

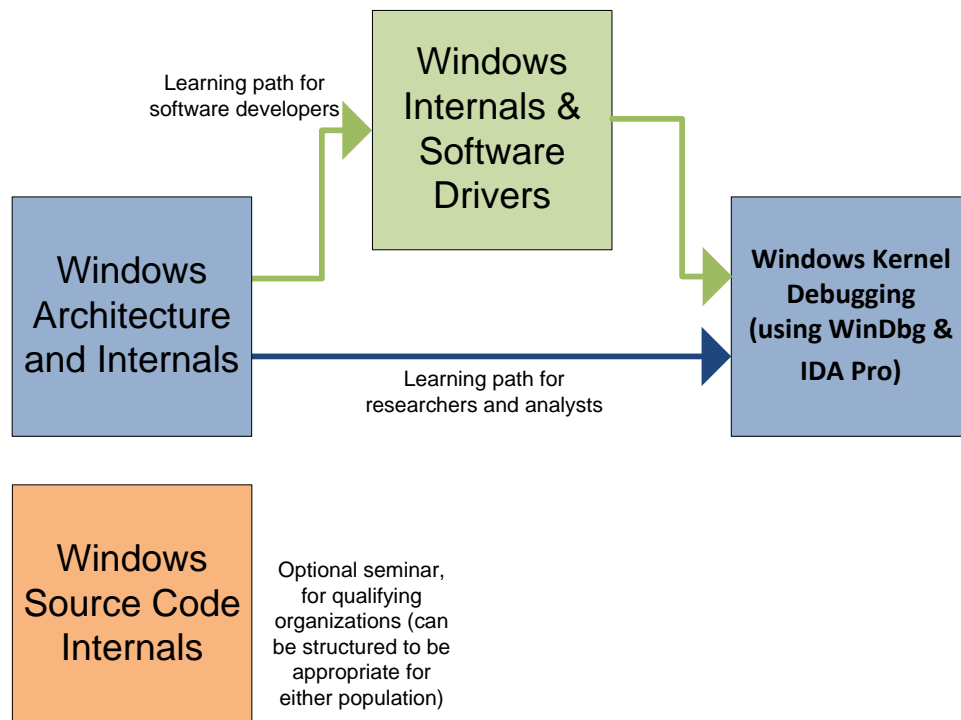
Security and Cyberwarfare Curriculum

Curriculum Overview

Given: The world is not the cyber-friendly place it once was. To enable students to fully understand both threats and potential mitigations in the Windows operating system environment, OSR has employed its unique knowledge of Windows internals and system software to create a unique new curriculum. This curriculum is designed specifically to meet the needs of those working in the security and cyberwarfare fields.

The seminars within the curriculum can be taken immediately following each other or can be selected and taught individually.

Various parts of the curriculum are appropriate for cyberwarfare and security researchers, analysts, and software developers, as shown by the learning map below. Seminars appropriate for all students in the security and cyberwarfare discipline (researchers, analysts, and software developers) are shown in blue blocks. Seminars that are most appropriate for software developers only are shown in green blocks.



As show on the learning map, all students start with the *Windows Architecture and Internals -- Security Edition* seminar, where they master the basics of Windows operating systems concepts. This includes the key tables and structures maintained by the operating system, and which are often the target of manipulation by malware.

Researchers and analysts who are not directly involved in software design or development continue their training with *Windows Kernel Debugging -- Security Edition*.

This hands-on seminar allows students to put their Windows internals knowledge to use, detecting threats and unraveling operating system problems.

After attending *Windows Architecture and Internals*, software developers and architects continue their training with *Windows Internals and Software Driver Development*. In this hands-on seminar, students master the information and skills necessary to write kernel-mode code to monitor, and potentially affect, Windows system operation. After this seminar, these students may optionally attend *Windows Kernel Debugging* to build their skills in problem diagnosis and detections.

OSR's seminars do not typically require Windows source code access. For organizations that have a Windows source code license, OSR offers one of its most interesting seminars: *Windows Source Code Internals*. This seminar takes students on a tour of key parts of the Windows operating system source code, focusing on the Kernel, Memory Manager, and I/O Subsystem. There is no better way to learn key portions of the Windows OS source code than to be guided through it by an experienced Windows kernel developer. This is exactly what OSR provides in this seminar.

