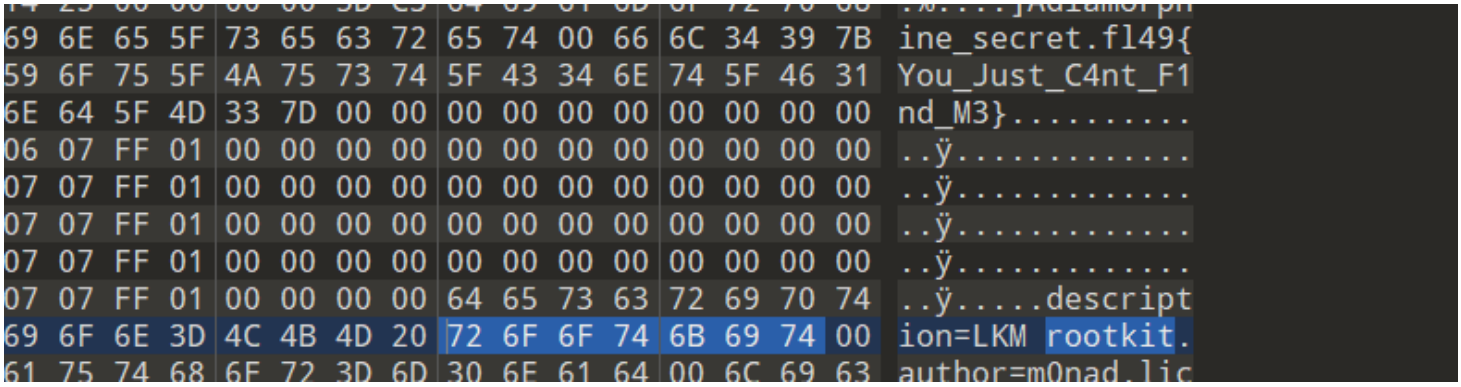
misc backdoor



rootkit上面有类似flag的东西
提交

pwn shop

比较简单的一道题，当时没仔细看真的是亏了

```
from pwn import *
io = process('./pwn')
io.recvuntil('> ')
payload = '3\x00\n'+'0\x00\n'+'3\x00\n'+'0\x00\n'
io.send(payload)

io.interactive()
```

利用的是sale的sleep时，可以造成竞争。

re aet-reverse

tea加密

```
#include <stdio.h>

int main() {
    unsigned char check[33] = { 66, 199, 202, 64, 193, 117, 22, 239, 231, 55, 110, 105, 27, 11,
    int a1[16] = { 0,1,3,4,5,6,7,8,9,10,11,12,13,14,15,0 };
    unsigned char flag[33] = {};
    int v7 = (a1[2] << 8) | (a1[1] << 16) | (*a1 << 24) | a1[3];
    int v6 = (a1[6] << 8) | (a1[5] << 16) | (a1[4] << 24) | a1[7];
    int v5 = (a1[10] << 8) | (a1[9] << 16) | (a1[8] << 24) | a1[11];
    int v4 = (a1[14] << 8) | (a1[13] << 16) | (a1[12] << 24) | a1[15];
    printf("%x %x %x %x\n", v7, v6, v5, v4);
    for (int i = 0; i < 4; i++) {
        unsigned int v12 = (check[8 * i] << 24) | (check[8 * i+1] << 16) | (check[8 * i+2] << 8)
        unsigned int v11 = (check[8 * i+4] << 24) | (check[8 * i + 5] << 16) | (check[8 * i + 6]
        unsigned int v10 = 0;
        v10 = 0xc6ef3720;
        //printf("%x\n", v10);
        for (int j = 0; j <= 0x1f; j++) {
            v11 -= (v12 + v10) ^ (16 * v12 + v5) ^ ((v12 >> 5) + v4);
            v12 -= (v11 + v10) ^ (16 * v11 + v7) ^ ((v11 >> 5) + v6);
            v10 -= 0x9e3779b9;
        }
        //flag[2 * i] = v12;
        //flag[2 * i + 1] = v11;

        flag[8 * i] = (v12>>24)&0xff;
        flag[8 * i + 1] = (v12>>16)&0xff;
        flag[8 * i + 2] = (v12>>8) & 0xff;
        flag[8 * i + 3] = v12 & 0xff;
        flag[8 * i + 4] = (v11>>24) & 0xff;
        flag[8 * i + 5] = (v11>>16) & 0xff;
        flag[8 * i + 6] = (v11>>8) & 0xff;
        flag[8 * i + 7] = v11 & 0xff;
    }
    for (int i = 0; i < 32; i++) {
        printf("%c", flag[i]);
    }
    return 0;

}
```

