DEFENCE TECH

Terra, Cielo, Mare, Spazio, Spazio cibernetico.
PROTEGGIAMOLI

Common EDR attack techniques

Malware Lab Analysis Report

Summary

DEFENCE TECH

1

Our Malware Lab

1. Our Malware Lab

Defence Tech Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malwares.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

CORRADO AARON VISAGGIO
Group Chief Scientist Officer & Malware Lab Director
a.visaggio@defencetech.it

DEFENCE TECH

2

Executive Summary

2. Executive Summary

Anti-malware or more recently Endpoint Detection and Response (EDR) software have become fundamental tools in the modern cyber-security landscape: they help businesses detect, and sometimes prevent, threats such as malware, ransomware, phishing, and data breaches. Such products are important for companies because they provide visibility into the network, enables faster incident response, and reduces the impact of cyberattacks.

Consequently, any tool capable of circumventing the policies imposed by EDRs is inherently malicious and if used in a successful attack, it can compromise the security of the infected system and clear the way to further stages of the attack. In that regard, there is a particular case which caught our attention.

A user going by the nickname of "spyboy", on a Russian-speaking forum named Ramp, has recently advertised a piece of software, supposedly able to circumvent the most common EDR software on the market.

At the time of the announcement, the software was presented as an "EDR killer" and was named "Terminator". In a demo-video[1] was shown how it could terminate the process of a well-known endpoint protection software, allowing the execution of a malicious payload that would have been otherwise blocked.

After much speculation from the cyber-security community, researchers discovered that the attack shown in the demo was based on an uncommon vulnerable driver part of a legitimate product named "Zemana Anti-Malware".

This kind of attack is called "Bring Your Own Vulnerable Driver" or BYOVD for short and consists in bundling a legitimate vulnerable driver along with the malicious payload to exploit the elevated kernel-level privileges of the driver and then bypass the EDR products. So, using drivers from legitimate software is necessary because they are cryptographically signed for distribution and Windows will refuse to load unsigned drivers.

[1] *https://streamable.com/h9n16x*

There are many vulnerabilities in signed drivers and not all of them have a CVE number assigned, the most extensive index is maintained by the loldrivers project[2].

BYOVD attacks usually requires administrator privileges to load the vulnerable driver which may seem unlikely in a corporate environment; however, this kind of attack is on the rise with several recent noteworthy campaigns such as in March[3] and April[4] 2023 and more[5]. Properly configuring endpoints is already a huge step towards mitigating this threat.

In this report we summarise the protection techniques used by modern EDR software to prevent threats and how BYOVD attacks can bypass them.

[2] *https://www.loldrivers.io/*

[3] *https://www.theverge.com/2022/10/16/23405739/microsoft-out-of-date-driver-list-windows-pcs-malware-attacks-years-byovd*

[4] *https://www.bleepingcomputer.com/news/security/ransomware-gangs-abuse-process-explorer-driver-to-kill-security-software/*

[5] *https://www.crowdstrike.com/blog/scattered-spider-attempts-to-avoid-detection-with-bring-your-own-vulnerable-driver-tactic/*

DEFENCE TECH

3 Analysis

3. Analysis

3.1 Typical Anti-Malware architecture

An anti-malware software solution is typically composed of three main components:

- A kernel driver that is used to intercept system-wide events such as process creation, filesystem activity and network traffic;
- A user-mode service which receives and processes the events captured by the driver applying the detection and analysis policies;
- A user-mode GUI process that lives in the session of the currently logged-on user, implemented to show notifications and provide a way for the user to configure the rest of the software. This component is typically not available in corporate-grade EDRs where the end-user is not intended to receive notifications.

This architecture is summarized in Figure 1. To prevent attacks on the system all components of the anti-malware must be protected from third-party applications that, in some cases, may even be running as administrator.

The kernel component is implicitly protected by the operating system due to the separation between user mode and kernel mode, but the user-mode components need particular attention to prevent attacks based on termination or process injection.

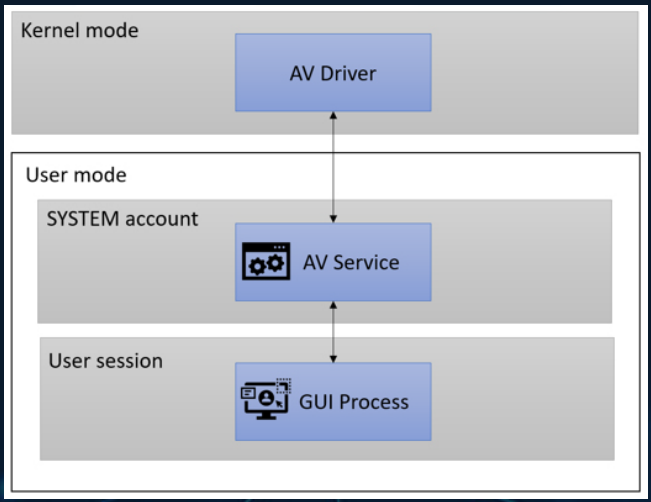

Figure 1. Typical architecture of an antimalware

As explained in the following sections, some facilities are provided by Windows in the form of process protection, while other must be implemented by the anti-malware itself.

