Matthew Strayhorn

mas583@pitt.edu

November 1, 2015

CS 0449 Project 2: What's the Password?

Executable 1: mas583_1

The passphrase is "AcHiisItrDvSldqbmmPJG".

I found the passphrase by using my mystrings program to print all the printable strings in the executable. I saw the above string right before the strings "Sorry not correct!" and "Congratulations unlocked with passphrase %s" and thought that it seemed suspicious so I wrote it down and tried it out. It worked so I tried it again to make sure that the correct passphrase didn't change on each run of the program.

Executable 2: mas583_2

The passphrase can be "matthewwehttam" or any other string entered forwards and backwards that is more than nine characters total

This one was a lot more difficult that the first executable and required me to examine the x86 assembly in gdb. I first used the mystrings program to look for any interesting strings and there were none so I then moved on to gdb. My second attempt involved spending a lot of time examining the x86 assembly of the functions main, c, s, and p. I realized that function s counted the number of characters that I typed minus the newline character. I saw two cmp functions in main that were interesting, one where there was a comparison between %eax and 0 and the other between %eax and 9. I examined the first comparison to 0 that happens right after a call to function p so I checked p and saw that there is a je instruction at 0x080484c0 where if it isn't taken the next instruction will set %eax to 0. In main when %eax is 0 at that comparison it jumps to main +102 where it outputs "sorry not correct". I then tried entering 1 character so that in p %eax would be set to 1 in p so that the comparison to 0 in main wouldn't result in the jump being taken but the second comparison between %eax and 9 made me realize that I had to enter more than nine characters in order for that jump to not be taken and instead output that I had the correct passphrase. So from that I knew that I had to enter a passphrase that would result in the jump at 0x080484c0 in p being taken and I saw that there was a comparison to %edi which held the last character in the input and %al and saw that I simply had to enter the same word forward and backwards for the jump to be taken and increment %ebx for the cmp instructions in function p.

Executable 3: mas583_3

This goal of disassembling this executable was to find the bug in it. I first immediately noticed that when I tried to create a breakpoint in main there was no main function so I figured that it had been removed so I would have to trace the program by doing the objdump –D command to follow the full assembly and find where main was. I figured that the main function started at 0x80483a0 and started tracing there. I think that the bug lies in a loop around instructions 0x804846b and 0x804847f because that is where it prompts me to enter a character however I noticed that it would not prompt me to enter a character each iteration through that loop. I believe that the program intended for me to enter 10 characters because I saw an interesting cmp instruction at 0x80484b9 where %eax which equals 10 at this point is compared to %ebp. However instead of entering 10 characters, I can only enter 5.