International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(12s) 2026 pp. 151-158
https://doi.org/10.70917/ijcisim-2026-3887
Article

Compositional Design Principles for Hardware-Backed Platform Security

Madhav Narayan Bhat

Senior Security Customer Engineer, Qualcomma

Abstract: Modern computing devices face adversaries that operate below the operating system, inside firmware, and increasingly across the manufacturing supply chain. Hardware-backed platform security—a discipline that anchors trust in immutable silicon primitives rather than mutable software—has become the default defense for laptops, smartphones, AI PCs, and edge devices. This article develops seven compositional design principles for platform security architects and OEM integration engineers, bringing up hardware roots of trust (HRoT), ARM TrustZone-based trusted execution environments (TEEs), TPM 2.0, and integrated security processors such as Microsoft Pluton. The article draws on standards from the Trusted Computing Group, NIST, GlobalPlatform, and ARM, as well as a decade of academic research on TEE vulnerabilities, including CLKSCREW, BOOMERANG, and ARMageddon, to provide concrete guidance on anchoring trust in fuses; designing verified and measured boot chains with rollback protection; minimizing the trusted computing base; planning debug security; shifting validation left; managing hardware-bound identity; hardening factory provisioning; and planning crypto-agile migrations to post-quantum algorithms. The central argument is compositional: each principle closes a gap that the previous one opens, and no single primitive—not a TPM, a TEE, nor a well-designed Pluton—is sufficient without the others.

Keywords: hardware root of trust; trusted execution environment; TrustZone; TPM; Microsoft Pluton; secure boot; platform firmware resiliency; post-quantum cryptography

1. Introduction

Firmware and supply-chain threats are no longer fringe concerns. A 2021 Microsoft-commissioned survey of 1,000 enterprise security decision-makers found that more than 80% of enterprises had experienced at least one firmware attack in the preceding two years, while only 29% of security budgets were allocated to firmware protection [1]. NIST's National Vulnerability Database recorded a five-fold increase in firmware-class vulnerabilities across the prior four years [1]. Since then, the landscape has continued to change: ESET's disclosure of LoJax—the first in-the-wild UEFI rootkit, attributed to Sednit/APT28—demonstrated that SPI-flash implants can survive operating-system reinstalls and disk replacements [2]. MoonBounce (2022) and CosmicStrand (2022) followed, burrowing into the DXE phase of vendor firmware [3], and BlackLotus (2023) became the first public UEFI bootkit to bypass UEFI Secure Boot on fully patched Windows 11 by exploiting CVE-2022-21894 to roll back to signed but vulnerable boot managers [4]. The SolarWinds campaign, meanwhile, delivered trojanized software updates to as many as 18,000 downstream customers, crystallizing supply-chain risk at the board level [5].

Hardware-backed platform security is the industry's structural response. A hardware root of trust (HRoT) is, per GlobalPlatform, "a set of unconditionally trusted functions" implemented in a computing engine that "must work properly no matter what software is executing on the platform, in order to be immune to software attacks" [6]. HRoTs underpin secure boot chains, seal cryptographic keys to platform state, attest platform integrity to remote verifiers and enforce the debug lifecycle policy through one-time-programmable (OTP) fuses. They are implemented today via discrete TPM 2.0 chips [7], CPU-integrated designs such as Microsoft Pluton [8], and ARM TrustZone-based TEEs embedded across nearly every modern mobile and laptop SoC [9].

The regulatory environment is converging with the threat landscape. The EU Cyber Resilience Act (Regulation 2024/2847) entered into force on 10 December 2024 and becomes fully applicable on 11 December 2027, mandating security-by-design, vulnerability handling, and software bills of materials across every product with digital elements

sold in the EU [10]. FIPS 140-2 validations move to the Historical List on 22 September 2026, forcing a market-wide transition to FIPS 140-3 [11]. NIST published the first three post-quantum standards—FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA)—in August 2024 [12], and NSA's CNSA 2.0 timeline requires exclusive use of quantum-resistant algorithms for firmware signing by 2030 and for most other categories by 2033 [13].

The following seven principles synthesize what works in production, what repeatedly fails in the field, and how to design systems that hold up against both an advanced adversary and a forthcoming regulatory audit.

