ONLOGIC

OnLogic Product Security

Protecting Your Edge Devices with a Custom Secure Boot Policy

ONLOGIC

Table of Contents

Executive Summary

A system deployed at the edge is typically equipped with threat monitoring tools such as antivirus software and Intrusion Detection Systems (IDS). While these tools actively monitor what is running on the system, the environment prior to the Operating System (OS) remains unmonitored. This pre-OS environment is appealing for malicious software as this lack of monitoring tools allows malware to target and install malicious bootloaders or boot an entirely different OS altogether. This can result in a production environment system becoming compromised.

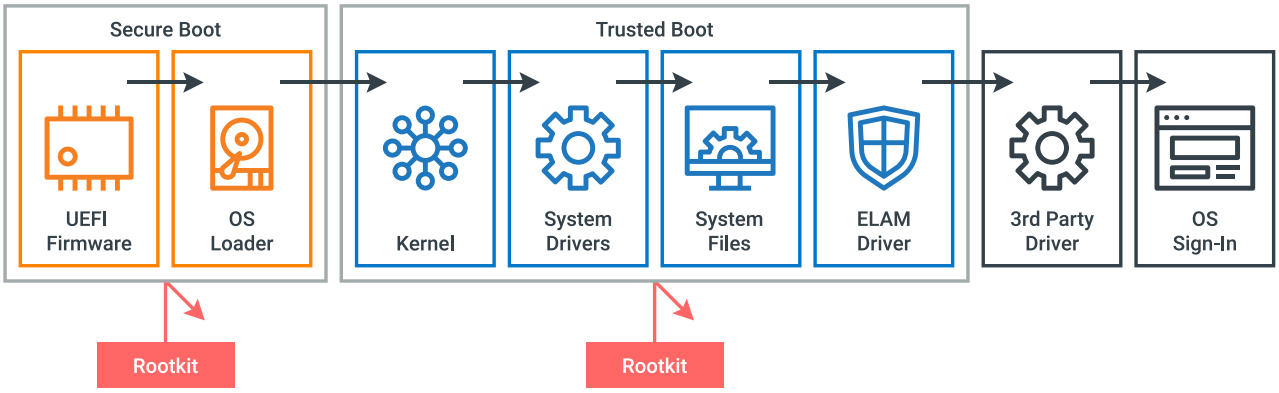
This is where Secure Boot, a security feature found in Unified Extensible Firmware Interfaces (UEFIs), can protect the early stages of a system's boot process. Enabling Secure Boot ensures your system's UEFI will only hand over control to trusted and verified bootloaders and executables to mitigate these pre-OS threats.

Keep reading to learn the benefits of customizing the Secure Boot policy and the steps to deploy it.

Introduction to UEFI Secure Boot

Secure Boot is a UEFI security feature that uses public-key cryptography, also known as asymmetric cryptography, principles to control what is allowed and not allowed to execute during the boot phase and even within the Operating System (such as device drivers.) This secures the early stages of the boot process of a system by preventing modified or undesired bootloaders from executing.

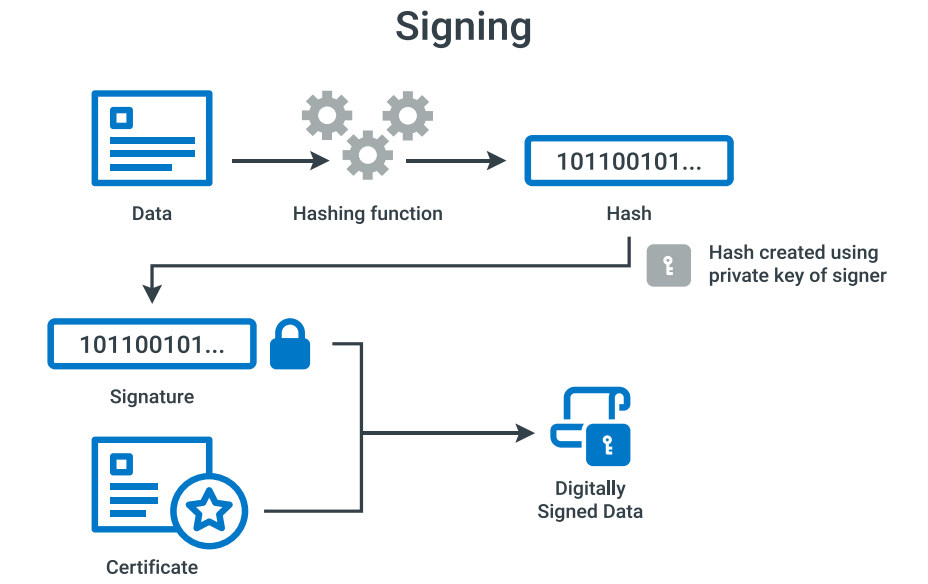
These kinds of early boot threats exist in the form of malware installing malicious bootloaders / bootkits, or Operating Systems booted from a USB by a threat actor. The challenge with preventing these kinds of attacks is the lack of protections typically found in an Operating System since they operate in the environment prior to any OS kernel being loaded. Here is where the Secure Boot policy of the UEFI comes into play.

