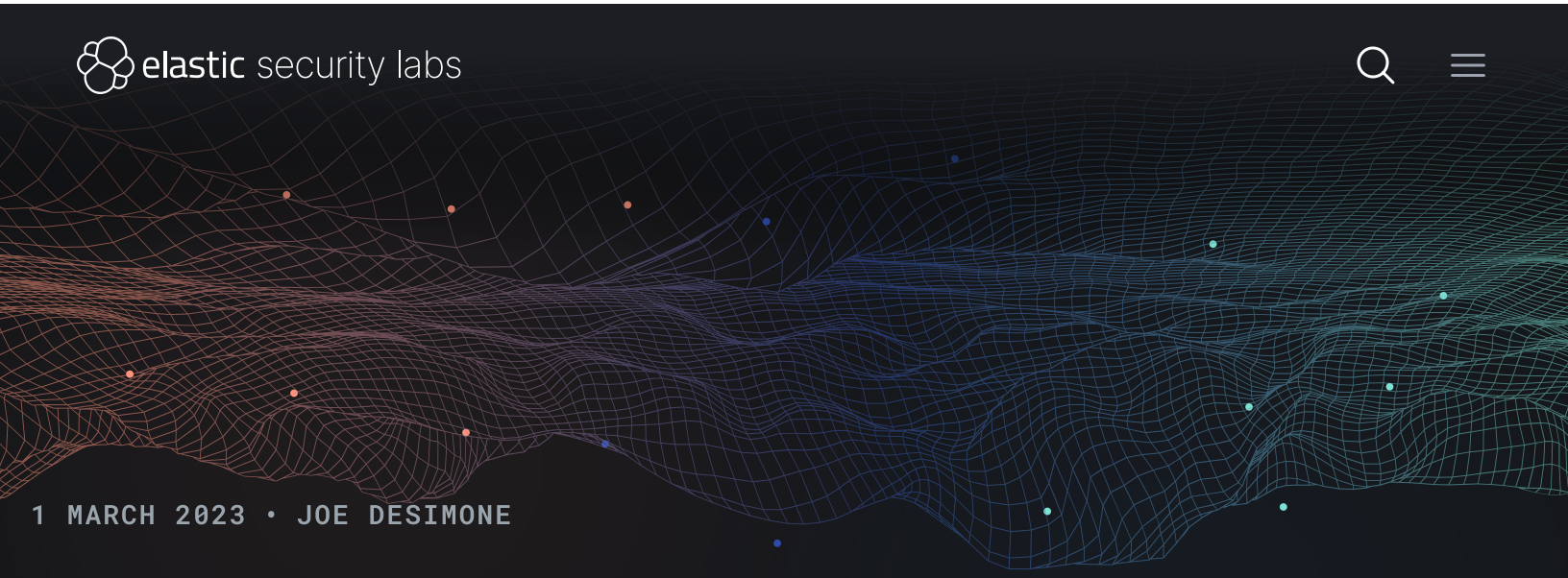
elastic security labs



1 MARCH 2023 • JOE DESIMONE

Stopping Vulnerable Driver Attacks

Using vulnerable drivers to gain kernel mode execution.

7 min read Security operations, Detection science

elastic security labs

Key takeaways

- Ransomware actors are leveraging vulnerable drivers to tamper with endpoint security products.
- Elastic Security **released** 65 YARA rules to detect vulnerable driver abuse.
- Elastic Endpoint (8.3+) protects users from this threat.

Background

In 2018, **Gabriel Landau** and **Joe Desimone** presented a **talk** at Black Hat covering the evolution of kernel mode threats on Windows. The most concerning trend was towards leveraging known good but vulnerable

protection (**HVCI**) and Windows Hardware Quality Labs (**WHQL**) signing requirement enabled. At the time, the risk to everyday users was relatively low, as these techniques were mostly leveraged by advanced state actors and top red teams.

Fast forward to 2022, and attacks leveraging vulnerable drivers are a growing concern due to a **proliferation** of open source **tools** to perform these **attacks**. Vulnerable drivers have now been **used by ransomware** to terminate security software before encrypting the system. Organizations can reduce their risk by limiting administrative user permissions. However, it is also imperative for security vendors to protect the user-to-kernel boundary because once an attacker can execute code in the kernel, security tools can no longer effectively protect the host. Kernel access gives attackers free rein to tamper or terminate endpoint security products or inject code into protected processes.

This post includes a primer on kernel mode attacks, along with Elastic's recommendations for securing users from kernel attacks leveraging vulnerable drivers.

