wp

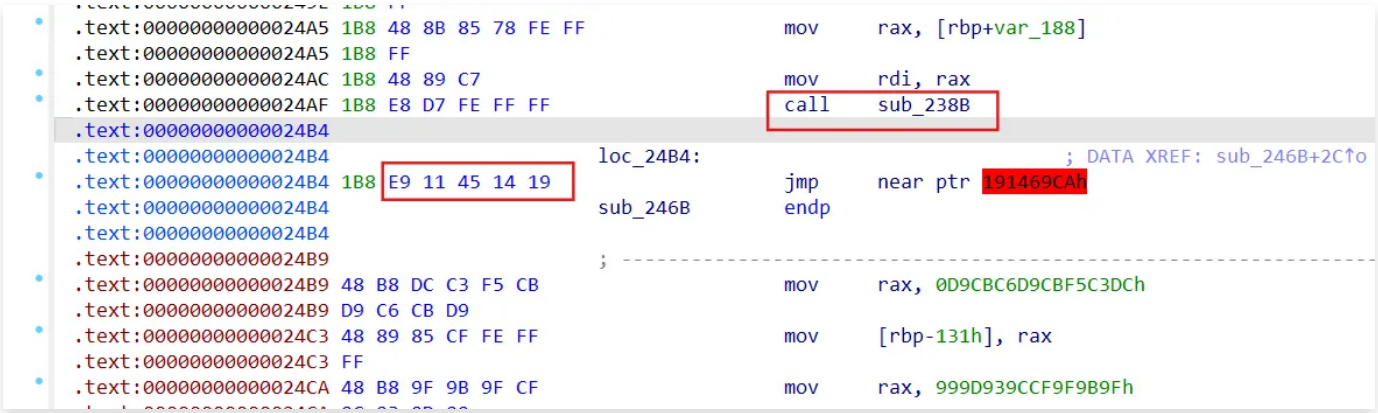
去花

新手时复现花指令很少见师傅放源码，其实对比观察源码就很清楚了，本质就是去除多余opcode，亲自动手写一遍再逆向感觉是完全不一样的。

源码如下

第三处

类似smc吧，一开始想整活但嫌麻烦，先加密再去加的花，所以不做太大改动，原理就是静态时是E9 jmp指令，运行后xor成5个nop，不影响程序流执行



```c
static inline void* page_align(void *addr) {
    size_t page_size = getpagesize();
    return (void*)((uintptr_t)addr & ~(page_size - 1));
}

void flower_with_addr(void *junk_addr) {
    void *page = page_align(junk_addr);
    size_t page_size = getpagesize();

    if (mprotect(page, page_size, PROT_READ | PROT_WRITE | PROT_EXEC) != 0) {
        perror("mprotect");
        return;
    }

    uint8_t key[] = {0x79, 0x81, 0xd5, 0x84, 0x89};
    for (int i = 0; i < 5; i++) {
        ((uint8_t*)junk_addr)[i] ^= key[i];  // -> 0x90 each
    }

    mprotect(page, page_size, PROT_READ | PROT_EXEC);
}

 void *junk_addr = &&junk_label;
    flower_with_addr(junk_addr);
junk_label:
    __asm__ volatile (
        ".byte 0xe9 \n\t"
        ".byte 0x11 \n\t"
        ".byte 0x45 \n\t"
        ".byte 0x14 \n\t"
        ".byte 0x19 \n\t"
        ::: "memory"
    );
```