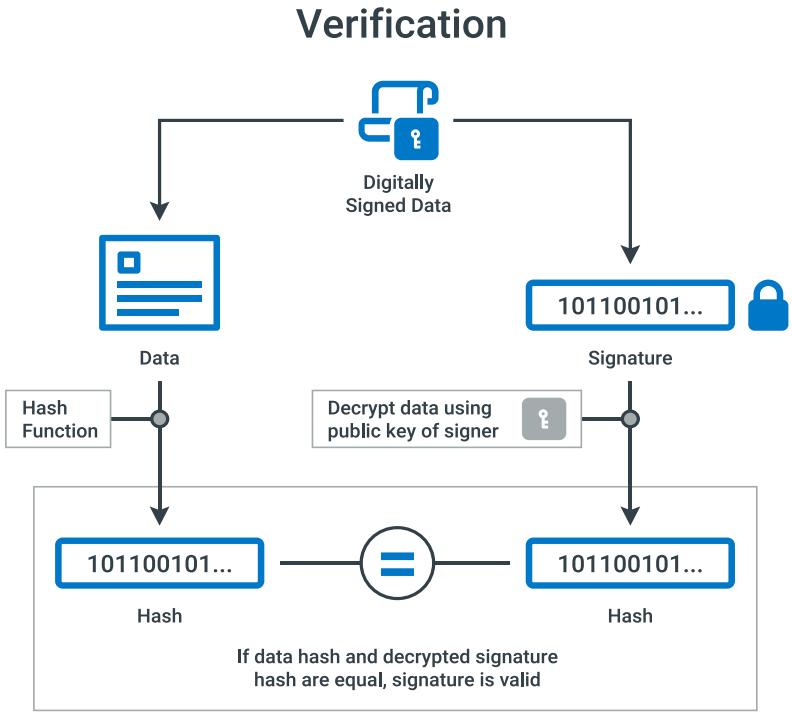


The areas of the boot process where the Secure Boot policy operates.

Asymmetric cryptography overview

The two main components in Secure Boot policy are the certificates that get enrolled in the UEFI and the signature that is appended to the end of the bootloader or binary. The certificate is the “public” half of the cryptography system that is meant to be shared out freely and will handle authorization of bootloaders based on the bootloader’s embedded signature. The embedded signature is generated using the private key of the respective certificate and then appended to the end of the target binary. A private key is a sensitive security asset that should not be shared out and its access should be heavily restricted to authorized personnel only.





Understanding UEFI Secure Boot repositories and their roles

The Secure Boot policy is more than just an enable/disable setting in the UEFI. It is a combination of different certificates residing in their own respective repositories. It can be challenging to keep track of the roles and responsibilities of each certificate, but they can be categorized as “administrative” or “operational” functions of the Secure Boot policy. The db and dbx can be viewed as the operational repositories as they are responsible for preventing or allowing executables to run on the system. If a Secure Boot violation occurs, the db and dbx should be investigated first to verify the expected certificates are enrolled for the respective signatures embedded in the executable attempting to run. The PK and KEK are the administrative repositories as their certificates are used to authenticate updates to the KEK, db, and dbx. Below is more detail about each of these repositories.

