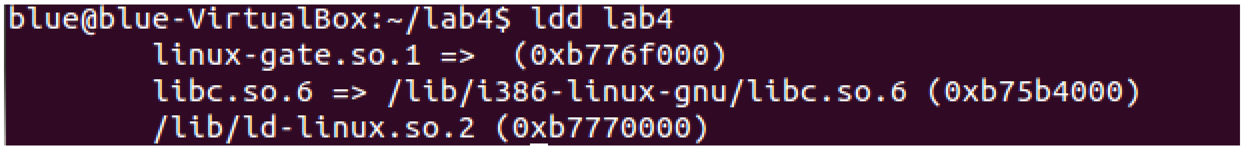
Rahul Prithu
CSCI 395 – Digital Forensics
Lab 4

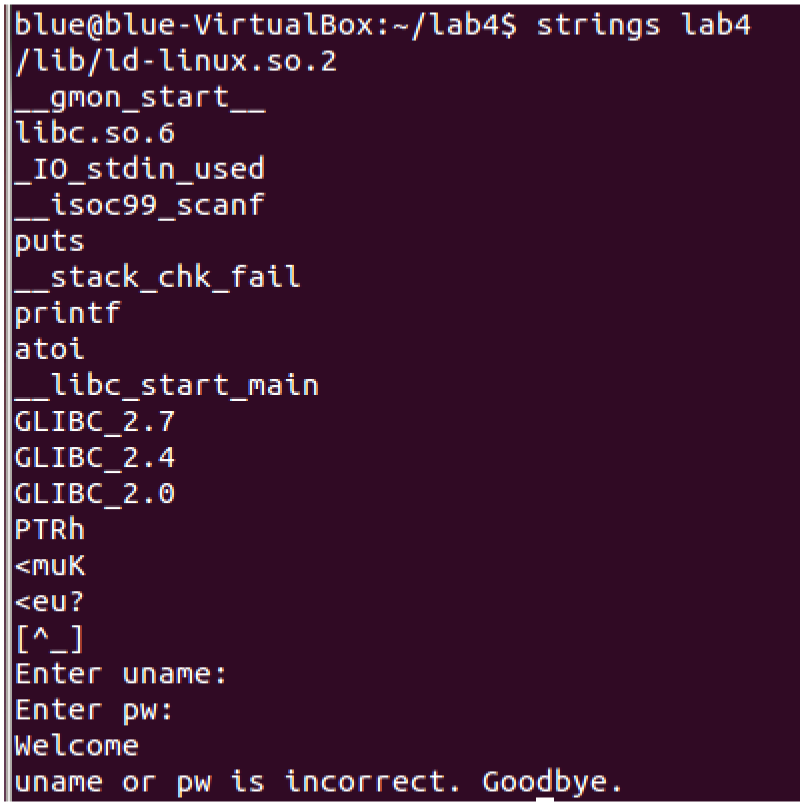
Background: A program requires a username and password to gain access. The username and password hints are in the program. We are tasked with looking at the program, finding the username and password, and modifying it to accept what it previously did not.

The ***ldd*** command is executed to see the dependency of the program:



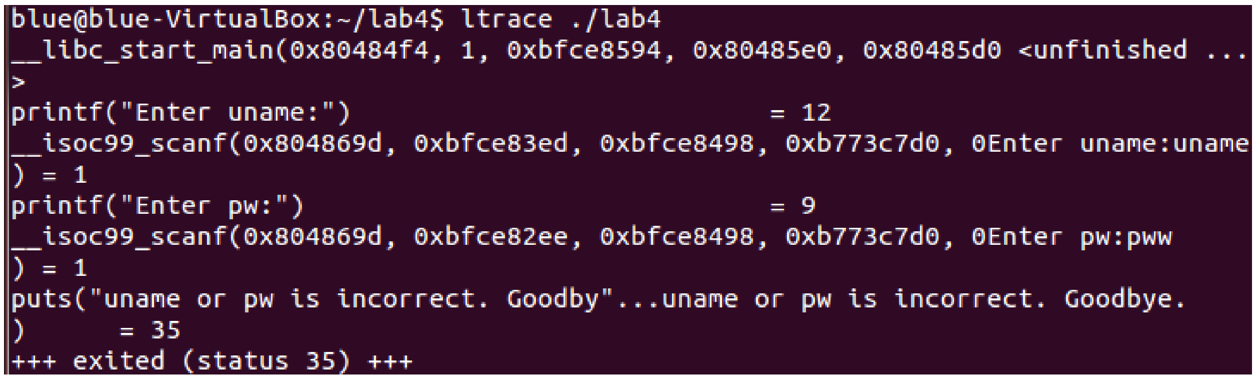
```
blue@blue-VirtualBox:~/lab4$ ldd lab4
        linux-gate.so.1 =>  (0xb776f000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb75b4000)
        /lib/ld-linux.so.2 (0xb7770000)
```

By running ***strings*** command, the identifiable strings in the program are displayed. The output also shows the important functions used in the program.



