SAMSUNG

Technical White Paper

Samsung Galaxy Book: How the Knox security platform provides protection below the OS

Introduction

Security Below the OS

This document provides an overview of the Samsung Knox security platform on **Samsung Galaxy Book** laptop PCs, focusing on the unique advantages that differentiate it from other options in the PC market from a security standpoint. The contents of this white paper are designed for C-level executives and IT security professionals looking to evaluate Samsung Galaxy Books with the Knox security platform as a comprehensive enterprise laptop PC solution to deploy to their workforce.

Supported devices

Note that this white paper specifically describes the Knox security platform on supported Galaxy Book models listed below. The security features described here don't necessarily reflect the security features available on other Samsung Galaxy Book laptop models. Also note that the Knox security platform for Galaxy Book PCs is distinctly different from the Samsung Knox platform for Android.

The following laptop PC device models are protected by the Knox security platform:

- Galaxy Book5 Pro 360
- Galaxy Book4 Pro 360
- Galaxy Book4 Pro
- Galaxy Book4 Ultra

Protecting below the OS

Laptop PCs can be particularly challenging for IT admins to protect since the BIOS — the software that initializes the computer's hardware components and starts the Operating System (OS) — is often the target of attackers. The BIOS needs to be particularly secure from threats to its *data* and its *code*, as these are most vulnerable to hackers. The following describes the types of compromises that are present if the BIOS isn't properly protected, as well as their impacts on the BIOS:

Data compromises

A data compromise occurs when an attacker gains access to protected information like passwords, certificates, or encryption keys. During a normal boot sequence, the BIOS relies on the **Trusted Platform Module (TPM)** to encrypt sensitive data and fend off hacking attempts on the hardware. Windows features like **BitLocker** then uses the TPM to ensure that protected data is only available when trusted firmware has been loaded. If the BIOS is compromised, it may provide false information to the TPM and trick it into loading a malicious OS, thus giving an attacker backdoor access to protected enterprise data.

Code compromises

The integrity of the OS depends on the integrity of the BIOS, since it is ultimately trusted for loading and executing the code that starts the OS. If the BIOS is compromised, then the self-check code Windows uses to detect threats may also be compromised. Without properly functioning update-validation systems, the BIOS (which normally validates update packages) can update itself with malicious code, granting backdoor access to hackers and preventing itself from being removed. This could lead to the entire device being compromised unless an IT Admin intervenes.

Impacts of BIOS compromise

If a BIOS is compromised, the following may happen to either its data or code:

- Tampered UEFI data — The BIOS maintains most of its boot configuration data in UEFI variables stored in onboard flash memory. If a malicious root or admin process modifies one of these variables, then security-critical features such as Secure Boot can be disabled, thus letting the device potentially boot untrusted or malicious software.
- Corrupted firmware — BIOS code is also stored in flash memory, which may be accessible to compromised OS kernels or physical attackers looking to gain privilege or persistency. In the last several years, there have been multiple instances of malware adding malicious code to the BIOS which then caused malicious code to be run on the host OS.
- Exposed enterprise network — Physical attacks can disable Secure Boot settings or circumvent the TPM. This could result in the loading of unapproved OS images, and exfiltration of sensitive data. If an attacker does manage to boot a malicious OS image, they may abuse VPN policies to gain access to a corporate network.

Although these problems are persistent, they aren't new to IT security professionals. Various technologies have been developed to help mitigate these problems. For example, UEFI introduced standards to the Secure Boot process to ensure that code only loads after it's been verified. Other technologies like Intel Boot Guard extend the Secure Boot process by ensuring code verification begins with the CPU hardware itself. Though these technologies help to mitigate some of the BIOS compromise issues mentioned above, they also present several limitations. For example:

- While the UEFI Secure Boot process validates firmware code, it doesn't validate UEFI configuration data. This can lead to users and software processes making modifications to corrupt the BIOS settings. Once an attacker has access to BIOS settings, they could remove Secure Boot features altogether to allow the loading of malicious firmware code.
- UEFI is limited in the response options it provides when it detects security issues. This can be a problem in situations where the device can't silently handle security issues by itself. For example, if a compromise is detected at boot time, the user or admin may need to trigger incident response processes, which might require unified and flexible response mechanisms out-of-scope from UEFI.
- The existing Secure Boot process is focused solely on the *detection* and *prevention* of unsigned or revoked firmware being loaded. During a normal boot sequence, the BIOS is loaded in several stages, with each stage verifying the authenticity of the next. If a tampered boot stage

is detected anywhere in the process, the BIOS would simply stop booting, and likely require administrator intervention – costing valuable time and resources – to be repaired.

With all these limitations in place, IT admins and executives need a more secure and robust solution to protect their enterprise from the growing number of threats below the OS. A solution that guarantees data and code protection before the OS ever boots up. This is why we built the Samsung Galaxy Book with the Knox security platform, starting with the Samsung Galaxy Book4 — our most secure and tamper-resistant laptop to date.

Introducing Samsung Galaxy Books with the Knox security platform

Samsung Galaxy Books with the Knox security platform were designed to provide best-in-class hardware-backed security to meet the needs of enterprise users. They pair industry-wide standards with powerful OEM features to provide high levels of firmware protection and robust firmware recovery capabilities.

Every Galaxy Book with the Knox security platform comes equipped with our **Secure Embedded Processor (SecEP)**, a standalone processor that validates firmware *before* the CPU ever powers up. This processor — along with its memory (RAM) and dedicated storage unit — are all isolated from the CPU in order to protect against compromises to the OS, hypervisor, and BIOS.

While the BIOS takes advantage of industry-standard security features like **Intel Boot Guard** and **UEFI Secure Boot**, the SecEP brings additional robust security features to help further protect the firmware. For example:

- Intel Boot Guard ensures that only an OEM-approved BIOS may run on the CPU. The SecEP takes security one step further by ensuring that only approved versions of the BIOS may modify any boot configuration data, and that a backup copy of the BIOS firmware is always available even if the original is corrupted.
- UEFI ensures that all firmware modules are verified by hash and validated against a binary revocation list. This means that only privileged processes have the ability to modify boot configuration data. The SecEP further enhances this process by making the revocation lists and boot configuration data tamper-resilient. It works with the BIOS to ensure that critical boot parameters only get modified by an authenticated user or admin through the BIOS setup utility. If UEFI detects an attempt to load suspicious firmware, the SecEP's **Tamper Alert** feature can be configured to require either user or admin confirmation before booting.

By introducing the SecEP, admins gain more awareness and control of common security compromises. Critical features like auto-recovery, data protection, and tamper alert are deeply integrated with the BIOS, but also completely isolated from the OS. This helps to ensure that an attack on the BIOS gets handled *before* firmware ever gets loaded.

Throughout the rest of this document, we'll describe the various hardware-backed security features and enhancements introduced by the Samsung Knox platform. Join us, as we showcase our best-in-class, most secure and protected laptop PC yet.

Hardware-rooted trust

In typical laptop PCs, the motherboard normally provides power directly to the CPU during the boot sequence. Once powered up, the CPU would then access its dedicated SPI flash storage to load and run the BIOS.

On Galaxy Books with the Knox security platform, the boot sequence begins with the SecEP. When the motherboard first powers up, it initializes the SecEP. The SecEP is responsible for providing power to the CPU. Before the SecEP powers on the CPU, it first verifies whether or not the BIOS is corrupted. If the BIOS is safe and corruption-free, the SecEP then powers on the CPU, and the CPU accesses SPI flash storage (to load the BIOS).

These layers of protection are put in place to prevent the device from loading and executing corrupt firmware and compromising the system. **Figure 2** further describes how the SecEP protects the CPU during the boot sequence.

Throughout this chapter, we'll explain the various security features used by the SecEP to provide a robust, hardware-backed system of trust to help protect the Galaxy Book from threats and compromises.

SecEP FW self-validation

The SecEP prevents the CPU from accessing the SPI Flash Storage until permission is granted. However, before that permission is even granted the SecEP goes through a series of self-validation checks to ensure that its own firmware hasn't been compromised.