第一处

标准jz/jnz

```
.text:0000000000002CB3 48 89 45 E8                      mov     [rbp-18h], rax
.text:0000000000002CB7 31 C0                            xor     eax, eax
.text:0000000000002CB9 53                               push    rbx
.text:0000000000002CBA 48 31 DB                         xor     rbx, rbx
.text:0000000000002CBD 48 85 DB                         test    rbx, rbx
.text:0000000000002CC0 75 02                            jnz     short loc_2CC4
.text:0000000000002CC2 74 01                            jz      short near ptr loc_2CC4+1
.text:0000000000002CC4
.text:0000000000002CC4                  loc_2CC4:                               ; CODE XREF: .text:0000000000002CC0↑j
.text:0000000000002CC4                                                          ; .text:0000000000002CC2↑j
.text:0000000000002CC4 E9 5B 48 B8 C9                   jmp     near ptr 0FFFFFFFFC9B87524h
.text:0000000000002CC4                  ; ---------------------------------------------------------------------------
.text:0000000000002CC9 CF CE C9 CE CB 92…               db 0CFh, 0CEh, 0C9h, 0CEh, 0CBh, 92h, 0CBh
.text:0000000000002CD0                  ; ---------------------------------------------------------------------------
.text:0000000000002CD0 48 BA C9 99 9F C9                mov     rdx, 93C9CB9FC99F99C9h
```

```
1   __asm__ volatile (
2           "pushq %%rbx \n\t"
3           "xorq  %%rbx, %%rbx \n\t"    //ida 识别$1 $1+1
4           "testq %%rbx, %%rbx \n\t"
5           "jnz   1f \n\t"
6           "jz    2f \n\t"
7
8           "1: \n\t"
9           ".byte 0xe9 \n\t"
10
11          "2: \n\t"
12          "popq %%rbx \n\t"
13          :
14          :
15          : "rbx", "memory"
16      );
```

第二处

call+retn，rsp具体加的值根据opcode计算其实很明显

```
.text:0000000000002DA0 BE 60 00 00 00                   mov     esi, 60h ; '`'
.text:0000000000002DA5 48 89 C7                         mov     rdi, rax
.text:0000000000002DA8 E8 4C F5 FF FF                   call    sub_22F9
.text:0000000000002DAD 48 8D 85 C0 FE FF                lea     rax, [rbp-140h]
.text:0000000000002DAD FF
.text:0000000000002DB4 48 89 85 A8 FE FF                mov     [rbp-158h], rax
.text:0000000000002DB4 FF
.text:0000000000002DBB B8 00 00 00 00                   mov     eax, 0
.text:0000000000002DC0 E8 2C FE FF FF                   call    sub_2BF1
.text:0000000000002DC5 E8 01 00 00 00                   call    loc_2DCB
.text:0000000000002DC5                  ; ---------------------------------------------------------------------------
.text:0000000000002DCA 83                               db 83h
.text:0000000000002DCB                  ; ---------------------------------------------------------------------------
.text:0000000000002DCB
.text:0000000000002DCB                  loc_2DCB:                               ; CODE XREF: .text:0000000000002DC5↑j
.text:0000000000002DCB 48 83 04 24 08                   add     qword ptr [rsp], 8
.text:0000000000002DD0 C3                               retn
.text:0000000000002DD0                  ; ---------------------------------------------------------------------------
.text:0000000000002DD1 F3 83 BD 9C FE FF…               db 0F3h, 83h, 0BDh, 9Ch, 0FEh, 2 dup(0FFh)
.text:0000000000002DD8 01 0F 8F ED 00 00                dq 48000000ED8F0F01h, 8B48FFFFFF90858Bh, 0FFFFF8858948040h
```

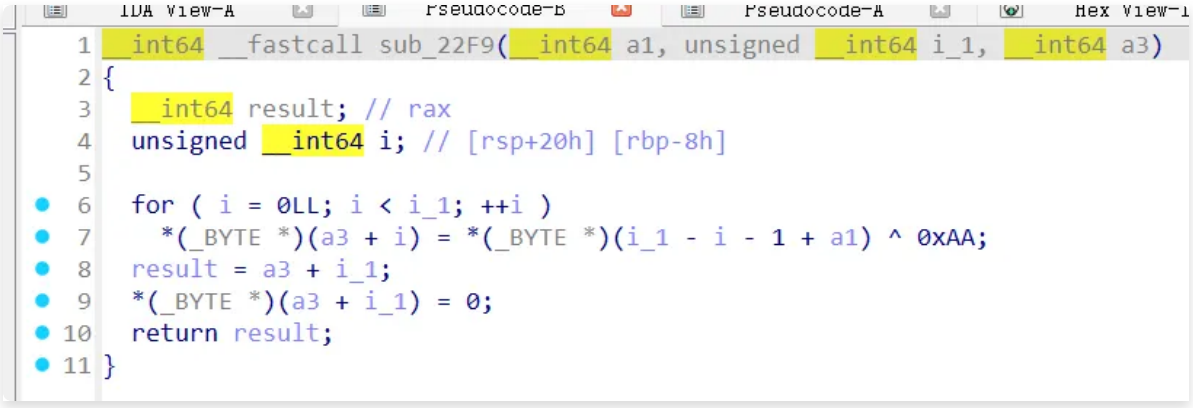
```c
__asm__ volatile (
        "call 1f \n\t"
        ".byte 0x83 \n\t"

        "1: \n\t"
        "addq $8, (%%rsp) \n\t"
        "ret \n\t"
        ".byte 0xf3 \n\t"
        :
        :
        : "memory"
    );
```

