Reverse Engineering Concept

Real malware analyst doing reversing

Content

- Overview of Reversing
- How to learn
- Basic x86
- Tools

Introduction

- Fundamentals of reversing engineering (RE)
- Using a hands-on experience with RE tools and techniques.
- You will be introduced to RE terms and processes
- Reviewing RE tools and malware techniques.

It's a ART of dissecting and rebuild. Understand the inner working and its mechanism

Overview of Reversing

To know what really the program do

Understand how they work

manipulate their behavior and so on

by reading, understanding and debugging the program's code.

What kind of analyst

- Two categories of malware analyst
- First:
  - Only do dynamic analysis
  - Using dynamic analysis tools such Sysinternals
  - Suite, sandboxes, etc
- Second:
  - All of above
  - Analyze deeper, static analysis and debugging, reversing

What does it mean to be a reverse engineer?

- **You can**
  - Take things apart to figure out how it works
  - Love puzzle solving
  - Develop experiments and tools
  - Think outside the box
  - Constantly learn new things

Use cases for RE

- Malware analysis
- Security / Vulnerability Research
- Driver development
- Compatibility fixes
- Legacy application support

Overview

- Malware analysis and vulnerability research often involves Reverse engineering.
- Reverse engineering focus on the assembly and the source code of the program

```
.text:00401244 loc_401244:                             ; CODE XREF: DialogFunc_0+D7↑j
.text:00401244                 mov     esi, esp
.text:00401246                 push    100h
.text:0040124B                 push    offset String   ; lpString
.text:00401250                 push    3E8h            ; nIDDlgItem
.text:00401255                 mov     eax, [ebp+hDlg]
.text:00401258                 push    eax             ; hDlg
.text:00401259                 call    ds:GetDlgItemTextA
.text:0040125F                 cmp     esi, esp
.text:00401261                 call    __chkesp
.text:00401266                 mov     esi, esp
.text:00401268                 push    100h            ; cchMax
.text:0040126D                 push    offset String   ; lpString
.text:00401272                 push    3E8h            ; nIDDlgItem
.text:00401277                 mov     ecx, [ebp+hDlg]
.text:0040127A                 push    ecx             ; hDlg
.text:0040127B                 call    ds:GetDlgItemTextA
.text:00401281                 cmp     esi, esp
.text:00401283                 call    __chkesp
.text:00401288                 cmp     eax, 5
.text:0040128B                 jnb     short loc_4012B1
.text:0040128D                 mov     esi, esp
.text:0040128F                 push    30h             ; uType
.text:00401291                 push    offset Caption  ; "crackme"
.text:00401296                 push    offset Text     ; "Name is too short"
```

