Abusing Vulnerable Drivers to Disable System Defenses: The Case of ThrottleStop

Cristian H. M. Souza[1,2] **, Eduardo O. Chavarro**[1] **, Daniel M. Batista**[2]

[1]Global Emergency Response Team (GERT), Kaspersky
[2]Department of Computer Science, University of São Paulo

{cristian.souza, eduardo.ovalle}@kaspersky.com, batista@ime.usp.br

Abstract. *`ThrottleStop.sys` is a legitimate driver that has been abused by adversaries in recent years as part of Bring Your Own Vulnerable Driver (BYOVD) attacks to elevate privileges on Windows systems and weaken their defenses by terminating processes associated with Endpoint Detection and Response (EDR) tools. In this paper, we provide a detailed technical analysis of CVE-2025-7771, a vulnerability affecting this driver that was discovered by Kaspersky's Global Emergency Response Team (GERT). We also present an in-depth analysis of an advanced artifact that abuses this vulnerability in real-world attacks. Finally, we review the adversary's tactics, techniques, and procedures (TTPs) and propose best practices for mitigating this type of threat.*

1. Introduction

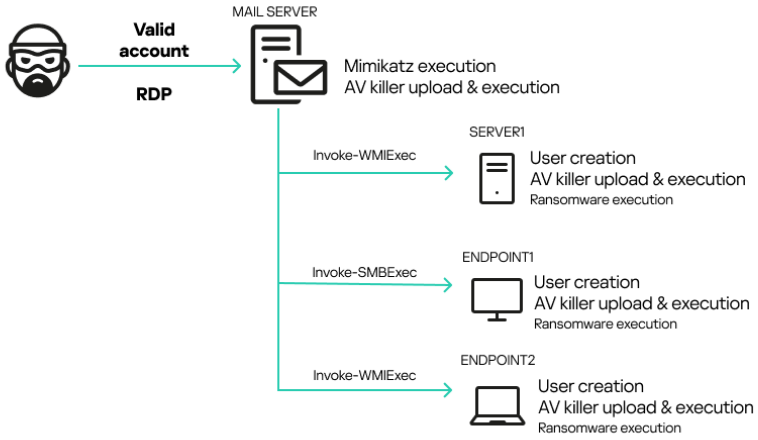
Vulnerable drivers are a known and dangerous problem in the cybersecurity industry [Gupta et al. 2022], and are actively abused by malware [Souza et al. 2025b]. Since they typically run in protection ring 0, or kernel mode, they can provide unrestricted access to critical operating system functions and libraries [Monzani et al. 2026, Soliven and Kimura 2022]. With this in mind, in the context of Windows systems, ransomware operators have adopted a logical tactic in recent years: killing Endpoint Detection and Response (EDR) solutions using vulnerable drivers before deploying the malware itself. This enables unrestricted malicious activity after the system's defenses have been disabled.

What makes this tactic particularly concerning is that hundreds of legitimate, signed, vulnerable drivers are available for download and can be shipped alongside specially crafted user-mode code designed to abuse privileged functions exposed by the vulnerable driver. Once adversaries obtain kernel-level access, they can terminate antivirus processes, install persistence mechanisms, and perform other operations within the compromised infrastructure [Souza et al. 2025a]. Additionally, detection can be very difficult, since the driver is signed by a legitimate and trusted vendor [Poslušnỳ 2022].

According to ESET, there are at least 54 EDR-killer programs based on the exploitation of vulnerable drivers in use by attackers as of 2026[1]. In an incident response case in Brazil, Kaspersky's Global Emergency Response Team (GERT) spotted an intriguing new antivirus (AV) killer software that has been circulating in the wild since at least October 2024[2]. This malicious artifact abuses the `ThrottleStop.sys` driver,

[1]https://www.eset.com/blog/en/business-topics/threat-landscape/what-are-edr-killers/
[2]https://securelist.com/av-killer-exploiting-throttlestop-sys/117026/

Figure 1. Incident overview



delivered together with the malware, to terminate numerous antivirus processes and lower the system's defenses.

