Emulators, Disassembly, and Anti-Debug tricks

赖纳文

December 17 2020

Abstract

This paper, as the title suggests, introduces a brief overview on several topics, including but not limited to:-

- Emulators[1]
- Disassemblers
- Anti-Debug tricks

For the purposes of demonstration, I will define a toy CPU architecture to emulate and to disassemble programs for that specific CPU architecture.

1 Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

[1]There is some discussion to be had on the difference between *Emulators*, *Simulators* and *Interpreters* (and perhaps some other things like Virtual Machines), which will be discussed later on.

1.1 Definition of my Toy Architecture

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Consectetur adipiscing elit ut aliquam purus. Sed risus ultricies tristique nulla aliquet enim. Faucibus purus in massa tempor nec. Euismod quis viverra nibh cras pulvinar mattis nunc sed. Sed blandit libero volutpat sed cras ornare arcu dui vivamus. Aliquam faucibus purus in massa tempor nec. Sit amet facilisis magna etiam tempor orci. Pellentesque elit eget gravida cum sociis natoque penatibus. Enim sit amet venenatis urna cursus eget nunc scelerisque viverra. Quis vel eros donec ac odio tempor orci. Viverra justo nec ultrices dui sapien eget mi. Vitae congue mauris rhoncus aenean vel elit. Id semper risus in hendrerit gravida rutrum quisque. Convallis tellus id interdum velit laoreet id. Dignissim suspendisse in est ante in

nibh mauris. Eu volutpat odio facilisis mauris sit. At elementum eu facilisis sed odio morbi quis. Tincidunt arcu non sodales neque.

Urna condimentum mattis pellentesque id nibh. Eget nullam non nisi est sit amet facilisis magna. Elit pellentesque habitant morbi tristique senectus. Mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan. Vel fringilla est ullamcorper eget nulla. Mauris cursus mattis molestie a iaculis at erat. Ac turpis egestas sed tempus urna. Eget duis at tellus at urna. Egestas sed sed risus pretium. Vulputate enim nulla aliquet porttitor lacus luctus accumsan. Donec pretium vulputate sapien nec sagittis. Tortor posuere ac ut consequat semper viverra nam libero justo. Eget nunc lobortis mattis aliquam faucibus purus. Sem viverra aliquet eget sit. Sit amet massa vitae tortor.

1.1.1 Opcodes

1.1.2 Addressing Modes

2 Disassembly

From the knowledge gleaned from the CPU Architecutre, given the bytes of a program running on that CPU, one could create a Disassembler.

3 Emulator

Assume we have a program meant to be executed in that architecture, and assuming we don't have a physical version of a machine running this toy architecture, one could simply use *static analysis* using the disassembler to find out what it does, by manually tracing the code in her/his mind. However, We can go one step further; we can utilise *dynamic analysis* by emulating/simulating the program.

4 Anti-Debugging Tricks

Completely different from the other sections.

5 Further Reading

See the References section.

References

[1] Asami. https://book.rvemu.app/index.html.

[2] V. M. del Barrio. *Study of the techniques for emulation programming.* URL `http://www.xsim.com/papers/Bario.2001.emubook`

[3] L. M. (M.Sc.). How to write an emulator (chip-8 interpreter). URL `http://www.multigesture.net/articles/how-to-write-an-emulator-chip-8-interpreter/`.

[4] I. Nazar. Gameboy emulation in javascript. URL `http://imrannazar.com/GameBoy-Emulation-in-JavaScript:-The-CPU`.