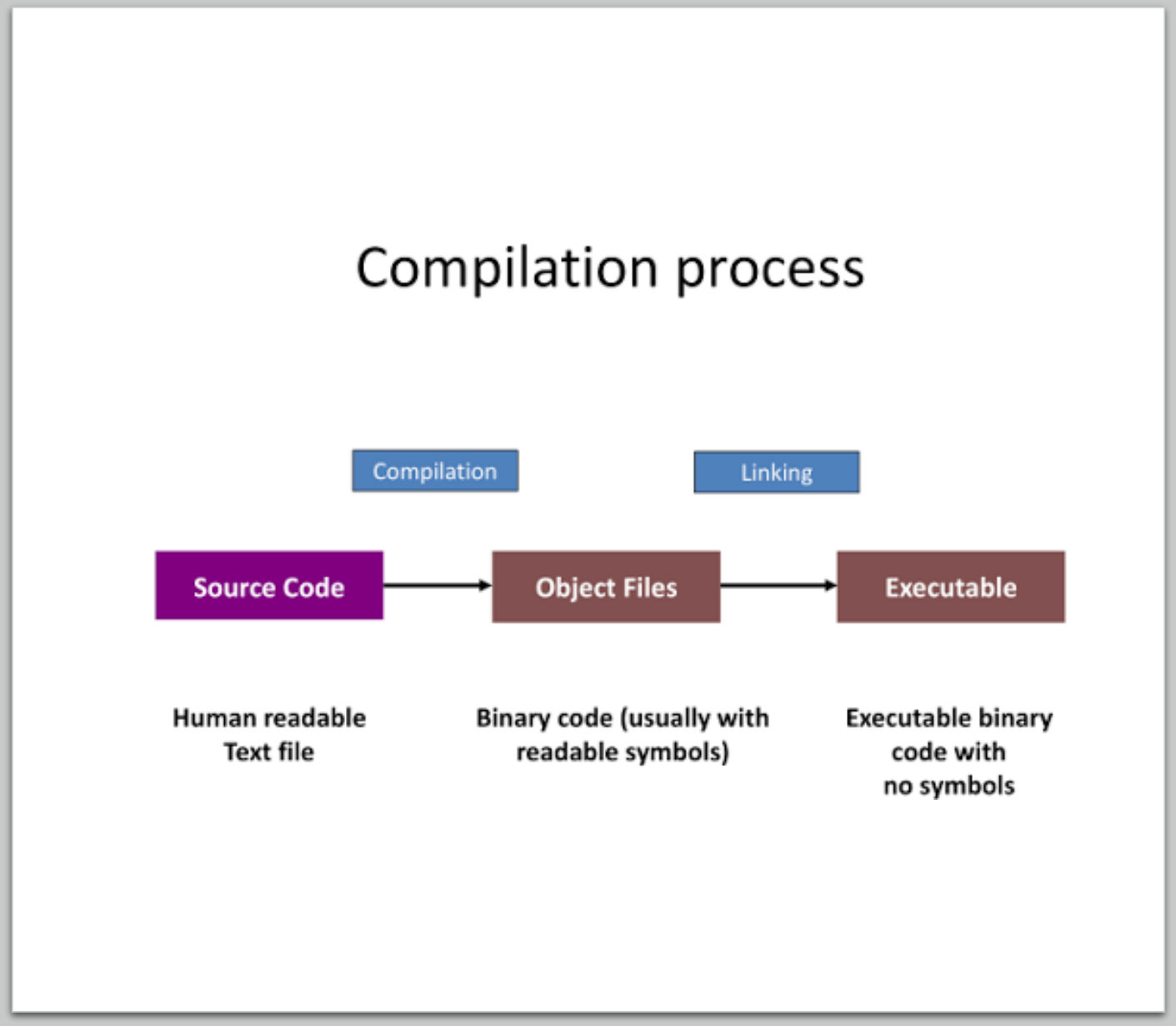
Decompile: _main - (crackme0x01.exe)

Decompile: _main - (crackme0x01.exe)

```
7
8   .text(local_20);
9   ___main();
10  .text("IOLI Crackme Level 0x01\n");
11  .text("Password: ");
12  .text("%d",&local_8);
13  if (local_8 == 0x149a) {
14    .text("Password OK :)\n");
15  }
16  else {
17    .text("Invalid Password!\n");
18  }
19  return 0;
20 }
```