解密

salsa20+aes_ecb无魔改

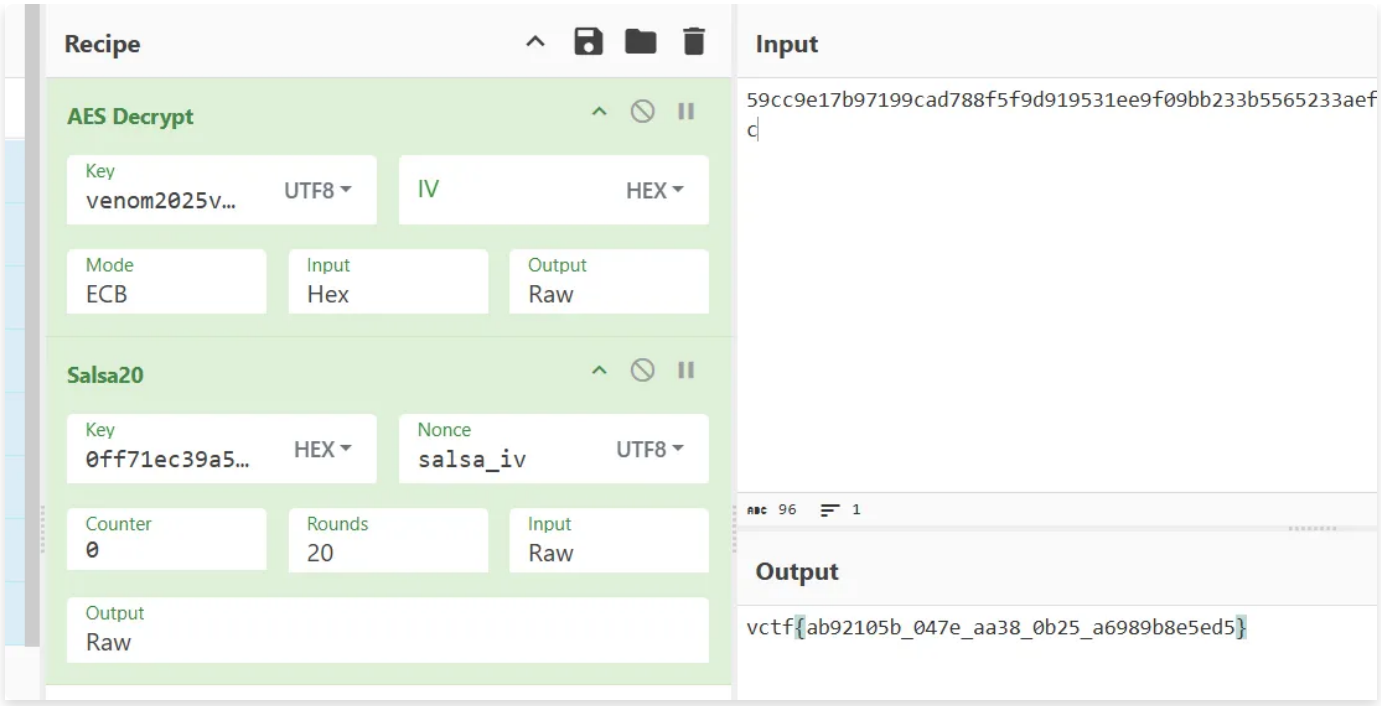
中间加了简单的倒序xor处理去除一些内容特征



```c
__int64 __fastcall sub_22F9(__int64 a1, unsigned __int64 i_1, __int64 a3)
{
  __int64 result; // rax
  unsigned __int64 i; // [rsp+20h] [rbp-8h]

  for ( i = 0LL; i < i_1; ++i )
    *(_BYTE *)(a3 + i) = *(_BYTE *)(i_1 - i - 1 + a1) ^ 0xAA;
  result = a3 + i_1;
  *(_BYTE *)(a3 + i_1) = 0;
  return result;
}
```

