GGSEC RESEARCH GGSEC.DE

Firmware Persistence Techniques

Advanced UEFI & Embedded Persistence Analysis

Author: Maciej Gojny | Organization: GG Advanced IT Security

Version: 1.1 FINAL | Date: May 2026

Classification: TLP:WHITE – Public Release

EXECUTIVE ABSTRACT

This whitepaper provides a technical overview of firmware-based persistence techniques observed in modern cyber operations targeting UEFI firmware, embedded systems, storage controllers, and boot environments. Unlike traditional operating system persistence, firmware-level mechanisms may survive operating system reinstallation, disk replacement, and conventional remediation procedures. The document focuses on real-world attack methodologies, defensive strategies, forensic visibility, and platform integrity risks associated with modern firmware ecosystems.

Table of Contents

1. Introduction & Scope

Firmware persistence represents one of the most sophisticated categories of long-term system compromise in modern cybersecurity operations. While conventional malware operates within user-space or kernel-space boundaries, firmware-based threats target low-level platform initialization, boot-chain integrity, and embedded hardware logic. These persistence mechanisms may remain active even after disk formatting, operating system reinstallation, or complete storage replacement.

This document consolidates publicly documented attack methodologies, persistence mechanisms, notable threat campaigns, forensic indicators, and defensive mitigations relevant to enterprise and embedded security environments. It is intended for defensive security practitioners, firmware auditors, and enterprise security architects.

RESPONSIBLE DISCLOSURE: Certain low-level offensive implementation details have intentionally been omitted. This document is intended for defensive security, threat modeling, and firmware integrity assessment purposes only.

2. UEFI Fundamentals

2.1 What is UEFI?

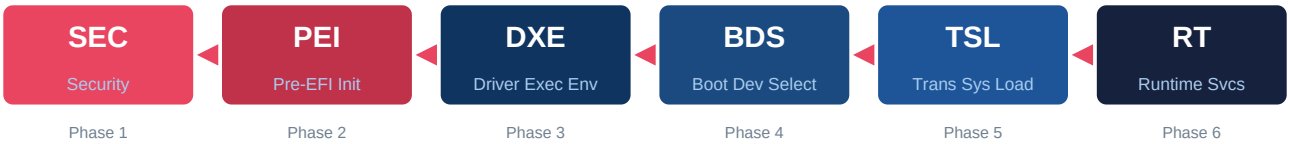
UEFI (Unified Extensible Firmware Interface) defines the interface between platform firmware and the operating system boot process, succeeding traditional BIOS architectures. Modern UEFI implementations use modular firmware components stored in SPI flash memory soldered directly to the motherboard. Because firmware operates independently from the operating system and storage media, compromise at this layer fundamentally undermines platform trust assumptions.

2.2 UEFI Boot Process – Complete Phase Reference

The full UEFI boot sequence consists of six phases. All six must be considered in firmware threat modeling:

Phase	Full Name	Description / Firmware Role	Attack Surface
SEC	Security	Root-of-trust, Cache-as-RAM init	Integrity violation; unauthorized code execution before root-of-trust established
PEI	Pre-EFI Init	Hardware init, memory detection, DRAM training	Malicious PEIMs
DXE	Driver Exec Env	Loads firmware drivers, publishes protocols	Malicious DXE driver insertion (MoonBounce)
BDS	Boot Device Sel	Boot policy, device enum, boot manager	Boot order manipulation, WPBT abuse
TSL	Transient Sys Load	OS loader execution, ExitBootServices() handoff	Bootkit injection before ExitBootServices (BlackLotus)

Phase	Full Name	Description / Firmware Role	Attack Surface
RT	Runtime Services	UEFI runtime services available post-boot (SetVariable)	Persistent NVRAM variable abuse



Note: TSL and RT phases are frequently omitted from vendor reference materials but represent critical attack surfaces. TSL is the execution window exploited by bootkits prior to kernel handoff.

3. Firmware Persistence Techniques

3.1 UEFI/BIOS Firmware Modification

Attackers modify firmware startup logic to execute malicious components before operating system initialization. A malicious UEFI DXE driver is inserted into the firmware image. Some variants embed an NTFS driver enabling write access to the OS partition directly from firmware, allowing malware to be re-dropped after OS reinstallation. This persistence survives disk formatting and storage replacement.