Flexible Seminar Formats and Logistics

Many of OSR's seminars are available in both lecture-only and lecture-with-lab formats. While we typically recommend the lecture-with-lab format to maximize hands-on learning, lecture-only seminars have the advantage of being shorter and easier to manage logistically.

For lab sessions, OSR provides numerous flexible options, including:

- OSR supplies all necessary computer infrastructure (systems, networks, etc);
- Using established in-house or commercial computer training facilities;
- Doing labs using (appropriately equipped) student-supplied laptop computers.

About OSR

With more than 15 years of experience working with Windows at the source code level and with clients all over the world, OSR is the recognized leader in Windows system software design, development, and training.

OSR has a long record of successfully working with leading-edge organizations in both the public and private sectors. It is a little-known fact that OSR's software tool frameworks form the basis of the majority of the anti-virus systems deployed world-wide. And OSR has designed and developed numerous complex software solutions, from device drivers to file systems to full disk encryption.

OSR's engineering team brings its hands-on analysis and development experience to every seminar they teach. Only our most experienced engineers are allowed to teach our seminars. Aside from provided training to all the "big names" in the computer industry (OSR provides regularly scheduled seminars for software engineers in Microsoft's Windows group), OSR has provided training to numerous government agencies and sub-

contractors world-wide, especially those focused in the security, intelligence, electronic warfare, and cyberwarfare fields.

OSR is also active in the Windows system software community: OSR publishes *The NT Insider*, a journal of Windows system software development. OSR hosts the well-known NTDEV, NTFSD, and WINDBG mailing lists and forums. Peter Viscarola and Tony Mason, OSR's Consulting Partners, are the authors of *Windows NT Device Driver Development* (New Riders Press, ISBN 1578700582) the best-selling book about Windows drivers of all time.

You can find out more about OSR, our areas of expertise, and the customers we serve, at our web site: <http://www.osr.com>

Windows Architecture and Internals Security Edition

Overview

This seminar "opens the black box" that is, for most people, the Windows operating systems and gives students a chance to look inside. Using a systematic approach, the seminar describes Windows basic operating system concepts, the fundamentals of various key processes as implemented by Windows, and the basic data structures on which the operating system relies.

Seminar Formats

This seminar is available as a 2 day lecture, 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

Security researchers, analysts, and software developers involved in security and threat analysis/modeling or working in the areas of intelligence and information warfare or cyberwarfare.

Prerequisites

Knowledge of common operating systems concepts will be assumed.

Seminar Outline

1) Windows System Architecture

A review of the general architecture of the Windows operating systems, including the organization of the Executive. Also covered: Threads and processes, system service processing, I/O flow, key and common Windows processes (CSRSS, LSASS, svchost, etc).

2) Windows Data Structures

This section describes major data structures and tables that Windows uses to manage its operations. Process structures, thread structures, I/O subsystem objects (Device Object, File Object, etc). System tables (such as the System Service Dispatch Tables). While this section delves deeply into the format of these structures, the emphasis is on how they are used and how the structures inter-relate.

3) Virtual Memory

How and why Windows implements virtual memory for user applications and the operating system. Page Tables, Page Directories, the PFN, and the VAD. Memory manager policy, including working sets. How the memory manager can be tuned.

4) The Registry

How the Registry can be viewed and changed. How the Registry is organized. Specific keys and values of where common system information is stored. System tuning parameters located in the Registry. The concepts of registry "hooks" and callbacks are briefly discussed.

5) Dispatching and Scheduling

A discussion of how the OS selects the next thread to run. Includes discussions of scheduling on both single and multiple CPU systems and NUMA architectures.

6) The I/O Subsystem

This section describes the architecture and structure of the Windows I/O Subsystem. Disk caching and the cache manager are discussed. There is also a detailed discussion of how device stacks are built by the Plug and Play Manager, and how requests are forwarded from device to device (and hence driver to driver) down the device stack.

7) Inter-process and Network Communications

How inter-process and network communications are implemented on Windows. Includes a discussion of how the network communications stack is implemented.