Allow list (db)

The allow list repository, commonly referred to as “db” in the Secure Boot policy, can contain multiple certificates or hashes of binaries. When the UEFI is attempting to run a bootloader, it will check the embedded signature of this bootloader against items in the db repository. If one of the embedded signatures of the bootloader satisfies any challenge from certificates or hashes in db, the UEFI will allow it to execute and hand control over to this bootloader to then run its subsequent steps in the boot process.

It’s important to note that a bootloader or binary can contain multiple signatures. Meaning that a single bootloader can contain signatures that satisfy multiple Secure Boot policies across different systems. This could be leveraged for technicians that may boot a USB drive containing multiple diagnostic tools for troubleshooting different systems in an operational environment that may have Secure Boot policies with different certificates.

Disallow list (dbx)

The disallow list repository, referred to as the “dbx”, contains the same contents as the db but operates in the opposite fashion. Any bootloader or binary that is signed by a certificate or matches a hash in the dbx repository will be blocked from executing. The dbx can be thought of as a revocation list. This allows system administrators to rotate certificates when current ones expire and then prevent any bootloaders signed with the old certificate to be blocked from execution on the system.

Key Exchange Key (KEK)

With the db and dbx being in charge of permitting bootloaders and binaries to run, the UEFI needs a method for restricting changes to the db and dbx repositories to only allow authorized updates. Without this kind of control, any foreign certificate can be added into the db and dbx repositories, and therefore any bootloader can be signed by a foreign private key.

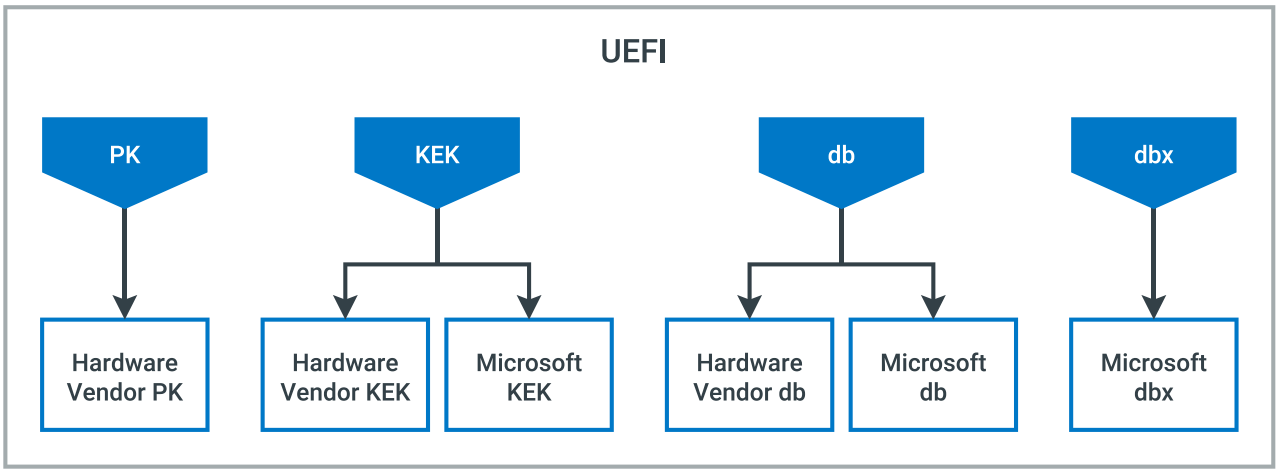
This is where the Key Exchange Key (KEK) repository comes into play. The KEK is responsible for permitting certificates with the appropriate signature to be added into the db and dbx repositories. Certificates destined for the db and dbx must contain the private key signature of a certificate in the KEK to be enrolled. With the implementation of multiple certificates in the KEK, different entities can sign updates destined for the db and dbx repositories.

Platform Key (PK)

Lastly, updates to the KEK repository must contain a signature from the private key of the single certificate in the Platform Key (PK) repository. This PK certificate is typically the hardware owner and acts as the top-level authority in administering the secure boot policy. Due to this responsibility of the PK, there can only ever be one PK certificate enrolled in the UEFI.

