CYBR 470/570 Reverse Engineering

Lab 5, Dynamic program analysis - Due, May 7, 2025

May 1, 2025

Dynamic program analysis

Today we'll use our new found `r2` skills to make programs do what we want while debugging. The goal is **NOT** to rewrite the binary, but to run it and dynamically change its data.

Part 1 – Get past the errors

First you need to dynamically analyze 2 binaries and edit them at runtime to get to the section of code that signifies "success." The two binaries you'll examine are `crackme`, and `good_job`.

For both binaries, make a step-by-step guide (be sure to include the instructions you run) that can be used to get to the section of code requested. Your input **must be** 123456 for both binaries, but you **must** change program values at runtime such that you still reach the "success" state.

Part 2 – What's wrong?

In this task you will analyze the binary `crackmetoo` and figure out what is wrong with it. Start by documenting what string the `encrypt_password` function creates for two different runs of the program. You should be able to direct the output to a file easily. For example, if you're at the correct memory location , you can use the command:

```
px 64 > output.txt
```

This should lead you down the right path to finding the first issue with this code.

Another error lurks in the code unrelated to the encrypt function. Find it and document why it would be an issue even if the `encrypt_password` function were correct.

In your write up, document what each error is and how you discovered it.

Once you discover the errors, find out if you can you call the program from the command line with any input and get it to succeed (without editing the binary)? If so, does the same input always work? Why or why not? Add to your write up your findings and include in your tarball any code you used to test the program.

Turn in

Create a tarball with the following:

1. **All programs needed to follow your guide**. This includes the binaries provided.
2. A PDF documenting the your process.

Reminder: a step-by-step guide should be more than just a paragraph describing roughly what to do. Your audience is someone who just knows how to click and follow directions (an average first year undergrad). You need both the **HOW** and the **WHY** in your guide for each step. You can assume your audience has `r2` installed.

Please use listings and verbatim over screenshots. If you have questions on how to do this most easily I can assist you.