psacertified™

Platform Threat Model and Security Goals

1.0

Document number:	DEN 0112
Release Quality:	Alpha
Release Number:	0
Confidentiality	Non-Confidential
Date of Issue:	29/09/2020

Contents

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.

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 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

1 Introduction

This document captures the rationale behind the development of the PSA Certified framework and certification scheme. It describes this motivation by performing threat analysis of a typical platform and outlines the broad security goals to mitigate these threats. The platform is a collection of the fundamental hardware and firmware components needed to boot and operate a system. The platform is the foundation of the system. A successful attack on the platform could render a system inoperable and may result in substantial disruptions to users. The security goals in this document guide the Platform Security Model, a precursor for the development of security requirements and the solution architectures needed to design and deploy secure products within ecosystems. These security goals broadly apply to compute-centric systems with updateable firmware, including Internet of Things (IoT) devices, embedded systems, servers, automotive devices, and client devices like smartphones, laptops, smart TVs, and so on. Implementers, including Original Equipment Manufacturers (OEMs), component suppliers, and system software vendors can use the principles in this document to build stronger security mechanisms into platforms. System architects, security specialists, and users can use this document to guide procurement strategies and priorities for future systems.

2 Methodology

This section details the processes used to arrive at the threat model and security goals of a typical platform.

2.1 Platform definition

The purpose of defining the platform is to obtain a detailed picture of the object of analysis. This is imperative as the robustness of analysis relies on the amount of detail provided about the platform. The description includes an overview of the platform and its lifecycle, assets, and the stakeholders who have vested interest in these assets.

2.2 Threat analysis

This includes defining the adversarial model and enumerating identifiable threats to the platform.

2.3 Mitigations

Security goals that are essential to mitigate the identified threats are derived in this stage.

3 Platform definition

System architectures are layered, with the operating system and applications software forming the upper-most layers. While these provide most of the functional capabilities for the users, they rely on functions and services provided by the platform layer as shown in Figure 1.

The platform includes the hardware and firmware components that are necessary to initialize components, boot the system, and provide runtime services to the application and system software. Market requirements and use cases determine the platform and software components needed in a system. Figure 1 illustrates a generic system architecture that represents various classes of systems.

