buu [ACTF新生赛2020]SoulLike wp

原创

liuxiaohuai_ 于 2020-12-22 20:03:46 发布 659 收藏 4

分类专栏： ida64 逆向 pwn 文章标签： python 安全

本文链接：https://blog.csdn.net/liuxiaohuai_/article/details/111565121

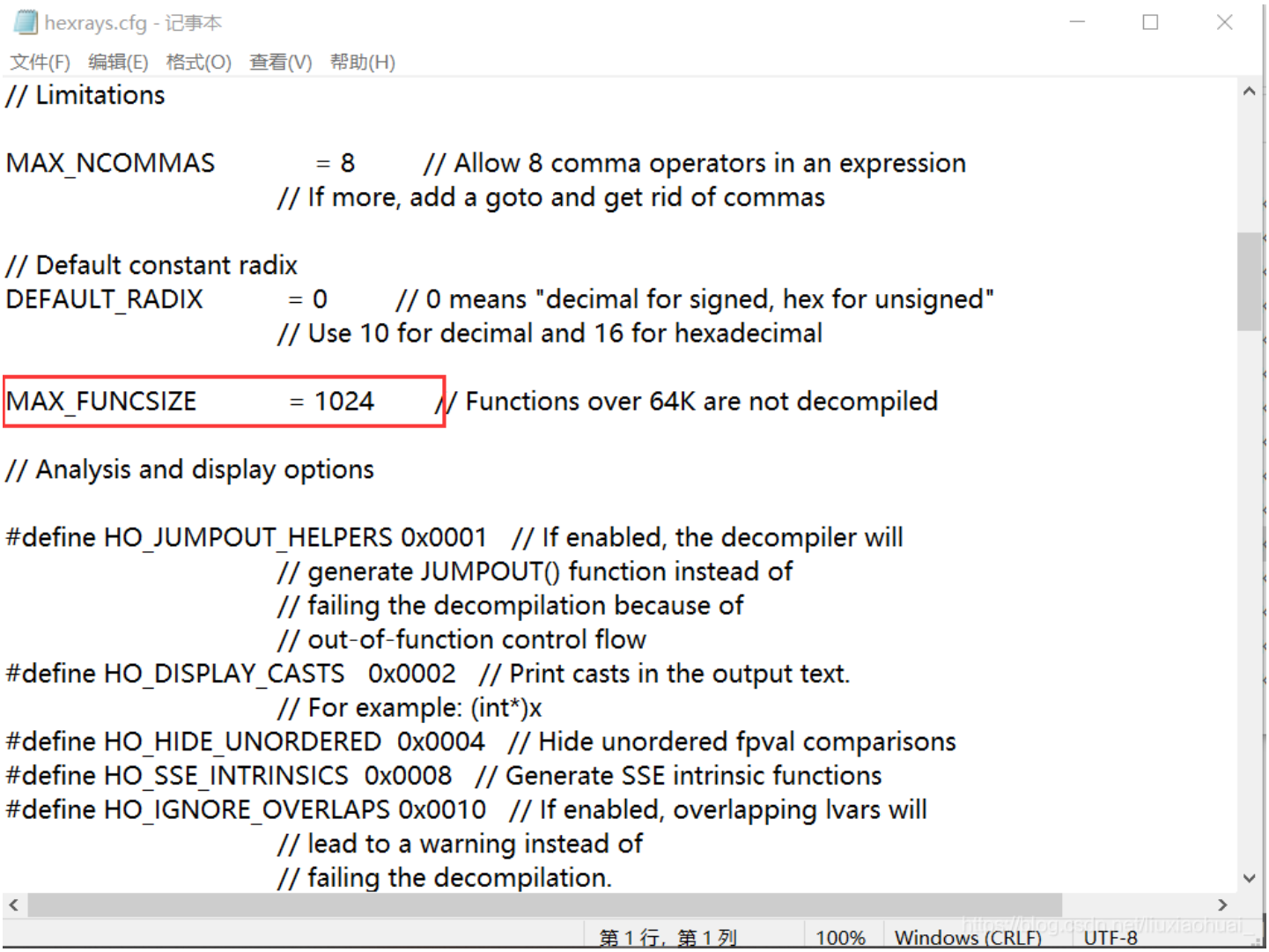
下载附件后查壳发现无壳直接拖进ida64查看`main()`函数

函数的结构非常简单。重点就在于`sub_83A()`这个函数

```
  v13 = __readfsqword(0x28u);
  printf("input flag:", a2, a3);
  scanf("%s", v11);                             // 输入flag
  v9 = 'ftca';                                  // 定义字符串actf{
  v10 = '{';
  v5 = 1;
  for ( i = 0; i <= 4; ++i )                    // 判断输入flag的前5位是否为“actf{”
  {
    if ( *(&v9 + i) != v11[i] )
    {
      v5 = 0;
      goto LABEL_6;
    }
  }
  if ( !v5 )
    goto LABEL_19;
LABEL_6:
  for ( j = 0; j <= 11; ++j )
    v8[j] = v11[j + 5];
  v3 = sub_83A(v8) && v12 == 125 ? 1 : 0;       // 判断flag最后以为是否为‘}’
  if ( v3 )
  {
    printf("That's true! flag is %s", v11);
    result = 0LL;
  }
  else
  {
LABEL_19:
```

```
    printf("Try another time...");
    result = 0LL;
  }
  return result;
}
```


