WHITE PAPER

CYTEX™
AI-Powered Resilience Command Center

YellowKey & GreenPlasma

Windows BitLocker Bypass and SYSTEM Privilege Escalation

hello@cytex.io | https://cytex.io

YellowKey and GreenPlasma

Technical analysis of two unpatched Windows zero-days disclosed by Chaotic Eclipse / Nightmare-Eclipse on May 12, 2026, with reproducible exploit confirmation, the FsTx Transactional NTFS cross-volume defect, and operational mitigation guidance.

On **May 12, 2026 — one day after Microsoft's May Patch Tuesday**, a security researcher operating under the handles **Chaotic Eclipse**, **Nightmare-Eclipse**, and **Dead Eclipse** published proof-of-concept exploits on GitHub for two unpatched Windows vulnerabilities:

- **YellowKey** — a BitLocker full-disk encryption bypass exploitable via the Windows Recovery Environment (WinRE). Requires physical access. No CVE assigned.
- **GreenPlasma** — a local privilege escalation to SYSTEM via the Windows Collaborative Translation Framework (CTFMON / ctfmon.exe). Partial PoC published. No CVE assigned.

Neither has a patch or a CVE. Both have public proof-of-concept code released by the researcher on GitHub. The disclosure follows the same researcher's prior releases of **BlueHammer** (CVE-2026-33825), **RedSun**, and **UnDefend**, a pattern of post-Patch-Tuesday zero-day drops that began earlier in 2026 and that the researcher has explicitly committed to continuing.

Two operational realities make this disclosure unusually consequential:

One. YellowKey is reproducible. Independent confirmation by **Will Dormann** (Tharros Labs) and **Kevin Beaumont** validates that the attack works on updated Windows 11 systems. Dormann's analysis surfaced a deeper defect underneath the bypass, Transactional NTFS logs on one volume can modify files on a *different volume* during WinRE replay. That cross-volume primitive is, in Dormann's framing, a standalone vulnerability arguably more significant than the BitLocker bypass it enables.

Two. The researcher has signaled additional disclosures for the **next Patch Tuesday**. Each blog post is PGP-signed, tying every release to a verifiable identity. The threat model is no longer "researcher might disclose"; it is "researcher will disclose, on a schedule, timed to maximize the unpatched window."

The Disclosure Context

The researcher publicly states that the disclosures are retaliation for Microsoft's handling of prior vulnerability submissions. The relevant background, drawn from the researcher's own published posts and confirmed in independent reporting:

- The researcher previously disclosed three Microsoft Defender vulnerabilities and the BlueHammer / RedSun / UnDefend trio.
- BlueHammer (CVE-2026-33825) began being exploited in the wild four days *before* Microsoft's April Patch Tuesday fix shipped.
- Each release is PGP-signed, allowing public cryptographic verification of authorship.

- The researcher has explicitly committed to a continued cadence: *"Next Patch Tuesday will have a big surprise for you Microsoft. And remember, I never failed to deliver a promise."*
- The researcher has further warned that future releases may "drag other companies into this", meaning vendors beyond Microsoft may appear in subsequent disclosures.

YellowKey

Vulnerability Profile

Attribute	**Detail**
Name	YellowKey
CVE	None assigned (as of this writing)
Class	BitLocker full-disk encryption bypass via WinRE
Component	autofstx.exe inside the Windows Recovery Environment image
Root mechanism	Cross-volume Transactional NTFS (TxF) log replay
Affected platforms	Windows 11; Windows Server 2022; Windows Server 2025
Not affected	**Windows 10**
Required access	Physical access to powered-off or restartable device
Required materials	USB drive with crafted FsTx content (NTFS, FAT32, or exFAT formatting works)
TPM-only BitLocker	Vulnerable (default corporate laptop configuration)
TPM+PIN BitLocker	Researcher claims a variant bypasses this; Dormann's reproduction supports PIN as an effective mitigation against the public PoC
Public PoC	Released May 12, 2026 on GitHub by Nightmare-Eclipse
Independent reproduction	Confirmed by Will Dormann (Tharros Labs) and Kevin Beaumont
Microsoft patch	None at time of writing
Microsoft CVE	None assigned at time of writing

How YellowKey Works

YellowKey abuses a code path inside autofstx.exe, a binary that exists only in the Windows Recovery Environment image and not in standard Windows installations. During WinRE startup, autofstx.exe scans attached storage volumes for **Transactional NTFS (TxF) log data** stored under \System Volume Information\FsTx, and replays any logs it finds.

