BOOT SECURITY MODES AND RECOMMENDATIONS

Modern computing platforms provide a variety of boot options. The security implications, advantages, and disadvantages are rarely identified in documentation. Some configuration options, such as Secure Boot and Trusted Platform Module (TPM)[1] may appear redundant despite serving complementary roles. Six different configurations are compared below. Recommendations for different use cases are presented at the end of this document.

1. LEGACY BIOS CSM COMPATIBILITY MODE

Many computing devices feature a legacy boot mode called Basic Input/Output System (BIOS) mode or Compatibility Support Module (CSM). Legacy boot modes are intended to be used with older peripherals, hardware, and software lacking support for Unified Extensible Firmware Interface (UEFI)[2] standards.

Advantages: Compatible with decades of solutions and protocols.

Disadvantages: Least restrictive solution covered in this fact sheet in terms of security features. Mainstream system and processor vendors are discontinuing legacy mode support starting in 2020. Some newer hardware components lack support for legacy mode or are constrained in performance and features. Patching of legacy firmware requires a make and model-specific solution that increases the opportunities for mistakes and slows delivery of new patches. Validation of boot binaries is weak or unimplemented.

2. UEFI NATIVE MODE

Unified Extensible Firmware Interface (UEFI) adheres to standards defined by the UEFI Forum – an industry consortium. Hardware, software, peripheral, system vendor, and solution providers collaborate to define common interfaces, variables, protocols, feature sets, and structures for use on modern computing platforms. UEFI provides a common, vendor and architecture-neutral foundation that accelerates the development of new products and patches.

Advantages: Binaries are arranged in the form of modules. Modules can be patched, replaced, added, removed, or otherwise altered individually as needed. Some modules can execute in parallel to accelerate boot time. Some modules can fail or enter error states without affecting a device's ability to boot. Some newer protocols, software, hardware, and solutions only support UEFI standards. Multiple vendors can work on patches simultaneously to accelerate delivery of new versions. Environment variables and services allow the firmware boot environment to pass data to the operating system, and vice versa.

Disadvantages: UEFI relies upon Secure Boot or vendor-specific boot protection solutions – no validation or protection of the boot process is granted simply by choosing UEFI over legacy mode. Some older hardware and software do not function in UEFI mode.

3. SECURE BOOT STANDARD MODE

Secure Boot is a signature and hash-checking mechanism added to the UEFI boot process. Each firmware and software executable at boot time must have an associated signature or hash. Secure boot validates signatures using RSA-2048 public key certificates. Hashes are listed in SHA-256 format. Secure Boot has multiple data stores for storing certificates and hashes. Most new computing devices are loaded with system vendor and Microsoft[3] values focused on compatibility with mainstream computing solutions.

Untrusted values are checked first. Any signature or hash that validates against an untrusted value is blocked from execution at boot time. Trusted values are checked second. A signature or hash match validated to a trusted value may execute at boot time. Any binaries that cannot be validated as untrusted or trusted – also known as unknown binaries – are treated as

---

[1] TPM (Trusted Platform Module) is property of the Trusted Computing Group (TCG)
[2] UEFI and UEFI Secure Boot are property of the UEFI Forum
[3] Microsoft, Windows, and BitLocker are registered trademarks of Microsoft Corporation

untrusted and blocked. However, denial of execution privileges does not necessarily stop the boot process.

Some systems feature boot speed adjustments. Systems placed in a “fast boot” or “minimal boot” mode may skip all firmware-related Secure Boot checks. See figure 1 for a comparison. Use the “full boot” or “thorough boot” mode to ensure all firmware binaries are checked. Some systems also feature a legacy/CSM fallback mode. Disable fallback mode to prevent unknown binaries from bypassing Secure Boot checks.

