Windows Kernel Programming Master Class

Course Summary Table

Duration:	6 Days (48 hours)
Target Audience:	Experienced windows developers, interested in developing kernel mode drivers
Objectives:	• Understand the Windows kernel driver programming model • Write drivers for monitoring processes, threads, registry and some types of objects • Write file system mini-filter drivers • Write Windows Filtering Platform Callouts
Pre-Requisites:	• At least one year of experience working with the Windows API (user mode) • Basic understanding of Windows OS concepts such as processes, threads, virtual memory and DLLs
Software requirements:	• Windows 10 or 11 64 bit (any SKU, latest stable version) • Visual Studio 2022 (any SKU) + latest update • Windows 11 SDK (latest) • Windows 11 WDK (latest) • Virtual Machine for testing and debugging

Instructor: **Pavel Yosifovich**

Abstract

The cyber security industry has grown considerably in recent years, with more sophisticated attacks and consequently more defenders. To have a fighting chance against these kinds of attacks, kernel mode drivers must be employed, where nothing (at least nothing from user mode) can escape their eyes.
The course provides the foundations for the most common software device drivers that are useful not just in cyber security, but also other scenarios, where monitoring and sometimes prevention of operations is required. Participants will write real device drivers with useful features that can then be modified and adapted to their particular needs.
The course includes tips and techniques employed by the instructor in their own projects, based on years of experience.

Syllabus

- Module 1: Windows Internals quick overview

    - Processes
    - Virtual memory
    - Threads
    - System architecture
    - User / kernel transitions
    - Kernel Design
    - Introduction to WinDbg
    - Windows APIs
    - Objects and handles
    - Summary

- Module 2: The I/O System
    - I/O System overview
    - Device Drivers
    - Driver Models
    - Driver types
    - Software drivers
    - Driver and device objects
    - Looking at Existing Drivers
    - I/O Processing and Data Flow
    - Accessing devices
    - Asynchronous I/O
    - Summary

- Module 3: Device Drivers Basics
    - Setting up for Kernel Development
    - Basic Kernel types and conventions
    - C++ in a kernel driver
    - Creating a driver project
    - The kernel API
    - Strings
    - Linked Lists
    - Object Attributes
    - The *DriverEntry* function
    - The *Unload* routine
    - Installation
    - Testing
    - Debugging
    - Summary
    - Lab: write and deploy a simple driver; debug a driver

- Module 4: The I/O Request Packet
    - Creating a device object
    - Exporting a device name
    - Building a driver client
    - Driver dispatch routines
    - Introduction to I/O Request Packets (IRPs)
    - Completing IRPs

    - Accessing User Buffers
    - Handling *DeviceIoControl* calls
    - Handling Asynchronous Operations
    - Summary
    - Lab: access any process; use Direct I/O

- Module 5: Kernel mechanisms
    - Interrupt Request Levels (IRQLs)
    - Deferred Procedure Calls (DPCs)
    - Dispatcher objects
    - Low IRQL Synchronization
    - Spin locks
    - Driver-Created Threads
    - Work items
    - Timers
    - Summary

- Module 6: Programming Techniques
    - Attaching to Processes Address Space
    - Object to Handle
    - Handle to Object
    - Handle Duplication
    - C++ RAII
    - Advanced Memory Management
    - Using Linked Lists
    - Strings

- Module 7: Process and thread monitoring
    - Motivation
    - Process creation/destruction callback
    - Specifying process creation status
    - Thread creation/destruction callback
    - Notifying user mode
    - Writing a user mode client
    - Preventing potentially malicious processes from executing
    - Summary
    - Lab: ProcMon-like process/thread operation monitoring

- Module 8: Object and Registry notifications
    - Process/thread object notifications
    - Pre and post callbacks
    - Registry notifications
    - Performance considerations
    - Reporting results to user mode
    - Summary

- Module 9: File system mini filters
    - File system model
    - Filters vs. mini filters
    - The Filter Manager
    - Filter registration
    - Pre and Post callbacks
    - File name information
    - Contexts
    - File system operations
    - Filter to user mode communication
    - Debugging mini-filters
    - Lab: preventing certain file deletion
    - Summary

- Module 10: Windows Filtering Platform
    - WFP Architecture
    - Layers, Filters, and Callouts
    - WFP API
    - WFP Management
    - Callout Drivers
    - Putting it all together
    - Summary

- Module 11: Programming Techniques II
    - Using Native APIs
    - Trace Logging
    - Hooking Drivers
    - Plug & Play
    - IRP Propagation
    - Writing Generic Filter Drivers
    - Completion Routines
    - Driver Verifier
    - Summary

- Module 12: Introduction to KMDF
    - Why KMDF?
    - KMDF Object Model
    - KMDF vs. WDM
    - Fundamental Objects
    - DriverEntry and AddDevice
    - I/O Request Flow
    - INF Files
    - Writing a Client
    - Summary