Transactional NTFS (TxF) was introduced in Windows Vista (2007) as a mechanism for atomic file-system operations, group multiple file changes into a transaction that can be committed or rolled back as a unit. Microsoft deprecated TxF years ago and has indicated it is on the path to removal but has not removed it. The FsTx directory and the replay logic remain present and active in WinRE.

The exploit chain:

Step 1 — Prepare the FsTx payload. The attacker stages a crafted System Volume Information\FsTx directory containing Transactional NTFS log entries on a USB drive. The PoC published on GitHub provides the exact directory contents. NTFS formatting works best; FAT32 and exFAT are also compatible.

Step 2 — Insert and reboot. The attacker connects the USB drive to a BitLocker-protected target and reboots into the Windows Recovery Environment. WinRE can be reached by holding Shift while clicking Restart, or by other standard WinRE entry paths.

Step 3 — autofstx.exe replays the crafted logs. During WinRE startup, autofstx.exe discovers the FsTx logs on the attached USB drive and replays them. **Here is the buried defect**: the TxF replay processes logs from one volume but applies file modifications to a *different volume* specifically, to the WinRE image mounted at X:.

Step 4 — winpeshl.ini is deleted on the recovery volume. The crafted logs cause X:\Windows\System32\winpeshl.ini to be deleted from the WinRE image. winpeshl.ini is the configuration file that tells WinRE which application to launch on startup.

Step 5 — WinRE falls back to cmd.exe. With winpeshl.ini missing, WinRE's startup logic falls back to launching cmd.exe directly. The user holds a specific key combination (the published PoC documents Ctrl) during the WinRE entry path to trigger the fallback.

Step 6 — Shell access to the decrypted volume. By the time WinRE is entered through the legitimate recovery path, the **TPM has already transparently unlocked the BitLocker-protected system volume**. The fallback cmd.exe shell now has full read/write access to the encrypted disk's contents, no recovery key, no PIN (in TPM-only configurations), no credentials of any kind.

Cross-Volume TxF Replay

Will Dormann's independent reproduction surfaced the technical layer underneath the BitLocker bypass:

"The result of this is that the X:\Windows\System32\winpeshl.ini is deleted, and when Windows Recovery is entered, rather than launching the actual Windows Recovery environment, it pops up a CMD.EXE. With the disk still unlocked." Will Dormann, Tharros Labs

Dormann's broader observation, posted to Mastodon and quoted across multiple outlets, is that the cross-volume modification capability is itself a vulnerability separate from YellowKey:

"The mere ability of the \System Volume Information\FsTx directory on one volume to modify the contents of another volume when replaying transactions is itself a standalone vulnerability."

This matters because the same TxF replay primitive may enable other attack chains beyond the BitLocker bypass, anywhere a privileged process replays TxF logs from attacker-controlled storage during boot or recovery flows.

The Trusted WIM Boot Defense That Should Have Caught This

Microsoft introduced a defense called **Trusted WIM Boot** alongside BitLocker recovery support. The mechanism stores a hash of the WinRE.wim image inside the BitLocker FVE (Full Volume Encryption) metadata block, which is itself integrity-protected via AES-CCM. During recovery boot, this hash is verified, if WinRE.wim has been tampered with (such as by deleting winpeshl.ini), the hash check should fail and the OS volume should remain locked.

Trusted WIM Boot would defeat YellowKey on devices where the BitLocker metadata contains the expected WIM hash entry. Per community analysis cited by ElcomSoft, **the metadata on a substantial share of laptops in the wild, roughly half, by one researcher's estimate, does not include the WIM hash**, meaning YellowKey works against those devices and fails against the rest. The factors that determine whether a given device has the WIM hash entry in its BitLocker metadata are not yet publicly documented.

The EFI Partition Variant

The researcher has documented a variant that requires no USB drive. If the attacker can physically open the device and extract the disk briefly, the FsTx folder can be copied directly to the **EFI System Partition**. The disk is reinserted, the device boots into recovery, and the exploit proceeds identically. This variant turns YellowKey from a "USB-stick attack" into an "anyone-with-a-screwdriver attack."