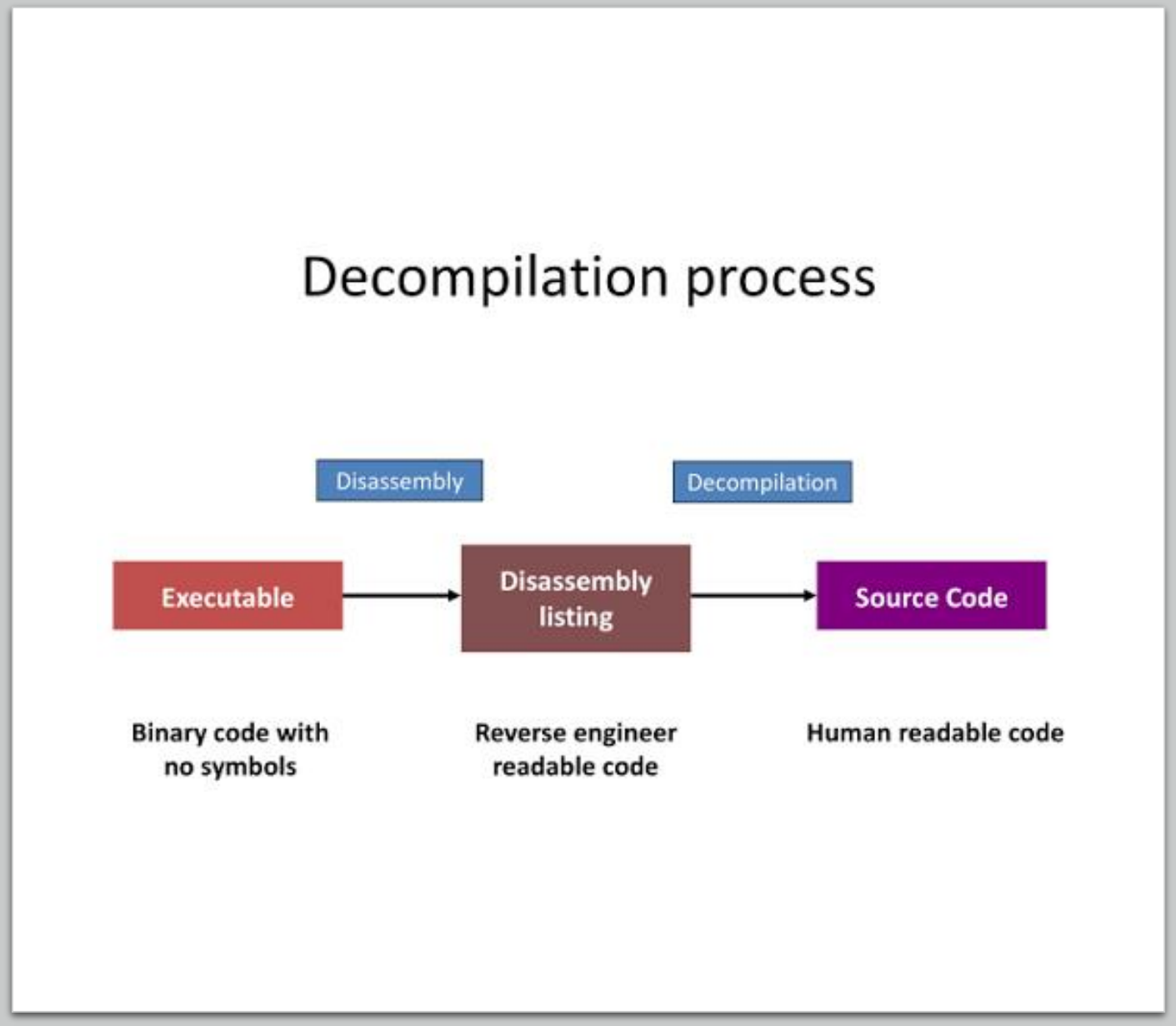
Methodology concept

- **Forward engineering**
- Method of creating or making an application
- It takes more time to construct or develop an application.