2. Principle 1—Anchor Trust in Hardware, Not in Software You Can Rewrite

The single most consequential design decision in a platform is what the rest of the system can trust. If software can modify the first code that runs, the root of trust has already moved. NIST SP 800-193 codifies this decision in three principles that every platform firmware designer should internalize: Protection, Detection, and Recovery [14]. The Protection principle requires that firmware code and critical data "remain in a state of integrity and are protected from corruption," typically via a Root of Trust for Update (RTU) that verifies digital signatures before any write to a protected region [14], [15]. Concretely, this means the first stage of the boot chain—the BL1 ROM on ARM platforms [16], the boot-ROM patch layer on a Qualcomm SPU, or the Pluton immutable firmware—must live in mask ROM or in fuse-locked flash, and the root public key hash must be committed to OTP fuses before the device leaves the factory. The TCG Roots of Trust Specification draws a sharp distinction between immutable and mutable roots: a mutable RoT is only as trustworthy as its RTU, and the RTU's verification key must itself be immutable [17]. Any architecture that allows software to rewrite the key used to verify other software has no root of trust at all.

Storing the root public key hash rather than the raw public key in fuses minimizes fuse-bit consumption and decouples algorithm choice from the anchor itself—leaving room to rotate to ML-DSA public keys without reburning the anchor (see Principle 7). On cost-sensitive MCUs that cannot absorb a TPM, the TCG Device Identifier Composition Engine (DICE) pattern provides a complementary primitive: DICE derives a Compound Device Identifier (CDI) from a hardware-unique Unique Device Secret (UDS) and the measurement of the first mutable code, then locks the UDS behind a latch that only the ROM can access [18]. The cryptographic strength this yields at near-zero silicon cost makes DICE the default recommendation for the constrained-device tier of any platform security architecture.

3. Principle 2—Design Secure Boot as a Verified and Measured Chain with Rollback Protection

Secure boot is not a single signature check; it is a chain of checks in which every stage authenticates the next before transferring control and independently records what it loaded in addition to verifying it. These two properties—verified boot and measured boot—are complementary, not redundant. Verified boot halts on a signature mismatch, enforcing policy locally. Measured boot extends cryptographic hashes of every loaded component into the TPM Platform Configuration Registers (PCRs) so that a remote verifier can later attest to the platform state [19]. ARM Trusted Firmware-A's Trusted Board Boot (TBBR) reference design implements precisely this chain: BL1 in ROM verifies BL2 against the Root of Trust The Public Key hash is committed to OTP registers, BL2 verifies the EL3 runtime and secure-world payloads, and measurements propagate up to a measured-boot subsystem [16], [20].

The chain is, however, only as strong as its weakest link, and the most frequent weak link is rollback protection. BlackLotus did not break Microsoft's signing infrastructure; it simply reinstalled a legitimately signed but vulnerable Windows boot manager whose hash had not yet been added to the UEFI forbidden-signature database (DBX) [4]. The Downgrade Attack on TrustZone research by Chen et al. showed the same class of failure against trusted-OS images on Samsung Galaxy S7 and several Nexus devices: when verification uses a version-independant key and no monotonic counter, an attacker with OTA access simply re-flashes last year's vulnerable trustlet and exploits a patched bug [21]. NIST SP 800-147 explicitly requires that authenticated update mechanisms "prevent the unauthorized rollback of the BIOS" [15], and the 2024 TPM 2.0 Library revision introduced Firmware-Limited Objects that bind TPM-resident keys to a specific firmware security version number (SVN) so that a downgrade invalidates the keys it would have unsealed [7], [22]. A secure-boot design without a monotonic anti-rollback counter is one that has not yet failed publicly.

Completing the chain requires aligning measurements with the PSA Attestation Token (RFC 9783) or a DICE-based equivalent so that data are not merely stored—they are reportable. An attestation claim a verifier cannot read does not protect anything.

4. Principle 3—Keep the Trusted Computing Base Small

The trusted computing base (TCB) is the set of code whose compromise would compromise the security properties you care about. In a TEE, the TCB includes the secure monitor, the trusted OS, every trusted application (TA), and every shared library they call. Pinto and Santos's comprehensive ACM Computing Surveys analysis of Arm TrustZone showed that commercial TEEs routinely accrue trusted OSes of tens to hundreds of thousands of lines, with TAs that pull in legacy image parsers, DRM libraries, and vendor-specific fingerprint stacks [9]. Cerdeira et al.'s Systematization-of-Knowledge paper at IEEE S&P 2020 cataloged vulnerabilities across Qualcomm QSEE, Trustonic Kinibi, Huawei, Nvidia, and Linaro OP-TEE and concluded that "TEE systems expose a wide attack surface," with kernel-mode drivers running in the secure world and wide inter-world interfaces [23].

