RISC-V Security Model

RISC-V Security Model Task Group

Version 0.11, 9/2022: This document is in development. Assume everything can change.

Preamble

This document is in the Development state

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by:

- Andrew Dellow
- Dong Du
- Colin O'Flynn
- Munir Geden
- Yann Loisel
- Manuel Offenberg
- Ravi Sahita
- Suresh Sugumar
- Steve Wallach

Table of Contents

1. Overview

1.1. Introduction

Security has often been a late consideration in the development of systems, hardware, and software. The emergence of exploits such as malware, trojans, the recent Spectre, Meltdown, RAMbleed attacks has resulted in serious financial and reputation losses. This illustrates the need to consider security as an essential component, directly built into a system rather than layered on top.

Unlike other commercial architectures such as X86 and ARM which carry a lot of legacies, RISC-V is a clean slate architecture that can invite a whole lot of new features and solutions both in hardware and software. Below are some key rationale for improving and accelerating RISC-V security

- Clean slate architecture with no legacy support complexity
- Open security model accelerates hardware security innovation
- Opportunity to incorporate industry learnings & best practices for security
- Open governance facilitates collaboration on the best security approach
- Royalty-free model enables wide access to new hardware security solutions thereby democratizing innovation.

1.2. Scope

1.2.1. Device Model

This diagram below represents a generic device model having an application processor, other IPs, etc. on the left side and root-of-trust on the right side that is isolated from each other. The only communication between the two domains is via a security mailbox.