3.2 Windows Platform Binary Table (WPBT) Abuse

WPBT is an ACPI table used by firmware to pass a PE executable to Windows for execution during early boot. Microsoft requires digital signing but does not validate certificate revocation status. Available as an attack surface since Windows 8 (October 2012).

Property	Detail
Secure Boot bypass required	No – WPBT runs within OS context after boot
Code execution level	Kernel-level at startup via Session Manager
Persistence method	Binary dropped to disk, re-executed at each boot
Signing enforcement	Digital signature required; revocation not checked by default
Mitigation	WDAC policy restricting unsigned or revoked PE binaries

3.3 SSD/HDD Firmware Modification

Modified storage controller firmware can maintain persistence independently of the file system. Demonstrated capabilities include hidden Host Protected Area (HPA) or Device Configuration Overlay (DCO) storage invisible to the OS, boot-time code injection from the controller, and survival of full disk format and partition table erasure. Typically requires physical access or supply-chain compromise.

3.4 UEFI Variable / NVRAM Persistence

UEFI NVRAM variables accessible via GetVariable/SetVariable runtime services can store attacker-controlled data or shellcode read by a malicious DXE driver at next boot. This technique requires existing ring-0 access to write variables but provides stealthy persistence that survives OS reinstallation.

3.5 OEM Service Mechanisms

Several OEM firmware implementations have historically contained undocumented recovery or service mechanisms. While originally intended for vendor support operations, these introduce security risk if discovered and abused by threat actors.

Vendor	Mechanism	Status
ASUS	ALT+R at password prompt removes Administrator, User, and HDD passwords without authentication	Patched June 14, 2019
HP	After excessive failed UEFI password attempts the system generates a 32-bit value via rdtsc; a backdoor password is derived from that value	Patched – vendor advisory issued

4. UEFI Bootkits & Advanced Threats

UEFI bootkits target boot-chain integrity and manipulate EFI components, boot managers, or OS loading behavior before kernel initialization. They represent the most technically sophisticated category of firmware persistence.

4.1 BlackLotus

BlackLotus is the first publicly documented UEFI bootkit capable of bypassing Secure Boot on fully patched Windows 11 systems under specific conditions, exploiting CVE-2023-24932 (a vulnerable but still-signed Windows Boot Manager). Key characteristics:

- Leverages a legitimate Microsoft-signed boot manager containing CVE-2023-24932
- Downgrade attack: the old bootloader remains trusted by Secure Boot despite the known vulnerability
- Patches Windows kernel in-memory to disable Driver Signature Enforcement (DSE)
- Achieves Ring 0 privileges and deploys an unsigned kernel driver
- Has been commercially available on underground forums since early 2023
- The CVE-2023-24932 patch (May 2023) required manual opt-in activation, causing widespread deployment delays; full protection requires DBX revocation update to block the vulnerable bootloader
- Does not represent a universal Secure Boot bypass; properly maintained Secure Boot with current DBX remains effective

4.2 Downgrade Attack Technique

Secure Boot validates cryptographic signatures but does not enforce version currency by default. An attacker substitutes a vulnerable but validly-signed older boot component, rolling back platform security to an exploitable state. Microsoft's Secure Boot DBX (revocation database) updates are the primary mitigation but require active deployment by administrators.

5. Detection & Forensics

5.1 Indicators of Compromise (IOCs)

IOC Type	Description	Priority
ESP Modifications	Unexpected .efi binaries in EFI System Partition; modified BCD store entries or boot order	P1 – Critical
Secure Boot Changes	Secure Boot disablement; tampered DBX/DB entries; Secure Boot audit log anomalies	P1 – Critical
HVCI / BitLocker Disabled	Sudden deactivation of Memory Integrity or BitLocker without administrative change record	P1 – Critical
NVRAM Variable Changes	Unexpected creation or modification of UEFI variables (especially BootOrder, Boot####)	P2 – High
Hardware Anomalies	Firmware version mismatch vs. baseline; inconsistent hardware telemetry; unexpected PCI device	P2 – High

IOC Type	Description	Priority
Unknown DXE Modules	Unsigned or unrecognized DXE modules present in firmware image vs. known-good baseline	P2 – High

