Project 2 Report

Executable 1:

For this program, I immediately started using the debugger to find the password. It was very similar to what we did in recitation. At first, I put a breakpoint in main. Then, I looked at the assembly and saw "chomp" so thought to put a breakpoint there. I examined the registers there and noticed that the string I had entered was being store in $edi. Then, I noticed that $edi and $esi were being used in the repz instruction. Thinking this might be significant, I set a breakpoint at the following instruction:

```
repz cmpsb %es:(%edi),%ds:(%esi)
```

I looked at $edi and $esi and noticed that a very long, random string was being stored in $esi, so I ran the program again, entered that string, and it worked! The passphrase that worked was "tXmqssaoAiaKjCIJYJNawgq". Actually, I was curious about what would happen if I used this executable as an input for my mystrings program. I looked at the output and noticed that the passphrase I discovered was actually in the output of the program.

Passcode to unlock twm17_1: tXmqssaoAiaKjCIJYJNawgq

Executable 2:

Since in the last program, I realized that mystrings could be used to show the output of the executable, I used it to try to find the output. However, in this case, the output was quite little and the passphrase was not in the output. Then, I moved to the debugger to find the passphrase. I placed a breakpoint at main and looked at the disassembled code. I noticed some nonlibrary functions that seemed important, particularly <c>, <s>, and <p>. I put breakpoints at each of them and stepped through the disassembled code. I reached <c> and stepped through the instructions using the "stepi" command. In <c>, a function "s" was being called so I placed a breakpoint there. So, I stepped through the "s" function. In this function, I looked at the value of $edx and saw that it was holding "twesha\n", which was the string I entered appended by a new line character. Something else I noticed while examining the "s" function was the register $eax. I noticed that it was initialized to 0 in the beginning of the function and 1 was being added to it in a loop. So, I assumed $eax was a counter variable. The following instruction in "s" caught my attention:

```
cmpb   $0x0,(%edx,%eax,1)
```

I did some research and looked at the X86 opcode cmpb and realized that this instruction takes a byte that $edx + $edx point to and compares it to 0, which is the ASCII code for NULL. If it was not 0, it was incrementing $eax. So, quickly I realized that "s" was counting the length of the string. The length of the string was in $eax, which was the counter variable. Then, back in "c", I noticed another cmpb instruction and saw that it was checking to see if the last character of the string was equal to the new line symbol. Next, I was in the function "p" so I went on to examine the assembly code of this function. There was a movzbl instruction in this function. I could see that it was loading the first byte of the string that I input into $eax. The next instruction was cmp (%edi),%al. I placed a breakpoint there and examined the values. $edi contained "a" and $al contained 0x74, which is the ascii code for "t", the first letter in my input string. Again, I noticed there was a loop here and I realized this function was checking whether the input string is symmetric. Back in the main function, there was another cmp instruction after <p> that I looked at. It was comparing $eax, which held the length of the input string and 0xd, which is 13 in hexadecimal. So, it seemed like since the program is checking whether the input string is symmetric and whether it's length is greater than 13, I ran my program again and entered the string "rats live on no evil star" because the phrase is symmetric and it is longer than 13 characters. This phrase unlocked the program! I also tried phrases like "bbbbbbbbccccbbbbbbbb" and "zzzzzzzzzzzzzzzz" which also unlocked the program.

Passphrase pattern for twm17_2: any symmetric string with more than 13 characters

Executable 3:

Again, with this program, I ran mystrings again to see if the passphrase was in the output of the program. However, again, similar to Program 2, the output was very small and the passphrase was not in the output. Then, I tried to run the debugger and set a break point at main, but when I tried to do that, the debugger told me there was no main function defined in the program. So, then I tried the objdump –D twm17_3 like the project description said. This output all of the assembly code. I tried to understand, what was happening, but was pretty confused about what the code was doing. I looked at the .text section of the object dump and set a breakpoint at some instructions where I saw cmp because I was hoping to see what was happening to my string. However, whenever I ran the disas command in the debugger, I kept getting the following: "No function contains program counter for selected frame". I tried to play around with the assembly code for a while, but unfortunately did not have enough time to keep digging due to other projects and exams. This executable was definitely the hardest to crack. I think it was because some functions were being dynamically loaded. Also, since I wasn't able to place a breakpoint in the main function, I couldn't entirely figure out what was happening in this one.