**.NET Debugging Demos Lab 2: Crash**

It was nice to see that so many people downloaded the demo site already and checked out the lab instructions for the first lab.

Here comes lab 2, a crash scenario on the BuggyBits site.

**Reproduce the problem**

1. Browse to the **Reviews** page , you should see a couple of bogus reviews for BuggyBits
2. Click on the Refresh button in the reviews page. This will crash the web host process (iisexpress.exe)

**Examine the event logs**

1. Open the Application and System event logs, the information in the event logs will differ based on the OS and web server you are running. Among other events you may have a System Event looking something like this

**Note:** In later years - the descriptions here have gotten a bit more descriptive

Event Type: Warning

Event Source: W3SVC

Event Category: None

Event ID: 1009

Date: 2008-02-08

Time: 10:12:06

User: N/A

Computer: MYMACHINE

Description:

A process serving application pool 'DefaultAppPool' terminated unexpectedly. The process id was '4592'.

The process exit code was '0xe0434f4d'.

For more information, see Help and Support Center at http://go.microsoft.com/fwlink/events.asp.

* What events do you see?
* What does the exit code 0xe0434f4d mean?
* Can you tell from the event logs what it was that caused the crash?

**Get a memory dump**

1. Browse to the reviews page again, but don’t click refresh
2. Run procdump with -e
3. procdump64.exe -ma -e iisexpress.exe
   * What is the debugger waiting for? Hint: Check the procdump help for -e
4. Reproduce the issue by clicking on the refresh button in the reviews page and watch the procdump window to verify if it captures a dump

**Open the dump in windbg**

1. Open the dump file in windbg (file/open crash dump).

**Note:** if you throw an exception (.net or other) you have a chance to handle it in a try/catch block. The first time it is thrown it becomes a 1st chance exception and is non-fatal. If you don’t handle the exception it will become a 2nd chance exception (unhandled exception) and any 2nd chance exceptions will terminate the process.

1. Set up the symbol path and load sos (see the setup instructions for more info)

In a crash dump, hte active thread is the one that caused the exceptions (since the dump is triggered on an exception).

* + Which thread is active when you open the dump? Hint: check the command bar at the bottom of the windbg window.

**Examine the call stacks and the exception**

1. Examine the native and managed call stacks.
2. kb 2000
3. !clrstack
   * What type of thread is it?
   * What is this thread doing?
4. Examine the exception thrown
5. !pe

Note: !pe/!PrintException will print out the current exception being thrown on this stack if no parameters are given

* + What type of exception is it?

**Note:** In some cases, like this one where the exception has been re-thrown, the original stacktrace may not be available in the exception. In cases like this you may get more information if you find the original exception

1. If your exception was re-thrown, look at the objects on the stack to find the address of the original exception
2. !dso
   * What is the address of the original exception?

Hint: Look at your previous pe output to see the address of the re-thrown exception. Compare this to the addresses of the objects on the stack. You should have multiple exceptions, a few with the re-thrown exception address but one of the bottommost exceptions will be the original one (look for one with a different address).

1. Print out the original exception and look at the information and the call stack
2. !pe <original exception address>
   * In what method is the exception thrown?
   * What object is being finalized?

**Note:** you could actually have gotten this information by dumping out the \_exceptionMethodString of the re-thrown exception as well, but with !pe of the original exception you get the information in a cleaner way.

* + Normally exceptions thrown in ASP.NET are handled with the global exception handler and an error page is shown to the user. Why did this not occur here? Why did it cause a crash?

**Examine the code for verification**

1. Open the code for the Review class to find the destructor/finalizer for the Review class
   * which line or method could have caused the exception

As an extra exercise you can also examine the disassembly of the function to try to pinpoint better where in the function the exception is caused

!u <IP shown in the exception stack>