FIRST EDITION – CHAPTER 1 REV 1

Kevin Thomas

Forward

I remember a time before the days of the internet where computers were simple yet elegant and beautiful in their design, logic and functionality.

I was a teenager in the 1980’s when I got my first Commodore 64 for Christmas and the first thing I did was tear it out of the box and get it wired up to my console TV as the only thing I needed to see was that blinking console cursor on that blue background with the light blue border.

It was a blank slate. There were no libraries. There were no frameworks. If you wanted to develop something outside of the handful of games that you could get for it, you program it from scratch.

In addition to the C-64 there was a 300 baud modem with a 5.25” floppy disk which read DMBBS 4.8. I quickly read the small documentation that came with it and quickly took over the only phone line in the household.

I set up my BBS or bulletin board system, and called it THE ALLNIGHTER. I set it up and no one called obviously as no one knew it existed. I joined a local CUF group, computer user federation, where I met another DMBBS 4.8 user which helped me network my message boards to him.

At a given time of day my computer would call his and send my messages to his board and I would receive his messages from his board. It was computer networking before the internet and it was simply magic.

Over the next few months he taught me 6502 Assembler which was my first programming language that I ever learned. Every single instruction was given consideration of the hardware and a mastery over the computer was developed as we did literally everything from scratch on the bare metal of the hardware.

Today we live in an environment of large distributed systems where there are thousands of libraries and dozens of containers within pods in a large orchestrated Kubernetes cluster which defines an application.

Between the 1980’s and current, the birth of higher-level languages has made it possible to develop in a timely manner even on the most sophisticated distributed systems.

As we work within a series of large cloud ecosystems, there exists a programming language called Golang, or Go for short, which allows for easy software development to take advantage of multiple cores within a modern CPU in addition to out-of-the-box currency and ease of scale for enterprise-level network and product design.

With every great technology there arises threat actors that exploit such power.

Go can be compiled easily for multiple operating systems producing a single binary. The speed and power of Go makes it an easy choice for modern Malware Developers.

There are literally thousands of books and videos on how to reverse engineer traditional C binaries but little on Go as it is so relatively new.

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

We will start our journey on the Windows architecture (Intel x64 CISC) and later repeat the adventure on a MAC M Series core (ARM 64 RISC) and finally on a ARM 32-bit microcontroller (ARM 32 RISC THUMB) so we get a complete mastery over the domain.

Let’s begin...

Table Of Contents

Chapter 1: Hello Distributed System World x64

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

We will then reverse engineer the binary in IDA Free.

Let’s first download Go for Windows.

https://go.dev/doc/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 Go extension within VS Code.

https://marketplace.visualstudio.com/items?itemName=golang.go

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

New File
main.go

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

```
package main

import "fmt"

func main() {
	fmt.Println("hello distributed system world")
}
```

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

```
go mod init main
go mod tidy
go build
```

Let’s run the binary!

```
.\main.exe
```

Output…

```
hello distributed system world
```

Congratulations! You just created your first hello world code in Go. 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!