

They Got In Through SonicWall. Then They Tried to Kill Every Security Tool



huntress.com/blog/encase-byovd-edr-killer



Summary

In early February 2026, Huntress responded to an intrusion where threat actors leveraged compromised SonicWall SSLVPN credentials to gain initial access to a victim network. Once inside, the attacker deployed an EDR killer that abuses a legitimate Guidance Software (EnCase) forensic driver with a revoked certificate to terminate security processes from kernel mode, a technique known as Bring Your Own Vulnerable Driver (BYOVD).

The attack was disrupted before ransomware deployment, but the case highlights a growing trend: threat actors weaponizing signed, legitimate drivers to blind endpoint security. The EnCase driver's certificate expired in 2010 and was subsequently revoked, yet Windows still loads it, a gap

in Driver Signature Enforcement that attackers continue to exploit.

Technical analysis

[Huntress Managed SIEM](#) ingested SonicWall telemetry from the victim environment, which proved critical in detecting and piecing together the intrusion timeline. Analysis of VPN authentication logs revealed the threat actor successfully authenticated to the SonicWall SSLVPN from a malicious external IP address in February 2026.

Notably, the logs also captured a denied portal login attempt from 193.160.216[.]221 just one minute prior; the account lacked privileges for portal access from that location. The attacker then successfully authenticated via VPN client from a different IP 69.10.60[.]250.

