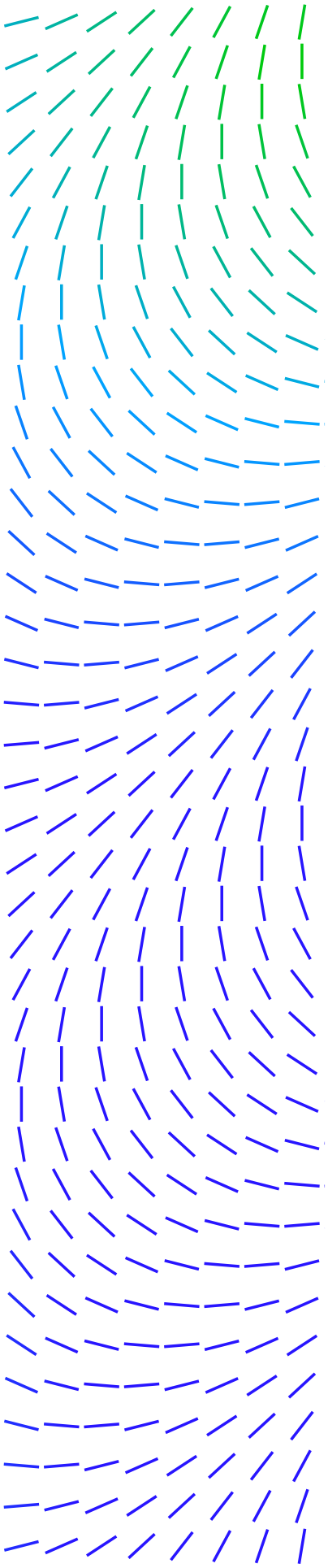

Mapping win32k User to Kernel tagWND Data Structures

By: Eoin Carroll

Trellix

Win32k Vulnerability Research

When performing win32k vulnerability research and exploit analysis it is important to be able to map the user and kernel tagWND data structure counterparts, but we are not aware f any automated way of doing this other than manually setting breakpoints and using data structure contents to deduce the mapping. During the exploit dissection of CVE-2021-1732, there was a generic requirement to track the state of the tagWND data structures and locate the logic that operates on them for root cause analysis. We developed and now share a windbg script to record the creation of tagWND data structures in the Windows kernel which are then mapped back to their user mode counterparts to ease the vulnerability and exploit analysis process. This technique can be used in general to aid in the research of win32k vulnerabilities.

CVE-2021-1732 Exploit Dissection

In February 2021, Dbappsecurity discovered a sample in the wild that exploited a win32k escalation of privilege (EoP) zero-day vulnerability (CVE-2021-1732) on Windows 10 x64. The Advanced Threat Research team performed a deep dive analysis of this vulnerability, to identify the primitives for detection and protection; you can find a full technical analysis in this **blog <insert link>** . The exploit is novel in its use of a new win32k arbitrary kernel memory read primitive using the GetMenuBarInfo API (Application Programming Interface) which, to the best of our knowledge, had not been previously known publicly.

Win32k is the graphical component of the Windows OS (Operating System), containing both user and kernel mode components. When a window is created in usermode using the CreateWindowEx API, a corresponding tagWND data structure is created in the kernel so the OS can manage the window. Per figure 1 below, when HWValidateHandle is supplied with a window handle it returns the address of the tagWND data structure from the usermode desktop heap. This is done for performance reasons as win32k has both usermode and kernel components. Microsoft have hardened this function from leaking kernel addresses, but the offsets within the data structure copied to usermode correspond to the kernel data structure, and it is these relative offsets that allow the exploit to control where it reads and writes to in kernel memory.

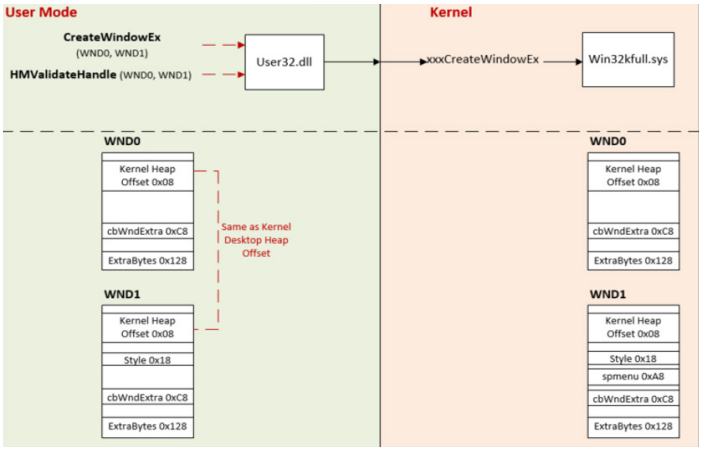

Figure 1. Windows CreateWindowEx API and tagWND data structures relationship

Win32k vulnerabilities are generally turned into a read/write kernel primitive by using a desktop object known as a tagWND data structure created by CreateWindowEx(). Once a vulnerability such as a use-after-free (UAF) has been discovered within the Windows kernel, the tagWND data structure is typically used to create read and write primitives within the kernel for a data-only attack. A data-only attack requires only a read and write primitive as it does not seek to execute malicious code in memory, but manipulate data structures used by the operating system to its advantage (e.g., steal privileged process tokens for escalation of privileges). CVE-2021-1732 is an out-of-bounds write vulnerability which can be turned into an arbitrary read and write "data only attack" by leveraging the tagWND data structure.