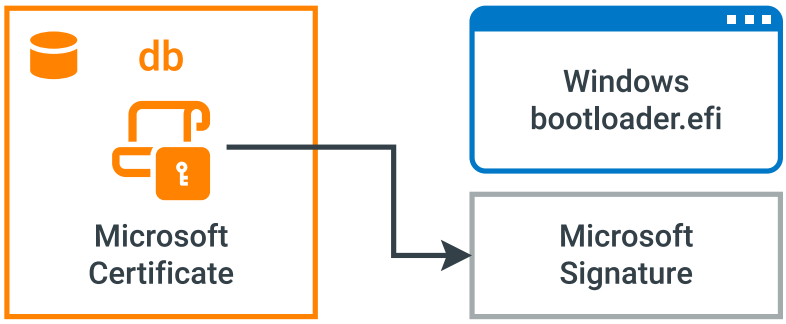
In most modern UEFIs, the Secure Boot repositories will ship with the following:

- **PK:** Motherboard Vendor's Platform Key certificate
- **KEK:** Motherboard Vendor's Key Exchange Key Certificate, Microsoft's Key Exchange Key Certificate
- **db:** Microsoft's db certificate, sometimes the Motherboard Vendor's db certificate
- **dbx:** Microsoft's list of disallowed certificates and hashes



The presence of Microsoft's certificates allows Microsoft to sign the Windows bootloader and subsequent boot chain items. Microsoft has also signed bootloaders for the Linux boot process, which allows for Secure Boot to be used with most major Linux distributions. Only components signed by one of these certificate owners will be allowed to boot on the system. This configuration works in consumer environments where nothing custom is being booted on the system. But customization of this Secure Boot policy has its benefits to IT professionals who want to mitigate threats operating in the early phases of a system's boot process by enforcing a check on all bootable executables to verify their permission to run on the system.

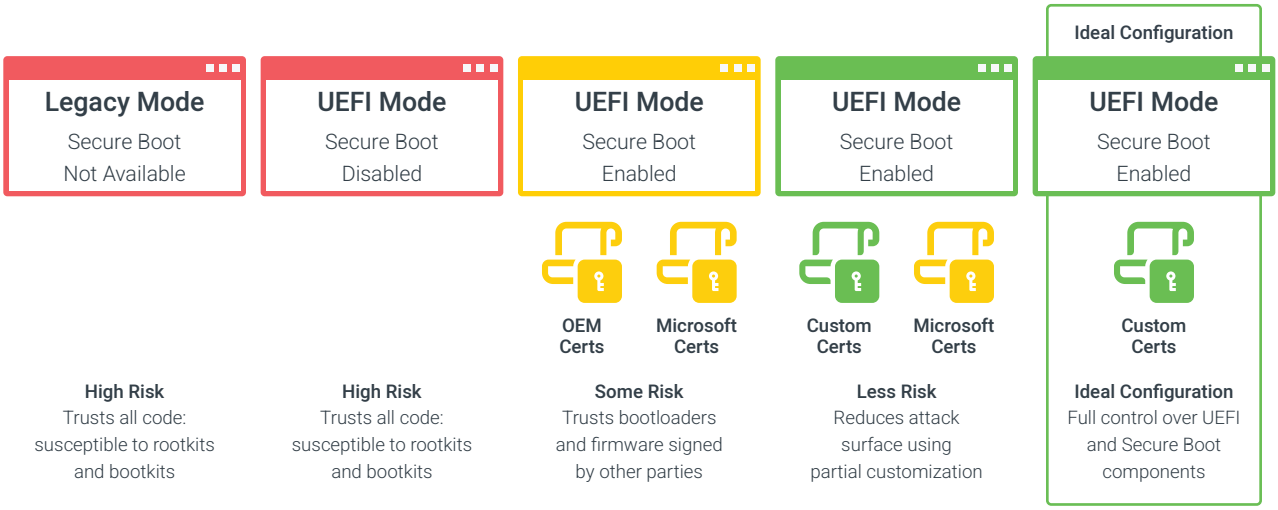
This effectively creates a “whitelist” of binaries and bootloaders permitted to run on the system while blocking everything else not explicitly in the whitelist.



