NATIONAL SECURITY AGENCY

CYBERSECURITY REPORT

UEFI DEFENSIVE PRACTICES GUIDANCE

A TECHNICAL REPORT FROM THE VULNERABILITY SOLUTIONS OFFICE

U/OO/217598-17

27 July 2017

Revised 10 October 2018

UEFI DEFENSIVE PRACTICES GUIDANCE

DOCUMENT CHANGE HISTORY

Date	Version	Description
10 October 2018	1.2	New template and formatting applied. Diagrams updated to match template style.
4 October 2018	1.1	Original serial number U/OO/800968-17 replaced by new serial number U/OO/217598-17. Added new NOTICE describing document as having been developed in the course of NSA's cybersecurity mission.
27 July 2017	1.0	Initial publication.

DISCLAIMER OF WARRANTIES AND ENDORSEMENT

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government, and shall not be used for advertising or product endorsement purposes.

NOTICE

The information contained in this document was developed in the course of NSA's cybersecurity mission including its responsibilities to identify and disseminate information on threats to national security systems and Department of Defense information technologies, develop and issue security implementation specifications for cybersecurity-enabled products, and to assist Executive departments and agencies with operational security programs. The information may be shared broadly to reach all appropriate stakeholders.

UEFI DEFENSIVE PRACTICES GUIDANCE

AUTHOR(S)

Vulnerability Solutions Office

Endpoint Security Solutions

CONTACT INFORMATION

Client Requirements and Inquiries or General Cybersecurity Inquiries

CYBERSECURITY REQUIREMENTS CENTER (CRC)

410-854-4200

Cybersecurity_Requests@nsa.gov

EXECUTIVE SUMMARY

Unified Extensible Firmware Interface (UEFI) is a replacement for the legacy Basic Input Output System (BIOS). UEFI comes with a variety of new configuration options, improved performance, extended security measures, and supported platform architectures. New capabilities introduce the opportunity for abuse or infection of malware. Traditional virus scanners are ineffective at cleaning the boot firmware environment necessitating new solutions.

Machines running legacy BIOS or UEFI in compatibility mode should be migrated to UEFI native mode to take advantage of new features. UEFI should be secured using a set of administrator and user passwords appropriate for a device's capabilities and intended use. Firmware comprising UEFI should be updated regularly and treated as importantly as Operating System (OS) updates. UEFI Secure Boot should be enabled and configured to audit firmware modules, expansion devices, and bootable OS images. Trusted Platform Module (TPM) should be leveraged to check the integrity of UEFI.

WHAT IS UEFI?

Unified Extensible Firmware Interface (UEFI) is an **abstraction layer** between the operating system and the underlying platform that provides boot and run-time services for a Personal Computer (PC). This abstraction provides a uniform set of functions, variables, and environment behaviors across a wide variety of devices and in an architecturally independent manner. As implemented, UEFI exists as platform firmware – purpose-built software that lives within physical chips on devices. Vendor-specific chip characteristics and architectures are thus abstracted away by UEFI's environment specification. UEFI offers several advantages over the prior boot mechanism, the Basic Input/Output System (BIOS).

UEFI's consistency allows for **extensibility**. UEFI code modules can be added or removed as needed by vendors, developers, system administrators, and end users depending on the hardware and software deployed to a machine. Extensible design promotes code reuse and may enable parallel execution at the firmware level on multi-core processors thus accelerating the boot process.

Implementations of UEFI may enable a **rich user interface** that can include graphics, help text, tutorials, dynamically updated stats, filesystem access, network connectivity, and more. A set of authenticated interfaces are also available that allow configuration changes from the Operating System (OS) environment post-boot.

Figure 1 provides an overview of the UEFI boot process from power button press to user login. Boot actions performed and corresponding auditing hashes stored in the Trusted Platform Module (TPM) are identified. Note that there is no device owner influence on the Security (SEC) and Pre-EFI (PEI) phases – both are controlled by the firmware vendor.

UEFI Boot Process

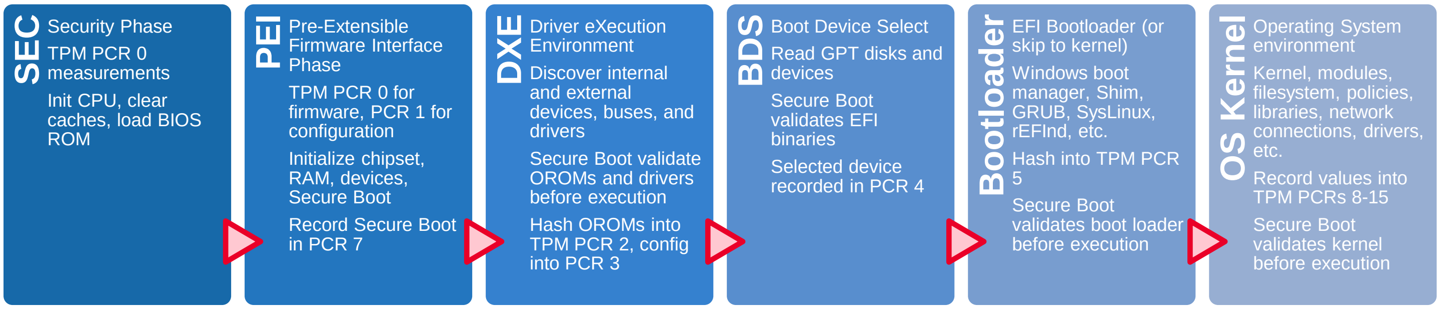
SEC
Security Phase
TPM PCR 0 measurements
Init CPU, clear caches, load BIOS ROM
PEI
Pre-Extensible Firmware Interface Phase
TPM PCR 0 for firmware, PCR 1 for configuration
Initialize chipset, RAM, devices, Secure Boot
Record Secure Boot in PCR 7
DXE
Driver eXecution Environment
Discover internal and external devices, buses, and drivers
Secure Boot validate OROMs and drivers before execution
Hash OROMs into TPM PCR 2, config into PCR 3
BDS
Boot Device Select
Read GPT disks and devices
Secure Boot validates EFI binaries
Selected device recorded in PCR 4
Bootloader
EFI Bootloader (or skip to kernel)
Windows boot manager, Shim, GRUB, SysLinux, rEFInd, etc.
Hash into TPM PCR 5
Secure Boot validates boot loader before execution
OS Kernel
Operating System environment
Kernel, modules, filesystem, policies, libraries, network connections, drivers, etc.
Record values into TPM PCRs 8-15
Secure Boot validates kernel before execution

Figure 1 -- UEFI boot process from power up to user OS environment.