This paper describes the abuse of the legitimate `ThrottleStop.sys` driver as part of a Bring Your Own Vulnerable Driver (BYOVD) attack chain used to disable security products before ransomware deployment. We analyze how the vulnerable driver was delivered, loaded, and abused by a user-mode component to access privileged kernel functions and terminate antivirus and EDR-related processes. We also discuss how this technique fits into the broader context of defense evasion, why signed vulnerable drivers remain a persistent operational risk, and which defensive measures can reduce the likelihood and impact of similar attacks.

2. Incident Description

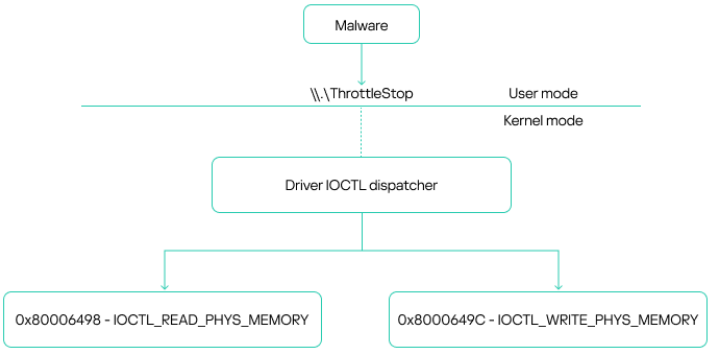
The adversary started with the use of valid administrative credentials, which allowed him to access a mail server through RDP from Belgium. After gaining access, the attacker used Mimikatz[3] to extract another user's NTLM hash and then performed pass-the-hash attacks with PowerShell-based Invoke-TheHash commands[4]. These commands were used to create local users on multiple machines, following a sequential naming pattern, such as `User1`, `User2`, and `User3`, while reusing the same password for all accounts.

The attacker uploaded several artifacts to the compromised mail server, including the AV killer, initially placing them under `C:\Users\Administrator\Music`. The same artifacts were later copied to other systems together with the ransomware sample, `haz8.exe`, under `C:\Users\UserN\Pictures`. Although Windows Defender initially contained the ransomware on some machines, the attacker later disabled the security solution. Figure 1 summarizes the incident flow and the main artifacts involved. In this case, the available evidence allowed us to reconstruct the attacker's workflow, although such artifacts are not always preserved in compromised environments.

This kind of attack highlights the importance of defense in depth. Although the

[3]https://github.com/gentilkiwi/mimikatz
[4]https://github.com/kevin-robertson/invoke-thehash

Figure 2. ThrottleStop device driver communication overview



organization had an AV in place, the attacker was able to use a valid account to upload an undetectable artifact that bypassed the defense. Such attacks can be avoided through simple security practices, such as enforcing the use of strong passwords and disabling RDP access to public IPs.

3. Technical Analysis

To disable the system's defenses, the attackers relied on two artifacts: `ThrottleBlood.sys` and `All.exe`. The first is a legitimate driver originally called `ThrottleStop.sys`, developed by TechPowerUp and used by the ThrottleStop app[5] for correcting CPU throttling issues. The second is the AV-killer itself, designed by the adversaries.

3.1. ThrottleBlood.sys Analysis

The driver involved in the incident has a valid certificate signed on 2020-10-06 20:34:00 UTC. When loaded, it creates a device at `.\\.\\ThrottleStop`, which is a communication channel between user mode and kernel mode. Communication with the driver is carried out via IOCTL calls, specifically using the Win32 `DeviceIoControl` function. This function enables the use of IOCTL codes to request various driver operations.

The driver exposes two vulnerable IOCTL functions: one that allows reading from memory and another that allows writing to it. Both functions use physical addresses. Importantly, any user with administrative privileges can access these functions, which constitutes the core vulnerability. An overview of this communication process is shown in Figure 2.

