Project Zero

News and updates from the Project Zero team at Google

(Move to ...)

Thursday, April 18, 2024

The Windows Registry Adventure #1: Introduction and research results

Posted by Mateusz Jurczyk, Google Project Zero

In the 20-month period between May 2022 and December 2023, I thoroughly audited the Windows Registry in search of local privilege escalation bugs. It all started unexpectedly: I was in the process of developing a coverage-based Windows kernel fuzzer based on the Bochs x86 emulator (one of my favorite tools for security research: see Bochspwn, Bochspwn Reloaded, and my earlier font fuzzing infrastructure), and needed some binary formats to test it on. My first pick were PE files: they are very popular in the Windows environment, which makes it easy to create an initial corpus of input samples, and a basic fuzzing harness is equally easy to develop with just a single GetFileVersionInfoSizeW API call. The test was successful: even though I had previously fuzzed PE files in 2019, the new element of code coverage guidance allowed me to discover a completely new bug: issue #2281.

For my next target, I chose the Windows registry. That's because arbitrary registry hives can be loaded from disk without any special privileges via the RegLoadAppKey API (since Windows Vista). The hives use a binary format and are fully parsed in the kernel, making them a noteworthy local attack surface. Furthermore, I was also somewhat familiar with basic harnessing of the registry, having fuzzed it in 2016 together with James Forshaw. Once again, the code coverage support proved useful, leading to the discovery of issue #2299. But when I started to perform a root cause analysis of the bug, I realized that:

- The hive binary format is not very well suited for trivial bitflipping-style fuzzing, because it is structurally simple, and random mutations are much more likely to render (parts of) the hive unusable than to trigger any interesting memory safety violations.
- On the other hand, the registry has many properties that make it an attractive attack surface for further research, especially for manual review. It is 30+ years old, written in C, running in kernel space but highly accessible from user-mode, and it implements much more complex logic than I had previously imagined.

And that's how the story starts. Instead of further refining the fuzzer, I made a detour to reverse engineer the

registry implementation in the Windows kernel (internally known as the Configuration Manager) and learn more about its inner workings. The more I learned, the more hooked I became, and before long, I was all-in on a journey to audit as much of the registry code as possible. This series of blog posts is meant to document what I've learned about the registry, including its basic functionality, advanced features, security properties, typical bug classes, case studies of specific vulnerabilities, and exploitation techniques.

While this blog is one of the first places to announce this effort, I did already give a talk titled "Exploring the Windows Registry as a powerful LPE attack surface" at Microsoft BlueHat Redmond in October 2023 (see slides and video recording). The upcoming blog posts will go into much deeper detail than the presentation, but if you're particularly curious and can't wait to find out more, feel free to check these resources as a starter. 🙂

Research results

In the course of the research, I filed **39 bug reports** in the Project Zero bug tracker, which have been fixed by Microsoft as **44 CVEs**. There are a few reasons for the discrepancy between these numbers:

- Some single reports included information about multiple problems, e.g. issue #2375 was addressed by four CVEs,
- Some groups of reports were fixed with a single patch, e.g. issues #2392 and #2408 as CVE-2023-23420,
- One bug report was closed as WontFix and not addressed in a security bulletin at all (issue #2508).

All of the reports were submitted under the Project Zero 90-day disclosure deadline policy, and Microsoft successfully met the deadline in all cases. The average time from report to fix was 81 days.

Furthermore, between November 2023 and January 2024, I reported 20 issues that had low or unclear security impact, but I believed the vendor should nevertheless be made aware of them. They were sent without a disclosure deadline and weren't put on the PZ tracker; I have since published them on our team's GitHub. Upon assessment, Microsoft decided to fix **6** of them in a security bulletin in March 2024, while the other 14 were closed as WontFix with the option of being addressed in a future version of Windows.

This sums up to a total of **50 CVEs**, classified by Microsoft as:

- 39 × Windows Kernel Elevation of Privilege Vulnerability
- 9 × Windows Kernel Information Disclosure Vulnerability
- 1 × Windows Kernel Memory Information Disclosure Vulnerability
- 1 × Windows Kernel Denial of Service Vulnerability

A full summary of the security-serviced bugs is shown below:

