Reverse Engineering

Accordi Gianmarco

1 Introduction

Reverse Engineering is the process of understanding how system made by man is working, by looking at it an trying to reverse the operations it does in order to understand its behavior. Its like a scientific research but in this case we are not interested in natural phenomenon. This kind of process has been applied to a lot of different fields, from mechanical engineering to chemical engineering, so as the name suggested it is applied to all field of engineering: these are the fields in which we can take apart the work done by other human being and trying to understand how they have build such a system. The necessity of this such an analysis of a system is usually required because creator of a system tends to disclose how they've built it, this mean that they obscure their work in order to protect their creators rights. The purpose of this document is to give an introduction on **Reverse Engineering** specifically in the field of Software Engineering. The process of Reverse Engineering applied to Software products can be done at any level of the production, but it usually involves two stages:

- **Redocumentation**: since usually the given software product that we want to analyze is already compiled we want to be able to reach an higher level of abstraction to better understand the code;
- **Design Understanding**: once that we have been able to get an higher abstraction of our code we can use our capacity and knowledge in order to understand what the program does and how it does it, so we want to follow the development process of the creators of such code.

Disclose the design features can be conducted with two approaches:

- **Dynamic Analysis**: this is an on the field approach, in which you launch the program and you register what the program is doing, and by analyzing the impact it has on the environment(print,systemcalls,...) you try to figure out what it does;
- **Static Analysis**: analyze the code without launch it, you analyze only the output you've obtained from the Redocumentation part.

In next sections we will analyze this approaches along with the stage involved in doing them. Then we proceed in order to define which is the best strategy based on what we have seen. Finally an example is provided taken from a CTF.

Reverse Engineering in Software Engineering As previously stated this document focus its attention on Reverse Engineering applied to software. When a new software is released it is usually provided already compiled to the specific architecture, so you get the binary of this code. The binary, as the name suggests, contains binary instructions that are targeted to be understood by a specific machine with a specific architecture, this means that it cannot be understand easily from a human being. For the developers this is usually a wanted feature, since this make the reverse engineering process more difficult for other companies that wants to disclose their design features. To be even more secure companies also rely on the usage of some Obfuscation Technique that makes the reverse engineering process even harder. This leads to the development of new research fields focused on trying to remove obfuscation, also in an automate way [5]. Some techniques used for obfuscate binaries are Packing(in which the source code pack and unpack the .text section as long as it execute)[2], Dynamic Code Mutation(in which the code mutate during execution), Code Generation(the program generates code during program execution) or by Binary Instumentation [4]. So in the end software is usually not fully disclose for two main reasons: protection of intellectual property and malware. In next sections we will see different techniques that can be used in order to defeat make the process of reverse engineering more easy. As a small note the fact that usually companies tends not to disclose their code leads to problem in computer security, since it cannot be reviewed from external person in order to find security problems. Such way of thinking is changing in recent days [1].

2 Stages

2.1 Redocumentation

The program we get is usually a binary file so the first steps in order to get an higher level representation of it is to disassemble it

If you need a reference to a link please use footnotes[1]. If you are referencing to a paper or book you the bibliografy. Use texttt when refering to register name or code in general like `EIP` or `call`

You can start from this example to build your report. Include in your final submission all the source code needed to perform any edit in the future including images source as well.

2.2 Example of code

Here you can fine an example of how to add some code. It is using the package minted with basic option enabled. You can write code in place:

```
1 import numpy as np
2
```

[1] `https://jinblack.it`

Figure 1: Compilation and Decompilation steps[3]

```
3  def incmatrix(genl1,genl2):
4      m = len(genl1)
5      n = len(genl2)
6      M = None #to become the incidence matrix
7      VT = np.zeros((n*m,1), int)  #dummy variable
8
9      #compute the bitwise xor matrix
10     M1 = bitxormatrix(genl1)
11     M2 = np.triu(bitxormatrix(genl2),1)
12 ...
```

You can reference to a line The important line is line 5.
You can use an external file and ave the code as listing.
The important line is line 3 of Listing 1.

3 Conclusion

References

[1] Jaap-Henk Hoepman and Bart Jacobs. Increased security through open source. *Communications of the ACM*, 50, 02 2008.

[2] Mario Polino. *Packers and Evasive Techniques Slide.*

[3] Mario Polino. *Reverse Egineering Slide.*

[4] Mario Polino, Andrea Continella, Sebastiano Mariani, Stefano D'Alessio, Lorenzo Fontana, Fabio Gritti, and Stefano Zanero. *Measuring and Defeating Anti-Instrumentation-Equipped Malware.* 2017.

Figure 2: Img example

[5] S. K. Udupa, S. K. Debray, and M. Madou. *Deobfuscation: reverse engineering obfuscated code*. 2005.

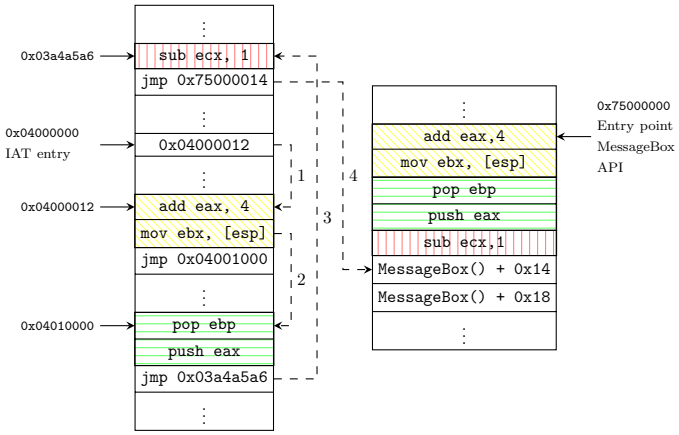

Figure 3: You can use tikz to draw nice memory drawing (example in file `stack_representation_example.pgf`.). Or google draw. If you use other tools, make sure to have all figure as `pdf`. Moreover, if you use an external tool, make sure to provide a way to modify the produced images. A link to google draw with correct permission is good enough.

```
1  import numpy as np
2
3  def incmatrix(genl1,genl2):
4      m = len(genl1)
5      n = len(genl2)
6      M = None #to become the incidence matrix
7      VT = np.zeros((n*m,1), int)  #dummy variable
8
9      #compute the bitwise xor matrix
10     M1 = bitxormatrix(genl1)
11     M2 = np.triu(bitxormatrix(genl2),1)
```

Listing 1: Example from external file