CTF-SMC 逆向练习

原创

流浪打浪 于 2021-11-28 20:25:50 发布 2405 收藏
分类专栏： ctf逆向 文章标签： unctf网络安全

SMC代码修改逆向分析--小白学习

IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums

```
1  int __cdecl main_0(int argc, const char **argv, const char **envp)
2  {
3    int result; // eax
4    int i; // [esp+4Ch] [ebp-70h]
5    char Str; // [esp+54h] [ebp-68h] BYREF
6    char v6[96]; // [esp+55h] [ebp-67h] BYREF
7    __int16 v7; // [esp+B5h] [ebp-7h]
8    char v8; // [esp+B7h] [ebp-5h]
9    size_t v9; // [esp+B8h] [ebp-4h]
10
11   v9 = 0;
12   Str = 0;
13   memset(v6, 0, sizeof(v6));
14   v7 = 0;
15   v8 = 0;
16   scanf("%100s", &Str);
17   v9 = strlen(&Str);
18   if ( v9 == 28 )
19   {
20     if ( v6[26] == 125 )
21     {
22       for ( i = 0; i < 67; ++i )
23         byte_414C3C[i] ^= 0x7Du;
24       (*(void (__cdecl **)(char *, void *))byte_414C3C)(&Str, &unk_414BE0);
25       result = 0;
26     }
27     else
28     {
29       result = 0;
30     }
31   }
32   else
33   {
34     printf("Try Again......\n");
35     result = 0;
36   }
37   return result;
38 }
```

000010B4 _main_0:23 (4010B4)

先进行异或，再进行函数的调用，说明为SMC加密，可以写IDC脚本进行解密，脚本如下：

```
static xor_setpl(){
    auto addr = 0x00414c3c;    //这里填入要解密字节串的起始地址
    auto i = 0;
    for(i=0;addr+i<0x00414C7F;i++)    //循环结束的条件为字节串的结束地址
    {
        PatchByte(addr+i,Byte(addr+i)^0x7D);    //异或的数字根据情况修改
    }
}
```


脚本解密前：

```
.data:00414C3C ; char byte_414C3C[100]
.data:00414C3C byte_414C3C     db 28h                  ; DATA XREF: _main_0+97↑r
.data:00414C3C                                         ; _main_0+A4↑w ...
.data:00414C3D                 db 0F6h
.data:00414C3E                 db  91h
.data:00414C3F                 db 0F6h
.data:00414C40                 db  38h ; 8
.data:00414C41                 db  75h ; u
.data:00414C42                 db 0FDh
.data:00414C43                 db  45h ; E
.data:00414C44                 db  1Bh
.data:00414C45                 db    8
.data:00414C46                 db  49h ; I
.data:00414C47                 db 0FDh
.data:00414C48                 db    5
.data:00414C49                 db  7Ch ; |
.data:00414C4A                 db  11h
.data:00414C4B                 db    8
.data:00414C4C                 db  53h ; S
.data:00414C4D                 db 0FDh
.data:00414C4E                 db    5
.data:00414C4F                 db  7Fh ;
.data:00414C50                 db  1Ch
.data:00414C51                 db    8
.data:00414C52                 db  55h ; U
.data:00414C53                 db 0FDh
.data:00414C54                 db    5
.data:00414C55                 db  7Eh ; ~
.data:00414C56                 db  1Ah
.data:00414C57                 db    8
.data:00414C58                 db  5Fh ; _
.data:00414C59                 db 0FDh
.data:00414C5A                 db    5
.data:00414C5B                 db  79h ; y
.data:00414C5C                 db    6
.data:00414C5D                 db    8
.data:00414C5E                 db  61h ; a
.data:00414C5F                 db 0F6h
.data:00414C60                 db  28h ; (
.data:00414C61                 db  71h ; q
.data:00414C62                 db  4Eh ; N
.data:00414C63                 db 0B4h
.data:00414C64                 db 0FDh
.data:00414C65                 db  49h ; I
```


