RE-实验吧recursive

原创

Alikas 于 2019-04-11 13:31:33 发布 148 收藏
分类专栏： 逆向 文章标签： RE 实验吧 writeup

Alikas-0x03
题目：实验吧recursive

拿到题先file一下，64位ELF文件

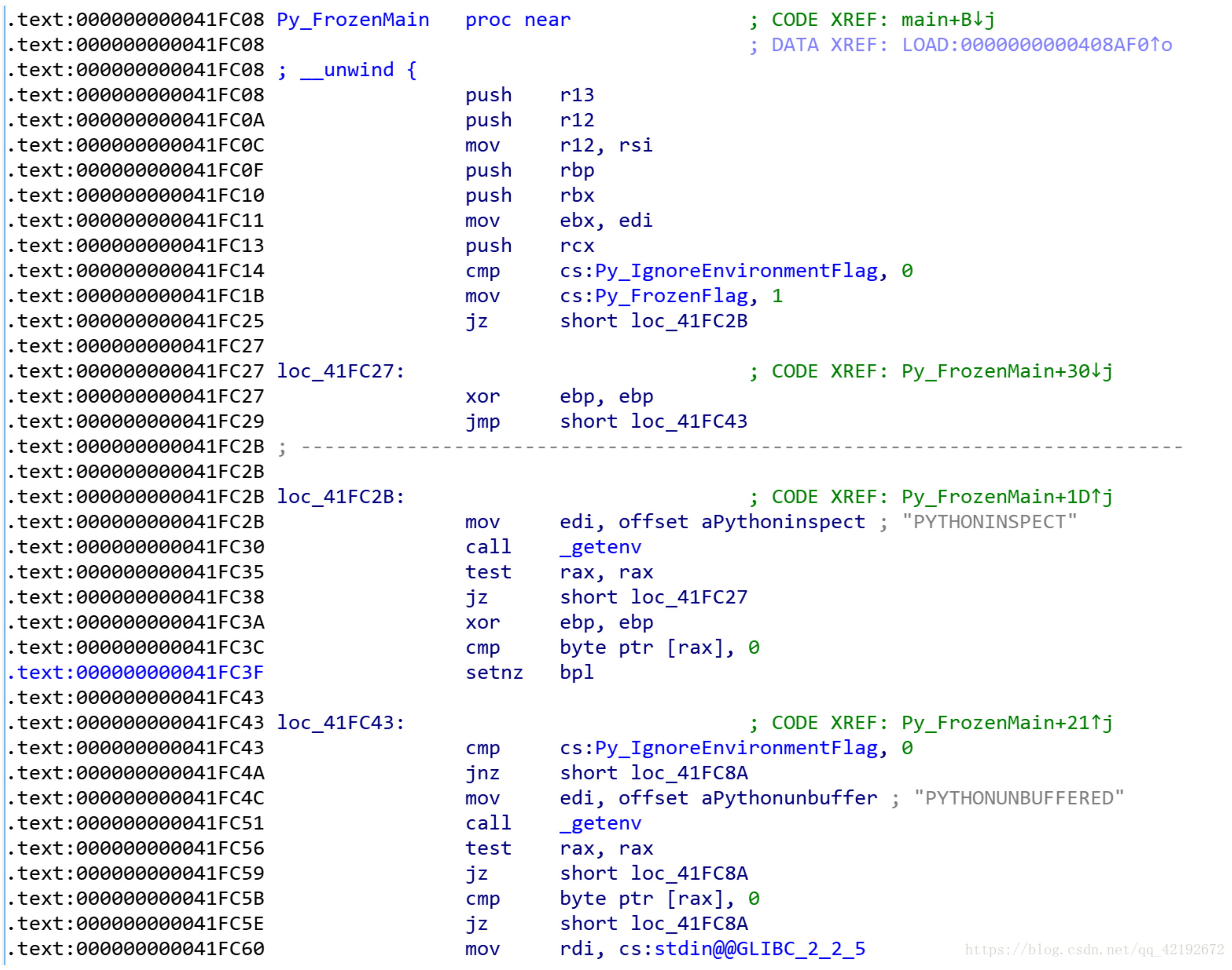
```
file recursive_python
recursive_python: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/l,
 for GNU/Linux 2.6.32, BuildID[sha1]=d33a21b0c7971d7dd951070fdd06cd393dc78cce, with debug_info, not stripped
```

