Tianyang Chen
Tic27@pitt.edu
3/1/2015

# CS449 Project 2 report

The project involves reverse engineering three executables that run on thot.cs.pitt.edu. Based on how a program works and various references for X86 assembly language online, passwords could be found easily. This report documents steps necessary to analyze what these programs do and what passwords are.

**Part A**
The general steps are basically the same as what we used in the Lab. First, set a break point at main and use command "disas" to see the assembly code. Then, find out what functions this program calls. It is very straight forward that this program uses <fgets> to take user input, <chomp> to process something, and <printf> to print success or failure message. Therefore, the first attempt was to print out the parameters used in <printf>. The first parameter was "Congratulations!\nUnlocked with passphrase %s\n". Then, by going back to the line that calls it, the password checking logic was found. "repz cmpsb %es:(%edi),%ds:(%esi)" is doing comparision on 0x1a characters. Therefore, $edi has user input and $es has the password.

Password: qZIOexERTMVVinrmIerCbxwha

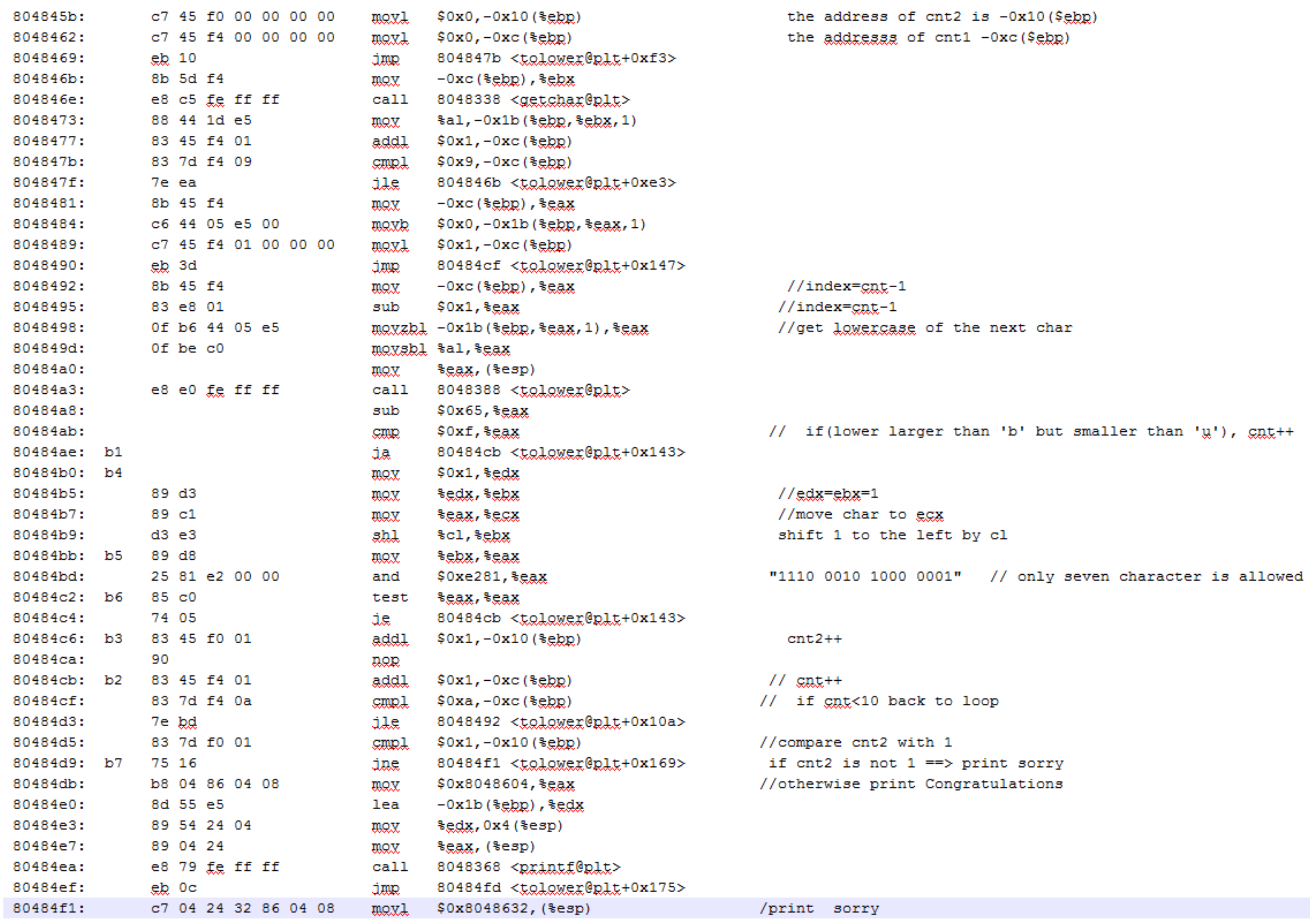
**Part B**
Following the same steps, the assembly listing shows that the program calls many functions to do processing.  The function <s> was called multiple times to check the length of user input including the carriage return at the end. At some point, back to function <c>, it gets rid of the last carriage return. By calling function <r>, the program checks if the user input is symmetric or not using a For loop. The last part of function <c>, however, checks if the length is larger than 10 characters, excluding the carriage return. If the user input is both symmetric and longer than 10 characters, it will unlock the program.

