INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-03-29 18:35 UTC

Warlock Ransomware Adopts Kernel-Level EDR Bypass: BYOVD Joins the Ransomware Playbook

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0065
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	EDR agents (specific vendors unspecified in available source data); Windows systems with kernel driver loading capability
Published	2026-03-21

Executive Summary

The Warlock ransomware group has added kernel-level driver abuse (BYOVD) to its attack chain, allowing it to disable endpoint detection tools before deploying ransomware. Any Windows environment relying on EDR as a primary defense layer is at elevated risk, as the technique specifically targets that control. Organizations should treat this as a signal that EDR alone is insufficient against current ransomware operations.

Technical Analysis

Warlock ransomware has incorporated Bring Your Own Vulnerable Driver (BYOVD) techniques into its post-exploitation chain. The technique loads legitimately signed but vulnerable kernel-mode drivers to achieve ring-0 execution (T1068, T1543.003), then abuses that privilege level to disable or blind EDR agents (T1562.001, T1562.006) before deploying ransomware encryption (T1486). Operating at ring-0 allows the attacker to interact directly with kernel structures that EDR sensors depend on, effectively removing visibility before the destructive payload runs. CWE-269 (Improper Privilege Management) and CWE-284 (Improper Access Control) apply to the driver abuse mechanism. Specific vulnerable drivers and associated CVEs leveraged by Warlock were not available in the sourced materials and are not included in this advisory. The BYOVD technique is documented across MITRE ATT&CK and has been tracked by CISA and Microsoft in multiple threat actor campaigns targeting EDR evasion. EDR vendors without kernel-level self-protection (tamper protection) are specifically at risk. No patch status is applicable to the campaign itself; remediation depends on driver blocklist enforcement and EDR hardening.

Action Checklist

1. Immediate: Verify EDR tamper protection is enabled on all endpoints, confirm agents cannot be stopped or unloaded by non-system processes or from kernel context.
2. Immediate: Cross-reference your environment's loaded drivers against Microsoft's recommended driver blocklist (WDAC) and the LOLDrivers project (loldrivers.io) to identify any known-vulnerable signed drivers present.
3. Detection: Hunt for driver load events involving signed drivers with known CVEs, query Sysmon Event ID 6 (driver loaded) or equivalent EDR telemetry for drivers matching the LOLDrivers blocklist hashes.
4. Detection: Review EDR health telemetry and agent check-in logs for any endpoints that went silent or reported degraded functionality in the past 30 days, EDR blind spots are a primary indicator of BYOVD activity.
5. Assessment: Audit Windows systems for kernel driver loading capability, identify any accounts or service contexts with SeLoadDriverPrivilege that are not operationally required and remove or restrict.
6. Hardening: Enforce Windows Defender Application Control (WDAC) or equivalent policy to block known-vulnerable drivers; Microsoft publishes a maintained blocklist suitable for direct deployment.
7. Long-term: Reduce reliance on EDR as a single defensive layer, ensure network-level detection (NDR), immutable logging to a SIEM, and behavioral analytics provide coverage independent of endpoint agent health.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to IR firm or CISO immediately if: (1) any endpoint reports EDR agent disabled or unresponsive during the past 30 days without documented cause, (2) any unsigned or LOLDriver-matched driver is found loaded, (3) any account outside System/Administrators holds SeLoadDriverPrivilege, or (4) WDAC/HVCI cannot be deployed organization-wide due to hardware/OS limitations (indicates defense gap against this threat).
Recovery Notes	Post-eradication: (1) Force full EDR agent reinstall and re-baseline tamper protection on all affected endpoints; validate agent check-in and health status for 48 hours before declaring recovery. (2) Deploy WDAC blocklist and validate no false-positives over 1-week test period. (3) Conduct forensic analysis of any endpoint where BYOVD activity was detected; submit disk image and memory capture to IR firm for kernel-level artifact recovery (rootkit detection, hidden process analysis, registry anomalies). (4) Review and harden network segmentation: isolate systems requiring driver loading capability to dedicated subnets with enhanced NDR and immutable logging. (5) Schedule post-incident review meeting to address detection gaps that allowed EDR bypass to occur undetected (if applicable) and update playbooks and thresholds.