In addition to boot services, UEFI has a few distinguishing security features over legacy BIOS implementations. **UEFI Secure Boot** is a signature checking scheme that validates binaries, such as drivers and boot loaders, prior to execution. Secure Boot features a database of keys and hashes that can be updated by vendors or customized by device owners to provide **boot time anti-malware**. Expanded support for Trusted Platform Module (TPM), longer measurement hashes, and audit log storage also exist to create a boot integrity record that includes Secure Boot's state. Finally, some **vendors offer proprietary boot security solutions** that integrate with UEFI to harden the boot process – no legacy BIOS implementation exists.

MIGRATING FROM BIOS TO UEFI

ADVANTAGES

UEFI has several advantages over legacy BIOS. The most important reasons for transitioning are:

- **Secure Boot** is an optional setting that enforces signature checking of the boot process. Secure Boot ensures only signed binaries are executed. Most implementations allow organizations to exert finely-grained control over what keys and binaries can validate. This also represents malware protection in the pre-boot environment prior to any traditional virus scanning (Wei, 2013).
- **GUID Partition Table (GPT)** replaces the Master Boot Record (MBR) storage drive partitioning scheme. GPT allows the use of storage media boot partitions greater than 2 TB in size. Additional improvements include support for 128 partitions or more and expanded partition labels of up to 36 characters. Disk layout structures are duplicated for redundancy with checksums in case of sector corruption.
- **Platform and architecture independence** allows UEFI to run on x86, x86_64, ARM, ARM64, PowerPC, Itanium, and other architectures. UEFI also supports emulation through hypervisors such as Hyper-V, VMware, Xen, and others.
- The **UEFI environment** is not tied to a specific piece of hardware or vendor implementation. UEFI's uniform environment provides the same services, variables, and other resources regardless of a particular machine make and model. Uniformity enables UEFI to tackle a wide variety of devices and use cases without impacting software built upon the environment.
- **UEFI is modular and extensible**. Firmware vendors, software vendors, end users, and Information Technology (IT) administrators can all add or remove components from system firmware. UEFI is also built to allow parallel execution at different stages of the boot process which speeds the rate of booting.
- **UEFI services are available to the OS** which creates a stronger link between the pre-boot environment and the administrative components of the OS. The results are improved power saving, sleep and resume mechanism consistency, and the ability for the OS to alter or update UEFI information when appropriate.
- **UEFI is an industry standard**. Older technologies such as BIOS, MBR, and other legacy mechanisms are being phased out. Support for older technologies will end.

REQUIREMENTS

Use of UEFI has the following requirements:

- **Hardware and firmware capable of booting in UEFI mode**. Note that pure UEFI is different from BIOS, Legacy, UEFI compatibility, or UEFI Compatibility Support Module (CSM). This requirement specifically refers to the motherboard of a computer.
    - All machines with Windows 8®[1] logos and newer are UEFI-compliant. UEFI was a requirement for the Windows 8 logo compliance. The requirement affected both business and consumer machines. Most machines with Intel Core i®[2] processors, AMD Bulldozer®[3] processors, and newer solutions support UEFI. Machines as old as 2008 may have UEFI support.
- **GPT OS storage drive partitioning or Preboot eXecution Environment (PXE) infrastructure**. Most UEFI implementations do not support booting from an MBR drive outside of legacy mode. The

[1] Microsoft Windows, Windows 7, Windows 8, Windows 8.1, and Windows 10 are registered trademarks of Microsoft Corporation.

[2] Intel, Intel Inside, Core, Core Duo, Core Quad, Core 2, Core i, i3, i5, i7, i9, Xeon, Atom, and Pentium are registered trademarks of Intel Corporation.

[3] AMD, Athlon, Bulldozer, Ryzen, Zen, Epyc, Opteron, and Threadripper are registered trademarks of Advanced Micro Devices, Inc.

MBR portioning format is obsolete, although the scheme is typically supported for non-booting drives. Some infrastructures may substitute storage drives for network-delivered OS images delivered via PXE boot. Although rarely used in the enterprise, booting from removable media is possible if formatted with GPT – like a storage drive.

- **OS compatibility**. Popular options include but are not limited to:
  - Windows 7 and newer
  - Red Hat Enterprise Linux®[4] 5 and newer
  - OS X®[5] 10.5 Leopard and newer
  - Ubuntu®[6] 11.10 and newer
  - Fedora®[7] 15 and newer

MIGRATION

System and infrastructure owners, and organizations who have currently deployed PCs with MBR formatted hard drives and who are still using BIOS based systems, should consider migrating to UEFI. An initial assessment of those systems will be required to understand whether the PC is UEFI capable (i.e., running in a BIOS legacy mode) and a strategy to convert from MBR to GPT formatted drives will be needed. There are multiple options for overcoming this obstacle:

Option 1 – Introduce GPT and UEFI through normal hardware refreshes.

Machines using newer technology can be introduced by infrastructure owners as part of normal hardware purchasing cycles. UEFI and GPT have no impact on the ability of a client to connect to domain servers, file shares, and other OS functions. Many vendors can preconfigure machines in this mode prior to delivery to customers.

Option 2 – Convert an existing installation from MBR to GPT, then enable UEFI.

Conversion is a two-step process. The process is not likely to be destructive, but modifying partitioning schemes does come with risks. User data should be backed up or stored remotely before beginning the conversion process.

The first step is to convert the OS storage drive from MBR to GPT partitioning. Many software packages are available to accomplish this -- some open source, some paid, and some built-into the OS distribution. The following list contains some but not all available options:

- MBR2GPT Windows built-in utility
- Windows Disk Management built-in utility (diskmgmt.msc)
- GPT fdisk Linux built-in utility (https://sourceforge.net/projects/gptfdisk/)
- Gptgen (http://sourceforge.net/projects/gptgen)
- GNOME™[8] Partition Editor (gparted) (http://gparted.org/index.php)

[4] Red Hat Enterprise Linux (RHEL) and Red Hat are registered trademarks of Red Hat, Inc.