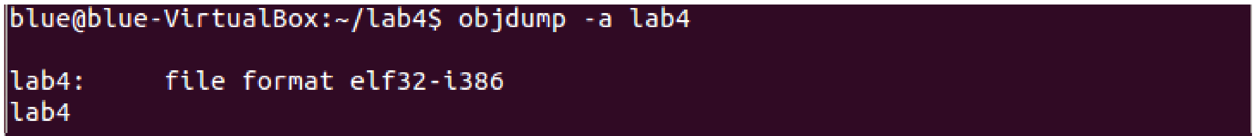
```
blue@blue-VirtualBox:~/lab4$ strings lab4
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
__isoc99_scanf
puts
__stack_chk_fail
printf
atoi
__libc_start_main
GLIBC_2.7
GLIBC_2.4
GLIBC_2.0
PTRh
<muK
<eu?
[^_]
Enter uname:
Enter pw:
Welcome
uname or pw is incorrect. Goodbye.
```

Now we execute ***ltrace*** to identify the program flow.

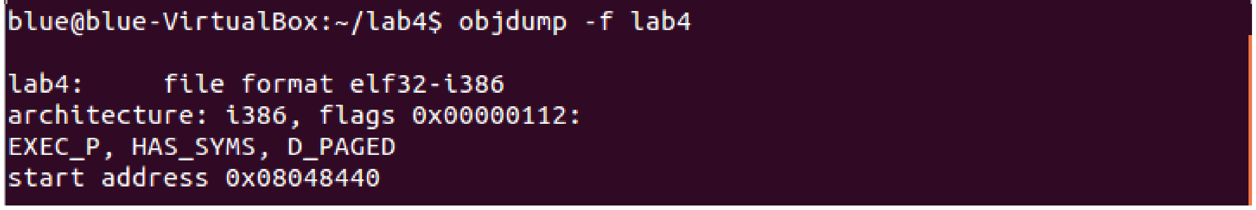


```
blue@blue-VirtualBox:~/lab4$ ltrace ./lab4
__libc_start_main(0x80484f4, 1, 0xbfce8594, 0x80485e0, 0x80485d0 <unfinished ...
>
printf("Enter uname:")                          = 12
__isoc99_scanf(0x804869d, 0xbfce83ed, 0xbfce8498, 0xb773c7d0, 0Enter uname:uname
) = 1
printf("Enter pw:")                             = 9
__isoc99_scanf(0x804869d, 0xbfce82ee, 0xbfce8498, 0xb773c7d0, 0Enter pw:pww
) = 1
puts("uname or pw is incorrect. Goodby"...uname or pw is incorrect. Goodbye.
)      = 35
+++ exited (status 35) +++
```

The file header and contents are analyzed with ***objdump***.



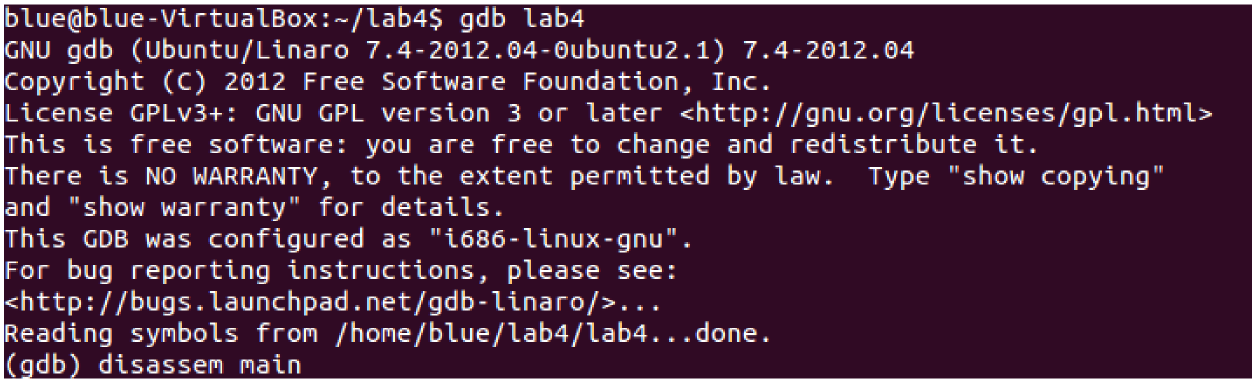
```
blue@blue-VirtualBox:~/lab4$ objdump -a lab4

lab4:     file format elf32-i386
lab4
```



```
blue@blue-VirtualBox:~/lab4$ objdump -f lab4

lab4:     file format elf32-i386
architecture: i386, flags 0x00000112:
EXEC_P, HAS_SYMS, D_PAGED
start address 0x08048440
```