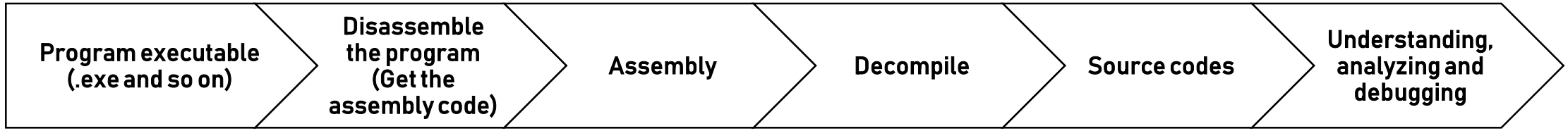


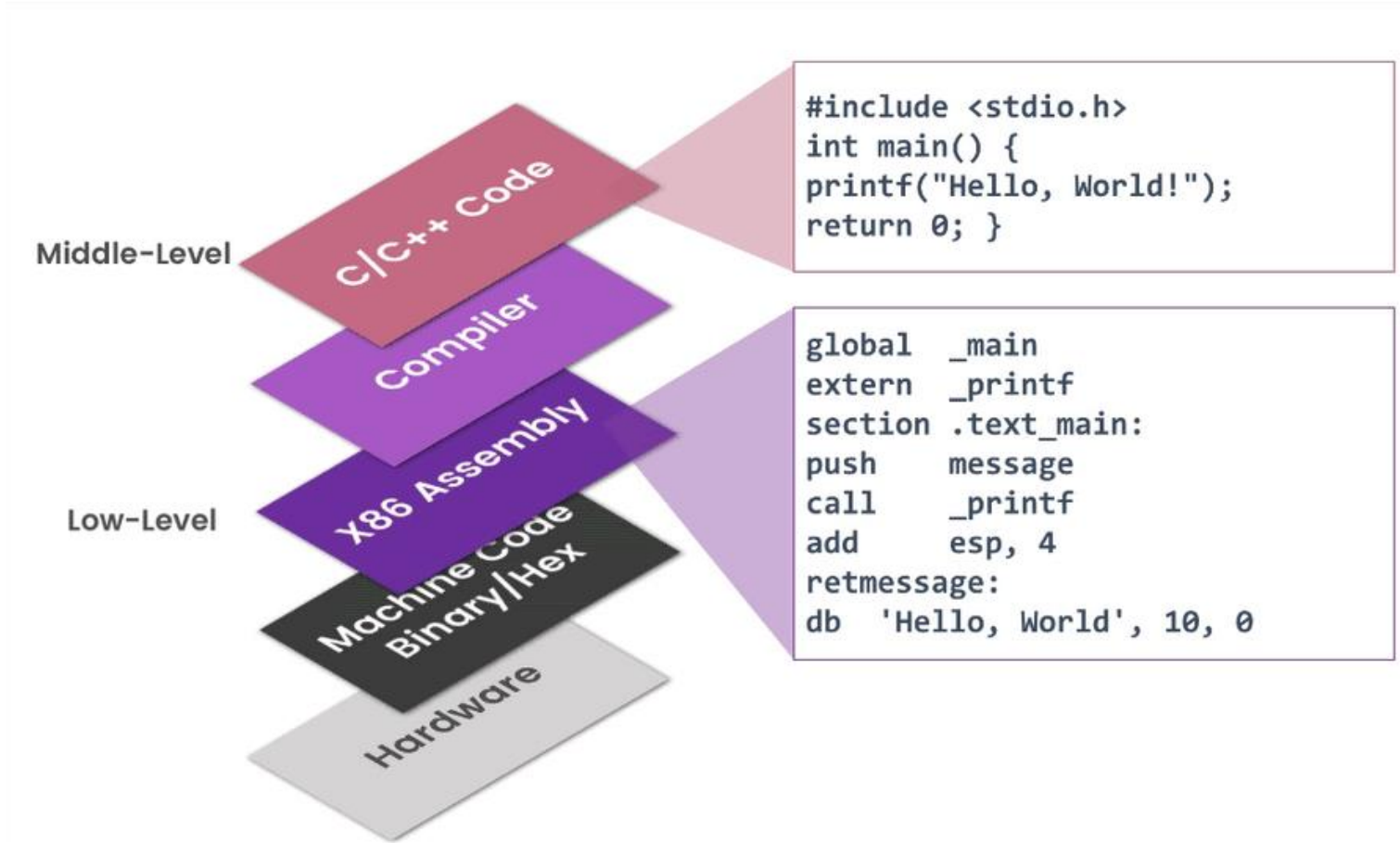
Methodology concept

- **Reverse engineering**
- Process of forward engineering in reverse.
- opposite of forward engineering.