Forensic Artifacts	Sysmon Event ID 6 (driver loaded) with full context: timestamp, driver name, file path, signer, hash, process context — 90-day retention minimum | Windows System Event Log (Event ID 7009 = service timeout, Event ID 7000 = service load failure) and Application log (Event ID 1000–1004 = Defender engine events) | Registry hives: HKLM\System\CurrentControlSet\Services (loaded driver registry keys with ImagePath, Start type, Type), HKLM\SOFTWARE\Microsoft\Windows Defender (tamper protection, definition version) | EDR agent health database or check-in log with per-endpoint timestamp, version, status, last successful telemetry upload — 30-day baseline minimum before incident | File hashes and digital signatures for all drivers present during incident window (Get-FileHash -Algorithm SHA256, Get-AuthenticodeSignature) cross-referenced against LOLDrivers project blocklist (downloaded timestamp, match status)

Per-Action IR Details

Immediate: Verify EDR tamper protection is enabled on all endpoints — confirm agents cannot be stopped or unloaded by non-system processes or from kernel context.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (Preparation phase — prevention and protection controls)

Controls: NIST 800-53 SI-7 (Information System Monitoring), NIST 800-53 AC-6 (Least Privilege), CIS Controls 8.3 (Address Unauthorized Software), CIS Controls 8.8 (Implement Security Awareness and Training)

Compensating: If EDR lacks native tamper protection: enable Windows Defender real-time protection with tamper protection via Group Policy (Computer Configuration > Administrative Templates > Windows Defender Antivirus > Tamper Protection). For non-domain systems, enforce via PowerShell: Set-MpPreference -DisableRealtimeMonitoring $false and lock via WMI class Win32_OSRecoveryConfiguration. Monitor via WMI query: Get-MpComputerStatus \| Select-Object RealTimeProtectionEnabled, IsTamperProtected.

Evidence: Capture EDR agent configuration baseline before verification: export agent version, tamper protection setting, and privilege level from EDR console. Document via screenshot or API export. Preserve Windows Defender tamper protection registry state: HKLM\SOFTWARE\Microsoft\Windows Defender\Features (TamperProtection = 5 is enforced). Capture before-state via Registry export: reg export HKLM\SOFTWARE\Microsoft\Windows /f before_defender_state.reg.

Immediate: Cross-reference your environment's loaded drivers against Microsoft's recommended driver blocklist (WDAC) and the LOLDrivers project (loldrivers.io) to identify any known-vulnerable signed drivers present.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.1 (Detection and Analysis — data sources and indicators)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 CM-3 (Configuration Change Control), CIS Controls 2.1 (Inventory and Control of Enterprise Assets), CIS Controls 4.1 (Establish and Maintain a Secure Configuration Management Process)

Compensating: Without WDAC: use Get-WindowsDriver -Online \| Select-Object Name, Version, Date to list loaded drivers. Cross-reference manually against LOLDrivers CSV (download from loldrivers.io/api/drivers.csv). For each driver, query registry for loaded driver details: Get-ItemProperty HKLM:\System\CurrentControlSet\Services\[drivername] \| Select-Object ImagePath, Start, Type. Compare ImagePath binary hashes against LOLDrivers blocklist using certUtil -hashfile [path] SHA256. Document findings in spreadsheet with driver name, file path, hash, and LOLDrivers match status.

Evidence: Export current driver load manifest before cross-referencing: wmic sysdriver list full /format:csv > drivers_baseline.csv. Capture Sysmon or WMI event log of all driver load events from past 90 days (Event ID 6 in Sysmon, or WMI class Win32_SystemDriver). Preserve file hashes and digital signatures: Get-Item [driver_path] \| Get-FileHash -Algorithm SHA256. Document registry keys for each driver: Export-Registry HKLM:\System\CurrentControlSet\Services > services_baseline.reg.

Detection: Hunt for driver load events involving signed drivers with known CVEs — query Sysmon Event ID 6 (driver loaded) or equivalent EDR telemetry for drivers matching the LOLDrivers blocklist hashes.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.2 (Detection and Analysis — analysis techniques)

Controls: NIST 800-53 SI-4(1) (Automated Tools and Mechanisms), NIST 800-53 AU-6 (Audit Review, Analysis, and Reporting), CIS Controls 8.3 (Address Unauthorized Software), CIS Controls 8.8 (Implement Security Awareness and Training)

Compensating: If no SIEM: query Sysmon log directly on suspect endpoints using PowerShell: Get-WinEvent -LogName 'Sysmon/Operational' -FilterXPath '*[System[EventID=6]]' | Select-Object TimeCreated, Message. Export to CSV for manual review. Cross-reference driver hashes against LOLDrivers using a lookup table (download CSV, use PowerShell Join-Object to match). For EDR-less environments, use osquery (free, cross-platform): queries table 'drivers' and 'loaded_modules'. Export results hourly to local JSON and parse offline. Document any unsigned drivers or drivers with suspicious load timestamps (e.g., 3 AM, during business hours off-schedule).

