

Syllabus: Windows Native API Programming

The Windows native API is the “real” API user-mode code used to talk to the kernel. This API, however, is mostly undocumented. The course teaches the fundamentals of the Native API implemented by NtDll.dll, as it relates to system calls, but not just system calls.

About TrainSec Academy

TrainSec is an online learning hub for current and aspiring cybersecurity pros who understand the need for excellence. Level up or change career using skill-expanding modular learning way. We turn beginners into experts and experts into masters. TrainSec is designed for those who are serious about being at the vanguard of the latest knowledge, skills and trends. We’re already working with the best.

Course Objectives

- NT architecture and the system-call dispatch path
- Controlling processes, threads, and their hidden PEB/TEB structures
- Navigating the Object Manager: handles, types, and objects
- Mastering virtual memory: allocation, protection, and section mapping
- File and device I/O leveraging native powers
- Working with the Registry including notifications and transactions
- Thread synchronization primitives

Pre-requisites

- C/C++ programming experience on Windows
- Familiarity with Visual Studio 2022, and the basics of the Windows API
- Basic understanding of processes, threads, and virtual memory concepts
- Comfort with command-line tools and reading technical documentation

Course Format

- **URL:** <https://trainsec.net/courses/windows-native-api-programming/>
- **Scope and Mode:** 90 lessons, spanning over 15+ hours of online videos and presentations.
- **Format:** Pre-recorded videos, presentations, interactive labs, discussion groups, and personal trainer support.

- **Pace:** Self-paced.
- **Platform:** TrainSec.net Learning Platform and Discord server.

Course Overview

Windows exposes two faces to developers in user mode: the familiar Win32 (Windows API) layer and a far leaner, more powerful interface the operating system itself relies on – the Native API. In this course, Pavel Yosifovich guides you past the Win32 façade and straight to that inner doorway. You begin with the NT architecture and the system-call path, then master the data types, processes, threads, objects, and memory structures that everything in Windows is built on. Each subsequent module peels back another layer: spawning true Native executables, leveraging the object manager’s namespace, tracing handles, using virtual memory, streaming data through the I/O manager, and editing the Registry atomically. Short labs turn every concept into working code in Visual Studio using open-source headers, so by the final lesson you can create, probe, and control kernel-backed resource from user mode.

Module 1: Windows System Architecture & Native API Fundamentals

Begin your native-level adventure with a clear map of the territory. First we dissect the Windows NT architecture-user mode, kernel mode, and the all-important NTDLL gateway that ties them together. Next comes a quick tour of the historical subsystems (POSIX, OS/2, Win32) to see how each layers is its own personality over the same kernel. With that context we crack open the undocumented Native API;, how NTDLL funnels higher-level APIs into the kernel, , and what you can (and can’t) do that Win32 hides. We wrap up in Visual Studio: adding the open-source PHNT headers for instant prototypes and experimenting.

Module 2: Native Data-Types

The Native API speaks its own dialect, and this module hands you the phrasebook. Pavel demystifies the core data structures-NTSTATUS, Unicode strings, OBJECT_ATTRIBUTES, linked-list nodes, CLIENT_ID, and time types-and then drills you through a lab where you put each one to work. By the end of the module, you’ll know the helper macros and functions that tame them, and be ready to pass well-formed parameters into any Nt* function that appears in later modules.

Module 3: Building & Launching Native Applications

Now that you know the API’s landscape, it’s time to compile real executables that live there. This module shows how Windows treats “Native” and “Boot” applications,

then guides you through creating, linking, and finally spawning a Native EXE entirely from your own code.

Module 4: System Information

Windows hides a vast amount of live data just below Win32. In this module, you learn to pull back the curtain. Starting with `NtQuerySystemInformation`, you'll enumerate processes, threads, and even every open handle on the system. Along the way Pavel demystifies the Object Manager-its types, names, and namespace hierarchy. The module wraps with a look at `KUSER_SHARED_DATA`, the always-mapped page that gives you fast, syscall-free access to clocks, CPU info, and more.

Module 5: Processes

Processes are the backbone of Windows. This module starts with crafting a process from raw pieces-image, initial thread, parameters-before moving on to deep inspection via `NtQueryInformationProcess` and direct reads of the Process Environment Block. You'll learn to suspend, resume, terminate, and tweak scheduling priority without ever touching Win32. By

Module 6: Threads & Concurrency

Processes may own resources, but threads get work done. In this module, you craft threads from scratch with `NtCreateThreadEx`, peek into their stacks and TEBs, and gather live stats via `NtQueryInformationThread`. Pavel then unpacks the Native APIs for keeping threads in line: waits, APCs, and low-level synchronization, culminating in thread-pool discussion that showcases practical concurrency patterns.

Module 7: Objects and Handles

Windows unifies named kernel objects through the Object Manager, and this module is your guided tour. You'll map the kernel namespace, enumerate every object it contains, and learn to create, link, duplicate, and query them using pure Native calls. With a solid grasp of directory and symbolic-link objects, you pivot to the synchronization primitives-mutexes, semaphores, events-and finish by managing whole process groups with job objects.

Module 8: Virtual Memory

Memory is everywhere. In this module, Pavel walks you from the hardware-backed basics up through high-level heaps and memory-mapped files. You'll allocate, protect, query, and even patch memory using pure Native calls, then peek into the

kernel's view of working sets and protection bits. The journey continues with custom heaps for performance-critical tasks and finishes with section objects—the backbone of DLLs and shared memory.

Module 9: Device and File I/O

Whether you're reading from disk, pinging a driver, or ringing the PC speaker, it all rides the same I/O manager. This module teaches you to open any path, translate it into the kernel's namespace, and push bytes through `NtReadFile`, `NtWriteFile`, or custom IOCTLs. You'll enumerate directories, grab metadata, harness I/O completion ports for efficiency, and even load a driver—all with pure Native calls.

Module 10: Registry

The Registry is Windows' configuration database, and the Native API lets you operate on it with surgical precision. In this module, Pavel demystifies its hive-backed architecture, then leads you through opening, creating, editing, and enumerating keys and values without a single Win32 call. You'll set up change notifications for instant event-driven reactions and wrap multiple edits inside kernel-level transactions for all-or-nothing safety.

Module 11: Security

Every access to a kernel object involves a user/group (SID) opening a handle to a relevant object that may be protected by a Security Descriptor. The user's "power" is represented by its Access Token, containing its privileges and other properties. This module connects those dots. You'll decode SIDs, inspect and duplicate tokens, enumerate logon sessions, and even create tokens from scratch. Finally, you'll craft security descriptors and ACLs, giving you full control over who can do what with your objects—essential knowledge for access management.

Instructor Information

- **Name:** Pavel Yosifovich
- **Title:** Windows Internals Expert
- **Bio:** Software developer, trainer, and author. Co-author of "Windows Internals" 7th edition (2017), and author of "Windows Kernel Programming, 2nd ed" (2023).
- **Social Handle:** @zodiacon

Contact & Support

For any course-related or billing-related inquiries, please contact info@trainsec.net