SAMSUNG

Galaxy Book2 Business

Technical White Paper

Security Below the OS

March 2022

Contents

Introduction

This document provides an overview of the **Samsung Galaxy Book2 Business** laptop PC, 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 the Galaxy Book2 Business as a comprehensive enterprise laptop PC solution to deploy to their workforce.

Please note that this white paper specifically describes the Samsung Galaxy Book2 Business. The security features described here don't necessarily reflect the security features available on other Samsung Galaxy Book laptop models.

Protecting below the OS

Laptops 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 **code**, as these are vulnerable to hackers. The following vulnerabilities are present if the BIOS isn't properly protected:

Data compromise

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 its 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 Samsung built the Galaxy Book2 Business; our most secure and tamper-resistant laptop to date.

Introducing Samsung Galaxy Book2 Business

The Samsung Galaxy Book2 Business was designed to provide best-in-class hardware-backed security to meet the needs of enterprise users. It pairs industry-wide standards with powerful OEM features to provide a high level of firmware protection and robust firmware recovery capabilities.

Every Galaxy Book2 Business 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 in the Samsung Galaxy Book2 Business. Join us, as we showcase our best-in-class, most secure and protected laptop PC yet.

Hardware-backed 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.

With the Samsung Galaxy Book2 Business, the boot sequence begins with the SecEP. When the motherboard first powers up, it initializes the SecEP and another module called the **Embedded Controller (EC)** in parallel.

The EC is responsible for providing power to the CPU. However, it doesn't proceed until the SecEP grants it permission to do so. Before the SecEP can grant this permission, it first verifies whether or not the BIOS is corrupted. If the BIOS is safe and corruption-free, the SecEP proceeds to allow the EC to boot up the CPU, and the CPU to access 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 1** 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 Samsung Galaxy Book2 Business from threats and compromises.

