Screwed Drivers

Mother of All Drivers

New Vulnerabilities Found in Windows Drivers

WINDOWS KERNEL SECURITY MODEL ENABLES ATTACKS

As part of our previous research, released in August 2019, Eclypsium researchers detailed how simple design flaws in widely distributed drivers can be abused by attackers to gain control over Windows-based systems including the underlying system and component firmware of the device. We originally named 17 vendors affected by these vulnerable drivers. Now, as part of our ongoing analysis, we have discovered additional vulnerable drivers that are some of the most feature-rich we have seen to date, and which directly affect Intel-based devices. In this update, we detail the latest findings on these drivers and share ongoing industry response to our previous disclosures.

A QUICK RECAP OF SCREWED DRIVERS VULNERABILITIES

At the heart of the issue are drivers that allow users to modify the Windows kernel or device firmware. Abuse of such capability can enable an attacker to gain incredible privileges over a machine while also avoiding traditional security controls. More specifically, an attacker or malware in user space of a device (ring 3) can use a vulnerable driver to read and write data to kernel space (ring 0) and even lower-level firmware components which can sometimes be referred to as negative rings. This low-level control provides an ideal position to steal data, damage the system, and persist on the system outside the view of security controls running at the operating system level.

Notably, these drivers are valid tools released by vendors to help manage or update devices, and as such were properly signed and would be trusted on almost any machine. Worse still, there is no universal mechanism to prevent a Microsoft OS from loading one of these bad drivers.

INTEL PMx DRIVER

In our previous research, we identified a variety of drivers, each with their own capabilities and potential impacts to a system. Although many of the vulnerable drivers were disclosed at the time of our previous publication, two drivers from Intel were held under embargo until the fix and security advisory was available. These are now public at Intel Processor Identification Utility for Windows Advisory and Intel Computing Improvement Program Advisory as of August 13th. Another driver that was held under embargo due to the complexity of the issue was the Intel PMx Driver (also named PMxDrv). During our analysis of the Intel PMx driver, we found it to be incredibly capable, containing a superset of all the capabilities that we had seen previously. For example, the driver has the ability to:

- Read/Write to physical memory
- Read/Write to Model Specific Registers (MSR)
- Read/Write to control registers
- Read/Write to the interrupt descriptor table (IDT) and the global descriptor table (GDT)
- Read/write to debug registers
- Arbitrarily gain I/O access
- Arbitrarily gain PCI access

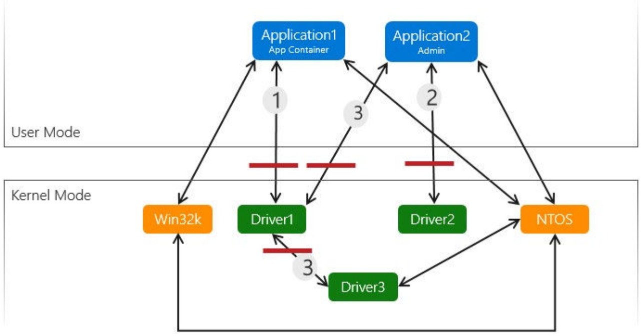
This level of access can provide an attacker with near-omnipotent control over a victim device. Just as importantly, this capability has been included as a staple component of many Intel ME and BIOS related toolsets going back to 1999. Ironically, the very tool released by Intel to detect and mitigate a recent AMT vulnerability included the vulnerable driver as part of the toolset used to solve the AMT issue. Intel likewise uses the vulnerable driver as part of the Flash Programming Tool, which is provided to OEM vendors and their customers to update Intel-based BIOS. This makes the Intel PMx/PMxDrv one of the most capable, feature-rich, and most common drivers we have seen to date.

Eclypsium researchers have been working closely with the Intel PSIRT team on the issue, and would like to thank them for their prompt and positive response. As of November 12th, Intel has released updated versions of pmxdrvx64.sys and pmxdrv.sys to mitigate this vulnerability.

ADMINISTRATOR TO KERNEL OR HARDWARE

The vast majority of drivers we examined in our research could be exploited by an unprivileged user to attack the running kernel or modify device firmware via unfiltered IO, PCI, or MMIO access, however a few drivers had additional restrictions and only allowed use by a process running with Administrator privileges.

Microsoft's Windows security model for driver developers discusses various security boundaries in how drivers operate within the Windows operating system and characterizes the path between an admin process and a kernel driver as a "noteworthy trust boundary":



"Path (2) is a lower risk path, as the app is running with admin rights and is calling directly into the kernel driver. Admin is already a fairly high privilege on the system so the attack surface from admin to kernel is less of an interesting target to attackers, but still a noteworthy trust boundary."

https://docs.microsoft.com/en-us/windows-hardware/drivers/driversecurity/windows-security-model

However, Microsoft's Security Servicing Criteria for Windows currently treats processes running in userspace with Administrator privileges effectively the same as the Windows kernel and there's no security boundary there. Superficially, that makes sense because the Administrator is intended to have administrative control over the system which necessarily includes security configuration of the device. However, under deeper analysis, some significant flaws in that justification become apparent.

Although the Administrator has control of the device, there are many security-sensitive operations that are additionally restricted even from the Administrator. Once Secure Boot is enabled, a reboot and a process intended to verify physical presence should be required to disable it. Likewise, the Administrator cannot load unsigned kernel modules without rebooting and performing physically-present operations during the boot process. There are many security controls which cannot be disabled at runtime without a reboot.