Attack flow

There are a number of flaws in drivers that can allow attackers to gain kernel mode access to fully compromise the system and remain undetected. Some of the **most common** flaws include granting user mode processes write access to virtual memory, physical memory, or **model-specific registers** (MSR). Classic buffer overflows and missing bounds checks are also common.

A less common driver flaw is unrestricted **handle duplication**. While this may seem like innocuous functionality at first glance, handle duplication can be leveraged to gain full kernel code execution by user mode processes. For example, the latest **Process Explorer** driver by Microsoft exposes **such a function**.

An attacker can leverage this vulnerability to duplicate a **sensitive handle** to raw physical memory present in the System (PID 4) process.

their associated physical addresses. This grants an arbitrary virtual read/write primitive, which attackers can leverage to easily tamper with kernel data structures or execute arbitrary kernel code. On HVCI-enabled systems, thread control flow can be hijacked to execute arbitrary kernel functions as shown below.

We reported this issue to Microsoft in the vulnerable driver **submission portal** on July 26, but as of this writing have not received a response. We hope Microsoft will consider this a serious security issue worth addressing. Ideally, they will release a fixed version without the vulnerable **IOCTLs** and include it in the default HVCI blocklist. This would be consistent with the **blocking** of the ProcessHacker (now known as **System Informer**) driver for the **same flaw.**

Blocklisting

Blocklisting prevents known vulnerable drivers from loading on a system, and is a great first step to the vulnerable driver problem. Blocklisting can raise the cost of kernel attacks to levels out of reach for some criminal groups, while maintaining low false positive rates. The downside is it does not stop more **advanced groups**, which can identify new, previously-unknown, vulnerable drivers.

Microsoft maintains a **catalog** of known exploited or malicious drivers, which should be a minimum baseline. This catalog consists of rules using various combinations of **Authenticode** hash, certificate hash (also known as **TBS**), internal file name, and version. The catalog is intended to be used by Windows Defender Application Control (**WDAC**). We used this catalog as a starting point for a more comprehensive list using the **YARA** community standard.

To expand on the existing list of known vulnerable drivers, we pivoted through VirusTotal data with known vulnerable import hashes and other metadata. We also combed through public attack tooling to identify additional vulnerable drivers. As common practice for Elastic Security, we made our **blocklist** available to the community. In Elastic **Endpoint Security** version 8.3 and newer, all drivers are validated against the blocklist in-line before they are allowed to load onto the system (shown below).

One of the most robust defenses against this driver threat is to only allow the combination of driver signer, internal file name, version, and/or hashes, which are known to be in use. We recommend organizations be as strict as feasible. For example, do not blanket trust all **WHQL** signed drivers. This is the classic application control method, albeit focusing on drivers. An organization's diversity of drivers should be more manageable than the entirety of user mode applications. Windows Defender Application Control (**WDAC**) is a powerful built-in feature that can be configured this way. However, the learning curve and maintenance costs may still be too high for organizations without well-staffed security teams. To reap most of the benefits of the allowlisting approach, but reduce the cost of implementation to the users (ideally to blocklisting levels), we recommend two approaches in tandem: behavior control and alert on first seen.

Behavior control

The concept behind behavior control is to produce a more manageable set of allowlistable behavior choke points that can be tuned for high confidence. For example, we can create a behavior control around which applications are allowed to write drivers to disk. This may start with a relatively loose and simple rule:

From there, we can allowlist the benign applications that are known to exhibit this behavior. Then we receive and triage hits, tune the rule until it becomes high confidence, and then ship as part of our **malicious behavior protection**. Elastic SIEM users can use the same technique to **create custom** Detection Engine **rules** tuned specifically for their environment.

First seen

Elastic Security in 8.4 adds another powerful tool that can be used to identify suspicious drivers. This is the **"New Terms" rule type**, which can be used to create an alert when a term (driver hash, signer, version,

This empowers security teams to quickly surface unusual drivers the first time they're seen in their environment. This supports a detection opportunity for even previously unknown vulnerable drivers or other driver-based adversary tradecraft.

Conclusion

Vulnerable driver exploitation, once relegated to advanced adversaries, has now proliferated to the point of being used in ransomware attacks. The time for the security community to come together and act on this problem is now. We can start raising the cost by collaborating on blocklists as a community. We should also investigate additional detection strategies such as behavior control and anomaly detection to raise the cost further without requiring significant security expertise or resources to achieve.