ISSN: 2349-6002

Measured Boot Detection of Tampered BIOS Image

Kushagra Dixit[1], Kartik Kumar Jaiswal[2], Rahul Mishra[3]
[1,2,3]*Department of Computer Applications, Invertis University, Bareilly, India*
doi.org/10.64643/IJIRTV12I11-207268-459

***Abstract*—The integrity of platform firmware has emerged as a critical component of system security, as the BIOS or UEFI represents the foundational layer that governs the boot process. Compromise at this level enables adversaries to establish persistent control, bypass operating system protections, and undermine trust anchors. This paper examines the role of measured boot in detecting tampered BIOS images by leveraging the Trusted Platform Module (TPM) and its Platform Configuration Registers (PCRs), together with the Trusted Computing Group (TCG) event log. We analyze how measured boot records can be applied to identify unauthorized modifications, including insertion of hidden code modules, removal of signature verification routines, alteration of bootloader sequences, and manipulation of firmware configuration data. A conceptual detection framework is proposed that combines hash comparison against known-good baselines with rule-based evaluation of event log sequences, enabling recognition of both direct binary changes and structural anomalies. The discussion highlights challenges such as distinguishing malicious alterations from legitimate vendor updates, managing variability across hardware platforms, and minimizing false positives. By synthesizing insights from existing standards and research, this study demonstrates how measured boot can serve as a reliable mechanism for BIOS integrity verification, strengthening supply-chain assurance and enhancing trust in modern computing environments.**

***Index Terms*—BIOS integrity, Bootloader manipulation, Firmware image modification, Option ROM injection, Platform Configuration Registers (PCRs), Secure Boot, Secure Boot bypass, TCG event log, Trusted Platform Module (TPM), UEFI firmware**

I. INTRODUCTION

The security of a computer system starts with the firmware that brings the hardware to life and loads the operating system. The BIOS or Unified Extensible Firmware Interface (UEFI) performs this job, and its integrity is critical for keeping the platform trustworthy.[1] If attackers manage to change this firmware, they can take control before the operating system even starts, bypass protections, and install malware that stays hidden and survives reinstallations. This makes BIOS/UEFI protection a key requirement for secure computing.

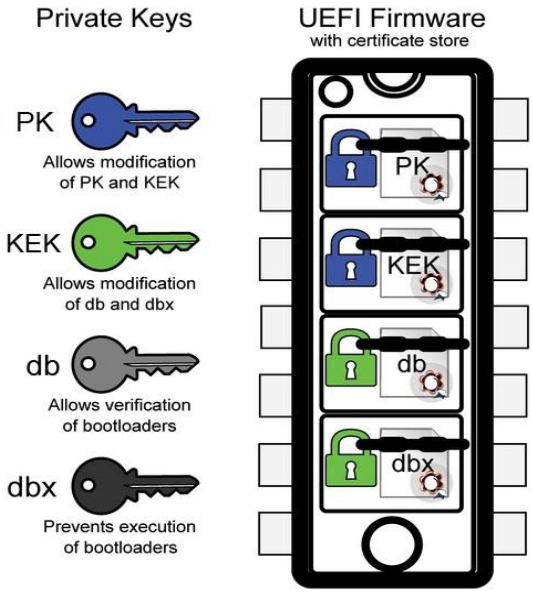

Fig. 1. UEFI Secure Boot Key Architecture Showing PK, KEK, db, and dbx for Trusted Boot Verification.

Operating system–level detection methods are not enough because they cannot fully observe or validate firmware behavior. To fill this gap, Fig1. represents the Trusted Computing Group (TCG) introduced measured boot, where each stage of the boot process is hashed and stored in Platform Configuration Registers (PCRs) inside the Trusted Platform Module (TPM).[2],[12] Event logs are also created, giving a verifiable record of the boot sequence. By checking these records, defenders can spot changes such as disabled verification routines, altered boot orders, or hidden modules. Secure Boot works by using a chain of trust built into the firmware.[9]. At the top is the Platform Key (PK), which acts as the root authority.