经过File----Select File 脚本解密后：

```
.data:00414C3C ; char byte_414C3C[100]
.data:00414C3C byte_414C3C     db 55h                  ; DATA XREF: _main_0+97↑r
.data:00414C3C                                         ; _main_0+A4↑w ...
.data:00414C3D                 db  8Bh
.data:00414C3E                 db 0ECh
.data:00414C3F                 db  8Bh
.data:00414C40                 db  45h ; E
.data:00414C41                 db    8
.data:00414C42                 db  80h ; €
.data:00414C43                 db  38h ; 8
.data:00414C44                 db  66h ; f
.data:00414C45                 db  75h ; u
.data:00414C46                 db  34h ; 4
.data:00414C47                 db  80h ; €
.data:00414C48                 db  78h ; x
.data:00414C49                 db    1
.data:00414C4A                 db  6Ch ; l
.data:00414C4B                 db  75h ; u
.data:00414C4C                 db  2Eh ; .
.data:00414C4D                 db  80h ; €
.data:00414C4E                 db  78h ; x
.data:00414C4F                 db    2
.data:00414C50                 db  61h ; a
.data:00414C51                 db  75h ; u
.data:00414C52                 db  28h ; (
.data:00414C53                 db  80h ; €
.data:00414C54                 db  78h ; x
.data:00414C55                 db    3
.data:00414C56                 db  67h ; g
.data:00414C57                 db  75h ; u
.data:00414C58                 db  22h ; "
.data:00414C59                 db  80h ; €
.data:00414C5A                 db  78h ; x
.data:00414C5B                 db    4
.data:00414C5C                 db  7Bh ; {
.data:00414C5D                 db  75h ; u
.data:00414C5E                 db  1Ch
.data:00414C5F                 db  8Bh
.data:00414C60                 db  55h ; U
.data:00414C61                 db  0Ch
.data:00414C62                 db  33h ; 3
.data:00414C63                 db 0C9h
.data:00414C64                 db  80h ; €
.data:00414C65                 db  34h ; 4
00014C3E 00414C3E: .data:00414C3E (Synchronized with Hex View-1)
```


按 c快捷键 转变成汇编语言，在 右键 --- Create Function 有

```
.data:00414C3C
.data:00414C3C arg_0           = dword ptr  8
.data:00414C3C arg_4           = dword ptr  0Ch
.data:00414C3C
.data:00414C3C                 push    ebp
.data:00414C3D                 mov     ebp, esp
.data:00414C3F                 mov     eax, [ebp+arg_0]
.data:00414C42                 cmp     byte ptr [eax], 66h ; 'f'
.data:00414C45                 jnz     short loc_414C7B
.data:00414C47                 cmp     byte ptr [eax+1], 6Ch ; 'l'
.data:00414C4B                 jnz     short loc_414C7B
.data:00414C4D                 cmp     byte ptr [eax+2], 61h ; 'a'
.data:00414C51                 jnz     short loc_414C7B
.data:00414C53                 cmp     byte ptr [eax+3], 67h ; 'g'
.data:00414C57                 jnz     short loc_414C7B
.data:00414C59                 cmp     byte ptr [eax+4], 7Bh ; '{'
.data:00414C5D                 jnz     short loc_414C7B
.data:00414C5F                 mov     edx, [ebp+arg_4]
.data:00414C62                 xor     ecx, ecx
.data:00414C64
.data:00414C64 loc_414C64:                             ; CODE XREF: sub_414C3C+30↓j
.data:00414C64                 xor     byte ptr [ecx+edx], 43h
.data:00414C68                 inc     ecx
.data:00414C69                 cmp     ecx, 5Ah ; 'Z'
.data:00414C6C                 jl      short loc_414C64
.data:00414C6E                 add     eax, 5
.data:00414C71                 push    offset unk_414A84
.data:00414C76                 push    eax
.data:00414C77                 call    edx
.data:00414C79                 pop     ecx
.data:00414C7A                 pop     ecx
.data:00414C7B
.data:00414C7B loc_414C7B:                             ; CODE XREF: sub_414C3C+9↑j
.data:00414C7B                                         ; sub_414C3C+F↑j ...
.data:00414C7B                 xor     eax, eax
.data:00414C7D                 pop     ebp
.data:00414C7E                 retn
.data:00414C7E sub_414C3C      endp
.data:00414C7E
.data:00414C7E ; ---------------------------------------------------------------------------
.data:00414C7F                 db    0
```



```
1  int __cdecl sub_414C3C(_BYTE *a1, _BYTE *a2)
2  {
3    int i; // ecx
4
5    if ( *a1 == 102 && a1[1] == 108 && a1[2] == 97 && a1[3] == 103 && a1[4] == 123 )
6    {
7      for ( i = 0; i < 90; ++i )
8        a2[i] ^= 0x43u;
9      ((void (__cdecl *)(_BYTE *, void *))a2)(a1 + 5, &unk_414A84);
10   }
11   return 0;
12 }
```