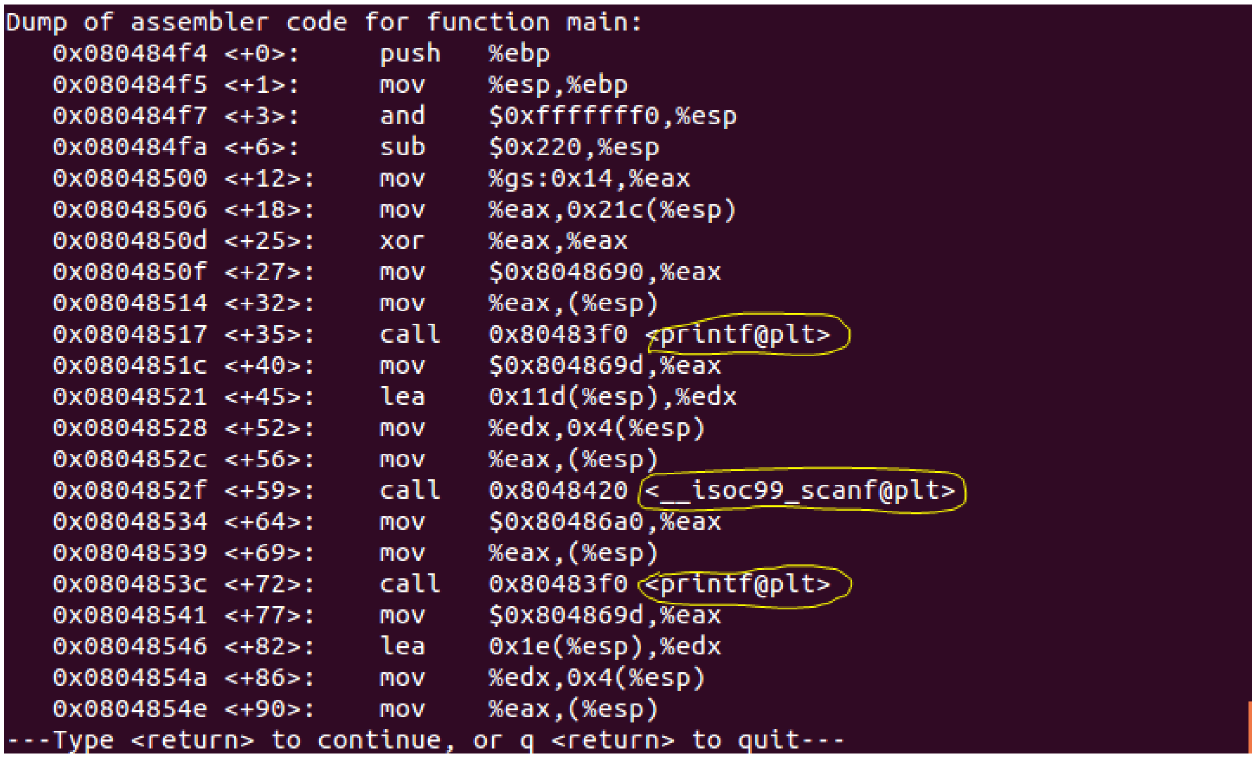
STEP 1: Analyzing the Program

To analyze the program code we disassembled it using ***gdb***

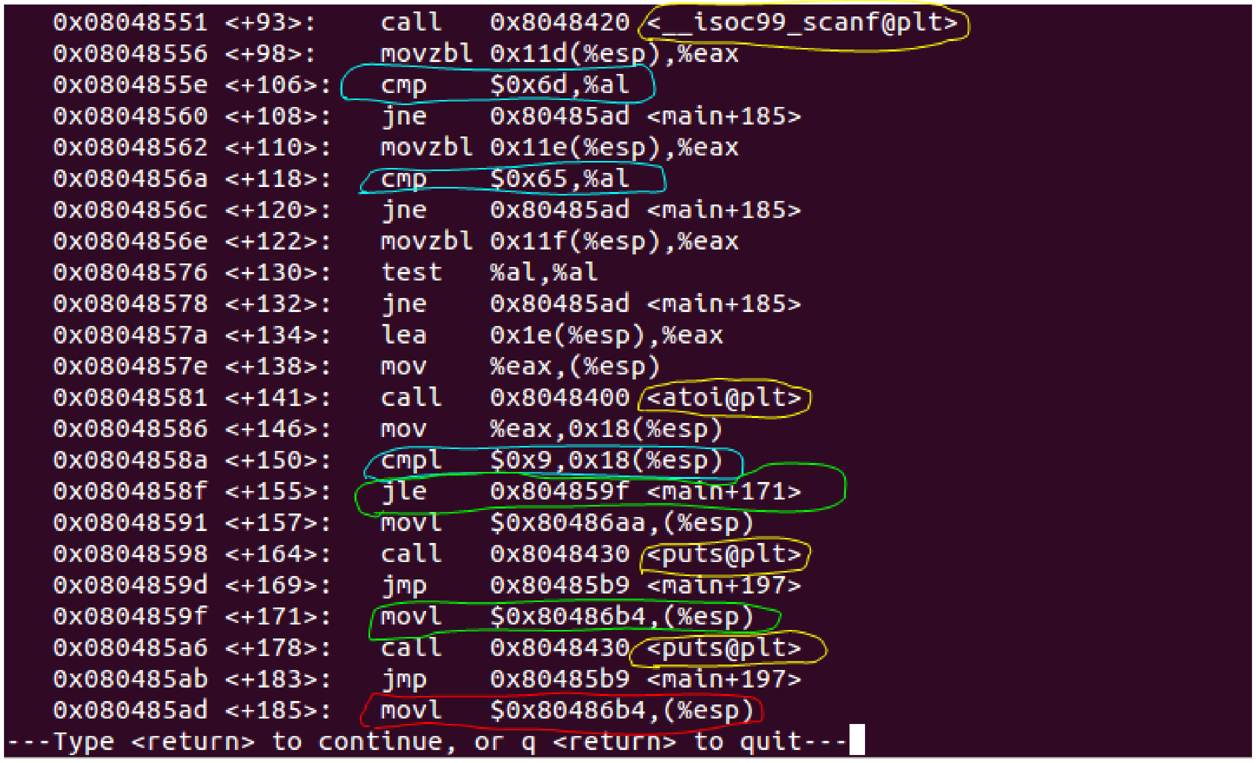


```
blue@blue-VirtualBox:~/lab4$ gdb lab4
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/blue/lab4/lab4...done.
(gdb) disassem main
```

