APPLICATION NOTE

Intel Corporation

Secure the Network Infrastructure – Secure Boot Methodologies

Authors

Kapil Sood

Seosamh O'Riordain

John Geary

Heqing Zhu

1 Introduction

Industry-proven platform security technologies are crucial to protect against ever-increasing cyber-attacks and threats. 5G networks will incorporate more open source technologies such as Linux* or DPDK on standard server systems. Edge computing will accelerate the convergence of multi-party services onto a single system. Therefore, a security first mindset needs to be front and center when planning network infrastructure. Using proven technologies will be critical in protecting data and IP in a secured network infrastructure. Intel has built security features into its silicon, firmware, and software technologies in conjunction with industry leaders.

This document describes the Secure Boot methodology using Intel® Boot Guard technology and UEFI Secure Boot. Intel® Boot Guard provides the hardware Root of Trust (RoT) for platform boot and UEFI Secure Boot is defined by the UEFI standards to verify IA firmware signatures prior to boot [20]. Together, Intel® Boot Guard and UEFI Secure Boot can be implemented to create a platform chain of trust for boot and to ensure that the Intel and OEM platform firmware is authorized to run on that platform. This document is targeted for the Intel® Xeon® Scalable platform (codenamed Purley) and second generation Intel® Xeon® Scalable platform (codenamed Purley-Refresh).

This document can act as a reference to implement Intel® Boot Guard and UEFI secure boot for networking platforms including packet processing workloads, and for bare metal, DPDK, vEPC, vCPE, and other virtualized and non-virtualized systems. This document is a guide and not meant to duplicate the reference documents in Table 2. It is highly recommended that readers refer to these documents for enabling and design details.

This document is part of the Network Transformation Experience Kit, which is available at: https://networkbuilders.intel.com/

Table of Contents

1 Introduction 1
1.1 Terminology 3
1.2 Reference Documents 3

2 Secure Boot Overview 4
2.1 Secure Boot Architecture and Flow 4
2.2 Firmware Threat Model 5
2.3 Secure Boot Best Practices 5
2.4 Secure Boot Use Cases and Requirements 6
2.5 Secure Boot Ecosystem Enabling Flow 6
2.6 Secure Boot support on Intel® Server Platforms 6

3 Intel® Boot Guard Basics 7

4 UEFI Secure Boot Basics 7
4.1 Authenticated Variables and Key Provisioning 8
4.2 Secure Firmware Update 8
4.3 Firmware Recovery 8

5 UEFI Secure Boot: Operating System Enhancements 8
5.1 Signed Kernel Modules 8
5.2 Machine Owner Keys 9
5.3 Kernel Security Enhancements 10
5.4 Enhanced Platform Awareness (EPA) for Secure Boot 10

6 Networking Workloads on UEFI Secure Boot Platforms 10
6.1 Networking Workload Impact 10
6.2 Networking Workload Mitigations for Secure Boot 10

7 Summary and Call to Action 11

Figures

Figure 1. Secure Boot Flow 4
Figure 2. Secure Boot Architecture 5
Figure 3. Intel® Architecture Secure Boot Ecosystem Enabling Flow 6
Figure 4. UEFI Secure Boot Components 7
Figure 5. Shim Certificate Signing 9
Figure 6. OSV Secure Boot Chain of Trust 9
Figure 7. MOK and User Signed Images 9
Figure 8. UEFI Secure Boot Impact on Networking Workloads 10
Figure 9. UEFI Secure Boot Mitigations 11

Tables

Table 1. Terminology 3
Table 2. Reference Documents 3
Table 3. Secure Boot Use Cases 6
Table 4. UEFI Authenticated Variables 8

1.1 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
BIOS	Basic Input / Output System
FW	Firmware
Intel® RDT	Intel® Resource Director Technology
KEK	Key Exchange Key
OS	Operating System
RoT	Root of Trust
UEFI	Unified Extensible Firmware Interface
VMM	Virtual Machine Monitor