Fig. 1 illustrates the breadth of a computing base of the various parts of the stack in a TEE. The empirical cost of a bloated TCB is measurable. BOOMERANG demonstrated that semantic-gap bugs in TA interfaces on QSEE, Huawei, Sierraware, and OP-TEE let unprivileged applications read and write arbitrary non-secure memory through the TA, affecting roughly 60% of Android phones at the time [24]. Ryan's Hardware-Backed Heist recovered a full 256-bit ECDSA private key from Qualcomm's hardware-backed Android Keystore via microarchitectural side channels introduced by normal TA operation [25]. BootStomp found seven vulnerabilities across HiSilicon, NVIDIA Tegra, and Qualcomm LK bootloaders that collapsed the chain of trust [26]. Each of these incidents shares a common precondition: a TCB large enough to contain a component its designers did not fully audit.

Partitioning aggressively along the GlobalPlatform TEE System Architecture boundaries [27] limit the damage any single TA compromise can cause: Trusted Applications communicate with the Rich Execution Environment through minimal, well-typed, length-checked interfaces defined by the TEE Internal Core API [28], and that boundary is where parser-bug classes—the dominant TA vulnerability category in Cerdeira et al.'s SoK [23]—get quarantined. Heavyweight parsers, protocol stacks, and UX code belong in the normal world; only the security-critical primitive (key derivation, attestation, and sealing) should cross into the TEE. This delegation discipline keeps the TCB auditable. Verified baselines provide the calibration point: seL4's formal proof covers 8,700 lines of C and 600 lines of assembly [29], and Komodo verifies an SGX-style enclave monitor on ARMv7 TrustZone in roughly 12 secure monitor calls [30]. McCune et al.'s TrustVisor fits its TCB into 5,306 lines of code [31]—a number most commercial trusted OSes exceed by two orders of magnitude. Reading these systems does not produce a verified TEE in 2026, but it calibrates what "auditable" actually means in practice, and that calibration is itself a design instrument.

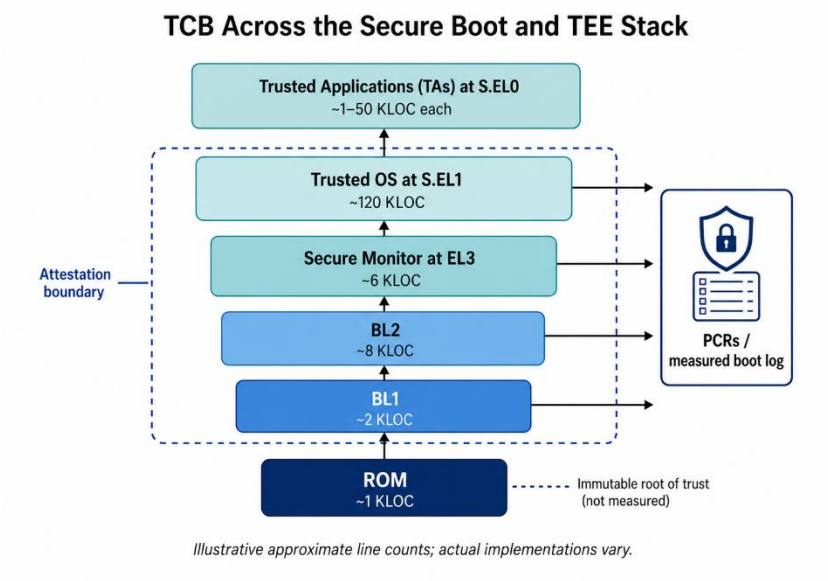

Figure 1. TCB growth and attestation boundary across the secure boot and TEE stack.

