Jamie Min

Project 2

Project File: jam433_1

Password: wMNqrFsszMPxSPkRYUexakR

Approach:

I placed a breakpoint at main and dumped the assembly file. There is no **strcmp** instruction, but there was an **fgets** and **chomp**. So, it's still taking user input and chomping the newline. Different from the lab, I noticed an **repz cmpsb** instruction that might be useful to checkout. The **cmpsb** which after a quick Google search means its comparing string operands.

```
0x0804830c <+61>:    repz cmpsb %es:(%edi),%ds:(%esi)
(gdb) x/s $edi
0xffffd08c:      "something"
(gdb) x/s $esi
0x80b388c <__dso_handle+4>:      "wMNqrFsszMPxSPkRYUexakR"
```

The **$esi** register looked promising and unlocked the program.

Project File: jam433_2

Password: jam433

Approach:

I placed a breakpoint at main and dumped the assembly file. Similar to the lab, I found a **strcmp** function and set an additional breakpoint there. I moved the program pointer to the new breakpoint and examined the relevant registers. There are two **mov** instructions preceding, perhaps those registers are what is being compared.

```
Breakpoint 2 at 0x80485db
(gdb) c
Continuing.
asd
Breakpoint 2, 0x080485db in main ()
(gdb) x/s $eax
0xffffd02b:      "_2"
(gdb) x/s $esi
0xffffd37e:      "jam433_2"
(gdb) x/s $esp
0xffffd010:      "(\320\377\377~\323\377\377@\004", <incomplete sequence \325>
(gdb) x/s $ebx
0xffffd028:      "asd_2"
```

The string jam433_2 looks promising but after trying it did not work. Looking at the above registers, it seems like it is mutating my string and adding "_2" at the end of it. So logically, it seems if I enter "jam433", it will append the "_2" thus triggering the correct string comparison to unlock the password.

Project File: jam433_3

Password: Any combination of the letters “CScs449” of length nine, or any combination of the aforementioned letters with a random char preceding it (ex. xCSCS44949494999cscs).

Approach:

This one was very difficult to figure out. Initially I tried to set a breakpoint at main but could not as the program was stripped. The symbol table was not included in the executable. With the hint of using **objdump** and some Google-fu, I dumped the assembly file using **objdump -d .text jam433_3** and examined the relevant information. I see that the program asks user for a char so I inferred the password is some series of chars. After some trial and error, I realized the program needs to be of at least length nine as the newline carrier just continues on to the next line.

After some trial and error, the password “sssssssss” worked for some reason but “sssssssss” of at length 10 didn’t work. Surprisingly, “xsssssssss” and any length of “s” with any random char preceding it worked as well (i.e. “xssssssssssssssss”) as it just shortens it to length 10. I can definitely use this to find something worthwhile in the assembly dump.

The cmp instruction is comparing the register to some hexadecimal values. Using **man ascii**, I figured out the compared ascii values to be “CScs449”. There is the s that originally worked, and after I matched the C and the 4, the answer became obvious.

```
8048470:       83 f8 43                cmp    $0x43,%eax
8048473:       74 20                   je     8048495 <puts@plt+0x141>
8048475:       83 f8 43                cmp    $0x43,%eax
8048478:       7f 0c                   jg     8048486 <puts@plt+0x132>
804847a:       83 f8 34                cmp    $0x34,%eax
804847d:       74 16                   je     8048495 <puts@plt+0x141>
804847f:       83 f8 39                cmp    $0x39,%eax
8048482:       74 11                   je     8048495 <puts@plt+0x141>
8048484:       eb 14                   jmp    804849a <puts@plt+0x146>
8048486:       83 f8 63                cmp    $0x63,%eax
8048489:       74 0a                   je     8048495 <puts@plt+0x141>
804848b:       83 f8 73                cmp    $0x73,%eax
804848e:       74 05                   je     8048495 <puts@plt+0x141>
8048490:       83 f8 53                cmp    $0x53,%eax
8048493:       75 05                   jne    804849a <puts@plt+0x146>
8048495:       83 45 f0 01             addl   $0x1,-0x10(%ebp)
```