re BabyTrans

很屎，ida看不到cfg，应该不是扁平化处理

```
00000004192A0 unk_4192A0      db  70h ; p             ; DATA XREF: main+104
00000004192A1                 db 0C2h
00000004192A2                 db  2Dh ; -
00000004192A3                 db 0DFh
00000004192A4                 db 0D8h
00000004192A5                 db  1Ch
00000004192A6                 db  87h
00000004192A7                 db 0EAh
00000004192A8                 db  1Dh
00000004192A9                 db  28h ; (
00000004192AA                 db  93h
00000004192AB                 db  5Bh ; [
00000004192AC                 db 0CFh
00000004192AD                 db  4Dh ; M
00000004192AE                 db    2
00000004192AF                 db 0FCh
00000004192B0                 db 0FFh
00000004192B1                 db 0D9h
00000004192B2                 db 0C6h
00000004192B3                 db  61h ; a
00000004192B4                 db  4Dh ; M
00000004192B5                 db  56h ; V
```

可以看到这个4192a0是check的值

```
:ext:000000000041104B                 jnz     loc_400C1D
:ext:0000000000411051                 cmp     byte ptr [rax+4], 7Bh ; '{'
:ext:0000000000411055                 jnz     loc_400C1D
:ext:000000000041105B                 cmp     byte ptr [rax+25h], 7Dh ; '}'
:ext:000000000041105F                 jnz     loc_400C1D
:ext:0000000000411065                 mov     edx, 20h ; ' '
:ext:000000000041106A                 mov     esi, offset unk_4192A0
:ext:000000000041106F                 mov     rdi, rbx
:ext:0000000000411072                 call    _CRYPTO_memcmp
:ext:0000000000411077                 test    eax, eax
:ext:0000000000411079                 jz      loc_411FFE
:ext:000000000041107F
:ext:000000000041107F loc_41107F:                             ; CODE XREF: main+11478
```

按照提示找的aes加密的地方

```
.text:0000000000400CF4                mov     edi, offset unk_419244
.text:0000000000400CF9                mov     rbx, rax
.text:0000000000400CFC                call    _AES_set_encrypt_key
.text:0000000000400D01                lea     rdx, [rsp+168h+var_118]
.text:0000000000400D06                mov     ecx, 1
.text:0000000000400D0B                mov     rsi, rbx
.text:0000000000400D0E                mov     rdi, rbp
.text:0000000000400D11                call    _AES_ecb_encrypt
.text:0000000000400D16                lea     rax, [rbx+10h]
.text:0000000000400D1A                lea     rdi, [rbp+10h]
.text:0000000000400D1E                lea     rdx, [rsp+168h+var_118]
.text:0000000000400D23                mov     ecx, 1
.text:0000000000400D28                mov     rsi, rax
.text:0000000000400D2B                call    _AES_ecb_encrypt
.text:0000000000400D30                mov     rax, rbx
.text:0000000000400D33                neg     rax
.text:0000000000400D36                and     eax, 0Fh
.text:0000000000400D39                jz      loc_411FD7
.text:0000000000400D3F                add     byte ptr [rbx], 6Fh ; 'o'
```

但是直接用aes解发现是乱码，可以知道肯定还有其他操作·。

跟踪时发现总共3个逻辑

首先对输入的每个字符都+6

```
:ext:0000000000400CB2                call    __ZNKSt7__cxx1112basic_stringIcSt11char_tra
:ext:0000000000400CB7                lea     rdi, [rsp+168h+var_138]
:ext:0000000000400CBC                mov     esi, 6
:ext:0000000000400CC1                call    sub_4189A0      ; a2=6 a1=input
:ext:0000000000400CC6                mov     rbp, rax
:ext:0000000000400CC9                mov     rax, [rsp+168h+var_138]
:ext:0000000000400CCE                lea     rdx, [rsp+168h+var_128]
:ext:0000000000400CD3                cmp     rax, rdx
:ext:0000000000400CD6                jz      short loc_400CE0
:ext:0000000000400CD8                mov     rdi, rax        ; void *
```

