MAL600

ADVANCED MALWARE ANALYSIS

MAL600 - Advanced Malware Analysis is an advanced course that exposes students to the theoretical knowledge and hands-on techniques to reverse engineer malware designed to thwart common reverse engineering techniques.

Students will learn how to identify and analyze the presence of advanced packers, polymorphic malware, encrypted malware, and malicious code that has been armored with cryptors, anti-debugging and anti-reverse engineering.

TARGET AUDIENCE

Mid-level malware analysts & reverse engineers, as well as programmers who want a different professional perspective as a means of better protecting their tools & intellectual property

OBJECTIVE

Provide an in-depth understanding of identifying & analyzing the presence of advanced packers, polymorphic malware, encrypted malware & malicious code

CYBRScore™

DAY 1

The course begins by examining a variety of network signatures associated with malware. Understanding the networking aspect is important because malware almost always uses network connectivity to infect, persist, receive command and control instructions, and exfiltrate data.

Students are asked to spend a significant amount of time reversing malicious command and control structure parsing routines to better understand the overall network activity, and how to identify and stop it.

Topics List

» Indications of malware activity
» Network countermeasures
» Snort & complex signatures
» Hiding in the noise by mimicking existing protocols
» Client initiated beacons
» Networking code & encoding data
» Networking from an attacker's perspective

DAY 2

Day 2 focuses on anti-disassembly techniques employed by malware authors to thwart analysis. Students learn about various techniques, like jump instructions with the same target, jump instructions with a constant condition and more. More complex techniques like return pointer abuse and misusing structured exception handlers give the student new conceptual knowledge.

This knowledge will help complete three complex hands-on challenges: identifying false conditional branches, improperly disassembled code, and return pointer abuse.

Topics List

» Defeating disassembly algorithms
» Same target jumps & constant condition jumps
» Rogue opcodes
» Multi-level inward jumping sequences
» Patching binaries to defeat return pointer abuse
» SEH abuse
» Reversing armored code designed to thwart stack frame analysis

DAY 3

Anti-debugging is used by malware authors to determine when their malware is under the control of a debugger or to thwart debugging efforts. On Day 3 students learn how Windows API can be used to detect debugger use, and how malware manually checks structs. Checking the ProcessHeap and NTGlobal flags is reviewed, as well as how some malware checks the analysis system for debugging tool residue in the registry.

The module concludes with a discussion of TLS callbacks, and exceptions to disrupt debugger use.

Topics List

» Using Windows API functions to detect debuggers
» PEB checks, ProcessHeap flag & NTGlobal flag
» TLS Callbacks
» Exceptions and Interrupts
» PE Header vulnerabilities
» OutputDebugString vulnerability

DAY 4

Although the presence of anti-virtual machine techniques seems to be declining, Day 4 is spent discussing how to identify various methods used by malware authors.

Students also learn how to manually unpack malware by finding tail jumps, the original entry point (OEP) and rebuilding Import Address Tables (IAT).

Topics List

» Anti-VM techniques & memory artifacts
» Red pill & no pill techniques
» Unpacking stub, tail jump, OEP & import resolution
» Manual IAT rebuilds
» Tips & tricks for dealing with several common packers

DAY 5

On the final day of class, students learn how to identify and reverse C++ code, in addition to conducting shellcode analysis. Virtual functions and the concept of polymorphism are discussed to prepare students to identify and reverse vtables using their cross references.

Position-independent shellcode is examined, as well as how to identify execution location. Day 5 ends with a look at 64-bit malware and the challenges analysts face when reversing this type of code.

Topics List

» Shellcode analysis, position independent-code & call/pop
» Shellcode use of LoadLibraryA & GetProcAddress for dynamic function location
» C++ Analysis
» Overloading functions, mangling and vtables
» Challenges of identifying inheritance between classes
» 64-bit malware, general-purpose & special-purpose registers
» X64 calling convention & exception handling

MAL600

ADVANCED MALWARE ANALYSIS

CYBRScore™