Advantages: System firmware, component firmware, bootloaders, kernels, and other boot-time executables are validated by Secure Boot to provide a boot-time anti-malware solution. Traditional operating system anti-malware solutions have limited effectiveness against boot-time threats. Standard mode relies upon Microsoft and system vendors to validate hardware and software – a solution that minimizes or eliminates overhead for system owners. Secure Boot’s blacklist database (DBX) and whitelist database (DB) can be updated to combat malware, leaks, and counterfeit products through operating system patches. There is mainstream and nearly seamless support from most hardware and software vendors.

Disadvantages: Secure Boot signing authorities may make mistakes in granting signatures or loading hashes. Bootloaders that ignore Secure Boot and boot-time malware have been mistakenly signed and released to the public in the past. Changes to Secure Boot configuration can be made without advanced notice in firmware patches. Custom hardware, firmware, and software may not pass Secure Boot validation without submission to Microsoft or system vendors. Secure Boot signing authority evaluation processes are slow and costly – facts that have driven the Linux community to develop Machine Owner Key (MOK), MOK Blacklist (MOKX), and the Secure Boot Standard Mode-ready bootloader SHIM. Many systems are configured from the factory to support fast boot or minimal boot to skip some Secure Boot checks in favor of decreasing boot time. Switching to full boot or thorough boot enables most or all checks at the cost of slower boot. Secure Boot is not active from the moment of power application meaning an early-boot blind spot exists.

