Project 2

Program 3

Andrew Beers acb75@pitt.edu

Procedure:

Even though this program was written in C, and I know it has a main function, I could not initially break here because the libraries are all being dynamically loaded at runtime. This caused for concern as there was no longer an initial place to stop the program for debugging. I came around to a solution to use disas from a starting register to an ending register. I based the starting register on the starting registers from the first two parts of this assignment, and made my ending register arbitrarily large compared to it. This reveal a lot of assembly code, most of which appeared to me as book keeping and managing the stack. I noticed some interesting calls include imul and xor so I looked into them. The imul calls seemed to use hex values of 0x72 which is the character r, 0x74 which is the character t, and other hex values relating to letters ‘q’,’w’,’e’, and ‘y’. By observation I noticed that all of these letters were the letters that spell qwerty.

Solution:

After finding the hex character of 0x72 I decided to just try ‘r’ as my passcode guess and it unlocked the passcode. I then proceeded to try the other letters in qwerty and noticed that the pass code unlocked for each of them. To my surprise entering the string qwerty did not unlock the pass code. From here on out I was puzzled so I attempted the following guesses with the results posed next to them

* e – passcode unlocked
* abcdfghijklmnopqrstuvwxyz – pass code failed
* abcdefghiklmnopqrstuvwxyz –pass code unlocked
* E – passcode unlocked
* Afafafafafafafafe – passcode failed
* Qq – pass code failed
* 0123456789e – pass code failed
* 123456789e – pass code unlocked
* 123456789q – pass code unlocked
* 123456789Q – pass code unlocked
* q e – pass code failed

My conclusion after much testing is that the program takes the first 10 characters given to it, either by continuously prompting until there are 10 characters or chomping off the first 10 characters if there are more and will test this string for the following things.

* A letter in the 10 characters matching a letter contained in “qwerty”
* It is not case sensitive
* If you repeat any letter in qwerty (i.e qq or qw or qW or q w) the pass code will fail if the repeated letter is in the first 10 characters of the string (this is why the entire alphabet passes)

Notes:

This program dynamically loads the libraries so breaking main wasn’t an option. Breaking from different endpoints of address values seemed to be somewhat valuable as I obtained information about individual characters. However personally I learned more about how this program works by testing with inputs.

Pass phrase – Any string of 10 characters with exactly one of those characters being ‘q’, ’w’, ’e’, ’r’, ’t’, or ‘y’.