[5] OS X, OS X Leopard, OS X Snow Leopard, and Mac OS are registered trademarks of Apple, Inc.

[6] Ubuntu is a registered trademark of Canonical Ltd.

[7] Fedora and Fedora Project are registered trademarks of Red Hat, Inc.

[8] GNOME is a trademark of the GNOME Foundation.

- Vendor-specific automated tools
- Custom command line and scripting tools
- OS boot-time repair and recovery tools

Instructions for the previously listed tools are likely to evolve over time. Therefore, specific instructions are not provided in this paper. See the help documentation on each resource's website or within each program for more details. Some of these programs can perform the conversion from within the MBR OS while others require the use of live media or system recovery mode.

The second step is to use a vendor-provided utility to change system configuration. The software would need to swap firmware configuration from BIOS, compatibility, or legacy mode to UEFI mode. Most large PC vendors have firmware configuration utilities that can make changes to machines without requiring visiting individual endpoints. Configuration changes can be distributed like update patches. The tools are normally vendor-specific and sometimes may be model-specific. The instructions for the use of these tools are likewise beyond the scope of this paper.

Option 3 – Create a new OS installation.

New installations may require installing over top of an existing OS partition. Backing up data is necessary since overwriting partition tables will be destructive. The computer will need to be switched into UEFI mode before performing the installation in GPT mode (GPT and MBR are incompatible). Once the OS storage drive is converted to GPT via the installation process, system administrators may convert other drives to the GPT format using built-in OS tools – this process may be entirely unnecessary if the drive isn't used for booting.

Some older machines may require the disabling of vendor logo overlays, fast booting, legacy option ROMs, and legacy I/O ports to enter UEFI mode.

LOCKING DOWN UEFI

UEFI CONFIGURATION PASSWORD

In general, UEFI implementations have multiple layers (types) of passwords available to limit access to system configuration parameters and devices. These may include **Administrative**, **User**, **System**, and **Storage Drive passwords**. Administrative and User passwords limit access to system configuration or customization of the boot process – they do not interfere with a normal, non-customized boot. System and storage drive passwords halt access to the entire device or a specific drive, respectively, until the password is given – a situation that could hamper updates. Always check the device to be locked down to determine which password options are available – only administrator passwords are required by specification, and password names/descriptions are not consistent across different device vendors.

Administrative Passwords

Administrative UEFI passwords lock out the UEFI configuration. The UEFI Forum **recommends** administrative passwords to stop normal users from altering low-level system parameters during boot time. **If given an option to lock out user changes when an administrator password is present, always enable that option.** A unique administrative password per machine is the strongest implementation of

UEFI administrative password security and the recommended solution. Some older vendor implementations use the administrative password (instead of digital signatures) as a gatekeeper to firmware update – a situation that could hinder firmware updates and system maintenance. When passwords hinder UEFI updates and maintenance, consider using a common administrator password rather than no password at all. Password complexity and the distribution of passwords may be determined by IT organizational policy.

Note that some vendor implementations allow remote tools to manipulate UEFI configuration. Tools authenticate using the UEFI administrator password or by using command packages signed by Secure Boot keys. Some implementations require administrators to prove physical presence with keyboard presses – a requirement that may complicate UEFI management. Confirm behavior per make and model.

User Password

User UEFI passwords, if available, typically constrain the ability of device users to change the boot order, boot to external media, or change some non-administrative settings. Not all UEFI implementations contain user-configurable options. Different vendors separate different options between administrative and user control.

For devices that don't leave the building, consider setting user UEFI passwords only when 1) access to the advanced boot selection menu should be disabled, and 2) the administrative configuration password does not lock out the alteration of boot device or order. This scenario allows user passwords to be treated like local administrator privileges. Use a unique user password per device unless the user password interferes with the UEFI update process, remote management, or places an excessive burden on IT support services (a common UEFI user password may be more appropriate in this case).

Assign unique user UEFI passwords to mobile devices and their owners. Securing devices such as laptops and tablets is the focus here. A lost or stolen device could be booted to external media where commands to steal data or wipe the machine can be issued. The user UEFI password may be able to mitigate these types of compromises.

System or Storage Password

Some UEFI systems have system or storage drive passwords that act like physical presence checks. The system refuses to complete booting until a system password prompt has been satisfied. In the case of a storage password, the system will not access a storage device until the associated password is entered – each storage device has its own password.

System and storage passwords are not recommended. Each can interfere with remote management, UEFI updates, and OS updates by halting the boot process. Windows BitLocker®[9] and Linux Unified Key Setup (LUKS) are recommended alternatives because they allow the UEFI boot process and OS kernel initialization to complete thus permitting update mechanisms to function.

**** Note that most machines feature a reset button, reset jumper, and/or coin-sized battery somewhere on the device. Each may be capable of wiping out system or storage drive passwords. Bitlocker and LUKS are unaffected due to the storage of decryption information on the storage drive**

[9] Microsoft BitLocker is a registered trademark of Microsoft Corporation

(sometimes with TPM-supported protection). **

TOGGLE DRIVES, PORTS, AND DEVICES

Disable all devices and boot options not required for a normal boot. These are normally devices listed in the **Boot Sequence** or **Boot Devices** displays. Recovery, support, and reset options may also appear here. Any device that does not contain the OS can be disabled. These devices will still be available after booting completes, but they will not be taken into consideration during the boot process. A thick client may need all devices disabled except for the OS storage drive. A thin client may need all devices disabled except for the Network / PXE Boot interface and/or an embedded OS boot device.

Leaving devices enabled can allow the user to change the boot order by activating the advanced boot menu at the firmware test or logo screen. This flexibility could allow them to bypass an HDD and boot to a live media disc, for example. The user could also boot to a USB external device. When these options are disabled through the UEFI configuration system, the options will not be given after entering the advanced boot menu. However, once in the OS environment, the user can still interact with a disc or USB device per OS policy.