Figure 1 describes how the SecEP performs this FW self-validation:

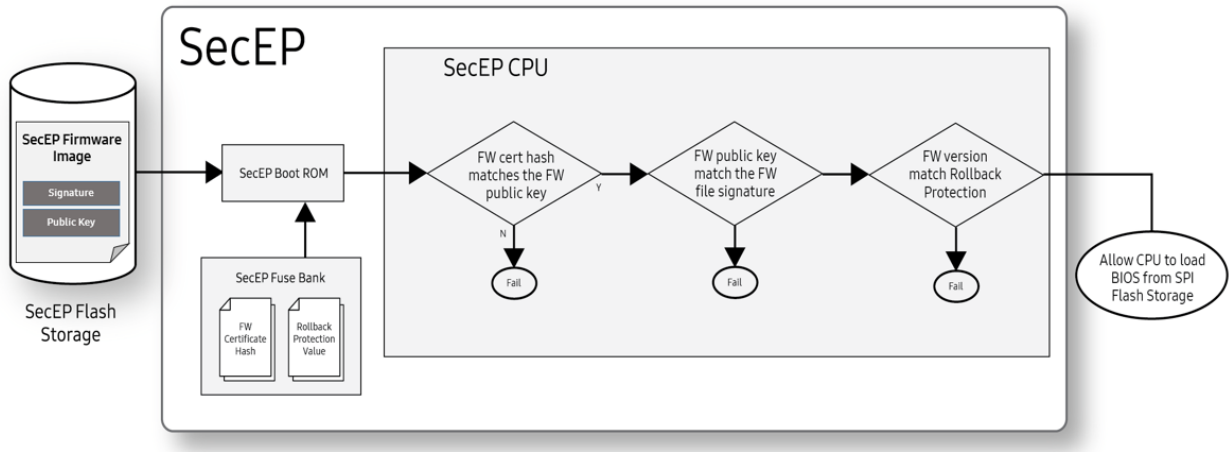

Figure 1: SecEP FW self-validation

- The SecEP FW self-validation first begins with a firmware hash check. This happens when the **SecEP Boot ROM** retrieves the **FW Certificate Hash** (SHA384) – generated using an EC SecP384r1 algorithm – from the **SecEP Fuse Bank**. The Boot ROM then uses this hash to validate the **Public Key** stored in the SecEP flash storage.

- Once the FW Cert Public Key is verified, the Boot ROM uses the **Public Key** to verify the integrity of the **SecEP Firmware Image** stored in flash storage.
- After the **SecEP Firmware Image** gets verified, the SecEP Boot ROM then verifies whether or not the firmware version number is valid. It does this by comparing the firmware version against the **Rollback Protection Value** stored in the fuse bank. Whenever the SecEP firmware gets updated, the fuses corresponding to the *previous* version get set to indicate that it is no longer valid. Once a Rollback Protection fuse is set, it can't be unset.
- If the SecEP firmware version number is valid, the Boot ROM loads and executes the SecEP firmware.

SecEP BIOS validation and CPU initialization

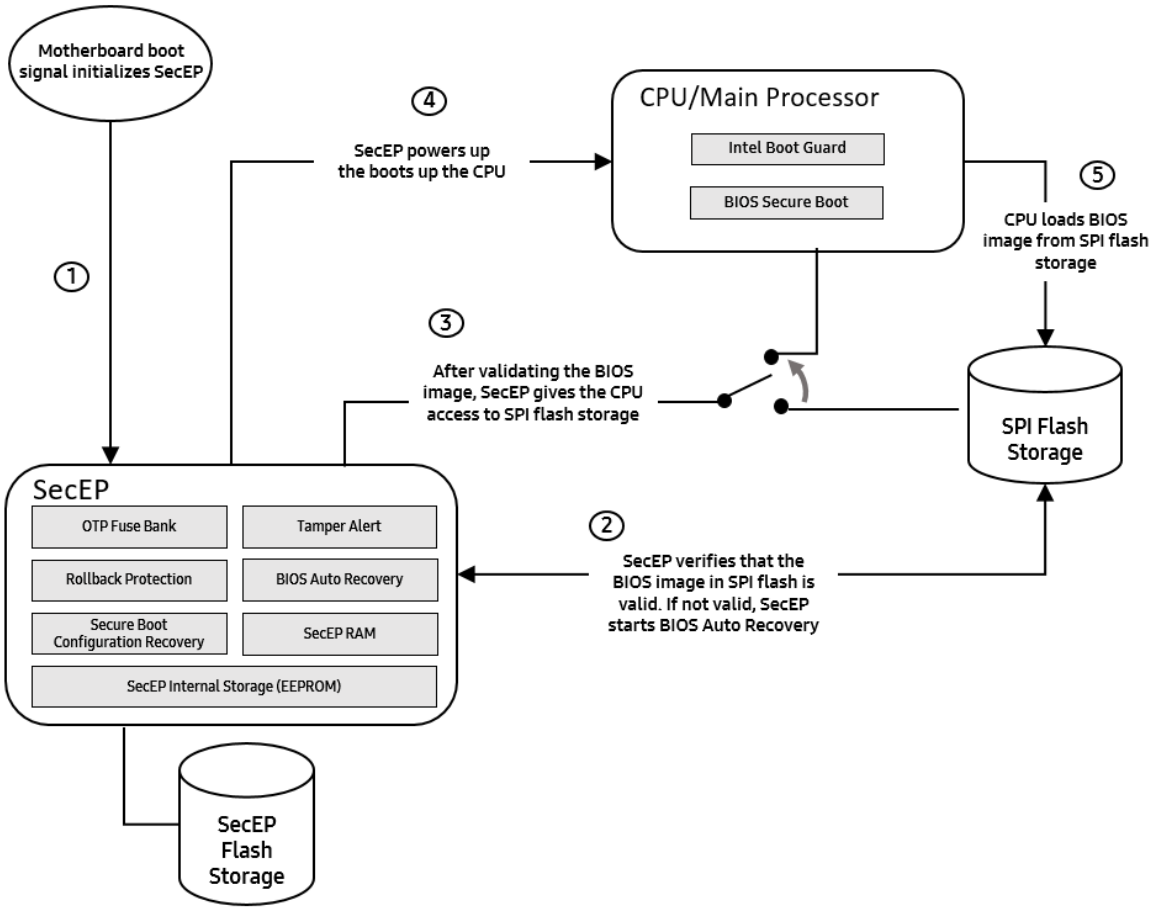

Figure 2: SecEP protection during boot up

Once the SecEP validates and runs its own firmware, it then proceeds to verify the main BIOS. In order to verify the BIOS, the following sequence of events occurs:

- First the SecEP reads the BIOS RSA2048 OEM public key from SPI flash storage as illustrated in Figure 2**, step 2**. This key is checked against a SHA384 hash stored in the SecEP's fuse bank. If the key is valid, the SecEP then checks the BIOS version number against the Rollback Protection Value stored in the SecEP fuse bank.
- Once both the BIOS public key and version number are validated, the SecEP gives the CPU access to SPI flash storage in order to load the BIOS, as illustrated in Figure 2**, step 3**.
- The SecEP then powers on the CPU. Once CPU has power, it can load the BIOS from SPI Flash Storage, as illustrated in Figure 2**, steps 4, and 5**.

If the SecEP detects any incorrectly-signed firmware in place of the BIOS, the SecEP will not grant the CPU access to the firmware, and the CPU will not boot.

After the CPU is powered on, **Intel Boot Guard** independently validates and runs the BIOS Bootloader, which begins the standard UEFI boot process. As part of UEFI, the BIOS Bootloader contains hardcoded **SHA256** hashes for every firmware module that loads. Since these hashes are hardcoded, they are validated by the same signature checks used to validate the BIOS Bootloader. When loading any firmware module, UEFI verifies that the hash of each loaded module is included in the hardcoded table.

BIOS Auto Recovery

During a normal boot flow, the SecEP depends on the CPU's flash storage to load the BIOS firmware. This firmware, while cryptographically verified before being loaded, is still susceptible to attacks from the CPU. If the firmware in flash storage gets corrupted, SecEP will prevent it from loading. While this could mitigate the threat, the device might still be unable to boot up (thus rendering it unusable) without IT admin intervention.

BIOS Auto Recovery addresses this problem by including a back-up copy of the BIOS firmware in the SecEP's flash storage. This storage is physically inaccessible from the CPU and other hardware running on the device. If the SecEP fails to verify the BIOS firmware stored on the CPU's flash storage, it overwrites that firmware with its own locally-stored copy. Once the BIOS firmware has been restored, the SecEP reboots the device so that the recovered firmware can once again be verified and loaded.

Secure Communications with the BIOS

Once the BIOS has been initialized, the CPU and the SecEP communicate with one another through a dedicated channel over Enhanced Serial Peripheral Interface (eSPI)**.** This channel helps ensure that only messages using an authentication key are passed back and forth between these two components.