It then leverages the `MmMapIoSpace` function to perform physical memory access. This kernel-level API maps a specified physical address into the virtual address space, specifically within the MMIO (memory-mapped I/O) region. This mapping enables reads and writes to virtual memory to directly affect the corresponding physical memory. This type of vulnerability is well-known in kernel drivers and has been exploited for years, not only by attackers but also by game cheaters seeking low-level memory access [Gupta et al. 2022]. The vulnerability in `ThrottleStop.sys`, discovered

[5]https://www.techpowerup.com/download/techpowerup-throttlestop/

by Kaspersky GERT, has been assigned `CVE-2025-7771`[6].

3.2. EDR-killer Analysis

Our analysis began with a basic inspection of the file. First, we inspected its properties. While searching for relevant strings, we noticed a pattern: multiple antivirus process names inside the binary. We were able to map the security products targeted by the malware based on the process names hardcoded in the binary. The list shows that the artifact was designed to target a broad set of major antivirus and endpoint security vendors, including Avast, AVG Technologies, BitDefender, CrowdStrike, ESET, Kaspersky, McAfee/Trellix, Microsoft, Quick Heal Technologies, Symantec/Broadcom, Panda Security/WatchGuard, SentinelOne, and Sophos.

When the binary is executed, it first loads the `ThrottleBlood.sys` driver using Service Control Manager (SCM) API methods, such as `OpenSCManagerA()` and `StartServiceW()`. The AV killer needs the ThrottleStop driver to hijack kernel functions and enable the execution of kernel-mode-only routines from user mode. To do this, the malware first retrieves the base address of the currently loaded kernel and the addresses of the target functions to overwrite by using the undocumented `NtQuerySystemInformation` function. By passing the `SystemModuleInformation` flag, the function returns the list of loaded modules and drivers on the current system, including `ntoskrnl.exe`. Since the kernel base address changes because of Kernel Address Space Layout Randomization (KASLR), the malware must collect this information dynamically.

To perform read/write operations through `MmMapIoSpace`, the malware must also determine the physical address used by the kernel. This is achieved using the SuperFetch technique, implemented through the open-source superfetch project[7], which translates kernel virtual addresses to physical addresses by querying current memory ranges and pages through `NtQuerySystemInformation` with `SystemSuperfetchInformation`. Once the physical base address has been collected, the malware uses `NtAddAtom` as an indirectly callable syscall from user mode. By loading `ntoskrnl.exe` with `LoadLibrary`, it identifies the offset of `NtAddAtom`, calculates its kernel address, and uses the vulnerable IOCTL codes to read and write its physical memory. Finally, the AV killer writes a small shellcode that jumps to the desired kernel function and restores the original kernel code after execution to prevent system crashes. An overview of this process is presented in Figure 3.

Having obtained all the necessary information, the AV killer starts a loop to find target processes using the `Process32FirstW()` and `Process32NextW()` API calls. As mentioned earlier, the list of targeted security software, such as `MsMpEng.exe` from Windows Defender, is hardcoded in the malware. The program checks all running processes against this list and, if any match is found, it uses the vulnerable driver to call the `PsLookupProcessById` and `PsTerminateProcess` kernel functions, as exposed in Figure 4.

If a process is successfully killed, the malware displays a message in the console with the name of the terminated process. Most antivirus software available today attempt

[6]https://nvd.nist.gov/vuln/detail/CVE-2025-7771
[7]https://github.com/jonomango/superfetch

