An Introduction to Software Reverse Engineering.

Biel A. P.

2020

Contents

1 Introduction **3**

2 Files **5**

2.1 What is a file? 5

2.2 How a file become a file? 5

2.3 How a program is executed? 7

1 Introduction

Hi, Welcome! My name's Biel and I'm a Cyber Security Analyst. I'm writing this book with the goal to introduce reverse engineering to people who has the interest in it but doesn't know where to start. It is assumed that the readers already have programming experience, if not, this book may not be for you at the moment.

We will talk about files, binary files, PE, ELF, and a lot of other reverse engineering related subjects so people who have never had contact with reverse engineering can start doing it.

During the book, the software I'm using is:

- OS: Arch Linux;
- Shell: Bash;
- Dumpers: objdump, hexdump;
- Text Editor: VIM;
- Hex Editor: hx;

- Tracers: ltrace, strace;
- Disassembler: radare2;

But feel free to use the software you are used to, I use those because they are free and Open Source but I won't try to convert you to the Open Source... Maybe on my next book...

2 Files

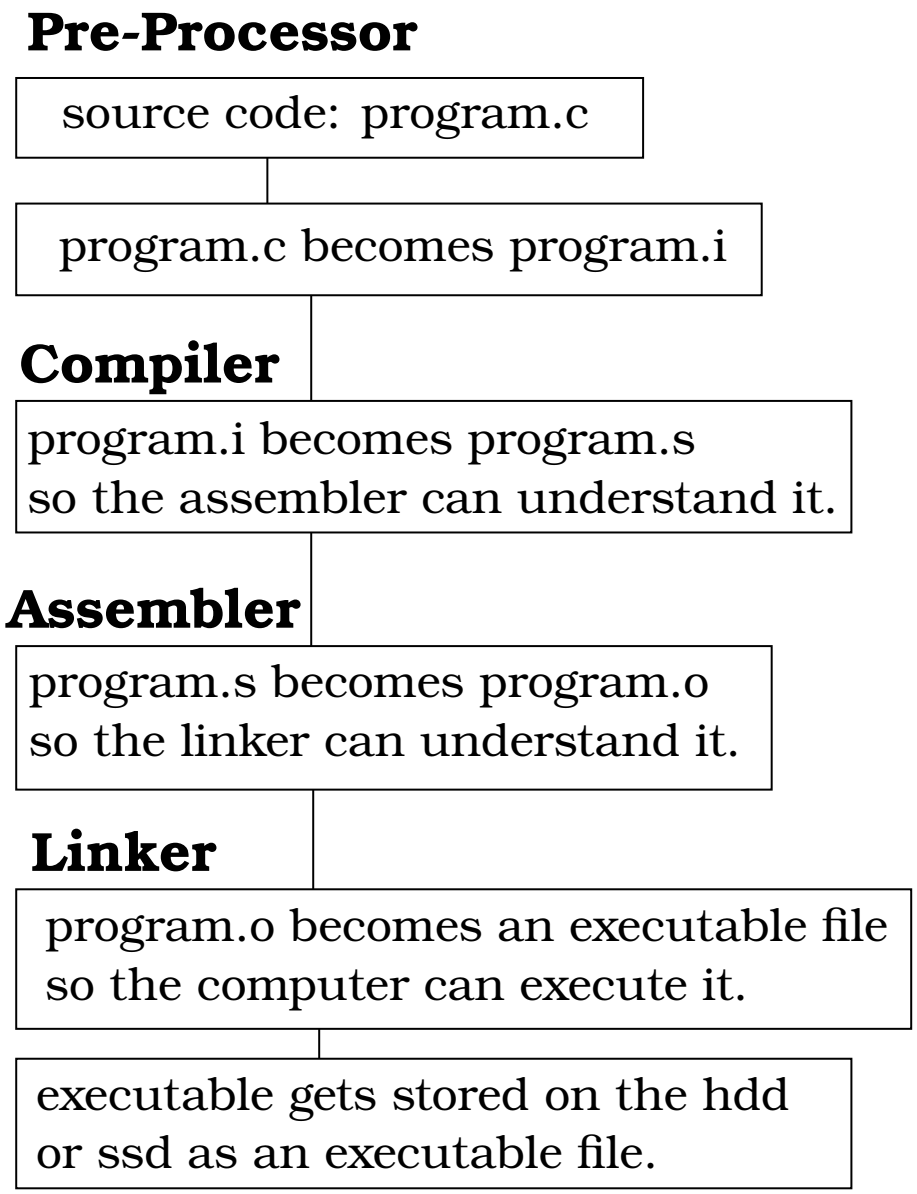
2.1 What is a file?

In short, a file is a sequence of binaries stored somewhere that is referenced by another element. The minimum size something has to have to be considered a file, is 1 byte, or 8 bits, or 10101010 and it can be anything, a text file, an audio file, an executable file, anything, as long as it has 1 byte.

2.2 How a file become a file?

So, as you learnt, a file is something with at least 1 byte, 1 byte in binary code is *10101010*, but how does this works? how the computer understands this?

Let's imagine you wrote a program in **C**, that is what happens behind the scenes when you compile the program:



After that, the program is ready for execution, depending on the hardware and the programming language you are using, this process can be slower or faster. Usually low level programming languages such as **Assembly** and **C** compile faster than high level pro-

gramming languages such as **Python** and **Java**.

2.3 How a program is executed?

Basically, after the program is stored on the hard drive or solid state drive, it becomes a process on the computer. This process go through a **loader** and gets allocated on the **Process Address Space**, usually on the **RAM**.