GPZ #	CVE	Title	Reported	Fixed

2295	CVE-2022-34707	Windows Kernel use-after-free due to refcount overflow in registry hive security descriptors	2022-May-11	2022-Aug-09
2297	CVE-2022-34708	Windows Kernel invalid read/write due to unchecked Blink cell index in root security descriptor	2022-May-17	
2299	CVE-2022-35768	Windows Kernel multiple memory problems when handling incorrectly formatted security descriptors in registry hives	2022-May-20	
2318	CVE-2022-37956	Windows Kernel integer overflows in registry subkey lists leading to memory corruption	2022-Jun-22	2022-Sep-13
2330	CVE-2022-37988	Windows Kernel registry use-after-free due to bad handling of failed reallocations under memory pressure	2022-Jul-8	2022-Oct-11
2332	CVE-2022-38037	Windows Kernel memory corruption due to type confusion of subkey index leaves in registry hives	2022-Jul-11	
2341	CVE-2022-37990	Windows Kernel multiple memory corruption issues when operating on very long registry paths	2022-Aug-3	
	CVE-2022-38039			
	CVE-2022-38038			
2344	CVE-2022-37991	Windows Kernel out-of-bounds reads and other issues when operating on long registry key and value names	2022-Aug-5	
2359	CVE-2022-44683	Windows Kernel use-after-free due to bad handling of predefined keys in NtNotifyChangeMultipleKeys	2022-Sep-22	2022-Dec-13
2366	CVE-2023-21675	Windows Kernel memory corruption due to insufficient handling of predefined keys in registry virtualization	2022-Oct-6	2023-Jan-10
2369	CVE-2023-21747	Windows Kernel use-after-free due to dangling registry link node under paged pool memory pressure	2022-Oct-13	
2389	CVE-2023-21748	Windows Kernel registry virtualization incompatible with transactions, leading to inconsistent hive state and memory corruption	2022-Nov-30	
2375		Windows Kernel multiple issues in the key replication feature of registry virtualization	2022-Oct-25	
	CVE-2023-21772			
	CVE-2023-21773			
	CVE-2023-21774			
2378	CVE-2023-21749	Windows Kernel registry SID table poisoning leading to bad locking and other issues	2022-Oct-31	
	CVE-2023-21776			

2379	CVE-2023-21750	Windows Kernel allows deletion of keys in virtualizable hives with KEY_READ and KEY_SET_VALUE access rights	2022-Nov-2	
2392	CVE-2023-23420	Windows Kernel multiple issues with subkeys of transactionally renamed registry keys	2022-Dec-7	2023-Mar-14
2408		Windows Kernel insufficient validation of new registry key names in transacted NtRenameKey	2023-Jan-13	
2394	CVE-2023-23421 CVE-2023-23422 CVE-2023-23423	Windows Kernel multiple issues in the prepare/commit phase of a transactional registry key rename	2022-Dec-14	
2410	CVE-2023-28248	Windows Kernel CmpCleanupLightWeightPrepare registry security descriptor refcount leak leading to UAF	2023-Jan-19	2023-Apr-11
2418	CVE-2023-28271	Windows Kernel disclosure of kernel pointers and uninitialized memory through registry KTM transaction log files	2023-Jan-31	
2419	CVE-2023-28272 CVE-2023-28293	Windows Kernel out-of-bounds reads when operating on invalid registry paths in CmpDoReDoCreateKey/CmpDoReOpenTransKey	2023-Feb-2	
2433	CVE-2023-32019	Windows Kernel KTM registry transactions may have non-atomic outcomes	2023-Mar-7	2023-Jun-13
2445	CVE-2023-35356	Windows Kernel arbitrary read by accessing predefined keys through differencing hives	2023-Apr-19	2023-Jul-11
2452		Windows Kernel CmDeleteLayeredKey may delete predefined tombstone keys, leading to security descriptor UAF	2023-May-10	
2446	CVE-2023-35357	Windows Kernel may reference unbacked layered keys through registry virtualization	2023-Apr-20	
2447	CVE-2023-35358	Windows Kernel may reference rolled-back transacted keys through differencing hives	2023-Apr-27	
2449	CVE-2023-35382	Windows Kernel renaming layered keys doesn't reference count security descriptors, leading to UAF	2023-May-2	2023-Aug-8
2454	CVE-2023-35386	Windows Kernel out-of-bounds reads due to an integer overflow in registry .LOG file parsing	2023-May-15	
2456	CVE-2023-38154	Windows Kernel partial success of registry hive log recovery may lead to inconsistent state and memory corruption	2023-May-22	