可以看出这里对输入的前5位有要求，按R快捷键得出前五为

```
1  int __cdecl sub_414C3C(_BYTE *a1, _BYTE *a2)
2  {
3    int i; // ecx
4
5    if ( *a1 == 'f' && a1[1] == 'l' && a1[2] == 'a' && a1[3] == 'g' && a1[4] == '{' )// 前五位为flag{
6    {
7      for ( i = 0; i < 90; ++i )
8        a2[i] ^= 0x43u;                           // 对第二层加密的数据循环90次与0x43进行异或，
9                                                  // 这里第二层加密的数据为前面入栈的&unk_414BE0
10     ((void (__cdecl *)(_BYTE *, void *))a2)(a1 + 5, &unk_414A84);
11   }
12   return 0;
13 }
```


对第二层进行解密，解密脚本为

```
static xor_setp2(){
    auto addr = 0x00414be0;   //这里填入要解密字节串的起始地址
    auto i = 0;
    for(i=0;addr+i<0x00414C3A;i++)   //循环结束的条件为字节串的结束地址
    {
        PatchByte(addr+i,Byte(addr+i)^0x43);   //异或的数字根据情况修改
    }
}
```


脚本解密前：

```
.data:00414BDF                 db    0
.data:00414BE0 unk_414BE0      db  16h                 ; DATA XREF: _main_0+B5↑o
.data:00414BE1                 db 0C8h
.data:00414BE2                 db 0AFh
.data:00414BE3                 db  12h
.data:00414BE4                 db  12h
.data:00414BE5                 db  10h
.data:00414BE6                 db 0C8h
.data:00414BE7                 db  1Eh
.data:00414BE8                 db  4Bh ; K
.data:00414BE9                 db 0CEh
.data:00414BEA                 db    6
.data:00414BEB                 db 0BBh
.data:00414BEC                 db  15h
.data:00414BED                 db  14h
.data:00414BEE                 db  70h ; p
.data:00414BEF                 db  91h
.data:00414BF0                 db  84h
.data:00414BF1                 db    6
.data:00414BF2                 db 0BBh
.data:00414BF3                 db 0DBh
.data:00414BF4                 db 0E7h
.data:00414BF5                 db 0EAh
.data:00414BF6                 db 0D0h
.data:00414BF7                 db 0C8h
.data:00414BF8                 db 0B8h
.data:00414BF9                 db 0CBh
.data:00414BFA                 db  16h
.data:00414BFB                 db 0BFh
.data:00414BFC                 db 0C8h
.data:00414BFD                 db 0B1h
.data:00414BFE                 db  68h ; h
.data:00414BFF                 db 0BBh
.data:00414C00                 db 0CEh
.data:00414C01                 db  0Eh
.data:00414C02                 db 0BBh
.data:00414C03                 db  40h ; @
.data:00414C04                 db  8Dh
.data:00414C05                 db 0C9h
.data:00414C06                 db  47h ; G
.data:00414C07                 db  4Ch ; L
00014BE0 00414BE0: .data:unk_414BE0 (Synchronized with Hex View-1)
```