Since the eSPI is connected to the CPU, the OS also sends messages to the SecEP, opening up the possibility for a compromised OS to impersonate the BIOS and execute privileged operations with the SecEP. To help protect against this type of compromise, the SecEP and BIOS negotiate a per-boot HMAC-SHA256 authentication key using ECDH-P256 during the Pre-EFI Initialization Phase.

This shared key gets maintained by the BIOS as it progresses through its boot stages and enters System Management Mode. Once in this mode, the shared key is hardware-isolated from the OS in a special area of main system RAM dedicated to system management data.

The early key negotiation, combined with the boot-time-guarantees provided by the SecEP, ensure that only the SecEP and the BIOS have access to this authentication key. This enables the BIOS to use the SecEP to secure its data, and the SecEP to restrict data access to only the BIOS.

The BIOS menu also has an indicator to show the health of this communication channel between the BIOS and the SecEP. Errors may occur due to hardware failures in extremely rare cases, or due to malicious activity by a physical attacker. If an error is indicated, several security features will not work, and corrective action such as contacting Customer Support is required.

Secured Data

The SecEP stores all of its data on a dedicated flash storage, which only it has access to. This isolated storage, along with the authenticated communication, enables the SecEP to protect its own data against compromises in the CPU's OS.

The SecEP further protects its data by associating it with a HMAC-SHA256 message authentication key. Each HMAC-SHA256 key is derived using a data identifier and a Master Storage Key unique to each device.

The isolated storage enables the SecEP to protect its data from other software running on the device, and the additional encryption enables the SecEP to protect its data from hardware tampering. This lets the SecEP provide stronger security for the BIOS than the BIOS is capable of providing for itself.

Enhancements to UEFI

The SecEP ensures that the CPU only executes a trusted BIOS, and the BIOS relies on the SecEP to protect security-critical data — this is the foundation that the Samsung Knox platform builds upon to provide best-in-class laptop PC security. In this chapter, we describe how this foundation is used improve on the standard UEFI boot process.

BIOS Configuration Recovery

NOTE: This feature was introduced on Samsung Galaxy Book5 models with the Knox platform. Samsung Galaxy Book4 models with the Knox platform support a prior version of this feature called **Secure Boot Configuration Recovery** that protected and recovered a subset of BIOS variables associated with secure boot.

The security of the UEFI boot process depends on the integrity of both the firmware and its configuration data. Secure boot configuration data is particularly security-sensitive, since it forms the basis to make decisions such as:

- Whether Secure Boot is enabled.
- Which keys to use for Secure Boot
- Which OS images have been revoked.

- Whether the device has certain debugging features enabled that may be abused by a physical attacker to modify in-memory BIOS code or data,
- Whether the device is classified as being in factory, which enables certain special operations during manufacture-time that can be abused by physical attackers.

Therefore, the protection of such UEFI configuration data, and the ability for code to be recovered from a "known-good" backup if corruption is ever detected, is critical in order to establish the root of trust.

The section on Hardware-backed Trust details how UEFI code is protected and auto-recovered from a backup if corruption is detected. However, the same techniques for code cannot be applied for configuration data, since configuration data can be modified after the BIOS signatures are generated. Thus, it is not possible to generate a signature for configuration data during compilation, *and* validate it using the same UEFI code validation flow.

Samsung's **BIOS Configuration Recovery** feature addresses this shortcoming.

As part of the UEFI standard, security-critical configuration data is written to flash as **UEFI/BIOS Variables**. UEFI Variables are stored as key-value pairs on the SPI flash storage, where each key consists of a **Globally-Unique Identifier (GUID)** and a variable name. UEFI configuration data is typically modified by an administrator through the BIOS menu. These "known-good" values are backed up in SecEP flash storage.

An attacker can attempt to subvert Secure Boot by modifying or corrupting variables on the SPI flash storage with inexpensive flash writing tools. For example, they can disable Secure Boot entirely, or change the public keys used to verify signatures. The BIOS Variables Protection and Recovery feature detects such out-of-band writes and also recovers these variables to the previously backed up values.

Detecting and Recovering from Unauthorized Offline Writes to Secure Boot Configuration

First, integrity information for the Secure Boot UEFI variables is stored on the SecEP's flash storage, protecting and isolating it from tampering by untrusted software. This information is cryptographically-protected using the **Master Storage** key kept within the SecEP. This key authenticates the variable's content, name, and GUID.

During boot, the SecEP provides integrity information to the BIOS. The BIOS uses this information to validate the UEFI variables. If an integrity corruption is detected, the BIOS automatically discards any corrupted values, and reverts all protected variables back to their "known-good" values. Once the SecEP completes the recovery, the user is notified through a BIOS notification before the boot process is continued.

Protecting the BIOS Password

The BIOS password is the basis for BIOS authentication, and is therefore particularly security-critical and important to recover if corrupted by hardware failure or malicious activity. The BIOS password hash is itself a UEFI variable. To offer additional isolation, robustness of storage, and recovery guarantees, the BIOS password hash is backed up in the SecEP's internal EEPROM storage. The SecEP's internal storage is very small, and is therefore used to store only the most security-critical data. If corruption of the BIOS

password hash is detected, recovery happens from the SecEP's internal storage (instead of the SecEP Flash Storage).

Tamper Alert

The **Tamper Alert** feature alerts administrators of certain security-critical events that indicate tampering with security-critical data or code in any layer of the software stack. Specifically, this feature deals with tampering that the system cannot automatically recover from. Such events typically require administrator action to restore the system to a good state. Tamper Alerts are presented to the administrator in a protected pre-boot environment. This feature can also require administrators to acknowledge alerts by entering their BIOS password before allowing the system to boot up.

Tamper Alerts are typically indicative of actions by malware or physical attackers that impact security guarantees on the Samsung Galaxy Book with the Knox security platform. For example, Tamper Alerts are raised when there are attempts to boot unauthorized firmware, if there are writes to protected SMM memory, or if corruption of security-critical BIOS configuration data is detected.

These types of attacks can lead to malicious actions such as attempts to persist on disk, manipulated security-critical configurations, privileged escalation, disabled security event logging, bypassed access control, and tampered storage.

Samsung Galaxy Books with the Knox security platform provide tamper alerts for the most common cybersecurity attacks. Each alert is mapped to the MITRE ATT&CK knowledge base, a global and industry-recognized cybersecurity resource providing detailed explanations of the tactics and techniques used in malicious activities, including:

- T1542 (Pre-OS Boot)
- T1565 (Data Manipulation)
- T1068 (Exploitation for Privilege Escalation)
- T1562 (Impair Defenses)
- T1489 (Service Stop)
- T1110 (Brute Force)
- T1490 (Inhibit System Recovery)

For a list of supported Tamper Alerts, see Event Integration with Windows Event Viewer.

Tamper tracking is handled through the SecEP's **Tamper Flag**, an indicator used to determine if there were any new policy violations on the device. The Tamper Flag is stored on the SecEP's flash storage, isolating and protecting it from any compromises on the OS. Authorized software, including the BIOS and SMM (System Management Mode), can request the SecEP to turn on the Tamper Flag when they detect tampering.

Tamper Alert can be configured to require either **admin** or **user** confirmation whenever a violation occurs, before the BIOS starts the OS. During system start up, the BIOS Bootloader reads the Tamper Flag state from the SecEP. If this flag indicates that corruption is detected, the following occurs:

- If admin confirmation is required, then the admin BIOS password must be entered to proceed with boot, or
- If user confirmation is required, then the user can acknowledge the alert by choosing to continue with the boot process.

In both cases, the device will continue to display a message indicating that a corruption was detected during system boot, until the Tamper Flag is cleared.

By providing a consistent method for different components to report a corruption, Tamper Alert ensures that IT admins and device users are made aware of compromises when they happen, thus providing enterprises an opportunity to trigger a response more quickly and effectively.

Protecting the SMM

The UEFI/BIOS installs code that remains persistent through the system's runtime in a mode called **System Management Mode (SMM)** to maintain control of the system even after system boot. SMM code handles a variety of low-level, system-wide tasks such as power management, read/write of persistent BIOS Variables and configuration, and also contains any OEM-specific code.

SMM is the highest privilege mode of the processor. Like the BIOS/UEFI, SMM code runs below the OS. SMM has full access to physical memory, SMM-specific memory called SMRAM, MSR special-purpose registers, the SPI flash region to read and write BIOS Variables, and I/O operations. Furthermore, the SMM is designed to be invisible to lower privilege layers such as the OS kernel or hypervisor.