5.2 Detection & Analysis Tools

Tool	Primary Function	Operational Note
CHIPSEC	Firmware security auditing, SPI flash analysis, UEFI variable inspection	Requires kernel driver; may require Secure Boot disable in some configurations – assess risk before production use
UEFITool	UEFI firmware image parsing, module extraction, section inspection	Safe for offline analysis; no driver required
Binwalk	Firmware extraction, entropy analysis, file carving	Effective for embedded/IoT firmware; limited for complex UEFI images
Ghidra	Reverse engineering of DXE drivers, PEI modules, UEFI PE binaries	Use community UEFI Ghidra scripts for correct loader support
fwupd / LVFS	Linux firmware update and integrity baseline tracking	Useful for firmware version drift detection across fleet

5.3 Sigma Rule – Suspicious EFI File in ESP

Both conditions (path filter and SignatureStatus) are required. Omitting SignatureStatus generates excessive false positives on every legitimate firmware update.

```
title: Suspicious EFI File in ESP
id: a1b2c3d4-e5f6-7890-abcd-ef1234567890
status: experimental
description: Detects unsigned .efi binaries written to EFI System Partition
references:
- https://attack.mitre.org/techniques/T1542/003/
author: GGSEC Research
date: 2026-05
logsource:
product: windows
service: sysmon
detection:
selection:
EventID: 11
TargetFilename|contains: '\EFI\'
SignatureStatus: 'unsigned'
condition: selection
falsepositives:
- Legitimate firmware update tools writing signed EFI components
- OEM provisioning software
level: high
tags:
```

```
    - attack.persistence
    - attack.t1542.003
```

5.4 Splunk Query – Secure Boot Tampering

```
index=windows sourcetype=WinEventLog:System EventID=1035
| eval change_detail=coalesce(Param1, "unknown")
| where like(change_detail, "%SecureBoot%") OR like(change_detail, "%BitLocker%")
| stats count by ComputerName, change_detail, _time
| sort - _time
```

Supplement with WinEventLog:Security EventID 4616 and Microsoft-Windows-Kernel-Boot/Operational for comprehensive boot integrity coverage.

6. Defensive Mitigations

Effective defense against firmware persistence requires layered platform security. No single control is sufficient; combine hardware root-of-trust technologies, Secure Boot validation, firmware integrity monitoring, restrictive execution policies, and incident response procedures specifically addressing firmware compromise scenarios.

6.1 Platform Security Controls

Control	Description	Threat Covered
Secure Boot	Only cryptographically signed EFI components permitted during boot chain	Bootkits, unauthorized DXE drivers
TPM 2.0	Measures firmware integrity (PCR 0–7) during boot; seals BitLocker key to expected platform state	Firmware tampering detection
Intel Boot Guard	Hardware-enforced verification of Initial Boot Block (IBB) before firmware execution	Pre-SEC firmware modification
AMD PSP	AMD Platform Security Processor – hardware root-of-trust equivalent to Intel Boot Guard	Pre-SEC firmware modification
HVCI / Memory Integrity	Prevents unsigned kernel drivers from loading at runtime via hypervisor enforcement	BlackLotus kernel driver stage
EDR with firmware scanning	Endpoint agents (e.g. CrowdStrike Falcon, SentinelOne) with firmware telemetry and change detection	Post-compromise detection
AMI Tektagon XFR PFR	Platform Firmware Resilience per NIST SP 800-193: detect, protect, and recover firmware integrity	Critical infrastructure

6.2 WDAC for WPBT Mitigation

Microsoft recommends Windows Defender Application Control (WDAC) to restrict binaries delivered via WPBT. A restrictive WDAC policy prevents unsigned or policy-non-compliant binaries from executing even when dropped by firmware mechanisms.

- Create WDAC base policy using New-CIPolicy (Windows 10 1903+)
- Set enforcement mode: Enforced (not Audit) in production environments
- Block unsigned PE binaries in the WPBT execution path
- Deploy via GPO: Computer Configuration > Windows Settings > Security Settings > WDAC
- Monitor Microsoft-Windows-CodeIntegrity/Operational for policy violations

6.3 Firmware Update Hygiene

- Source firmware updates exclusively from official vendor sites or LVFS (Linux Vendor Firmware Service)
- Validate SHA-256 hash of firmware binary before flashing
- Prefer offline (pre-boot) update mode over OS-based flashing utilities
- Enable strong UEFI setup password; disable physical DMA ports where not required

