FIRST EDITION – CHAPTER 1 REV 1

Kevin Thomas

Forward

This book is not associated or endorsed by the Rust Foundation. This is purely a FREE educational resource going step-by-step on how to build and reverse engineer Rust binaries.

Rust redefines how a binary is compiled. It does not operate with similar patterns like we see in older languages and is simply the most difficult language to reverse engineer in history.

Many of the C-style decompilers within the popular reversing tools are literally rendered useless with Rust.

Malware Authors are going the route of Rust as there are few tools that exist yet to properly statically analyze the compiled binaries which coupled with its speed and memory safety, it is very difficult to understand what it is actually doing.

The aim of this book is to teach basic Rust and step-by-step reverse engineer each simple binary to understand what is going on under the hood.

We will develop within the Windows architecture (Intel x64 CISC) as most malware targets this platform by orders of magnitude.

In later chapters we will within a Raspberry Pi 64-bit ARM OS so that you can get a perspective of what hacking that architecture looks like in Golang at the binary level.

Let’s begin...

Table Of Contents

Chapter 1: Hello Rust

We begin our journey with developing a simple hello world program in Rust on a Windows 64-bit OS.

We will then reverse engineer the binary in IDA Free.

Let’s first download Rust for Windows.

https://www.rust-lang.org/tools/install

Let’s download IDA Free.

https://hex-rays.com/ida-free/#download

Let’s download Visual Studio Code which we will use as our integrated development environment.

https://code.visualstudio.com/

Once installed, let’s add the Rust extension within VS Code.

https://marketplace.visualstudio.com/items?itemName=rust-lang.rust-analyzer

It is important to update Rust so I would encourage you to run this before every project.

```
rustup update
```

Let’s create a new project and get started by following the below steps.

```
Cargo new one_hello
```

Now let’s populate our **main.rs** file with the following.

```
fn main() {
    println!("Hello, world!");
}
```

Let’s open up the terminal by click CTRL+SHIFT+` and type the following.

```
Cargo build
```

Let’s run the binary!

```
.\target\debug\one_hello.exe
```

Output…

```
Hello, world!
```

Congratulations! You just created your first hello world code in Rust. Time for cake!

We simply created a hello world style example to get us started.

In our next lesson we will debug this in IDA Free!