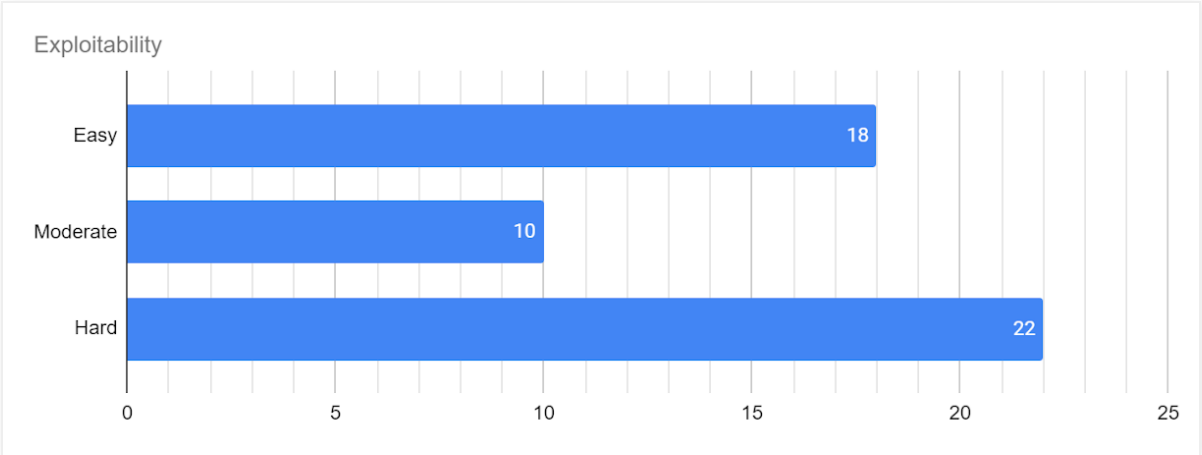
2457	CVE-2023-38139	Windows Kernel doesn't reset security cache during self-healing, leading to refcount overflow and UAF	2023-May-31	2023-Sep-12
2462	CVE-2023-38141	Windows Kernel passes user-mode pointers to registry callbacks, leading to race conditions and memory corruption	2023-Jun-26	
2463	CVE-2023-38140	Windows Kernel paged pool memory disclosure in VrpPostEnumerateKey	2023-Jun-27	
2464	CVE-2023-36803	Windows Kernel out-of-bounds reads and paged pool memory disclosure in VrpUpdateKeyInformation	2023-Jun-27	
2466	CVE-2023-36576	Windows Kernel containerized registry escape through integer overflows in VrpBuildKeyPath and other weaknesses	2023-Jul-7	2023-Oct-10
2479	CVE-2023-36404	Windows Kernel time-of-check/time-of-use issue in verifying layered key security may lead to information disclosure from privileged registry keys	2023-Aug-10	2023-Nov-14
2480	CVE-2023-36403	Windows Kernel bad locking in registry virtualization leads to race conditions	2023-Aug-22	
2492	CVE-2023-35633	Windows registry predefined keys may lead to confused deputy problems and local privilege escalation	2023-Oct-6	2023-Dec-12
2511	CVE-2024-26182	Windows Kernel subkey list use-after-free due to mishandling of partial success in CmpAddSubKeyEx	2023-Dec-13	2024-Mar-12
None (MSRC-84131)	CVE-2024-26174	Windows Kernel out-of-bounds read of key node security in CmpValidateHiveSecurityDescriptors when loading corrupted hives	2023-Nov-29	
None (MSRC-84149)	CVE-2024-26176	Windows Kernel out-of-bounds read when validating symbolic links in CmpCheckValueList	2023-Nov-29	
None (MSRC-84046)	CVE-2024-26173	Windows Kernel allows the creation of stable subkeys under volatile keys via registry transactions	2023-Nov-30	
None (MSRC-84228)	CVE-2024-26177	Windows Kernel unsafe behavior in CmpUndoDeleteKeyForTrans when transactionally re-creating registry keys	2023-Dec-1	
None (MSRC-84237)	CVE-2024-26178	Windows Kernel security descriptor linked list confusion in CmpLightWeightPrepareSetSecDescUoW	2023-Dec-1	
None (MSRC-84263)	CVE-2024-26181	Windows Kernel registry quota exhaustion may lead to permanent corruption of the SAM database	2023-Dec-11	

Exploitability

Software bugs are typically only interesting to either the offensive/defensive sides of the security community if they have practical security implications. Unfortunately, it is impossible to give a blanket statement regarding the exploitability of all registry-related vulnerabilities due to their sheer diversity on a number of levels:

- **Affected platforms:** Windows 10, Windows 11, various Windows Server versions (32/64-bit)
- **Attack targets:** the kernel itself, drivers implementing registry callbacks, privileged user-mode applications/services
- **Entry points:** direct registry operations, hive loading, transaction log recovery
- **End results:** memory corruption, broken security guarantees, broken API contracts, memory/pointer disclosure, out-of-bounds reads, invalid/controlled cell index accesses
- **Root cause of issues:** C-specific, logic errors, bad reference counting, locking problems
- **Nature of memory corruption:** temporal (use-after-free), spatial (buffer overflows)
- **Types of corrupted memory:** kernel pools, hive data
- **Exploitation time:** instant, up to several hours