Reverse engineering process



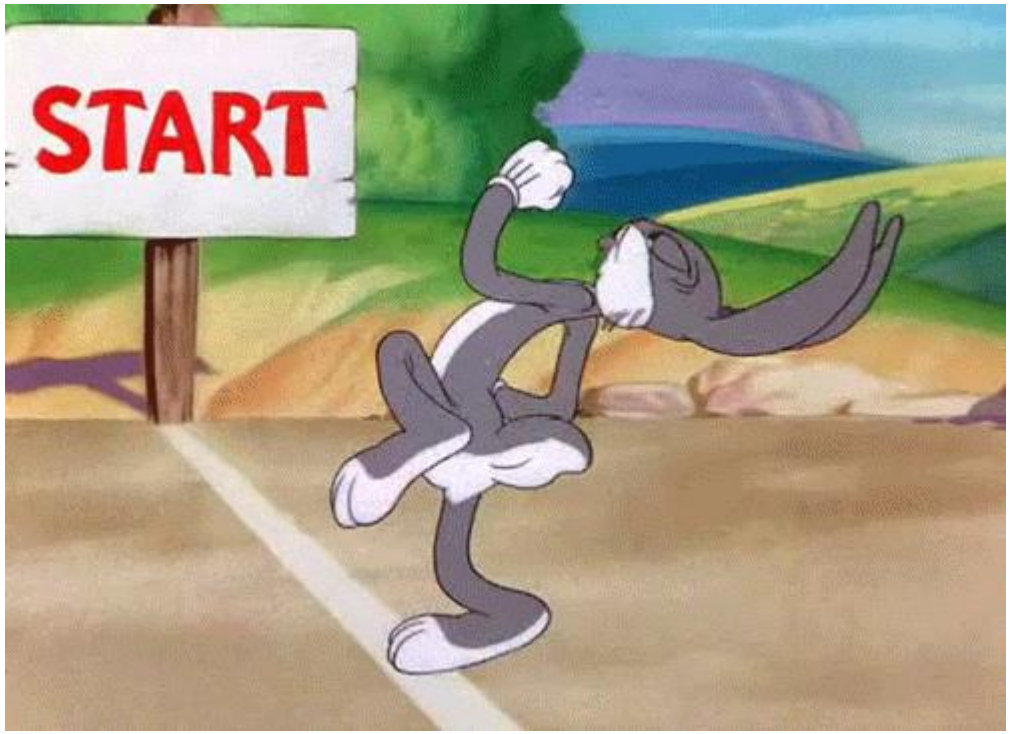

Middle-Level
Low-Level
C/C++ Code
Compiler
X86 Assembly
Machine Code
Binary/Hex
Hardware
#include <stdio.h>
int main() {
printf("Hello, World!");
return 0; }
global _main
extern _printf
section .text_main:
push message
call _printf
add esp, 4
retmessage:
db 'Hello, World', 10, 0

Requirement in Reverse Engineering

Skills and knowledge required (Technically for industry)

- Understanding of computer architecture and organization
- C/C++ programming
- Assembly programming
- Operating Systems Internals (e.g. Windows Internals)
- File Formats (e.g. Portable Executable (PE))
- Logic thinking and Research skills
- Scripting (Python, Javascript, VB Script)

