Common WinDbg Commands (Thematically Grouped)
Posted by Robert Kuster - 01 Feb 2009 - 01:00

_____

** This thread discusses the content article: Common WinDbg Commands (Thematically Grouped) **

==========

Pattern matching
Posted by adrian hodos - 12 May 2009 - 18:17

_____

First thank you for compiling this document , it is very good. I have a small problem though with pattern matching and conditional breakpoints. I have tried setting a conditional breakpoint on LoadLibraryExW like the examples in this document. The name of the dll I'm trying to match is protection_engine.dll , the pattern I use is *protect*. I've set the breakpoint like this: bu kernel32!LoadLibraryExW ";as /mu ${/v:MyAlias} poi(@esp+4); .if ($spat(@"${MyAlias}", "*protect*") != 0) { .echo ok - dll loaded; kP; } .else { g }". However it only stops when it's loading comctl32.dll so there must be something wrong in the syntax. Do you have any ideas ? I've been staring at it for quite some time but I can't figure out where I'm doing wrong.

==========

Re: Pattern matching
Posted by Robert Kuster - 14 May 2009 - 03:05

_____

Hey hey Adrian,

Thanks for your feedback. I encountered similar problems with this breakpoint command. It turns out that here and then the aliases get messed up by WinDbg. You can easily check what is going on by the "al" (alias list) or "bl" (breakpoint list) commands. If you see some unusual values the following will help:

a) "ad *" == deletes all aliases
b) Reset the breakpoint in question

I'll try to find a more stable version for this breakpoint command and publish it later this year with the next "WinDbg Commands" update.

Kind Regards,
RK

==========

windbg question
Posted by kam - 12 Feb 2010 - 18:28

_____

Hi,

This article is very nice, I'm a beginner to windbg and this is helping me a lot.
Also, I have a question:
Could you tell me please, how do I set a bp on a driver entrypoint if driver name is hex convertible?

example:
driver name is 77fba431.sys
so, normally I would do something like "bp 77fba431+rva_entrypoint" (just like lets say "bp ntfs+rva") but of course 77fba431 is read as an address, so windbg will actually set a bp to address 77fba431+rva_entrypoint. So, how can I tell windbg that 77fba431 is actually a module name not an address ?

Thank you!

==========

Re: windbg question from kam
Posted by Robert Kuster - 18 Feb 2010 - 18:02

_____

Kam, hi.

Let's assume that the entry point of your driver is called DriverEntry. In this case setting a breakpoint is simple:
> bp 77fba431!DriverEntry

And if you prefer to work with offsets you can easily get the base address of your driver too:
> !lmi 77fba431
OR
> lm vm 77fba431

Both commands will return the base/start address of your driver in memory. Then you would do something like this:

> bp BaseAddress (retrieved in previous step) + rva_entrypoint

I hope this helps,
Robert

==========

windbg question
Posted by kam - 19 Feb 2010 - 21:36

_____

Hi,

Let's say that the driver I want to debug doesn't have symbols, so I can't use DriverEntry.

!lmi 77fba431 (same problem: address not found (so name is interpreted as hex))
Also, the problem is that I would like to set a breakpoint before the driver is loaded. So I would need something like "bu driver+rva_ep".

Your method to retrieve base_address and the set a "bp" means that driver needs to be already loaded, right? so why would I set a bp on entry point then...

The only method that works that I've found so far, is "bp" on a driver with symbols, walking up the stack and find the address of driver loading function, just before it's calling drivers' EP, and breakpoint there. The problem is that this method will break on every driver that is going to be loaded...

==========

Re: break on driver load - question from kam

Posted by Robert Kuster - 02 Apr 2010 - 21:15

Kam,

In general you don't need symbols to know the entry point of a PE image. The entry point is conveniently stored to the PE header and can be read from it. Further the OS loader loads an image into memory (be it an EXE, DLL, or kernel mode driver) and calls its entry point thereafter. In other words by the time DriverEntry is called the driver will always be loaded. If all you need is break into WinDbg after a driver is loaded but before its entry point is called the situation is simple. And if you want to break even before the image is loaded into memory the situation is still simple enough. I described both scenarios bellow.