1.2 Reference Documents

Table 2. Reference Documents

Ref #	Document	Document No./Location
1	Intel® Xeon® Processor Scalable Family Technical Overview	https://software.intel.com/en-us/articles/intel-xeon-processor-scalable-family-technical-overview
2	UEFI Specification and open source	http://www.uefi.org/sites/default/files/resources/UEFI%20Spec%202_6%20Errata%20A%20final.pdf https://www.tianocore.org/
3	ETSI NFV Security Standards (SEC001, SEC012, SEC013, others)	http://www.etsi.org/technologies-clusters/technologies/nfv
4	Linux* Usage	https://firmware.intel.com/sites/default/files/SF12_EFIS001_100.pdf https://firmware.intel.com/blog/using-mok-and-uefi-secure-boot-suse-linux
5	Windows* Usage	https://technet.microsoft.com/en-us/library/hh824987.aspx
6	Open Source Implementation	https://github.com/tianocore/edk2/tree/master/SecurityPkg
7	NIST BIOS Security Guidelines: SP800-147 and NIST SP800-147B	http://dx.doi.org/10.6028/NIST.SP.800-147B
8	Red Hat* UEFI Secure Boot	https://www.redhat.com/es/blog/uefi-secure-boot https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sec-uefi_secure_boot https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Kernel_Administration_Guide/sect-signing-kernel-modules-for-secure-boot.html
9	Ubuntu* UEFI Secure Boot	https://wiki.ubuntu.com/UEFI/SecureBoot
10	UEFI Rootkits and Attacks	https://www.blackhat.com/presentations/bh-usa-07/Heasman/Presentation/bh-usa-07-heasman.pdf https://www.blackhat.com/docs/asia-17/materials/asia-17-Matrosov-The-UEFI-Firmware-Rootkits-Myths-And-Reality.pdf https://www.welivesecurity.com/2018/09/27/lojax-first-uefi-rootkit-found-wild-courtesy-sednit-group/ https://eclypsium.com/2018/10/01/uefi-attacks-in-the-wild/
11	Google* Infrastructure Security White Paper	https://cloud.google.com/security/infrastructure/design/resources/google_infrastructure_whitepaper_fa.pdf https://cloud.google.com/blog/products/gcp/titan-in-depth-security-in-plaintext
12	SUSE* Linux* Enterprise Server: UEFI Secure Boot	https://www.suse.com/documentation/sles-15/book_sle_admin/data/sec_uefi_secboot.html
13	Microsoft* UEFI Secure Boot	https://docs.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-secure-boot
14	Dell* Servers with Intel® Boot Guard	https://downloads.dell.com/solutions/servers-solution-resources/Direct%20from%20Development%20-%20Cyber-Resiliency%20In%20Chipset%20and%20BIOS.pdf
15	Open Compute Project Cerberus*	https://azure.microsoft.com/en-us/blog/microsofts-project-olympus-delivers-cloud-hardware-innovation-at-scale/
16	Google* Titan	https://cloud.google.com/blog/products/gcp/titan-in-depth-security-in-plaintext

Ref #	Document	Document No./Location
17	Intel® Platform Firmware Resilience (Intel® PFR)	https://www.intel.com/content/www/us/en/data-center-blocks/business/pfr-server-blocks.html
18	DPDK Linux Drivers: VFIO	http://doc.dpdk.org/guides/linux_gsg/linux_drivers.html#vfio
19	Dell Server Infrastructure Firmware Security Paper	https://delltechnologiesworldonline.com/2017/connect/fileDownload/session/F6DCEE362410E8EEC1CD044710AC8FB4/server.07%20Is%20your%20Server%20Infrastructure%20Secure.pdf
20	Establishing the root of trust	http://www.uefi.org/sites/default/files/resources/UEFI%20RoT%20white%20paper_Final%208%208%2016%20%28003%29.pdf
21	Node feature discovery for Kubernetes*	https://github.com/kubernetes-incubator/node-feature-discovery
22	Kubernetes EPA enhancements for Secure Boot	https://github.com/kubernetes-sigs/node-feature-discovery/pull/87
23	National Security Agency (NSA) Cybersecurity Report: UEFI Defensive Practices Guidance	https://www.nsa.gov/Portals/70/documents/what-we-do/cybersecurity/professional-resources/ctr-uefi-defensive-practices-guidance.pdf?ver=2018-11-06-074836-090
24	Fedora* Signing Kernel Modules for Secure Boot	https://docs.fedoraproject.org/en-US/Fedora/26/html/System_Administrators_Guide/sect-signing-kernel-modules-for-secure-boot.html
25	Ubuntu Secure Boot Signing	https://blog.ubuntu.com/2017/08/11/how-to-sign-things-for-secure-bootq

2 Secure Boot Overview

Secure Boot mechanisms can be enabled by platform owners to ensure that only authorized firmware images can be installed and run on their platforms. This helps mitigate the risk of attacks targeting low-level, highly privileged platform components.

2.1 Secure Boot Architecture and Flow

Platform Secure Boot is achieved by progressively cryptographically verifying every critical piece of firmware as it is installed on the platform, starting with a hardware-based Root-of-Trust (RoT). A verified firmware is installed, and then subsequently verifies the digital signature check on the next firmware component before installing that new component. This essentially creates a boot chain of trust, starting from the hardware RoT (Intel® Boot Guard) all the way up to the platform OS, which can extend the chain to the loading of signed kernel modules.

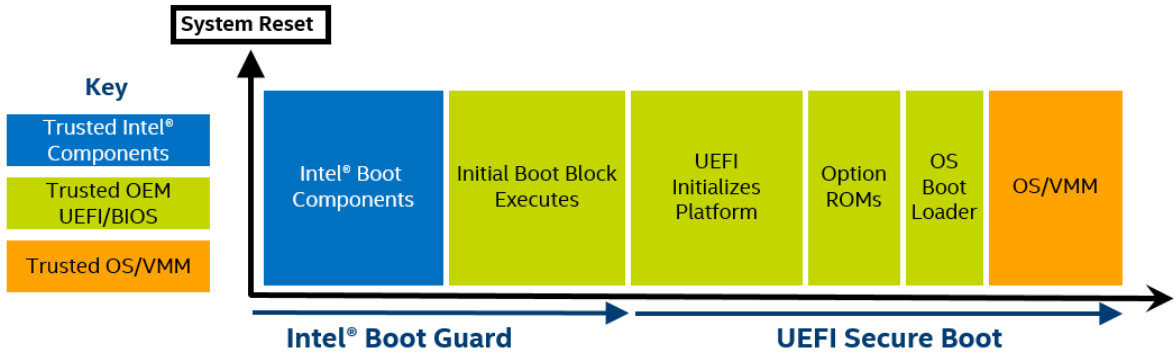

Figure 1. Secure Boot Flow

Figure 2 illustrates a system Secure Boot architecture for a network platform system, including key Intel® architecture hardware RoT, UEFI, kernel, and workload composition.