Figure 3. Kernel code injection diagram

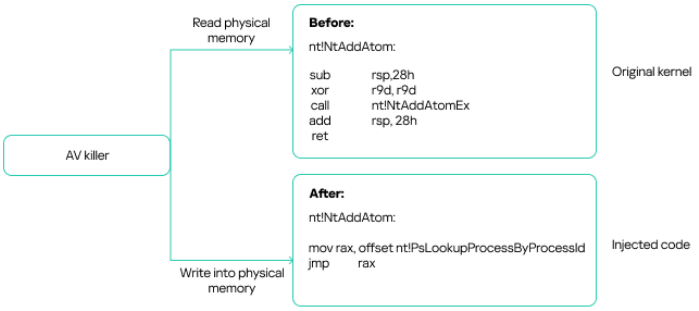
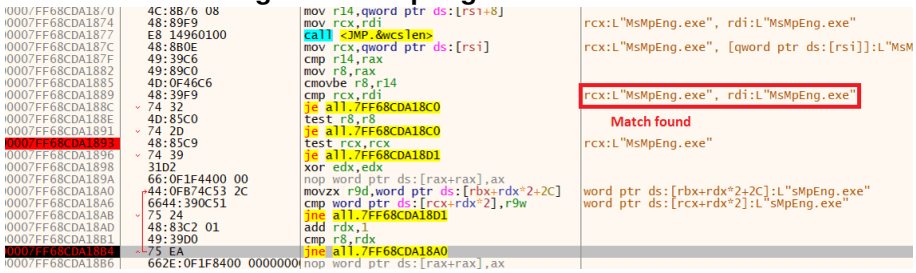


Figure 4. MsMpEng.exe match found



to restart their services to protect the machine. However, the AV killer keeps running in a loop, continuously identifying and terminating the associated security process whenever it reappears, as shown in Figure 5.

Based on Kaspersky's telemetry and information collected from public threat intelligence feeds, adversaries have been using this artifact since at least October 2024. The majority of affected victims are in Russia, Belarus, Kazakhstan, Ukraine, and Brazil.

3.3. Tactics, Techniques and Procedures (TTPs)

The adversary demonstrated in-depth knowledge of Windows internals and binary exploitation techniques. Table 1 lists observed TTPs aligned with the MITRE ATT&CK framework.

Figure 5. MsMpEng.exe was killed

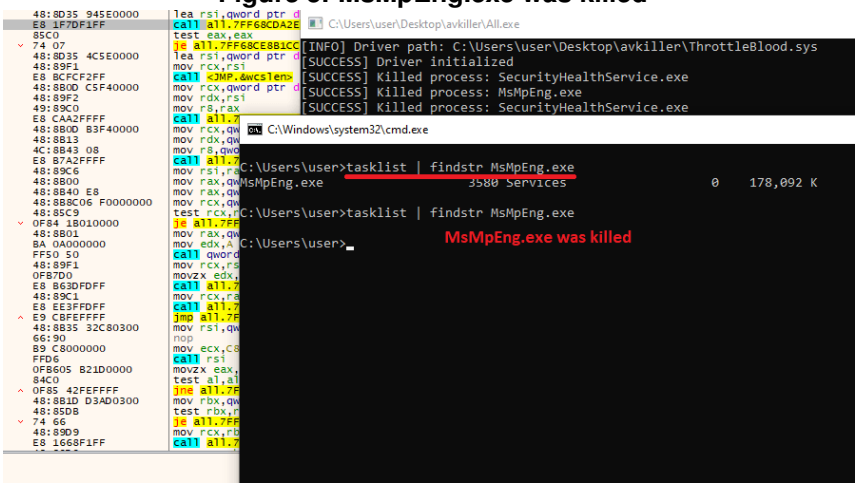


Table 1. Tactics, techniques, and procedures identified in the AV killer

Tactic	Technique	ID
Discovery	Process Discovery	T1057
Defense Evasion	Impair Defenses: Disable or Modify Tools	T1562.001
Defense Evasion	Impair Defenses: Indicator Blocking	T1562.006
Privilege Escalation	Create or Modify System Process: Windows Service	T1543.003
Impact	Service Stop	T1489

4. Recommendations

This incident highlights the need for strong hardening practices and defense in depth. Restricting public RDP exposure and enforcing robust password policies could have reduced the likelihood of the initial compromise. Multi-factor authentication should also be enforced for all remote-access services, especially those exposed to the Internet or used by administrative accounts. In addition, organizations should adopt least-privilege access policies and application allowlisting to reduce the attack surface available after an initial compromise [Al-Karaki 2025].

