# Memory Access errors in the Kernel

Posted by Brett K - 27 Apr 2010 - 03:39

---

Hello, I am new to WinDbg, but not debugging in general. I am trying to step through certain calls like CreateFileA to see how they work, and I get to an area where it looks like the screenshot below:

http://j.imagehost.org/0320/windbg1.png

I am running WinDbg on Windows 7 64 as an Administrator, and it is connecting to a VMWare Windows XP 32-bit system. I am not sure why I would get access errors to this memory?

==========

# Re: Memory Access errors in the Kernel

Posted by Robert Kuster - 06 May 2010 - 13:04

---

Brett, welcome.

Note that you are trying to debug user-mode code (kernel32!CreateFileW is user-mode code...) from a kernel-mode debug session. To do this you must ensure the context of your process is actually used (its virtual addresses space paged-in etc.). A simple .process ?? and .reload /user should fix your memory issues described.

A few more words
The transition from user-mode to kernel-mode is accomplished trough an opcode called sysenter/sysexecute (Intel/AMD). You can see this if you debug a few steps into kernel32!CreateFileW. The call stack will then look like this:

```
0:000> k
ChildEBP RetAddr
0012f7a0 7c90d0ba ntdll!KiFastSystemCall+0x2
0012f7a4 7c8109b6 ntdll!NtCreateFile+0xc
0012f83c 0045c771 kernel32!CreateFileW+0x35f
```

And the disassembly of ntdll!KiFastSystemCall:

```
0:000> uf ntdll!KiFastSystemCall
ntdll!KiFastSystemCall:
7c90e510 8bd4            mov     edx,esp
7c90e512 0f34            sysenter
7c90e514 c3              ret
```

As soon as sysenter is executed we switch execution to kernel-mode. Unfortunately we cannot debug this transition from WinDbg directly so there is a little bit more to know about system services and their implementation on Windows. If we take a look at ntdll.dll and its implementation we see something like this on Windows XP SP3:

http://windbg.info/images/fbfiles/images/ntdll.png

Note how a different value is stored into register EAX for each function? Basically this is an index into the system-service dispatch table (nt!KiServiceTable) which is located in the kernel; this table holds function pointers to all available system services. The system service dispatcher conveniently uses this index from EAX to get a function pointer and calls the function thereafter. With this knowledge we can set the following breakpoint in kd for our CreateFile:

0: kd> bp nt!KiFastCallEntry ".if (@eax == 0x25) { .echo ***** CreateFile called *****;} .else {g;};"

This breakpoint will break whenever an application calls CreateFile in user-mode and will do nothing for other services requested.
Anyway you can use the same "trick" for any imaginable system service by simply changing the index-constant in the breakpoint. Note that the index will differ between different Windows versions and service packs. For example, if ntdll!NtCreateFile has index 0x25 on Windows XP SP3 it might have the value 0x34 on Vista. Fortunately we can easily obtain the correct value by examining ntdll.dll...

I hope this helps,
Robert

==========

## Re: Memory Access errors in the Kernel
Posted by Brett K - 12 May 2010 - 00:46

_____

Thanks for the help! I learned a bit about SYSENTER and was using...

rdmsr 176
bp /t @$thread addr

...as a one-shot break point on going into the Kernel, but I will have to give your bp script a try as well.

To do this you must ensure the context of your process is actually used (its virtual addresses space paged-in etc.). A simple .process ?? and .reload /user should fix your memory issues described.

I'm not sure I understand what this means. Do you have a more detailed explanation?

==========

## Re: Memory Access errors in the Kernel
Posted by Robert Kuster - 28 May 2010 - 15:46

_____

Brett, hi again.

The following excerpt of the .process (Set Process Context) command explains it quite well. You might also take a look at .context (Set User-Mode Address Context) which is a very similar one.

Typically, when you are doing kernel debugging, the only visible user-mode address space is the one that is associated with the current process.

The .process command instructs the kernel debugger to use a specific user-mode process as the process context. This usage has several effects, but the most important is that the debugger has access to the virtual address space of this process. The debugger uses the page tables for this process to interpret all user-mode memory addresses, so you can read and write to this memory.

In short the following picture explains it:

http://windbg.info/images/fbfiles/images/kernel_vs_user_mode.PNG
Note that at any time there is only one active user-mode process. A kernel-mode debugger will only see user-mode addresses of this one process. To peek into another process's memory you have to switch the context via the .process or .context command.

There a few concepts that you should understand about a multitasking and multithreading environment, about paging, process address spaces, virtual memory, address translations, and so on. I suggest that you take a look at these articles which explain it all very well:

http://en.wikipedia.org/wiki/Architecture_of_Windows_NT
http://en.wikipedia.org/wiki/Page_table
http://en.wikipedia.org/wiki/Address_space
http://en.wikipedia.org/wiki/Memory_management_unit
http://en.wikipedia.org/wiki/Virtual_memory
http://en.wikipedia.org/wiki/Paging
http://en.wikipedia.org/wiki/Context_switch
http://en.wikipedia.org/wiki/Thread_%28computer_science%29

About Processes and Threads
Process Address Space

I hope this helps,
Robert

==========

## Re: Memory Access errors in the Kernel

Posted by Brett K - 28 May 2010 - 22:09

---

Thanks again for you help, I fully understand it now!

==========