Reverse Engineering Challenges solved step-by-step to demonstrate the many uses of reverse engineering

Joel Maldonado Rivera
Advisor: Jeffrey Duffany
Computer Science Department

Abstract

The field of reverse engineering has seen many different applications such as analysis of computer viruses and malware such as trojan, worms, viruses, ransomware, and so on. Other uses involve analyzing legacy code to possibly recreate the code in a more modern program and can even be used to test the quality of software. These are just some of the uses of reverse engineering, which should be discussed and be more well known throughout people who practice coding. There are many different approaches to reverse engineering since one can use tools such as IDA, CFF Explorer, Ghidra, Hopper, GDB, and many others in order to examine the programs. Being able to properly understand how to use these tools will help in the proper understanding of what the code is doing and how it’s behaving which will be better demonstrated by solving reverse engineering challenges and explaining the methodology behind how they were solved.

Problem

Reverse engineering is an area that has seen many different implementations and approaches which has helped tackle different problems such as working with legacy code, analyzing malware, and has also been used to improve existing software. Despite the wide range of implementations that reverse engineering offers it is seldom taught as a core class in many programs which is why bringing more awareness to it and being able to demonstrate to other computer scientists the possibilities and uses it brings is what this project hopes to demonstrate.

Methodology

Using the methodology that was described all the challenges that were chosen were solved by using GDB and a guide was made in order to show the solutions and how each challenge’s solution was reached. Reverse engineering has various applications which not many computer scientists are aware of such as using it to test the quality of software, verifying legacy code in order to create a newer version of said code, malware analysis, and so on. The step-by-step guide will hopefully help those who are interested understand the basics of assembly and reverse engineering concepts which in turn may provide them with better insight on the topic of reverse engineering and the various uses it has.

Results and Discussion

This project aimed to solve six challenges from the crackmes one page which are all referenced. All of the challenges are unique and are a good starting place for those who have limited to no experience in reverse engineering since they help the user start getting used to using GDB and to the assembly language. These problems have a walkthrough that were made as part of the Design Project as well. Each write up explains each challenge and the assembly language generated by these in more depth and in a step-by-step manner in order to allow people who are interested in learning to have a guide.

In order to solve problems that were chosen from the crackmes one site, they were first analyzed by using commands such as the file command in order to get more details about what type of executable file was being worked on and then proceeding to use GDB in order to take a much more in-depth analysis of the executable file. GDB has many different commands that have different uses in helping better understand the code such as:

- disassemble(disas) which allows the user to disassemble a specific function or a function fragment thus allowing a closer look at the assembly code to that specific area.
- breakpoint which allows the user to specify a breakpoint which will make the execution of the program halt once reached.
- next instruction(ni) which allows the user to go to the following instruction in the assembly code therefore giving the user a chance to verify changes that occurred.
- examine(s) which allows the user to examine the examination of the provided memory but also allows one to place a flag in order to indicate how the values should be represented before it is displayed.
- instructions in order to see debugging symbols that can be accessed in order to help with the analysis of the program.

These challenges are not stripped executables, which allows for the use of functions commands to find debugging symbols and functions to set the breakpoints. For every challenge, these commands can be used in order to view functions of interest and set breakpoints for them to then go and disassemble them in order to analyze what the code is doing. By understanding the assembly and verifying areas in memory with the examine command the solutions of the challenges are determined.

Conclusions

The challenges that were chosen were all able to be solved by using GDB and interpreting the assembly which was disassembled. There are tools that make this process easier but having a core understanding of the assembly and using GDB set up good fundamentals. A step-by-step documentation on how to solve the challenges was also created ‘hopefully motivating computer scientists to get started. Students gain some basic knowledge of reverse engineering concepts.

Future Work

There are many different proposals that can be considered for future works such as using specific tools such as IDA or Ghidra to highlight the capabilities of said tool and how they are able to fend off some of the techniques that exists to try to stop the reverse engineering process from being achieved. Reverse engineering can also be utilized for malware analysis, which is more pertinent to cybersecurity, to understand how the malware work and what vulnerabilities are being exploited to be able to patch and counter the use of these. A future project where some diverse types of malwares are analyzed, handled and properly reverse engineered would be ideal, especially to highlight why it’s important for cybersecurity.

Acknowledgements

I would like to acknowledge and thank Dr. Jeffrey Duffany for his recommendation and guidance during the process of the Design Project. I’m also thankful for the master’s program overall and I am thankful to the Computer Science Department for the growth they have let me develop.

References