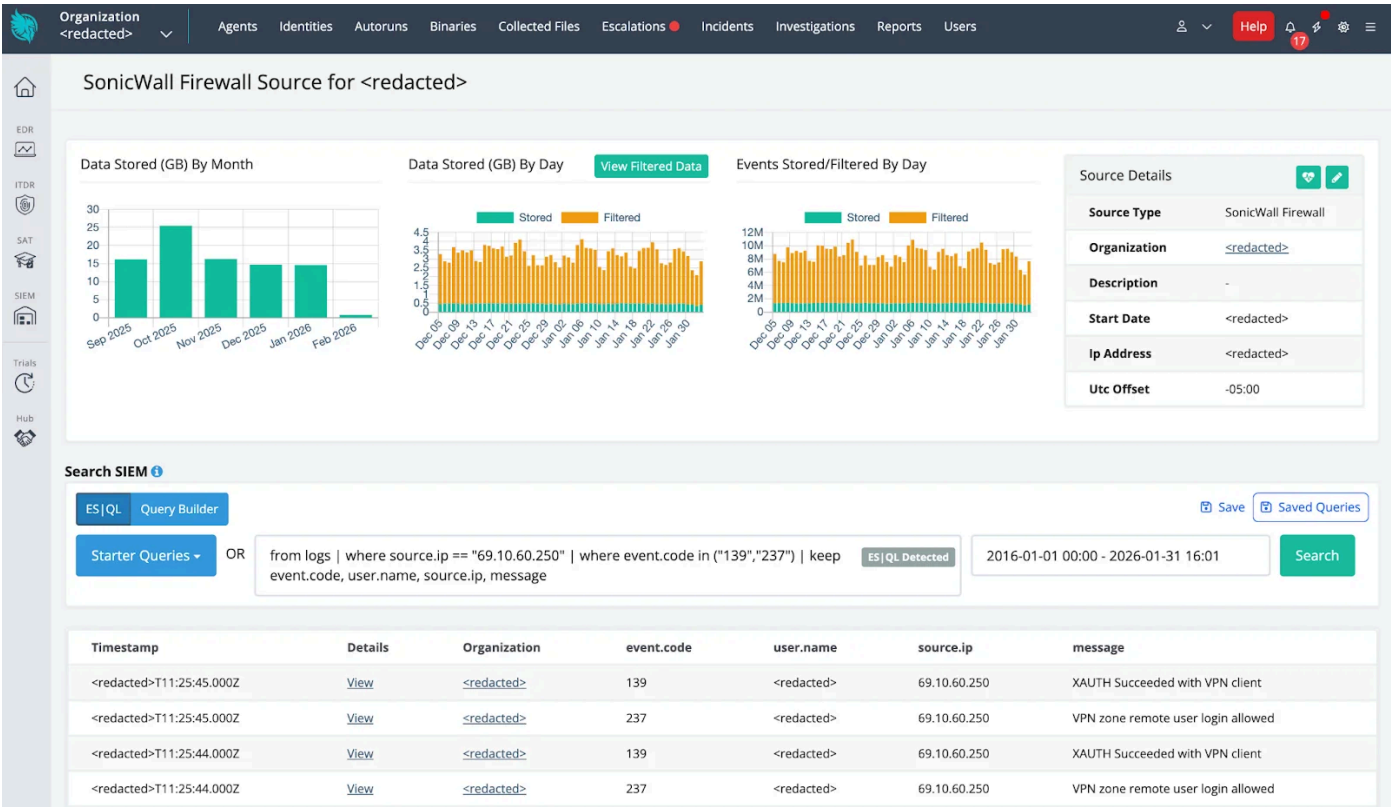


Figure 1: Huntress SIEM dashboard showing successful authentication events from threat actor’s IP

Correlation between SIEM telemetry and endpoint visibility allowed Huntress to quickly identify the threat, quarantine affected systems, and provide the client with actionable remediation recommendations.

	Account: Redacted Account Organization: Redacted Org Host:  Redacted Host	Rule Name: Redacted SIEM Detection Event Category: authentication  Event Provider: Microsoft-Windows-Security-Auditing  Event Code: 4779  Remote IP Lookup: Remote IP Logs: SIEM Logs: SIEM Dashboard  Source Host Name: WIN-redacted Source IP: 192.168.7.250 User Name: Administrator
	Account: Redacted Account Organization: Redacted Org Host:  Redacted Host	Rule Name: Redacted EDR Detection </> Username: Administrator Process Name: C:\Users\Public\svchost.exe   Command Line: "C:\Users\Public\svchost.exe" 

Figure 2: SIEM correlation with EDR detection

Discovery

Once authenticated, the threat actor conducted aggressive network reconnaissance. SonicWall IPS alerts captured ICMP ping sweeps, NetBIOS name request probes, and SMB-targeted activity, including SYN flood behavior at rates exceeding 370 SYNs/second.

Execution

EDR killer

The binary is a 64-bit Windows executable that deploys a kernel driver to terminate security software processes. It masquerades as a legitimate firmware update utility.

Embedded BYOVD

The EDR killer binary leverages a custom encoding scheme to conceal its embedded kernel driver payload. Rather than storing the driver as raw bytes or using traditional encryption, the malware developers opted for a wordlist-based substitution cipher that transforms each byte of the driver into an English word. This technique is particularly effective at evading static analysis tools, as the encoded payload appears to be nothing more than innocuous English text scattered throughout the binary's data section.