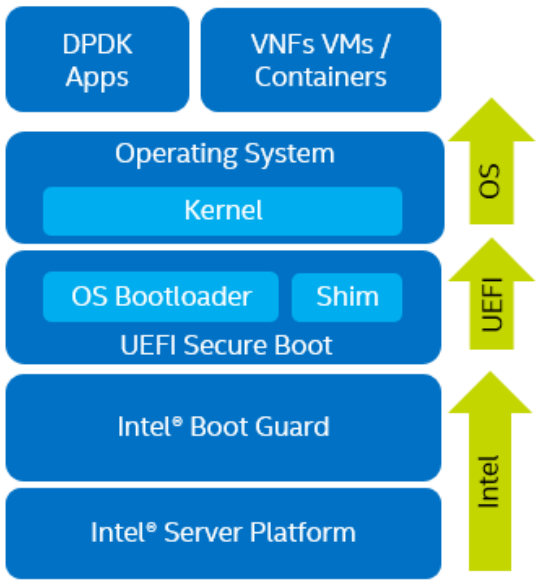

Figure 2. Secure Boot Architecture

The following sections describe these boot flows and architecture. Refer to Reference docs [1], [2], [8], and [9] in Table 2 for further details.

2.2 Firmware Threat Model

Exposures and vulnerabilities on the platforms continue to grow and attacks are referenced in [10]. Attacks increasingly are now being rooted in the most basic component on every platform: the underlying firmware which is based on the UEFI standard. As the capabilities of the UEFI firmware grows (including LAN access), so does the potential attack surface that is exposed to attackers, including zero day vulnerabilities. Firmware-based Bootkits and Rootkits can stay undetected and be exploited for long periods of time. This is a serious problem for critical infrastructure including Networks and Cloud deployments.

UEFI standard implementations are open sourced, and increasingly offered as the firmware of choice on standard servers, often modified and updated by a complex firmware supply chain including hardware and platform vendors, Independent BIOS Vendors (IBVs), ODM/OEMs, and Platform Owners. The eventual firmware installed on the production servers must be thoroughly security tested, policy enforced and authenticated at every boot and update to minimize security exposures.

2.3 Secure Boot Best Practices

Multiple exploits and vulnerabilities have been detected in the UEFI firmware design and implementation, and alongside, possible mitigations have also been suggested. These include correct implementation of the UEFI Secure Boot, deploying platforms with a hardware based RoT, following best security BIOS practices prescribed by NIST [7] and NSA [23] for secured server deployments, and also illustrated in Dell Server Infrastructure security document [19]. ETSI NFV has defined multiple security specifications and security requirements for NFV Infrastructure (NFVI) deployments [2], which are now being implemented in NFV and 5G products and open source projects. Very large, full scale-out deployments at Cloud Providers like Google* tout the use of a secure verification chain as basis of their Secure Boot stack. Google “use cryptographic signatures over low-level components like the BIOS, bootloader, kernel, and base operating system image. These signatures can be validated during each boot or update” [11] to assure customers of the security of their infrastructure.

Intel has continued to lead in this space, both in driving security research to detect vulnerabilities before they can be exploited to working with the industry to make UEFI implementations more secure. Intel’s hardware RoT components is branded as Intel® Boot Guard which was launched on Intel servers starting with the Intel® Xeon® Scalable platform. Intel has co-defined the UEFI standard [2], including the UEFI Secure Boot with the industry and community, facilitating the open source implementations. The UEFI Secure Boot solution is widely supported and deployed for Red Hat*, SUSE*, and Ubuntu* operating systems, and most large OEMs have a deployment-ready solution.

Code signing indicates that the Platform Owner has authorized that image to be deployed on certain platforms provisioned with the corresponding Public Keys. It is, therefore, highly recommend that the Platform Owner security validate the code and images prior to signing and assign a security version.

Intel highly recommends that our customer server deployments follow the NIST BIOS Security guidelines [7] and industry boot security best practices for platform firmware based on a hardware RoT [11].

2.4 Secure Boot Use Cases and Requirements

There are multiple scenarios where Secure Boot remains a fundamental security building block, especially for critical and sensitive infrastructure deployments. In addition, deployments that are in remote locations often without physical controls may need to mandate hardware-based RoT secure boot policy. Table 3 contains a subset of the use cases and corresponding platform security requirements.