经过File----Select File 脚本解密后 按 C快捷键有：

```
.data:00414BE0 ; ---------------------------------------------------------------------------
.data:00414BE0
.data:00414BE0 loc_414BE0:                             ; DATA XREF: _main_0+B5↑o
.data:00414BE0                 push    ebp
.data:00414BE1                 mov     ebp, esp
.data:00414BE3                 push    ecx
.data:00414BE4                 push    ecx
.data:00414BE5                 push    ebx
.data:00414BE5 ; ---------------------------------------------------------------------------
.data:00414BE6                 db  8Bh
.data:00414BE7                 pop     ebp
.data:00414BE8                 or      [ebp+5756F845h], cl
.data:00414BEE                 xor     edx, edx
.data:00414BF0                 mov     dword ptr [ebp-8], 93A9A498h
.data:00414BF7                 mov     edi, ebx
.data:00414BF7 ; ---------------------------------------------------------------------------
.data:00414BF9                 db  88h
.data:00414BFA                 db  55h ; U
.data:00414BFB                 cld
.data:00414BFC                 mov     esi, edx
.data:00414BFE                 sub     edi, eax
.data:00414C00                 lea     ecx, [ebp-8]
.data:00414C03                 add     ecx, esi
.data:00414C03 ; ---------------------------------------------------------------------------
.data:00414C05                 db  8Ah
.data:00414C06                 db    4
.data:00414C07 ; ---------------------------------------------------------------------------
.data:00414C07                 sysenter
.data:00414C09                 int     3               ; Trap to Debugger
.data:00414C0A                 cmp     al, [ecx]
.data:00414C0C                 jnz     short near ptr loc_414C2D+4
.data:00414C0E                 inc     esi
.data:00414C0F                 cmp     esi, 4
.data:00414C0F ; ---------------------------------------------------------------------------
.data:00414C12                 db  7Ch ; |
.data:00414C13                 in      al, dx
.data:00414C14                 mov     ecx, [ebp+0Ch]
.data:00414C14 ; ---------------------------------------------------------------------------
.data:00414C17 unk_414C17      db  80h ; €             ; CODE XREF: .data:00414C22↓j
.data:00414C18                 xor     al, 0Ah
```


（上图黄色部分 需要一个一个按 C快捷键转化成汇编代码， Create Function才能成功）

在 右键 --- Create Function，转换成伪C 代码：

```
.data:00414BE0
.data:00414BE0 sub_414BE0      proc near               ; DATA XREF: _main_0+B5↑o
.data:00414BE0
.data:00414BE0 var_8           = dword ptr -8
.data:00414BE0 var_4           = byte ptr -4
.data:00414BE0 arg_0           = dword ptr  8
.data:00414BE0
.data:00414BE0                 push    ebp
.data:00414BE1                 mov     ebp, esp
.data:00414BE3                 push    ecx
.data:00414BE4                 push    ecx
.data:00414BE5                 push    ebx
.data:00414BE6                 mov     ebx, [ebp+arg_0]
.data:00414BE9                 lea     eax, [ebp+var_8]
.data:00414BEC                 push    esi
.data:00414BED                 push    edi
.data:00414BEE                 xor     edx, edx
.data:00414BF0                 mov     [ebp+var_8], 93A9A498h
.data:00414BF7                 mov     edi, ebx
.data:00414BF9                 mov     [ebp+var_4], dl
.data:00414BFC                 mov     esi, edx
.data:00414BFE                 sub     edi, eax
.data:00414C00
.data:00414C00 loc_414C00:                             ; CODE XREF: sub_414BE0+32↓j
.data:00414C00                 lea     ecx, [ebp+var_8]
.data:00414C03                 add     ecx, esi
.data:00414C05                 mov     al, [edi+ecx]
.data:00414C08                 xor     al, 0CCh
.data:00414C0A                 cmp     al, [ecx]
.data:00414C0C                 jnz     short loc_414C31
.data:00414C0E                 inc     esi
.data:00414C0F                 cmp     esi, 4
.data:00414C12                 jl      short loc_414C00
.data:00414C14                 mov     ecx, [ebp+0Ch]
.data:00414C17
.data:00414C17 loc_414C17:                             ; CODE XREF: sub_414BE0+42↓j
.data:00414C17                 xor     byte ptr [edx+ecx], 55h
.data:00414C1B                 inc     edx
.data:00414C1C                 cmp     edx, 15Bh
.data:00414C22                 jl      short loc_414C17
.data:00414C24                 lea     eax, [ebx+4]
.data:00414C27                 push    offset unk_414A30
.data:00414C2C                 push    eax
.data:00414C2D                 call    ecx

00014C0C 00414C0C: sub_414BE0+2C (Synchronized with Hex View-1)
```



```
1  int __cdecl sub_414BE0(int a1, void (__cdecl *a2)(int, void *))
2  {
3    int v2; // edx
4    int v3; // esi
5    int v5; // [esp+Ch] [ebp-8h] BYREF
6    char v6; // [esp+10h] [ebp-4h]
7
8    v2 = 0;
9    v5 = 0x93A9A498;
10   v6 = 0;
11   v3 = 0;
12
13   while ( (*((_BYTE *)&v5 + v3 + a1 - (_DWORD)&v5) ^ 0xCC) == *((_BYTE *)&v5 + v3) )// 这里 a1 就除去前五位的数
14                                                 // 可知while代码执行的条件为:  *(a1+v3)^0xCC == (&v5+v3),
15   {
16
17     if ( ++v3 >= 4 )
18     {
19       do
20         *((_BYTE *)a2 + v2++) ^= 0x55u;         // 对第三层与0x55进行加密
21       while ( v2 < 347 );
22       a2(a1 + 4, &unk_414A30);                  // // unk_414A30 第四层加密数据
23                                                 //
24       return 0;
25     }
26   }
27   return 0;
28 }
```