At the heart of this encoding scheme lies a 256-word dictionary embedded within the binary. Each word in this dictionary corresponds to a specific byte value; the word's index in the array equals the byte it represents. For example, "about" sits at index 0 and decodes to 0x00, "block" at index

```

.rdata:000000014002DDC8 aBlack db 'black',0 ; DATA XREF: .data:000000014009E268↓
.rdata:000000014002DDCE align 10h
.rdata:000000014002DDD0 aBleed db 'bleed',0 ; DATA XREF: .data:000000014009E270↓
.rdata:000000014002DDDB align 8
.rdata:000000014002DDD8 aBlock db 'block',0 ; DATA XREF: .data:000000014009E278↓ [77] - 0x4D (M)
.rdata:000000014002DDDE align 20h
.rdata:000000014002DDE0 aBlood db 'blood',0 ; DATA XREF: .data:000000014009E280↓
.rdata:000000014002DDE6 align 8
.rdata:000000014002DDE8 aBlow db 'blow',0 ; DATA XREF: .data:000000014009E288↓
.rdata:000000014002DDED align 10h
.rdata:000000014002DDF0 aBlue db 'blue',0 ; DATA XREF: .data:000000014009E290↓
.rdata:000000014002DDF5 align 8
.rdata:000000014002DDF8 aBoard db 'board',0 ; DATA XREF: .data:000000014009E298↓
.rdata:000000014002DDFE align 20h
.rdata:000000014002DE00 aBoat db 'boat',0 ; DATA XREF: .data:000000014009E2A0↓
.rdata:000000014002DE05 align 8
.rdata:000000014002DE08 aBody db 'body',0 ; DATA XREF: .data:000000014009E2A8↓
.rdata:000000014002DE0D align 10h
.rdata:000000014002DE10 aBoil db 'boil',0 ; DATA XREF: .data:000000014009E2B0↓
.rdata:000000014002DE15 align 8
.rdata:000000014002DE18 aBone db 'bone',0 ; DATA XREF: .data:000000014009E2B8↓
.rdata:000000014002DE1D align 20h
.rdata:000000014002DE20 aBook db 'book',0 ; DATA XREF: .data:000000014009E2C0↓
.rdata:000000014002DE25 align 8
.rdata:000000014002DE28 aBorder db 'border',0 ; DATA XREF: .data:000000014009E2C8↓
.rdata:000000014002DE2F align 10h
.rdata:000000014002DE30 aBorn db 'born',0 ; DATA XREF: .data:000000014009E2D0↓
.rdata:000000014002DE35 align 8
.rdata:000000014002DE38 aBorrow db 'borrow',0 ; DATA XREF: .data:000000014009E2D8↓
.rdata:000000014002DE3F align 20h
.rdata:000000014002DE40 aBoth db 'both',0 ; DATA XREF: .data:000000014009E2E0↓ [90] - 0x5A (Z)
.rdata:000000014002DE45 align 8
.rdata:000000014002DE48 aBottle db 'bottle',0 ; DATA XREF: .data:000000014009E2E8↓
.rdata:000000014002DE4F align 10h
.rdata:000000014002DE50 aBottom db 'bottom',0 ; DATA XREF: .data:000000014009E2F0↓

```

4/14

```

.rdata:000000014002E3B0 aBlockBothChoice db 'block both choice about act about about about active about about '
.rdata:000000014002E3B0 ; DATA XREF: .data:off_14009E810↓o
.rdata:000000014002E3F1 db 'about escape escape about about country about about about about a'
.rdata:000000014002E432 db 'bout about about bell about about about about about about about a'
.rdata:000000014002E473 db 'bout about about about about about about about about about about '
.rdata:000000014002E4B4 db 'about about about about about about about about about about about '
.rdata:000000014002E4F5 db ' about about about about about about about about about about all ar'
.rdata:000000014002E536 db 'ea cover all about cost again desk army country above bleed desk '
.rdata:000000014002E577 db 'army boil bring brother by arm busy buy business bright buy brave'
.rdata:000000014002E5B8 db ' burn arm break brave bus bus business cake arm bread breath arm '
.rdata:000000014002E5F9 db 'buy call bus arm brother bus arm better blow body arm burn busine'
.rdata:000000014002E63A db 'ss breakfast breath bad air air age art about about about about a'
.rdata:000000014002E67B db 'bout about about by dirty child eight bear cost dry cold bear cos'
.rdata:000000014002E6BC db 't dry cold bear cost dry cold bear cost drop cold air cost dry co'
.rdata:000000014002E6FD db 'ld below aunt clever cold bath cost dry cold below aunt cloud col'
.rdata:000000014002E73E db 'd bath cost dry cold along buy children cold basket cost dry cold'
.rdata:000000014002E77F db ' along buy cloud cold bean cost dry cold below aunt children cold'
.rdata:000000014002E7C0 db ' bean cost dry cold along buy choose cold basket cost dry cold al'
.rdata:000000014002E801 db 'ong buy coat cold bean cost dry cold along buy close cold bean co'
.rdata:000000014002E842 db 'st dry cold boat brother break bring bear cost dry cold about abo'
.rdata:000000014002E883 db 'ut about about about about about about blue between about about b'
.rdata:000000014002E8C4 db 'reakfast change add about bowl class am bit about about about abo'
.rdata:000000014002E905 db 'ut about about about about effect about around about ago across a'
.rdata:000000014002E946 db 'fter about about difficult about about about and about about abou'
.rdata:000000014002E987 db 't about about about after along above about about along about abo'
.rdata:000000014002E9C8 db 'ut about about above about about about about about about along ab'
.rdata:000000014002EA09 db 'out about about across about about add about about about add abou'
.rdata:000000014002EA4A db 't about about activity about across about about about about about'
.rdata:000000014002EA8B db ' about ball above about about active about about cause city above'
.rdata:000000014002EACC db ' about above about about about about about active about about abo'
.rdata:000000014002EB0D db 'ut about about about along about about about about about about ab'
.rdata:000000014002EB4E db 'out about along about about about about about about about along abo'
.rdata:000000014002EB8F db 'bout about about about about about about about about about along abo'
.rdata:000000014002EBD0 db 'about about about about about about about about about about break'