SYSTEM OPTIONS CONFIGURATION

Disable legacy Option ROMS (OROMs). Legacy OROMs are older and weakly validated (if at all) firmware packages. They interact with older storage, graphics, and network devices. Use newer OROMs that include cryptographically verifiable digital signatures compatible with UEFI Secure Boot. Signatures provide an element of supply chain integrity and firmware validation. Secure Boot can also whitelist or blacklist newer OROMs for granular boot device control.

Graphics cards are the most common stumbling point which rely upon legacy OROMs. NVidia®[10] cards in the 7-series (e.g.: GeForce™ 760) and newer as well as AMD®[11] cards in the 2-series (e.g.: Radeon™ 280) and newer have Secure Boot-compatible OROMs.

Disable OROM Keyboard Access if given the option. This mechanism allows a user to press key combinations like CTRL + I to alter the behavior of low-level storage controllers. Users may also be able to alter graphics processor properties, display properties, on-board audio, and other features. These features are normally handled by the IT administrative organization. Administrators would be able to re-enable OROM keyboard access only when an organizationally-requested change is necessary. Failure to set the “disable OROM keyboard access” option opens machines up to unauthorized drive mirroring carried out at the firmware level.

Enable and Activate Trusted Platform Module (TPM). A functioning TPM can be leveraged by several mechanisms on Windows and Linux®[12]. Windows 8, 8.1, and 10 will attempt to automatically provision and use a TPM if detected in the functional state. Leaving the TPM disabled weakens the system integrity and device identification properties of Microsoft’s Bitlocker, degrades Virtual Secure Mode (VSM) integrity isolation, prevents remote attestation systems from leveraging TPM keys, and limits the system’s ability to store boot-time measurements.

[10] NVidia and GeForce are registered trademarks of NVidia Corporation

[11] AMD and Radeon are trademarks of Advanced Micro Devices, Inc.

[12] Linux is a registered trademark of Linus Torvalds

Disable SATA adapters, eSATA, USB ports, USB headers, audio headers, SATA headers, serial ports, printer ports, and other communications buses if not in use. If these devices are left on, then an insider threat could gain physical access to a machine and swap physical hardware, add unauthorized hardware, or boot to unauthorized devices.

BOOT ORDER

Place the OS drive or adapter at the top of the boot list. Do not place removable media, USB, disc drives, floppies, or other devices at a higher priority than the OS drive or adapter. In a dual-boot situation, place both OS drives next to each other with the more-frequently accessed one being at a higher priority. In a thin client or PXE situation, place the network adapter or embedded OS drive at the highest boot priority. Devices not used for booting should be disabled.

RECOMMENDED SETTINGS

Table 1: Recommended Settings

Option	Recommended Setting	Comment
Boot mode	UEFI	Use UEFI boot mode instead of Legacy, CSM, or BIOS
Boot sequence	*	OS drive first. Disable devices not used for boot
Storage OROM access	Disable	Only enable for administrators
Legacy OROMs	Disable	Disable unless required by expansion devices (video card, storage controller, etc.)
Integrated NIC	Enable	Enable PXE if required by organization; Disable if not used
UEFI Network Stack	Enable	Enable if PXE or image servers are used by organization; Disable if not used
Parallel Port	Disable	Enable if required for legacy device
Serial Port	Disable	Enable if required for legacy device
SATA Operation	AHCI	Enable RAID or IRST (Intel Rapid Storage Technology) if appropriate
SATA ports	Connected only	Disable SATA ports not in use
SMART Reporting	Enable	Storage drive error reporting mechanism
USB Boot Support	Disable	Allows USB devices to boot; May be needed by some developers
External USB ports	*	Disable unused ports
USB power share	Disable	Charges devices through USB power
Keyboard backlight		May have levels of brightness
Unobtrusive mode		Disables or dims system indicator lights

Option	Recommended Setting	Comment
Internal modem	Disable	Enable if required for legacy network
Microphone		
eSATA port	Disable	Enable if external SATA ports are used
Free-fall protection		Relevant to spinning platter hard drives
Webcam		
ExpressCard	Disable	Enable if required by expansion device
SmartCard		Storage drive error reporting function
Module bay	Enable	Laptops with hot-swap bays; Controls disc media device
Optimus / Dynamic graphics	Enable/Auto	Energy-saving graphics switching
Video adapter	Auto	Switches between integrated and discrete graphics if present
Admin password	Set	UEFI administrative control options access
User password	Set	UEFI user boot configuration options access
System password	Not set	Stops system boot process. Interrupts updates
SATA password	Not set	Stops boot drive access. Interrupts updates
Strong passwords	Enable	Applies password complexity requirements to UEFI configuration accounts
Password configuration		Defer to organizational policies
Password bypass		Defer to organizational policies
Non-admin password changes	Disable	Do not allow non-admins to alter system config
Wireless switch changes		Defer to organizational wireless access policy
TPM security	Enable and Activate	Send power and I/O to the TPM
TPM ACPI support	Enable	Controls loading of measurements during boot
TPM PPI deprovision override	Enable	Allows OS to clear and re-enable TPM
TPM PPI provision override	Enable	Allows OS to activate TPM
Computrace		Anti-theft solution on some machines
CPU XD support	Enable	Execute-disable bit feature
OROM keyboard access	Disable	Only enable for administrators
Non-admin user setup lockout	Enable	Only allow admins into UEFI config
UEFI Secure Boot	Enable	Use in conjunction with supporting OS and/or hypervisor
Secure Boot custom mode	Disable	Enable custom if using custom key chain
Multi-core support	All	Controls energy use, heat, and performance of CPU
SpeedStep / CPU power states	Enable	CPU energy-saving features
C states / S3 sleep	Enable	CPU energy-saving features
TurboBoost / TurboCore	Enable	CPU performance boost feature
HyperThread / SMT	Enable	CPU scheduling optimizer
Rapid start		Accelerated boot from slow storage drives
Wake on AC		Influences boot behavior after power loss