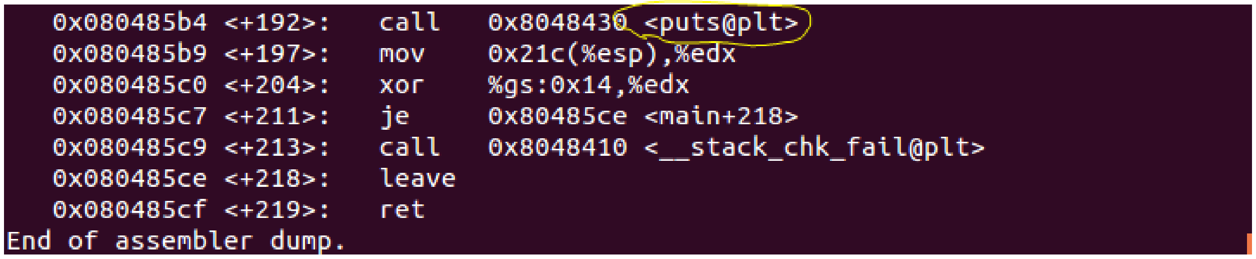
In the disassembled code, we locate the functions that we identified using the ***ltrace*** command earlier.



```
Dump of assembler code for function main:
   0x080484f4 <+0>:     push   %ebp
   0x080484f5 <+1>:     mov    %esp,%ebp
   0x080484f7 <+3>:     and    $0xfffffff0,%esp
   0x080484fa <+6>:     sub    $0x220,%esp
   0x08048500 <+12>:    mov    %gs:0x14,%eax
   0x08048506 <+18>:    mov    %eax,0x21c(%esp)
   0x0804850d <+25>:    xor    %eax,%eax
   0x0804850f <+27>:    mov    $0x8048690,%eax
   0x08048514 <+32>:    mov    %eax,(%esp)
   0x08048517 <+35>:    call   0x80483f0 <printf@plt>
   0x0804851c <+40>:    mov    $0x804869d,%eax
   0x08048521 <+45>:    lea    0x11d(%esp),%edx
   0x08048528 <+52>:    mov    %edx,0x4(%esp)
   0x0804852c <+56>:    mov    %eax,(%esp)
   0x0804852f <+59>:    call   0x8048420 <__isoc99_scanf@plt>
   0x08048534 <+64>:    mov    $0x80486a0,%eax
   0x08048539 <+69>:    mov    %eax,(%esp)
   0x0804853c <+72>:    call   0x80483f0 <printf@plt>
   0x08048541 <+77>:    mov    $0x804869d,%eax
   0x08048546 <+82>:    lea    0x1e(%esp),%edx
   0x0804854a <+86>:    mov    %edx,0x4(%esp)
   0x0804854e <+90>:    mov    %eax,(%esp)
---Type <return> to continue, or q <return> to quit---
```



```
   0x08048551 <+93>:    call   0x8048420 <__isoc99_scanf@plt>
   0x08048556 <+98>:    movzbl 0x11d(%esp),%eax
   0x0804855e <+106>:   cmp    $0x6d,%al
   0x08048560 <+108>:   jne    0x80485ad <main+185>
   0x08048562 <+110>:   movzbl 0x11e(%esp),%eax
   0x0804856a <+118>:   cmp    $0x65,%al
   0x0804856c <+120>:   jne    0x80485ad <main+185>
   0x0804856e <+122>:   movzbl 0x11f(%esp),%eax
   0x08048576 <+130>:   test   %al,%al
   0x08048578 <+132>:   jne    0x80485ad <main+185>
   0x0804857a <+134>:   lea    0x1e(%esp),%eax
   0x0804857e <+138>:   mov    %eax,(%esp)
   0x08048581 <+141>:   call   0x8048400 <atoi@plt>
   0x08048586 <+146>:   mov    %eax,0x18(%esp)
   0x0804858a <+150>:   cmpl   $0x9,0x18(%esp)
   0x0804858f <+155>:   jle    0x804859f <main+171>
   0x08048591 <+157>:   movl   $0x80486aa,(%esp)
   0x08048598 <+164>:   call   0x8048430 <puts@plt>
   0x0804859d <+169>:   jmp    0x80485b9 <main+197>
   0x0804859f <+171>:   movl   $0x80486b4,(%esp)
   0x080485a6 <+178>:   call   0x8048430 <puts@plt>
   0x080485ab <+183>:   jmp    0x80485b9 <main+197>
   0x080485ad <+185>:   movl   $0x80486b4,(%esp)
---Type <return> to continue, or q <return> to quit---
```



```
   0x080485b4 <+192>:   call   0x8048430 <puts@plt>
   0x080485b9 <+197>:   mov    0x21c(%esp),%edx
   0x080485c0 <+204>:   xor    %gs:0x14,%edx
   0x080485c7 <+211>:   je     0x80485ce <main+218>
   0x080485c9 <+213>:   call   0x8048410 <__stack_chk_fail@plt>
   0x080485ce <+218>:   leave
   0x080485cf <+219>:   ret
End of assembler dump.
```