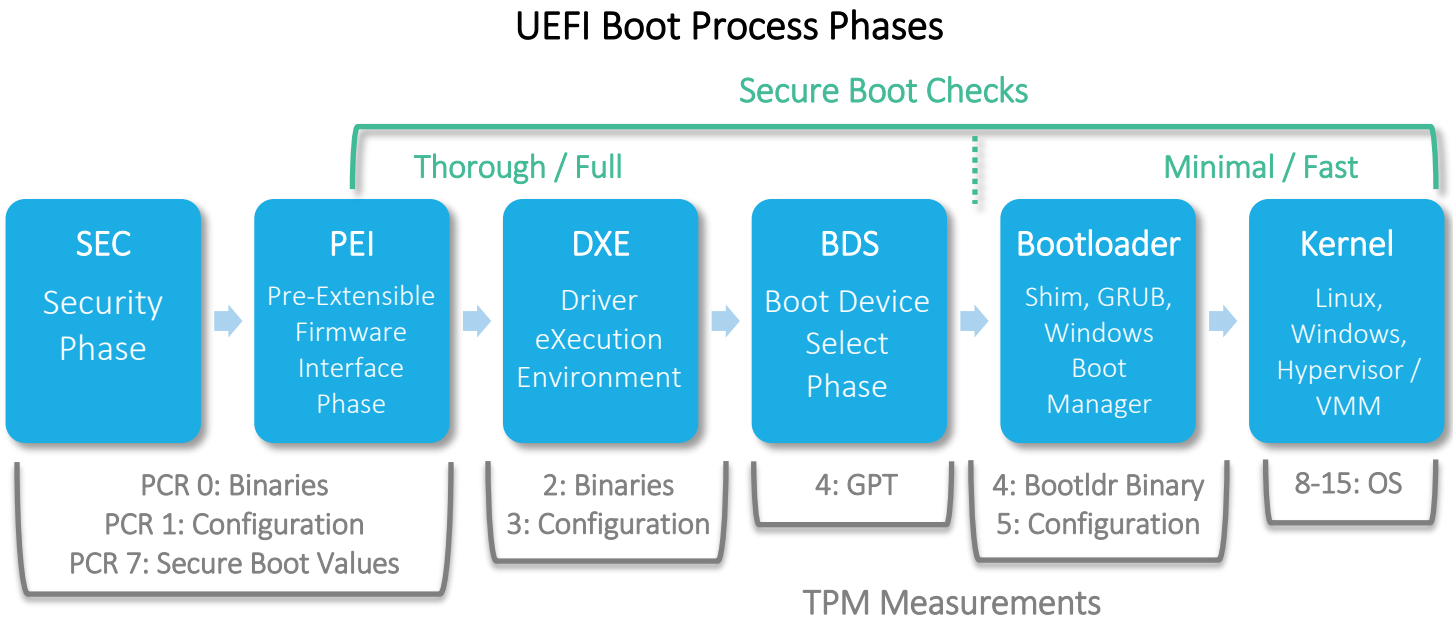

Figure 1 – *The interaction of Secure Boot and TPM with UEFI boot phases is shown. TPM operates as a passive observer of all phases. Secure Boot actively enforces checks depending on configuration scope (full vs minimal).Diagram assumes a Secure Boot-aware bootloader and kernel. Kernel may continue Secure Boot checks (commonly used for driver signing).*

4. SECURE BOOT CUSTOM MODE

Custom mode allows the system owner to exert control over Secure Boot’s data stores. The system owner assumes full

responsibility for signing and hashing trusted content (e.g. storage/RAID controllers, network controllers, bootloaders, kernels) rather than relying upon the ecosystems set up by Microsoft and system vendors. There are four data stores which are checked in the following order:

1. Blacklist Database (DBX) stores certificates and SHA-256 hashes
2. Whitelist Database (DB) stores certificates and SHA-256 hashes
3. Key Exchange Key(s) (KEK) stores one or more certificates
4. Platform Key (PK) stores one certificate

System owners can use custom mode to specify which hardware components, firmware images, boot loaders, kernels, and drivers are acceptable at boot. Untrusted or unknown items are blocked while trusted items are granted boot-time execution privileges. An item blocked at boot time does not necessarily stop the entire boot process.

Advantages: Administrators can trust specific hardware components, boot methods, and boot software. Allows system owners such granular control that they can trust an individual version of an OS kernel, for example. Older, untrusted, or unknown items are blocked from execution at boot time. Can reduce or eliminate the threat of misconfiguration, insider tampering, and boot malware. System owners can react to new boot threats instantly and are insulated from signing mistakes made by vendors. Remote update of DBX, DB, and KEK values are possible via OS patches and scripts.

Disadvantages: Administrative overhead is required to sign binaries and identify the hashes of items that cannot be signed. Systems must be provisioned with a custom Secure Boot. Administrators must create custom Secure Boot update scripts. System and software vendors rarely consider the implications of custom Secure Boot when distributing updates – testing and signing of new boot executables, especially bootloaders and kernels, is important. Just as in Secure Boot Standard Mode, Secure Boot Custom Mode is not active from the moment of power application meaning an early-boot blind spot exists. Custom mode introduces additional management challenges in enterprises utilizing a diverse assortment of machines and vendor solutions.

5. UEFI OR LEGACY BOOT WITH TPM AUDITING

UEFI and Legacy modes can record hashes of firmware components to the Trusted Platform Module (TPM). The TPM must be both activated and enabled for hashes to be written. Hashes normally capture firmware images, firmware configuration, expansion component firmware images, expansion component firmware configurations, and the bootloader. TPM-aware bootloaders can continue logging hashes to describe the kernel, initial file system, and any modules. Kernels, applications, and drivers can log runtime hashes to the TPM too.

Hashes are stored in the TPM's Platform Configuration Registers (PCR). Most TPMs have 24 PCRs per supported hash algorithm. TPM 1.2 supports SHA-1 (24 PCRs). TPM 2.0 supports SHA-1 and SHA-256 at the minimum (48 PCRs minimum). PCR values are computed via a series of one-way hashes where each measurement hash is appended to the current PCR value, then the combination is hashed and becomes the new PCR value. Measurement hashes are recorded in an audit log for verification later.

TPM has no automatic enforcement mechanism like Secure Boot does. To take advantage of TPM auditing, system administrators will need to perform PCR data sealing and unsealing, create integrity reports via the quoting function, or generate keys linked to the TPM's unique identifier post-boot. See figure 1 for the intended scope of each PCR.