```

Figure 4: Wordlist-encoded driver payload

The decoding routine implements a straightforward lookup algorithm. It initializes a string parser to tokenize the encoded blob word by word, then iterates through each extracted word. For every word encountered, the function performs a linear search through the 256-entry wordlist, comparing the current word against each dictionary entry until it finds a match. Once matched, the word's index becomes the decoded byte value, which is appended to an output buffer.

The first few words of the encoded payload demonstrate this transformation clearly: "block both choice about" decodes to "4D 5A 90 00", the infamous MZ signature of a DOS executable header. This confirms the decoded output is a valid Windows PE file. The decoded driver is then written to "C:\ProgramData\OEM\Firmware\OemHwUpd.sys", after which the binary applies anti-forensic measures: setting the file attributes to hidden and system to conceal it from casual directory browsing, and copying timestamps from the legitimate ntdll.dll to make the malicious driver blend in with system files.

```

11 hFile = CreateFileW(L"C:\\Windows\\System32\\ntdll.dll", 0x80000000, 1u, 0, 3u, 0, 0);
12 if ( hFile == (HANDLE)-1LL )
13     return 0;
14 GetFileTime(hFile, &CreationTime, &LastAccessTime, &LastWriteTime);
15 CloseHandle(hFile);
16 wstring_ptr = (const WCHAR *)mw_get_wstring_ptr(a1);
17 hObject = CreateFileW(wstring_ptr, 0x100u, 3u, 0, 3u, 0, 0);
18 if ( hObject == (HANDLE)-1LL )
19     return 0;
20 SetFileTime(hObject, &CreationTime, &LastAccessTime, &LastWriteTime);
21 CloseHandle(hObject);
22 return 1;
23 }

```

Figure 5: Timestomping: The binary copies timestamps from ntdll.dll to the dropped driver to blend in with legitimate system files

This encoding approach offers several advantages for the attacker. Traditional string-based detection fails because the payload contains no suspicious API names, file paths, or binary signatures in its encoded form. Entropy-based analysis designed to detect packed or encrypted malware can be ineffective here; the wordlist-encoded payload registers around 4 bits/byte, well below the 7-8 bits/byte threshold that typically flags suspicious encrypted or compressed content.