Using the Microsoft signed bootloader to demonstrate embedded signature. This signature can be validated by the respective certificate in db.

Importance of customizing UEFI Secure Boot

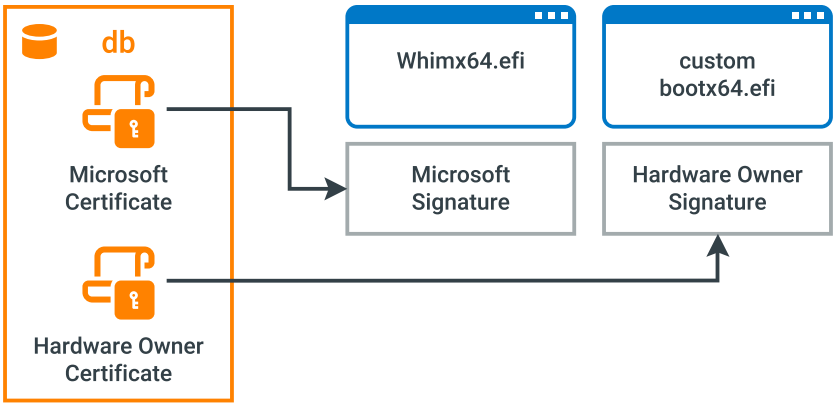
With the typical UEFI containing the motherboard vendor’s and Microsoft’s certificates, the average user will never notice any Secure Boot violation as long as they’re booting Windows or Linux OS (such as Ubuntu). However, a custom live USB, custom Linux OS, or custom EFI shell application that doesn’t contain an embedded signature from Microsoft or the hardware vendor will be prevented from running.



The different levels of Secure Boot policies.

This configuration alone however does little to prevent tampering. Someone with physical access to the system could boot anything signed by Microsoft or the motherboard vendor such as their own Windows installer or a live Linux image. From there the system could be maliciously re-imaged or wiped.

This is where customizing the Secure Boot policy, in addition to password protecting the UEFI, is beneficial. Secure Boot customization grants system administrators the ability to sign components and authorize their execution on the target system. For example, if field technicians have a bootable USB with diagnostic and troubleshooting tools, the bootloader for this USB can be "remotely" signed by the respective IT department that controls the Secure Boot policy of the end device. This will restrict the UEFI to only allow these signed components to boot on the system.



Example of how multiple certificates can authorize embedded signatures of different bootloaders

Customization options

This section will outline the high-level process overview for customizing Secure Boot. This begins with the creation of your own keys and certificates. The most commonly used tools for this are openssl for Linux and MakeCert for Windows. These tools will be used to generate the PK, KEK, and db public and private key pairs and also sign the certificates themselves. Once generated, the workflow for enrolling the certificates should be:

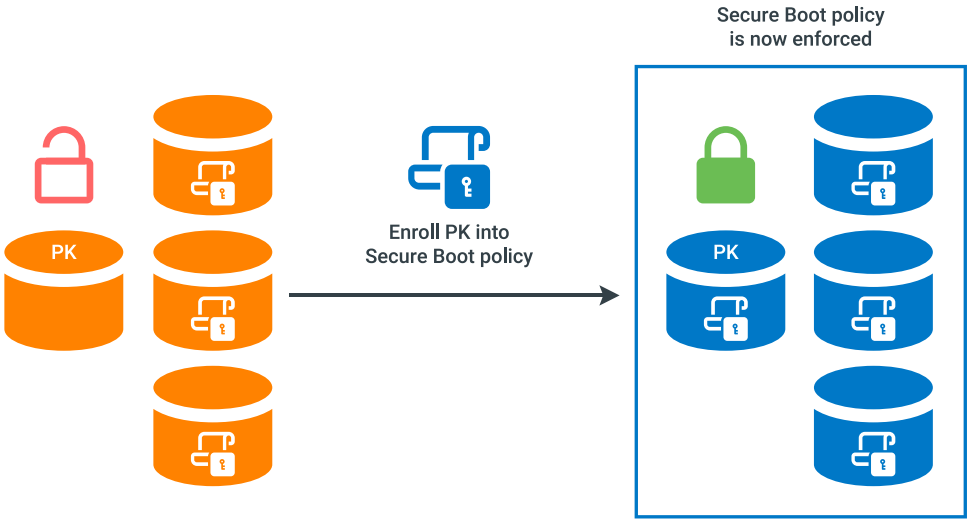
1. Add the KEK signed db and dbx certificates first
2. Add the PK signed KEK certificates second
3. Add the PK signed PK certificate