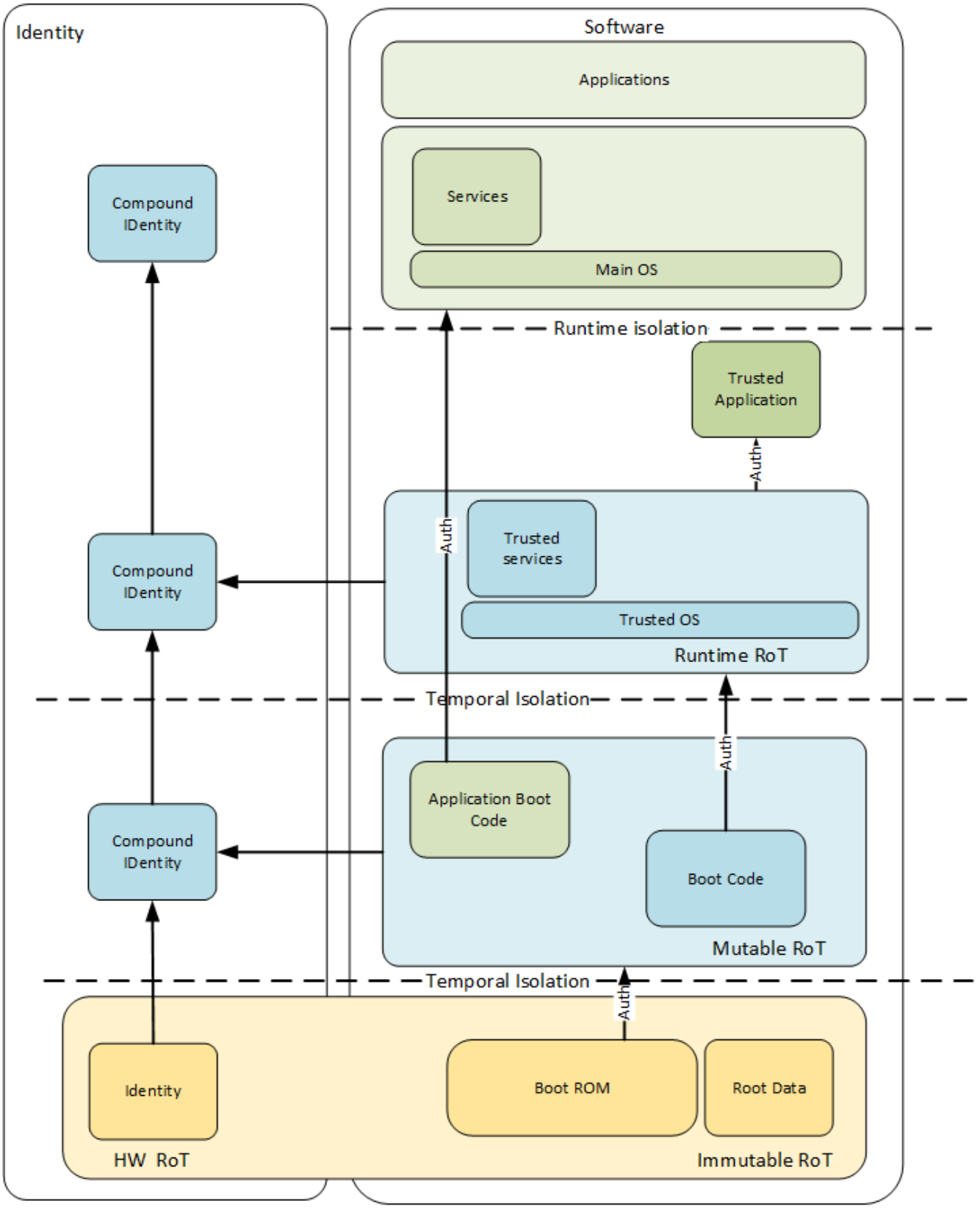

Figure 1. Device Model

The root-of-trust consists of the following elements:

1. An immutable code and data that are typically programmed into ROM/ OTP and root-keys either provisioned into OTP or generated every time from the PUF (physical unclonable function) hardware
2. A mutable code and data that are updated onto the flash to fix bugs or enhance features, which are authenticated and verified before consumption

3. A secure storage mechanism to store secrets/ assets bound to the platform, ex: root-key
4. A secure mailbox is the only mechanism to communicate between both the domains

1.2.2. Recommended Compositions

- Cloud/HPC
- Edge
- high-end embedded
- low-end embedded

Device Models: virtualized vs no-virtualized

1.3. Document Organization

2. Definitions & Conventions

2.1. Definitions & Abbreviations

2.1.1. Definitions

2.1.2. Abbreviations

CFI: Control Flow Integrity CSP: Critical Security Parameter

DRBG: Deterministic Random Bit Generator

ePMP: enhanced PMP

PMP: Physical Memory Protection

PQC: Post-Quantum Cryptography

RoT: Root of Trust

TEE: Trusted Execution Environment

2.2. Conventions & Normative Text

2.2.1. Conventions

2.2.2. Normative vs. Informative Text

3. References

This section provides normative and informative references to documents that, in whole or in part, are indispensable to the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

1. https://www.intel.com/content/www/us/en/newsroom/opinion/zero-trust-approach-architecting-silicon.html
2. https://www.forrester.com/blogs/tag/zero-trust/
3. https://docs.microsoft.com/en-us/security/zero-trust/
4. https://github.com/riscv/riscv-crypto/releases
5. https://github.com/riscv/riscv-platform-specs/blob/main/riscv-platform-spec.adoc
6. https://www.commoncriteriaportal.org/files/ppfiles/pp0084b_pdf.pdf
7. https://docs.opentitan.org/doc/security/specs/device_life_cycle/
8. https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8320-draft.pdf
9. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-193.pdf
10. https://www.rambus.com/security/root-of-trust/rt-630/
11. https://docs.opentitan.org/doc/security/specs/
12. https://trustedcomputinggroup.org/work-groups/dice-architectures/
13. https://ieeexplore.ieee.org/iel7/8168766/8203442/08203496.pdf
14. https://dl.acm.org/doi/10.1145/168619.168635
15. https://dl.acm.org/doi/abs/10.1145/3342195.3387532
16. https://github.com/riscv/riscv-debug-spec/blob/master/riscv-debug-stable.pdf
17. https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf
18. https://www.iso.org/standard/60612.html
19. https://ieeexplore.ieee.org/document/6176671
20. https://tches.iacr.org/index.php/TCHES/article/view/8988
21. https://ieeexplore.ieee.org/abstract/document/1401864
22. https://www.electronicspecifier.com/products/design-automation/increasingly-connected-world-needs-greater-security
23. https://www.samsungknox.com/es-419/blog/knox-e-fota-and-sequential-updates
24. https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/supply-chain-malware
25. https://dl.acm.org/doi/10.1145/3466752.3480068
26. https://arxiv.org/abs/2111.01421
27. https://www.nap.edu/catalog/24676/foundational-cybersecurity-research-improving-science-engineering-and-institutions

28. https://trustedcomputinggroup.org/work-groups/dice-architectures/

4. Threats & Security Objectives

4.1. Threats

4.1.1. Pointer Safety

Asset:	Pointers
Location:	Memory
Description:	Pointers stored in programs to store addresses
Security Property:	Integrity
Threat:	Tamper
Entry Point of Threat:	Misusing pointers to access unauthorized memory, manipulating stack, heap regions, executing data pointers, use after free, out of range access, etc
Impact of Vulnerability:	Memory misuse
Severity CVSS v3 Rating:	HIGH: 7.5 CVSS v3.1 Vector AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Extending pointer virtual address width or using unused bits if any of pointer virtual address to hold type, permissions, and tag inserted by malloc function and checked during page, walk to prevent memory misuse