这个函数很大直接f5是会报错的，要将ida /ctg目录下的hexrays.cfg文件中的MAX_FUNCSIZE=64 改为 MAX_FUNCSIZE=1024



再按f5等待一段时间后出伪c代码。

```
3007  v1[10] ^= 0x73u;
3008  v1[11] ^= v1[2];
3009  *v1 ^= 0x59u;
3010  v1[1] ^= 0x7Eu;
3011  v1[2] ^= 0x4Fu;
3012  v1[3] ^= 0x2Bu;
3013  v1[4] ^= *v1;
3014  v1[5] ^= 0x25u;
3015  v1[6] ^= 0x2Eu;
3016  v1[7] ^= 0x3Cu;
3017  v1[8] ^= 0x6Bu;
3018  v1[9] ^= 0x70u;
3019  v1[10] ^= 0x29u;
3020  v1[11] ^= 0x3Bu;
3021  v4 = 0x7E;
3022  v5 = 0x32;
3023  v6 = 0x25;
3024  v7 = 0x58;
3025  v8 = 0x59;
3026  v9 = 0x6B;
3027  v10 = 0x35;
3028  v11 = 0x6E;
3029  v12 = 0;
3030  v13 = 0x13;
3031  v14 = 0x1E;
3032  v15 = 0x38;
3033  for ( i = 0; i <= 11; ++i )
3034  {
3035    if ( *(&v4 + i) != a1[i] )
3036    {
3037      printf("wrong on #%d\n", i);
3038      return 0LL;
3039    }
3040  }
3041  return 1LL;
3042}
```


这里就是将输入的flag进行接近3000行的异或操作后与最后的v4~v15的数值进行比较。但是当判断错误时会输出`wrong on # + 出错的位置`

```
for ( i = 0; i <= 11; ++i )
{
  if ( *(&v4 + i) != a1[i] )
  {
    printf("wrong on #%d\n", i);
    return 0LL;
  }
}
```