The keys and certificates should not be created on the system(s) that will ultimately enroll the signed certificates into the UEFI. All keys and certificates should be created on a server or system within the organization's managed IT infrastructure to mitigate any risk of the private keys being intercepted. Ideally there is a dedicated server for creating and signing Secure Boot components with access restricted to authorized users.

Once all Secure Boot assets have been created, the public key Secure Boot assets can be freely distributed and used within and outside an organization. Target Bootloaders, binaries, or 3rd party generated certificates can be received and uploaded to the signing server where they can be signed by the respective private keys and then sent back for use by the 3rd party.

Once the PK is enrolled, the UEFI's Secure Boot policy will change from Setup Mode to User Mode. User Mode will only allow signed updates to the PK, KEK, db, and dbx UEFI repositories to occur.

Secure Boot can be configured through the UEFI interface or from user space where the Secure Boot policy and target binaries can be managed with command line tools such as sbctl or efitools for Linux and Powershell commands for Windows. With the ability to update the Secure Boot policy from a remote shell, a fleet of systems can have their Secure Boot policies updated through a shell script given the updates have been signed by the expected private keys.



The Secure Boot policy will not be enforced until a certificate is enrolled into the PK repository.

Alternatively, the UEFI itself can be customized through collaboration with the hardware manufacturer to create a custom UEFI binary. This customized UEFI binary can contain the setting changes and customized certificates required to begin enforcing the custom Secure Boot policy. This allows for a standard UEFI update to automatically handle all the enrollment and enablement steps done through the demonstrated manual process.

Implementation guide

The first step in customizing Secure Boot is to generate the keys and certificates. These private keys and certificates can then be used to administer the Secure Boot policy on target systems through multiple enrollment methods. These enrollment methods all achieve the same goal, but work with different operational workflows. Each UEFI implements its Secure Boot administration differently. The system manual or hardware vendor will have steps for all the supported update methods and certificate formats.

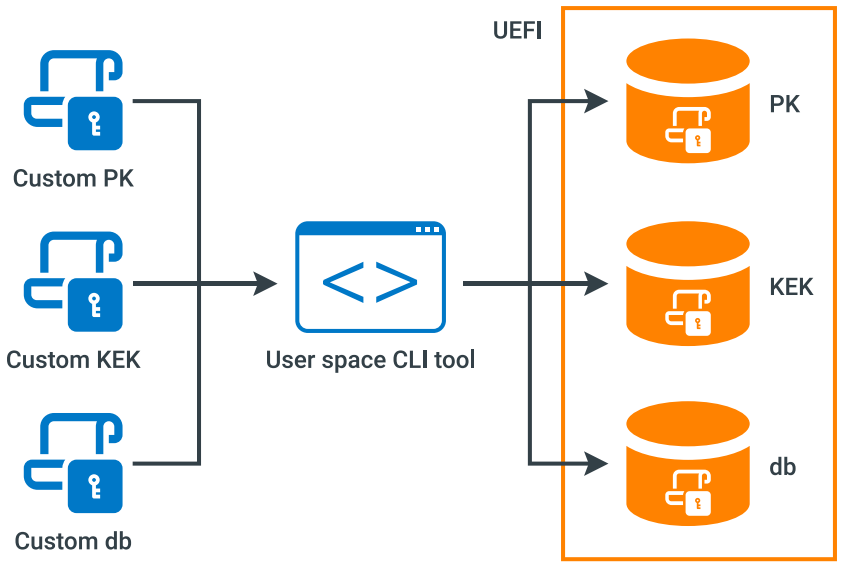
Manually enrolling certificates through the UEFI menu