Option	Recommended Setting	Comment
Wake on LAN		Allows monitoring of network traffic for wake commands
USB wake support		Allow USB devices to wake computer on action
WLAN		Wireless network toggle
WWAN		Cellular network toggle
Fastboot	Auto	Shortens some device self-check routines
Virtualization / VT-x / VPro	Enable	Virtualization extensions for hypervisors
VT-d / Virt directed I/O	Enable	Hypervisor performance optimization
Tagged TLB	Enable	
Rapid virtualization indexing / RVI	Enable	AMD-only. Equivalent to EPT
Extended Page Tables / EPT	Enable	Intel-only. Equivalent to RVI
Trusted execution / TXT		Windows: used when Trusted eXecution Engine (TXE) is installed. Linux and hypervisors: install TBoot and follow directions. Provision with TXT disabled. Enabling TXT locks NVRAM
Chassis intrusion		Log case-opening events
Overclocking		Increase CPU performance above factory limits
XMP memory profiles		High-performance RAM profiles
Fan control	Auto	Customizable cooling fan thresholds/levels

UEFI SECURE BOOT

UEFI Secure Boot is a **signature checking mechanism** that is added to the machine boot process. Only drivers, devices, OROMs, and other binaries with valid signatures will be executed at boot time. The validity of signatures is determined by the Secure Boot key chain. Most platform vendors provide several keys that are derived from a **Microsoft Root CA**. This key chain means that most device vendors who have a relationship with Microsoft and most Microsoft software will be able to use UEFI Secure Boot immediately without any additional configuration (Jumelet & Lich, 2017).

Each step of the UEFI boot process covered by Secure Boot requires a signature to be calculated before a binary can be executed. Firmware performs the signature check during DXE, BDS, and Bootloader phases. However, the exact operating system kernel and modules selected by the bootloader are not measured by the firmware UEFI Secure Boot implementation (see figure 2). **The bootloader software implementation must be Secure Boot-aware or otherwise continue the Secure Boot signature check chain.** Microsoft bootloaders leverage UEFI Secure Boot keys and databases. In contrast, **Linux bootloaders rely upon Machine Owner Key (MOK) and Shim** to switch to a key chain provided by Red Hat or Canonical rather than continuing to use UEFI variables.

Linux distributions use MOK and Shim to avoid the logistics involved with having Microsoft sign every single kernel update.

UEFI Secure Boot Signature Checking Zones

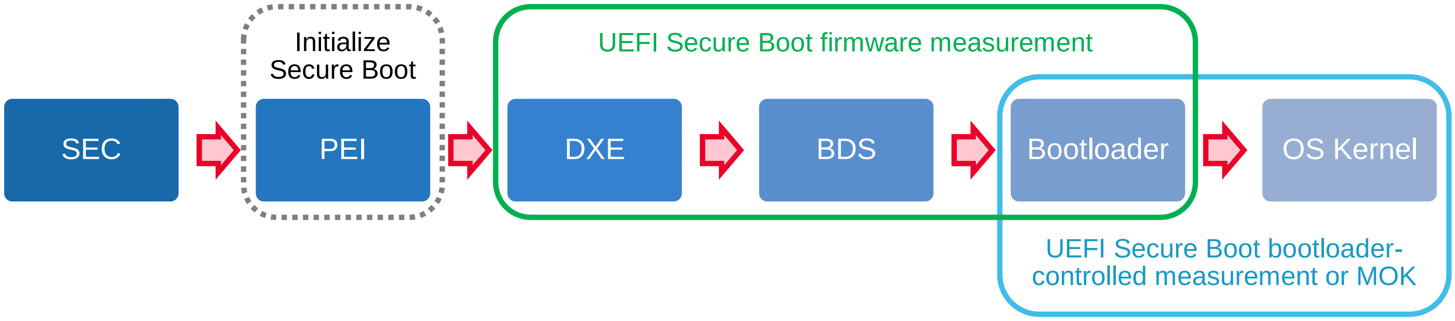
Initialize Secure Boot
UEFI Secure Boot firmware measurement
SEC
PEI
DXE
BDS
Bootloader
OS Kernel
UEFI Secure Boot bootloader-controlled measurement or MOK

Figure 2 -- UEFI boot process with Secure Boot implementation phases.

CUSTOMIZED KEYS

Some IT organizations may want to further constrain Secure Boot or employ the use of their own bootable, signed binaries. Most systems employing UEFI Secure Boot have a custom or advanced mode that allows replacement or augmentation of the preloaded, factory keys. The key hierarchy is as follows:

- Platform Key (PK)
- Key Exchange Key (KEK)
- Whitelist Database (DB) and Whitelist Database Keys (DBK)
- Blacklist Database (DBX)

The PK, an RSA 2048 public/private key pair, is the root key. Each PK is stored on the machine in the form of an X.509 certificate. The PK controls access to platform UEFI environment variables, UEFI configuration changes requested by the OS, and restricts changes to the KEK(s) and DB keys. The PK does not need to sign the KEK(s) meaning that there is no requirement for the PK and KEK(s) to be linked in a public key cryptography certificate chain. Most machines ship with PKs established by the OEM. A Dell machine will have a Dell PK, for example. A PK can be unique per machine, identical across a product line, identical based on location, or some other configuration to meet the security needs of an organization. The PK can be changed by using the UEFI configuration interface through an administrative session or via automated, vendor-specific tools.

A unique PK per machine is the most secure solution for replacing the PK. However, PKs may interfere with the UEFI firmware update mechanism. In that case, the PK should be common across the IT infrastructure to simplify the process of updating firmware. **Consider having the PK match the distribution of UEFI configuration passwords.** If machines have common UEFI configuration passwords, then also use custom PKs. If they employ unique passwords, then also use unique PKs.

The **KEK, another RSA 2048 public/private key pair, is responsible for signing keys** in the whitelist DB, blacklist DBX, and any EFI binaries that should be trusted during boot. There can be multiple KEKs. Most vendors supply a default KEK generated by Microsoft. The KEK is meant to link the OS environment to the firmware by defining which drivers, devices, boot loaders, and kernels can be used as part of the boot process. Replacing the KEK means putting an organizational KEK in place. Leaving the default vendor or Microsoft KEK allows any device or binary they've signed to be used during boot – nearly any version of

Windows, RHEL, Fedora, or Ubuntu can boot. Altering the KEK store requires UEFI config administrative access or vendor-specific utilities that have been signed with the PK. Adding a custom KEK enables anything it has signed to be executed when checked by UEFI Secure Boot. The custom KEK can also authorize DBKs. This extensibility can allow Secure Boot to trust custom peripherals, OS images, or firmware modules.