A compromise of the SMM has extremely high impact on security since the SMM is the highest privileged mode and invisible to lower privileged layers. For example, an attacker that compromises the SMM can write to the SPI flash to persist stealthy malware code, disable BIOS Variables such as Secure Boot, and change system configuration. Moreover, the attacker can completely conceal such activities from any OS or hypervisor detection techniques like anti-virus and bypass mitigations.

How SMM Exploitation Works

Attackers typically escalate privileges to the SMM by exploiting vulnerabilities in the SMM code. The OS calls SMM code through **System Management Interrupts (SMI)**, and passes parameters to SMI handlers using a shared memory area called the SMM Communication Buffer.

If the attacker finds a vulnerability such as memory corruption in any of the SMI handlers, they can craft an input in the SMM Communication Buffer to exploit vulnerable code in the SMI handler. An attacker typically uses well-known exploitation techniques like **Return-Oriented Programming (ROP)** to escalate to SMM privileges, disable SPI write protections, and persist malware.

The Knox platform introduces several features dedicated to detecting, hardening, and mitigating the effect of SMM exploits. These features are described below.

SMI Guard: Protecting SMI Calls

The SMI Guard is a security routine that intercepts SMI calls and analyzes the content of the shared buffer to detect malicious inputs or known exploit patterns based on rules, as shown in**Figure 3**.

Once an attack is detected, remedial actions include blocking the SMM call from going through, and alerting an admin.

This feature enables IT admins to define the following rules to limit calls to the SMM at runtime:

- **Attack surface reduction** – This type of rule blocks SMIs that are not required at run-time, including blocking SMI setup function calls and security-critical BIOS Variable updates that legitimately happen only during system initialization at boot-time.
- **Exploit detection –** This type of rule prevents attacks on specific SMI handler calls by detecting exploit patterns and comparing SMI call argument values against known-bad values. If a known-bad argument value is detected, SMI Guard can block such known-bad calls to the vulnerable SMI handler until a firmware patch is deployed.

The Exploit detection feature is used for rapid incident response against newly discovered SMM vulnerabilities until the enterprise deploys a fully patched BIOS version. Enterprises may intentionally delay BIOS updates for stability and compliance purposes. In such scenarios, the SMI Guard minimizes the period of time a device is vulnerable by detecting and blocking exploit attempts on the firmware.

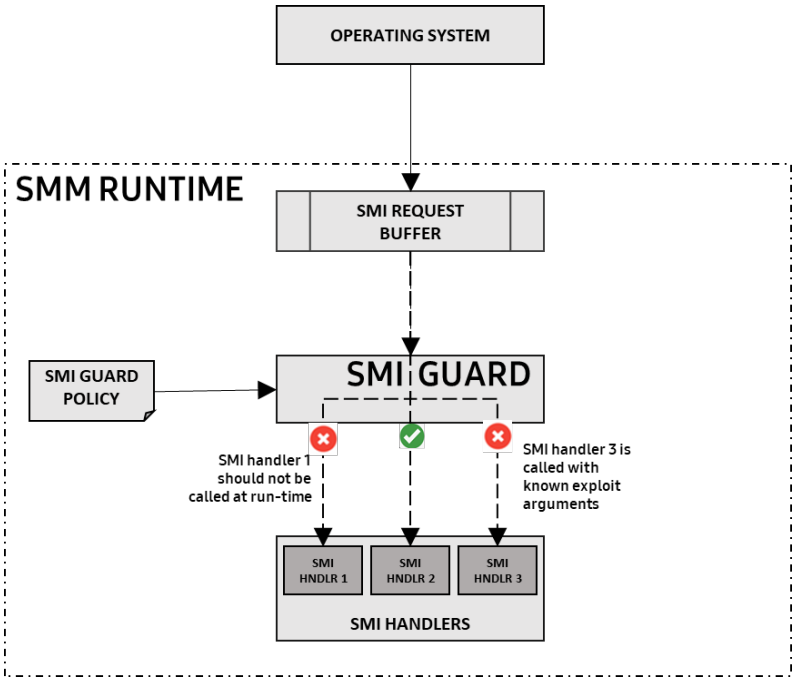

Figure 3: SMI Guard blocks SMI calls that should not be called at run-time and SMI calls with known-bad arguments indicative of exploits

The typical rapid incident response flow is described as follows (**Figure 4**).

1. A new SMM vulnerability is reported (Step 1).

2. If the vulnerability affects a Galaxy Book, and can be blocked by SMI Guard, Samsung rapidly releases SMI Guard rules to block exploitation on vulnerable firmware versions (Step 2).
3. SMI Guard rules are deployed along with a notification (Step 3). The Galaxy Book is now protected from attempts to exploit the vulnerability.
4. Finally, when an updated BIOS firmware version that fixes the vulnerability is deployed, the SMI Guard rule is automatically disabled.

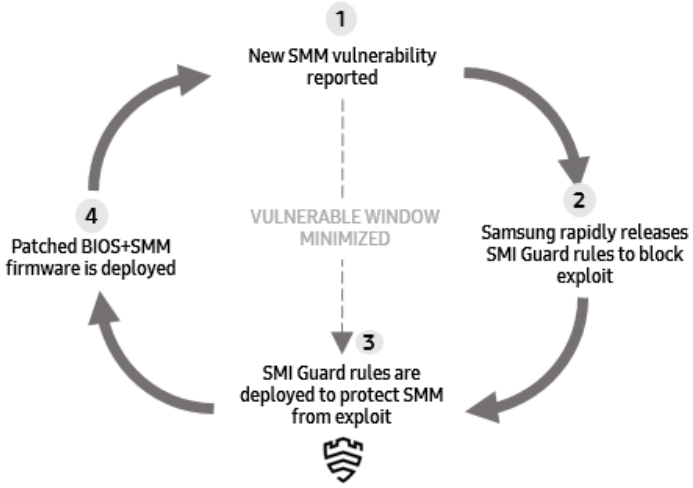

Figure 4: SMI Guard enables rapid incident response and minimizes the window of vulnerability

Advanced SMM Protection

This feature adds a layer of exploit mitigation to protect SMM even in the face of previously unknown vulnerabilities and zero-day attacks.

While chip firmware manufacturers and BIOS vendors take special efforts to ensure that SMM code has no vulnerabilities, given the significant volume of SMM code, attackers are sometimes able to discover previously unknown vulnerabilities (also called “zero days” vulnerabilities). Advanced SMM Protection aims to block the attacker from taking advantage of such these previously unknown vulnerabilities.

Leveraging Existing SMM Exploit Mitigations

Samsung BIOSes have basic exploit mitigations, such as stack cookies, compiled into them. In addition, Intel introduced two security features aimed at making the SMM runtime environment more resilient to attackers, namely **Intel Runtime BIOS Resilience** and **Intel System Resources Defense**, both of which are enabled on Samsung PCs.

The goal of these features is to make it more difficult for an attacker with access to a zero-day to perform anything useful, such as leaking SMM memory or modifying the privileged processor state.

Intel Runtime BIOS Resilience protects page tables by locking them so they cannot be modified later, and enables strong defense mechanisms such as read-only code sections and non-executable data sections.

Intel System Resources Defense further locks-down the SMM runtime environment by running SMI handlers at ring 3 (lower privilege) instead of ring 0 (higher privilege). By forcing all SMI handlers to run at a lower privilege, any attempts to access hardware such as MMIO or MSR's by an SMI handler will trigger a fault and transfer control to a trusted ring-0 component within SMM called the Policy-shim.

The Policy-shim has the power to allow/disallow any resource access, allowing OEM's like Samsung to tightly control which system resources the SMI handlers can access. This makes it much more difficult for a potential attacker to force one of our SMI handlers to perform a malicious action and further compromise the platform.

Introducing Novel Mitigations

While the above mitigations significantly reduce the attack surface and impact of an SMM exploit, they all focus on protecting code but not data. Security-critical data structures in SMM are still vulnerable to unauthorized write attempts that can be leveraged by an attacker. For example, one such data structure describes non-SMM memory regions, which could be modified by an exploit to disclose or modify SMM memory.

Advanced SMM Protection provides additional exploit mitigation by marking security-critical SMM data as read-only, making it impossible for an attacker to overwrite or modify (**Figure 5**).