3.2 Windows processes and process protection

Like any modern operating system, Windows processes are isolated among themselves using unique address spaces. However, the operating system also offers a set of APIs for cross-process interaction: the simplest one is the process termination, but debugging APIs that allow reading or writing the private address spaces of other processes are available too.

For instance, if a process needs to interact with another one, the first will need to open a handle to the second one using the system API OpenProcess[6]. When using OpenProcess, the caller also specifies the "desired access", that is a set of flag to ask the operating system for specific access rights. Examples of these flags are:

- PROCESS_TERMINATE: allows to terminate the target process
- PROCESS_QUERY_INFORMATION: allows to obtain information about the target process, such as the exit code after termination
- PROCESS_VM_READ: allows to read the private memory of the process

Clearly, not all these flags have the same security-wise impact, that's why Windows enforces isolation between users, preventing calls to OpenProcess when the source and target processes belong to different users. This is done via Access Control Lists (ACLs) which are pervasive in the Windows security model.

[6] *https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-openprocess*

However, isolation does not affect accounts that are part of the Administrators' group, or more broadly, users and groups who have been granted the SeDebugPrivilege by an administrator.

This means that an administrator user can potentially terminate or tamper with any process on the system, and this is clearly undesirable for security applications.

To support such use cases, Microsoft introduced in Windows Vista the concept of Protected Processes (PP) mainly meant for DRM enforcement and later, in Windows 8.1, the more flexible Protected Processes Light (PPL): these are special processes that are hardened from several kinds of attacks, such as DLL Hijacking and code injection. This is done by enforcing signature checks on all the code loaded in the process and rejecting unsafe process access rights (such as PROCESS_VM_READ) when OpenProcess is called, even for administrators.

Without diving too much in the implementation details, executables signed with specific Microsoft-approved certificates can be launched as protected processes; this includes several critical Windows components and, starting with Windows 8.1, antimalware services as well.

Although historically they had several vulnerabilities[8] [9], PPL mitigate most process injection vectors preventing rogue administrators from terminating or otherwise tampering with anti-malware processes.

[7] *https://learn.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/debug-programs*

[8] *https://github.com/itm4n/PPLdump*

[9] *https://github.com/itm4n/PPLmedic*

3.3 Attacking EDR processes

Given our overview on how antimalware software operates, one can conclude that the most general way to attack them is to focus on the analysis service.

There is a variety of known attacks but the most common consists in simply terminating the service process. Terminating a process on Windows requires opening a handle to it with the "terminate" access right and then call the TerminateProcess[10] function. As we explained earlier, this won't work for protected processes since the system prevents opening such handles for user mode applications as part of the threat model of PPL.

However, there is one catch: kernel-mode callers are considered trusted parts of the system and can bypass access checks for securable objects. This means that if one were to use a kernel driver, they could easily terminate any process.

The reason kernel drivers are considered trusted is because Windows 10 Microsoft's driver signing policy[11] only allows loading drivers that were signed with an Extended validation certificate and subsequently submitted to the Windows Hardware Developer Center for approval[12]. This means an attacker cannot simply create a malicious driver because Windows will prevent it from being loaded.

This works well in theory but given the considerable amount of signed kernel drivers being distributed in third party products, some are bound to contain vulnerabilities. Typically, when talking about driver vulnerabilities in this context we mean logic bugs rather than memory corruption vulnerabilities: we are looking for primitives that allow user mode applications to open handles to protected processes through a driver that does not implement appropriate safety checks.

A collection of such vulnerable drivers is maintained by the loldrivers.io project, their database currently counts more than 300 of them.

There are several publicly available proofs of concept showing how to perform a BYOVD attack, one such example is Blackout by ZeroMemoryEx[13].