Data Structure Mapping Technique

Using the below windbg script we can now automate the mapping of the tagWND data structures from user mode and kernel, the output of which can be seen in figure 2.

```
ba e 1 win32kfull!NtUserCreateWindowEx+0x6a 0 "r @$t2 = poi(@$teb+800+28); r @$t3 =
poi(rax+30); r @$t4 = $t2+$t3; .printf\"Kernel tagWND Address = %p || Kernel tagWND User
Copy Address = %p || tagWND Handle = %x || tagWND Desktop Heap Offset = %x || User tagWND
address = %p \\n\", rax, poi(@rax+28), poi(@rax), $t3 , $t4;g"
```

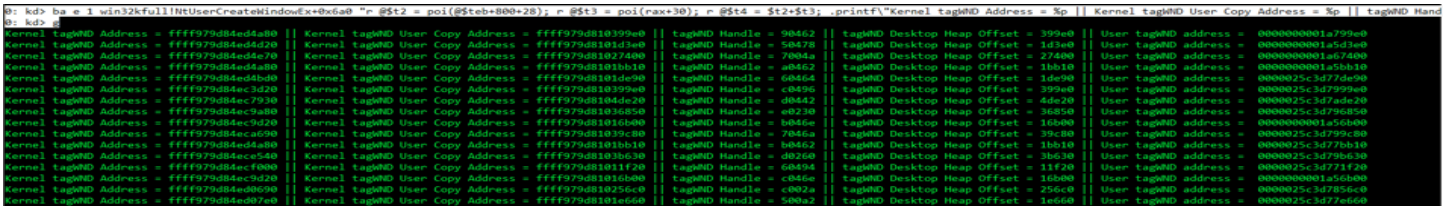

Figure 2. win32k User to Kernel tagWND data structure mapping using our windbg script

The windbg script does the following (see figure 3):

1. Breaks on the return call from win32kfull!xxxCreateWindowEx to retrieve the kernel tagWND address
2. At offset 0x28h within the kernel tagWND address, there is a pointer to the kernel tagWND user copy address. Microsoft have fragmented the kernel desktop heap and removed the kernel addresses from the user copy tagWND data structure that is copied to the user mode desktop heap to prevent kernel address leakage
3. Both kernel tagWND data structures we just located have a reference to the window handle (in this case 6064) and the offset within the desktop heap (1de90 in this case). Note the same desktop offset value is used for locating the windows in both the user mode and kernel desktop heaps which allows one to read and write memory using relative offsets without leaking their absolute address during the exploitation process
4. The Thread Environment Block (TEB) contains a pointer to the Win32ClientInfo data structure at offset 0x800h. The base address of the user mode desktop heap is then located at offset 0x28h within the Win32ClientInfo data structure. By adding the tagWND desktop heap offset to the user mode desktop heap base address we get the absolute address of the user mode tagWND (we are just emulating what happens within the operating system when one calls user32.dll HMValidateHandle)

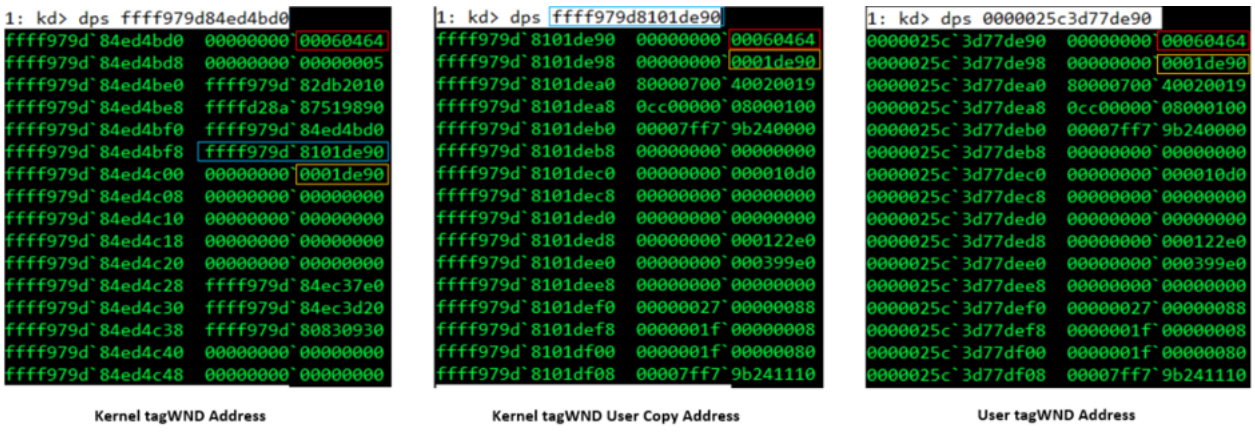

Figure 3. win32k tagWND data structure address and contents in Kernel and User mode

The below script could also be used and would operate independent of any future changes to win32kfull!NtUserCreateWindowEx+0x6a0, but it does generate some warnings in the output, so it is not as clean of an option, but it does work.

```
ba e 1 win32kfull!xxxCreateWindowEx "gu; r @$t2 = poi(@$teb+800+28); r @$t3 = poi(rax+30);
r @$t4 = $t2+$t3; .printf\"Kernel tagWND Address = %p || Kernel tagWND User Copy Address
= %p || tagWND Handle = %x || tagWND Desktop Heap Offset = %x || User tagWND address = %p
\\n\", rax, poi(@rax+28), poi(@rax), $t3 , $t4;g"
```

All testing was performed on Windows 10 version 1909.

Summary

When hunting for vulnerabilities or analyzing critical industry vulnerabilities you must use every known tool and technique at your disposal so you can focus on achieving your research goals with the least path of resistance. While analyzing CVE-2021-1732 I had to manually set breakpoints within the kernel and then visually compare the data structure addresses to user mode which was a tedious process. Win32k is such a targeted component for exploitation that automating this tagWND data structure mapping process adds the value of speeding up future vulnerability hunting and critical industry vulnerability analysis.