- Apply Secure Boot DBX revocation updates promptly – these revoke exploitable signed bootloaders
- Track firmware versions across fleet using SCCM, Intune, or osquery for drift detection

7. Case Studies

7.1 LoJax – APT28 (Sednit / Fancy Bear)

Attribute	Detail
Discovery	2018 – ESET Research; first UEFI rootkit confirmed in the wild
Threat Actor	APT28 (Sednit, Fancy Bear) – attributed to Russian GRU military intelligence
Targets	Government organizations in the Balkans and Central-Eastern Europe
Technique	Modified SPI flash firmware image; malicious module survives disk replacement
Payload	Ring-0 agent drops Win32/XAgent user-space malware for C2 communication
Detection	CHIPSEC SPI flash integrity check; unexpected module present in firmware image

7.2 MosaicRegressor – Unknown APT-Linked Actor

Discovered in 2020 by Kaspersky GReAT. Attribution remains unconfirmed. MosaicRegressor used a modified version of the leaked Hacking Team UEFI implant framework. Targets included NGOs and diplomatic entities in Asia and Africa. The implant dropped a multi-stage downloader into the Windows filesystem at each boot.

7.3 MoonBounce – APT41

Attribute	Detail
Discovery	2021/2022 – Kaspersky GReAT
Threat Actor	APT41 (Winnti Group) – attributed to Chinese state-sponsored actors
Technique	Modified existing UEFI component (CORE_DXE) rather than adding a new module – significantly harder to detect than LoJax
Payload	Installer dropped into user space; establishes persistent C2 communication channel
Forensic evasion	No new files in the EFI System Partition; modification of a pre-existing firmware component only
Detection difficulty	Undetectable by conventional AV/EDR; requires CHIPSEC or direct firmware image inspection

7.4 BlackLotus – Multiple Actors (2023)

First UEFI bootkit capable of bypassing Secure Boot on fully patched Windows 11, exploiting CVE-2023-24932. Commercially available on underground forums from early 2023. Microsoft released a patch in May 2023 with opt-in activation, significantly delaying real-world protection. The bootkit disables HVCI and BitLocker before loading an unsigned kernel driver for persistent Ring-0 access.

8. Timeline of Notable Firmware Attacks (2018–2025)

Year	Campaign	Threat Group	Type	Key Novelty
2018	LoJax	APT28 (GRU)	UEFI rootkit	First UEFI rootkit confirmed in the wild; SPI flash modification survives disk replacement
2020	MosaicRegressor	Unknown (APT-linked)	UEFI rootkit	Based on leaked Hacking Team implant; multi-stage downloader dropped at each boot
2021	MoonBounce	APT41	UEFI component modification	Modified existing CORE_DXE; no new ESP files; highest stealth of documented UEFI implants
2022	CosmicStrand	Unknown (Chinese-linked)	UEFI rootkit	Targets GIGABYTE/ASUS consumer boards; kernel hook injected via CSMCORE driver
2023	BlackLotus	Multiple actors	UEFI bootkit	First Secure Boot bypass on patched Windows 11; CVE-2023-24932; sold commercially on underground forums
2024	Supply-chain (reported)	Nation-state suspected	Supply chain	Firmware modifications detected in enterprise hardware prior to delivery (investigative reporting)

Trend: Firmware persistence tooling is becoming increasingly commercialized and accessible, lowering the barrier to entry beyond nation-state actors.

9. Executive Recommendations

Prioritized Action List

Priority	Action	Rationale
P1 – Immediate	Enable Secure Boot + TPM 2.0 + BitLocker	Foundational platform integrity chain; mitigates the majority of known bootkit techniques
P1 – Immediate	Block unsigned/revoked binaries via WDAC policy	Closes WPBT abuse vector; prevents unsigned kernel driver loading
P1 – Immediate	Apply Secure Boot DBX updates (CVE-2023-24932 and subsequent)	Revokes exploitable signed bootloaders used by BlackLotus and related threats
P2 – Short-term	Run CHIPSEC audit on a representative device sample	Detects SPI flash modifications and firmware security misconfigurations