Table 3. Secure Boot Use Cases

Use Case	Platform Security Requirements
System Owner/Admin. ensures that their deployed platform(s) boot with their authorized FW/SW images only (including OEM authorized by Operator).	Hardware-Based Root of trust HW-rooted crypto-signatures on all FW and OS images Signature verification at boot of all FW and OS images Signing Key Management and Boot Policy Control Anti-rollback controls
System Owner/Admin wants to prevent installation of boot update images that may be tampered with while in repository.	Signature verification at boot of all updated FW and OS images HW Fuse-Based Security Versioning
System Owner/Admin. meets regulatory, industry and standards compliance security requirements.	NIST SP800-147B ETSI NFV SEC001, SEC012, SEC013
System Owner/Admin. prevents booting from unauthorized replacement of image flash memory.	Securely verify and deploy boot images from flash memory
Platform is available and serviceable through Operator's trusted parties (including OEMs, TEMs, and others).	Secure Recovery mechanism, per policy

2.5 Secure Boot Ecosystem Enabling Flow

Deploying a Secure Boot solution on Intel Platforms comprises multiple vendors, each delivering a portion of the overall integrated solution, as shown in Figure 3.

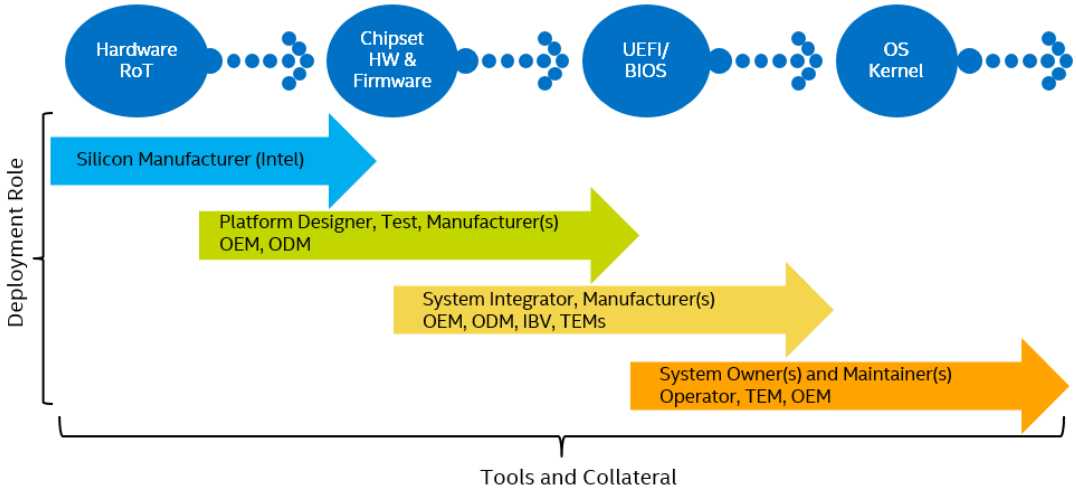

Figure 3. Intel® Architecture Secure Boot Ecosystem Enabling Flow

The role of the various vendors is as follows:

- Intel delivers a hardware-based root of trust, which is the Intel® Boot Guard technology. Intel has worked with the UEFI and OS vendors to enable Secure Boot platforms to our end customers.
- OEM and ODMs design the platforms and integrate the BIOS on the platform, designing in Intel® Boot Guard by provisioning the signing key hashes in their manufacturing flows. OEMs validate the platform firmware.
- OEMs integrate UEFI Secure Boot credentials, based on their end customer requirements and keys. They also provide integrated secure Recovery and secure Firmware Update system components.
- The OS vendors provide the OS Boot Loader and signed OS kernels. These system components are integrated into the platform by OEMs, TEMS, and/or System integrators.
- The secure boot platform is delivered to the marketplace by the OEM.

2.6 Secure Boot support on Intel® Server Platforms

Intel Server platforms use the UEFI specifications based firmware, and as such, support UEFI Secure Boot on all platforms. The evolution of UEFI Secure Boot follows the UEFI standards specifications revisions and open source, TianoCore project [2].

UEFI Secure Boot, by itself, can be deployed on any Intel Server platform and in this case, the UEFI firmware itself is the Root of Trust for Secure Boot. Clearly, trust for platform boot is rooted on the UEFI firmware, and as such, any adversary capable of installing their firmware on a machine can effectively control that machine and possibly other machines, possibly as an Advanced Persistent Threat (APT).

Intel® Boot Guard is a hardware-based Root-of-Trust (RoT) that was introduced on the Intel® Xeon® Scalable platform. Starting in 2018, Intel® Boot Guard is available on Intel® Xeon® Scalable platform and Intel® Xeon® D processor-based server products ,which brings the architectural innovations of the Intel® Xeon® Scalable platform to a system-on-a-chip (SoC) processor for lower-power, high-density solutions, integrating essential network, security and acceleration capabilities. Intel® Boot Guard does not have any software requirements; it is enabled at the factory, and cannot be disabled.

Intel and the industry has started investigating technologies for Secure Boot, Attestation and Firmware Reslience covering verification of additional platform firmware. Technologies like Intel's Platform Firmware Resilience (PFR) [17], Cerberus* [17] from Open Compute Project and Microsoft*, and Google* Titan [16] are introducing an on-platform chip or an FPGA that can perform platform hardware RoT function. These technologies are currently in development and out of scope for this paper.