As we can see, there are multiple factors at play that determine how the bugs came to be and what state they leave the system in after being triggered. However, to get a better understanding of the impact of the findings, I have performed a cursory analysis of the exploitability of each bug, trying to classify it as either "easy", "moderate" or "hard" to exploit according to my current knowledge and experience (this is of course highly subjective). The proportions of these exploitability ratings are shown in the chart below:

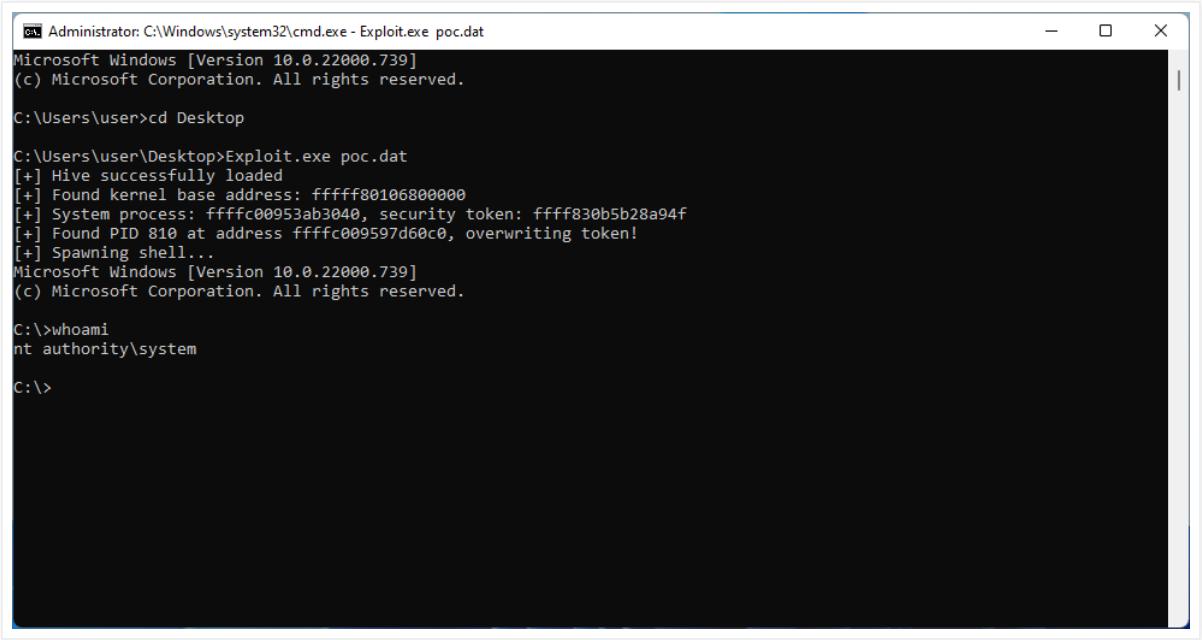


The ratings were largely based on the following considerations:

- Hive-based memory corruption is generally considered easy to exploit, while pool-based memory corruption is considered moderate/hard depending on the specifics of the bug.
- Triggering OOM-type conditions in the hive space is easy, but completely exhausting the kernel pools is more difficult and intrusive.
- Logic bugs are typically easier and more reliable to exploit than memory corruption.
- The kernel itself is typically easier to attack than other user-mode processes (system services etc.).

- Direct information disclosure (leaking kernel pointers / uninitialized memory via various channels) is usually straightforward to exploit.
- However, random out-of-bounds reads, as well as read access to invalid/controlled cell indexes is generally hard to do anything useful with.

Overall, it seems that more than half of the findings can be feasibly exploited for information disclosure or local privilege escalation (rated easy or moderate). What is more, many of them exhibit registry-specific bug classes which can enable particularly unique exploitation primitives. For example, hive-based memory corruption can be effectively transformed into both a KASLR bypass and a fully reliable arbitrary read/write capability, making it possible to use a single bug to compromise the kernel with a data-only attack. To demonstrate this, I have successfully developed exploits for CVE-2022-34707 and CVE-2023-23420. The outcome of running one of them to elevate privileges to SYSTEM on Windows 11 is shown on the screenshot below:



Upcoming posts in this series will introduce you to the Windows registry as a system mechanism and as an attack surface, and will dive deeper into practical exploitation using hive memory corruption, out-of-bounds cell indexes and other amusing techniques. Stay tuned!

Google Project Zero at 9:45 AM

Share

No comments:

Post a Comment

‹ Home ›

View web version