The UEFI menu will have a page for adding, removing, and resetting the Secure Boot policy. The certificates generated can be placed on a storage device like a USB stick and inserted into the target system. While the specifics of the update process will differ between UEFIs, the high-level workflow with the UEFI is:

4. Select target repository to update (PK, KEK, db, dbx)
5. Navigate to respective certificate on USB
6. Enroll target certificate into repository

Manually enrolling certificates through the operating system

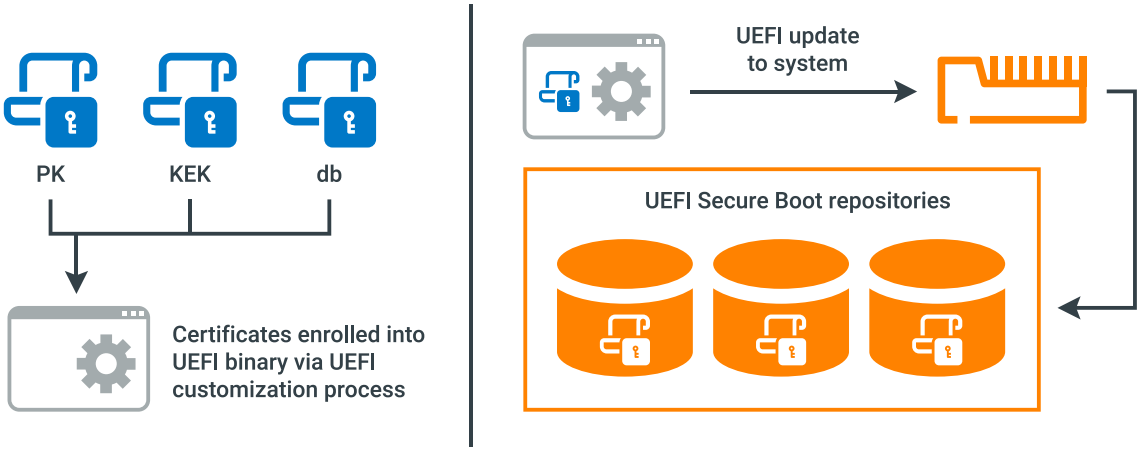
Command line interface tools can be used to enroll target certificates into the UEFI from userspace. This can be implemented through update scripts so the Secure Boot policy can be administered through an automated update process.



Example of a Command Line Interface tool enrolling public certificates into the UEFI's Secure Boot policy.

Custom UEFI containing certificates

The hardware vendor may be able to support the enrollment of the custom certificates into the UEFI binary for automatic enrollment and enablement of the customized Secure Boot policy. This can be implemented through a normal UEFI update to the system(s), cutting out the manual steps of enrolling the certificates through the UEFI menu or OS command line.



Enrolling custom Secure Boot certificates via a UEFI update.

Challenges and considerations

Before deploying a Secure Boot enabled system into your operational environment, processes for maintaining a system with Secure Boot should be identified. The Secure Boot policy can add complications to a normal IT administration workflow. For example, a Secure Boot policy that contains Microsoft certificates allows Windows to perform updates to the OS because the updates themselves are signed by Microsoft's private key. This means the updates to the OS through a system's lifecycle don't result in a Secure Boot policy violation. However, if your OS contains a custom Linux kernel that is signed by one of your db certificates, any kernel update processes will need to include a step for signing the new updated kernel before implementing into a production environment.

Any existing processes for handling digital IT assets should be extended to incorporate Secure Boot keys and certificates. There should also be processes in place for managing these assets through their lifecycle such as key rotation plans, revocation processes that include updates to deployed systems, and access control practices.

Additionally, the UEFI / Secure Boot policy should require a password for users to change any settings. This means that access to UEFI settings will be blocked for anyone troubleshooting a system's UEFI settings. It will be important to evaluate workflows for system lifecycle management to identify items that will need to be signed and workflows that may be hindered by the presence of a UEFI password. The reason a UEFI password is recommended is because the absence of a UEFI password effectively negates the Secure Boot policy as anyone wishing to get around the Secure Boot protections can simply disable Secure Boot.