3 Intel® Boot Guard Basics

UEFI BIOS code execution is generally untethered to the underlying hardware, which means this UEFI BIOS code runs without being verified or measured. Hence, this makes the entire boot process vulnerable to subversion of the BIOS, whether that can happen through an unprotected update process or simple hardware attacks using SPI flash memory replacement or using a Dediprog*.

Intel® Boot Guard provides robust hardware-enforced boot policy controls to platform manufacturers and platform owners to authorize which BIOS code is allowed to run on that platform. Intel® Boot Guard provides that hardware based Root-of-Trust (RoT) for platform boot verification, which is responsible for verifying the BIOS image prior to BIOS execution. Intel® Boot Guard raises the security bar of the platform, reducing the above attack vectors and making it harder to launch attacks to subvert the boot process. See [14] in Table 2 for an example of an OEM incorporating Intel® Boot Guard into its Platform Security solution.

4 UEFI Secure Boot Basics

This section gives an overview of UEFI Secure Boot architecture components, defined in the UEFI Specification [2]. In a nutshell, UEFI uses the cryptographic signature verification mechanisms to first verify and then install the verified firmware components. The Platform Owner or OEM creates a cryptographic hash of the firmware component image, and then signs this hash with their firmware signing private key. For example, a Platform Owner could create a cryptographic hash using the SHA-256 algorithm and then sign the hash using the RSA PKCS#1 signature standard.

Figure 4 shows that the cryptographic hash(es) of the Public Key(s) corresponding to the signature Private Keys are securely made available as Certificates (step 1) to the firmware image for verification (steps 2A, 2B, and 2C). The UEFI Option ROMs and the OS Boot Loader are verified using the provisioned certificates.

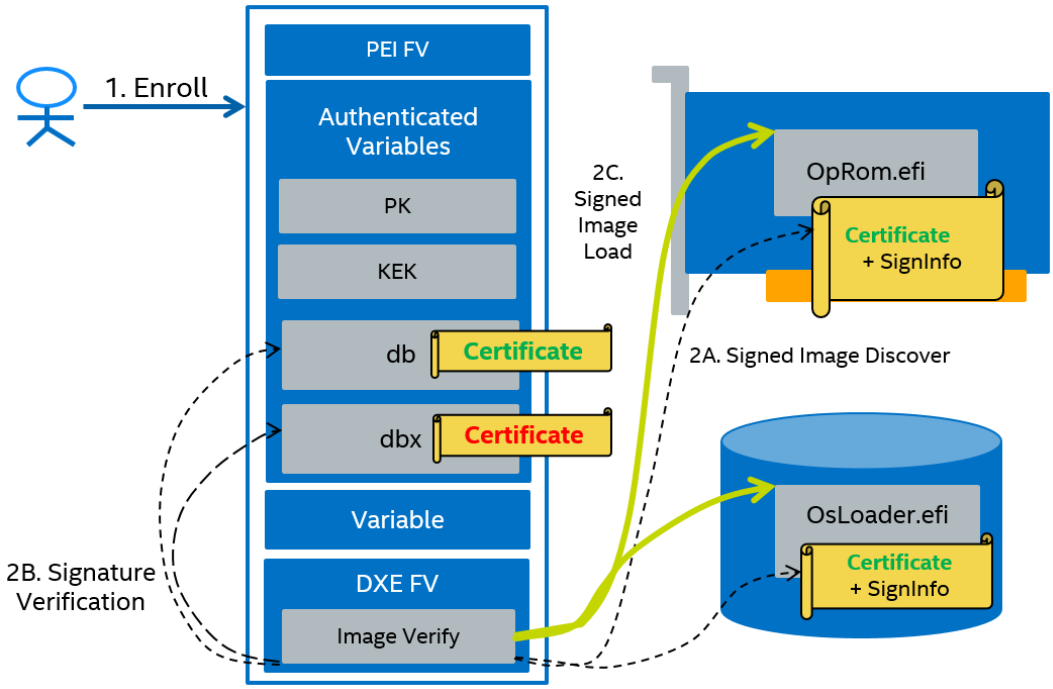

Figure 4. UEFI Secure Boot Components

The UEFI Secure Boot will stop the platform boot if signatures are not valid. In such cases, the remediation or recovery mechanisms need to be built as per the platform security policy. For instance, in some cases for Comms devices, the Platform Owners prefer a "dead" platform when the Intel® Boot Guard or UEFI detects signature verification failure. In other cases, the platforms would implement an OEM provided secure recovery procedure or remediate the platforms to a remediation network for further analysis.

4.1 Authenticated Variables and Key Provisioning

Table 4 summarizes the UEFI secure credentials and use.

Table 4. UEFI Authenticated Variables

Credential	Purpose	Provisioning Entity
PK	Platform Key – Root key set to enable Secure Boot	Platform Manufacturer
KEK	Key Exchange Key. - List of certificate owners with db, dbx update privilege.	OS Partner (Optional: OEM Key)
db	Authorized Database. - List of Allowed Driver or App. Signers (or hashes).	OS Partner, or CA (Optional: OEM Appl. Key)
dbx	Exclusion Database. - List of Revoked Signers (or hashes).	Signing Authority
SetupMode	1= in Setup Mode, 0 = PK is Set (User Mode)	-
SecureBoot	1 = Secure Boot in force	-

The owner of the certificate in KEK can update the DB and DBX, which is usually the OSV. The owner of the certificate in PK can update the KEK. If the image is signed by a key in DB, then it is allowed to run. If the image is signed by a key in DBX, then it is forbidden to run.