从调用地方可以看出a1 为 除去前五位后剩下的数

```
int __cdecl sub_414C3C(_BYTE *a1, _BYTE *a2)
{
  int i; // ecx

  if ( *a1 == 'f' && a1[1] == 'l' && a1[2] == 'a' && a1[3] == 'g' && a1[4] == '{' )// 前五位为flag{
  {
    for ( i = 0; i < 90; ++i )
      a2[i] ^= 0x43u;                           // 对第二层加密的数据循环90次与0x43进行异或，
                                                // 这里第三层加密的数据为前面入栈的&unk_414BE0
    ((void (__cdecl *)(_BYTE *, void *))a2)(a1 + 5, &unk_414A84);
  }
  return 0;
}
```


根据while循环写出脚本对4位进行解密，脚本如下

test1.py

```
flag=''
xor = [0x93,0xA9,0xA4,0X98]
for i in range(len(xor)):
    flag += chr(xor[i] ^ 0xCC)
print(flag)
```


得到结果

```
PS C:\Users\hanye\Desktop\攻防世界\SMC2> & "D:/Program Files (x86)/P
_ehT
```


因为存储是从右到左，所以为 The_

对第三层进行解密，解密脚本为

```
static xor_setp3(){
    auto addr = 0x00414a84;    //这里填入要解密字节串的起始地址
    auto i = 0;
    for(i=0;addr+i<0x00414a84 + 347;i++)    //循环结束的条件为字节串的结束地址
    {
        PatchByte(addr+i,Byte(addr+i)^0x55);    //异或的数字根据情况修改
    }
}
```


第三层加密数据：

```
int __cdecl sub_414C3C(_BYTE *a1, _BYTE *a2)
{
  int i; // ecx

  if ( *a1 == 'f' && a1[1] == 'l' && a1[2] == 'a' && a1[3] == 'g' && a1[4] == '{' )// 前五位为flag{
  {
    for ( i = 0; i < 90; ++i )
      a2[i] ^= 0x43u;                           // 对第二层加密的数据循环90次与0x43进行异或，
                                                // 这里第三层加密的数据为前面入栈的&unk_414BE0
    ((void (__cdecl *)(_BYTE *, void *))a2)(a1 + 5, &unk_414A84);
  }
  return 0;
}
```