**Part C**
As odd as it looks, the program doesn't have a main function. Based on our TA Qihang's hint, the objdump could be used to get the assembly listing of the program.
This program reads 10 characters and checks if all inputs are **larger** than 'd' and **smaller** than 'u'. If so, it counts how many characters are. If there are more than **seven** characters in the range, it fails.

Important logic:

1) sub    $0x65,%eax
   cmp    $0xf,%eax              //  if(lower larger than 'b' but smaller than 'u'), cnt++
   ja     80484cb <tolower@plt+0x143>    //ja performs unsigned comparisons. Therefore, anything **smaller** than 'e' will cause the jump as well as anything **larger** than 't'.
2) and    $0xe281,%eax                                "1110 0010 1000 0001"   // only seven character is allowed.

In conclusion, the password could be anything that has seven or less characters that are in the range. The following is the translated C program.



```
804845b:        c7 45 f0 00 00 00 00    movl   $0x0,-0x10(%ebp)              the address of cnt2 is -0x10($ebp)
8048462:        c7 45 f4 00 00 00 00    movl   $0x0,-0xc(%ebp)               the addresss of cnt1 -0xc($ebp)
8048469:        eb 10                   jmp    804847b <tolower@plt+0xf3>
804846b:        8b 5d f4                mov    -0xc(%ebp),%ebx
804846e:        e8 c5 fe ff ff          call   8048338 <getchar@plt>
8048473:        88 44 1d e5             mov    %al,-0x1b(%ebp,%ebx,1)
8048477:        83 45 f4 01             addl   $0x1,-0xc(%ebp)
804847b:        83 7d f4 09             cmpl   $0x9,-0xc(%ebp)
804847f:        7e ea                   jle    804846b <tolower@plt+0xe3>
8048481:        8b 45 f4                mov    -0xc(%ebp),%eax
8048484:        c6 44 05 e5 00          movb   $0x0,-0x1b(%ebp,%eax,1)
8048489:        c7 45 f4 01 00 00 00    movl   $0x1,-0xc(%ebp)
8048490:        eb 3d                   jmp    80484cf <tolower@plt+0x147>
8048492:        8b 45 f4                mov    -0xc(%ebp),%eax               //index=cnt-1
8048495:        83 e8 01                sub    $0x1,%eax                    //index=cnt-1
8048498:        0f b6 44 05 e5          movzbl -0x1b(%ebp,%eax,1),%eax      //get lowercase of the next char
804849d:        0f be c0                movsbl %al,%eax
80484a0:                                mov    %eax,(%esp)
80484a3:        e8 e0 fe ff ff          call   8048388 <tolower@plt>
80484a8:                                sub    $0x65,%eax
80484ab:                                cmp    $0xf,%eax                   //  if(lower larger than 'b' but smaller than 'u'), cnt++
80484ae:   b1                           ja     80484cb <tolower@plt+0x143>
80484b0:   b4                           mov    $0x1,%edx
80484b5:        89 d3                   mov    %edx,%ebx                    //edx=ebx=1
80484b7:        89 c1                   mov    %eax,%ecx                    //move char to ecx
80484b9:        d3 e3                   shl    %cl,%ebx                     shift 1 to the left by cl
80484bb:   b5   89 d8                   mov    %ebx,%eax
80484bd:        25 81 e2 00 00          and    $0xe281,%eax                "1110 0010 1000 0001"   // only seven character is allowed
80484c2:   b6   85 c0                   test   %eax,%eax
80484c4:        74 05                   je     80484cb <tolower@plt+0x143>
80484c6:   b3   83 45 f0 01             addl   $0x1,-0x10(%ebp)               cnt2++
80484ca:        90                      nop
80484cb:   b2   83 45 f4 01             addl   $0x1,-0xc(%ebp)             // cnt++
80484cf:        83 7d f4 0a             cmpl   $0xa,-0xc(%ebp)            //  if cnt<10 back to loop
80484d3:        7e bd                   jle    8048492 <tolower@plt+0x10a>
80484d5:        83 7d f0 01             cmpl   $0x1,-0x10(%ebp)           //compare cnt2 with 1
80484d9:   b7   75 16                   jne    80484f1 <tolower@plt+0x169>  if cnt2 is not 1 ==> print sorry
80484db:        b8 04 86 04 08          mov    $0x8048604,%eax            //otherwise print Congratulations
80484e0:        8d 55 e5                lea    -0x1b(%ebp),%edx
80484e3:        89 54 24 04             mov    %edx,0x4(%esp)
80484e7:        89 04 24                mov    %eax,(%esp)
80484ea:        e8 79 fe ff ff          call   8048368 <printf@plt>
80484ef:        eb 0c                   jmp    80484fd <tolower@plt+0x175>
80484f1:        c7 04 24 32 86 04 08    movl   $0x8048632,(%esp)          /print  sorry
```