eg.cyber_chef


Recipe
AES Decrypt
Key
venom2025v...
UTF8
IV
HEX
Mode
ECB
Input
Hex
Output
Raw
Salsa20
Key
0ff71ec39a5...
HEX
Nonce
salsa_iv
UTF8
Counter
0
Rounds
20
Input
Raw
Output
Raw
Input
59cc9e17b97199cad788f5f9d919531ee9f09bb233b5565233aefc
ABC 96
1
Output
vctf{ab92105b_047e_aa38_0b25_a6989b8e5ed5}

eg.code

Python

```python
import binascii
import struct
from Crypto.Cipher import AES,Salsa20

#aes_dec
# enc = [
#     0xc9, 0xcf, 0xce, 0xc9, 0xce, 0xcb, 0x92, 0xcb, 0xc9, 0x99, 0x9f, 0xc9, 0x9f, 0xcb, 0xc9, 0x93,
#     0xcc, 0xc9, 0x9d, 0x9e, 0xce, 0x93, 0x9b, 0x92, 0xc9, 0x99, 0x9a, 0x9d, 0x9d, 0x99, 0x98, 0x9f,
#     0x99, 0x9c, 0x9c, 0xce, 0x9b, 0xcb, 0x93, 0x99, 0xcf, 0xce, 0x9b, 0xcc, 0xcf, 0xcb, 0x99, 0x99,
#     0x98, 0x9f, 0x9c, 0x9f, 0x9f, 0xc8, 0x99, 0x99, 0x98, 0xc8, 0xc8, 0x93, 0x9a, 0xcc, 0x93, 0xcf,
#     0xcf, 0x9b, 0x99, 0x9f, 0x93, 0x9b, 0x93, 0xce, 0x93, 0xcc, 0x9f, 0xcc, 0x92, 0x92, 0x9d, 0xce,
#     0xcb, 0xc9, 0x93, 0x93, 0x9b, 0x9d, 0x93, 0xc8, 0x9d, 0x9b, 0xcf, 0x93, 0xc9, 0xc9, 0x93, 0x9f
# ]
# hoge=bytes(b ^ 0xAA for b in enc[::-1])
# print(hoge)
# b'59cc9e17b97199cad788f5f9d919531ee9f09bb233b5565233aef1de39a1d6635237703c819d47cf9ca5c53ca8adcdec'
hoge=bytes.fromhex('59cc9e17b97199cad788f5f9d919531ee9f09bb233b5565233aef1de39a1d6635237703c819d47cf9ca5c53ca8adcdec')
aes_key=b"venom2025venom25"
cipher_aes=AES.new(aes_key, AES.MODE_ECB)
pad=cipher_aes.decrypt(hoge)
pad_len=pad[-1]
#去除PKS7填充
enc_tmp=pad[:-pad_len]

#salsa20_dec
# nonce=0xD9CBC6D9CBF5C3DC
# salsa=[0x999D939CCF9F9B9F,0x9A93C89CC898CF93,0x93CB9FCE9998989D,0x9ACCCC9D9BCFC999,
#        0x999D939CCF9F9B9F,0x9A93C89CC898CF93,0x93CB9FCE9998989D,0x9ACCCC9D9BCFC999]
# hoge1=[]
# hoge2=[]
# hoge1.extend(nonce.to_bytes(8, 'little'))
# for qword in salsa:
#     hoge2.extend(qword.to_bytes(8, 'little'))
# nonce=bytes(b ^ 0xAA for b in hoge1[::-1])
```

```
# salsa_key=bytes(b ^ 0xAA for b in hoge2[::-1])
# print(nonce,salsa_key)
# b'salsa_iv'
# b'0ff71ec39a5d322709b6b2e93796e5150ff71ec39a5d322709b6b2e93796e515'
nonce=b'salsa_iv'
salsa_key=bytes.fromhex('0ff71ec39a5d322709b6b2e93796e5150ff71ec39a5d322709b6b2e93796e515')
cipher_salsa=Salsa20.new(key=salsa_key,nonce=nonce)
flag=cipher_salsa.decrypt(enc_tmp)
print(flag.decode())
# yctf{ab92105b-047e-aa38-0b25-a6989b8e5ed5}
```