1) TODOs - break after driver load & before its entry point is called

1) Break into WinDbg -> Debug (menu) -> Event Filters
2) In the "Event Filters" select the "Load module" event
  > select Execution -> Enabled
  > click Arguments and enter the name of the driver in question (77fba431 in our case)

http://windbg.info/images/fbfiles/images/event_filters.PNG

With this settings WinDbg will brake after loading any driver with the name 77fba431 and before calling its entry point. Here is what happens:

```
0: kd> .lastevent
Last event: Load module 77fba431.sys at ba644000
  debugger time: Wed Mar 31 21:12:56.937 2010 (GMT+2)

0: kd> lm vm 77fba431
start    end        module name
ba644000 ba645e00   77fba431   (deferred)
    Image path: 77fba431.sys
    Image name: 77fba431.sys
    Timestamp:        Tue Mar 30 20:10:51 2010 (4BB23EAB)
    CheckSum:         0000AD61
    ImageSize:        00001E00
    Translations:     0000.04b0 0000.04e4 0409.04b0 0409.04e4

;get ImageEntry (== DriverEntry)
0: kd> ? $iment( ba644000)
Evaluate expression: -1167828646 = ba64595a
```

```
;now that we have the DriverEntry address we can conveniently set a breakpoint on it
0: kd> bp ba64595a
*** ERROR: Module load completed but symbols could not be loaded for 77fba431.sys

0: kd> bl
 0 e ba64595a     0001 (0001) 77fba431+0x195a

0: kd> g
Breakpoint 0 hit
77fba431+0x195a:
ba64595a 8bff            mov     edi,edi
```

2) TODOs - break before our driver is loaded into memory (XP SP 3)

Behind the scenes a driver is loaded by nt!IopLoadDriver. Its pseudo-code looks like this:

```
nt!IopLoadDriver( hRegKeyOfDriver, ...)
{
   // get full path of our driver
   UNICODE_STRING driverPath = call nt!IopBuildFullDriverPath(...);

   // load driver into memory
   call nt!MmLoadSystemImage( driverPath , ...)

   // retrieve drivers entry point and call it
   call nt!RtlImageNtHeader
   call nt!IopPrepareDriverLoading
   call    dword ptr        // call DriverEntry
}
```

With this on mind we can use the following conditional breakpoint. It will cause WinDbg to break if the driver being loaded contains the name "77fba431" and continue execution in any other cases:

```
bu nt!IopLoadDriver ".block {as /c HandleOutput !handle poi(@esp+4)};  .block {.if ( $spat(
"${HandleOutput}", "*77fba431*" ) ) { .echo ***** Loading our driver *****; ad *; } .else { ad *; g;}};"
```

Explanation:!handle poi(@esp+4) ..... get handle information and registry path of hRegKeyOfDriver; store it into HandleOutput (string alias)$spat("${HandleOutput}", "*77fba431*") .... is 77fba431 found in HandleOutput? If yes break. If not go (g)
I peeked into nt!IopLoadDriver on Windows XP SP3. It might bee slightly different on Windows Vista or Windows 7 installations. Nevertheless I think you got the idea and will be able to change the breakpoint if necessary. You could set a similar conditional breakpoint on nt!MmLoadSystemImage. Note that its first parameter is the path of the driver:

```
0: kd> kb
ChildEBP RetAddr  Args to Child
ba507c74 8058107b ba507cf8 00000000 00000000 nt!MmLoadSystemImage
ba507d54 80581487 80000748 00000001 00000000 nt!IopLoadDriver+0x371
ba507d7c 80538789 80000748 00000000 8a8e8640 nt!IopLoadUnloadDriver+0x45
ba507dac 805cff72 ae22acb0 00000000 00000000 nt!ExpWorkerThread+0xef
ba507ddc 8054611e 8053869a 00000001 00000000 nt!PspSystemThreadStartup+0x34
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
```

```
0: kd> dS ba507cf8
e10e69b8  "??C:CpDrv77fba431.sys"
```

I hope this helps,
Robert

==========