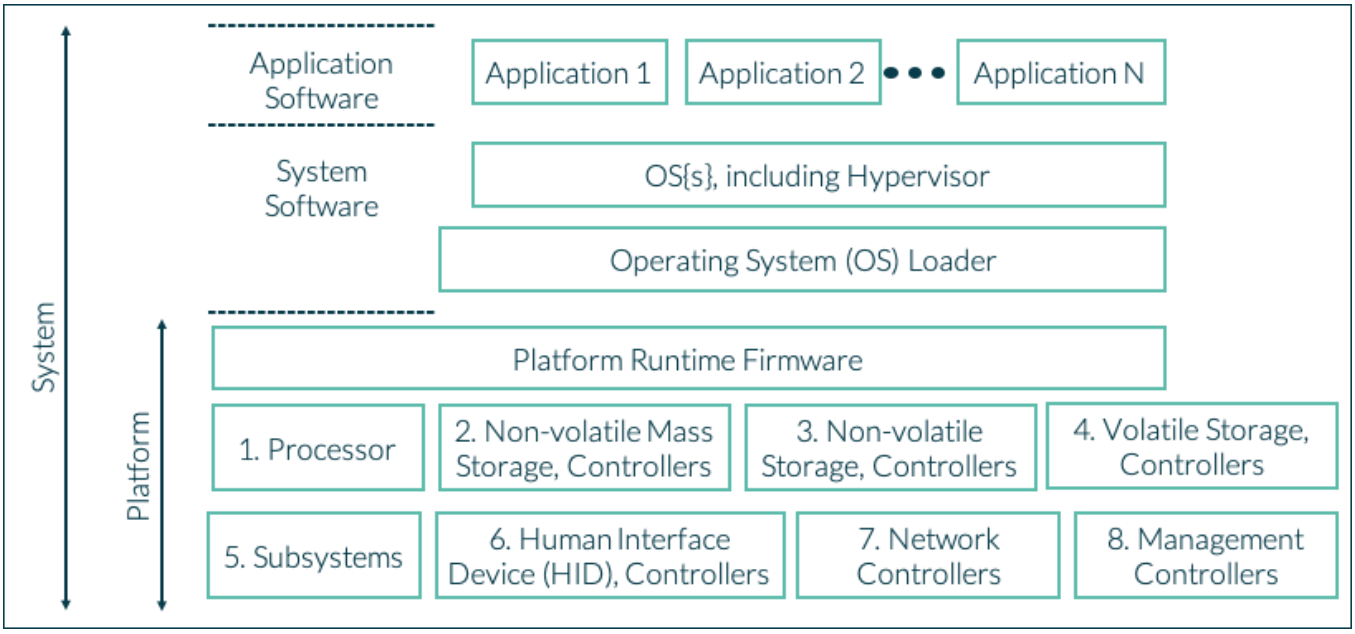

Figure 1: Platform and software components in a generic system

The components numbered in Figure 1 are described below.

1. Primary processing unit on which OS (and/or Hypervisor) as well as applications run
2. Solid State Drive (SSD), Hard Disk Drive (HDD)
3. Flash memory, One-Time Programmable (OTP) memory like fuses. Typical placeholder for processor firmware
4. Main memory like DRAM, caches, and so on
5. Trusted Platform Module (TPM), Secure Enclave (SE), Hardware Secure Element (HSE), and so on
6. Keyboard, Mouse, GPU in some platforms, and so on
7. Network Interface Controller (NIC) and other hardware to support connectivity options like wired, Wi-Fi, Cellular, Bluetooth
8. Out-of-band platform management devices like Baseboard Management Controller (BMC)

Many of the platform hardware components shown in Figure 1 have firmware and configuration data that drive their behavior, which must remain in a state of integrity for the system to function correctly.

Many systems support security critical use cases. For example, mobile wallet and payment service in a mobile device, confidential computing in a server/cloud environment, relaying secure smart metering information in embedded systems, fleet management in IoT device, and so on. These applications process sensitive user and system data that must be protected from attackers and may require support from the platform layer, for example using platform cryptographic service to encrypt network communication data.

The systems receive over-the-air (OTA) updates to provide feature updates for better user experience, fix bugs, patch security vulnerabilities, and so on. Users of the system may also install third-party applications. These scenarios make the systems susceptible to remote attacks.

The ever-increasing complexity of embedded systems aggravates the security problem. As the platform forms the foundation of the system, any compromise to the integrity of the platform compromises the integrity of the system. This necessitates a recipe to build secure platforms.

3.1 Platform Lifecycle

A generalized lifecycle diagram of a typical platform is shown in Figure 2. Each platform component may have its own lifecycle, for example, trusted subsystems and component firmware. The overall system may also have its own lifecycle. Such lifecycles, and any associated data, are implementation specific and are outside the scope of this document. However, such lifecycles must never be in a state that conflicts with or compromises the security of the platform.

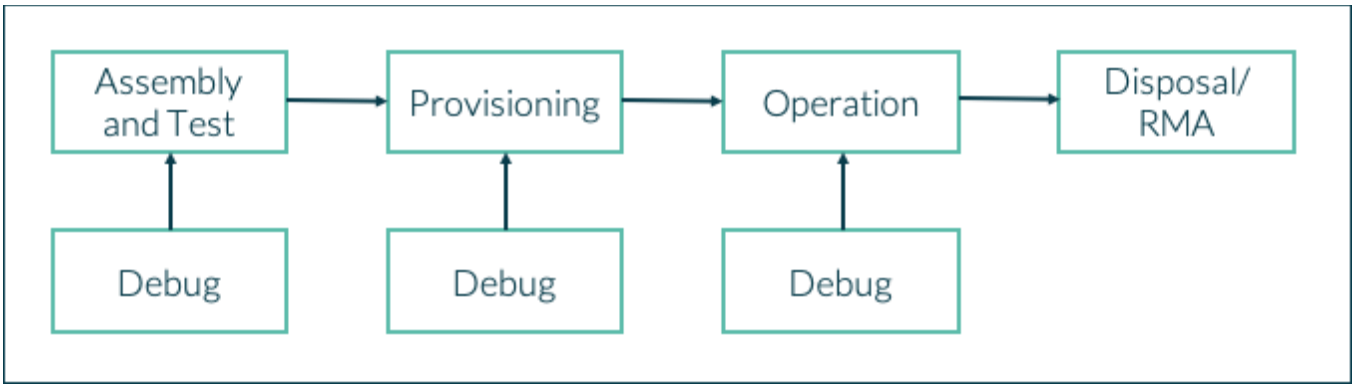

Figure 2: Generic platform lifecycle

The different phases in the platform lifecycle are described in the following table. Unless otherwise mentioned, the term *software* refers to platform firmware.