From the assembly code we observe that when checks fail, the instruction at +185 is executed. However, if all conditions are met, instruction at +171 is executed.

The instructions at address +106 and +118, compares the inputted username with two characters:
0x6d = m
0x65 = e

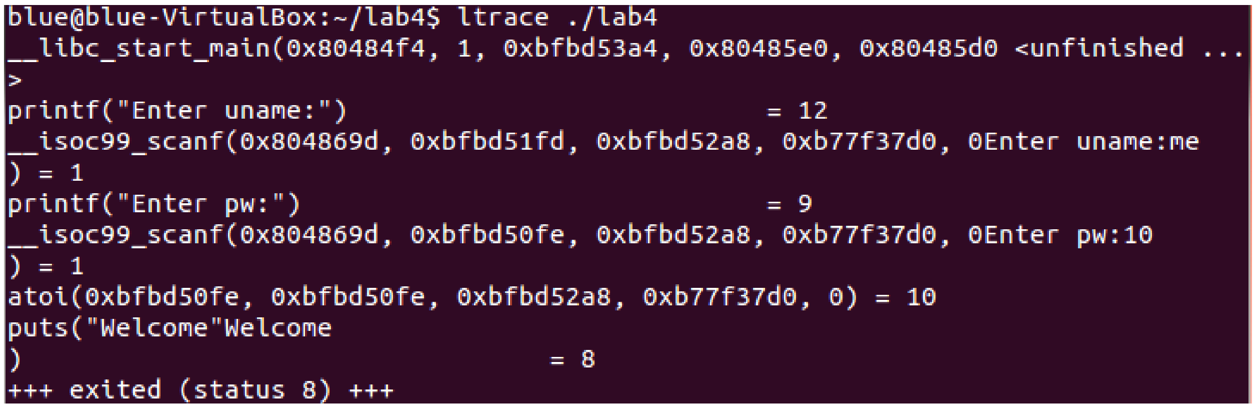
This gives the acceptable username, "*me*".

Instruction at +141 converts the password from ASCII to Integer. It indicates that the password to be used is possibly an integer number.

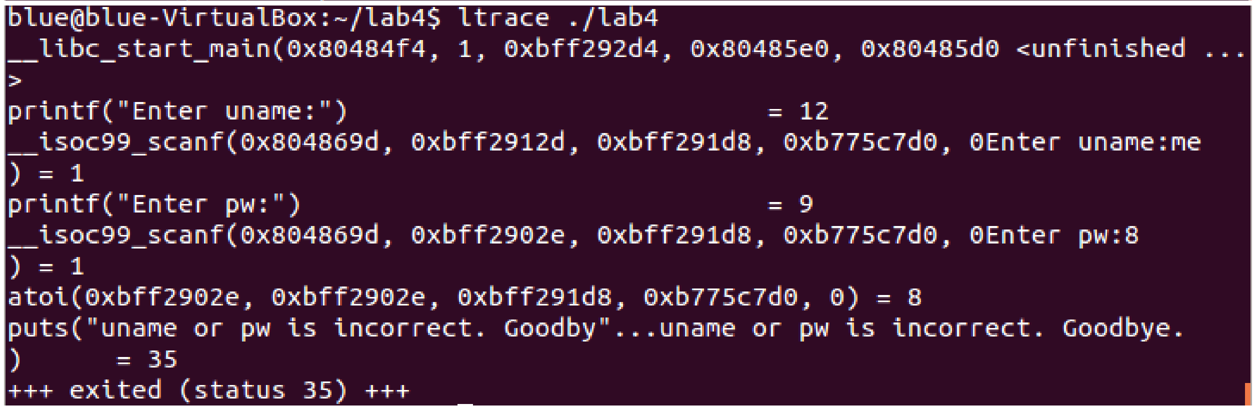
Instruction at +150 compares the above integer value with 0x9 (Decimal: 9).
The program jumps to +171 if 0x9 is less than the inputted value.