8) Interrupt Request Levels and DPCs

How Windows manages interrupts, and queues completion requests.

9) Event Tracing and Trace Points

Windows has an extensive, built-in, trace logging system. Using these trace points, sophisticated users can gain an understanding of what's happening within a Windows system. In this modules, Event Tracing is described in detail and examples of using this tracing are shown.

10) Tools for Internals

A review of helpful internals tools that are distributed with the OS or as part of the Resource Kit. These include tools that shed light on activities such as the Performance Monitor MMC snap-in, or that help you view special information or perform specialized activities like Traceview, OH, POOLMON, WINOBJ, and GFLAGS. Also includes brief discussion of the tools necessary for developing and debugging kernel mode software, and how a debug environment is setup for decoding the mysteries of crashed systems and crash dumps.

Windows® Internals and Software Driver Development

Overview

This seminar is designed for software engineers and architects who need to understand the details of the major Windows architectural components, as well as how to create software only kernel-mode drivers that serve as "extensions" to the Windows operating system. The seminar includes a review of basic Windows architecture, and covers the basics of building and debugging kernel-mode drivers for Windows. It then discusses the details of driver structure, including the I/O and PnP subsystems, device discovery, enumeration, and interrupt processing. Other extension mechanisms, such as Object Manager, Process Manager, and Registry callbacks are discussed. The seminar also includes in-depth discussions of the Windows Kernel, process and thread instantiation, and the Memory Manager.

During lab sessions, participants create and modify software-only drivers to perform various kernel-mode tasks. Note that this seminar focuses on the development of drivers that service as Windows "operating system extensions" but do not service any hardware. Thus, while the basic concepts of drivers for devices are discussed, the lab sessions focus strictly on the development of software (i.e. pseudo) drivers which typically use "legacy" or "NT V4" style interfaces (as opposed to PnP).

Seminar Formats

This seminar is available as a 3 day lecture or a 5 day lab session. Either version 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

Security researchers, government contractors, engineers involved in security and threat analysis/modeling, who need a solid understanding of key Windows internals and data structures, and the options available to monitor and extend the actions of the Windows operating system. This seminar is also appropriate for engineers actively involved in the areas of intelligence and information warfare or cyberwarfare.

Prerequisites

This is not a seminar for beginners without knowledge or experience in either operating systems or firmware. Rather, it is an intense, practical, architectural study of specific parts of the Windows O/S, interspersed with practical exercises.

Students attending this seminar will need a good understanding of general O/S concepts and will be well served by starting a basic understanding of Windows O/S architectural concepts. Because it is a hands-on seminar, students need a working knowledge of the C programming language, as well as the basic ability to use a Windows system.

Seminar Outline

1. Windows Operating System Architecture Overview

A brief review of the general architecture of the Windows operating system.

2. Kernel Mode and Key Structures

In this module, we discuss fundamentals of Kernel Mode in Windows, including coverage of key object types (Dispatcher, Control, Executive) and data structures (KPCR, KTHREAD, EPROCESS) that Windows uses.

3. Device Stacks – The Windows I/O Subsystem

A discussion of how the PnP Manager works with drivers to build stacks of devices through the enumeration process. The role of Function Drivers, Bus Drivers, and Filter Drivers is discussed. PDOs, FDOs, and Filter Device Objects are defined. The relationship of the PnP process to the process of loading "legacy" style drivers is discussed.

4. Installing Legacy Drivers on Windows

In this section we discuss how to create installation control files for "legacy" style, software-only drivers. Also included is a brief discussion of PnP (WDM) driver installation using INF files.

5. Building and Debugging

This section describes how WDM drivers are built using the WDK build environment as well as the basics of how to setup and use the Windows kernel mode debugger, WinDbg.

Lab: Setting Up the Debugger
Building Drivers

6. Interrupt Request Levels and Deferred Procedure Calls

Windows synchronizes kernel mode activity by using a set of Interrupt Request Levels (IRQLs). This section covers how IRQLs are used to achieve synchronization within the OS. Also the processing that occurs at these IRQLs - including Deferred Procedure Calls (DPCs) and dispatching - are discussed.