Evidence: Preserve Sysmon operational log before hunting: wevtutil export-log Sysmon/Operational C:\forensics\sysmon_baseline.evtx. If available, export raw EDR telemetry (driver load events) with full context: timestamp, driver name, file path, hash, signer, privilege context, process initiator. Capture memory image of any endpoint where suspicious drivers are found: wmic pagefile list /format:list, then use DumpIt or WinPMEM if incident suspected. Preserve timeline of driver loads: extract Sysmon Event 6 with 1-hour granularity and correlate against EDR agent health (next step).

Detection: Review EDR health telemetry and agent check-in logs for any endpoints that went silent or reported degraded functionality in the past 30 days — EDR blind spots are a primary indicator of BYOVD activity.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2.3 (Detection and Analysis — indicator prioritization)

Controls: NIST 800-53 SI-4(5) (System-Generated Alerts), NIST 800-53 AU-6(3) (Audit Review, Analysis, Reporting — Automated Anomaly Reporting), CIS Controls 3.3 (Detect and Investigate Unauthorized Network Devices), CIS Controls 8.3 (Address Unauthorized Software)

Compensating: Without EDR console: manually audit check-in logs from central management server (if available). For on-premises antivirus: check Windows Update for Health (MDE for Windows 10+) via Get-MpComputerStatus -ErrorAction SilentlyContinue. Query event logs for EDR heartbeat failures: Get-WinEvent -LogName 'System' -FilterXPath '*[System[EventID=7009]]' (service start timeout). Export EDR/AV process logs: Get-WinEvent -LogName 'Application' | Where-Object {$_.ProviderName -like '*Defender*' -or $_.ProviderName -like '*[EDR_Vendor]*'} | Select-Object TimeCreated, Message | Export-Csv edroverview_30days.csv. Create manual spreadsheet of endpoint health: hostname, last check-in time, agent version, status (online/offline/degraded). Flag any gaps > 24 hours or version mismatches.

Evidence: Export EDR agent health database/log before analysis: if using native EDR console, export health summary as CSV. Preserve system event logs from all endpoints: wevtutil export-log System C:\forensics\system_30days.evtx /l:MM/DD/YYYY /query:*[System[TimeCreated[@SystemTime>'[timestamp-30d]']]]. Capture Windows Defender/security event logs (Event ID 1000 = engine started, 1001 = engine stopped, 3002 = signature update failed). Preserve EDR agent execution logs if available: look for agent process (e.g., msedgeagent.exe, crwd.exe) in parent process chain and command-line args via Sysmon Event ID 1. Document any process termination events targeting EDR processes (Sysmon Event ID 5).

Assessment: Audit Windows systems for kernel driver loading capability — identify any accounts or service contexts with SeLoadDriverPrivilege that are not operationally required and remove or restrict.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (Preparation — access control)

Controls: NIST 800-53 AC-2 (Account Management), NIST 800-53 AC-6 (Least Privilege), NIST 800-53 AC-6(1) (Least Privilege — Authorize Access), CIS Controls 5.1 (Establish and Maintain an Inventory of Accounts), CIS

Controls 5.3 (Disable Unused Accounts)

Compensating: Use whoami /priv to enumerate current user privileges (requires elevated prompt). For full audit: create a PowerShell script querying all local and domain accounts. SeLoadDriverPrivilege is assigned via Group Policy or local security policy. Check local policy: secpol.msc > User Rights Assignment > 'Load and unload device drivers'. Compare against authorized list (typically System, Administrators, and service accounts only). For domain: query GPO: Get-GPOReport -All -ReportType Html | Select-String 'SeLoadDriverPrivilege'. Audit Active Directory service accounts: Get-ADUser -Filter 'userAccountControl -band 512' | Select-Object SamAccountName, Enabled. Document all accounts with privilege and business justification. Remove privilege via: Remove-LocalGroupMember -Group '[group_with_privilege]' -Member '[user_to_remove]' or via GPO security policy update.