4.1.2. Stack Safety

Asset:	Stack
Location:	Memory/ CPU Registers
Description:	System Stack
Security Property:	Integrity
Threat:	Tamper
Entry Point of Threat:	Return Oriented Programming (ROP) attack using stack smashing by either buffer overrun or injecting code into the stack
Impact of Vulnerability:	Program control-flow hijack
Severity CVSS v3 Rating:	HIGH: 7.5 CVSS v3.1 Vector AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)

Mitigation/Security Requirement:	Use shadow stack to compare return addresses for control-flow transfer instructions if a mismatch is detected then raise an exception to the kernel to handle it

4.1.3. Call/ Jump Safety

Asset:	Call/ Jump Targets
Location:	Memory/ CPU Registers
Description:	Indirect call/ jump target addresses
Security Property:	Integrity
Threat:	Tamper
Entry Point of Threat:	Call/ Jump Oriented Programming (COP/ JOP) attack using ata tampering to perform indirect call/ jump to invalid locations
Impact of Vulnerability:	Program control-flow hijack
Severity CVSS v3 Rating:	HIGH: 7.5 CVSS v3.1 Vector AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Track indirect call/jump instructions and permit only valid call/jump locations of the code

4.1.4. Code/ Data Confidentiality

Asset:	Code/ Data
Location:	Memory/ CPU Registers
Description:	Software Code and Data
Security Property:	Confidentiality
Threat:	Disclosure
Entry Point of Threat:	Vulnerable OS/ VMM can be exploited with privilege escalation o tamper code/ data of an application or hosted software
Impact of Vulnerability:	Compromised confidentiality of secrets
Severity CVSS v3 Rating:	HIGH: 7.5 CVSS v3.1 Vector AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Encrypt code/ data via hardware mechanisms with hardware enerated keys invisible to OS/ VMM

4.1.5. Code/ Data Integrity

Asset:	Code/ Data
Location:	Memory/ CPU Registers
Description:	Software Code and Data
Security Property:	Integrity
Threat:	Tamper
Entry Point of Threat:	Vulnerable OS/ VMM can be exploited with privilege escalation to tamper code/ data of an application or hosted software
Impact of Vulnerability:	Compromised integrity of interesting assets, eg: code
Severity CVSS v3 Rating:	HIGH: 7.5 CVSS v3.1 Vector AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:L/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Integrity check (is a threat protection mechanism that checks the drivers and system files on your device for signs of corruption) of code/ data by hardware that is attested by the hardware which can be verified locally/ remotely. Integrity checking should/shall be a permanently running mechanism.

4.1.6. Timing Side-Channel Safety

Asset:	Any secret (see section 5.14)
Location:	Cache, TLB, Memory
Description:	Leakage
Security Property:	Confidentiality
Threat:	Disclosure
Entry Point of Threat:	Covert channel - Spy & Trojan attacking the victim
Impact of Vulnerability:	Disclosure of secret
Severity CVSS v3 Rating:	HIGH: 6.2 CVSS v3.1 Vector AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)

Mitigation/Security Requirement:	Timing protection (temporal partitioning) to prevent interference that affects observable timing behavior. The new fence.T ISA extension proposed for RISC-V for temporal partitioning prevents any interference between security domains, each such microarchitectural state must be reset to a state that is independent of execution history before a context switch to a different thread/ process.

4.1.7. Hardware Supply Chain Safety

Asset:	Hardware IP
Location:	Design (GDSII)
Description:	IP theft, Counterfeiting, Overproduction
Security Property:	Confidentiality
Threat:	Disclosure
Entry Point of Threat:	Design in GDSII form
Impact of Vulnerability:	Loss of IP, Loss of revenue
Severity CVSS v3 Rating:	HIGH: 4.6 CVSS v3.1 Vector AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Logic locking is one of the new emerging technology that enables the hardware to lock the IP/ SoC using a password only known to the design house and can only be unlocked after the parts come back to the design house. Without this password, the IP/ SoC is literally defunct or unusable.