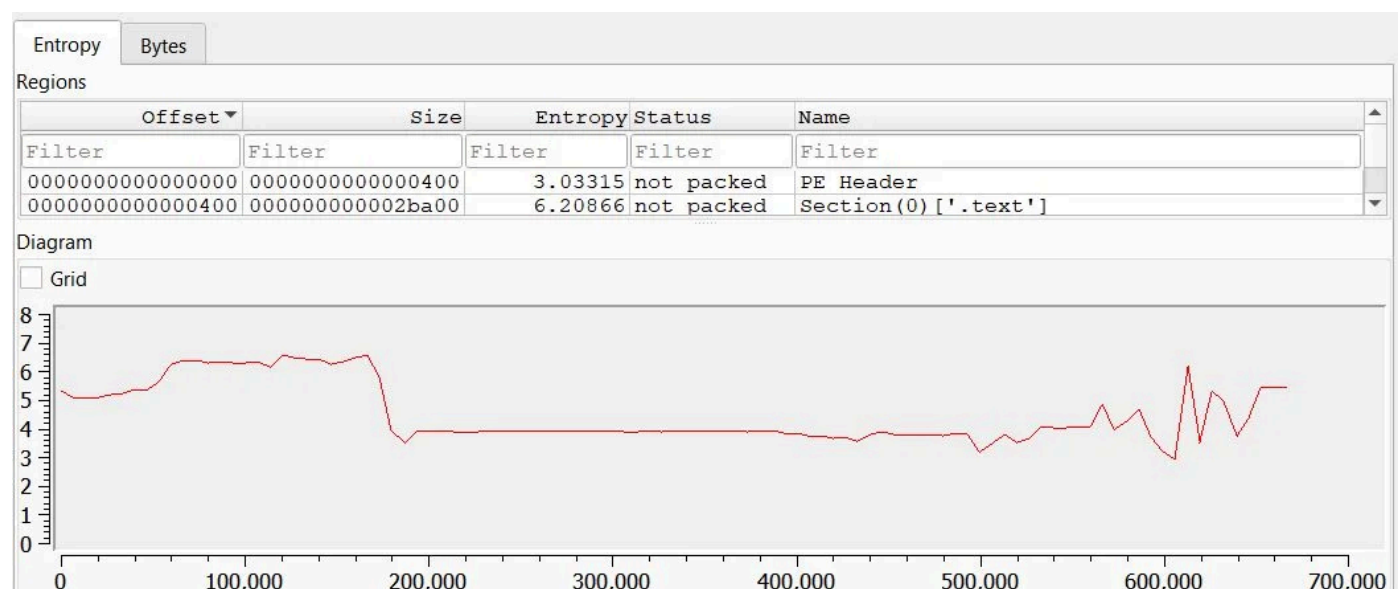


Figure 6: Entropy of the svchost.exe binary (Source: DiE)

Target identification

The binary maintains a list of 59 target process names that are hashed at initialization using FNV-1a (seed 0x811C9DC5). During runtime, it enumerates running processes via CreateToolhelp32Snapshot, converts each name to lowercase, computes its FNV-1a hash, and

compares against the pre-computed target hashes. This allows fast integer comparison rather than 59 string comparisons per process.

The kill loop runs continuously with a 1-second sleep interval, ensuring any security process that restarts is immediately terminated again.

The EDR killer targets 59 processes spanning major endpoint security vendors:

Vendor	Targeted Processes
Microsoft Defender	msmpeng.exe, mssense.exe, nissrv.exe, sensendr.exe
CrowdStrike	csfalconservice.exe, csagent.exe
SentinelOne	sentinelagent.exe, sentinelstaticengine.exe, sentinelhelper.exe, sentinelservice.exe
Carbon Black	cb.exe, cbdefense.exe, repmgr.exe
Palo Alto Cortex	cyveraservice.exe, traps.exe, cyserver.exe
Elastic Security	elastic-endpoint.exe, elastic-agent.exe
Cybereason	cybereason.exe, minionhost.exe, crsensor.exe
Cylance	cylancesvc.exe, cylanceui.exe
Symantec/Broadcom	ccsvchst.exe, smc.exe, symcorpui.exe
McAfee/Trellix	mcshield.exe, mfevtps.exe, mfeesp.exe, mfevtp.exe
Trend Micro	tmntsrv.exe, ntrtscan.exe, pccntmon.exe, tmlisten.exe
Sophos	savservice.exe, sophoshealth.exe, sophossp.exe, sophosfilescanner.exe, sophosclean.exe, sophosquery.exe
Kaspersky	avp.exe, kavsvc.exe
ESET	ekrn.exe, egui.exe
Bitdefender	bdagent.exe, vsserv.exe, bdservice.exe
Fortinet	forticlient.exe, fortiesnac.exe
Malwarebytes	mbam.exe, mbamservice.exe
Avast/AVG	avastsvc.exe, avgsvc.exe

Vendor	Targeted Processes
FireEye/Trellix	xagt.exe
Tanium	taniumclient.exe
Qualys	qualysagent.exe
Rapid7	ir_agent.exe
Splunk	splunkd.exe

[view raw list_of_processes.md](#) hosted with ♥ by [GitHub](#)

Notably absent from the kill list? **Huntress**. While the EDR killer targets nearly every major EDR and AV vendor on the market, the Huntress agent was not among the 59 processes targeted for termination.

Persistence

Once the kernel driver is decoded and written to disk, the binary establishes persistence by registering it as a Windows kernel service. This allows the driver to be loaded into the kernel and survive system reboots.

The binary first checks if a service with the same name already exists; if so, it deletes the old service before creating a new one. It then registers the driver with the Service Control Manager using carefully chosen names designed to blend in with legitimate OEM software:

Service Name: OemHwUpd

Display Name: OEM Hardware HAL Service

Description: Manages hardware abstraction layer compatibility

Binary Path: C:\ProgramData\OEM\Firmware\OemHwUpd.sys

Service Type: Kernel Driver

Start Type: Demand Start

Key	Value
Detector(s)	<redacted>
Created At	<redacted> 12:37:12 UTC ()
Updated At	<redacted> 12:37:12 UTC ()
From Survey At	<redacted> 12:37:11 UTC ()
Huntress Category	N/A
Persistence	Service (Kernel Driver) - OEM Hardware HAL Service
Binary	3111f4d7d4fac55103453c4c8adb742def007b96b7c8ed265347df97137fbee0
Virus Total	View Details
File Path	c:\programdata\oem\firmware\oemhwupd.sys 
Name	OemHwUpd 
Path	c:\programdata\oem\firmware\oemhwupd.sys 
State	RUNNING 
Status	OK 
Command	\\?\C:\ProgramData\OEM\Firmware\OemHwUpd.sys 
Display Name	OEM Hardware HAL Service 
Binary Mod Time	2026-01-22 00:06:09 EST 
Binary Create Time	2026-01-22 00:06:09 EST 
Event Summary	At <redacted> 12:37:12 UTC, a Windows Service with the name "OEM Hardware HAL Service" was detected on the endpoint.

Figure 7: Driver persistence (Source: Huntress portal)

Driver Signature Enforcement

Since Windows Vista x64, Microsoft has required all kernel-mode drivers to be digitally signed by a trusted Certificate Authority. This policy, known as Driver Signature Enforcement (DSE), prevents unsigned or tampered drivers from loading into the kernel. When a driver is loaded, Windows verifies that the digital signature is cryptographically valid and chains back to a Microsoft-trusted root certificate.

However, there's a critical gap in this verification process: the kernel does not check Certificate Revocation Lists (CRLs). While Windows can verify that a signature is mathematically valid and was issued by a trusted authority, it cannot determine whether that certificate has since been revoked by the issuing CA. This limitation exists for practical reasons: drivers load early in the

boot process before network services are available, and CRL checks would significantly impact boot performance. Even when a CRL is manually imported into local certificate storage, the kernel bypasses this check entirely.

