Kernel Debugging and Crash Analysis for Windows® with IDA Pro

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 using a combination of WinDbg and IDA Pro. 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 both the Windows kernel debugger (WinDbg) and Hex-Ray's IDA Pro debugger/disassembler.

Hardware Requirement

Students are expected to bring their own laptop, with the 32-bit version of WinDbg (the Windows kernel-mode debugger) and Hex-Ray's IDP Pro 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.

- **Windows Operating System Architecture Overview**
  A brief review of the general architecture of the Windows operating system, including topics most relevant to debugging.

- **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.

- **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.

- **Introduction to IDA Pro** IDA Pro is a powerful disassembler and debugger, created by Hex-Rays SA. Recent versions contain support for the Windows debugging interface. In this section, we describe the basics of the tool and how it can be used best for disassembling and tracing paths within Windows kernel mode modules.

- **Overview of the x86 Processor**
  Crashes aren't random events, even though it might sometimes seem that way! In reality, a series of events occur on the processor that result in Windows deciding to crash the machine. In order to properly reconstruct the series of events leading up to the crash, we must have an understanding of the underlying processor architecture on which Windows is executing.

In this section we discuss some basic characteristics of the x86 processor architecture, particularly focusing on those features that are used within Windows. The Application Binary Interface (ABI) used by Windows is also discussed, including calling conventions (CDECL, STDCALL, FASTCALL, and LTCG) and stay frame layouts (EBP and FPO).

- **Overview of x64 Processor**
  This section describes the basics of the x64 processor architecture. The processor architecture itself is a natural evolution from the x86, though the ABI has changed considerably. Windows uses of the Home Space and Unwind Data are discussed, as well as how the ABI can be leveraged in a debugging environment.

- **Debugging Invalid Memory Access Problems**
  No crash is more common in a Windows driver than one due to dereferencing a bad pointer, or referencing a “good” pointer at the wrong IRQL. In this module, we’ll discuss how Windows traps invalid memory references, and the specific differences between the bug check codes IRQL_NOT_LESS_OR_EQUAL, PAGE_FAULT_IN_NONPAGED_AREA, and KERNEL_DATA_INPAGE_ERROR, with the goal of making it easier and faster to debug commonly encountered driver problems.

- **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.

- **Handling Deadlock and Livelock**
  Defining deadlock and livelock and examining their causes. Techniques for determining what is causing deadlock are discussed.

- **Advanced Debugger Usage**
  Cover advanced WinDbg command line syntax, including discussions of the MASM and C++ expression evaluators. The WinDbg scripting language is discussed, as well as the underlying Debugger Engine Library as the basis for both C based extensions and standalone debugger applications.

- **Testing and Debugging Best Practices**
  A random collection of topics aimed at providing the students with practical, useful tools for streamlining their testing or debugging environments. Creation of Symbol and Source servers is discussed. Using KDFiles to automatically update the driver under test. Providing remote access to a kernel debug session.

- **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: Lab sessions are provided at the end of each day (excluding the first day) to reinforce the topics discussed. 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 and answering specific student questions.