Kernel Debugging and Crash Analysis for Windows®

Overview

When Windows detects an inconsistency within the operating system that's too big to ignore, it crashes and displays the infamous Blue Screen of Death. Optionally, the system also writes the contents of memory at the time of the crash to a crash dump file.

The successful analysis of a crash dump requires a good background in Windows internals and data structures. But it also lends itself to a rigorous, methodical approach. Crash analysis is a skill that can be taught and learned. And that's precisely what we do in this intensive 5-day, hands-on seminar.

Seminar Formats

This seminar is available as a 5 day lecture with labs, which can be customized with additional or different topics for presentation at your location. Please contact an OSR Seminar Coordinator for details on arranging an on-site seminar.

Target Audience

Driver developers who need to understand how to set up for, use, and analyze OS crash dumps. Support personnel who need to know how to debug Windows system crashes in the lab or at customer sites, in situ or post mortem. Advanced, hands-on, systems administrators who are comfortable with Windows systems internals and need to be able to work through system crashes to identify the proximate reasons for the crashes they are seeing.

Prerequisites

This is an intermediate-level seminar offering for active practitioners. This is not a class for beginners. All attendees are expected to understand operating system concepts in general, and the basic concepts of the Windows operating system. Attendees must have previous hands-on experience working with Windows, including some (moderate) applications or driver level development experience. It is a hands-on class intended to give students real, practical experience in using the Windows kernel debugger (WinDBG) and in understanding the data provided by the various kernel debugger extensions.

Hardware Requirement

Students are expected to bring their own laptop, with the 32-bit version of WinDbg (the Windows kernel-mode debugger) pre-installed and tested. The most recent version of the Debugging Tools for Windows is distributed as part of the Windows SDK, and may be downloaded from here: http://msdn.microsoft.com/en-us/windows/hardware/gg463009.aspx -- We recommend laptops have a minimum of 20GB of free space (though this is not a requirement). If it is impossible for you to provide a suitable laptop, call our seminar coordinator to discuss your options.

Seminar Outline

- **Principles of Debugging**
  Debugging is something we do in everyday life - it is nothing more than problem solving, working to "figure it out". The techniques we use with other aspects of life also apply to debugging with Computer Systems. We look at the fundamental precepts as well as the specific environment in which we will be operating.

- **Introduction to WinDBG**
  WinDBG is the Windows debugger, used primarily for kernel mode debugging although it also can be used to debug applications. This initial section describes the basics of the tool and provides some focused discussions on how to use it for kernel debugging. Note: the goal is not to provide a comprehensive overview of the tool. Students are referred to the WinDBG documentation for a thorough description of the abilities of this tool.

  Lab: Using WinDBG to examine the local (student) system and become familiar with basic debugger operations.

- **Overview of the x86 Processor**
  Most of the current "real world" debugging is done based upon x86 platforms. In this section we discuss some basic characteristics of the processor architecture as well as how the specific characteristics of the processor architecture are used within Windows, framed in the context of our interest in debugging.

  Lab: Using information collected from the previous lab, students are asked to describe the behavior of the given Windows functions. In addition, students are then asked to build upon this by choosing one other function and repeating this process. Upon completion of the lab, students will discuss their findings with the rest of the class.

- **Overview of x64 Processor**
  This section describes the basics of the ia64 and amd64 processor architectures. Unlike the x86 architecture, their use of the system is far more "structured" and this simplifies the debugging process.

  Lab: Using a post-mortem dump provided by the instructor, students will repeat the process from the previous lab, this time using an amd64 system image.

Kernel Debugging & Crash Analysis for Windows   Page 2
Copyright (c) 2017 OSR Open Systems Resources, Inc.

- **Calling Conventions**
  One of the most important aspects of debugging is the ability to find the origin of data. This module describes the commonly used calling conventions (with the primary focus on x86) that are normally used by the Windows OS compilers. In addition, we will describe the process of structured exception handling and show code examples of how 'C' code is converted into assembly.

  Lab: For this lab, students are provided with a post-mortem dump and asked to analyze the failed system. The goal then is to have students construct a theory of what caused the failure; in this instance, it ties together concepts of structured exception handling described in this section, although in a context for which the debugger provides basic support.

- **Windows Data Structures**
  This section ties together specific Windows OS data structures together, providing students with a better understanding of how Windows works and then tying those data structures into the process the debugger uses for extracting information. The ultimate goal of this discussion is to further ground students so they can further hone their understanding and skills with the kernel debugger.

- **Fault Isolation**
  Discussion of the process of isolating faults and attempting to find the "root cause". This is a short section striving to "tie together" the concepts from previous modules as students continue to hone their debugging skills via "hands on" interactions.

  Lab: For this lab, students are provided with a post-mortem dump and asked to analyze the failed system. The goal then is to have students construct a theory of what caused the failure and to use the skills built up to this point to extract as much information as possible. Ultimately, students are expected to be able to write a coherent and well-targeted bug report for this post-mortem crash.

- **Handling Deadlock and Livelock**
  Defining deadlock and livelock and examining their causes. Discussion of common deadlock causes, such as file system reentrancy and worker thread exhaustion. Techniques for determining what is causing deadlock are discussed.

  Lab: For this lab, students are provided with three post-mortem dumps. The first two are synthetically generated deadlock scenarios utilizing different types of resources. The third is a "real world" deadlock scenario demonstrating a worker thread related deadlock. The goal for all of these cases is to have students identify the threads and resources involved, thereby allowing them to write a well-targeted and highly useful bug report.

- **Moving Beyond the Debugger**
  Discussion of kernel debugger extension libraries, how they are used and how they are constructed.

  Lab: For this lab, students are provided with a post-mortem dump and asked to analyze the failed system. The goal then is to have students construct a theory of what caused the failure and to use the skills built up to this point to extract as much information as possible. Ultimately, students are expected to be able to write a coherent and well-targeted bug report for this post-mortem crash.

- **Virtual Memory**
  Most system crashes are software related. Even more so, most of these crashes are caused by some type of invalid memory reference. In order to understand how and why these crashes occur, a clear and complete understanding of the Virtual Memory subsystem is required. In this section students will be introduced to basic virtual memory concepts as well as Windows' usage of virtual memory.

- **Demonstration (based upon time available)** Students are invited to bring their own post-mortem or live system crashes for demonstrative analysis. If no student crashes are available, the instructor will have some post-mortem crashes to use for further demonstration.

Note: Individual lab sessions are not provided with any sort of "answer key". Rather, at the completion of each lab session the instructor will do a live "walk through" of the lab exercise, demonstrating appropriate technique. Logs of those sessions will be taken at that time and made available to students. We do this because the tools are in a constant state of flux and this ensures we provide students with demonstrations that are appropriate to the current state of the tools.