Network segmentation is another important measure, as it limits lateral movement and reduces the impact of compromised credentials. Regular patch management and vulnerability scanning should be combined with continuous monitoring to identify exposed services, outdated components, and weak configurations. Centralized logging and alerting are also essential to detect suspicious authentication events, abnormal administrative activity, and the deployment of unexpected binaries across multiple hosts [Mohammed 2023].

The growing use of AV killer software also reinforces the importance of self-defense mechanisms in security products. These mechanisms should protect application files, prevent unauthorized process termination, monitor critical memory regions, and ensure that detection rules remain updated across customer endpoints. Security teams should also monitor for known vulnerable drivers, suspicious driver loading events, and unusual interactions between user-mode processes and kernel-mode components, which machine learning-based mechanisms can help detect [Souza and Batista 2025].

5. Conclusion

This article presented an analysis of an AV killer that abuses the legitimate `ThrottleStop.sys` driver as part of a BYOVD technique. By loading a signed vulnerable driver and abusing its privileged functionality, the malware was able to execute kernel-level operations from user mode and terminate security-related processes. This behavior illustrates how legitimate but vulnerable drivers can become powerful tools for defense evasion when incorporated into malicious attack chains.

The analyzed incident also shows that BYOVD-based AV killers are not isolated technical cases, but practical components used in real ransomware operations. After obtaining valid credentials, the attacker moved laterally, deployed malicious artifacts, disabled security defenses, and executed ransomware across the victim environment. This reinforces the importance of combining preventive controls, endpoint protection, network visibility, and incident response readiness.

References

Al-Karaki, J. N. (2025). Defense in depth: a multilayered approach. *Defense in Depth: Modern Cybersecurity Strategies and Evolving Threats*, pages 51–72.

Gupta, R., Dresel, L. P., Spahn, N., Vigna, G., Kruegel, C., and Kim, T. (2022). Popkorn: Popping windows kernel drivers at scale. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 854–868.

Mohammed, A. (2023). Soc audits in action: Best practices for strengthening threat detection and ensuring compliance. *Baltic Journal of Engineering and Technology*, 2(1):62–69.

Monzani, A., Parata, A., Oliveri, A., Aonzo, S., Balzarotti, D., Lanzi, A., et al. (2026). Unveiling byovd threats: Malware's use and abuse of kernel drivers. In *Network and Distributed System Security (NDSS) Symposium 2026*, pages 1–19. NDSS.

Poslušnỳ, M. (2022). Signed kernel drivers–unguarded gateway to windows' core.

Soliven, R. and Kimura, H. (2022). Ransomware actor abuses genshin impact anti-cheat driver to kill antivirus. *Trend Micro. URl: https://www. trendmicro. com/en_us/research/22/h/ransomwareactor-abuses-genshin-impact-anti-cheat-driver-to-killantivirus. html (visited on Oct. 1, 2022)*.

Souza, C., Muñoz, A., Ovalle, E., Figurelli, F., and Leite, A. (2025a). Driver of destruction: How a legitimate driver is being used to take down av processes. Technical report, Kaspersky Securelist.

Souza, C. H., Pascoal, T., Neto, E. P., Sousa, G. B., SL Filho, F., Batista, D. M., and Silva, F. S. D. (2025b). Sdn-based solutions for malware analysis and detection: State-of-the-art, open issues and research challenges. *Journal of Information Security and Applications*, 93:104145.

Souza, C. H. M. and Batista, D. M. (2025). On the use of machine learning for modern iot elf malware detection. In *2025 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6.