5. Principle 4—Plan Debug Security from Day One, Not Day Three-Hundred

Debug interfaces are the permanent reason you can ship silicon and the permanent reason attackers try. Ning and Zhang's Nailgun attack at IEEE S&P 2019 exploited ARM's inter-processor debugging model—specifically the DBGEN, NIDEN, SPIDEN, and SPNIDEN authentication signals—to run arbitrary payloads at EL3 on devices like

the Huawei Mate 7, from which the authors extracted an AES key and a fingerprint image held in the TEE [32]. The attack worked not because ARM's debug architecture is broken, but because the signals were configured permissively on production parts.

A production-grade debug policy begins with fuse-controlled policy registers whose state transitions are one-way: development to secure-development to production. The NXP RW61x debug-authentication application note illustrates this pattern with per-CPU control of DBGEN/NIDEN/SPIDEN/SPNIDEN via OTP fusewords [33], and the invariant is generalizable: no debug authentication signal should be controllable by runtime software on a production device. Alongside these measures, device-specific, cryptographically signed debug tokens replace the single golden key that historically protected an entire device family. Each token binds to a unique device identifier, an expiration, and a scope—normal world only or secure world with key store disabled—so that a compromised token degrades to a bounded incident rather than a platform-class breach. Arm's PSA secure-debug test suite and GlobalPlatform's TEE Management Framework both define interfaces compatible with this model [34], [35].

The debug-policy hash must be committed to immutable storage so that a compromised trusted OS cannot widen scope at runtime—an invariant that field experience has shown is routinely skipped until the third iteration of a bring-up program. Rehearsing the failure mode belongs in the same early planning phase: a production device that accidentally ships with SPIDEN asserted is replaceable only via physical replacement. Designing the token infrastructure before the first engineering board powers on costs days; field-debugging a locked production device without that infrastructure costs weeks.

6. Principle 5—Shift Security Validation Left, and Automate It at Factory Scale

Shift-left validation in platform security means testing security invariants as soon as each layer exists—ideally in simulation, then on pre-silicon emulators, then on engineering-sample boards—rather than discovering misconfigurations in the first factory-floor lot of a hundred thousand units. Intel's open-source CHIPSEC framework, first released at CanSecWest 2014, provides a concrete baseline: modules for SPI-flash lockdown, BIOS write protection, SMRAM integrity, and UEFI variable protection that can be run from Windows, Linux, or a UEFI shell [36]. A resilient platform separates itself from LoJax's target set by detecting a single missing BIOSWE/BLE/SMM_BWP configuration bit with CHIPSEC [2].

Factory-scale automation extends the same principle. Every security-critical fuse state, every chain-of-trust public-key hash, every debug-policy register, and every attestation endpoint should have a corresponding automated test that runs on the final test station and fails the unit loudly rather than shipping a silent mis-fused part. NIST SP 800-193's Detection principle reinforces this [14], as does the TCG PC Client Platform Firmware Profile's event-log schema, which gives validation code a canonical format to diff against golden measurements [37]. A useful architectural pattern is to treat the attestation claim itself as the test oracle: if the factory verifier cannot produce a passing PSA Attestation Token against a known good policy [38], the unit fails. This collapses "security tested" and "security attestable" into a single state—the invariant that the EU CRA's vulnerability-handling obligations will require manufacturers to defend at audit [10].

Shift-left also catches economic failures before they become fleet-scale problems. TEEzz demonstrated that fuzzing Trusted Applications on commercial Android devices uncovers exploitable bugs in shipping TAs [39]; running such fuzzers pre-silicon, on emulated trusted OSes, is orders of magnitude cheaper than patching them through an OTA fleet deployment. The cost of a security bug discovered at factory testing is bounded by the yield loss on that lot. The cost of the same bug discovered post-shipment is limited by the size of the deployment.

7. Principle 6—Treat Device Identity and Key Management as First-Class Citizens

Device identity is the substrate on which licensing, attestation, and zero-trust network access all rest. The industry has converged on two interoperable standards. IEEE 802.1AR-2018 DevID defines an Initial Device Identifier (IDevID) that is globally unique, manufacturer-provisioned, and cryptographically bound to the device, along with a Locally Significant Device Identifier (LDevID) that administrators can provision after deployment [40]. The 2018 revision added ECDSA P-384/SHA-384 signature support, and an in-progress amendment (P802.1AReg) is adding ML-DSA post-quantum signature suites. TCG DICE provides the complementary low-cost primitive for IoT and MCU-class devices [18].