Allowing a compromised Administrator process to read and write kernel memory and otherwise launch attacks against the kernel renders these controls ineffective and leaves a gaping security hole. Alex Ionescu characterized the situation as *"Windows 10: Kernel arbitrary writes from Admin are not bugs, there's a party in ring0 and the bouncer is off duty"*.

In contrast, Apple's System Integrity Protection is specifically intended to protect critical parts of the Mac operating system against malicious software even running as root with full administrator privileges. System Integrity Protection can be disabled by the Administrator if necessary, but it cannot be done at runtime and they must turn the system off and boot into the Recovery OS to disable this protection.

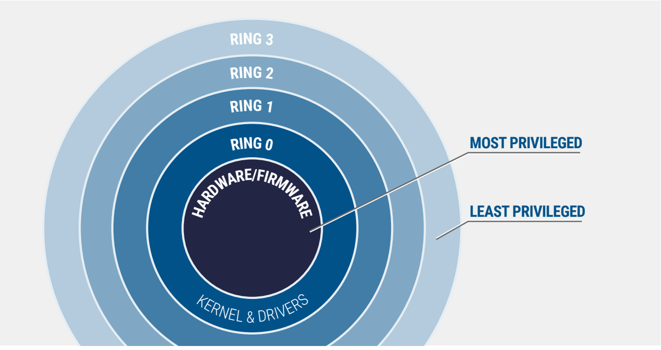
Likewise, within the Linux ecosystem, the Kernel Lockdown feature also prevents the root user from performing operations which can compromise the integrity of the kernel. This is an important security control and the majority of Linux distributions have been shipping versions of this protection mechanism for years and the patch has now been accepted into the mainline Linux kernel.

In addition, the Linux Self Protection Project has been working to strengthen the protections for this important security boundary. Kees Cook, one of the founders of LSPP, described their goals as *"It's about creating a bright line between uid-0 and ring-0. The most powerful of these distinctions was made long ago with signed modules. It hasn't been enough, though, since there have been many ways for uid-0 to read or write kernel memory. My expectation for this was to reasonably fill all the remaining gaps."*

It's certainly a very hard task to defend this security boundary, but attackers are exploiting this in real-world malware campaigns such as Lojax and Slingshot. Ignoring the problem doesn't make it go away.

UPDATES ON WINRING0

As part of our previous research, we posted a list of vulnerable drivers, available here. One of the most significant was a driver often referred to as "WinRing0". Recently, researchers at SafeBreach provided a thorough proof-of-concept analysis of an attack against HP Touchpoint Analytics, which utilizes the WinRing0 driver which was included as part of the OEM-installed software. SafeBreach demonstrated how the driver could be used to read arbitrary kernel memory and discussed how it could be misused in a variety of additional ways including bypassing application whitelisting, signature validation, and driver signature enforcement.



WinRing0 is also notable in that it is part of the OpenHardwareMonitor library and is readily accessible to anyone. As such, it can easily be used and signed by multiple vendors, and can potentially be found under a variety of names and hashes. For example, a check of VirusTotal reveals that the original WinRing0 driver analyzed as part of our research has been identified under a variety of names including:

- 11bd2c9f9e2397c9_winring0x64.sys
- ActiveHealth.sys
- CAM_V3.sys
- GameFire.sys
- MODAPI.sys
- OpenHardwareMonitor.sys
- OpenHardwareMonitorLib.sys
- OpenHardwareMonitorLib_IObitDel_qzcdbx.sys
- OpenHardwareMonitorReport.sys
- SmartDashboard.sys
- SystemGauge.sys
- SystemGaugeX7.sys
- VideoNovaServerControllerService.sys
- WinRing0.sys
- WinRing0x64.sys
- ellp_service.sys
- hardwareproviders.sys
- modapi.sys
- monitor.sys
- ohm.sys
- openhardwaremonitorlib.sys
- sensorsview32_64.sys
- systemgauge.sys
- touchpointanalyticsclient.sys
- winring0x64.sys

Currently, our research into vulnerable drivers is ongoing and we are actively working with additional vendors as part of our responsible disclosure process. Users and organizations should consider enabling Hypervisor-protected Code Integrity (HVCI) for devices that support the feature. A list of requirements and instructions for enabling HVCI is available here. We will continue to analyze this important area and provide updates in coordination with affected vendors.

SHORT-TERM AND LONG-TERM FIXES

One of the key issues noted above is that there is no universally applicable way to prevent Windows from loading any of the bad drivers that have been identified thus far. Going forward, Microsoft is addressing the issue through their HVCI technology. This will allow Microsoft to act as their own virtual firewall to protect the operating system kernel.

However this approach will not be available universally for some time. HVCI requires a 7th generation or newer processor, new processor features such as mode-based execution control, and is not supported by many 3rd party drivers. As a result, many devices in use today will not be able to enable HVCI and will not be protected.

The only universally available option possible today is to block or blacklist old, known-bad drivers. To this end, we would like to specifically commend the response of Insyde Software, a UEFI firmware vendor. Of the 19 vendors we notified early this summer, Insyde is the only vendor to date to proactively contact Microsoft and ask that the old version of the driver be blocked. Due to this request, Windows Defender will proactively quarantine the vulnerable version of the driver so it can't cause damage to the system.