Only whoever controls this key can change the next layer, the Key Exchange Keys (KEK). These KEKs then decide which updates are allowed for the signature databases. The DB (allowed list) holds certificates of trusted bootloaders and drivers, while the DBX (blocked list) contains revoked or unsafe binaries. In practice, this setup means the system can both permit new trusted components and quickly block vulnerable ones, keeping the boot process secure.

II. THREAT MODEL

A. Attacker Objectives

Firmware image modification involves attackers rewriting the BIOS/UEFI image stored in SPI flash [10]. Fig2. demonstrated this process requires firmware flashing tools, reverse engineering expertise, and local administrative or physical access, with the primary goal of achieving persistence across operating system reinstallations and hardware changes. Bootloader manipulation refers to tampering with the UEFI boot manager or the EFI System Partition to hijack the boot sequence [8].

It demands knowledge of bootloader internals and the ability to exploit update mechanisms, enabling early execution before operating system protections are activated. Secure Boot bypass includes disabling or patching signature verification routines or injecting unauthorized entries into the DB/DBX [3], [4]. This approach requires a deep understanding of the Secure Boot hierarchy and cryptographic signing processes, with the objective of breaking the trust chain and executing unsigned EFI binaries.

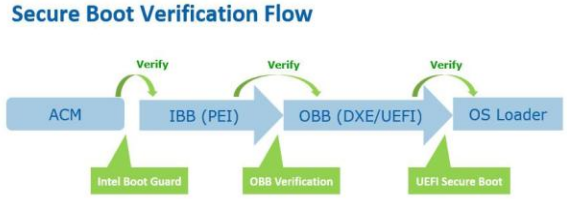

Fig. 2. Secure Boot Verification Flow Showing the Verification Chain from ACM to the OS Loader

Option ROM Injection: Embedding rogue PCIe Option ROMs that execute during hardware initialization. Needs hardware access and ability to craft malicious ROMs. [14] Goal: stealthy execution below the OS level. Configuration Tampering: Modifying NVRAM variables such as Secure Boot or Boot Order. [13] Requires administrative privileges and firmware editing tools. Goal: weaken protections or redirect trust anchors.

B. Security Assumptions

Trusted Platform Module (TPM): We assume the platform includes a functioning TPM chip that is uncompromised and correctly measuring boot components. Integrity of Platform Configuration Registers (PCRs): PCR values generated during measured boot are considered trustworthy. Attackers are assumed unable to forge or manipulate these logs. Baseline Measurements: Defenders have access to known good baseline measurements of firmware and boot components, which serve as a reference for detecting tampering. Cryptographic Operations: Signature verification routines are assumed to function correctly unless explicitly bypassed by an attacker. Keys used in Secure Boot (DB and DBX) are valid and uncompromised at the start. Hardware Integrity: The CPU, chipset, and TPM are assumed to be physically intact and not subject to invasive hardware attacks. The focus remains on firmware/software level threats.

III. PROPOSED FRAMEWORK

The proposed framework builds on existing industry mechanisms to strengthen firmware and boot level security. It integrates Secure Boot enforcement, TPM based Measured Boot, and vendor specific protections into a layered defense model. Measured Boot and TPM Attestation: Cryptographic measurements of each boot component are recorded in TPM PCRs. [12], [11] These values can be attested to remote verifiers, enabling detection of unauthorized modifications against known baselines. Vendor and Commercial Solutions: OEMs and hardware vendors provide additional protections such as Intel Boot Guard and BIOS integrity checks [14], [15], [16]. Commercial platforms like Elysium extend visibility into firmware and peripheral components, offering monitoring and alerting capabilities. Framework Integration: By combining Secure Boot validation, TPM based measurement, and vendor monitoring tools [21], organizations can establish a practical defense strategy. This layered approach ensures that tampering attempts—whether through BIOS modification, DB/DBX manipulation, or Option ROM injection—are more likely to be detected.