7. Passing Requests to Other Drivers

IRPs, I/O Stack Locations, and how requests are sent from driver to driver using IoCallDriver are all discussed. Synchronous and asynchronous IRP completion. A brief discussion of completion routines is also included.

8. I/O Function Codes and Buffer Methods

A specific discussion of I/O function codes (major and minor), as well as defining custom Device Control requests (IOCTLs). An detailed discussion of the way that Windows passes

buffers from user-mode applications to kernel-mode, including the security implications of each method.

9. DriverEntry -- Legacy vs. PnP

This section describes how software only drivers are initialized, and includes a walk-through and discussion of the actions taken in a typical DriverEntry function. The discussion is rounded-out with a description of the initialization functions used by PnP drivers (Add Device and Start Device).

10. Dispatching and Completing Requests

This module describes Dispatch Routines including I/O request validation, as well as the basics of request queuing and completion.

11. Serialization: Wait Locks and Spin Locks

A discussion of the various mechanisms used in Windows kernel-mode programming to perform synchronization and serialization. The different types of spin locks, mutexes, and other locks are discussed, along with guidelines for when each might be appropriate for use.

Lab: Handling DriverEntry and Dispatching Requests
Queuing, Buffering, Async I/O
Driver Communication

12. Tools for Driver Quality

In this section, we discuss tools that are available in the Windows Driver Kit to test and validate drivers. Windows Driver Verifier, OACR/PREfast, and Static Driver Verifier are all discussed, along with the strengths and weaknesses of each tool and recommendations for the best use of each.

13. Boot, Crash Dump, and Hibernation Processes

A discussion of how Windows starts, including transitions from 16-bit mode. Also, a discussion of the crash dump and hibernation processes (which are remarkably similar on Windows). Version-specific differences are also discussed.

14. Windows System Services

How system service calls are implemented in Windows, both in new versions of the OS and historically. A description of a selection of NtXxxx and ZwXxx functions, and how their use in user-mode and kernel mode differs.

15. Processes, Threads and Dispatching

In this section, we discuss the role of the Process Manager, including how processes and threads are instantiated. The major data structures (ETHREAD, EPROCESS) are described. The native system services for creating processes and threads are also briefly discussed. How dispatching (scheduling) is performed in Windows.

16. Other Kernel Extensions

An extended discussion of methods, other than those in the I/O domain, of extending the Windows operating system using software only drivers. This includes a discussion of Object Manager, Process Manager, and Registry callbacks, including the significant differences in the availability of these methods among various Windows versions.

Lab: Continue previous labs
Kernel Extensions

17. Cleanup, Close, and Cancel

Cleanup, close and request cancellation are compared and contrasted. When one might need to implement support for each in a typical WDM driver is discussed. Guidelines for supporting request cancellation, including Cancel Safe Queues and in-progress request handling are presented.

18. Windows Virtual Memory

A discussion of Windows virtual memory subsystem, including portions of the Memory Manager and Cache Manager. Page tables. Paging and page fault handling.

19. The Details of I/O Completion

In this section, we describe the details of how Windows completes I/O requests, as well as the correct handling of different types of I/O completion requirements within drivers. This includes a discussion of how the I/O Manager manages maintains synchronous request handling when requested by an application, even when a driver performs asynchronous I/O processing. Guidelines for proper driver implementation are developed and discussed.

Lab: Continue previous lab sessions
Cancel

Windows Kernel Debugging -- Security Edition

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.

Only by analysis of Windows data structures can one determine the cause of the problem. While a solid background in Windows internals and data structures is mandatory for success in crash dump analysis, it is not sufficient. Rather, a rigorous, methodical approach is required to trace the crash from its immediate cause to its root reason. 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.

This is a hands-on class intended to give students real, practical experience in using the Windows kernel debugger (WinDBG), as well as including demonstrations of (and optional labs using) Hex-Ray's IDA Pro debugger/disassembler.

Seminar Format

This seminar is a 5 day lecture, intermixed with practical lab exercises. The seminar 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

Security researchers, analysts, and software developers who need to understand how to set up for, use, and analyze OS crash dumps. Supporting personnel who need to know how to debug Windows system crashes in the lab or at customer sites, in situ or post mortem.