4.1.8. Software Supply Chain Safety

Asset:	Software IP
Location:	Software/ Application binary
Description:	Cloning, Tampering
Security Property:	Confidentiality, Integrity
Threat:	Disclosure, Tamper
Entry Point of Threat:	Build tools, build servers, release servers, etc
Impact of Vulnerability:	Loss of IP, Loss of revenue

Severity CVSS v3 Rating:	HIGH: 4.6 CVSS v3.1 Vector AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Encryption, Attestation, and protection of code signing certificates, build tool attestation, etc

4.1.9. Peripheral/ IP Authentication

Asset:	Peripherals/ IPs
Location:	SoC/ Platform
Description:	Fake/ rogue Peripheral/ IP communicating with the victim
Security Property:	Integrity, Availability
Threat:	Disclosure, Tamper
Entry Point of Threat:	Procurement channels
Impact of Vulnerability:	Insecure products
Severity CVSS v3 Rating:	HIGH: 5.2 CVSS v3.1 Vector AV:P/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:L (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	Peripheral/ IP mutual authentication. Recent developments in the industry to address this concern include opencompute.org, dmtf.org, and pce.org where they propose peripheral extensions to enable mutual authentication and encrypted communication among N parties on the platform. This could be extended to even to the IP level inside the SoC, which needs careful evaluation to make sure the trade-offs for PPA$ are worth the additional security it offers for the particular product.

4.1.10. Non-CPU IPs/ Peripherals outside TEE

Asset:	Peripherals/ IPs
Location:	SoC/ Platform
Description:	Non-CPU IPs & Peripherals are outside scopes of TEE and hence the code & data do not get any security guarantees from the TEE, and so are unprotected
Security Property:	Confidentiality, Integrity, Availability
Threat:	Disclosure, Tamper, DoS
Entry Point of Threat:	Untrusted OS/ VMM

Impact of Vulnerability:	Weak security to code/ data
Severity CVSS v3 Rating:	HIGH: 7.2 CVSS v3.1 Vector AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H (nvd.nist.gov/vuln-metrics/cvss/v3-calculator)
Mitigation/Security Requirement:	TEEs need to be extended to include non-CPU IPs such as GPU, etc., and peripheral devices into the enclave.

4.2. Security Objectives

The security objectives are high-level essential security features for a platform to implement, for the product to be secure and trustworthy. Note that depending on the final use case, some or all of these goals may not be required and that this will be defined in the mapping to platform specifications (TBD - point to the mapping section later).

#	Goal	Description
1	A platform is uniquely identifiable	A platform shall have an immutable identity that is both verifiable and attestable
2	A platform shall only execute authorized software	A platform shall verify and authenticate any software before execution
3	A platform shall support device bound storage	A platform shall support a secure storage mechanism to store keys/ secrets that are tied to a particular platform (confidentiality & integrity protection of secret keys, integrity protection for public keys)
4	A platform shall support secure TCB update	A platform shall verify and authenticate any software image updates before loading/ storing/ executing them
5	A platform shall prevent software version rollback	A platform shall prevent software version rollback with anti-rollback mechanism using techniques like monotonous counters
6	A platform shall support security through its lifecycle	A platform shall support security through various lifecycle stages such as development, deployed, returned, end-of-life, etc
7	A platform shall support isolation for code & data	A platform shall support both temporal and spatial isolation for security sensitive code and data
8	A platform shall implement all zero-trust principles	See section 3
9	A platform shall offer crypto services	A platform shall offer classical crypto & optionally post-quantum crypto operations

#	Goal	Description
10	A platform shall protect customer sensitive data	A Platform shall offer confidentiality, integrity, and authenticity protection to sensitive data
11	A platform shall establish its trustworthiness remotely	A platform shall be remotely attestable to prove its trustworthiness and security properties
12	Security guidelines matched to product segment/ profile	A platform shall follow the security guidelines below to incorporate the right level of security for its product segment/ profile needs
13	Software toolchains shall enforce security properties	A platform toolchain shall provide security, ex: C compiler when compiling to WebAssembly certain security features like stack canary are being omitted, and so the hardware cannot also protect
14		

4.3. Adversary Model

The following are the adversary models we consider for this document:

#	Adversary	Description
1	Unprivileged Software Adversary	This includes software executing in U-mode. Application workloads are typically being managed by S/M-mode system software. This adversary can access U-mode CSRs, process/task memory, CPU registers in the process context.
2	System Software Adversary	This includes system software executing in S and HS-modes. Such an adversary can access privileged CSRs, all of the system memory, CPU registers, and IO devices.
3	Startup Code Adversary	This includes system software executing in early/boot phases of the system, including BIOS, memory configuration code, device option ROM/firmware that can access system memory, CPU registers, IO devices, and IOMMU, etc.
4	Simple Hardware Adversary	This includes adversaries that can use hardware attacks such as bus interposers to snoop on memory/device interfaces, which may give the adversary the ability to tamper with data in memory.