Dormann confirmed the USB-based variant in reproduction. He was not able to reproduce the EFI variant in his own testing, though that does not invalidate the researcher's claim that it works.

The Researcher's Backdoor Claim

The researcher publicly asserts that YellowKey functions as an intentional backdoor planted by Microsoft. The supporting argument:

- The autofstx.exe component responsible for the FsTx replay path **exists only inside the WinRE image** and is not present in normal Windows installations.
- A binary with the same name exists in normal Windows but lacks the functionality that triggers the BitLocker bypass.
- Online documentation of autofstx.exe and the FsTx replay behavior is, by the researcher's account and confirmed in independent searches, effectively absent.

Microsoft has not commented. Independent reporting has been careful to flag that the backdoor claim is the researcher's allegation and **has not been independently confirmed**. Kevin Beaumont's public comment was that *"BitLocker has a backdoor"*, agreement with the researcher's framing, but not corroborated by Microsoft documentation or third-party forensic analysis.

The honest defensive read: regardless of whether the placement was intentional or simply a deeply latent design flaw, the **behavior is real, the exploitation is reproducible, and the trust assumption that WinRE will not act as a privileged code path during physical attacks is broken**.

GreenPlasma

Vulnerability Profile

Attribute	Detail
Name	GreenPlasma
Researcher classification	"Windows CTFMON Arbitrary Section Creation Elevation of Privileges Vulnerability"
CVE	None assigned (as of this writing)
Class	Local privilege escalation to SYSTEM
Component	ctfmon.exe — Windows Collaborative Translation Framework Monitor
Affected platforms	Windows 11 and some Windows Server editions
Required access	Unprivileged local user account
Public PoC	Partial PoC released — final component for SYSTEM shell deliberately withheld
Microsoft patch	None at time of writing
Microsoft CVE	None assigned at time of writing

How GreenPlasma Works

ctfmon.exe is the **Collaborative Translation Framework Monitor**, a Windows process that runs as SYSTEM in every interactive session and handles text input services, alternative input methods, handwriting recognition, speech, and keyboard layouts. Because it runs as SYSTEM and accepts input from user-context processes by design, the trust boundary between unprivileged code and ctfmon is structurally narrow.

The exploit primitive published by the researcher:

Step 1 — Create an arbitrary memory section. An unprivileged user creates a memory **section object** (a kernel object that represents a shareable region of memory) inside a directory object that is writable by SYSTEM. The section's contents are attacker-controlled.

Step 2 — Manipulate the registry and permissions to redirect CTFMON. Through a chain of registry modifications and permission-rule manipulations, the exploit causes ctfmon.exe, running as SYSTEM, to interact with the attacker-created section object as if it were a trusted system resource.

Step 3 — SYSTEM trusts the attacker-controlled memory. Once CTFMON treats the section as legitimate, the attacker controls a piece of memory that a SYSTEM-context process fully trusts. This is the privilege escalation primitive.

Step 4 — Plant malicious shellcode or a fake DLL. From this trusted memory position, the attacker can plant shellcode for SYSTEM execution or place a fake DLL that CTFMON or another SYSTEM service will load.

The Withheld Component

The researcher deliberately released an **incomplete proof-of-concept**, the published code provides the primitive but stops short of the final step required to spawn a full SYSTEM shell. The researcher framed the omission as a "capture-the-flag challenge" for other researchers. The implication for defenders is straightforward: the published code is a working privilege escalation primitive; the final escalation step is documented but unimplemented; a competent exploit developer can fill the gap.

Independent analysis by security researcher **Het Mehta** confirmed the mechanism, arbitrary memory section creation in a SYSTEM-trusted directory object, registry-and-permission manipulation to redirect CTFMON, leading to attacker-controlled memory trusted at SYSTEM level.

Why CTFMON Is a High-Value Target

CTFMON has structural properties that make it an attractive privilege-escalation target:

- It runs as **SYSTEM** in every interactive user session.
- It is **always present** on Windows desktops; it is not optional.
- It accepts input from user-context processes by design.

- It is a relatively under-studied attack surface compared to higher-profile SYSTEM services.

Any vulnerability that breaches CTFMON's trust boundary turns an unprivileged process into SYSTEM. GreenPlasma is one such breach. Other CTFMON vulnerabilities, including the historical "CTF protocol" issues disclosed by Tavis Ormandy in 2019, suggest this surface produces these flaws periodically.