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 have a solid command of Windows operating system internals and data structures.

Seminar Outline

Note for all labs: 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

1) 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.

2) 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.

3) Introduction to IDA Pro

IDA Pro is a powerful disassembler and debugger, created by Hex-Rays SA. Support for the Windows debugging interface has recently been added to this excellent tool. 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. The relative advantages and disadvantages of WinDbg and IDA Pro are also discussed.

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

4) Examining System Tables and Structures

In this section, we discuss how to use the skills of using WinDBG gained in the first lab session to examine both live and post-mortem systems, and analyze what's happening. What processes and threads are running? What kernel-mode code modules are loaded? We also discuss techniques for identifying when a system has been compromised.

Second Lab: Students examine key Windows data structures and construct scenarios that describe the activity on the system. In addition, students search for traces of malware as implemented by several well-known "root-kit" attacks.

5) 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.

Third Lab: Using information collected from the previous labs, 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.

6) Overview of 64-bit Processors

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.

Fourth 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.

7) 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.

Fifth 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.

8) 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.

Sixth 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.

9) 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.

Seventh 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.

10) Moving Beyond the Debugger

Discussion of kernel debugger extension libraries, how they are constructed, and how they are used by WinDbg.

Seventh 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.

Windows Source Code Internals

Overview

You've read the books and digested the concepts. Now you have to know how things really work. This seminar takes you on a guided tour of carefully selected source code modules from NTOS. From dispatching a system service to retiring the DPC list, you'll be introduced to the vital portions of the Windows source code base. Points of emphasis for discussion in class may be selected in advance by the client.

The Windows source code base is very large and can be imposing, even for experienced developers. This seminar can be highly effective if you have team members who need to come up to speed quickly on the layout, data structures, and key algorithms in the Windows operating systems at a source code level. This seminar may be taught as a multi-day stand-alone seminar, or may be combined with our Writing WDM Kernel Mode Drivers for Windows seminar.

Please Note: Before this seminar can be scheduled, OSR will verify directly with Microsoft that you have a current, valid, Windows source code license.

Target Audience

This class is intended for Microsoft Source Code Licensees who are experienced developers. Attendees seek to understand the most important aspects of the Windows operating system implementation at the details source code level.

Prerequisites

Because this seminar is only delivered as a custom seminar, the level of prerequisite knowledge entirely depends on the content chosen. OSR has successfully taught this seminar to support teams with basic understanding of the C language and solid knowledge of Windows architecture, as well as to developers actively working on Windows kernel-mode code. Let our seminar consultants work with you to craft a seminar that will specifically fit your team's needs and abilities.

Seminar Outline

This is a completely customized seminar. The following outline is an example that provides more than three days of jam-packed material. However, when you schedule this seminar, you choose the topics that you want to cover, based on your learning goals and your team's level of experience.

1) Kernel and Executive Overview

The Object Manager Kernel Dispatcher and Control objects; Creating objects with the

Object Manager; Executive functions; Integral subsystems; System services and objects exported by each, source code modules for each.

2) Source Kit Layout

How the source kit is structured; What's in which directory; How service packs are distributed.

3) Processing System Services

Initializing system service tables; Invoking system services; Interrupt, trap, and syscall handling conventions; Processing a simple system service; Processing an I/O related system service.

4) Interrupt Handling and DPCs

Connecting to interrupts; Handling device interrupts and DPCs; Servicing DISPATCH_LEVEL interrupts; Processing the DPC list.

5) Threads and Processes System Startup and Shutdown

Key data structures: EPROCESS, KPROCESS, PEB, KTHREAD, ETHREAD, TEB; Process creation; Process deletion; Thread creation; Thread deletion; System process creation; System process deletion.

6) Dispatching and Scheduling

APCs and DPCs Scheduling algorithm; Data structures; Uniprocessor scheduling; Multiprocessor issues; APC Management; Context switching; DPC queue management.

7) Memory Management

Page directories, page tables, page frames; Page fault handling; Working set management; Shared memory; Paging files; Caching.