Replacing default KEK(s) allows IT organizations to specify which DB records, DBX records, and binaries are permissible during boot without the upstream influence of vendors. In this situation, the organization's CA should certify the KEK. **All machines within an infrastructure should have the same custom KEK.** Avoid using the KEK for signing binaries. Removing default KEK(s) and only using a custom one exerts maximum control over a machine's UEFI Secure Boot process.

The DB and DBX are to whitelist and blacklist boot content, respectively. **Using one or more DB Keys (DBK) is recommended even in the presence of custom KEK(s).** DBKs can be easily removed or swapped between the whitelist DB and blacklist DBX without necessitating a KEK change. Create a DB key and then load the certificate into the DB whitelist. Then use that same key to sign boot drivers, binaries, and OS loads. Hashes of approved drivers, binaries, and OS loads can also be placed in the DB without using the DBK.

The whitelist DB allows a machine to boot to known-good content authorized by an organization. Not just any version of Windows or Linux would be allowed – specifically Windows 10 build 1607 or Ubuntu 16.04 could be allowed. Additional specific restrictions could include: RAID controller firmware 2.06.0005, TPM in FIPS 140-2 mode rather than a FIPS non-compliant mode, or a known-good boot loader rather than a debug one that ignores UEFI Secure Boot. Organizations can use this power to prevent misconfigured machines from booting into unapproved states or to stop the introduction of unauthorized firmware or software during the boot process.

TRUSTED PLATFORM MODULE (TPM)

Trusted Platform Module (TPM) is a hardware security module available on most business-class machines. Some organizations, such as the DOD, mandate the purchasing of machines equipped with TPMs. Some vendors, such as Microsoft, require the TPM for logo certification compliance.

TPM is a passive measurement holder. **Built-in Platform Configuration Registers (PCRs) summarize the integrity of a given machine.** There are typically 24 PCR banks. 0 through 7 are filled out by the UEFI firmware. 8 through 15 are in the realm of the OS. 16 through 23 are flexible. Each PCR holds a SHA-1 (TPM 1.2) or SHA-256 (TPM 2.0) hash. The hashes start at an all zero value. To create a PCR hash: a measurement of a boot binary is taken, the current PCR value is appended, the combination is hashed using the appropriate hash algorithm per TPM generation, and then the new value becomes the updated "extended" PCR as seen in the formula below. This scheme creates a measurement log and history of 1-way hashes describing the integrity of boot and runtime (if available) system integrity. See figure 3 for an overview of what each PCR covers.

Extended PCR value = SHA(SHA(measured file) + initial PCR value)

A TPM has no active system integrity enforcement mechanism. TPM is a passive observer of boot and runtime activities. A TPM alone cannot stop the booting of a compromised or incorrectly configured system like UEFI Secure Boot can. However, tools have been developed for Windows, Linux, and hypervisors that leverage the integrity information contained in the TPM.

TPM Hashes of UEFI Boot Phases

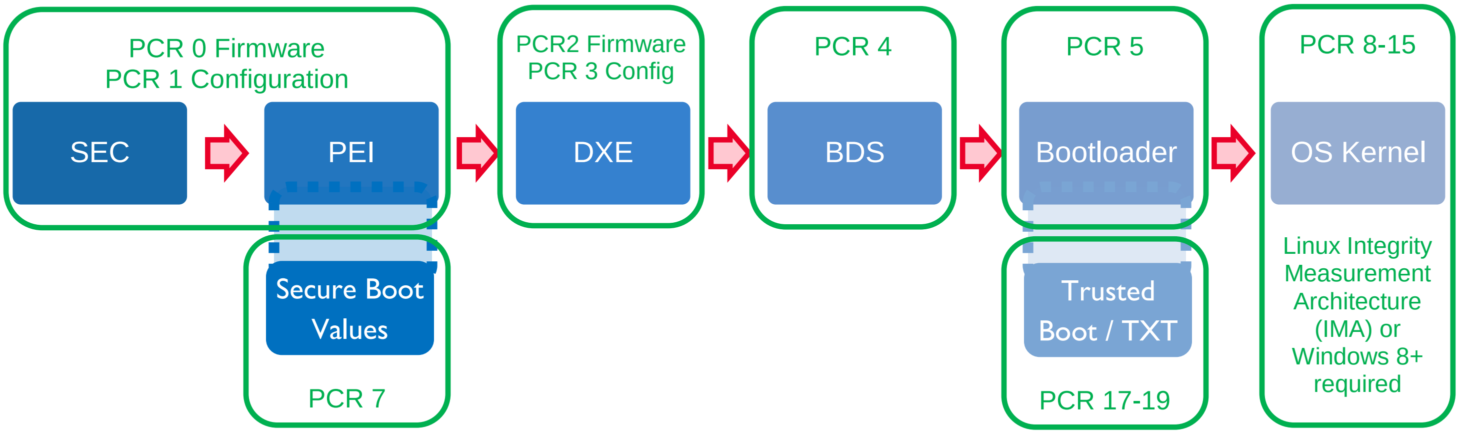

Figure 3 -- TPM PCR scope with relation to the UEFI boot process.

Windows can use Bitlocker to encrypt the contents of storage drives. **Bitlocker can leverage the TPM Primary Key (2.0) or Storage Root Key (1.2) along with PCRs to protect the Bitlocker Volume Master Key (VMK).** If the system boots up in the known-good configuration, then the PCRs will be in the expected state and the Bitlocker decryption key will be released (Lich, 2017). Changes to UEFI configuration, firmware updates, and equipment changes may cause the TPM PCRs to change – events that would change the nature of PCRs and requite the Bitlocker key recovery system built in to some update mechanisms and Active Directory®[13] (AD). Failure to enable both Bitlocker and TPM introduces the situation where the system's storage drive is unlocked without the use of firmware measurements, VMK key blob is subject to unauthorized migration, or booting with unapproved Secure Boot keys is allowed.