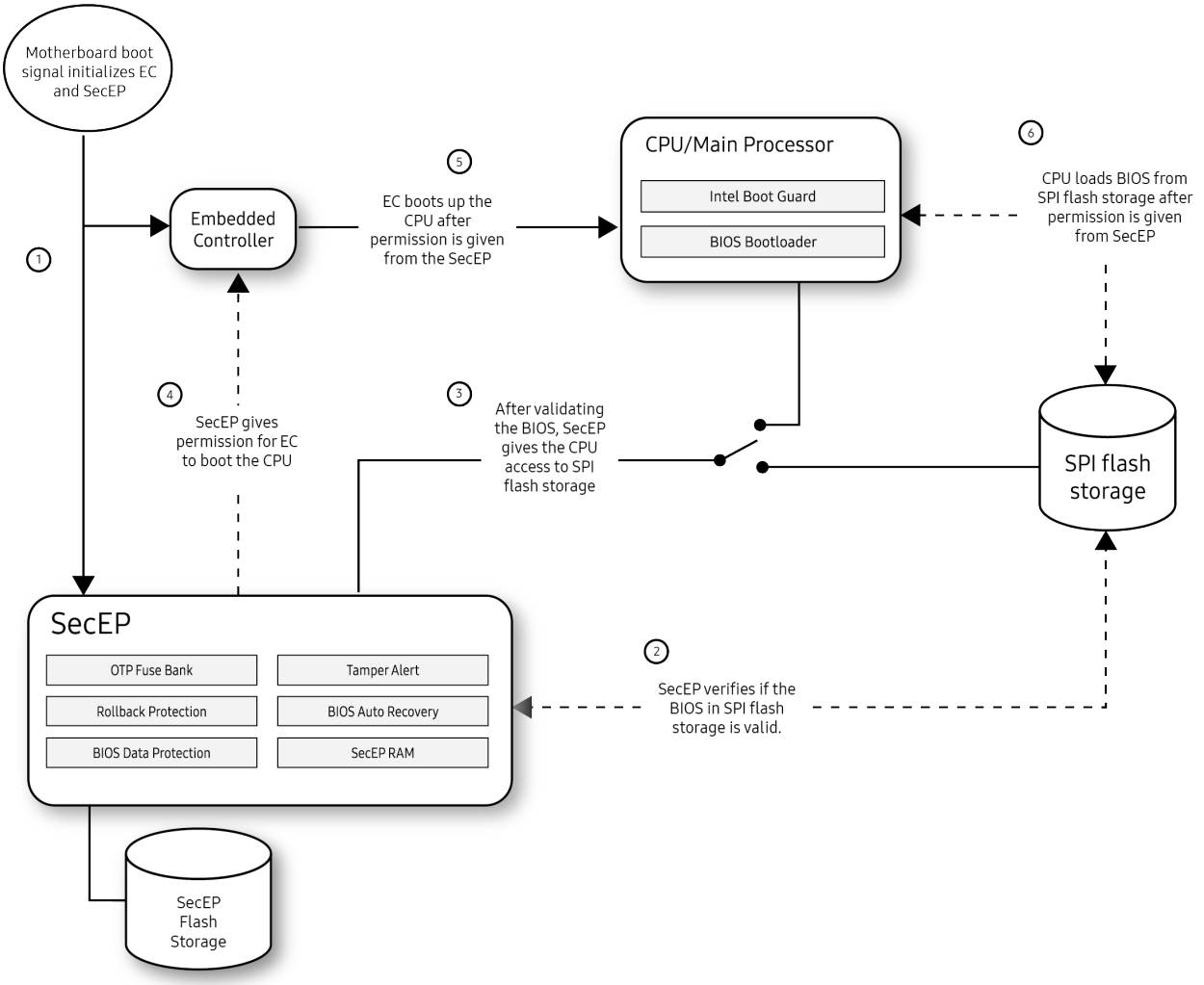

Figure 1: SecEP protection during boot up

SecEP FW self-validation

With Galaxy Book2 Business, 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 2 describes how the SecEP performs this FW self-validation:

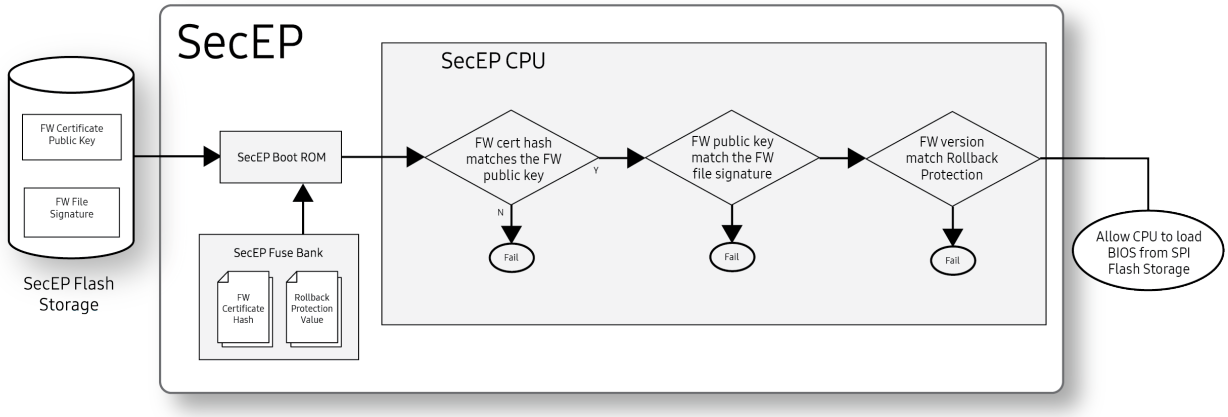

Figure 2: SecEP FW self-validation

- 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 **FW Certificate Public Key** stored in the SecEP flash storage.
- Once the FW Cert Public Key is verified, the Boot ROM uses the key to validate the **Firmware File Signature** stored in flash storage.
- After the FW File Signature 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

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 1, 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 1, step 3**.
- The SecEP then notifies the EC that it can power on the CPU. Once CPU has power, it can load the BIOS from SPI Flash Storage, as illustrated in **figure 1, steps 4, 5, and 6**.

If the SecEP detects any incorrectly-signed firmware in place of the BIOS, the SecEP won't grant the CPU access to the firmware, and the CPU won't 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 own 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 line called the System Management BUS (SMBUS). This line helps ensure that only messages using an authentication key are passed back and forth between these two components.

Since the SMBUS 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.

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 Samsung Galaxy Book2 Business 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 Data Protection

The security of the UEFI boot process depends on the integrity of both the firmware and its configuration data. While signatures can be validated for immutable firmware, the same level of protection can't be applied to the mutable configuration data used by that firmware. Secure Boot depends on configuration data to determine:

- Whether secure boot is enabled.
- Which keys to use for secure boot.
- Which OS images have been revoked.

Since configuration data can be modified after the BIOS Bootloader signatures are generated, it isn't possible to protect configuration data using the UEFI validation flow. BIOS Data Protection addresses this shortcoming.

As part of the UEFI standard, security-critical boot configuration data is written to flash as UEFI Variables. UEFI Variables are stored as key-value pairs, where each key consists of a globally-unique identifier (GUID) and a variable name. These variables get read and modified by the BIOS and the OS.

An access control policy is assigned to each variable, determining when the variable can be modified. Based on this policy, some variables can only be modified by the BIOS Bootloader, while others can be modified by both the BIOS Bootloader and OS.

While UEFI allows for variables to be write-protected – thus defending them from modification by a malicious OS – they still aren't completely secure against tampering for several reasons:

- First, the OS may be able to exploit the BIOS's variable assignment logic to allow for illegal variable writes.
- Second, a local attacker could directly modify variables on the flash storage with inexpensive flash writing tools.

BIOS Data Protection solves these problems in two ways:

Detecting out-of-band variable writes

First, the critical UEFI variables on flash are paired with **a Message Authentication Code (MAC).** This MAC is generated from the Master Storage key (kept within the SecEP), and a SHA256 hash of the variable's content, name, and GUID.