#	Adversary	Description
5	Advanced Hardware Adversary	This includes adversaries that can use advanced hardware attacks, with unlimited physical access to the devices, and use mechanisms to tamper with the hardware TCB e.g., extract keys from hardware, using capabilities such as scanning electron microscopes, fib attacks, glitching attacks, etc.
6	Side/ Covert Channel Adversary	This includes adversaries that may leverage any explicit/implicit shared state (architectural or micro-architectural) to leak information across privilege boundaries via inference of characteristics from the shared resources (e.g. caches, branch prediction state, internal micro-architectural buffers, queues). Some attacks may require the use of high-precision timers to leak information. A combination of system software and hardware adversarial approaches may be utilized by this adversary.

text

5. Memory Protection

5.1. Privilege Modes

M / ? / H / S / U Min. / small TCB sec properties

5.2. Physical Memory Protection

5.2.1. Enhanced PMP

5.2.2. S-mode PMP

5.2.3. I/O PMP

5.2.4. I/O MMU

secure vs non-secure view using PMPs

5.3. Memory Security

encryption/integrity/replay protections

5.4. Link Security

Link protections, e.g., PCIe, CXL, inter-die

6. Cryptography

6.1. Recommended Primitives, Modes, and Algorithms

6.2. Randomness

6.2.1. Entropy Sources

6.2.2. Deterministic Random Bit Generators

6.3. Critical Security Parameters

6.3.1. Strength

length / strength entropy

6.3.2. Management

Storage / Access controls

6.3.3. Life-Cycle

generation, derivation, storage, destruction

6.4. Constant Time Execution

Execution time independent of data being computed on

6.5. Post Quantum Cryptography

note, SP800-208

7. Platform Identity

7.1. Root keys

7.2. PUFs

7.3. OTP

7.4. DICE

8. Root-of-Trust & Secure Boot

8.1. Root-of-Trust

8.1.1. Services

Measurements, Integrity, Storage, etc.

8.1.2. Use Models

RoT uses case, e.g., PRoT, RoT for measurement , RoT for Update, RoT for Recovery (see OCP)

8.2. Secure Boot

8.2.1. Verified Boot

Definition: Code is only loaded if it is properly signed s

8.2.2. Measured Boot

Definition: TCG Trusted boot Measurements

9. Attestation

9.1. Local vs Remote

9.2. Protocols

```
e.g., TPM, SPDM
```

10. Run-time Integrity

10.1. Pointer Masking

10.2. Memory Tagging

(Memory Coloring)

10.3. Control Flow Integrity

10.3.1. Landing Pads

10.3.2. Shadow Stack

11. Trusted Execution Environments

11.1. Memory Isolation

Running multiple workloads (> 2) on one processor Memory protection with PMP/SPMP and external checkers (IOPMP) Initiator-side and/or target-side access control

11.2. TEE Models

11.3. Confidential Compute

Appendix A: Zero Trust Principles

#	Principle	Description	Example
1	Verify Explicitly	Verify every access explicitly without any trust assumptions	Data packet from any sender needs to be authenticated before using it to ensure non-repudiation
2	Least Privilege	A subject should be given only those privileges that it needs to complete its task	An entity having access to unauthorized resources, can lead to security vulnerabilities such as data leakage
3	Assume Breach	Assume everything eventually gets broken	A rogue peripheral attached, or a malware installed in a system can lead to system compromise
4	Fail Securely	Ensure that error conditions don't leave secrets around	Any secrets left in the memory after a reboot can lead to data leakage
5	Complete Mediation	Unless a subject is given explicit access to an object, it should be denied access to that object	Caching of access privilege information, when reused without being updated dynamically can lead to unauthorized access to resources
6	Separation of Duty	Every agent in a system has only a single purpose	Augmenting the functionality of an entity with unrelated features or bug fixes can lead to creation of backdoors
7	Least Common	Access mechanisms should not be shared	Every entity should have their own custom access permissions to avoid unauthorized access
8	Secure Weak Link	Protect the weakest link in the chain	A decryption key stored in unprotected memory can lead to loss of data confidentiality and availability
9	Defense in Depth	Build multiple layers/ walls of security	If a system has only a single layer of defense mechanism, and if it is bypassed, it can lead to system compromise
10	Simplicity	Keep it simple	Complexity can lead to vulnerabilities

Index

Bibliography