IV. LIMITATIONS

Dependence on Vendor Keys: Secure Boot's strength is tied to how well vendors manage PK, KEK, DB, and DBX. [13] If these keys are outdated or mishandled, attackers can exploit them. Baseline Challenges: Measured Boot needs a clean baseline to compare against. [10] With so many hardware and firmware variations, keeping those baselines accurate and up to date is difficult. Detection Blind Spots: Sophisticated attackers can make subtle firmware changes that don't trigger obvious PCR deviations [18], slipping past detection. Peripheral Firmware Risks: Devices like GPUs, NICs, and storage controllers often run their own firmware outside the main boot path, leaving hidden areas unprotected. Operational Overhead: Attestation and monitoring solutions add cost and require expertise. [5] Smaller organizations may struggle to deploy or maintain them. Supply Chain Visibility Limits: These mechanisms protect runtime integrity but don't fully address upstream risks [22], such as compromised vendor build systems or malicious firmware updates.

V. CASE STUDIES

CVE 2024 7344: In 2024, ESET researcher Martin Smolár uncovered CVE 2024 7344 [3][4], a critical vulnerability in a Microsoft signed third party UEFI application. The issue arose because Microsoft had signed recovery tools from vendors that used a custom PE loader (reloader.efi) instead of the secure UEFI functions LoadImage and StartImage. This design flaw allowed attackers to load unsigned binaries from a crafted file (cloak.dat) during system startup, bypassing Secure Boot entirely. Root Cause: Microsoft's signing of third-party software without rigorous validation introduced a trusted but vulnerable component into the Secure Boot chain. Impact: Systems with the Microsoft UEFI CA 2011 certificate enrolled were exposed to bootkits such as BlackLotus or Bootkitty, even with Secure Boot enabled. Response: ESET reported the issue to CERT/CC in July 2024. Vendors patched their products, and Microsoft revoked the vulnerable binaries through DBX updates in January 2025's Patch Tuesday. Pre-Boot DMA Attack: In December 2025, researchers Nick Peterson and Mohamed Al Sharifi of Riot Games uncovered [6], [7] a set of vulnerabilities in UEFI firmware across multiple vendors. These issues were assigned several CVE identifiers, including CVE 2025 11901 and related IDs. The flaws stemmed from incorrect initialization of the IOMMU during the pre-boot phase. Firmware reported that "Pre-Boot DMA Protection" was enabled, but in reality, the protection was inactive. This allowed malicious PCIe or Thunderbolt devices with physical access to perform Direct Memory Access (DMA) operations before the operating system loaded. Impact: Attackers could inject code or read sensitive data directly from memory, bypassing Secure Boot checks and enabling stealthy bootkits. Response: Vendors including ASUS, ASRock, GIGABYTE, and MSI released firmware updates to correctly enforce IOMMU initialization and block DMA during pre-boot.

VI. CONCLUSION

Firmware and boot level protections such as Secure Boot and Measured Boot form a strong foundation against many classes of attacks, but they are not flawless. Their effectiveness depends on careful key management, reliable baselines, and visibility into peripheral firmware. To maintain resilience, organizations should regularly check for the latest UEFI and BIOS updates, apply vendor signed patches promptly, and monitor firmware integrity using available tools. Strengthening these practices will help ensure trust in the boot process and improve supply chain security against evolving threats.

REFERENCES

[1] National Institute of Standards and Technology (NIST), *BIOS Protection Guidelines*. Gaithersburg, MD, USA: NIST, 2011.

[2] Trusted Computing Group (TCG), TCG PC Client Platform Firmware Profile Specification. Beaverton, OR, USA: TCG, 2020.

[3] M. Smolár, "Under the cloak of UEFI Secure Boot: Introducing CVE-2024-7344," ESET Research, 2024. [Online]. Available: https://www.welivesecurity.com/en/eset-research/under-cloak-uefi-secure-boot-introducing-cve-2024-7344

[4] CERT Coordination Center (CERT/CC), "Vulnerability Note: Coordinated disclosure of CVE-2024-7344," CERT/CC, 2024.