Keys must be hardware-bound—generated inside the TPM, Pluton, or TEE and non-exportable, with the endorsement key (EK) anchored by the manufacturer's certificate chain. Attestation should be mutual—device-to-service and service-to-device—with mTLS terminated against a hardware-held key and a verifier that validates both

the certificate chain and a fresh attestation token. Offline licensing scenarios impose an additional requirement: deriving a short, stable device fingerprint from immutable fuse fields and an HRoT-held secret produces a value that is unforgeable yet verifiable from a signed license blob, enabling air-gapped activation without surrendering the hardware-binding guarantee.

Key hierarchy design is where production deployments most frequently fail. Architectures that allow a single HSM loss to compromise an entire fleet are common; the corrective structure—per-device keys derived via KDFs from per-batch master secrets that are HSM-sealed, so that any HSM breach invalidates one batch rather than a generation—is straightforward in principle yet rarely implemented without a prior incident to motivate it. EK certificate management warrants equal rigor. A missing or unattested EK cert is the earliest visible symptom of a counterfeit device entering the distribution channel, yet EK certs are routinely treated as software configuration files rather than supply-chain attestation artifacts. Correcting this requires logging EK cert provenance at the factory test station and verifying certificate chain integrity before any unit ships.

8. Principle 7—Harden the Factory and Plan the Post-Quantum Migration Simultaneously

The factory is where trust is either born or betrayed. Guidance from AWS and Azure converges on a small number of principles: generate root and batch keys inside a FIPS 140-2/140-3 Level 3 HSM, never expose plaintext key material to the factory floor, and deliver per-devic℃e key material as encrypted, signed blobs that only the device's HRoT can unwrap [41], [42]. NVIDIA's Factory Secure Key Provisioning guidance adds that fuse-burn order matters—burning a SecurityMode fuse before its dependent SecureBootKey fuses will brick a device[43]. NCC Group's heavy-truck provisioning study makes the cultural dimension explicit: factory employees are often unaware of product security requirements, and the factory must therefore be treated as untrusted by design [44]. The control is architectural, not procedural: pre-provisioned HSMs shipped to the contract manufacturer, remote provisioning appliances that hold only ephemeral data, and factory test stations that cannot extract keys even under full compromise.

NIST's FIPS 203, 204, and 205, published 13 August 2024, standardize ML-KEM, ML-DSA, and SLH-DSA respectively [12], and NSA CNSA 2.0 requires the exclusive use of quantum-resistant signatures for firmware signing by 2030 [13]. Firmware signing should migrate to LMS or XMSS stateful hash-based signatures (per NIST SP 800-208) for long-lived code-signing operations today and to ML-DSA-87 for general-purpose signing as ecosystem support matures. Device identity anchors require crypto-agility by design: storing a hash of the root public key rather than a raw ECDSA point in fuses and allocating spare fuse regions for a second, post-quantum root key costs nothing at manufacturing time but avoids a costly architectural retrofit at upgrade time. SLH-DSA provides the conservative fallback for signatures whose keys must survive decades of cryptanalysis. Devices fielded in 2026 with a 10-year service life will face CNSA 2.0 exclusive-use deadline before they are decommissioned; crypto-agility is not a future consideration—it is a present manufacturing requirement.

Table 1 presents the recommended migration path for each major platform primitive, with current algorithms, target migration paths, and NSA CNSA 2.0 deadlines.

Platform Primitive	Current Algorithm	Migration Path	CNSA 2.0 Deadline
Firmware signing	ECDSA P-256/P-384	LMS/XMSS (near-term); ML-DSA-87 (long-term)	Exclusive use by 2030
Key encapsulation	RSA-2048 / ECDH	ML-KEM-768/1024	Exclusive use by 2030
Device identity (EK)	RSA-2048 / ECDSA	ML-DSA-65 (per P802.1AReg)	2033
Boot chain measurements	SHA-256	SHA-384 / SHA-512 (already quantum-resistant)	Already compliant
Attestation tokens	ECDSA P-256	ML-DSA-44 / SLH-DSA-128f	2030-2033
Long-lived code signing	ECDSA	LMS (per NIST SP 800-208)	2028

Table 1. Post-Quantum Migration Matrix for Platform Primitives. Migration paths per NIST FIPS 203/204/205 [12] and NSA CNSA 2.0 [13].