所以我们可以直接爆破。

```python
from pwn import *

T = ['\x00', '\x01', '\x02', '\x03', '\x04', '\x05', '\x06', '\x07', '\x08', '\t', '\n', '\x0b', '\x0c', '\r', '
\x0e', '\x0f', '\x10', '\x11', '\x12', '\x13', '\x14', '\x15', '\x16', '\x17', '\x18', '\x19', '\x1a', '\x1b', '
\x1c', '\x1d', '\x1e', '\x1f', ' ', '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', '
0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '<', '=', '>', '?', '@', 'A', 'B', 'C', 'D', 'E', 'F'
, 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '[', '\\',
 ']', '^', '_', '`', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', '
s', 't', 'u', 'v', 'w', 'x', 'y', 'z', '{', '|', '}', '~', '\x7f']

a = 'actf{'
b = 0
pty = 1
while 1:
    if b == 12:
        break
    for i in T:
        io = process('./SoulLike')      #这里括号里的内容要写自己电脑上SoulLike文件的下载位置!
        #启动本地程序进行交互，用于gdb调试
        flag = a + i
        flag = flag.ljust(17,'@')#返回一个原字符串左对齐,并使用空格填充至指定长度的新字符串。如果指定的长度小于原字符串的长
度则返回原字符串。
        flag += '}'
        success(flag)
        io.sendline(flag)       ## sendline发送数据会在最后多添加一个回车
        io.recvuntil('#')       #读取到指定数据
        if b < 9 :
            n = int(io.recv(1)) # recv(n)读取n个字节
        else:
            n = int(io.recv(2))
        io.close()              #关闭连接
        if n == b + 1:          #判断这一位是否正确
            a = a + i
            b = b + 1
            break
```

不知道我的kali为什么运行这段代码会报错。所以我是直接在命令行中使用的python

```
root@kali:~/Desktop# python
```

然后将上面这个段代码直接粘贴进来

```
>>> from pwn import *
>>>
>>> T = ['\x00', '\x01', '\x02', '\x03', '\x04', '\x05', '\x06', '\x07', '\x08', '\t', '\n', '\x0b', '\x0c', '\r', '\x0e', '\x0f', '\x10',
\x11', '\x12', '\x13', '\x14', '\x15', '\x16', '\x17', '\x18', '\x19', '\x1a', '\x1b', '\x1c', '\x1d', '\x1e', '\x1f', ' ', '!', '"', '#',
$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '<', '=', '>', '?',
@', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '[',
\\', ']', '^', '_', '`', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z', '{', '|', '}', '~', '\x7f']
>>> #for i in range(128):
...     #  T.append(chr(i))
... #print(T)
... now = 'actf{'
>>> num = 0
>>> pty = 1
>>> while 1:
...     if num == 12:
...         break
...     for i in T:
...         io = process('./SoulLike')
...         flag = now + i
...         flag = flag.ljust(17,'@')
...         flag += '}'
...         success(flag)
...         io.sendline(flag)
...         io.recvuntil('#')
...         if num < 9 :
...             n = int(io.recv(1))
...         else:
...             n = int(io.recv(2))
...         io.close()
...         if n == num + 1:
...             now = now + i
...             num = num + 1
...             break
```


爆破出flag

```
[+] actf{b0Nf|Re_LiT
[*] Process './SoulLike' stopped with exit code 0 (pid 9365)
'input flag:wrong on #'
[x] Starting local process './SoulLike'
[+] Starting local process './SoulLike': pid 9366
[+] actf{b0Nf|Re_LiT
[*] Process './SoulLike' stopped with exit code 0 (pid 9366)
'input flag:wrong on #'
[x] Starting local process './SoulLike'
[+] Starting local process './SoulLike': pid 9367
[+] actf{b0Nf|Re_LiT
[*] Process './SoulLike' stopped with exit code 0 (pid 9367)
'input flag:wrong on #'
[x] Starting local process './SoulLike'
[+] Starting local process './SoulLike': pid 9368
[+] actf{b0Nf|Re_LiT }
[*] Process './SoulLike' stopped with exit code 0 (pid 9368)
'input flag:wrong on #'
[x] Starting local process './SoulLike'
[+] Starting local process './SoulLike': pid 9369
[+] actf{b0Nf|Re_LiT!}
[*] Process './SoulLike' stopped with exit code 0 (pid 9369)
Traceback (most recent call last):
```


得到：flag{b0Nf|Re_LiT!}