Pre-Reverse Engineering Workshop

CTF Training Session

November 9, 2025

Contents

1 Introduction

This document is a preparatory guide for beginners in reverse engineering for CTF competitions. It covers essential concepts, tools, and exercises needed before attempting binary reversing challenges.

2 Computer Basics

- How programs run: source code → compilation → binary execution
- Process vs program
- Memory layout: stack, heap, data, text
- System calls

Resources:

- Crash Course: Computer Science (YouTube):
  `https://www.youtube.com/playlist?list=PL8dPuuaLjXtNlUrzyH5r6jN9ulIgZBpdo`

3 Binary and Data Representation

- Bits, bytes, hexadecimal
- ASCII representation
- Endianness (Little vs Big)
- Signed vs unsigned integers (two's complement)

Exercises:

1. Convert decimal to binary and hex.
2. Decode hex strings to ASCII text.

Tools:
CyberChef: `https://gchq.github.io/CyberChef/`
Hexed.it: `https://hexed.it/`

4 Operating Systems Basics

- Executable file formats:
  - ELF (Linux)
  - PE (Windows)
  - Mach-O (macOS)
  - APK (Android application package)

- Libraries (`.so`, `.dll`)

Notes on APK:

- APK is a packaged Android app. It contains compiled bytecode (DEX), resources, and a manifest.
- Reversing APKs often uses tools like `apktool`, `jadx`, and Android debuggers.

Resources:

- ELF file guide:
  `https://linux-audit.com/elf-binaries-on-linux-understanding-and-analysis/`
- PE file format basics:
  `https://learn.microsoft.com/en-us/windows/win32/debug/pe-format`

5 Programming Basics (C)

- Variables, functions, pointers
- Control flow: if, loops, switch
- Why C/C++ binaries are common in reversing

Resources:

- Learn C in Y Minutes:
  `https://learnxinyminutes.com/docs/c/`
- TutorialsPoint: C Programming:
  `https://www.tutorialspoint.com/cprogramming/index.htm`

6 Assembly Introduction

- CPU registers, memory, and instructions
- Common x86/x64 instructions:

```
mov eax, 1
push ebx
pop ecx
call func
ret
cmp eax, ebx
jmp label

```

Resources:

- GuidedHacking: Assembly Tutorial:
  `https://guidedhacking.com/threads/x86-assembly-tutorial.7730/`
- Online Assembler Visualizer (Compiler Explorer):
  `https://godbolt.org/`

7 Static vs Dynamic Analysis

- Static: reading disassembly
- Dynamic: running program in a debugger
- Basic tools: Ghidra, IDA Free, x64dbg, GDB, Cutter
- Android-specific tools for dynamic analysis: `adb`, `frida`, `Android Studio` debugger

Resources:

- Ghidra Beginner Guide:
  `https://ghidra-sre.org/`
- x64dbg Introduction (YouTube):
  `https://www.youtube.com/watch?v=6P7p6pC1XxE`

8 CTF Context

- Reversing challenges: password checks, license checks, simple crackmes, Android app reversing
- Examples for practice:
  - CTF101 - Reverse Engineering:
    `https://ctf101.org/reverse-engineering/overview/`
  - picoCTF - Reverse Challenges (easy):
    `https://play.picoctf.org/practice?category=3&difficulty=1`

9 Exercises (Optional)

1. Write a simple C program that compares a password and returns OK/FAIL.
2. Compile it and open in Ghidra or GDB to analyze how it works.
3. Get an APK from a simple Android sample. Use `apktool` and `jadx` to inspect manifest and code. Identify where the password check is implemented.