Advantages: TPM PCR hash extensions are automated at the firmware level from the earliest stages of boot. Newer versions of Windows and Linux also automatically detect the presence of TPM and begin recording integrity information. Some vendor solutions can use integrity attestation to make a judgement about system health.

Disadvantages: There is no automatic enforcement mechanism (TPM performs passive observation during boot). Malware is not denied execution privileges at boot time. Any valid UEFI, Legacy, or binary is allowed to execute without signature, hash, or integrity checks. A TPM-aware bootloader or boot module, such as Trusted Boot (TBoot) or Trusted GRUB[4] (TGRUB) can be used, at the risk of Denial of Service (DOS) vulnerability should a system deviate from a narrowly-defined

[4] GRUB (GRand Unified Bootloader) is property of the GNU Project.

trusted set of PCRs. TPM boot enforcement solutions are all open source, non-production solutions at the time of publication.

6. UEFI SECURE BOOT WITH TPM AUDITING

Secure Boot can be used in standard mode or custom mode in conjunction with TPM. TPM provides the ability to cover the early-boot blind spot that exists in Secure Boot. Secure Boot allows the flexibility to handle multiple trusted system images, devices, and configurations when necessary (particularly important when updating versions). Configure TPM-aware solutions to monitor PCR 0 firmware images, PCR 1 firmware configuration, and PCR 7 Secure Boot data stores – all system integrity indicators that rarely change. Utilize Secure Boot to validate expansion component firmware, boot loaders, kernels, and drivers – things that are likely to change due to parallel UEFI execution and frequent software updates.

See figure 1 for a breakdown of how Secure Boot and TPM complement each other. Leverage the TPM by applying a TPM-aware bootloader, a TPM-enabled disk encryption product (e.g., Microsoft BitLocker or Linux[5] Unified Key System (LUKS)), or by leveraging TPM PCR sealing and unsealing.

Advantages: Most restrictive option covered in this fact sheet. Windows 10 and newer automatically leverages Secure Boot and TPM when BitLocker is enabled. Secure Boot and TPM complement each other to eliminate blind spots in the auditing of boot processes. Boot integrity is actively enforced. Multiple trusted configurations are accepted at boot time. Excellent protection against zero-day vulnerabilities. Customized Secure Boot can be leveraged.

Disadvantages: Linux distributions require the addition of Trusted Shim[6], Host Integrity at Startup (HIS)[7], Host Integrity at Runtime and Startup (HIRS)[7], and/or Integrity Measurement Architecture (IMA). The most developer and administrator overhead is incurred. TPM hashes must be tracked by administrators. Custom Secure Boot signatures and hashes must also be tracked if in use. Changes to bootloaders, kernels, and firmware may require new hashes and/or signatures.

RECOMMENDATIONS

Do not use Legacy BIOS CSM Compatibility Mode. Migrate away from legacy boot to avoid interruptions in availability of updates as well as increased infrastructure maintenance resulting from vendor deprecation in 2020.

UEFI Secure Boot in Standard Mode with TPM Support is the best balance of protection and overhead for most organizations and most user workstations. Enable the thorough boot or full boot options to ensure comprehensive Secure Boot checks.

UEFI Secure Boot in Custom Mode with TPM Support provides the best protection against threats. To minimize the overhead costs, focus deployments on the most at-risk machines (like traveling laptops and computing devices on the network perimeter). Enable the thorough boot or full boot options to ensure comprehensive Secure Boot checks.

(U) DISCLAIMER OF WARRANTIES AND ENDORSEMENT

(U) The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement, recommendation, or favoring by the United States Government, and this guidance shall not be used for advertising or endorsement purposes.

CONTACT INFORMATION

Cybersecurity Requirements Center
410-854-4200
Cybersecurity_Requests@nsa.gov

[5] Linux is a registered trademark of Linus Torvalds

[6] https://github.com/nsacyber/TrustedSHIM

[7] https://github.com/nsacyber/HIRS