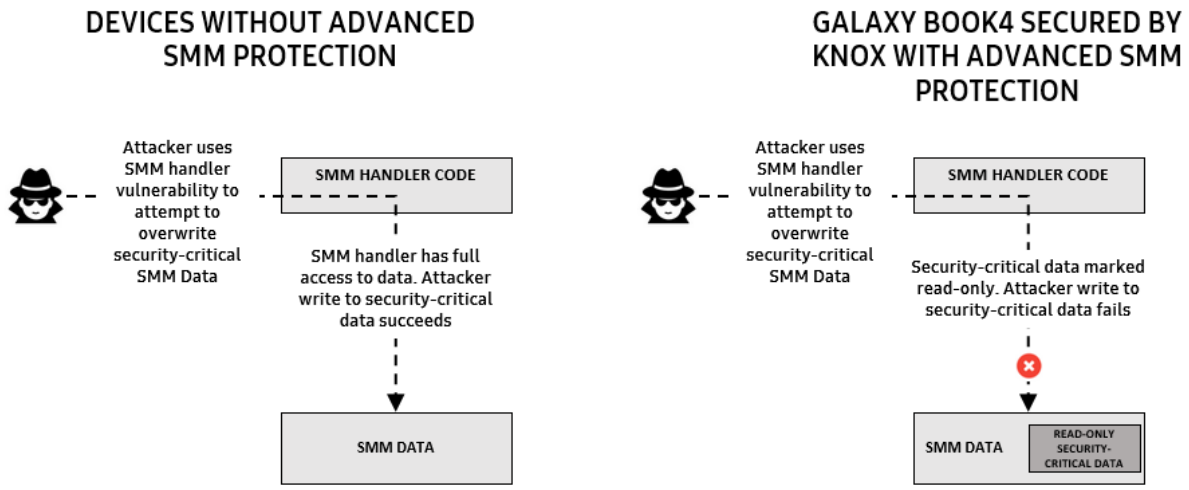

Figure 5: Advanced SMM Protection mitigates zero-day SMM vulnerabilities by blocking writes to security-critical data

Extending Below-the-OS Security into the OS

Having established a secure Below-the-OS platform, it is critical to extend security into the OS. Several security-sensitive operations, such as logging, occur either in the OS kernel or in processes. For robust security, Below-the-OS platform components need to integrate with, and protect these security-critical OS components.

Samsung PCs have special Below-the-OS features to protect security-critical OS components from tamper, and to integrate Below-the-OS security events into Windows events logs.

Samsung Security Service Tamper Protection using Below-the-OS

Malware and ransomware often attempt to kill, disable and uninstall security components such as logging services to evade detection. Therefore, it is important to protect such services from tamper. However, protection is especially challenging since Windows malware often executes with elevated system privileges, allowing malware to easily kill and uninstall security-related services.

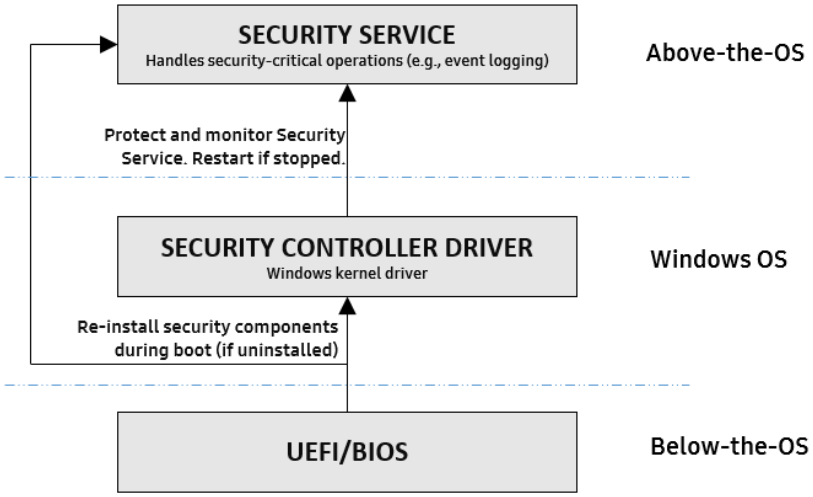

Figure 6: Samsung Security Service Tamper Protection from Below-the-OS

Samsung's Security Service handles critical operations such as logging. It is therefore important that the Samsung Security Service remains protected from tamper.

This protection begins at boot-time and extends through run-time. During boot-time, as shown in **Figure 6**, the UEFI/BIOS, securely re-installs the Samsung's Security Controller Driver and the Samsung Security Service if either of these are uninstalled or not present. It does this by using a standard Windows mechanism called WPBT during boot. This process verifies that the security driver and service binaries are signed by Samsung before starting them.

The Security Controller Driver and Security Service binaries are embedded into the UEFI image, and are thus immune from filesystem tampering. At run-time, the Samsung Security Controller Driver monitors and protects the Security Service by using layered defense mechanisms. If the Security Service process is somehow stopped, the Security Controller Driver is notified through callbacks and instantly restarts the service. The Security Service binary is itself compiled with protection against run-time attacks such as code or DLL injection using various defense-in-depth techniques.

Event Integration with Windows Event Viewer

Another important aspect of extending Below-the-OS security into the OS is to ensure that Below-the-OS security events are appended to existing OS logs in a secure manner. Samsung's Below-the-OS and in-OS security components generate boot-time and runtime security events that are stored in logs in persistent and secure SPI flash memory.

The Security Service's **Event Logging Service** appends these events to Windows Event Logs, which can be viewed using the Windows Event Viewer. To ensure that event logging cannot be bypassed, the Event Logging Service is started on every boot and is protected from tamper or shut-down at run-time (as described in the Samsung Security Service Tamper Protection section).

The Event Logging Service logs several types of security events and information that can be viewed using the Windows Event Viewer under a dedicated "Samsung Security" application event log section. The following events are available:

- **Below-the-OS security events** such as Tamper Alerts, BIOS Auto Recovery and BIOS Configuration Recovery attempts, unauthorized writes to BIOS Variables and protected SMM areas, and suspicious SMI calls.
- **In-OS security events** such as uninstalls, unloads of security-critical components such as the Security Service, and writes to security-critical registry keys.
- **Security state of the system**, such as the security software version information, enable/disable status of security-relevant BIOS settings, and variables such as Secure Boot, Advanced SMM Protection, SMI Guard, and TPM activation status.

For a complete list of security events integrated into the Windows Event Viewer, please refer **Table 1.**

Event severity levels are one of **Information** (normal but security-relevant events), **Warning** (notice of abnormal events that have been automatically resolved), and **Error** (unrecoverable events that need administrator action).

User Notifications

When certain security-critical events happen at runtime, in addition to being logged, the user is notified of the event through a Toast Notification in Windows. The user has the option to obtain more information or dismiss the event (see **Figure 7**).

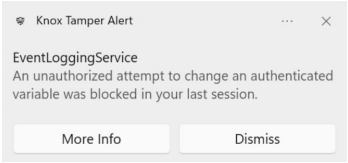

Figure 7: Notification of a Tamper Alert

List of Security Events

Table 1 shows a summary of the security events logged to the Windows Event Viewer, along with their IDs. The logged events contain additional event-specific details and mapping to the MITRE ATT&CK knowledge base, where you can learn more about specific techniques corresponding to potential attack scenarios.

Table 1: A list of security events with their Windows Event Viewer ID that are supported by Samsung Galaxy Book models with the Knox security platform.

