [3] specifies that, if a system implements a TPM, it must be compliant with version 2.0 of the TCG specifications.

| ID | Requirement |
|-----------|--|
| R170_BBSR | Mutable firmware components and critical data must be measured into PCR[0] through PCR[7] during boot, following the PCR usage guidelines in the TCG PC Client Platform Firmware Profile Specification Revision 1.04 [4]. |
| R180_BBSR | Mutable, Secure world firmware components must be measured into PCR[0]. |
| R190_BBSR | Signed critical data must be measured into PCR[0]. |
| R200_BBSR | All measurements that are made into TPM PCRs must be made with a SHA-256 or stronger hashing algorithm. |
| R210_BBSR | All measurements that are made into TPM PCRs must be logged in an event log compliant with the definition in the TCG PC Client Platform Firmware Profile Specification Revision 1.04 [4] specification. |
| R220_BBSR | For systems that implement system description using ACPI, a TPM 2.0 device must be advertised through ACPI tables, as specified in the TCG ACPI Specification [6]. This enables the TPM to be discovered by an operating system or hypervisor. |
| R230_BBSR | UEFI firmware must implement the EFI_TCG2_PROTOCOL as defined in the TCG EFI Protocol Specification Family 2.0 [5]. |
| R240_BBSR | If physical presence authorization for a TPM is implemented by firmware in a platform, the implementation must follow the requirements in TCG PC Client Platform Physical Presence Interface Specification [7]. The TCG specification describes two methods to provide physical presence authorization: the command method and the hardware method. Either method is acceptable to comply with this specification. |

Note: It is acceptable for the first mutable firmware component to measure itself, if that component has been verified by the immutable bootloader.

As required by the TCG PC Client Platform Firmware Profile Specification, 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_ELO_%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. |

Note: A TPM 2.0 can be implemented as a firmware component in a protected environment like the Secure world. The threats to a firmware TPM are different from a physical TPM. Threat modeling should be done to evaluate potential threats and mitigations.

2.5 Platform reset attacks

A platform reset attack occurs when an attacker with physical presence causes a system to be unexpectedly rebooted without a clean shutdown of the operating system. The attacker then makes the system boot on an alternate boot device, like a USB drive or DVD, into an OS that is under the control of the attacker. Actions like resetting the system, power cycling the system, or causing a kernel crash can cause the reboot. A key to the attack succeeding is that that volatile system memory can retain its contents across the attacker-forced reboot. The retention of memory contents can happen even when cycling the system 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 [8]. The TCG specification defines two UEFI variables that can be set by an operating system to mitigate the attack. The UEFI MemoryOverwriteRequestControl variable tells the firmware to clear memory prior to booting an operating system. The variable MemoryOverwriteRequestControlLock is a protection flag which prevents MemoryOverwriteRequestControl from being modified.

Note: When the UEFI MemoryOverwriteRequestControl variable, ACPI _DSM method, and PSCI memory protection API co-exist, Operating Systems may call any of these interfaces to mitigate the reset attack.