Christian Mananghaya (cam314)

Jarrett Billingsley

CS 0449 (Tues/Thurs 1:00–2:15pm)

9 March 2018

Write-up for Project 2

File #1: cam314\_1 (answer: YQUQidiZSiHyHlRtfrHAINADNNmtz)

For this file, I started by running the executable file through my mystrings.c file. It did not provide any clues to the password. Next, I compared the outputs of strings and mystrings. It produced a lot of characters that I did not understand, so I went to look at the mystrings.out file that was created after the comparison. There, I found the passphrase “YQUQidiZSiHyHlRtfrHAINADNNmtz”. I passed this through the debugger and it worked!

File #2: cam314\_2 (answer: 64.121.138.80)

For this file, I started by comparing the outputs of strings and mystrings. The diff command did not print anything, so therefore the files are identical. Like the first one, I took a look at the mystrings.out file that was created, and found the passphrase “do this project on thoth, cmon, stop being afraid of thoth”. Unfortunately, that did not work, so I went looking in the gdb for more investigation. I decided to put a break point at the main and disassemble the code to see what I could find there. I began looking through the disassembled versions of the functions called in the main(), such as sc(), u(), and s(). I put break points on all of them and looked at their assembly code after each function. One specific function that caught my eye was c(), because right before c()’s break point, the executable prompted me for the password. This had me thinking that the c() function served as the comparison between the input and the actual password. From there, I used the ‘x’ command to look at the values at every register when c() is called. Those numbers didn’t make any sense, so I decided to use the ‘x/s’ command on the addresses of the previously mentioned registers. Low and behold, the registers $eax and $ebx held whatever I typed in as input and its respective comparison to the string “64.121.138.80 59155 22”. I was originally going to try this string as a potential password, but I noticed in the c() function call that register $ebx was subtracted by register $ebi, so I removed the last 7 digits. After testing the string “64.121.138.80”, the “Congratulations” text appeared. Noticing that this set of numbers seemed familiar, it turned out to be my current IP address. I infer that the actual password is not my specific IP address, but whatever IP address is traced back when the executable is ran.

File #3: cam314\_3 (answer: ???)

This one was the hardest to solve. I started by using mystrings.c to look at string outputs. The string “[^\_]” appeared about the text “Congratulations”, so I tried that as a potential password, but that did not work. Next, I compared the outputs of strings and mystrings. The diff comparison showed no difference in characters, so the outputs were identical. Looking at both .out files did not help either, as they produced the same strings given earlier. After performing an ‘objdump’ on the executable, I found out that there was no main(). I had trouble utilizing the gdb for this executable, as there were no identifiable functions to place a break point at. As a last resort, seeing that the password can take up to four inputs and the ‘objdump’ produced values in sets of four, I tested some of those sets of values – all of which to no avail. Based on the formatting of the second executable, I infer that the password is stored in register $ebx. I don’t know how I would access it without a main(), but my best guess is that it would be through <.text>.