Brendan Drusda

CS449

Project 2 Write-Up

Program 1

In attempting to discover the password to the first program, I simply followed the general guideline provided to us in lab. First, I placed a breakpoint in the main method and ran the program. I then disassembled the program and took note of each of the method calls. I disassembled the only non-library method, chomp, and examined its code. I noted that edx was the input and that there was a byte comparison being done 0xFFFFFFFF times or until a zero flag was reached, presumably being the null terminator at the end of the input string. There did not appear to be any manipulation of the input, so I returned to the main method. At this point, I noticed that a value was moved into esi and another string comparison(repz cmpsb) was done with edi, our input. So, I set a breakpoint after the move and checked the contents of esi, which provided the string "xAuBqsstzHRhYYyFrPHSD". I attempted to use this string as my input on the next iteration and found that it was, in fact, the correct password for the program. Upon further inspection, mystrings revealed that the password was explicitly written in the code.

Program 2

I approached the second program in the same general manner as the first, placing a break at main and disassembling the code. I noticed that there were more unknown methods this time around: r, c, and s. After examining these methods, I took note of a string comparison between esi and ebx. Placing a breakpoint before the comparison, I checked the values of the common registers. eax contained an _2, which I recognized as the last two characters of the executable name and ebx stored the user input. esi, being the other half of the comparison, was equally important, so I checked it and noticed it was the full name of the executable file. Being that ebx and eax would need to combine to form esi, and that eax was the last two characters, I realized that the user input needed to be the rest of the executable name. I applied the concept, inputting "bmd66" and the password was accepted. After examining the objdump code, as was expected, the password was not hardcoded into the program.

Program 3

I attempted to set a breakpoint in main, as I did with the other programs, however there was not a main in the program. This lead me to recognize that the program was dynamically loaded. However, before I noticed this, I attempted to disassemble the code, entering "disas" twice during runtime, which, to my surprise, was a suitable password for the program. After struggling to figure out how to examine the code without being able to disassemble it, I remembered that use of objdump was recommended in the project description. I read up on the process and viewed the code in a text file. Here I noticed that there was a set of comparisons between eax and immediate values. I made note of these and recorded their ASCII values(although, I did not immediately realize that they were written in hex rather than decimal, and struggled to understand the correlation between them and the input). I then recognized that the program required ten characters of input and that exactly three of them needed to match. I tested various input strings making use of combinations of the input characters(c, s, C, S, 4, 9). As I thought, any combination of exactly three of these characters in a ten character input string resulted in a successful password(cs4abdefgh, 1234abcs08, 9990000000, iiiiiiCS4, etc.).

The majority of the learning I did in this project came from this program. In finding the password I needed to learn to use objdump. In addition, I had to learn to understand the dynamically loaded code and follow the numerous jumps throughout execution.