[5] Eclypsium, “There’s a Hole in the Boot,” Eclypsium Blog, 2020. [Online]. Available: https://eclypsium.com/blog/theres-a-hole-in-the-boot

[6] The Hacker News, “New UEFI flaw enables early-boot DMA attacks on ASRock, ASUS, GIGABYTE, MSI motherboards,” Dec. 2025. [Online]. Available: https://thehackernews.com/2025/12/new-uefi-flaw-enables-early-boot-dma.html

[7] N. Peterson and M. Al-Sharifi, “Pre-boot DMA vulnerabilities in UEFI firmware (CVE-2025-11901 and related IDs),” Riot Games Security Research, Dec. 2025.

[8] UEFI Forum, UEFI Specification Version 2.9A: Boot Services. UEFI Forum, 2023. [Online]. Available: https://uefi.org/specs/UEFI/2.9A/07_Services_Boot_Services.html

[9] Intel Corporation, Secure the Network Infrastructure: Secure Boot Methodologies. Intel Developer Zone, 2023.

[10] National Institute of Standards and Technology (NIST), Platform Firmware Resiliency Guidelines, NIST Special Publication 800-193, 2018. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-193/final

[11] National Institute of Standards and Technology (NIST), Security and Privacy Controls for Information Systems and Organizations, NIST Special Publication 800-53 Rev. 5, 2020. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final

[12] Trusted Computing Group (TCG), TPM 2.0 Library Specification. Beaverton, OR, USA: TCG, 2019. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification

[13] Microsoft, “Securing hardware and firmware supply chains,” Microsoft TechCommunity, 2024.

[14] Intel Corporation, Intel Boot Guard Technology Overview, 2023. [Online]. Available: https://www.intel.com/content/www/us/en/security/boot-guard.html

[15] Advanced Micro Devices (AMD), AMD Secure Processor Technical Overview, 2023. [Online]. Available: https://www.amd.com/en/technologies/secure-processor

[16] Dell Technologies, BIOS Verification Whitepaper, 2022.

[17] J. Butterworth et al., “Firmware attacks and defenses,” in Proc. USENIX Security Symp., 2013.

[18] A. Kovah et al., “Firmware vulnerabilities in modern systems,” in Proc. Black Hat USA, 2015.

[19] R. Wojtczuk and J. Rutkowska, “Attacking Intel BIOS,” Invisible Things Lab, 2009.

[20] C. Kallenberg et al., “UEFI rootkits and mitigation strategies,” in Proc. CanSecWest, 2014.

[21] Open Compute Project, Supply Chain Technical Working Group – Secure Firmware Development Best Practices. Open Compute Project, 2019.

[22] Eclypsium, Supply Chain Security for the Modern Enterprise. Eclypsium Research, 2024. [Online]. Available: https://eclypsium.com/research

[23] W.-K. Chen, Linear Networks and Systems. Belmont, CA, USA: Wadsworth, 1993.

[24] A. Cichocki and R. Unbehaven, Neural Networks for Optimization and Signal Processing. Chichester, U.K.: Wiley, 1993.

[25] R. A. Scholtz, “The spread spectrum concept,” in Multiple Access, N. Abramson, Ed. Piscataway, NJ, USA: IEEE Press, 1993.

[26] G. Young, “Synthetic structure of industrial plastics,” in Plastics, 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964.

[27] J. Williams, “Narrow-band analyzer,” Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, 1993.

[28] N. Kawasaki, “Parametric study of thermal and chemical nonequilibrium nozzle flow,” M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[29] J. P. Wilkinson, “Nonlinear resonant circuit devices,” U.S. Patent 362412, Jul. 16, 1990.

[30] American National Standards Institute (ANSI), Letter Symbols for Quantities, ANSI Standard Y10.5-1968. [Online]. Available: http://www.halcyon.com/pub/journals/21ps03-vidmar