Security Events	Windows Event Viewer IDs
Category: Hardware Root-of-Trust	
BIOS Auto Recovery has occurred	0x301-0x303[1]
The recovery BIOS image has been updated	0x304[1]
Error in BIOS Auto Recovery	0x3FE[1]
Error in the secure communication channel with SecEP	0x3FF[2]
Failed to store or retrieve from SecEP flash storage	0x306-0x308[1], 0x308-0x30A[2]
Boot device verification fail (db, dbx, other)	0x430[1], 0x431[1], 0x432[2]
Default keys provisioned for secure boot, device in user mode	0x600[2]
Failed to provision default keys for secure boot, device still in setup mode	0x601[2]
Secure boot BIOS Variables were deleted	0x602[2]
Secure boot keys changed to default	0x603[2]
Critical secure boot configuration was removed (PK, KEK, DB, DBX)	0x610-0x613[2]
Critical secure boot configuration was set (PK, KEK, DB, DBX)	0x614-0x617[2]
Critical secure boot configuration was appended to (KEK, DB, DBX)	0x618-0x61A[2]
Category: BIOS Configuration Recovery	
Corrupted BIOS Variable detected	0x400[1], 0x403-0x406[2], 0x440[2], 0x444[2], 0x906-0x90D[2]
Failed to recover corrupted BIOS Variable	0x401[1], 0x441[2], 0x445[2], 0x966-0x96D[2]
Successfully recovered corrupted BIOS Variable	0x402[1], 0x442[2], 0x446[2], 0x936-0x93D[2]
Error enabling BIOS Configuration Recovery	0x407[1], 0x447[2], 0x409[2]
Error disabling BIOS Configuration Recovery	0x408[2], 0x443[2]
Category: Tamper Alerts	
Tamper Alert log is full	0x410[1]
Tamper Alert count is invalid	0x411[1]
Tamper Alerts are cleared	0x412[1]
Attempt to boot unallowed, known-vulnerable, or unsigned OS	0x100[1], 0x101[1], 0x119[2]

Unauthenticated attempt to write to BIOS Variable (PK, KEK, DB, DBX)	0x102-0x105[1]
Corruption of Tamper Flag detected	0x10B[1]
Advanced SMM Protection detected invalid write to SMM memory	0x10D[1]
Security Service was uninstalled	0x10E[1]
Invalid SMI Guard rule detected	0x10F[1]
The Windows Platform Binary Table (WPBT) is disabled	0x113[1]
Invalid BIOS Password entered 3 times	0x115[1]
Write to SecEP internal storage failed	0x116[1]
Write to SecEP external storage failed	0x117[1], 0x118[1]
Category: SMI Guard	
Malformed SMM request detected by SMI Guard rule	0x420[1]
Category: Samsung Security Service and Security Controller Driver	
Security Controller driver loaded	0x500[1]
Security Controller driver unloaded	0x501[1]
Security Service was killed but restarted	0x502[1]
Security Service loaded	0x504[1]
Security Service started	0x505[1]
Security Service stopped	0x506[1]
Security Service shutdown	0x507[1]
Security Service has unhandled exceptions	0x508[1]
Security Service or Controller Driver was installed	0x700[1]
Failed to install Security Service or Controller Driver	0x701[1]
Failed to clean up after Security Service or Controller Driver install	0x702[1]
Reboot required after Security Service or Controller Driver install	0x703[1]
Security Service or Controller Driver was validated after install	0x720[1]
Failed to validate Security Service or Controller Driver after install	0x721[1]

[1] This security event is supported on Galaxy Book 4 and later models with the Knox security platform.
[2] This security event is supported on Galaxy Book 5 and later models with the Knox security platform.

Secure Manageability

The Samsung Galaxy Book with the Knox security platform supports a wide range of remote and local management functionality that can be used by IT administrators to configure their fleet. These include the ability to import and export BIOS configurations, and integration with Microsoft Intune for mass configuration.

Password-less BIOS Management

NOTE: This feature is supported in the Samsung Galaxy Book5 and later models with the Knox security platform

The traditional usage of BIOS passwords for remote and local management results in several security risks for an organization. Passwords have inherent security weaknesses such as being vulnerable to phishing, keyloggers, and brute-forcing. These risks are amplified if an organization uses the same BIOS password across its fleet.

Storing and managing per-device BIOS passwords securely is also a challenge. Remote BIOS management tools require the administrator to send the BIOS password over the network to the managed machine for authentication, and such passwords are vulnerable to sniffing over the network, or malware on a compromised target machine.

To address these security risks, Samsung Galaxy Book with the Knox security platform supports BIOS authentication using password-less public key cryptography, in addition to traditional password-based BIOS authentication.

In this public key cryptography scheme, the IT administrator first generates a key pair: a private key and a public key. The IT administrator then enrolls the public key with the BIOS of a managed device, as shown in **Figure 8(a**), and later authenticates themselves to the BIOS of the managed device by performing cryptographic operations using the corresponding private key. Unlike with a traditional password, the private key is not directly exposed, but only used to perform cryptographic operations.

Samsung Galaxy Book with the Knox security platform supports the following workflows for enhanced security using public key authentication:

Log in to the BIOS menu

This feature enables IT administrators that are physically present at the managed device to log into the BIOS menu using a one-time PIN (OTP) instead of a password.

As shown in **Figure 8 (b)**, the BIOS menu login screen generates a challenge QR code by encrypting an OTP with the previously enrolled IT administrator public key. The IT administrator uses an app on a mobile device to scan the BIOS challenge QR code. The app decrypts the OTP using the IT administrator's corresponding private key and displays the decrypted OTP. The IT administrator then types in the displayed OTP, which the BIOS verifies and allows login its BIOS menu.

Remote BIOS configuration through Microsoft Intune

This feature allows IT admins to remotely push BIOS configuration to managed devices through Microsoft Intune using password-less authentication.

As shown in **Figure 8 (c)**, The IT administrator creates a digital signature over the BIOS configuration using their private key, and uploads the signed BIOS configuration to Microsoft Intune. Microsoft Intune then pushes the configuration down to the managed devices. The BIOS on the managed devices verifies the

digital signature on the configuration using the previously enrolled IT administrator public key and applies the configuration.

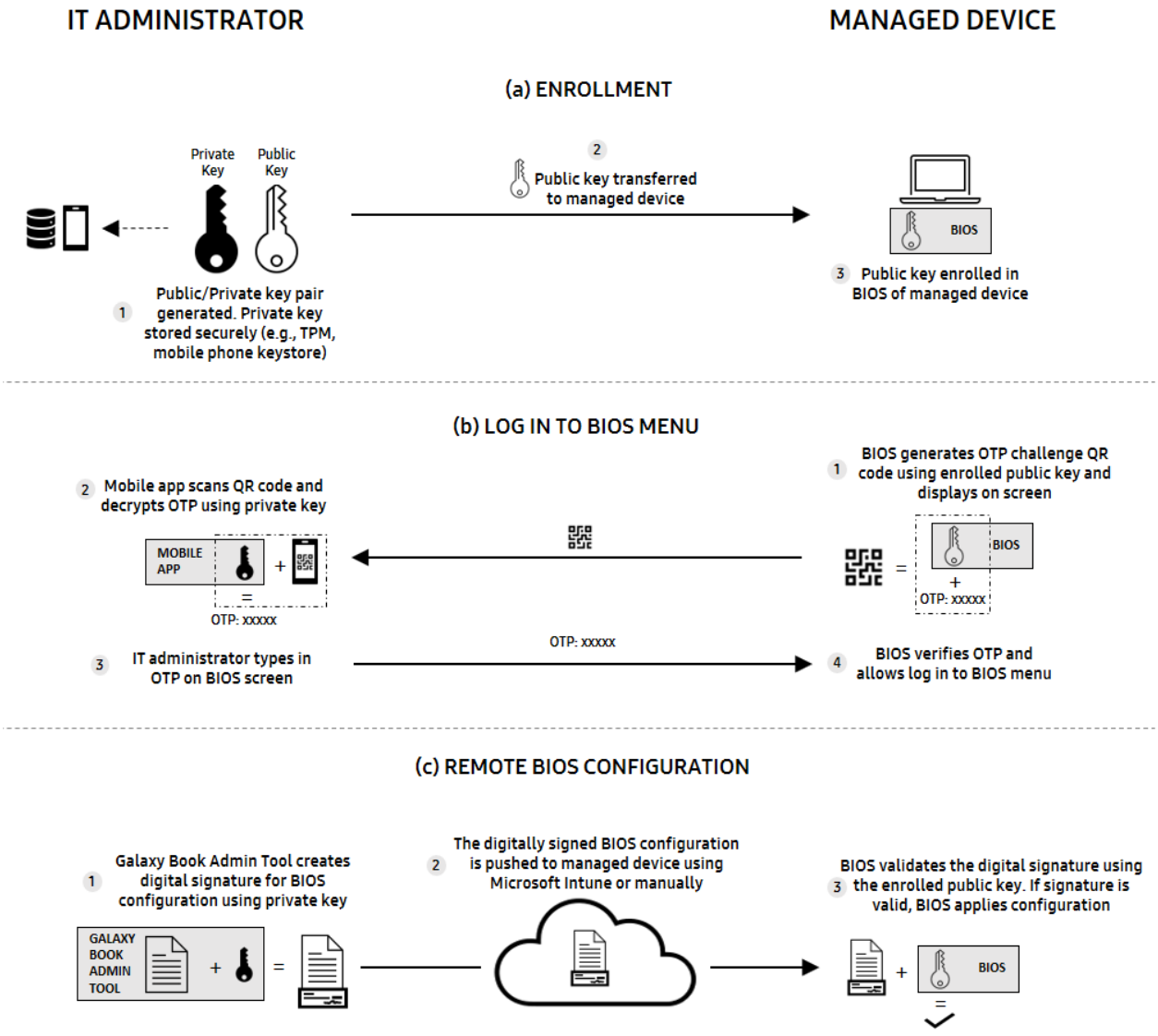

Figure 8: Public key BIOS Management Flows

Enhanced Security

Public key-based password-less authentication significantly enhances the security of BIOS management operations as follows:

- **Limited Risk of Private Key Exposure:** Unlike for passwords, there exist standard and commonly available ways for an IT administrator to securely store and use private keys without exposing them to software, such as Trusted Platform Modules (TPMs) on PCs, and hardware-backed keystores on mobile devices.

