University of Pittsburgh | School of Computing and Information

Week 7

Lab 4 (Part 1): Defusing Traps

CS 449 Spring 2020

Part 1: Overview

- You are given 2 executable programs (Trap 1 and Trap 2) and you have to crack them by uncovering a valid password
- Each Trap takes user input and determines if that is a correct password or not
  - But we don't have the requirements for a valid password, so we have to look through the program to find them
  - There can be more than one solution!!
- Normal approach: Open the source (.c) file and look at the code
- Issue: We don't have access to the source code!
  - But we do have access to the executables themselves…
- Solution: Look through the assembly code to figure out what the code is doing and uncover the requirements for a valid password!

Trap 1

```
int trap1(int input)
```

Expects one integer as input (what the user enters)

Trap 1 will compare this input to the password requirements

Trap 2

```
int trap2(int arg1, int arg2)
```

Expects two integers as input (what the user enters)

Trap 2 will compare these inputs to the password requirements

What do I do?

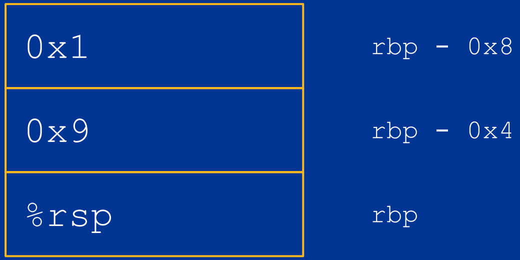
Disassemble the assembly!

1. Run GDB
   a. `$ gdb ./trap1`
2. Disassemble:
   a. `(gdb) disas trap1`
3. Try to understand the assembly code
4. Set a breakpoint at trap1
   a. `(gdb) b trap1`
5. Run until breakpoint
   a. `(gdb) r`
6. Print register contents
   a. `(gdb) p <register>`
7. Move to the next instruction
   a. `(gdb) ni`
8. Continue until next breakpoint or finished running
   a. `(gdb) c`

Keep printing register contents as you step through code!

Tips

1. Look for comparison instructions (e.g. `cmpl`)
    a. Usually followed by a jump (e.g. `jne`)
    b. Note: In these executables, it will not be the case that user input will be compared directly with the password
        i. NOT `cmp <answer>, <input>`
        ii. Instead, the programs will check if user input meets a certain condition
            1. This allows for multiple correct passwords!
        iii. All you have to do is figure out the requirements
2. Draw the stack as it grows and changes



Demo

Let's start looking at `trap 1` together...