Module Structure

AV & EDR HOOKING EVASION

INTERMEDIATE | EDR EVASION | 0X12 DARK DEV

THEORY

00 Introduction

Scope, ethical context, and userland hooking as the primary EDR detection layer.

01 What is Function Hooking?

Hook layers, ntdll.dll patching, EDR DLL injection via kernel callbacks, and the in-process weakness.

02 How Does Function Hooking Look Like?

Identifying hooked stubs byte-by-byte — clean vs patched prologue, JMP trampoline anatomy.

TECHNIQUES

T1 Direct Syscalls

Bypassing ntdll.dll entirely — writing inline ASM stubs that fire the syscall instruction directly.

T2 Dynamic Resolving Syscall Codes

Hell's Gate: reading SSNs from live ntdll stubs at runtime. SysWhispers4 integration for MSVC.

T3 Fresh Copy

Remapping a clean ntdll.dll from disk and overwriting the hooked .text section in memory.

T4 Perun Farts

Fileless variant — reading a clean ntdll from a suspended process and restoring only the syscall stub region.

T5 Unhook Detected Hooks

Scanning the live export table for patched stubs and restoring only the functions the EDR has touched.

T6 Unhook Desired Hooks

Surgical restoration — specifying exactly which NT functions to unhook with zero detection phase.

→ Exercise

Challenge 1: Hell's Gate + Halo's Gate injector with hooked-stub SSN inference. Challenge 2: selective unhooker with ETW-TI & kernel callback awareness.

■ Conclusions

Userland hooks as the weakest detection layer — and why kernel callbacks and ETW-TI remain untouched.