How can I get the name of the function that crashed given just a module name and offset?

devblogs.microsoft.com/oldnewthing/20160302-00

March 2, 2016

Raymond Chen

The only information a customer had regarding a crash was the following:

> Faulting application name: Contoso.exe, version: 1.0.0.0, time stamp: 0x4a425e19
> Faulting module name: Contoso.exe, version: 1.0.0.0, time stamp: 0x4a425e19
> Exception code: 0xc0000005
> Fault offset: 0x000050d0
> Faulting process id: 0x1910
> Faulting application start time: 1cad18414e63580

They wanted to know what function crashed.

This is an application of techniques for restoring symbols to a stack trace that was generated without symbols, but in the simplified case where there is only one address, not an entire stack trace (so you need to do the work only once), and in the special case where all you have is a module name and an offset.

The first step is to find the correct executable. The time stamp in the event log is `0x4a425e19`, which we recognize as a UNIX style timestamp. This handy online converter says that it's June 24, 2009 at 17:10:49 GMT. Dig into your archives and find a build generated around that time and check the time stamp in the file header. The `link /dump /headers` command will tell you:

```
FILE HEADER VALUES
             14C machine (x86)
               3 number of sections
        4A1ECBC2 time date stamp Thu May 28 10:37:06 2009
```

Okay, that's the wrong one since the time stamps don't match. Keep looking until you find the right one, and also grab its matching symbol file (`contoso.pdb`).

Once you do, you can load it up in the debugger.

```
C:\> ntsd -z contoso.exe

ModLoad: 00100000 00130000   contoso.exe

0:001> u 0x00100000+0x50d0 L1
contoso!CViewReportTask::Run+0x102:
001050d0 8a18            mov     bl,[eax]
```

Okay, so at least you know that the crash was in the `CViewReportTask::Run` method. You can also ask for line number information:

```
0:001> .lines
Line number information will be loaded
0:001> u 0x00100000+0x50d0 L1
contoso!CViewReportTask::Run+0x102 [viewreporttask.cpp @ 250]:
001050d0 8a18            mov     bl,[eax]
```

We see that the crash was on line 250.

To figure out what part of line 250, you'll have to dig into the disassembly and reverse-compile the code to see exactly which part of line 250 is being executed at `001050d0`. You don't know what value is in any of the registers, so all you know is that the pointer is invalid; you don't know whether it is null or wild, or how it got that way.

Bonus chatter: You probably should sign up for Windows Error Reporting so that you will receive crash dumps automatically, which provide a full stack trace instead of a single address, and it also captures register values and limited contents of the stack. You can also ask for more information to be captured in future crash dumps.

Bonus exercise: Use your time stamp recognition skills to determine what *Faulting application start time* corresponds to.

Raymond Chen

Follow