9. Broader Implications for Industry, Regulators, and Users

The seven principles above are not merely engineering hygiene; they are the technical substrate of a rapidly formalizing regulatory regime—one that ties the design decisions across each principle directly to legal and commercial liability. The EU Cyber Resilience Act will hold manufacturers accountable for the entire lifecycle of products with digital elements by late 2027, with vulnerability-handling and incident-reporting obligations beginning in 2026 [10]. The EUCC certification scheme, adopted in January 2024 and applicable from February 2025, operationalizes ISO/IEC 15408 Common Criteria at "substantial" and "high" assurance levels for the European market [45]. FIPS 140-3 validations are becoming the de facto floor for any cryptographic module sold into U.S. federal or CMMC-regulated supply chains after September 2026 [11]. Platform security architects who have already internalized NIST SP 800-193's Protection/Detection/Recovery trio, GlobalPlatform's TEE Protection Profile [46], and the TCG DICE pattern will find that compliance artifacts are largely a documentation exercise. Those who have not will face a redesign.

The societal stakes grow with each shift of computation to the edge. Copilot+ PCs, which ship Pluton by default [8], concentrate increasingly sensitive user data—recorded activity, local large-language-model context, and biometric templates—behind hardware-rooted key hierarchies. ENISA's 2024 threat landscape catalogued 11,079 cybersecurity incidents and 19,754 vulnerabilities in a single year, with supply-chain attacks among the seven prime threats [47]. IBM's Cost of a Data Breach reports place the 2024 global average at USD 4.88 million and flag supply-chain compromise as the second-most-common and slowest-to-resolve initial attack vector, averaging 267 days to contain [48], [49]. Hardware-backed platform security is the layer at which these costs are avoided rather than merely insured against, and the compositional design approach argued throughout this article is the mechanism by which that avoidance is made systematic rather than incidental.

10. Limitations

The analysis presented here draws on publicly reported field incidents, peer-reviewed academic research, and normative standards documents. Quantitative benchmarks—including TCB size figures, fuse-burn operational parameters, and attestation token overhead—are drawn from published literature and should be validated against the specific SoC, manufacturing environment, and deployment context before prescriptive application. Post-quantum migration timelines reflect NSA CNSA 2.0 guidance as of mid-2025 and remain subject to cryptanalytic developments and NIST guidance updates. The seven compositional principles are grounded in production practice across consumer, enterprise, and OEM platforms; they are not a formally complete threat model for any specific platform configuration, and systematic threat modeling against a defined adversary profile remains an essential complement to any application of this guidance.

11. Conclusion

The central argument of this article is compositional rather than additive. A hardware root of trust anchored in fuses gains nothing if the boot chain that follows it permits rollback. A well-implemented TEE is undermined by a bloated trusted computing base that widens the very attack surface the TEE was designed to contain. Both are irrelevant if the factory that provisioned the device treated key material as a software configuration artifact rather than a physical security control, or if the debug policy left SPIDEN asserted on production parts. The seven design principles examined here hold together because each one closes a gap that the previous one leaves open—and because an adversary who finds any one gap open does not need the others.

That compositional logic carries a regulatory corollary. The EU Cyber Resilience Act, FIPS 140-3, and NSA CNSA 2.0 are not independent compliance requirements—they are different projections of the same underlying demand: that manufacturers be accountable for the full security lifecycle of every product with digital elements, at every layer of its stack. Platform security architects who have internalized the TCG's immutability requirements, GlobalPlatform's partitioning discipline, and NIST SP 800-193's Protection, Detection, and Recovery triad will find that compliance audit artifacts are largely a documentation exercise. Those who have not will face a redesign, and post-quantum timelines do not leave room for one.

The threat landscape has already moved below the operating system. LoJax, MoonBounce, BlackLotus, and BOOMERANG are not aberrations—they are the visible frontier of a sustained campaign by adversaries who understand that software defenses built on mutable firmware are defenses built on sand. The engineering response,

codified across two decades of academic research and pressure-tested in production bring-ups across OEM, consumer, and industrial platforms, is to move the anchor below the software: into fuses, into ROM, and into the silicon itself. That is what hardware-backed platform security means in production. The regulatory and economic forces are, unusually, aligned with that engineering imperative. The opportunity to design this correctly—at scale, before the CNSA 2.0 deadlines close—will not persist indefinitely.