4.2 Secure Firmware Update

Once the platform is up and running, it is a good operational security practice to securely update the platform with the latest CPU patches, firmware, OS, and driver revisions. Often, these updates are driven by security fixes in prior and existing deployed versions. Hence, system administrators should require that these updates adhere to the commensurate security level and trust policies that were applied to the system boot.

At a minimum, all firmware and OS updates should follow a secure firmware update process, which includes that the update images are verified prior to install. This should also include checking for Security Version information to ensure that the attacker is not reverting back to an older vulnerable image. This is done with the Anti-rollback feature supported by Intel and OEM firmware.

4.3 Firmware Recovery

Intel® Boot Guard and UEFI Secure Boot enforce a strict security policy of installing only the authorized images that pass the cryptographic signature verification checks as per the security policy. It is quite possible that a signature failure, possibly due to operational error or security attack, may occur, which leaves the platform in a cold state. In essence, the platform is unbootable using the currently available firmware and OS images.

OEMs and System Owners have deployed measures for reliably detecting such situations and performing recovery. Intel® Boot Guard and UEFI Secure Boot specifications do not define Recovery procedures, which are largely deployment and system-specific. In some cases like Routers, sensitive Base Stations, and others, the security policy may not allow any recovery procedures. Firmware boot failures are considered catastrophic and hence, must be security evaluated. In some other cases, for instance in Cloud and controlled deployments, secure recovery procedures may be instituted via Out Of Band or in band mechanisms. Dell* describes a mechanism in [11].

5 UEFI Secure Boot: Operating System Enhancements

This section covers various techniques Operating Systems use to improve the security posture on a platform when they detect UEFI Secure Boot is used to boot the platform.

5.1 Signed Kernel Modules

As described in Section 4, UEFI Secure Boot ensures that OS boot loaders are signed appropriately prior to execution. OEMs by default ship their production platforms with Microsoft* UEFI Certificates already installed. Linux* distributions (such as Red Hat*, Ubuntu*, and SUSE*) make use of this capability by shipping their first stage OS loader, called Shim which is signed by the Microsoft UEFI Certificate Authority. Inside their Shim, they have their own UEFI key, which they use to sign their own next stage OS loader, predominantly Grub2. This means the OSVs can be in total control of the Grub2 (bug fixing, feature updating, etc.) without having to re-sign it with the Microsoft UEFI Certificate Authority for every change.

Figure 5 shows how this is done for Linux.

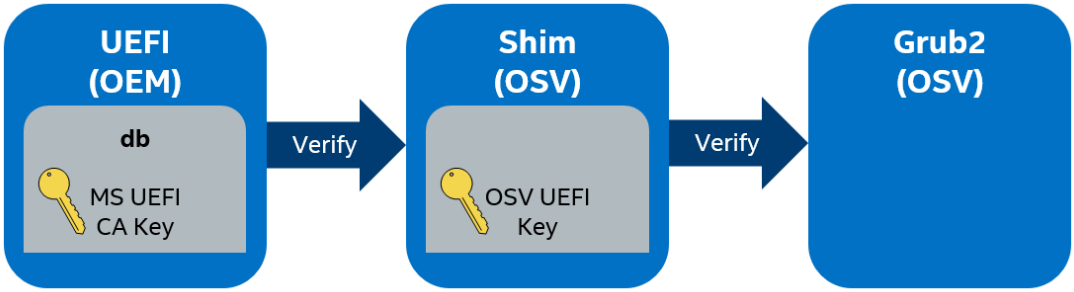

Figure 5. Shim Certificate Signing

Figure 6 shows how the OSVs extend this trust boot chain to the OS kernel and drivers.

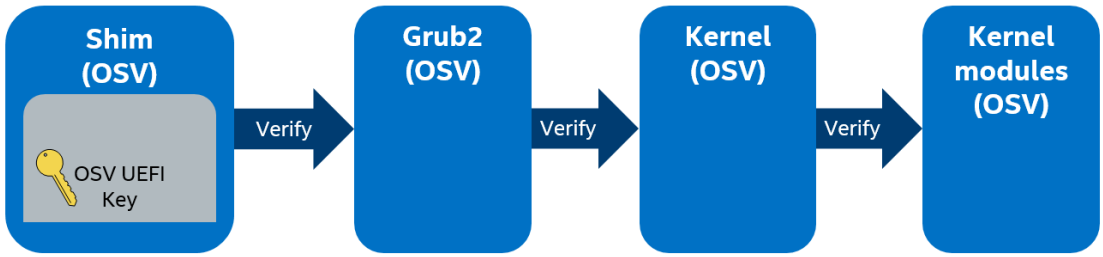

Figure 6. OSV Secure Boot Chain of Trust

The UEFI aware OSes continue this sign verification trust into any kernel drivers that may be installed after the system OS boots. This ensures that the system software remains trusted. The grub2, kernel, and kernel module components of this chain of trust will not be loaded unless they are signed appropriately by the OSV. All of the aforementioned Linux OSVs support UEFI Secure Boot out-of-the-box and they include all these properly signed components as part of their standard distributions [8, 9, 12]. Refer to Table 2.

