Reversing Modern Binaries: Practical Rust & Go Analysis

Instructors: Fuzzinglabs

This 3-day training combines two comprehensive modules, focusing on reverse engineering Rust binaries and Golang binaries. Participants will gain hands-on experience with tools, techniques, and advanced malware analysis, concluding with a capstone project to reinforce their skills. Designed for malware analysts, reverse engineers, and security professionals, this course provides the knowledge needed to tackle real-world reverse engineering challenges in modern programming languages.

KEY LEARNING OBJECTIVES

- Understand Rust & Go Fundamentals
- Comprehend Rust & Go Compilation and Runtime
- Analyze Rust & Go Structures and Control Flow
- Utilize Tools for Rust & Go Reverse Engineering
- Apply Advanced Reverse Engineering Techniques
- Reverse Engineer Malware Techniques in Rust and Go

Day 1: Rust Reverse Engineering

Module 1: Introduction to Rust and Reverse Engineering

- History, philosophy, and features of Rust
- Why attackers use Rust for malware development
- Fundamentals of reverse engineering
- Setting up the reverse engineering environment (Ghidra, GDB, LLDB, Radare2)

Module 2: Rust Compilation and Runtime

- Compilation process: From source code to machine code (LLVM/MIR)
- Rust runtime (Linux and Windows)
- Rust's calling conventions, memory management, symbol mangling/demangling

Module 3: Analyzing Rust Structures and Control Flow

- Understanding data structures (Option, Result enums, slices, structs)
- Analyzing control flow: Functions, methods, loops, conditionals
- Exercise: Reverse engineering a simple Rust program

Module 4: Tools and Techniques for Rust Reverse Engineering

- Configuring Ghidra for Rust binaries
- Debugging with GDB/GEF and LLDB

- Practical exercises with Rust debugging and analysis

Module 5: Advanced Reverse Engineering and Obfuscation Techniques

- Tackling stripped binaries: Symbol recovery and function inlining
- Handling obfuscation techniques: String obfuscation, anti-debugging
- Exercise: Reverse engineering an obfuscated Rust binary

Module 6: Advanced Malware Reverse Engineering Techniques in Rust

- Analyzing process hollowing, API hooking, and DLL injection in Rust malware
- Understanding loader injection and packers in Rust binaries
- Practical exercises with Rust malware samples

Module 7: Rust Malware Analysis

- Case studies (e.g., Luca Stealer)
- Writing YARA rules for Rust binaries
- Exercise: Analyzing a Rust malware sample and crafting YARA rules

Day 2: Golang Reverse Engineering

Module 8: Introduction to Golang and Reverse Engineering

- Overview of Go language and its features
- Why attackers use Go for malware development
- Go compilation (go build, go install) and runtime basics
- Exercise: Reversing a simple Go binary

Module 9: Basics of Golang Reversing

- Challenges in reversing Go binaries
- Tools for Go analysis: Ghidra, Radare2, GDB, and IDA Pro
- Go runtime analysis: Common runtime functions, calling conventions
- Analyzing Go binary sections and basic structures (int, string, slice, map)
- Exercise: Analyzing a Go binary with basic structures

Module 10: Analyzing Go Structures and Control Flow

- Go structs and interfaces
- Understanding control flow in Go: Functions, methods, loops, and error handling
- Exercise: Reverse engineering a Go program

Module 11: Advanced Go Reversing Techniques

- Challenges with stripped Go binaries
- Advanced Go features: Goroutines, channels, synchronization (wait groups, mutex)

- Analyzing Go's memory management and reflection
- Exercise: Reversing a Go binary with advanced features

Module 12: Go Malware Analysis

- Case studies of Go-based malware
- Writing YARA rules for Go binaries
- Exercise: Reversing Go malware and crafting YARA rules

Day 3: Capstone Projects

Module 13: Capstone Project and Conclusion

- Reverse engineering complex Go & Rust applications
- Guided capstone project combining advanced techniques from both Rust and Go
- Review of core concepts and open Q&A session
- Additional resources for further learning

Prerequisites

- Basic knowledge of Rust and Go programming.
- Familiarity with reverse engineering concepts and assembly language.
- Familiarity with scripting (Python, Bash) and Linux.
- SKILL LEVEL: BEGINNER / INTERMEDIATE

HARDWARE REQUIREMENTS

- A working laptop capable of running virtual machines
- 8GB RAM required, at a minimum
- 40 GB free Hard disk space
- Administrator / root access MANDATORY

SOFTWARE REQUIREMENTS

- VirtualBox installed with guest addition
- IDA Pro and/or Binary Ninja would be helpful but not required

WHO SHOULD ATTEND

Malware analysts, reverse engineers, security researchers, vulnerability researchers, software developers