The EnCase driver exploited in this attack perfectly illustrates this gap. Its signing certificate expired in January 2010 and was subsequently revoked, yet Windows still accepts the signature because the cryptographic verification succeeds. From the kernel's perspective, the signature remains valid.

Microsoft's blocklist approach

Rather than implementing CRL checking, Microsoft introduced the Vulnerable Driver Blocklist, a hash-based deny list that prevents specific known-bad drivers from loading. Since the Windows 11 22H2 update, this blocklist is enabled by default when Hypervisor-Protected Code Integrity (HVCI) or Memory Integrity is active. The blocklist is updated with each major Windows release, typically once or twice per year.

This approach offers precision that certificate revocation cannot: blocking individual driver hashes avoids the collateral damage of revoking an entire certificate, which could break legitimate software from the same vendor. However, it's inherently reactive; a driver must first be identified as vulnerable, reported to Microsoft, and added to the blocklist before protection takes effect.

The certificate for the “EnPortv.sys” driver that the TA deployed has been expired for over 15 years and was subsequently revoked, yet Windows accepts it because the kernel only validates the cryptographic chain, not expiration or revocation status.

Once loaded, the driver exposes an IOCTL interface that allows usermode processes to terminate arbitrary processes directly from kernel mode. This bypasses all usermode protections, including Protected Process Light (PPL) that typically guards critical system processes and EDR agents.

Despite the invalid certificate status, Windows loads this driver because:

The July 29, 2015, exception: Starting with Windows 10 version 1607, Microsoft required all new kernel drivers to be signed via the Hardware Dev Center. However, to maintain backward compatibility, Microsoft created an exception: drivers signed with certificates issued before July 29, 2015, that chain to a supported cross-signed CA are still permitted to load. The EnCase driver's certificate was issued on December 15, 2006, well before this cutoff.

Valid timestamp: The driver contains a timestamp from Thawte Timestamping CA (a VeriSign service). When code is signed with a timestamp, Windows validates the signature against the time the signature was created, not the current date. Because the driver was timestamped while the certificate was still valid ([before January 31, 2010](#)), the signature remains valid indefinitely, even though the certificate has since expired.

Trusted certificate chain: The signature chains through VeriSign Class 3 Code Signing 2004 CA to Microsoft Code Verification Root, which Windows trusts for kernel-mode code signing.

Once loaded, the driver exposes an IOCTL interface that allows usermode processes to terminate arbitrary processes directly from kernel mode. This bypasses all usermode protections, including Protected Process Light (PPL) that typically guards critical system processes and EDR agents.

The EnCase forensic driver (EnPortv.sys) exposes **18+ IOCTL** functions designed for forensic acquisition, including process termination (KillProc), DKOM process hiding (HideProc/UnhideProc), kernel-mode file deletion (DeleteFile), physical memory access (OpenPhysicalMemory/ReadPhysicalMemory), process memory reading (PidMemory), and VAD enumeration (GetVadList).

Out of all these capabilities, the threat actors in this incident only used IOCTL 0x223078 (KillProc) to terminate EDR processes. The usermode component opens a handle to the driver's device (\\.\OemHwUpd) and sends target PIDs via DeviceIoControl. The driver, executing in kernel mode, calls ZwOpenProcess with PROCESS_TERMINATE access followed by ZwTerminateProcess. When kernel-mode code calls Zw* functions, the wrapper sets PreviousMode to KernelMode, signaling that parameters come from a trusted source, Windows skips the security validation it would enforce on usermode callers.

```
2 bool __fastcall mw_send_kill_pid_to_driver(__int64 a1, unsigned int a2)
3 {
4     DWORD BytesReturned; // [rsp+44h] [rbp-24h] BYREF
5     __int64 InBuffer; // [rsp+48h] [rbp-20h] BYREF
6
7     if ( *(_QWORD *)(a1 + 48) == -1 )
8         *(_QWORD *)(a1 + 48) = CreateFileW(L"\\\\.\\OemHwUpd", 0xC0000000, 3u, 0, 3u, 0, 0);
9     if ( *(_QWORD *)(a1 + 48) == -1 )
10         return 0;
11     InBuffer = a2;
12     BytesReturned = 0;
13     return DeviceIoControl(*(HANDLE *)(a1 + 48), 0x223078u, &InBuffer, 8u, &InBuffer, 8u, &BytesReturned, 0);
14 } // IOCTL 0x223078: Send PID to driver - driver terminates process via ZwTerminateProcess
```