Phase	Description
Assembly and test	• Components from multiple vendors are sourced, assembled, and tested for manufacturing defects in this phase. The components may contain secrets like cryptographic keys fixed in hardware • Manufacture and diagnostic software may be run. Some test secrets and identities may be present • In this phase, the platform may have a slightly different configuration from the provisioning phase. For example, debug features like hardware debug ports may be open to triage software defects
Provisioning	• Provisioning of secrets happens in this phase. For example, injection of cryptographic assets such as unique keys • Production software and production parameters like implementation ID, instance ID, and so on are provisioned • Hardware debug ports should be disabled or managed through an access control mechanism

Operation	A typical operational phase consists of boot and run-time phases. Boot: • At reset, the boot flow starts code from non-volatile storage like boot ROM that loads firmware bootloader which in turn loads and runs additional firmware components that culminate in an OS or Hypervisor being loaded. Modern OSes have larger footprint (code) and complexity compared to their predecessors. So, it is reasonable to assume that they have not received substantial security evaluation • Otherwise, some platform services are trusted as they are certified by trusted certification entities. A platform may host trusted services from multiple mutually mistrusting vendors and may need to support interactions within these services and other untrusted software. From platform point of view, examples of untrusted software include system and application software Run-time: • The device owner or user runs applications. Sensitive user and platform data need to be protected from untrusted software • Any problem during the operational phase may require a reset to factory settings and repeat the operational phase. The device owner may approach a repair vendor for issues • Configuration operations to support various use cases may be permitted. For example, some network ports and/or sockets relating to services that are not required
Disposal/return merchandize authorization (RMA)	• If a major fault develops during the operational phase, a return to the manufacturer is required to triage issues. The manufacturer 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

3.2 Assets

From the lifecycle diagram and use cases, a generalized list of assets includes:

- Primary assets
    - Critical data
        - System data. For example, configuration data associated with platform firmware, access policies, audit logs, and so on
        - User data. For example, user configuration settings, a DRM certificate purchased by the device user to playback stored or streamed media, credentials like biometrics, credit card information in a mobile wallet and payment service, and so on
    - Firmware that includes firmware bootloader, platform component firmware, and services

- Auxiliary assets
    - For example, cryptographic keys, debug ports, Direct Memory Access (DMA) capable IP, communication interfaces within and between platform firmware and software layers as shown in Figure 1, and so on that if compromised can be used to undermine confidentiality and integrity of primary assets

While the stakeholders for system data, firmware and auxiliary assets are chip manufacturers and OEM, the stakeholders for user data are device owners and/or users.

4 Threat analysis

4.1 Adversarial Model

An adversary or an attacker is a threat agent (a person or a process acting on behalf of an agent) that tries to undermine the platform security policy and the security functionality. The attacker can try to change the properties of the assets defined above. When considering the threats and attacks on a system, it is widely recognized that these must be divided into two phases.

1. The identification phase that has an attacker discover a vulnerability and create an exploit to gain access to an asset
2. The exploitation phase when the exploit is executed on one or more systems by the same or different attackers. If different attackers are performing the exploit, it is expected that the identification phase attacker has prepared the exploit as either a set of detailed instructions or a piece of software

A secure implementation must consider both the phases. The system should be designed to resist widespread exploitation, because this form of attack is often the most damaging to the asset owners.

The following table contains an adversary model that all platforms should be protected against.

Attacker	Type	Access to System	Equipment
Software attackers	• Remote attackers • Attackers with physical access during the identification phase of an attack that can later be exploited remotely	Multiple production devices in case of attackers with physical access	Network connectivity for remote access of the target device
Simple hardware attackers	• Simple hardware attackers with limited resources, knowledge, or equipment • Attackers with temporary legitimate access like a smartphone repair vendor or device reseller • Thieving attacker who steals the device	Multiple production systems	From simple USB interface to relatively inexpensive and easy to acquire equipment. For example, JTAG interfaces controllers, soldering irons, and oscilloscopes.