UEFI Variable hashes and corresponding MACs are stored on the SecEP's dedicated storage, protecting and isolating them from software and hardware-based tampering. At boot time, the SecEP verifies if the stored MACs match the stored UEFI variable hashes. If there is a match, the SecEP provides the UEFI variable hashes to the BIOS Bootloader.

When a corruption is detected — where the MAC and stored hashes don't match — the BIOS will automatically discard the corrupted value and revert all protected variables back to factory default values. The violation is then reported to the SecEP **Tamper Alert** service, which allows an admin to configure the response behavior through the BIOS setup utility.

Limiting changes to UEFI Variables

Second, BIOS Data Protection ensures that protected UEFI Variables may only be configured from the BIOS setup utility when Secure Boot is enabled.

As previously discussed, the BIOS and SecEP communicate with one another using authenticated message keys over a secure SMBUS. Before generating a MAC for an updated UEFI Variable value, the SecEP will validate that the request came from the BIOS. This helps to ensure that only the BIOS can interact with the SecEP to generate these MACs, and not any other system.

If another system — like the OS — attempts to update the UEFI variables directly without updating the MAC, the BIOS will detect that the new hash doesn't match the validated hash provided by the SecEP and reject the updated values. Similarly, if an attacker had physical access to the CPU's flash storage, they wouldn't be able to modify a UEFI variable without tampering with the SecEP storage and updating the corresponding hash and MAC values.

By forcing all UEFI modification requests to be made between the BIOS and the SecEP, the Galaxy Book2 Business provides a BIOS-level guarantee that these variables can't be manipulated by a compromised OS or physical attacker.

BIOS Tamper Alert

Once the BIOS Bootloader has initialized the runtime environment, it is expected to load and hand over execution control to the OS. The image for the OS is validated by signature to ensure that it was approved by the correct party, and its hash is checked against a revocation list to ensure that the image isn't known to be malicious or compromised.

Once validated, the BIOS Bootloader transfers control to the OS. The BIOS continues to run in System Management Mode, and the OS may invoke BIOS APIs through software interrupts.

As a final validation before handing over execution to the OS, Tamper Alert checks if there were any error cases reported by other boot components. If so, it ensures that the device user and admin are made aware. This final step is important for two reasons:

- Loss of ground truth. When data is corrupted, the BIOS may lose access to the ground truth policy that it must enforce. This happens when BIOS Data Protection recovers the secure boot policy configuration data. In cases like these — where the policy data is corrupted — the BIOS might not be able to recover the original intended configuration, therefore reverting back to a Samsung provided default configuration. This can result in the BIOS violating an enterprise-level policy.
- Off-device response procedures. The BIOS may execute response procedures on-device, but enterprise policies can require a broader response involving other parties. The BIOS can't directly trigger these responses, and so it must inform a party capable of triggering these responses.

Because of these limitations, it is important to raise awareness of corruptions and enable other enterprise response processes. Tamper Alert does this by tracking violations and ensuring that the device user and admin are made aware of them.

Tracking is handled through the SecEP **Tamper Bit**, a binary indicator used by the BIOS and SecEP to determine if there were any new policy violations on the device. The Tamper Bit is stored on the SecEP's storage, isolating and protecting it from any compromises on the OS.

Through the SMBUS, the BIOS Bootloader directs the SecEP to set the Tamper Bit during the following scenarios:

- When keys and certificates for protecting authenticated boot configuration data (including KEK, DB, DBX) haven't been properly signed by the root key, PK. This is detected as part of the UEFI boot process.
- When protected boot configuration data has been corrupted. This is detected by BIOS Data Protection, and happens when attempting to read a protected UEFI Variable that was corrupted on flash, updated without the appropriate BIOS APIs, or modified while the OS was running.
- When attempting to load firmware in violation of the boot policy. This happens when the OS image hasn't been signed by an approved party, or when the OS image has been revoked.

Tamper Alert can be configured to either require **admin** or **user** confirmation of any violation before the BIOS starts the OS. During system start up, the BIOS Bootloader reads the Tamper Bit state from the SecEP. If this bit is set, and a corruption is detected, then the following occurs:

- If admin confirmation is required, the device won't boot until the Tamper Bit has been cleared, which can only be done by providing an admin password.
- If user confirmation is required, the user may choose to continue booting the device without clearing the Tamper Bit.

In both cases, the device will continue to display a message indicating a corruption during boot until the Tamper Bit 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 quickly and effectively.

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 Book2 Business feature through which the Secure Embedded Processor will automatically recover a working BIOS if the original has been corrupted.

BIOS Data Protection — A Samsung Galaxy Book2 Business feature through which offline or abnormal updates to UEFI variables are rejected by the BIOS.

Embedded Controller — A processing unit running alongside the main CPU. The Embedded Controller is the first device powered on by the motherboard on boot.

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 Book2 Business 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 Book2 Business 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.