Windows Malware Techniques

Abstract

User mode malware on Windows is ubiquitous and custom user mode implants are used regularly in red-team engagements. Knowledge of the latest malware techniques helps red teamers improve their custom tooling, malware analysts in taking apart malware, and anti-malware solution developers in designing behavioral solutions to detect malicious activity.

The common theme amongst all Windows malware and implants is that they abuse the facilities provided by the Windows platform to achieve their objectives. Knowledge of the rich set of Windows APIs, understanding their usage in various stages of an implant, and leveraging them to detect and bypass various defenses in the system is essential for red and blue teamers.

This training course takes attendees through a practical journey with a hands-on approach to teach them about the post-exploitation techniques used by PE file-based implants at every stage of their execution.

Beneficial to both the offensive and the defensive side of the camp, the knowledge and hands-on experience gained in this training will help attendees with real-world red teaming engagements and in defending against both custom advanced persistent threat (APT) tooling and common-off-the-shelf (COTS) malware. Attendees will learn about how malware and implants interact with the latest version of Windows and how the different stages of malware abuse and exploit various components of Windows OS to achieve their goals and evade defenses.

Learning Objectives

Students will:

- Build custom tooling for offensive operations.
- Build position independent shellcode using C/C++.
- Perform basic tasks required by user-mode implants.
- Inject and execute shellcode and DLLs in code in privileged processes.
- Perform code flow subversion through hooking and subvert anti-malware hooks.
- Beacon out and receive tasking from a C2 infrastructure.
- Exfiltrate data using protocol tunneling.
- Implement persistence and auto-execution to survive system reboots.
- Detect and evade various defensive mechanisms in the system.

Target Audience

Security researchers, malware analysts, 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 Windows internals and familiarity with user-mode development on Windows using Win32 APIs. This is a developer-oriented course and attendees are expected to have prior experience with C/C++ programming on Windows 10.

Course Outline:

Topics

- Shellcoding
- System Interfaces
- Code Injection
- Hooking
- Persistence
- Communications
- Self-Defense

Introduction

The objective of this section is to introduce attendees to the Windows malware landscape and discuss the malware ecosystem. It covers topics such as the different stages of malware execution, common vectors through which stage zero establishes a beachhead on the system, differences between staged and stageless malware, and logging mechanisms that can be enabled to detect various execution stages.

- Offense and defense
- Platform mitigations
- Attack execution stages
- Initial access methods
- Staging payloads
- System logging
- Ecosystem review

Shellcoding

The objective of this section is to learn about developing shellcode using raw x64 assembler and high-level languages. It covers topics such as shellcoding tools (MASM/NASM/YASM), x64 assembler limitations, methods for generating position-independent and self-contained functions, shellcode injection and execution techniques, sharing data assembler and C/C++, building trampolines to interface between x64 assembler and C/C++ and per-module/per-function level runtime dependencies.

- Shellcoding tools
- Shellcode injection
- Position independent code
- Trampolines
- Compiler and linker flags
- Runtime checks & dependencies

System Interfaces

The objective of this section is to learn about the internal interfaces and mechanisms used by the system to support PE file execution and how these are exploited by malware. It covers topics such as the PEB, TEB, compiler tricks to access low-level interfaces, NTDLL loader data structures, APIs for extracting information from PE files, looking up imported functions by checksums, structured and vectored exception handlers, and exception handling in shellcode.

- Module lists
- Compiler intrinsics
- PE parsing
- Import hashing
- Structured exception handling
- Dynamic exception handlers

Code Injection

The objective of this section is to learn about different code injection and execution techniques. It covers topics such as various methods of injecting shellcode and PE files into remote processes, using various system execution vectors to run the injected code, implementing a custom PE loader, mapping and unmapping PE files, challenges in injecting and executing

shellcode code into WoW64 processes. Commonly used process injection techniques and their variants are also covered.

- Injection & execution
- Process injection techniques
- Classic DLL injection
- Reflective injection
- Process hollowing
- WoW64 process injection

Hooking

The objective of this section is to learn about various code flow subversion techniques in EXEs and DLLs. It covers topics such as prolog and epilog hooking, evading scanners with code caves, injecting shellcode in unsigned PE files, import address table hooking, window local and global hooks, using RunDLL32 to host hook DLLs, and detecting and circumventing user-mode hooks.

- Inline hooking
- Code caves
- Binary trojaning
- Import hooking
- Windows hooks
- Hook subversion

Persistence

The objective of this section is to learn about various user-mode persistence and auto-execution vectors. It covers topics such as registry-based auto-start execution points (ASEPs), persistence using native boot executables, image file execution options (IFEO), DLL search order hijacking, DLL shimming, missing COM object hijacking, COM object search order hijacking, persistence through executable and DLL based services.

- Registry ASEPs
- System execution vectors
- DLL hijacks

- DLL proxies
- COM object hijacks
- Service hijacks

Communications

The objective of this section is to learn about malware's communication with and its command and control (C2) servers. It covers topics such as WinINET APIs, using proxy aggregators and open web proxies to hide listening posts (LP), hosting C2 infrastructure, using whitelisted protocols for sending beacons and receiving tasking orders, data compression, chunking and encoding for exfiltration.

- Network enumeration
- HTTP proxies
- C2 infrastructure
- Beacons and tasking
- Protocol tunneling
- DNS data exfiltration

Self-Defense

The objective of this section is to learn about the most common detection and protection mechanisms in the latest version of Windows 10 and how malware can circumvent them. It covers topics such as detecting hostile environments, event log entries that identify malicious activity in the system, detecting endpoint security products, and common techniques for evading behavioral detection.

- Environment detection
- Debugger detection
- VM detection
- Event logging bypass
- Security product detection
- Evasion techniques