References

1. Microsoft Security, "New Security Signals study shows firmware attacks on the rise," Microsoft Security Blog, Mar. 30, 2021.
2. J.-I. Boutin et al., "LoJax: First UEFI Rootkit Found in the Wild, Courtesy of the Sednit Group," ESET Research White Paper, Sept. 2018.
3. M. Lechtik, I. Kwiatkowski, and M. Galliard, "MoonBounce: the dark side of UEFI firmware," Securelist (Kaspersky), Jan. 20, 2022; and M. Lechtik et al., "CosmicStrand: the discovery of a sophisticated UEFI firmware rootkit," Securelist, Jul. 25, 2022.
4. M. Smolar, 'BlackLotus UEFI Bootkit: Myth Confirmed,' ESET WeLiveSecurity, Mar. 1, 2023; National Security Agency, 'BlackLotus Mitigation Guide,' U/OO/167397-23, Jun. 2023.
5. U.S. Government Accountability Office, "SolarWinds Cyberattack Demands Significant Federal and Private-Sector Response," GAO, 2021.
6. GlobalPlatform, "Root of Trust Definitions and Requirements," v1.1, Jun. 2018.
7. Trusted Computing Group, "Trusted Platform Module Library Specification, Family 2.0," Revision 01.83 (ISO/IEC 11889:2015), TCG, 2024.
8. D. Weston, "Meet the Microsoft Pluton processor," Microsoft Security Blog, Nov. 17, 2020; Microsoft Corporation, "Microsoft Pluton security processor," Microsoft Learn, 2024.
9. S. Pinto and N. Santos, "Demystifying Arm TrustZone: A Comprehensive Survey," ACM Computing Surveys, vol. 51, no. 6, Art. 130, pp. 1-36, Jan. 2019, doi: 10.1145/3291047.
10. European Union, "Regulation (EU) 2024/2847 ... (Cyber Resilience Act)," Official Journal of the EU, Nov. 20, 2024.
11. NIST, "Security Requirements for Cryptographic Modules," NIST FIPS 140-3, Mar. 22, 2019; NIST CSRC, "FIPS 140-3 Transition Effort," updated 2025.
12. NIST, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, Aug. 13, 2024; "Module-Lattice-Based Digital Signature Standard," FIPS 204, Aug. 13, 2024; "Stateless Hash-Based Digital Signature Standard," FIPS 205, Aug. 13, 2024.
13. National Security Agency, "Announcing the Commercial National Security Algorithm Suite 2.0," Cybersecurity Advisory, Sep. 2022; rev. 2024.
14. A. Regenscheid, "Platform Firmware Resiliency Guidelines," NIST SP 800-193, May 2018, doi: 10.6028/NIST.SP.800-193.
15. D. Cooper et al., "BIOS Protection Guidelines," NIST SP 800-147, Apr. 2011, doi: 10.6028/NIST.SP.800-147.
16. Arm Limited, "Trusted Board Boot Requirements Client (TBBR-Client)," Arm DEN0006D; Trusted Firmware-A reference implementation.
17. Trusted Computing Group, "TCG Roots of Trust Specification," Family 1.0, Level 00, Revision 0.20, Public Review.
18. Trusted Computing Group, "Hardware Requirements for a Device Identifier Composition Engine (DICE)," Family 2.0, Level 00, Rev. 78, 2018.
19. A. Regenscheid and K. Scarfone, "BIOS Integrity Measurement Guidelines (Draft)," NIST SP 800-155 (Draft), Dec. 2011; TCG, "TCG PC Client Platform Firmware Profile Specification," v1.06, Rev. 52, Dec. 4, 2023.
20. Arm Limited, "ARM Security Technology: Building a Secure System using TrustZone Technology," White Paper PRD29-GENC-009492C, Apr. 2009.
21. Y. Chen, Y. Zhang, Z. Wang, and T. Wei, "Downgrade Attack on TrustZone," arXiv:1707.05082, 2017.
22. Trusted Computing Group, "TPM 2.0 Library Specification, Part 1: Architecture," Family 2.0, Revision 01.83, TCG, 2024. [Incorporates Firmware-Limited Objects binding TPM-resident keys to firmware security-version numbers (SVN).]
23. D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the Prevailing Security Vulnerabilities in TrustZone-assisted TEE Systems," in Proc. 2020 IEEE S&P, May 2020, pp. 1416-1432, doi: 10.1109/SP40000.2020.00061.
24. A. Machiry et al., "BOOMERANG: Exploiting the Semantic Gap in Trusted Execution Environments," in Proc. NDSS, Feb. 2017.
25. K. Ryan, "Hardware-Backed Heist: Extracting ECDSA Keys from Qualcomm's TrustZone," in Proc. 2019 ACM CCS, Nov. 2019, pp. 181-194, doi: 10.1145/3319535.3354197.
26. N. Redini et al., "BootStomp: On the Security of Bootloaders in Mobile Devices," in Proc. 26th USENIX Security Symp., Aug. 2017, pp. 781-798.
27. GlobalPlatform, "TEE System Architecture," v1.3, Doc GPD_SPE_009, May 2022.
28. GlobalPlatform, "TEE Internal Core API Specification," v1.3.1, Doc GPD_SPE_010, Feb. 2021.

