CS 0449 Project 2 Write Up

Jason Walker
jaw280@pitt.edu

1. **Executable 1**
    a. Unlocked with passphrase WXohtRQUREtWDWaBvsfDXOf
    b. For this program, the first thing I did was disassemble the program and look at the assembly. However, after seeing a very short main function, I figured that I might as well try and use the other tools I had at hand before diving into the code. I used the mystrings program I had just written and ran it on jaw280_1. I looked through to see if I could find anything worthwhile. Most of the strings did not mean anything. But I found the place where it said "Sorry! Not correct!" and "Congratulations!" Above this was the string I unlocked the program with. I reasoned that in the program, the string that was used to unlock was probably placed near the ending messages because it had to be compared to the input and then determined if it was correct or not.
2. **Executable 2**
    a. Unlocked with passphrase jaw280
    b. For this program, I tried the same thing that I did with the first program and looked at the mystrings output. Of course, the same did not work this time so I decided to disassemble the main function and look further into it. The first thing I looked at were all the function calls. I saw some functions with short names such as <u> and <s> and stepped into them thinking that they were going to create the string somehow. However, I realized that I should be looking further down to the stdio functions. I looked at strcmp and the steps around it. I saw that before, things were being stored into the eax register as well as an immediate being passed in. I used the x/s command to read each of them at a breakpoint before strcmp was called and saw that one of them was my input and one of them was jaw280. I assumed that is what I needed to enter.
3. **Executable 3**
    a. Unable to unlock
    b. For this program, the first thing I did was try and disassemble main. This didn't work so I assumed that this program did not have a main function. In order to figure out where it started, I used the objdump command and saw that it had a start address at 0x08048370. I placed a breakpoint there and began to disassemble the code using $pc. Since most of the functions were unnamed, I placed breakpoint after breakpoint throughout the function and saw a few things. I realized that the string had to be 16 characters long and any input over 16 characters will be ignored. Further, there is a section with alternating compare and jump operations. The comparisons are taking whatever is in the $eax register and comparing it to an immediate one-byte value. Even though I was not able to figure out what the password was, I know the password is dynamic because of how many comparisons are made in the assembly. I also noticed that before the comparisons are made, the values are 0-extended when moved. The values that are being compared are one-bye values that most likely stand for the ascii characters <, >, &, etc. After stepping through the program one line at a time and getting nowhere, I tried other commands such as hexdump and strings to no avail.