Evidence: Export security policy baseline before audit: secedit /export /cfg C:\forensics\security_policy_baseline.inf. Capture current privilege assignments via GPO: Get-GPResult -Computer [hostname] -User [username] -ReportType Html -Path C:\forensics\gp_report.html. Document all local and domain accounts with SeLoadDriverPrivilege: use Get-LocalGroupMember or Get-ADGroupMember (domain). Preserve token-level privilege information for any suspicious process execution: use Sysmon Event ID 1 with LogonGuid to correlate process start with account privilege context. Extract from security log (Event ID 4688, Include Command Line enabled).

Hardening: Enforce Windows Defender Application Control (WDAC) or equivalent policy to block known-vulnerable drivers; Microsoft publishes a maintained blocklist suitable for direct deployment.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2.1 (Preparation — hardening controls)

Controls: NIST 800-53 CM-6 (Configuration Settings), NIST 800-53 CM-7 (Least Functionality), NIST 800-53 SI-7(1) (Software, Firmware, and Information Integrity — Integrity Checks), CIS Controls 4.1 (Establish and Maintain a Secure Configuration Management Process), CIS Controls 4.7 (Enforce Configuration Management Through Version Control)

Compensating: If WDAC deployment is not feasible: use WDIF (Windows Defender Exploit Guard) via Group Policy or via cmdline. Alternatively, use AppLocker for code integrity: Group Policy > Computer Configuration > Windows Settings > Security Settings > Application Control Policies > AppLocker. Create rules to block drivers by path pattern (e.g., C:\temp*.sys, C:\downloads*.sys). For vulnerable driver blocklist: download Microsoft WDAC blocklist from Microsoft Endpoint Manager. Convert to WDAC policy using ConvertTo-CIPolicy. For non-WDAC environments: use Device Guard/Hypervisor Code Integrity (HVCI) if CPU supports it (check via Get-CimInstance -ClassName Win32_ComputerSystem | Select-Object HyperVRequirementsMet). Enable via Registry: HKLM\SYSTEM\CurrentControlSet\Control\DeviceGuard\Scenarios\HypervisorEnforcedCodeIntegrity = 1. Test against vulnerable driver hashes from LOLDrivers blocklist.

Evidence: Baseline current WDAC policy state before deployment: Get-CimInstance -ClassName Win32_CodeIntegrityPolicy to check if policy is enabled. Export existing WDAC policies: Copy-Item 'C:\Windows\System32\CodeIntegrity\' -Recurse -Destination C:\forensics\wdac_baseline\. Capture driver load attempts that would be blocked: enable WDAC audit mode initially (not enforce) to log violations without blocking. Export Code Integrity operational log: Get-WinEvent -LogName 'Microsoft-Windows-CodeIntegrity/Operational' | Export-Csv wdac_audit.csv. Document baseline vulnerable driver inventory from LOLDrivers before policy enforcement: exported as drivers_vulnerable.csv. Create policy test environment first: deploy WDAC to test group of 5-10 systems, monitor for 1 week, validate no legitimate driver load failures, then roll to production.

Long-term: Reduce reliance on EDR as a single defensive layer — ensure network-level detection (NDR), immutable logging to a SIEM, and behavioral analytics provide coverage independent of endpoint agent health.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4.2 (Post-Incident Activities — lessons learned)

Controls: NIST 800-53 SI-4 (Information System Monitoring), NIST 800-53 AU-2 (Audit Events), NIST 800-53 AU-4 (Audit Log Storage Capacity), NIST 800-53 SC-7(3) (Boundary Protection — Access Points), CIS Controls 8.2 (Protect Against Malware), CIS Controls 8.6 (Protect Against Supply Chain Software Risks)

Compensating: For teams without commercial NDR/SIEM: deploy open-source Zeek (formerly Bro) for network-level detection at network boundaries. Configure Zeek to monitor for suspicious driver load patterns (kernel API calls visible

via DNS exfiltration, lateral movement, known C2 IP/domain traffic). Set up rsyslog on a centralized log server (immutable, append-only, no write permission for application accounts). Forward Windows event logs via WinLogBeat or native syslog-ng. Use Splunk Free (500MB/day cap) or ELK Stack (free, open-source) for SIEM function. For behavioral analytics without EDR: deploy osquery (open-source) on all endpoints; forward system state snapshots hourly to central log aggregator; use anomaly detection on process creation patterns, privilege escalation, and driver loads. Create detection rules for: (1) unsigned driver loads, (2) driver loads from temp/downloads, (3) SeLoadDriverPrivilege usage by non-service accounts, (4) EDR process termination or pause events. Document detection logic in Sigma format for portability across SIEM platforms.