运行一下，结果被调侃了0.0…

```
./recursive_python
You wish it was that easy!
```

拖进IDA中看一波

发现该文件在内部运行python解释器，It is created by Freeze，查资料发现，基本上这种情况都可以找到函数Py_FrozenMain，跟进



```
.text:000000000041FC08 Py_FrozenMain   proc near               ; CODE XREF: main+B↓j
.text:000000000041FC08                                         ; DATA XREF: LOAD:0000000000408AF0↑o
.text:000000000041FC08 ; __unwind {
.text:000000000041FC08                 push    r13
.text:000000000041FC0A                 push    r12
.text:000000000041FC0C                 mov     r12, rsi
.text:000000000041FC0F                 push    rbp
.text:000000000041FC10                 push    rbx
.text:000000000041FC11                 mov     ebx, edi
.text:000000000041FC13                 push    rcx
.text:000000000041FC14                 cmp     cs:Py_IgnoreEnvironmentFlag, 0
.text:000000000041FC1B                 mov     cs:Py_FrozenFlag, 1
.text:000000000041FC25                 jz      short loc_41FC2B
.text:000000000041FC27
.text:000000000041FC27 loc_41FC27:                             ; CODE XREF: Py_FrozenMain+30↓j
.text:000000000041FC27                 xor     ebp, ebp
.text:000000000041FC29                 jmp     short loc_41FC43
.text:000000000041FC2B ; ---------------------------------------------------------------------------
.text:000000000041FC2B
.text:000000000041FC2B loc_41FC2B:                             ; CODE XREF: Py_FrozenMain+1D↑j
.text:000000000041FC2B                 mov     edi, offset aPythoninspect ; "PYTHONINSPECT"
.text:000000000041FC30                 call    _getenv
.text:000000000041FC35                 test    rax, rax
.text:000000000041FC38                 jz      short loc_41FC27
.text:000000000041FC3A                 xor     ebp, ebp
.text:000000000041FC3C                 cmp     byte ptr [rax], 0
.text:000000000041FC3F                 setnz   bpl
.text:000000000041FC43
.text:000000000041FC43 loc_41FC43:                             ; CODE XREF: Py_FrozenMain+21↑j
.text:000000000041FC43                 cmp     cs:Py_IgnoreEnvironmentFlag, 0
.text:000000000041FC4A                 jnz     short loc_41FC8A
.text:000000000041FC4C                 mov     edi, offset aPythonunbuffer ; "PYTHONUNBUFFERED"
.text:000000000041FC51                 call    _getenv
.text:000000000041FC56                 test    rax, rax
.text:000000000041FC59                 jz      short loc_41FC8A
.text:000000000041FC5B                 cmp     byte ptr [rax], 0
.text:000000000041FC5E                 jz      short loc_41FC8A
.text:000000000041FC60                 mov     rdi, cs:stdin@@GLIBC_2_2_5
```

我们可以看到这个函数里有两个变量"PYTHONINSPECT"和"PYTHONUNBUFFERED"，之后都会调用函数getenv()

函数说明:getenv()用来取得参数envvar环境变量的内容。参数envvar为环境变量的名称，如果该变量存在则会返回指向该内容的指针。

那么说明如果这两个变量都存在，会产生一些新的东西，我们修改完再运行一下（随便赋值就好），如下:

```
export PYTHONINSPECT=6
export PYTHONUNBUFFERED=6
./recursive_python
You wish it was that easy!
>>>
```

```
其中Linux export命令用于设置或显示环境变量
```

运行完我们发现多了几个文件，猜测flag蕴藏其中

```
unstep_579c82e9  unstep_f67baaeb unstep_34a4d33b      unstep_84fc2d39
```

运行一下，怎么还是没有=-=...

```
 ✘ □ □ root@DESKTOP-VU7HR7F □ /mnt/d/CTF/Games/题库/实验吧/RE/recursive_python □ ./unstep_34a4d33
b
You wish it was that easy!
>>> ^Z
[2]  + 154 suspended  ./unstep_34a4d33b
 ✘ □ □ root@DESKTOP-VU7HR7F □ /mnt/d/CTF/Games/题库/实验吧/RE/recursive_python □ ./unstep_84fc2d3
9
You wish it was that easy!
^Z
[3]  + 165 suspended  ./unstep_84fc2d39
 ✘ □ □ root@DESKTOP-VU7HR7F □ /mnt/d/CTF/Games/题库/实验吧/RE/recursive_python □ ./unstep_579c82e
9
You wish it was that easy!
^Z
[4]  + 178 suspended  ./unstep_579c82e9
 ✘ □ □ root@DESKTOP-VU7HR7F□ /mnt/d/CTF/Games/题库/实验吧/RE/recursive_python □ ./unstep_f67baaeb
You wish it was that easy!
^Z
[5]  + 187 suspended  ./unstep_f67baaeb
```

拖进IDA后发现看不懂…

但猜测flag就在文件中，故直接用string搜索以及正则表达式匹配：

```
strings ./unstep_f67baaeb g| grep -o 'flag{.*}'
flag{python_taken_2_far}
```

```
*在正则表达式中表示匹配任意文本
```

最后在最后一个文件中找到flag

总结：
1.第一次遇到python逆向，查了不少资料，姿势学到了！
2.WSL真好用0.0！