4.2 Threats

Considering platform definition, platform lifecycle, assets requiring protection, and adversarial model, the STRIDE methodology (Spoofing, Tampering, Repudiation, Information, Denial-of-Service and Elevation-of-Privilege) is used to identify corresponding attack scenarios. This document does not provide comprehensive analysis of Denial-of-Service attacks as they are specific to market segments and use cases.

Category	Threats	Adversarial Model	Description
Spoofing	Spoofing the system identity	Software attack	Spoofing the initial measured state of the system (by exploiting weakness in cryptography or intercepting network communication data) to fake trustworthiness of the system to a requesting entity.
	System cloning	Software attack	Cloning firmware components to create a system that appears to have same identity as the authentic system. This threat also applies to app cloning, that is, spoofing application-specific data and secrets to clone the application on other systems.
	Spoofing the firmware update sender's identity	Software attack	Spoofing the identity of the system manufacturer or another authorized entity to fake trustworthiness of the firmware update image.
Tampering	Image tampering	Software attack	Firmware to be run on the system may be tampered and executed to gain control of the system execution and configuration in order to access assets. For example, installation of untrustworthy and/or tampered firmware, execution of unauthorized firmware, and so on.
	Rollback	Software attack	Firmware image and sensitive configuration like manifest data on the system may be subjected to unauthorized version rollback to exploit a version with a legacy bug.
	Data tampering	Hardware attack	Assets in off-chip storage like removable media may be altered by accessing them physically. For example, replacing the storage, bus interposition to corrupt or inject new memory transactions, and so on.
		Software attack	A software component may try to tamper with the private data belonging to another software component. For example, the scenario is possible within and between trusted services and less trusted software due to unrestricted access to physical memory.

	Persistent malware	Software attack	Vulnerabilities in firmware may be exploited to install malicious software that remains persistent across reset.
Repudiation	Deniability	Software attack	An attacker may compromise the confidentiality and integrity of the assets and deny any wrongdoing. For example, erasing security events through unauthorized access to audit logs. Exploiting weak cryptography to forge identity, lack of appropriate track controls, and so on.
Information disclosure	Side-channel attacks	Software attack	Sensitive data stored in the system may be extracted through software-initiated side-channel attacks like speculative-based cache timing side-channel attacks.
	Data extraction	Hardware attack	Sensitive data in the system may be extracted through physical access. For example, DRAM interposition or eavesdropping on the buses or retrieving contents from an off-chip storage like external flash memory.
		Software attack	Sensitive data in the system may be extracted by abusing communication interfaces, for example, between the application or system software and the firmware layer, exploiting weak cryptography, and so on.
Elevation of privilege	Abusing system configuration and/or lifecycle	Hardware attack	Debug interfaces may be used to elevate privileges or obtain assets. For example, assets in the system may be obtained during device repair or RMA or end of lifecycle phase of the system or by altering the system lifecycle to cause privilege escalation attacks.
	Abusing communication interfaces	Software attack	Interactions within and between trusted services and less trusted software may be abused to mount privilege escalation attack, such as confused deputy attack.
	Unrestricted access to system resources	Software attack	A trusted service may be subjected to privilege escalation from a less trusted software or another trusted service through unrestricted access to shared system resources like memory, interrupts, and others. Also, external off-chip peripherals may be replaced or exploited to gain elevated privileges (such as Direct Memory Access transactions) either to attack the system or to directly access assets.

- Some platform implementations 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
    - Active non-invasive attacks like fault injection attacks that involve power, clock, temperature, and energy glitches to cause faults such as instruction skipping, data corruption while reading from or writing to memory or instruction decode errors
    - Semi-invasive attacks in which systems are subjected to fault injection through optical techniques

  A security risk assessment of the product is needed to evaluate the protection against these attacks. Mitigations are dependent on the product implementation and are not described here.
- Recovery mechanisms of firmware and critical data when corruption is detected, or when instructed by an administrator is dependent on the product implementation and are not described here
- Some market requirements and use cases may demand that certain types of DoS attacks be addressed by secure platforms. For example, availability of platform services, low latency support to service requests, managing platform resources like memory in case of fatal errors, and so on. Guidance or mitigations to such attacks may need to be dealt in the solution architectures. However, the security goals outlined in this document provide mitigations against attacks that can render a class of systems unusable. For example, corrupting firmware to mount a permanent DoS attack on connectivity hardware like NIC, sending an unmodified but incompatible firmware update, and so on.