5.2 Machine Owner Keys

Figure 7 illustrates how the OSVs also cater for scenarios whereby the user, or machine owner, may want to create their own version of grub2 or use their own modified kernel or third-party/out-of-tree kernel drivers. This capability is provided via the MOK (Machine Owner Key) List feature of the Shim. It enables the machine owner to use their own key and enroll it into the Shim for subsequent use when verifying images (OS loader, kernel, kernel modules).

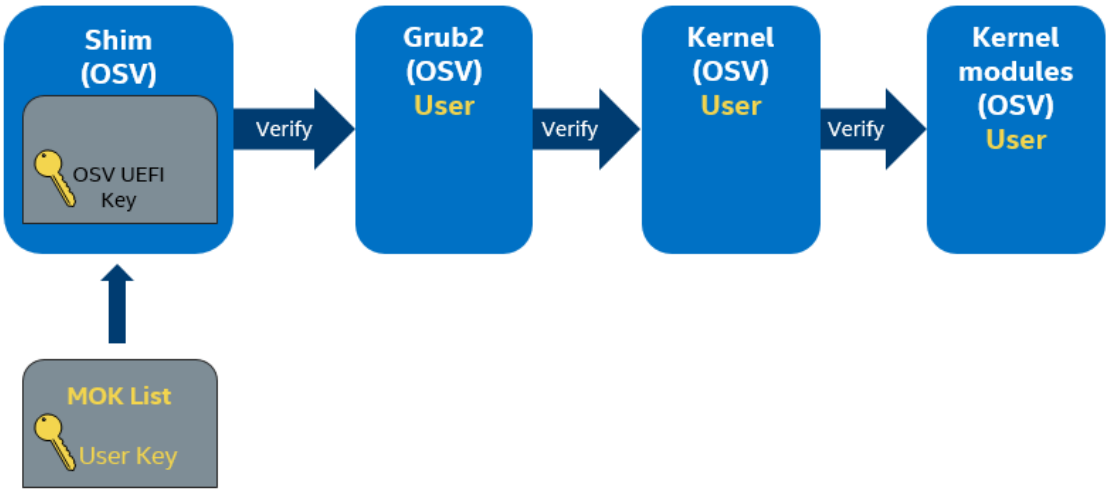

Figure 7. MOK and User Signed Images

See references [24, 25] for examples of how Linux OSVs use the MOK utilities to sign third-party and out-of-tree kernel modules/drivers.

5.3 Kernel Security Enhancements

On platforms with UEFI Secure Boot enabled, Linux OSVs have taken extra steps to ensure the OS is not compromised by applying a set of kernel lockdown patches. The main principle of the security patches are to disable the use of mechanisms that could be used to modify the resident kernel with unauthenticated code. The steps taken include the following:

- User mode direct hardware access is not allowed. This prevents access to PCI resources, DMA, and read-write operations to ports.
- Write access to `/dev/mem/` and `/dev/kmem/` is not allowed.
- Hibernation and the user space software suspend interface are disabled.
- MSR Writes are disabled.

Intel strongly recommends that once the platform is up and running, it is a good operational security practice to securely update the platform with the latest CPU patches, firmware, OS, and driver revisions.

5.4 Enhanced Platform Awareness (EPA) for Secure Boot

In orchestration environments such as OpenStack* and Kubernetes*, the discovery of compute node capabilities has seen significant research and development. The detection of a compute node's platform security posture is one area where this extra platform information can assist in the decision making process of where to deploy trusted workloads.

In Kubernetes, there has been no way to identify hardware capabilities or the ability for a workload to request certain hardware features. The Node Feature Discovery capability was created to bring Enhanced Platform Awareness (EPA) to Kubernetes [21].

EPA has been updated to detect if a compute node has been booted with Secure Boot. It can now detect if Intel® Boot Guard is enabled and if UEFI Secure Boot is enabled on a running node [22].

6 Networking Workloads on UEFI Secure Boot Platforms

This section describes the impacts and mitigations for certain networking workloads when executed on platforms with UEFI Secure Boot enabled.

6.1 Networking Workload Impact

Figure 8 provides an illustration of sample networking workloads that use ingredients that are impacted by the kernel security enhancements described in the previous section.