Therefore, the acceptable password is any integer greater than 9.

Validating Observations:



```
blue@blue-VirtualBox:~/lab4$ ltrace ./lab4
__libc_start_main(0x80484f4, 1, 0xbfbd53a4, 0x80485e0, 0x80485d0 <unfinished ...
>
printf("Enter uname:")                           = 12
__isoc99_scanf(0x804869d, 0xbfbd51fd, 0xbfbd52a8, 0xb77f37d0, 0Enter uname:me
) = 1
printf("Enter pw:")                              = 9
__isoc99_scanf(0x804869d, 0xbfbd50fe, 0xbfbd52a8, 0xb77f37d0, 0Enter pw:10
) = 1
atoi(0xbfbd50fe, 0xbfbd50fe, 0xbfbd52a8, 0xb77f37d0, 0) = 10
puts("Welcome"Welcome
)                                         = 8
+++ exited (status 8) +++
```



```
blue@blue-VirtualBox:~/lab4$ ltrace ./lab4
__libc_start_main(0x80484f4, 1, 0xbff292d4, 0x80485e0, 0x80485d0 <unfinished ...
>
printf("Enter uname:")                           = 12
__isoc99_scanf(0x804869d, 0xbff2912d, 0xbff291d8, 0xb775c7d0, 0Enter uname:me
) = 1
printf("Enter pw:")                              = 9
__isoc99_scanf(0x804869d, 0xbff2902e, 0xbff291d8, 0xb775c7d0, 0Enter pw:8
) = 1
atoi(0xbff2902e, 0xbff2902e, 0xbff291d8, 0xb775c7d0, 0) = 8
puts("uname or pw is incorrect. Goodby"...uname or pw is incorrect. Goodbye.
)      = 35
+++ exited (status 35) +++
```

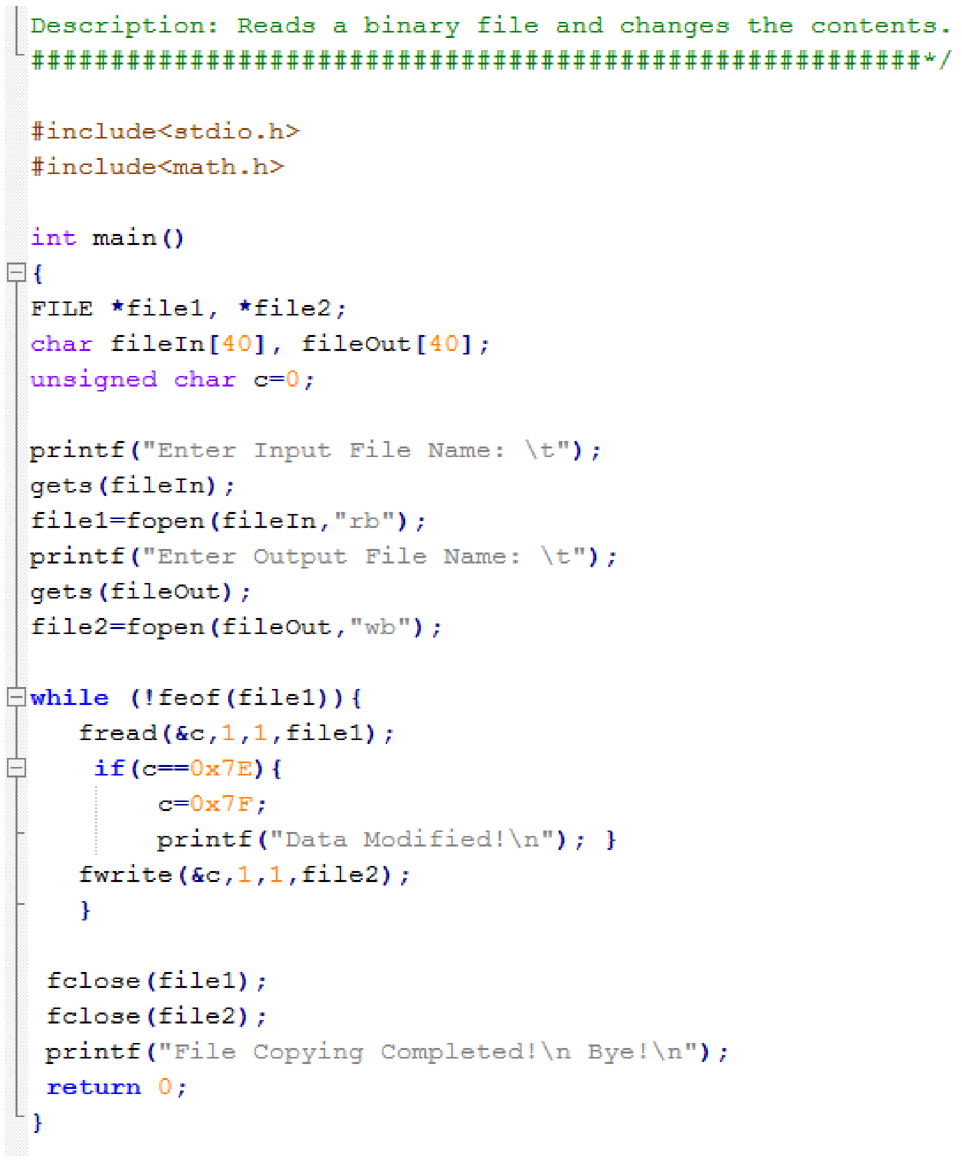
Conclusion:
Username: *me*
Password: (Any integer greater than 9)

