



Reverse Engineering

Abstract

This course is intended for anyone just starting on their reverse engineering journey. It covers all of the basics required for more specialized applications of reversing like malware analysis, vulnerability research, and program analysis. From processors and assembly languages, to memory addressing and layout, to compilers and common language features, this class starts from ground zero and covers everything you need to know.

Prerequisites

Familiarity with Python and C would be helpful, but not required. You must understand variables, if/while/for/etc logic, numbers in different bases, truth tables, and similar fundamental computing concepts.

Learning Objectives

Students will understand:

- What a 'program' really is
- How programs work / how computers run programs
- How programs store, manipulate, and interpret data
- The reversing process
- Being able to reason about the behavior of a program

Target Audience

- Reverse Engineers
- Vulnerability Researchers
- Malware Analysts

- CTF Players
- Beginners

Requirements

Students will need:

A laptop that can run Binary Ninja (Ubuntu 20.04/22.04 x64; Windows 10/11 x64; MacOSX 11+ x64, MacOSX 12+ arm64).

(Optional) A VM to run binaries (most binaries are also provided for Linux, Mac, and Windows, though we often focus on just the Linux versions)

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:

At the end of each topic during the class, students get to vote on which topic they'd like to explore next. This allows us to accommodate a wide array of student backgrounds by spending more time on the things you need to know, and less time on the things you're already a pro at.

- How to reverse engineer (reading source code)
- Advanced program theory
- How computers work / why we need those 1's and 0's (compiler theory)
- Basic processor features (registers, memory, instructions)
- Assembly* (see below)
- How to reverse engineer (reading assembly)
- Why reading assembly sucks
- Decompilation theory
- How to read Binary Ninja's ILs
- Simple crackmes
- Binary annotation / interaction
- Structuring data

- Finding data and references
- Binary patching / transforms / unpacking
- Dataflow analysis
- Source-to-sink analysis in Binary Ninja
- Using the debugger
- And more!

***This course will cover the following assembly concepts:**

- Registers
- Opcodes and operands
- Operand order
- Specifics about the most common opcodes
- Delay Slots
- Branching/Flags
- Memory Addressing
- Stack/Heap
- etc

You'll learn the basic skills every reverse engineer should know about reading assembly (x86, x86-64, ARM, MIPS, etc), but also how to circumvent the need to do so as much as possible. Instead, this class aims to make you fluent in the high-level abstractions provided by modern decompilers and analysis frameworks, their associated interactive features, and how to achieve an understanding of a program.