[10] https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-terminateprocess

[11] https://learn.microsoft.com/en-us/windows-hardware/drivers/install/driver-signing

[12] https://learn.microsoft.com/en-us/windows-hardware/drivers/install/kernel-mode-code-signing-policy--windows-vista-and-later-

[13] https://github.com/ZeroMemoryEx/Blackout

3.4 The Zemana case

OSINT sources[14] revealed the vulnerable driver to be "zam64.sys" with SHA-256 hash: 543991ca8d1c65113dff039b85ae3f9a87f503daec30f46929fd454bc57e5a91

The driver has been uploaded to loldrivers[15] which is how we obtained it for the analysis. The function at address 0x11918 contains the logic relevant to process termination, so we followed the cross-references of this function, which leads to the device IO request handler. In the relevant snippet shown in Figure 2, we can see that it can be called from user mode applications using the IOCTL code 0x80002048.

```
case 0x80002048:
  debug_log(1, "Main.c", 645, "DeviceIoControlHandler", 0, "IOCTL_TERMINATE_PROCESS");
  v3 = ioctl_terminate_process(systemBuffer);
  v14 = v3;
  goto LABEL_80;
```

Figure 2. The IOCTL handler for terminating processes

Information on this driver is scarce, however searching for the relevant IOCTL code produced interesting results: it seems the GitHub user "hfiref0x" posted a list of IOCTL codes[16] from this driver and potential ways to abuse them, in 2020. That means this driver has been known vulnerable for at least three years without being noticed.

Either way, even now that the vulnerability is known, no CVE seems to be assigned for this driver.

[14] *https://twitter.com/SBousseaden/status/1663930984130134017*

[15] *https://www.loldrivers.io/drivers/49920621-75d5-40fc-98b0-44f8fa486dcc/*

[16] *https://gist.github.com/hfiref0x/e116dcf7e99b8d5d36c333a1f1048916*

3.5 BYOVD Mitigations

BYOVD attacks usually require administrative privileges in order to load the vulnerable driver and then open a handle to it; the first line of defence against this type of attack is properly configuring endpoints to prevent regular users from using accounts in the Windows' Administrator group.

Adversaries may exploit vulnerabilities of the system or third-party software for privilege escalation purpose. To prevent the beginning of a full-scale attack, we refer to the mitigation strategies documented by the well-known MITRE| ATT&CK[17].

It's important to note that these drivers typically come from software solutions composed by multiple components, but since the kernel-mode component can be loaded standalone, BYOVD attacks bundle the driver executable as part of the payload. It doesn't matter if the target system does not have the vulnerable software installed.

Furthermore, many known vulnerable drivers have not been issued a CVE, which means they may fly under the radar of traditional vulnerability scanning software.

It's possible to directly block the execution of known vulnerable drivers through application control solutions such as Windows Defender Application Control[18] (WDAC)[19]. However, note that the Microsoft-provided driver block list is not regularly updated and will not block all the known drivers, this is perhaps due to compatibility concerns, so we recommend integrating it with other policies or third-party tools to detect and block exploitation attempts.

[17] *https://attack.mitre.org/techniques/T1068/*

[18] *https://learn.microsoft.com/en-us/windows/security/threat-protection/windows-defender-application-control/microsoft-recommended-driver-block-rules#blocking-vulnerable-drivers-using-wdac*

[19] *https://learn.microsoft.com/it-it/windows/security/threat-protection/windows-defender-application-control/wdac-and-applocker-overview*

DEFENCE TECH

4

Conclusions

4. Conclusions

As we discussed in this report BYOVD attacks are dangerous since they can effectively blind EDRs rendering the system defenceless, however they require specific circumstances to be successful which makes them not as widespread as other attack vectors:

1. First, the attacker needs to use a lesser-known vulnerable driver or a 0-Day otherwise it will be blocked by security solutions.

2. Then, the adversary must obtain initial access to the target system using traditional attack vectors.

3. At last, the attacker can escalate to Local Administrator privileges in order to launch the attack.

These conditions make BYOVD harder to execute successfully compared to more common stealer or ransomware-based attacks.

As for mitigations we recommend hardening endpoints against privilege escalation and to configure Application Guard policies to prevent loading vulnerable drivers. And, where applicable, monitor event logs related to unexpected third-party driver loads using SIEM or EDRs capable of forwarding Windows events. It is also important to develop a robust cyber threat intelligence capability to track active malicious campaigns and to determine what type of threat could be used against an organization, such as software exploits or 0-Days.

More general mitigations include applying restrictions to third-party software execution, for example by allowing only signed software, using Windows AppLocker or similar technologies. Although BYOVD attacks use signed drivers, the initial entry point must be a regular application and blocking it is just as effective. Keep in mind that malware campaigns have been observed to abuse stolen certificates to sign malicious payloads, so this is just a mitigation and not a solution.

Finally, one of the most important things is to keep software updated as much as possible, to reduce the attack surface exposed by software bugs or outdated certificate revocation lists.

DEFENCE TECH

Terra, Cielo, Mare, Spazio, Spazio cibernetico.
PROTEGGIAMOLI

DONE IT IT SECURITY

NEXT INGEGNERIA DEI SISTEMI

FORAMIL RADAR TECHNOLOGIES & DEFENCE SYSTEMS

INNOVESI electronic systems

Defence Tech Holding S.p.A Società Benefit

Via Giacomo Peroni, 452 - 00131 Roma
tel. 06.45752720 - fax 06.45752721
info@defencetech.it - www.defencetech.it