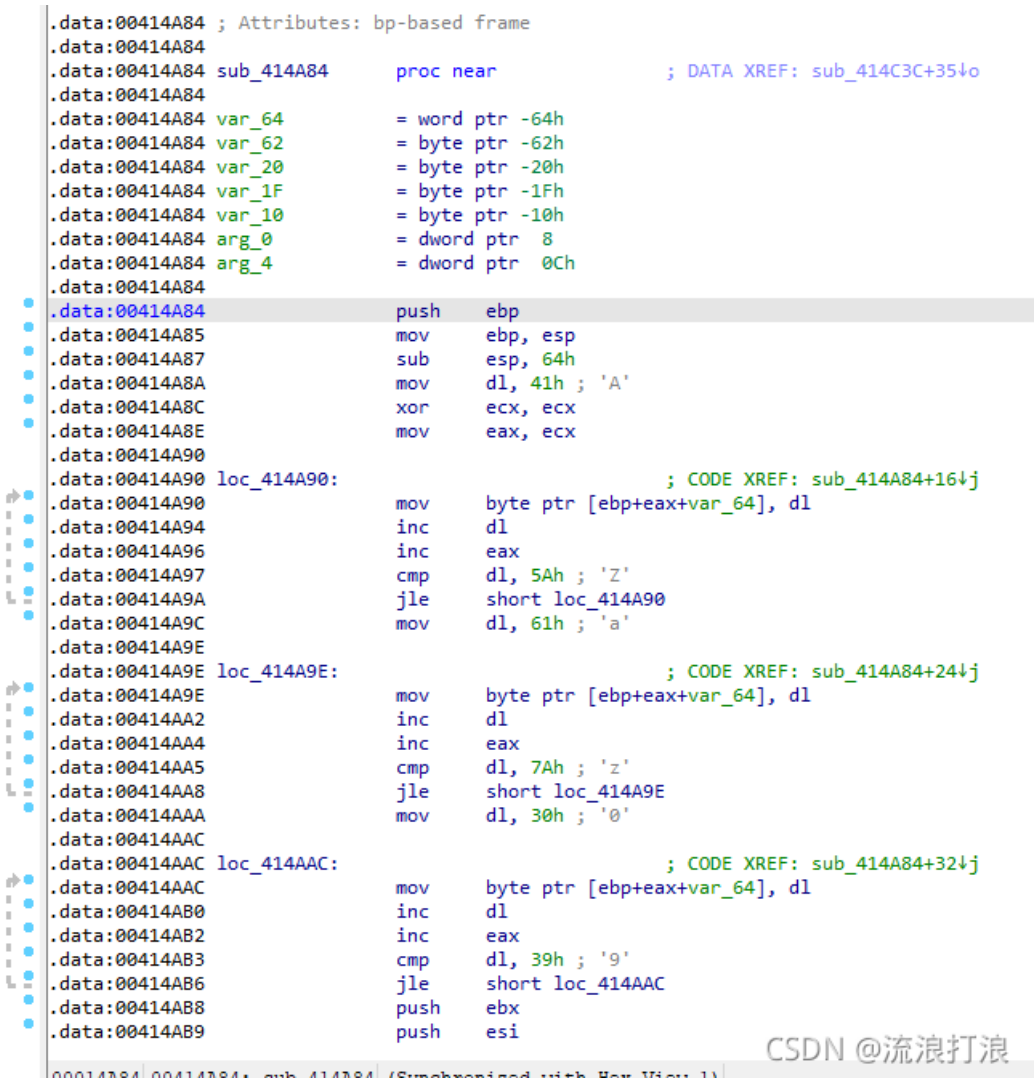
解密前：

```
.data:00414A83                  db    0
.data:00414A84 unk_414A84       db    0                 ; DATA XREF: sub_414C3C+35↓o
.data:00414A85                  db 0DEh
.data:00414A86                  db 0B9h
.data:00414A87                  db 0D6h
.data:00414A88                  db 0B9h
.data:00414A89                  db  31h ; 1
.data:00414A8A                  db 0E7h
.data:00414A8B                  db  14h
.data:00414A8C                  db  66h ; f
.data:00414A8D                  db  9Ch
.data:00414A8E                  db 0DEh
.data:00414A8F                  db  94h
.data:00414A90                  db 0DDh
.data:00414A91                  db    1
.data:00414A92                  db  50h ; P
.data:00414A93                  db 0C9h
.data:00414A94                  db 0ABh
.data:00414A95                  db  97h
.data:00414A96                  db  15h
.data:00414A97                  db 0D5h
.data:00414A98                  db 0AFh
.data:00414A99                  db  0Fh
.data:00414A9A                  db  2Bh ; +
.data:00414A9B                  db 0A1h
.data:00414A9C                  db 0E7h
.data:00414A9D                  db  34h ; 4
.data:00414A9E                  db 0DDh
.data:00414A9F                  db    1
.data:00414AA0                  db  50h ; P
.data:00414AA1                  db 0C9h
.data:00414AA2                  db 0ABh
.data:00414AA3                  db  97h
.data:00414AA4                  db  15h
.data:00414AA5                  db 0D5h
.data:00414AA6                  db 0AFh
.data:00414AA7                  db  2Fh ; /
.data:00414AA8                  db  2Bh ; +
.data:00414AA9                  db 0A1h
.data:00414AAA                  db 0E7h
.data:00414AAB                  db  65h ; e
```