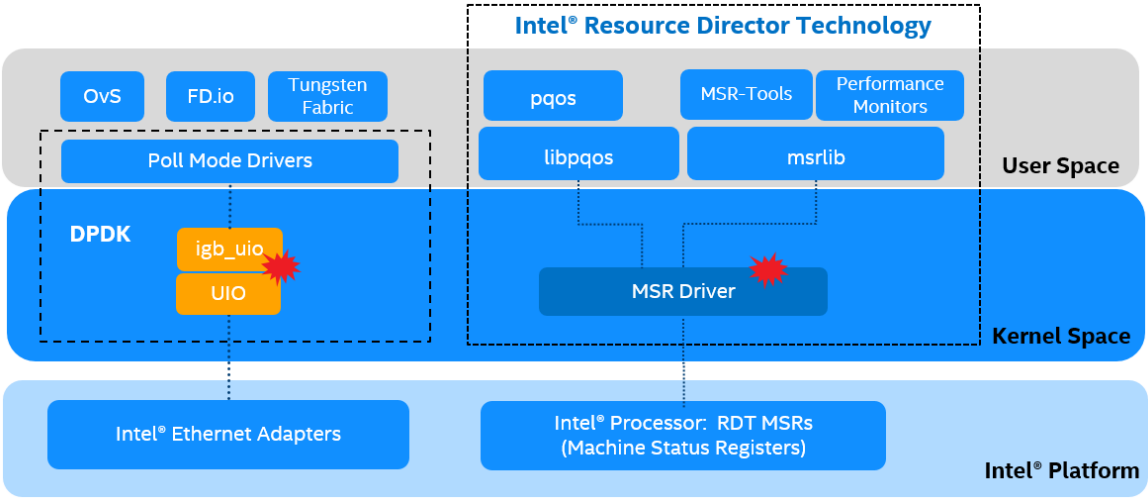

Figure 8. UEFI Secure Boot Impact on Networking Workloads

Applications that use DPDK, e.g. OVS, VPP, are impacted when DPDK is configured to use the igb_uio and supporting UIO kernel module to map NIC resources into user space. The UIO kernel module is prevented from performing the mapping operation when the kernel detects, at runtime, that UEFI Secure Boot is enabled.

Similarly, the Intel® Resource Director Technology (Intel® RDT) based platform quality of service (pQoS) application and supporting library is prevented from *writing* to MSRs using the standard MSR module within the kernel.

6.2 Networking Workload Mitigations for Secure Boot

Figure 9 illustrates the mitigations possible for impacted networking workloads.

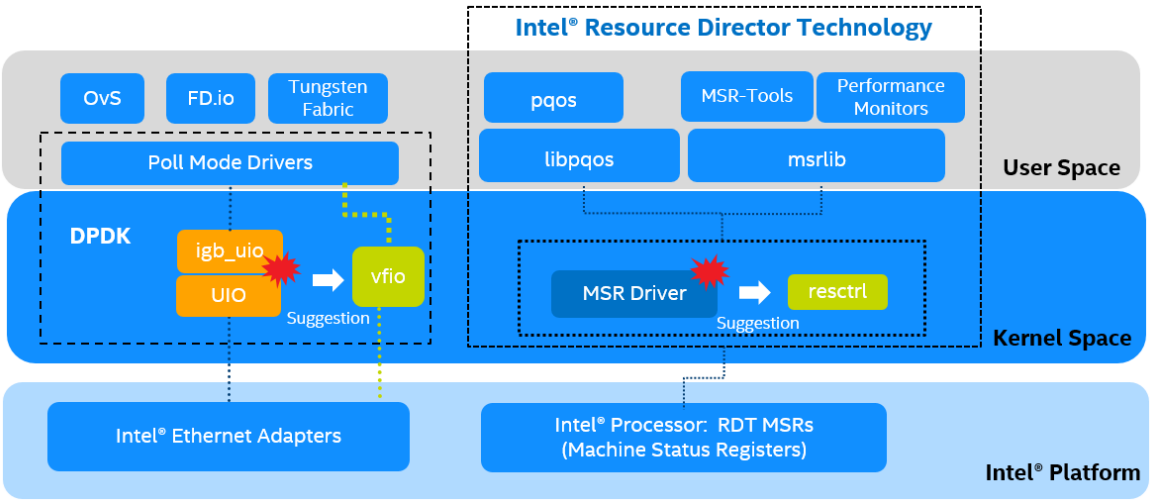

Figure 9. UEFI Secure Boot Mitigations

Applications using DPDK can be configured to use the VFIO kernel module. In fact, this is the recommended configuration promoted by many in the industry as the most secure, since it makes use of the protection provided by the IOMMU [18]. VFIO has been supported in the Linux kernel since version 3.6, and has been introduced to DPDK since version 1.7.

The Intel® RDT pQoS application and libraries are now built with the ability to use multiple mechanisms to configure MSRs. As well as using the impacted MSR kernel module, it can work with the resctrl filesystem module, which is now an upstreamed kernel module and supported by all major Linux OSVs.

7 Summary and Call to Action

Applying best practice security methodologies such as creating a hardware based Root of Trust (RoT) is an effective way of protecting network infrastructure against ever increasing cyber-attacks. The objective of this paper is to share best practices as recommended by Intel, specifically in the use of Intel® Boot Guard technology and UEFI Secure Boot to create secure networking platforms for NFV, 5G, Edge, and Comms systems. Intel has also worked with industry leaders (OEM/OSVs) to create an ecosystem that supports Secure Boot. In addition to taking these measures at the platform level, Intel also strongly recommends that customers implement a Security Strategy at the network orchestration level. Using these proven technologies will be critical in protecting data and IP in a Secure Network Infrastructure.

For any further questions, please contact your Intel support team.

0619/DN/PTI/PDF

338941-002US