Linux has a solution like Bitlocker called LUKS. LUKS can have TPM support added by integrating the tpm-luks open source patch. LUKS key recovery can be integrated into LDAP much like Bitlocker's AD integration (Red Hat, 2017).

Linux and some hypervisors can use Trusted Boot (TBoot). **TBoot uses TPM PCRs and TPM onboard memory to store a known-good configuration and boot policy.** TBoot executes after the bootloader but before the OS kernel – a policy enforcement point where the system integrity is checked, hardware security features are activated, and an auditing layer is placed between kernel permissions and user-space permissions. Failure to use TBoot could leave the sleep and resume scripts vulnerable to replacement, and allow a machine to boot with unapproved Secure Boot keys (Canonical, 2017).

FIRMWARE UPDATE

Firmware updates include security fixes in addition to expanded hardware support and bug fixes. A seemingly insignificant update, such as setting the SPI flash lock bit as part of boot, can determine if a machine is vulnerable to an S3 resume vector hijack. This attack writes malicious firmware prior to OS

[13] Active Directory is a registered trademark of Microsoft Corporation

restricting the use of direct memory access. **1 bit can be the difference between a safe boot and acquiring an advanced persistent threat.**

Apply firmware updates when security fixes are provided by the machine vendor. Test an individual machine for deployment authorization problems, Secure Boot key chain conflicts, and any impact on software security mechanisms like Bitlocker or LUKS.

Some components within a machine may have individual firmware updates not covered by the machine vendor. Example devices include TPMs, wireless controllers, and discrete video cards. Check these devices for firmware updates periodically.

POTENTIAL STUMBLING BLOCKS

UEFI firmware configuration passwords can cause update problems on some systems. The firmware update utility may request or require provisioning with a password to apply an update to the system. Check with the update and firmware vendor to verify if this condition is the case. Machines that fall into this category are best managed with a common UEFI password to minimize complexity.

UEFI Secure Boot may refuse to execute a firmware update when custom keys are applied. This situation could be caused by the vendor update package not being provisioned for the custom PK or KEK. Check with the update vendor to determine which key or database is responsible for permitting updates. If the PK is used, then use a consistent PK across all machines in the infrastructure to minimize update complexity.

Bitlocker, TBoot, tpm_luks, and other utilities that look at TPM PCRs will detect a change in PCR values following a firmware update and could cause a system to fail to boot. Windows AD and Linux LDAP have mechanisms to recover a cleared or inaccessible (due to PCR sealing) disk encryption. TBoot will need Trusted eXecution Technology (TXT) disabled, an updated Launch Control Policy (LCP) loaded into the TPM's memory, and then TXT re-enabled.

RECOMMENDATIONS FOR OEMS

Secure by default recommended settings – Systems should ship with UEFI configuration options set like those outlined in section 3.E. Some systems continue to ship with disabled or inactive TPMs, non-administrative user setup lockout booleans not set, and legacy communication standards (parallel port, serial, eSATA, etc.) enabled.

Unify update mechanism – There is considerable variation in firmware update mechanisms (Dell, 2017). The firmware industry needs to reach consensus regarding common, secure, and reliable tools that can be pushed out like Windows and Linux update packages. Decades of improving security behaviors and update practices on the OS environment part of computing need to migrate to the firmware realm.

Some systems update via an executable run inside Windows, some require entering UEFI config with a USB drive plugged in that contains a binary firmware image, and others require a separate DOS-like boot disc. Worse, some UEFI implementations allow unauthenticated updates delivered via unencrypted internet connections (Indrora, 2017). Vendors should standardize around digitally signed UEFI update capsules that can be distributed by automated, OS-integrated, update mechanisms.

Zero-touch update – Administrators should not have to visit machines to manually apply firmware updates. Firmware updates should be protected by cryptographic signatures that can be verified by endpoints before automatic installation. A message identifying a firmware update in progress should be displayed with a note not to power off the machine.

Clear support lifetimes – Platform firmware update support lifetimes are often unclear or not specified. Companies like Microsoft and Canonical give clear timelines for the support duration of their software products. Vendors should do the same with regards to firmware update support. An indication of warranty is not sufficient. Clearly publicize firmware update support lifetimes.

Provide known-good hashes that match TPM measurements – Enterprise customers need to know that the firmware they're running is genuine and intended. Providing SHA-1 and SHA-256 hashes that match those collected by the TPM is critical for auditing the integrity of a machine. **IT departments should not observe a machine to be correct – they should have a hash that validates correctness.**

RECOMMENDATIONS FOR NETWORK OPERATORS

Firmware updates – Firmware updates should be applied to machines in a timely fashion just like any other patch. Organizations have gotten strong at patching vulnerabilities in different software products. However, firmware is commonly ignored. Machines may never get a firmware newer than the one they shipped with. The software embedded within a motherboard, hard drive, graphics processor, or other device should get the same update-awareness as a virus scanner or document reader.

Configuration lock down – Many organizations fail to take even the minimum step of setting a UEFI configuration password on their machines. This omission allows users to subvert restrictions on boot devices and potentially compromise infrastructure resources. Malicious configurations could also result in damage to a machine via manipulating energy use and cooling systems. Policies don't allow for running Windows without the separation of users and administrators. The same mentality needs to be adapted to the UEFI realm.

Establish a UEFI configuration administrator password. Make the password unique per machine. If firmware updates require a password to apply, then make the password the same on all machines. Do not leave administrative UEFI passwords blank or unset. Apply UEFI user passwords to secure mobile devices and provide a local administrative privilege to fixed workstations and servers.

Enable UEFI Secure Boot – If organizational hardware and software support the requirements for Secure Boot, then enable it and use it. Traditional virus scanners and anti-malware solutions don't start until after the OS kernel is executed – sometimes delayed further by other boot processes and services. Secure Boot is the pre-OS anti-malware solution designed to keep firmware and initial OS software in a known good state.

Consider establishing a custom Secure Boot key chain. Investigate what impact custom keys would have on the ability to deliver firmware and software updates.