The Parallel Threat — Intrinsec Bootloader Downgrade

In the same disclosure window, the French security firm **Intrinsec** described a separate BitLocker attack chain based on **downgrading the bootloader version** via certificate trust manipulation:

- Secure Boot verifies the *signing certificate* of bootmgfw.efi but not its *version*.
- An attacker can load an older, vulnerable version of bootmgfw.efi signed with the trusted **PCA 2011** certificate.
- A second WIM image with a modified blob table is added to the SDI (System Deployment Image) file. The bootloader verifies the integrity of the first legitimate WIM but boots from the second attacker-controlled one, which contains a malicious WinRE image with cmd.exe.
- The attack executes in under five minutes on fully updated Windows 11 systems with physical access.

The Intrinsec chain is **separate from YellowKey** but reinforces the same broader concern: **the WinRE trust boundary, the bootloader certificate hierarchy, and BitLocker's recovery-path assumptions are all under active attack in the same disclosure window**. Microsoft has indicated plans to revoke the legacy PCA 2011 certificate in the near future, which would address Intrinsec's chain but not YellowKey.

Indicators of Compromise and Detection Signals

YellowKey and GreenPlasma do not produce traditional malware IOCs, no file hashes specific to the exploits beyond the published PoC code, no network indicators, no command-and-control infrastructure. The exploitation primitives use legitimate Windows mechanisms. **Detection is behavioral**.

YellowKey Detection Signals

Signal	Source / Mechanism
USB drives connected to a host containing the directory \System Volume Information\FsTx	Removable media auditing; USB device telemetry
autofstx.exe execution outside legitimate recovery operations	Process telemetry inside WinRE (rarely instrumented in enterprise EDR)

Signal	Source / Mechanism
Modification of X:\Windows\System32\winpeshl.ini on the WinRE volume	File integrity monitoring on the recovery partition
Unexpected cmd.exe launch as the first interactive process inside WinRE	WinRE process audit (typically requires custom instrumentation)
Unscheduled physical access events to laptops, desktops, or servers, particularly devices that boot into recovery shortly after	Physical security correlation; bootlog analysis
Devices that boot into WinRE multiple times in short succession without a corresponding maintenance event	Event log correlation; centralized boot telemetry

The published GitHub PoC repository name (Nightmare-Eclipse account) and the autofstx.exe filename are the closest things to file-level IOCs that exist for YellowKey. The repository itself is the canonical exploit reference; security teams should hash-fingerprint the published PoC artifacts and add them to threat intelligence feeds for retrospective hunting.

GreenPlasma Detection Signals

Signal	Source / Mechanism
Unprivileged processes creating section objects (NtCreateSection / ZwCreateSection) inside directory objects writable by SYSTEM	Sysmon Event ID 1 with command-line analysis; kernel object monitoring (limited coverage in most EDR products)
Registry modifications to keys that influence CTFMON's behavior, particularly permission and ACL changes	Sysmon Event ID 13/14 on relevant registry paths
ctfmon.exe loading DLLs from unusual paths, or interacting with memory regions backed by unprivileged user-created section objects	Sysmon Event ID 7 (Image Loaded) for ctfmon; advanced memory telemetry
Process tree anomalies where ctfmon.exe parents unexpected child processes after a registry/section manipulation sequence	EDR process-tree analysis

GreenPlasma's primitives are subtle and overlap with legitimate Windows behavior. Effective detection generally requires Sysmon with a tuned ruleset focused on object-manager and registry telemetry around CTFMON, plus an EDR layer capable of correlating section-object creation with downstream SYSTEM process behavior.

Mitigation and Defensive Actions

YellowKey — Highest Priority Mitigations

Enable BitLocker pre-boot authentication (PIN or USB key). TPM-only BitLocker is the default vulnerable configuration. Pre-boot PIN authentication is the most effective publicly documented mitigation against the public YellowKey PoC. Dormann's reproduction confirms PIN as effective against the released exploit, though the researcher claims a private variant bypasses TPM+PIN. Defenders should still implement PIN, it raises attacker cost and defeats the publicly available exploit code.

Group Policy path: Computer Configuration → Administrative Templates → Windows Components → BitLocker Drive Encryption → Operating System Drives → Require additional authentication at startup. Set "Configure TPM startup PIN" to "Require startup PIN with TPM".