The following threats are outside the scope of this document

- Threats to system software and application software stacks
- Physical attacks
    - Invasive attacks in which systems are unpackaged and subjected to microprobing
    - Supply chain security attacks
- Denial of service attacks like Distributed DoS attacks, application-layer attacks, and so on that occur outside the device

Mitigations against these attacks may be necessary to meet specific target market security requirements and should be dealt at the product implementation level.

5 Security Goals

Considering the security threats identified in earlier sections, the following high-level security goals are required as countermeasures to build secure platforms. Abstraction allows these goals to be applied as required, for example, to an end user device, a platform hardware component, or a software layer as shown in Figure 1. In describing the goals, the term *device* is used to represent any entity that must be secure and trustworthy.

Security Goal	Description
Unique identification	**Devices shall be uniquely identifiable.**
Security lifecycle	**Devices shall support a security lifecycle.** The security state of a device depends on software versions, run-time measurements, hardware configuration, status of debug ports, and the device lifecycle phase. The security states shall be attestable and may impact access to data that is bound to the device.
Attestation	**Devices shall be securely attestable.**
Software authorization	**Devices shall ensure that only authorized software is executed.** Secure boot and secure loading processes are necessary to prevent unauthorized software from being executed.
Secure update	**Devices shall support secure update of software, or platform critical data like hardware configuration.** Software update shall be validated for authenticity and legitimacy before being installed on the system. However, execution of any updated software must follow the principles outlined in *software authorization* security goal described in this table.
Anti-rollback	**Devices shall prevent unauthorized rollback of updates.** Preventing rollback to a known version with security vulnerabilities is essential. However, rollback for recovery purposes may be allowed if authorized.
Isolation	**Devices shall support isolation.** Isolation of trusted services from one another and from less trusted services is essential to protect confidentiality and integrity of that service.
Interaction	**Devices shall support interaction over isolation boundaries.** The interfaces must not be used to compromise confidentiality and integrity of the device.
Device binding of stored data	**All devices shall support unique binding of stored sensitive data to the device.**
Cryptographic and trusted services	**All devices shall support a minimum set of trusted services and cryptographic operations that are necessary to support other security goals.** Trusted services may include configuration of the hardware to support security lifecycle, isolation, and so on. Cryptographic services may be used to manage critical functions like security lifecycle, attestation, secure boot, secure loading, and device binding.

- For software side-channel attacks, mitigations should be implemented at the processor level as this is where software runs. The firmware interfaces between system or application software and platform services should also be analyzed to protect the confidentiality and integrity of the assets

The following table maps security goals to the threats they intend to protect against. "N/A" means that a security goal does not thwart the corresponding threat. Mitigations for DoS and physical attacks described in the *Threats* section in this document are specific to product implementation and are not described in this document.

Security Goals/Threats	Spoofing	Tampering	Repudiation	Information Disclosure	Elevation of Privilege
Unique identification	Spoofing the system identity	N/A	N/A	N/A	N/A
Security lifecycle	N/A	N/A	N/A	N/A	Abusing system configuration and/or lifecycle
Attestation	Spoofing the system identity, spoofing the firmware update sender's identity	Image tampering, persistent malware	Deniability	N/A	Abusing system configuration and/or lifecycle
Software authorization	N/A	Image tampering, persistent malware	N/A	N/A	N/A
Secure update	Spoofing the firmware update sender's identity	Image tampering	N/A	N/A	N/A
Anti-rollback	N/A	Rollback	N/A	N/A	N/A
Isolation	N/A	Persistent malware, data tampering (software attack)	N/A	Data extraction (software attack)	Unrestricted access to system resources
Interaction	N/A	Data tampering (software attack)	N/A	Data extraction (software attack)	Abusing communication interfaces

Device binding of stored data	System cloning	Data tampering (hardware attack)	N/A	Data extraction (hardware attack)	N/A
Cryptographic and trusted services	From the threats and security goals, a minimal set of trusted services and cryptographic operations should be implemented as the building blocks of a trusted system. Refer *Security Goals* section in this document for more details.				

6 Conclusions

From the threat model, it is clear that many systems require a high level of security assurance. The PSA Certified framework provides a number of levels of assurance that the security goals outlined by this document are met, recognizing that security must be addressed at both hardware and software levels. The PSA Certified deliverables and resources follow principles that are the basis for best practices and security requirements specified in target markets. PSA Certified aims to help partners and industry stakeholders accelerate the development and deployment of secure products.