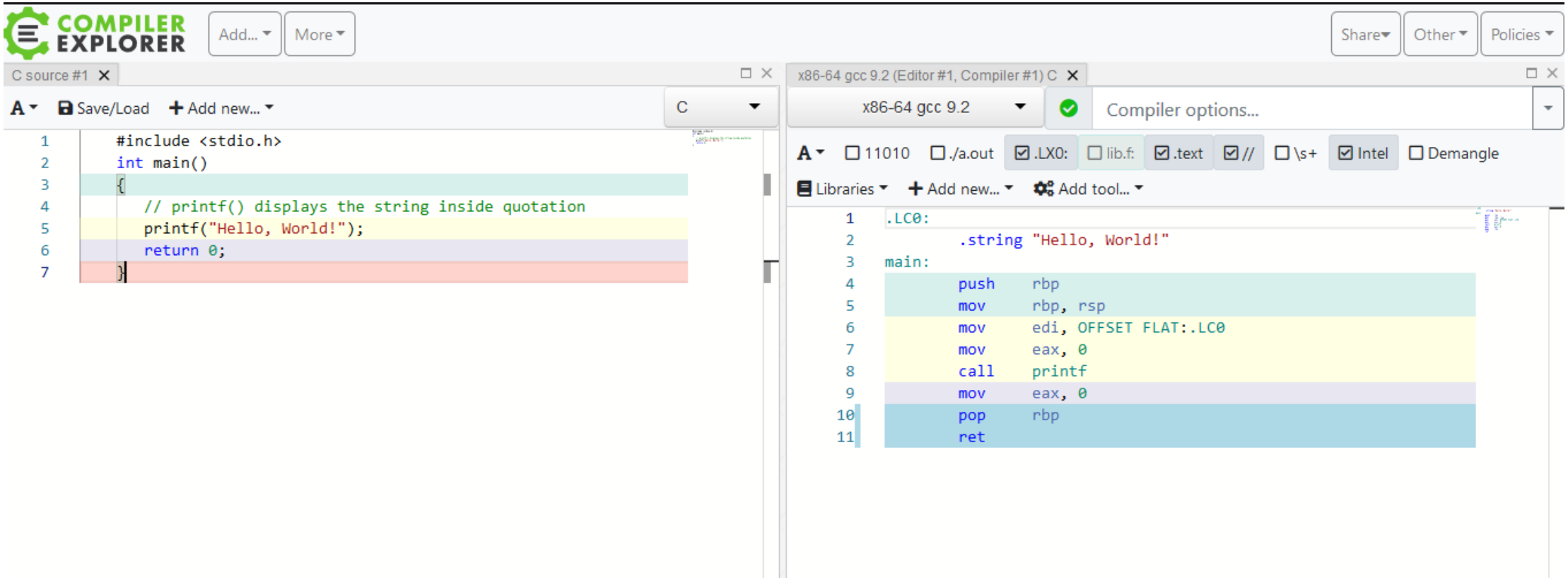
How to start?



How to start

- Start with C programming
  - C programming language
  - Learn how to use debugger to debug your C
  - program
- Assembly language for the platform architecture
  - x86
- Scripting
  - Python, Javascript

Try to convert from C to Assembly



Scripting

- A programming language for a special run-time environment that automates the execution of tasks.
- Automation
  - Save time

Scripting (cont.)

- Let’s take a look of an example how scripting are used in reverse engineering.
- IDA Pro and Python

Types of approach

- Static Code Analysis
  - Read assembly code
  - Read the source code
  - Search for strings!
  - Analyzing without executing the program
  - Analyze the packer, obfuscator, crypter, and protector using tools

Types of approach

- Dynamic Code Analysis
  - Debug the assembly code or the decompiled code.
  - Step through the code line by line,
    - study and analyze how it executes,
    - what it does and
    - how the behavior.
  - For fast results of understanding the output and accurate.

Tools

- Static
  - IDA Pro
  - Ghidra
  - DnSpy
  - Cutter (Radare2)
- Dynamic
  - IDA Pro
  - x64Dbg
  - OllyDbg
  - Immunity Debugger
  - DnSpy

The challenges of Malware Reversing

- Obfuscation
- Packed and obfuscated malware
- The anti
  - Anti-debug
  - Anti-dump
  - Anti-disassembly
- Other clever techniques to prevent reversing

Other tips

- Always be coding (in C) and reversing.
- Reverse takes time
- Practice make perfect
- Participate reverse engineering challenges (CTF)

Books to learn Reversing

- Any Assembly Language Programming books.
- Any C Programming books.
- IDA Pro Book
- Practical Malware Analysis
- Practical Reverse Engineering
- Reversing: Secrets of Reverse Engineering
- Reverse Engineering for Beginners