x86/x86-x64 Assembly, Software Reverse-Engineering and Analysis

WORKSHOP OVERVIEW

This professional deep technical workshop offers a systematical foundation for low-level security software reverse-engineering intel architecture x86/x86-x64 and software analysts, taught from an offensive security research perspective. This class helps you bootstrap into the areas of reverse engineering, vulnerability exploitation, operating system design, code optimization, and compiler design. Key objectives: Software, Intel Architecture, Reverse-Engineering, and Analysts.

Once you've taken this class, it will open the door to all the other specialty areas that depend on assembly knowledge. Although Intel® 64 and IA-32 Architectures Software Developer's Manual have 5060 pages+, students will develop the ability to understand the core concepts behind most programs by learning common programming instructions and their variations.

This workshop expose students to all the essential theory required to analyze software for defensive and offensive, respectively. as well as fundamental practical skills. 40% of the time will be spent bootstrapping knowledge of fully OS-independent aspects of Intel architecture. 60% will be spent learning Reverse-Engineering and analysis of malicious programs.

All labs in this workshop are based on real life tasks.

AUDIENCE

- Professional security researchers, malware reverse engineering and reverse engineers.
- People interested in reverse engineering, malware analysis, vulnerability research, exploits, and mitigations.
- Developers who want to understand the correspondence between high level code and machine code.
- People who want to better understand the low level hardware mechanisms which support binary program execution and operating system design.

PREREQUISITES

Essential:

- Must have familiarity with basic programming.
- Assembly/C/C++ (reading level);
- Linux command line and build environment.

Recommended:

- x86/x86_64 assembly language;
- some experience with software analysts.
- basic OS and hardware design concepts.

Hardware:

Students should bring a laptop capable of running a 64 bit version of Windows as specified below, with at least 8GB of RAM, so it can comfortably run multiple virtual machines.

ABOUT THE INSTRUCTOR

Fatah Hashim is a software security researcher specializing in offensive and defensive security research, analysis, and development. His professional career involves low-level programming, reverse engineering, antivirus operation, evasion, malware, vulnerability, and windows internals. His recent work includes research about antivirus research engine bypass: static/dynamic evasion.

Having devoted most of his life to the researching of software security, Throughout his teen years he alternated learning between different programming languages, C, C++, Assembly, Powershell, VB, and Python. His ultimate goal is to understand how computers work from the highest of high levels to the lowest of low levels.

LEARNING OBJECTIVES

Upon completion of this workshop class the students are expected to have obtained the following knowledge and skills:

- Ability to understand the core set of Intel x86/x86-x64 architecture and assembly so as to be able to read and understand short programs in disassembled form.
- Live demo lecture style. Lots of practice lab/hands-on exercises and real-world case studies in a controlled environment.
- Knowledge of the classes, techniques for analyzing binary programs with both disassemblers and debuggers. As well as providing a deeper understanding of how both disassemblers and debuggers actually work.
- Provide detailed knowledge on reverse-engineering methodology, tools and techniques.
- Ability to analyze code and behavior, identify and characterize functionalities of malicious software, particularly focusing on self-defense mechanisms in Windows executables.

WORKSHOP OUTLINE

Module 1: Lab Setup - Building Your Reverse Engineering & Analysis Lab Environment

- Setting up the Lab (lab architecture overview, setting up Reverse Engineering & Windows analysis lab)
- Optimizing the lab for better and more efficient Reverse Engineering & Analysis.
- Deploying Flare-VM, REmnux and connecting to InetSim
- Tools of the trade (deployment and overview)

Module 2: Introduction to Code Reverse Engineering

- Legal Aspects
- Basic C Programming and Reverse Engineering
- Decompilation and Architecture
- Lab 1: Decompiling JIT Program

Module 3: x86/x86-x64 Assembly - Data, Modes, Registers, Memory Access, Instructions

- Introduction to x86/x86-x64
- Assembly Syntax
- Data Representation
- Registers
- Memory Access
- Addressing Modes
- Lab 2: Building and Running Assembly Programs
- Understanding Condition Code
- Analyzing and Debugging Assembly Code
- Functions and Control Flow
- Compilers and Optimizers

Module 4: Reverse Engineering & Cracking - Tools and Strategies

- Basic REconnaissance
- Reverse Engineering Strategy
- Key Checkers and Generators
- Lab 3: Keygen Introductory
- Patching and Advanced Tooling

Module 5: Advance Static Analysis

- The Portable Executable(PE) Structure
- Purpose and Goals of Malicious Code Analysis
- Understanding Signature Names and VirusTotal Overview
- IoC vs. IoA
- Identifying File Types
- Calculating Hashes
- Strings Extraction
- Introduction to the Windows API
- IDA Pro
- Recognizing C Code Constructs In Assembly
- Lab 4: Packing Analysis
- Lab 5: Dissecting Real World Malicious Code

Module 6: Advance Dynamic Analysis

- Introduction to Dynamic Analysis
- Working with Process Explorer
- Extracting IoCs
- Working with Procmon
- Lab 6: Attack Flow PCAP Analysis Overview
- Lab 7: Dynamic Reverse Engineering Malicious Code
- Lab 8: Detecting Malicious Code with YARA

Module 7: Defense and Advance Anti-Reverse Engineering

- Obfuscation
- Anti Debugging
- Lab 9: Anti-Debugging
- Tamper-Proofing
- Packing
- Lab 10: Detecting and Unpacking
- Virtualization
- Cryptors/Decryptors

RESERVED FOR NOTES

CHANGE LOG

10.06.2024. Syllabus Published.