Priority	Action	Rationale
P2 – Short-term	Deploy Sysmon with Sigma rule (Section 5.3) for ESP monitoring	Provides early warning of unsigned EFI binary writes to the EFI System Partition
P2 – Short-term	Enable HVCI (Memory Integrity) on all endpoints	Prevents unsigned kernel driver loading; blocks the kernel-stage component of BlackLotus
P3 – Medium-term	Integrate firmware persistence scenarios into IR playbooks	Ensures responders know to image SPI flash and inspect firmware before reimaging
P3 – Medium-term	Implement firmware version tracking via osquery, SCCM, or Intune	Enables drift detection; a firmware baseline is required for anomaly identification
P3 – Medium-term	Consider AMI Tektagon XFR PFR for critical infrastructure devices	Provides NIST SP 800-193-compliant detect/protect/recover firmware resilience capability

For Individual Users

- Purchase hardware from established vendors with documented firmware security programs
- Update UEFI/BIOS exclusively from official vendor sources; validate SHA-256 checksum
- Enable Secure Boot and BitLocker with TPM binding
- Set a strong UEFI setup password and disable legacy boot options
- Apply Secure Boot DBX revocation updates promptly when released

For Organizations

- Implement EDR/XDR solutions with firmware scanning and telemetry capabilities
- Establish firmware version baselines and monitor for drift across the entire fleet
- Apply WDAC policies to restrict executed binaries at firmware-delivered paths
- Monitor EFI System Partition integrity and Secure Boot status via SIEM
- Include firmware persistence in threat modeling exercises and red team scope

10. Methodology & Limitations

This whitepaper is based on publicly available vendor documentation, peer-reviewed security research, conference proceedings (Black Hat, DEF CON, REcon), and threat intelligence reporting available as of May 2026. Content has been cross-referenced against MITRE ATT&CK and relevant vendor advisories.

Attribution claims reflect the security research community consensus at time of writing. Firmware-level threat attribution remains technically challenging; campaigns listed as 'unknown' or 'APT-linked' reflect genuine uncertainty rather than editorial omission.

Low-level offensive implementation details – including exploit code, SPI flash write procedures, and bootkit loader internals – have intentionally been omitted. This document is intended for defensive security practitioners, firmware auditors, and enterprise security architects.

11. Conclusion

Firmware persistence remains one of the most advanced and difficult-to-detect attack vectors in modern cybersecurity. Threat actors ranging from nation-state APT groups to commercially-motivated cybercriminals have demonstrated repeatable, operational firmware persistence capabilities. The barrier to entry continues to decrease as tooling becomes commercially available – evidenced by the public sale of BlackLotus in 2023 [6].

Unlike OS-level malware, firmware compromise operates below the visibility horizon of conventional endpoint security. It survives disk replacement, OS reinstallation, and standard forensic imaging. Remediation frequently requires physical component replacement or factory re-flash – a reality that incident response playbooks must explicitly address.

Secure Boot, TPM, HVCI, and WDAC – deployed together and maintained with current DBX revocation data – represent the minimum viable firmware security baseline for any enterprise environment in 2026. Firmware integrity must be treated as a strategic security requirement, not a niche hardening measure. Organizations should implement the P1 actions from Section 9 before progressing to advanced controls such as Platform Firmware Resilience [9] and application allowlisting via WDAC [12][13].

References

[1] ESET Research – A Machine-Learning Method to Explore the UEFI Landscape (2019)

[2] WeLiveSecurity – Needles in a Haystack: Picking Unwanted UEFI Components

[3] Kaspersky GReAT – MoonBounce: The Dark Side of UEFI Firmware (2022)

[4] Kaspersky GReAT – MosaicRegressor: Lurking in the Shadows of UEFI (2020)

[5] AMI – Keeping Platforms Secure from Advanced UEFI Firmware Exploits

[6] Huntress – CVE-2023-24932 Secure Boot Bypass Analysis (2023)

[7] Microsoft Security Response Center – CVE-2023-24932 Advisory

[8] MITRE ATT&CK – Persistence Techniques TA0003; Pre-OS Boot T1542

[9] NIST SP 800-193 – Platform Firmware Resiliency Guidelines

[10] UEFI Forum – UEFI Specification 2.10 (2022)

[11] Intel – Boot Guard Technology Overview

[12] Microsoft – Windows Defender Application Control (WDAC) Documentation

[13] chipsec – Platform Security Assessment Framework (GitHub: chipsec/chipsec)