UEFI training – Do users know not to cut power to a machine that is performing a firmware update? Do network administrators know how to push out a firmware update and then test for successful installation? Are auditors and analysts looking at UEFI configuration variables and firmware integrity? The likely answer to most of these question is no – a call for education about the firmware environment that sits below the well-known software OS environment.

REFERENCES

Canonical (2017, March 2). Trusted Platform Module. Retrieved May 23, 2017, from https://wiki.ubuntu.com/Security/Features#Trusted_Platform_Module

Dell (2016, June 1). Dell Driver and Firmware Update Strategies for Server Storage and Networking Systems. Retrieved May 23, 2017, from http://www.dell.com/support/article/us/en/04/SLN293301/dell-driver-and-firmware-update-strategies-for-server-storage-and-networking-systems?lang=EN

Indrora (2016, June 6). DeadUpdate; Or, How I learned to stop worrying and execute arbitrary executables from HTTP. Retrieved May 23, 2017, from http://teletext.zaibatsutel.net/post/145370716258/deadupdate-or-how-i-learned-to-stop-worrying-and

Jumelet, A., & Lich, B. (2017, April 24). Control the health of Windows 10-based devices. Retrieved May 23, 2017, from https://docs.microsoft.com/en-us/windows/device-security/protect-high-value-assets-by-controlling-the-health-of-windows-10-based-devices

Lich, B. (2017, April 24). BitLocker Countermeasures. Retrieved May 23, 2017, from https://docs.microsoft.com/en-us/windows/device-security/bitlocker/bitlocker-countermeasures#uefi-and-secure-boot

Red Hat (2017, January 20). Using LUKS Disk Encryption. Retrieved May 23, 2017, from https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Encryption.html

Wei, D. & Long, Q. & Shen, J. (2013, January 17). Build Safety from Bare Metal. Retrieved May 23, 2017, from https://firmware.intel.com/sites/default/files/BJ13_PTAS002_101_ENGf.pdf

ACRONYMS

Acronym	Meaning
ACPI	Advanced Configuration and Power Interface
AD	Microsoft corporation product Active Directory
AHCI	Advanced Host Controller Interface
AMD	Microprocessor company named Advanced Micro Devices
ARM	Microprocessor company formerly known as Advanced RISC Machine
BDS	Boot Device Select UEFI boot phase
BIOS	Basic Input/Output System
CA	Certificate Authority
CPU	Central Processing Unit
CRTM	Core Root of Trust for Measurement starts system integrity hashing chain
CSM	Compatibility Support Module providing some BIOS functions omitted from UEFI
DB	Secure Boot Whitelist Database
DBK	Database Key used with Secure Boot databases
DBX	Secure Boot Blacklist Database
DoD	US government Department of Defense
DOS	Disk Operating System
DXE	Driver Execution Environment UEFI boot phase
GPT	GUID Partitioning Table
EFI	Extensible Firmware Interface – the foundation which UEFI is built upon. Originally created by Intel corporation as a proprietary solution. Binaries designed to run in the UEFI environment may also be called EFI binaries as opposed to UEFI binaries
EPT	Extended Page Tables Intel corporation equivalent to RVI
eSATA	External Serial Advanced Technology Attachment
FIPS	Federal Information Processing Standard
GNOME	Linux desktop user environment
GRUB	Linux boot loader
HDD	Hard Disk Drive
IMA	Integrity Measurement Architecture provides runtime TPM hashing
I/O	Input/Output
IRST	Intel corporation Rapid Storage Technology for attached storage disks
IT	Information Technology (department or device)
KEK	Secure Boot Key Exchange Key
LAN	Local Area Network connection
LCP	Launch Control Policy used by TBoot
LDAP	Lightweight Directory Access Protocol is Linux equivalent to Microsoft AD
LUKS	Linux Unified Key Setup used for drive encryption
MBR	Master Boot Record partition scheme
MBR2GPT	Utility to convert from MBR disks to GPT disks
MOK	Machine Owner Key used for Linux extension of Secure Boot

Acronym	Meaning
NIC	Network Interface Controller
NVRAM	Non-Volatile Random-Access Memory storage space on TPMs
OROMs	Option Read-Only Memory firmware configuration branching mechanism
OS	Operating System such as Microsoft Windows or Red Hat Linux
PC	Personal Computer
PCR	Platform Configuration Register used by TPM to store hashes of integrity hashes
PEI	Pre-EFI Initialization phase for UEFI boot
PK	Secure Boot Platform Key
PPI	Physical Presence Interface
RAID	Redundant Array of Independent Disks
rEFInd	UEFI Boot Loader
RAM	Random-Access Memory
RHEL	Red Hat Enterprise Linux operating system
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RSA	Ron Rivest, Adi Shamir, and Leonard Adleman cryptosystem algorithms
RVI	Rapid Virtualization Indexing AMD corporation equivalent to EPT
TPM	Trusted Platform Module security chip
TXE	Trusted Execution Environment restricted kernel memory space
TXT	Intel corporation Trusted Execution Technology
S3	Sleep state 3 shuts down power to most PC components except RAM
SHA	Secure Hashing Algorithm
SMT	Symmetric Multithreading for multiple CPU cores, threads, paths
SATA	Serial Advanced Technology Attachment
SEC	Security phase of UEFI boot
UEFI	Unified Extensible Firmware Interface that is a derivative from the proprietary EFI solution created by Intel corporation. Governed by an industry consortium called the UEFI Forum
TBoot	Trusted Boot open source Intel mechanism
TLB	Translation Look-aside Buffer memory management accelerator
VMK	Volume Management Key for Microsoft Bitlocker
VSM	Virtual Secure Mode suite of device-hardening features in Microsoft Windows
VT-d	Virtualization Technology for Directed I/O
VPRO	Intel corporation branding for devices supporting multiple virtualization enhancements and TBoot
WLAN	Wireless Local Area Network
WLAN	Wireless Local Area Network
WWAN	Wireless Wide Area Network normally indicates presence of cellular adapter
XD	Execute Disable bit allows CPU to disable execution in memory spaces
XMP	Extreme Memory Profile used for controlling RAM timing
USB	Universal Serial Bus connects peripheral devices