Restrict USB boot in UEFI/BIOS firmware policy. Disable boot from removable media in firmware on all corporate devices. Protect firmware configuration with a strong BIOS/UEFI administrator password. This defeats the USB-stick variant of the exploit.

Set a BIOS/UEFI administrator password. Prevents attackers with physical access from re-enabling USB boot or modifying boot order.

Audit BitLocker configurations across the fleet. Identify which devices are TPM-only vs. TPM+PIN vs. TPM+USB. Prioritize PIN deployment to TPM-only devices in highest-risk roles (executives, field workers, devices that travel).

Treat lost or stolen devices as compromised. Standard incident-response posture: assume the BitLocker protection has failed on any device that left organizational control on a vulnerable Windows version. Rotate credentials and assume data exposure on the affected device.

Identify and prioritize high-physical-risk devices. Field laptops, shared workstations, devices in shared office space without lockable docks, kiosks, and conference-room machines are the immediate attack surface for YellowKey.

GreenPlasma — Mitigations

GreenPlasma is harder to compensate for at the configuration layer because CTFMON is an integral Windows component that cannot be disabled in standard configurations. Defensive actions:

Apply least-privilege controls aggressively. GreenPlasma requires an unprivileged local user foothold to escalate. Any control that prevents that initial foothold, endpoint protection against phishing, browser exploitation, Office macro execution, drive-by downloads, also prevents GreenPlasma from being reached.

Enable Sysmon with a CTFMON-aware ruleset. Detection coverage is the realistic compensating control. Instrument section-object creation, registry permission changes, and CTFMON process behavior.

Application allowlisting (AppLocker, WDAC). Limits the binaries an attacker can run to attempt the privilege escalation in the first place.

Restrict access to registry keys CTFMON depends on. Where operationally feasible, audit and tighten ACLs on registry paths that influence CTFMON behavior, though Microsoft has not published authoritative guidance on which keys are exploitation-relevant for GreenPlasma specifically.

Strategic Posture

Plan for the next disclosure. The researcher has committed to releasing additional exploits at the next Patch Tuesday. Defensive programs should assume one or more additional unpatched Windows zero-days will drop in the same window. Plan the patch-cycle response accordingly, the standard "wait for Patch Tuesday" cadence will be exposed during the disclosure week.

Treat WinRE as a privileged code path. Most enterprise security tooling does not instrument WinRE because WinRE is short-lived and operates before the normal OS environment. YellowKey demonstrates that WinRE is a viable attack surface, defensive instrumentation should extend into the recovery path where feasible, including boot log centralization and firmware-level event collection.

Why This Disclosure Pattern Is Operationally Different

Three properties of this disclosure pattern distinguish it from a typical zero-day release:

One — Schedule. The researcher releases on a *schedule*, immediately after Patch Tuesday, to maximize the unpatched window. This converts vulnerability disclosure from a random arrival process into a predictable adversarial event that defenders must plan against. The next disclosure is announced. The cadence is committed to publicly.

Two — Identity continuity. PGP-signed releases provide cryptographic continuity of authorship. Defenders know who is releasing what, when, and can monitor the canonical sources directly. This is unusual, most threat actors operate under disposable identities precisely to avoid this kind of monitoring surface.

Three — Trust assumption breakage. YellowKey specifically attacks the trust assumption that *native Windows components themselves are not the attack vector*. Most EDR and endpoint controls are calibrated to detect *malware*, not to detect *Windows behaving exploitably*. autofstx.exe is a signed Microsoft binary running in a Microsoft-controlled recovery environment processing Microsoft-defined transaction logs. There is no malware to flag. The signature chain validates. The behavior is the vulnerability.

Together, these properties produce a defensive problem that is *less about any single CVE* and *more about the architectural exposure to a hostile, predictable, identity-stable disclosure stream targeting the most-trusted components of the platform*.

Sources:

- Original disclosures: Chaotic Eclipse / Nightmare-Eclipse / Dead Eclipse, PGP-signed blog posts and GitHub PoC repository, published May 12, 2026.
- Will Dormann (Tharros Labs), public posts on Mastodon, May 12–13, 2026.
- Kevin Beaumont, public commentary, May 12–13, 2026.