- **Secure against OS or Network Compromise:** Malware on the OS or network cannot forge a valid digital signature for a maliciously generated or modified configuration. Since the BIOS checks the digital signature associated with the configuration before applying it, there is a strong guarantee that the BIOS configuration applied is exactly the configuration that was intended by the IT administrator.

Device Firmware Configuration Interface

NOTE: This feature is supported in the Samsung Galaxy Book5 and later models with the Knox security platform

Device Firmware Configuration Interface (DFCI) is a standard from Microsoft and implemented by device manufacturers that focuses on hardware security configuration from the BIOS, especially for device peripherals such as the camera or microphone.

DFCI provides high assurance that particular hardware security features are enabled or locked during boot directly by the BIOS. For example, it can ensure that device peripherals such as the camera are disabled at boot by the BIOS before handing over control to the OS. Therefore, malware, even if it compromises the OS, cannot re-enable these features. By using DFCI to disable certain peripherals like the camera, enterprises can provide strong policy enforcement guarantees, and improve the overall security posture of their fleet.

Samsung adds DFCI support to the BIOS of Galaxy Books with the Knox security platform (starting from Galaxy Book5), enabling IT admins to support the secure disabling of several peripherals including the camera, microphone, speaker, WiFi, NFC, Bluetooth, and USB.

NIST Requirements

The National Institute of Standards and Technology (NIST) publishes standards that outline security requirements for PC platforms. This section describes how Samsung Galaxy Books with the Knox security platform satisfy the following requirements:

Firmware Resiliency:

- NIST 800-193 (Platform Firmware Resiliency)
- NIST 800-147 (BIOS Protection)
- NIST 800-155 (BIOS Integrity Measurement)

Cryptography Support:

- NIST 800-131A (Cryptographic Algorithms)

Media Sanitization Support:

- NIST 800-88 (Media Sanitization)

Platform Firmware Resiliency (NIST 800-193)

NIST 800-193 — published in 2018 — is an extensive set of guidelines that help secure platform firmware from unauthorized corruption. These guidelines outline the requirements for protecting critical firmware code and data from corruption, detecting corruption, and recovering from corruption.

NIST 800-193 is the most comprehensive standard for firmware protection, superseding the much older and narrower-in-scope NIST 800-147 (BIOS Protection) and NIST 800-155 (BIOS Integrity Measurement) standards.

Platform Requirements: Protected, Recoverable, and Resilient

The requirements for classifying a platform as Protected, Recoverable, or Resilient are below:

Protected Platform:

- A Protected Platform must have a trusted component (Root of Trust for Update) that authenticates code updates and data modifications for critical platform firmware, and
- A Protected Platform must protect the integrity of critical platform firmware code and data on non-volatile storage and in runtime memory in addition to the above authenticated update mechanisms.

Recoverable Platform:

- A Recoverable Platform must have a trusted component (Root of Trust for Detection and Recovery) that is able to detect corruption to critical platform firmware code and data and recover code and data from the corruption

Resilient Platform:

- A Resilient platform must support both Protected and Recoverable platform requirements

The protection, detection, and resiliency properties apply to critical platform device firmware code and data:

- **Firmware code:** Firmware executables in storage and in memory at runtime
- **Firmware data:** Firmware configuration, settings, and policy in storage

Definition of Critical Platform Device Firmware

As stated in NIST.SP.800-193, Platform Firmware is responsible to “initialize components, boot the system, and provide runtime services implemented by hardware components” . While platforms support a wide range of devices with associated firmware, the scope of this document specifically covers ***critical platform devices*** and their corresponding firmware:

““For a platform as a whole to claim resiliency to destructive attacks, the set of platform devices necessary to minimally restore operation of the system, and sufficient to restore reasonable functionality, should themselves be resilient. We call this set of devices critical platform devices. The particular resiliency properties may vary from platform to platform.”

In other words, critical platform devices are those responsible to: (i) **boot** and (ii) **minimally restore** the system to a reasonable working state. Critical platform device firmware that runs on Samsung Knox Galaxy Books with the Knox security platform are listed in **Table 2**.

Table 2: Critical Platform Device Firmware for Galaxy Books with the Knox security platform

Critical Platform Device Firmware [1]	Description: Why Critical?
SecEP (Secure Embedded Processor) Firmware	The SecEP is a specialized security chip on Samsung Galaxy Books with the Knox security platform. It is the first code that runs during boot, is the root of trust (RoT) that starts the secure boot process chain, stores the cryptographic keys necessary to protect and detect the integrity of critical UEFI / BIOS data, and restores its own firmware and the UEFI / BIOS firmware if corruption is detected.
Host Processor Boot Firmware (UEFI / BIOS)	The UEFI/BIOS initializes the system (including hardware) and extends the secure boot chain to the OS bootloader.
Platform Runtime Firmware (SMM)	The System Management Mode (SMM) firmware provides critical services to the OS at runtime and is a trusted interface for updating UEFI / BIOS data.
Trusted Platform Module (TPM)	The TPM stores cryptographic keys used for data decryption during boot and a secure record of boot measurements for remote attestation
Intel Management Engine (CSME) Firmware	The Intel CSME firmware loads firmware into hardware blocks and is essential to booting a system.
Power Delivery (PD) Firmware	The PD firmware is responsible for USB-C power delivery

[1] The Network Interface Controller (NIC) firmware is not considered critical to booting the platform, since setups such as PXE boot are not typical for laptop PCs. However, the NIC firmware is still protected by being a part of the BIOS image.

Samsung Galaxy Books with the Knox security platform and NIST 800-193

Samsung Galaxy Books with the Knox security platform meet the firmware resiliency requirements (including protection, detection, and recovery) for critical platform code and data, as detailed in **Table 3**:.

In summary, firmware code is protected in storage by mechanisms such as secure storage and Intel BIOS Guard. Firmware code updates are only allowed after signature verification. Firmware data is stored in secure storage, protected by integrity measurements, and access is allowed only through trusted

channels. Finally, firmware code and data are restored from secure backups whenever corruption is detected.

Table 3: Satisfaction of NIST 800-193 Resiliency Requirements

Criteria	Critical Code Protection	Critical Data Protection
Protection[1]	The SecEP firmware is stored in secure internal flash. The UEFI/BIOS, SMM, CSME, and PD firmware is stored in SPI flash protected by Intel BIOS Guard and write protections The SecEP firmware is the RoT and performs SecEP self-validation and SecEP BIOS Validation to start off Intel Boot Guard and UEFI Secure Boot chain that protects all other firmware, including UEFI/BIOS, SMM, CSME, and PD. At runtime, the SMM firmware runs in a special CPU mode and is protected from the OS and other non-critical code. In addition, SMI Guard and Advanced SMM Protection protects the SMM code from any exploits at runtime. Firmware updates are packed in update capsules and are digitally signed in a Hardware Security Modules (HSMs) by cryptographic keys that are derived from a Samsung Root of Trust As part of the update process using capsules, the integrity of all platform firmware is checked using digital signatures on the update capsules. The secure update process also includes checks for build version to prevent rollback to older versions	Critical UEFI/BIOS/SMM firmware data is stored as EFI variables in secure SPI flash storage SPI flash storage is write-protected and writes allowed only from SMM. EFI variables can only be modified through APIs defined in SMM firmware EFI variables without RT attribute are protected from runtime updates. EFI variables with Authenticated attribute(s) are protected from any un-authorized modification and are protected by signatures * The SecEP firmware's data is stored in dedicated flash storage that only the SecEP has access to
Detection	The secure boot process checks for the integrity of all platform firmware. Only platform firmware that is digitally signed by	The Secure Boot Configuration Recovery feature stores cryptographic hashes of

	approved authorities is allowed to boot. Otherwise, modification is detected and alerts raised.	security-critical boot configuration data, using which corruption can be detected.
Recovery	The BIOS Auto Recovery feature recovers updatable firmware if corruption is detected, including the UEFI/BIOS, SMM, and PD firmware[2]	The Secure Boot Configuration Recovery feature recovers UEFI/BIOS security-critical boot configuration data from known-good backups in protected storage if corruption is detected. Further, factory defaults of UEFI/BIOS/SMM settings can be restored from protected storage
Logging	Tamper Alert alerts admins of certain security-critical events that indicate tampering with firmware data or code. Tamper Alert event logging is integrated with Windows Event Viewer	

[1] The TPM firmware's protection mechanisms for code and data are detailed in TCG TPM 2.0

[2] Recovery of the Intel CSME firmware is not supported. However, Intel CSME firmware has its own recovery mechanisms.

