WinDbg User-Mode Debugging: Beginner Edition

Objective

Debugging can be a tedious and involved process. And with Windows being such a complex operating system, it is difficult for a beginner to know what to do and where to begin. WinDbg is the debugger of choice for debugging the Windows operating system; one that is used by Microsoft itself! It's large feature set makes it a powerful debugger, but it also makes it an intimidating tool for anyone getting started on it.

This bootcamp is aimed at providing a good understanding of the use of WinDbg in user-mode debugging. During the bootcamp, you will get better acquainted with Windows concepts such as processes and threads, and how Windbg can be used to debug them.

The bootcamp is completely hands-on with accompanying lab exercises to help you get familiar with WinDbg.

Prerequisites

- Basic understanding of processes, threads, virtual memory, handles and objects
- Basic programming experience in any language

Course Content

A non-exhaustive list of topics that will be covered includes:

Module I: Introduction

Objective:

Learn about the Debugging Tools for Windows package and the debuggers included in it, including WinDbg, and learn to install and set up the environment. Also, get introduced to different types of commands supported by WinDbg, the basics of user-mode debugging, the different debugger windows, and the difference between 64-bit vs 32-bit debugging.

- Overview of Debugging Tools for Windows
- Debuggers included in Debugging Tools for Windows
    - Cbd
    - Ntsd
    - Kd
    - WinDbg
- Installation and Environment setup
- Types of Commands
    - Regular Commands
    - Meta Commands
    - Extension (or "bang") Commands
- User Mode Debugging
- Debugger windows
    - Stack
    - Threads
    - Breakpoints
    - Locals
    - Watch
    - Disassembly
- 64-bit vs 32-bit Debugging
    - Overview
    - Calling Convention

Module II: Basic Debugger Operations

Objective:

Learn about the basics of user-mode debugging. Get acquainted with symbol files and configure a symbol server to download the symbols for debugging an application. Learn the fundamental commands that would come in handy while doing user-mode debugging. Debug processes and threads both with and without source code by setting breakpoints, inspecting registers, stack, memory, threads, getting more information on the process and threads structures, address space of the process and the modules loaded in a process.

- Symbols
    - Introduction
    - PDB Files
        - Private Symbols
        - Public Symbols

    - Symbol Server
        - Using the official Symbol Server
    - Configuring Symbols
        - For developer-created binaries
        - For Windows binaries
            - _NT_SYMBOL_PATH environment variable
            - .symfix debugger command
- Processes and Threads
    - Debugging a process / executable
    - Debugging multiple processes
    - Loaded Modules
    - Inspecting threads
    - Inspecting process associated data structures
    - Inspecting thread associated data structures
    - Switching between threads
    - Call Stack
    - Breakpoints
        - Setting breakpoints
        - Managing breakpoints
            - Listing breakpoints
            - Enabling, Disabling, Deleting breakpoint(s)
    - Inspecting Registers
    - Displaying Data
    - Disassembling instructions
    - Modules
        - Examining symbols
        - Searching symbols
        - Inspecting Module Information
    - Process address mapping
    - Detaching the process
- Source Level Debugging
    - Loading source code
    - Setting breakpoints and watchpoints
    - Inspecting different windows for values
    - Modifying data

Module III: Advanced Debugger Operations

Objective:

Learn about exceptions and events and how to configure the default behaviour of a debugger when an exception or event occurs. Also learn how to set advanced breakpoints including conditional, memory access, pattern and hit breakpoints. Learn about the different syntax supported by the debugger and how to switch between those. Also learn memory manipulation, child process debugging, generating and analyzing dump files and analyzing the application heap.

- Exceptions and Events
  - Configuring events
  - Configuring exceptions
- Advanced Breakpoints
  - Conditional breakpoints
    - Breakpoint on a certain thread
    - Execute commands on breakpoint hit
    - Hit count
  - Memory Access breakpoint
  - Unresolved breakpoints
  - Pattern breakpoints
- Expression Syntax
  - MASM and C++
  - Switching between expression syntax
- Memory Manipulation
  - Editing/enter memory
  - Fill memory
  - Move memory
  - Search memory
  - Compare memory
- Debugging Child Processes
  - Enabling child process debugging
  - Listing debugged processes
  - Switching between processes
- Dump Files
  - Dump File types
    - "Mini" dump
    - Full dump

    - Generating process dump
        - Using Task Manager
        - Using Process Explorer
    - Crash Dump Analysis
- Application Heap Analysis

Module IV: Practical Debugging Scenarios

Objective:

Apply everything learnt in the previous modules to analyze a deliberately vulnerable application containing issues such as heap overflows, race condition and memory leaks.

- Analysis of application containing the following issues:
    - Race Condition
    - Memory issues
        - Thread Handles leak
        - Use After Free
        - Heap Overflow