4192a0就是对每个输入字符都加6的函数。

```
ext:0000000000400CF4                mov     edi, offset unk_419244
ext:0000000000400CF9                mov     rbx, rax
ext:0000000000400CFC                call    _AES_set_encrypt_key
ext:0000000000400D01                lea     rdx, [rsp+168h+var_118]
ext:0000000000400D06                mov     ecx, 1
ext:0000000000400D0B                mov     rsi, rbx
ext:0000000000400D0E                mov     rdi, rbp
```

然后用aes加密，密钥为[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]，unk_419244就是密钥了。

ecb模式加密

对加密后的字符这个就是普通的函数，（恐怕就是造成cfg弄不出来的原因）每个都-0x40

得到上述值

```
from Crypto.Cipher import AES

ciphertext = [112, 194, 45, 223, 216, 28, 135, 234, 29, 40, 147, 91, 207, 77, 2, 252, 255, 217,
for i in range(32):
    ciphertext[i] += 0x40
    ciphertext[i] = ciphertext[i]%256
#AES ECB模式解密
key =[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]    #密钥
aes = AES.new(key=bytes(key),mode=AES.MODE_ECB)  #mode是设置模式
string = bytes(ciphertext)
print(aes.decrypt(string))
#answer = list(b'BY\x8dD(\xe6\xac\xa5\x89\x8d\xcf\xd9V@\xaa\x88\xb1ngB\xab\x03\xd3\x8a\x1dJ/\xa8
#for i in range(32):
#    print(hex(answer[i]),end=' ')

process =  list(aes.decrypt(string))
print(aes.decrypt(string))
print(process)
for i in range(32):
    process[i] -= 6
print(bytes(process))
```

re goodpy

很屎

pyinstractor解压后，就一行代码，重点在pyd内部。。。

可恶啊

pyd拖入ida

看导出表

Name	Address	Ordinal
PyInit_goodpy	0000000180005340	1
DllEntryPoint	**0000000180001354**	**[main entry]**

看init

```
1 __int64 PyInit_goodpy()
2 {
3   return PyModuleDef_Init(&importance);
4 }
```

这个importance是很重要的

参考

https://bbs.pediy.com/thread-259124.htm

```
000018000973F                   db    0
0000180009740 importance        dq 1                    ; DATA XREF: PyInit_goodpy↑o
0000180009748                   dq 0
0000180009750                   dq 0
0000180009758                   dq 0
0000180009760                   dq 0
0000180009768                   dq offset aGoodpy_3     ; "goodpy"
0000180009770                   dq 0
0000180009778                   dq 0
0000180009780                   dq offset unk_18000A878
0000180009788                   dq offset struct_pyx
0000180009790                   db    0
```

找到了结构体（自定义）

看到有个xor，盲猜就是个普通的xor，关键就是找的check和xor的值了。

可惜这个没啥卵用，最后搜了一下goodpy.pypypy

```
v9 = (_QWORD *)right_sign;
v71 = right_sign;
v10 = PyList_New(38i64);
v11 = (_QWORD *)v10;
if ( !v10 )
{
  qword_18000A950 = (__int64)aGoodpyPy;
  dword_18000A90C = 4;
  dword_18000A908 = 1310;
  goto LABEL_109;
}
v12 = (_QWORD *)number14;
v13 = (_QWORD *)number91;
v14 = number1;
v15 = (_QWORD *)number95;
v16 = (_QWORD *)number86;
v17 = (_QWORD *)number87;
++*(_QWORD *)number1;
v18 = (_QWORD *)number94;
v19 = (_QWORD *)number0;
**(_QWORD **)(v10 + 24) = v14;                // 2
v20 = (_QWORD *)number3;
++*(_QWORD *)number3;
*(_QWORD *)(*(_QWORD *)(v10 + 24) + 8i64) = v20;// 4
++*v12;
*(_QWORD *)(*(_QWORD *)(v10 + 24) + 16i64) = v12;// 15
++*v20;
```

定义到了关键函数。

这时候不知道怎么办了。

动调走起

发现是和goodgoodgood....异或，还有一组值如上图。

```
key = list(b'goodgoodgoodgoodgoodgoodgoodgoodgood!!')
data = [1,3,14,3,28,95,86,0,86,94,91,80,84,14,91,87,87,91,14,82,95,87,94,83,86,87,12,83,87,93,92

flag =[]
for i in range(38):
    flag.append(key[i]^data[i])
print(bytes(flag))
```