CENTER FOR CYBER SECURITY TRAINING

Windows Internal Architecture

Abstract

The Windows PC continues to be the primary productivity device in enterprises small and large alike. Due to its ubiquity, the Windows desktop remains the favorite target for attackers to gain initial access into an organization, move laterally, and maintain their foothold. Whether you analyze malware, perform security research, conduct forensic investigations, engage in adversary simulation or prevent it, or build security solutions for Windows, understanding how Windows works internally is critical to be effective at your task.

This unique course takes you through a journey of Windows internals as it applies to user-mode execution i.e. applications and services. Everything is examined through the lens of security both from an offense and defense perspective.

For each topic that is covered, components, architecture, data structures, debugger commands, and APIs are discussed with the hands-on labs helping with observing things in action and thus solidifying the understanding of the topic.

This training course focuses on security-related topics and **does not cover** topics related to Win32 application development.

Learning Objectives

Students will:

- Understand the key principles behind the design and implementation of the Windows operating system.
- Understand the components in the Windows operating system and the functionality they provide.

- Understand the functionality provided by Windows that make applications and services tick.
- Understand the facilities in the system that are commonly abused by malware.
- Understand the security mitigations available in Windows that raise the bar against exploits and malware.
- Be able to investigate system data structures using the debugger and interpret the output of debugger commands.
- Be able to navigate between different data structures using the debugger.
- Be more effective at analyzing malware on Windows systems.
- Be more effective at forensic analysis of Windows systems.

Target Audience

Security researchers, malware analysts, threat hunters, incident responders, digital forensics investigators, red-teamers, blue-teamers, and security software developers.

Requirements

Students will need:

Hardware

- x64 CPU with support for hardware virtualization
- Dual monitors (for instructor's screen share and hands-on labs)
- Reliable and fast Internet connection
- Working audio system (speaker and mic)

Software

- Windows 10 64-bit
- Visual Studio 2019 (Community+), SDK, WDK 20H2 **OR** Enterprise WDK 20H2 (EWDK)
- WinDBG Preview
- Virtualization Software [Hyper-V built into the Windows 10 PRO and ENT editions]
- GotoTraining native Windows client
- All other software and tools will be provided before the training.

Students' Knowledge Pre-Requisites:

Attendees must have a solid understanding of operating system concepts and have a working knowledge of Windows. This course does not require any programming knowledge.

Course Outline:

Topics

- System Architecture
- User Mode Execution
- Memory Management
- PE Files
- Objects and Handles
- Security
- Services Infrastructure
- Security Mitigations

System Architecture

The objective of this section is to learn about the architecture of the modern Windows platform with topics such as user-mode and kernel-mode execution, user and kernel components, process and system address space, functionality provided by NTDLL, call flow from Win32 applications to the kernel, WinDBG and symbols, differences between system and process memory dumps and their contents, Hyper-V, VBS, virtual trust levels (VTLs) and trustlets.

- Execution rings
- System architecture
- NTDLL functionality
- Win32 and Native APIs
- User-mode and kernel-mode debuggers
- Process and system memory dumps
- Virtualization based security (VBS)
- Secure kernel and trustlets

User Mode Execution

The objective of this section to learn about how the details of user-mode code execution. It covers topics such as processes, user-mode threads, system process, kernel-mode threads, process and thread data structures (PEB & TEB), system processes, user-mode startup sequence, signing levels and access restrictions of protected processes, thread register contexts, user-mode APCs and jobs.

- Processes
- Threads
- System process
- Process and thread data structures
- Process hierarchy
- Host processes
- Execution vectors
- Jobs

Memory Management

The objective of this section is to understand the process virtual address space and its components. It covers topics such as types of ALSR, page protection flags, no access pages, guard pages, execute and no-execute pages, virtual address descriptors, usage of VAD in memory forensics, remote process attachment, reading and writing remote process memory, section objects, shared memory and memory-mapped files, executable image mapping, pagefile backed regions, thread stacks, stack cookies and process heaps.

- Process address space
- Address space layout randomization (ASLR)
- Memory protection
- Data execution prevention (DEP)
- Virtual address descriptors (VADs)
- Cross process memory access
- Memory-mapped files
- Thread stacks

- Heaps

PE Files

The objective of this section to discuss PE files and how they are loaded into memory. It covers topics such as PE files structures, navigating PE headers, sections and permissions, .NET executable, parsing export directory, importing functions and import address table, 64-bit relocation table, NTDLL loader, PEB module list, thread local storage (TLS), TLS callbacks, kernel image notification callbacks.

- PE files layout
- Data directories
- Debug directory
- CLR Header
- Export directory
- Import directory
- Relocations
- Loader functionality
- DLL load callbacks

Objects and Handles

The objective of this section is to understand objects and handle tables. It covers topics such as kernel objects, GDI objects, user objects, global and session-specific namespace, symbolic links, object headers, optional object headers, handle tables, granted access mask, handle creation and duplication, kernel object callbacks, object reference counting, types objects and object type procedures.

- Windows Objects
- Object namespace
- Object headers
- Process handle tables
- Handle duplication
- Object callbacks

- Object reference counting
- Type objects

Security

The objective of this section to learn about how the kernel secures access to objects. It covers topics such as tokens, SIDs, well-known SIDs, privileges, restricted tokens, user account control (UAC), security descriptors, discretionary access control lists (DACLs), system access control lists (SACLs), access control entries (ACEs), access masks, ACE inheritance, mandatory integrity control, mandatory policy, ACE evaluation, impersonation tokens and impersonation levels.

- Tokens
- Security identifiers (SID)
- Privileges
- Security descriptors
- Access control lists
- Integrity levels
- Access checks
- Impersonation

Services Infrastructure

The objective of this section is to understand how services work in Windows. It covers topics such as service control applications, service control manager (SCM), types of services, service dependencies, executable services, DLL based services, trusted installer service, logon architecture, logon sessions, window stations, desktops, shatter attacks, session 0, user interface privilege isolation (UIPI).

- Service architecture
- Service control manager
- Service configuration
- SVCHost.exe
- Service SIDs
- Trusted Installer

- Logon sessions
- Session isolation

Security Mitigations

The objective of this section is to understand the different security mitigations that have been added to protect applications over the years. It covers topics such as process/thread attributes, process mitigations policies, exploit guard functionality, code integrity guard, control flow guard, enhanced control flow guard, shadow stacks, low-box tokens, app-container namespace, filtering native API calls into Win32K.sys.

- Process mitigation policies
- Exploit guard
- Arbitrary code guard (ACG)
- Control flow guard (CFG)
- Enhanced control flow guard (xFG)
- Control-flow enforcement technology (CET)
- Win32K API call filtering