Evidence: Post-incident forensics must preserve: (1) full packet capture (PCAP) from incident window if NDR available, or Zeek logs showing connection timeline, (2) centralized syslog or SIEM archive with immutable proof of delivery (no gaps, no overwrites), (3) baseline network telemetry (DNS queries, HTTP user-agents, outbound connections to known-good destinations) for future anomaly detection, (4) system event log archive from all endpoints during incident, (5) memory dumps from any endpoint where driver load was detected (send to forensic lab for kernel debugger analysis), (6) full disk image of any system showing unauthorized driver load, prioritizing System32/drivers/ directory, Registry hives (SYSTEM, SOFTWARE, SECURITY), and MFT. Preserve chain of custody documentation for all evidence.

Detection Guidance

Primary behavioral indicator: EDR agent process termination or loss of telemetry immediately preceding ransomware activity. Query EDR health dashboards for agents that stopped reporting. For Sysmon-instrumented environments, query Event ID 6 (ImageLoaded) for driver loads occurring outside of patch windows or from unusual paths, correlated with Event ID 7045 (new service installed) for driver-as-service installation. Windows Security Event ID 7045 and 4657 (registry value modification) can indicate driver installation. Cross-reference loaded driver hashes against LOLDrivers (loldrivers.io), this community-maintained list catalogs signed but vulnerable drivers known to be abused in BYOVD campaigns. Also monitor for SeLoadDriverPrivilege use by non-system accounts via Event ID 4673. Alert on any process with kernel handle access to EDR driver objects. Specific Warlock IOCs (driver hashes, C2 infrastructure) were not available in the sourced materials and are not included in this advisory.

Framework Mappings

MITRE-ATTACK

- **T1195.003** — Compromise Hardware Supply Chain
- **T1486** — Data Encrypted for Impact
- **T1562.001** — Disable or Modify Tools
- **T1562.006** — Indicator Blocking
- **T1068** — Exploitation for Privilege Escalation
- **T1543.003** — Windows Service

NIST-800-53R5

- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **AC-6** — Least Privilege
- **SC-7** — Boundary Protection

- **SI-2** — Flaw Remediation
- **AC-3** — Access Enforcement
- **IR-4** — Incident Handling
- **SC-13** — Cryptographic Protection
- **SI-4** — System Monitoring

OWASP-TOP10-2021

- **A01:2021** — Broken Access Control

CIS-V8

- **5.4**
- **6.8**
- **6.1**
- **6.2**
- **8.2** — Collect Audit Logs

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(7)(ii)(A)** — Data Backup Plan
- **164.312(e)(1)** — Transmission Security

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **A.8.24** — Use of cryptography

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195.003	Compromise Hardware Supply Chain	Initial-Access
T1486	Data Encrypted for Impact	Impact
T1562.001	Disable or Modify Tools	Defense-Evasion
T1562.006	Indicator Blocking	Defense-Evasion

Technique ID	Technique Name	Tactic
T1068	Exploitation for Privilege Escalation	Privilege-Escalation
T1543.003	Windows Service	Persistence

Sources

Source	URL	Tier
Security News	https://www.darkreading.com/threat-intelligence/warlock-ransomware-...	T3
Attacking EDRs Part 3: One Bug to Stop them all - InfoGuard Labs	https://labs.infoguard.ch/posts/edr_part3_one_bug_to_stop_them_all/	T3
Aurora EDR - Malware and Vulnerabilities Analysis - Medium	https://medium.com/@enyel.salas84/aurora-edr-deb34e0a1e55	T3
ThreatsDay Bulletin: OAuth Trap, EDR Killer, Signal Phishing ...	https://thehackernews.com/2026/03/threatsday-bulletin-oauth-trap-ed...	T3
How unusual is it for SaaS vendors not to use EDR on servers?	https://www.reddit.com/r/sysadmin/comments/1lhy9dx/how_unusual_is_i...	T3

Generated 2026-03-29 18:35 UTC by TJS Security Command Center