Support for NIST 800-147 (BIOS Protection)

NIST 800-147, published in 2011, focuses specifically on protecting the BIOS image from unauthorized modifications. The standard requires tight controls around the few approved mechanisms that can legitimately change the BIOS image. In addition, the standard lists particular threats that are commonly used to compromise the integrity of the BIOS image, and requires that these threats are addressed. A summary of the requirements is below:

- Controls around authorized mechanisms to modify BIOS image on flash.
- **Authenticated BIOS Update:** The BIOS update mechanism should update the BIOS image in non-volatile storage only after checking the integrity of the updated BIOS image using digital signatures generated by approved cryptographic algorithms through a Root of Trust (RoT), and
- (Optional) **Secure Local BIOS Update:** In cases where the above authenticated BIOS updates are not possible (e.g., due to corruption), local BIOS updates are optionally allowed. Local BIOS update should require physical presence (e.g., through administrator passwords) before updating the BIOS image in non-volatile storage
- Blocking unauthorized updates to the BIOS image.
- **Integrity:** The non-volatile memory storing BIOS image and the RoT keys should be write-protected from unauthorized modifications

- **Non-bypassability:** There should be no other mechanisms to update the BIOS image beyond the above authorized mechanisms.

NIST 800-147 is largely superseded by NIST 800-193. Whereas NIST 800-147 focuses on protecting the BIOS image, NIST 800-193 addresses all critical firmware images (including the BIOS image). However, for completeness, we list how Galaxy Books with the Knox security platform satisfy NIST 800-147 requirements in **Table 4**.

Table 4: Satisfaction of NIST 800-147 BIOS Protection Requirements

Criteria	How Satisfied
BIOS Update Authentication	* BIOS updates are packed in update capsules and are digitally signed in a Hardware Security Modules (HSMs) by cryptographic keys that are derived from a Samsung Root of Trust * As part of the secure update process, the integrity of the BIOS update image is checked by verifying digital signatures on the update capsules. * The secure update process also includes checks for build version to prevent rollback to older versions * The secure update process is protected by itself being part of the BIOS image or in System Management Mode (SMM) memory and by UEFI Secure Boot * The cryptographic keys used to verify the update capsule digital signature are part of the BIOS image, which is stored in write-protected SPI flash
Digital Signature Strength	* BIOS update images are digitally signed using RSA-2048 or higher, thus bringing the strength of the signature above 112 bits as recommended by NIST 800-131A
Secure Local Update (Optional)	* The secure BIOS recovery and update flow requires the BIOS administrator password, therefore enforcing physical presence
Integrity Protection of Storage	* The BIOS image is stored in SPI flash protected by multiple layers of write protection – Intel BIOS Guard and SPI write lock
Non-Bypassability	* SPI flash write access is available only to authorized update code running in the secure System Management Mode (SMM) memory at runtime

	* SPI flash write access is otherwise locked early in the boot process, and especially much before any third-party firmware driver is executed

Support for NIST 800-155 (BIOS Integrity Measurement)

NIST 800-155 deals with integrity measurements and reporting of the BIOS. The standard requires that any changes to the BIOS code (firmware image) and data (settings) must be measured faithfully, stored securely, and reported to a verifier (IT admin). A summary of the requirements are as follows:

- **Measurement:** A trusted component (Root of Trust for Measurement) measures, through a cryptographic hash:
- **BIOS Code:** BIOS boot code modules, including the BIOS Boot Block and SMM code
- **BIOS Data:** User-configurable settings of the BIOS
- **Storage:** The integrity measurement summary must be stored in secure, tamper-evident storage managed by a trusted component (Root of Trust for Storage). A log describing the measurements may optionally also be stored.
- **Reporting:** A trusted component (Root of Trust for Reporting) reports the stored integrity measurements ensuring authentication, identity of the endpoint, and freshness of the report
- Requirements of remediation and restoration of the BIOS.

As mentioned, NIST 800-193 is much broader in scope than NIST 800-155 and supersedes it. However, for completeness, However, for completeness, we list how the Galaxy Books with the Knox security platform satisfy NIST 800-155 requirements in **Table 5**.

Table 5: Satisfaction of NIST 800-155 BIOS Integrity Measurement Requirements

Criteria	How Satisfied
Measurement	* **BIOS Code:** During boot, all BIOS boot components are measured as part of the Secure Boot. The measurements include all components specified in NIST 800-155 (BIOS boot block, SMM, BIOS recovery and update mechanism in the SecEP, Option ROMs, ACPI/POST) * **BIOS Data:** During boot, the integrity of security-critical boot configuration data is not only measured and verified, but also recovered from if corruption is detected.[1]
Storage	* The integrity measurement is stored as an extended hash in secure, tamperproof Platform Configuration Registers of the TPM * Measurement log entries are added to the TPM event log

Reporting	* TPM Attestation reports the measurements stored in PCRs. * Attestation is signed by a key provisioned into the TPM during manufacture that proves its authenticity using a certificate chain rooted in a trusted key. This attestation key also uniquely identifies the endpoint. * Attestation protocol ensures freshness using a nonce.
Remediation and Restoration	* The BIOS image and data are recovered after authentication through BIOS Auto Recovery and Secure Boot Configuration Recovery

[1] Integrity measurements of only the security-critical boot configuration data and not all user-configurable settings is supported

Other NIST Standards

- **NIST 800-131A (Cryptographic Algorithms):** This standard prescribes minimum strengths of cryptographic algorithms for various purposes such as encryption and digital signatures. Samsung Galaxy Books with the Knox security platform satisfy (or exceed) these requirements where relevant as outlined in the above sections.
- **NIST 800-88 (Media Sanitization):** This standard describes guidelines for secure media sanitization that ensures proper destruction of data. Galaxy Books with the Knox security platform support secure media sanitization for solid-state drives (SSDs) connected through SATA and NVMe.

Glossary

BIOS — A collection of firmware components that includes the below the OS environment and the BIOS setup utility. The BIOS setup utility is used to configure hardware and BIOS features.

BIOS Auto-Recovery — A **Samsung Galaxy Book4** feature through which the Secure Embedded Processor (SEC) will automatically recover a working BIOS if the original has been corrupted.

Secure Boot Configuration Recovery — A **Samsung Galaxy Book4** feature through which UEFI variables related to BIOS Secure Boot are protected and recovered if corrupted.

Intel Boot Guard — A feature available on Intel processors to validate the initial boot block of the BIOS by signature and version number. The initial boot block verifies and loads subsequent BIOS components.

One Time Programmable (OTP) Fuse Bank — A collection of hardware fuses that can only be set once. This is used to store the firmware rollback protection value and firmware certificate hash.

SPI Flash Storage device — A flash storage device used for storing the BIOS firmware and any data required by the BIOS.

Secure Embedded Processor (SecEP) — A **Samsung Galaxy Book4** processing unit running alongside the main CPU. The Secure Embedded Processor provides security features for the BIOS.

System Management Mode — A privileged CPU mode used for power management, UEFI variable management, and firmware management.

Tamper Alert — A **Samsung Galaxy Book4** feature through which booting an OS requires user or admin confirmation if any boot policy violation has been detected.

Trusted Platform Module (TPM) — A dedicated microcontroller designed for cryptographic operations.

UEFI Standard — Either the UEFI Specification that defines the software interface between an OS and firmware or the reference implementation for the UEFI Specification.

UEFI variables — Persistent data stored on SPI Flash Storage. The format and interface for UEFI Variables are included as part of the UEFI Specification. This data is used by the BIOS and by UEFI components running in the OS.