Figure 8: Usermode component sending target PID to kernel driver via IOCTL 0x223078 (Source: svchost.exe)

```

10     DbgPrint("KillProc(1351):");
11     DbgPrint("KillProc Beg-devExt[%8.8X] Pid[%8.8X]\n", a1, (_DWORD)a2);
12 }
13 KeAttachProcess(*(PRKPROCESS*)(a1 + 1648920));
14 if ( dword_1F128 )
15 {
16     DbgPrint("KillProc(1353):");
17     DbgPrint("Attached to PID[%8.8X] Process!\n", (_DWORD)a2);
18 }
19 ClientId.UniqueProcess = a2;
20 ClientId.UniqueThread = 0;
21 ObjectAttributes.Length = 24;
22 memset(&ObjectAttributes.RootDirectory, 0, 20);
23 ObjectAttributes.SecurityDescriptor = 0;
24 ObjectAttributes.SecurityQualityOfService = 0;
25 v4 = ZwOpenProcess(&ProcessHandle, 0x40u, &ObjectAttributes, &ClientId);
26 if ( v4 < 0 )
27 {
28     if ( dword_1F128 )
29     {
30         DbgPrint("KillProc(1374):");
31         DbgPrint("OpenProcess Failed on Pid[%8.8X]\n", (_DWORD)a2);
32     }
33 }
34 else
35 {
36     v4 = ZwTerminateProcess(ProcessHandle, 0);
37     if ( v4 < 0 )
38     {
39         if ( dword_1F128 )

```

Figure 9: KillProc implementation (from driver)

Conclusion

This intrusion demonstrates how BYOVD attacks have become a standard component of the modern ransomware playbook. By repurposing a legitimate forensic tool signed over 15 years ago, threat actors can bypass Driver Signature Enforcement, and gain kernel-level capabilities to terminate any security process on the system.

The attack chain with compromised VPN credentials, EDR killer deployment follows a well-established pattern seen in ransomware precursor activity.

Recommendation

- **Enable MFA on all remote access services** - the initial compromise was possible because the SonicWall SSLVPN account lacked multi-factor authentication
- **Review VPN authentication logs** for anomalous login patterns and denied access attempts preceding successful authentication
- **Enable HVCI / Memory Integrity** - ensures Microsoft's Vulnerable Driver Blocklist is enforced
- **Alert and monitor services** with names mimicking OEM/hardware components created outside normal software deployment
- **Deploy [Microsoft's recommended driver block rules](#)** via WDAC to block known vulnerable drivers
- **Enable ASR rule "[Block abuse of exploited vulnerable signed drivers](#)"** to prevent applications from writing vulnerable drivers to disk

Indicators of compromise

Item	Description
C:\ProgramData\OEM\Firmware\OemHwUpd.sy	Driver Path
OemHwUpd	Service Name
OEM Hardware HAL Service	Service Display Name
69.10.60[.]250	Threat actor IP
193.160.216[.]221	Threat actor IP
3111f4d7d4fac55103453c4c8adb742def007b96b7c8ed265347df97137fbee0	EnCase driver (OemHwUpd.sys)
6a6aaeed4a6bbe82a08d197f5d40c2592a461175f181e0440e0ff45d5fb60939	EDR killer binary (svchost.exe)