29. G. Klein et al., "seL4: Formal Verification of an OS Kernel," in Proc. 22nd ACM SOSP, Oct. 2009, pp. 207-220, doi: 10.1145/1629575.1629596.
30. A. Ferraiuolo et al., "Komodo: Using Verification to Disentangle Secure-Enclave Hardware from Software," in Proc. 26th ACM SOSP, Oct. 2017, pp. 287-305, doi: 10.1145/3132747.3132782.
31. J. M. McCune et al., "TrustVisor: Efficient TCB Reduction and Attestation," in Proc. 2010 IEEE S&P, May 2010, pp. 143-158, doi: 10.1109/SP.2010.17.
32. Z. Ning and F. Zhang, "Understanding the Security of ARM Debugging Features," in Proc. 2019 IEEE S&P, May 2019, pp. 602-619, doi: 10.1109/SP.2019.00061.
33. NXP Semiconductors, "AN13814 - Debug Authentication on RW61x," Application Note, Rev. 4.0, Nov. 2024.
34. Arm Limited, "Platform Security Model 1.1," JSADEN014, 2021; PSA Certified scheme documentation.
35. GlobalPlatform, "TEE Management Framework (TMF) including ASN.1 Profile," v1.1.2, Nov. 2020.
36. Intel Corporation, "CHIPSEC: Platform Security Assessment Framework," 1st release CanSecWest 2014.
37. Trusted Computing Group, "TCG PC Client Platform TPM Profile Specification for TPM 2.0," v1.06, Rev. 32, Apr. 2024.
38. H. Tschofenig et al., "Arm Platform Security Architecture (PSA) Attestation Token," RFC 9783, IETF, Jun. 2025.
39. M. Busch et al., "TEEzz: Fuzzing Trusted Applications on COTS Android Devices," in Proc. 2023 IEEE S&P, pp. 1204-1219, doi: 10.1109/SP46215.2023.00070.
40. IEEE, "IEEE Standard for Local and Metropolitan Area Networks - Secure Device Identity," IEEE Std 802.1AR-2018, Aug. 2, 2018, doi: 10.1109/IEEESTD.2018.8423794.
41. Amazon Web Services, "Device Manufacturing and Provisioning with X.509 Certificates in AWS IoT Core," AWS Whitepaper.
42. Microsoft, "Security practices for manufacturers - Azure IoT Device Provisioning Service," Microsoft Learn.
43. NVIDIA Corporation, "Factory Secure Key and Expansion Key Provisioning," NVIDIA Jetson Linux Developer Guide, r36.4.
44. NCC Group, "Secure Device Provisioning Best Practices: Heavy Truck Edition," NMFTA, 2020.
45. European Commission, "Commission Implementing Regulation (EU) 2024/482 of 31 January 2024 (EUCC scheme)," Official Journal of the EU, Feb. 7, 2024.
46. GlobalPlatform, "TEE Protection Profile," v1.3, Dec. 2016.
47. European Union Agency for Cybersecurity (ENISA), "ENISA Threat Landscape 2024," Sept. 2024.
48. IBM Security and Ponemon Institute, "Cost of a Data Breach Report 2024," IBM Corp., Jul. 2024.
49. IBM Security and Ponemon Institute, "Cost of a Data Breach Report 2025," IBM Corp., Jul. 2025.