As an alternative to a UEFI password, the Secure Boot policy can be enforced while hiding the UEFI settings from end users and redirecting the management of the Secure Boot policy through UEFI updates or user-space tools in the OS. Since each UEFI is different, the hardware vendor should be consulted for the feasibility of hiding the Secure Boot menu in the specific UEFI.

While the Secure Boot policy can protect the pre-OS boot environment of a system, the data that lives on a storage device is still exposed in the event that a storage device is permanently or temporarily removed from the system. Pairing Secure Boot with full disk encryption methods is highly recommended to mitigate the loss or interception of sensitive data within the system.

Each target system can have different Secure Boot policy controls available in both the UEFI and the OS. It's important to consult with the hardware or UEFI vendor to get a full understanding on what the specific system's Secure Boot management method is and which format of certificates are supported.

Full-customization and Microsoft Windows

If you plan on using a custom Secure Boot policy and a Windows OS, it's important to keep the Microsoft certificates in the UEFI so the policy can continue to support the Windows boot process and subsequent OS updates. Removal of the Microsoft certificates may render the OS unbootable.

Future trends and developments

Secure Boot is one of several important UEFI security features available today for protecting systems at the edge. Measured boot is another feature that allows a system's boot process and components to be measured and stored in the Trusted Platform Module (TPM). The items in the boot sequence are then checked against these measurements for each boot cycle. If the live measurement of a component in the boot processes is different from the stored measurement, it will be blocked from executing.

Conclusion

Secure Boot is a worthwhile security feature that reduces the attack surface of Operational Technology (OT) by monitoring and restricting execution to only authorized executables. This is achieved by blocking all pre-operating system executables that are not signed by an authoritative entity (typically the hardware owner). This effectively creates a "whitelist" of bootable media that can be updated from the OS.

These efficiencies in the customization lifecycle can be leveraged to create "automatic" updates through scripts using command-line interface (CLI) tools running on the target system, or by working with the motherboard vendor to create a custom UEFI with the custom certificates enrolled into the UEFI binary. Any organization that is concerned with the security of their systems in transit, or looking to protect Operational Technology from physical threats should consider the benefits of Secure Boot and its customization options.

If you're interested in taking advantage of the protections that Secure Boot offers, reach out to our technical sales team to find the best solution for your needs. Whether it's using the standard OnLogic Secure Boot policy, generating your own keys for a fully custom UEFI, OnLogic's Secure Boot solutions, or personalized consultation services, OnLogic can help you bring the security of your edge devices to the next level.

About the author

Vincent Flores is a Hardware Security Engineer and a member of the Product Security Team at OnLogic. Vincent has over a decade of experience with Industrial and Edge computers and prior to joining the Product Security team was an Applications Engineer, working directly with customers to deploy OnLogic devices. Vincent leverages his experience with customer use cases to identify and implement security solutions for OnLogic products.

Appendix

Further readings on Secure Boot customization

- NSA Cybersecurity Technology Report - Secure Boot Customization
- UEFI Specification - Secure Boot
- Managing EFI Boot Loaders for Linux: Controlling Secure Boot

Tools for managing Secure Boot

- sbctl - Linux tool for managing the secure boot policy

Computers you can depend on

Here at OnLogic, we create highly-configurable small form factor computers that thrive where others fail. We collaborate with innovative companies around the world, working together to solve their most complex technology challenges. Our consultative approach combined with a powerful online platform, our quick-turn production capabilities, and modular hardware design, can help make the impossible possible.

Industrial Automation | **Machine Learning** | **Edge Computing** | **Embedded Applications** | **Product Security**

Ready to get started? Contact us!

Our team of experienced hardware specialists are here to help you select and customize your ideal computing platform. Reach out for a free project consultation.

North America

Call: +1 (802) 861 2300
Email: info@onlogic.com
www.onlogic.com

Europe

Call: +31 88 5200 700
Email: info.eu@onlogic.com
www.onlogic.com/eu-en/