STEP 2: Modifying the Program

To accept the password(s) rejected earlier we need to change the conditional operator from ***jle*** to ***jg***.

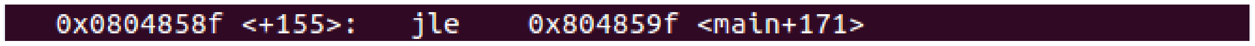
The C program below reads a binary file bit-by-bit. Before writing to an output file, the program compares the data with 0x7E (Hex value of ***jle*** operator). If found, the value is replaced with 0x7F (Hex value of ***jg*** operator).

This is a simple C program which will change all ***jle*** instructions and also any tilde symbol (**~**) if used in the program. We can modify the program to find the character sequence (eg: 0x7c, 0x24, 0x18, 0x09, 0x7e) to confirm that other instructions are not affected.



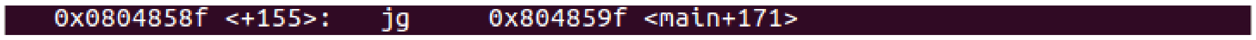
```
Description: Reads a binary file and changes the contents.
##########################################################*/

#include<stdio.h>
#include<math.h>

int main()
{
FILE *file1, *file2;
char fileIn[40], fileOut[40];
unsigned char c=0;

printf("Enter Input File Name: \t");
gets(fileIn);
file1=fopen(fileIn,"rb");
printf("Enter Output File Name: \t");
gets(fileOut);
file2=fopen(fileOut,"wb");

while (!feof(file1)){
   fread(&c,1,1,file1);
    if(c==0x7E){
        c=0x7F;
        printf("Data Modified!\n"); }
   fwrite(&c,1,1,file2);
   }

 fclose(file1);
 fclose(file2);
 printf("File Copying Completed!\n Bye!\n");
 return 0;
}
```

Before Change:



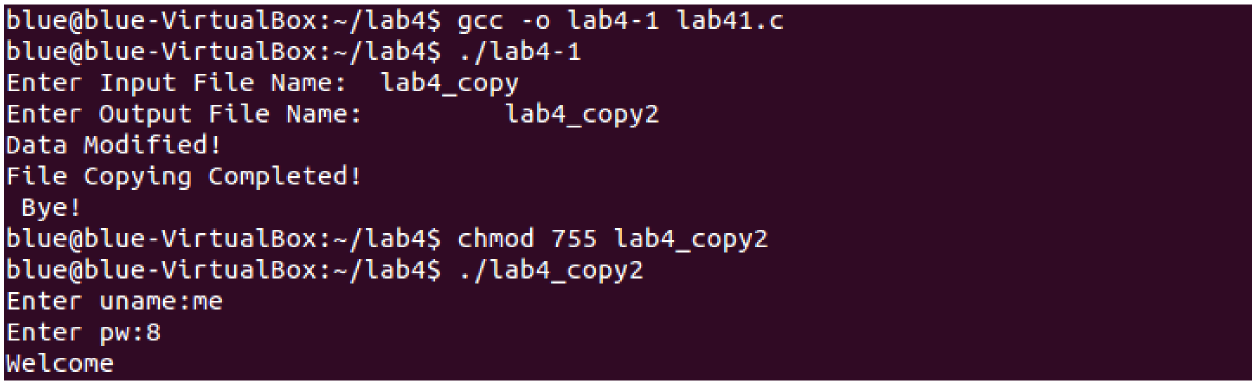
```
0x0804858f <+155>:   jle    0x804859f <main+171>
```

After Change:



```
0x0804858f <+155>:   jg     0x804859f <main+171>
```

Compiling the C-code and Testing:

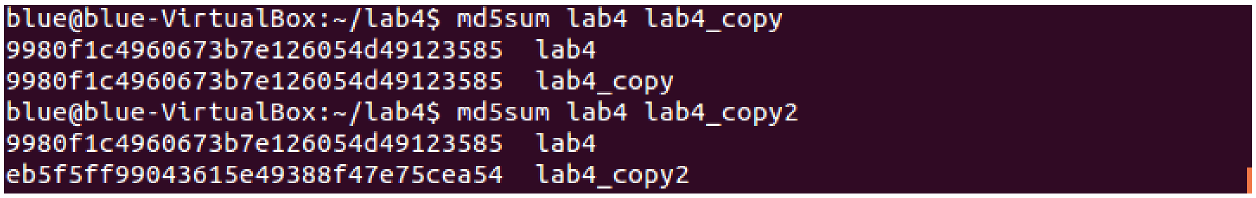


```
blue@blue-VirtualBox:~/lab4$ gcc -o lab4-1 lab41.c
blue@blue-VirtualBox:~/lab4$ ./lab4-1
Enter Input File Name:  lab4_copy
Enter Output File Name:         lab4_copy2
Data Modified!
File Copying Completed!
 Bye!
blue@blue-VirtualBox:~/lab4$ chmod 755 lab4_copy2
blue@blue-VirtualBox:~/lab4$ ./lab4_copy2
Enter uname:me
Enter pw:8
Welcome
```

STEP 3: Detecting Modifications

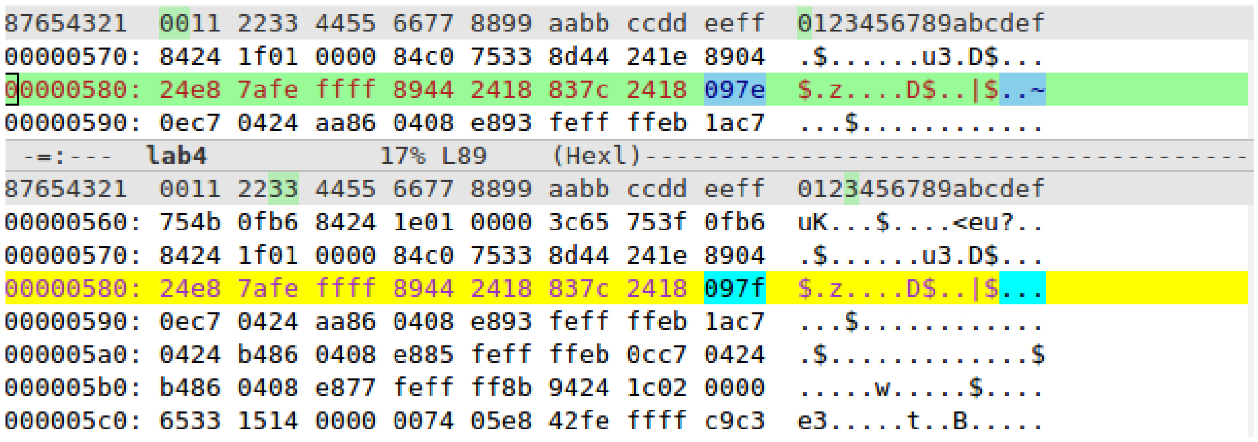
lab4	:	Main file
lab4_copy	:	A copy of lab4
lab4_copy2	:	Modified file

Executing MD5 checksum, we observe that the checksum values of lab4_copy2 is different from original file lab4.



```
blue@blue-VirtualBox:~/lab4$ md5sum lab4 lab4_copy
9980f1c4960673b7e126054d49123585  lab4
9980f1c4960673b7e126054d49123585  lab4_copy
blue@blue-VirtualBox:~/lab4$ md5sum lab4 lab4_copy2
9980f1c4960673b7e126054d49123585  lab4
eb5f5ff99043615e49388f47e75cea54  lab4_copy2
```

Difference In Emacs:



```
87654321  0011 2233 4455 6677 8899 aabb ccdd eeff  0123456789abcdef
00000570: 8424 1f01 0000 84c0 7533 8d44 241e 8904  .$......u3.D$...
00000580: 24e8 7afe ffff 8944 2418 837c 2418 097e  $.z....D$..|$..~
00000590: 0ec7 0424 aa86 0408 e893 feff ffeb 1ac7  ...$............
 -=:---  lab4           17% L89     (Hexl)-----------------------------------
87654321  0011 2233 4455 6677 8899 aabb ccdd eeff  0123456789abcdef
00000560: 754b 0fb6 8424 1e01 0000 3c65 753f 0fb6  uK...$....<eu?..
00000570: 8424 1f01 0000 84c0 7533 8d44 241e 8904  .$......u3.D$...
00000580: 24e8 7afe ffff 8944 2418 837c 2418 097f  $.z....D$..|$...
00000590: 0ec7 0424 aa86 0408 e893 feff ffeb 1ac7  ...$............
000005a0: 0424 b486 0408 e885 feff ffeb 0cc7 0424  .$.............$
000005b0: b486 0408 e877 feff ff8b 9424 1c02 0000  .....w.....$....
000005c0: 6533 1514 0000 0074 05e8 42fe ffff c9c3  e3.....t..B.....
```

Other common ways to compare files are hex diffs and timestamps.