解密后：

```
.data:00414A83                  db    0
.data:00414A84 unk_414A84       db  55h ; U             ; DATA XREF: sub_414C3C+35↓o
.data:00414A85                  db  8Bh
.data:00414A86                  db 0ECh
.data:00414A87                  db  83h
.data:00414A88                  db 0ECh
.data:00414A89                  db  64h ; d
.data:00414A8A                  db 0B2h
.data:00414A8B                  db  41h ; A
.data:00414A8C                  db  33h ; 3
.data:00414A8D                  db 0C9h
.data:00414A8E                  db  8Bh
.data:00414A8F                  db 0C1h
.data:00414A90                  db  88h
.data:00414A91                  db  54h ; T
.data:00414A92                  db    5
.data:00414A93                  db  9Ch
.data:00414A94                  db 0FEh
.data:00414A95                  db 0C2h
.data:00414A96                  db  40h ; @
.data:00414A97                  db  80h ; €
.data:00414A98                  db 0FAh
.data:00414A99                  db  5Ah ; Z
.data:00414A9A                  db  7Eh ; ~
.data:00414A9B                  db 0F4h
.data:00414A9C                  db 0B2h
.data:00414A9D                  db  61h ; a
.data:00414A9E                  db  88h
.data:00414A9F                  db  54h ; T
.data:00414AA0                  db    5
.data:00414AA1                  db  9Ch
.data:00414AA2                  db 0FEh
.data:00414AA3                  db 0C2h
.data:00414AA4                  db  40h ; @
.data:00414AA5                  db  80h ; €
.data:00414AA6                  db 0FAh
.data:00414AA7                  db  7Ah ; z
.data:00414AA8                  db  7Eh ; ~
.data:00414AA9                  db 0F4h
.data:00414AAA                  db 0B2h
.data:00414AAB                  db  30h ; 0
```


转换成汇编


.data:00414A84 ; Attributes: bp-based frame
.data:00414A84
.data:00414A84 sub_414A84 proc near ; DATA XREF: sub_414C3C+35↓o
.data:00414A84
.data:00414A84 var_64 = word ptr -64h
.data:00414A84 var_62 = byte ptr -62h
.data:00414A84 var_20 = byte ptr -20h
.data:00414A84 var_1F = byte ptr -1Fh
.data:00414A84 var_10 = byte ptr -10h
.data:00414A84 arg_0 = dword ptr 8
.data:00414A84 arg_4 = dword ptr 0Ch
.data:00414A84
.data:00414A84 push ebp
.data:00414A85 mov ebp, esp
.data:00414A87 sub esp, 64h
.data:00414A8A mov dl, 41h ; 'A'
.data:00414A8C xor ecx, ecx
.data:00414A8E mov eax, ecx
.data:00414A90
.data:00414A90 loc_414A90: ; CODE XREF: sub_414A84+16↓j
.data:00414A90 mov byte ptr [ebp+eax+var_64], dl
.data:00414A94 inc dl
.data:00414A96 inc eax
.data:00414A97 cmp dl, 5Ah ; 'Z'
.data:00414A9A jle short loc_414A90
.data:00414A9C mov dl, 61h ; 'a'
.data:00414A9E
.data:00414A9E loc_414A9E: ; CODE XREF: sub_414A84+24↓j
.data:00414A9E mov byte ptr [ebp+eax+var_64], dl
.data:00414AA2 inc dl
.data:00414AA4 inc eax
.data:00414AA5 cmp dl, 7Ah ; 'z'
.data:00414AA8 jle short loc_414A9E
.data:00414AAA mov dl, 30h ; '0'
.data:00414AAC
.data:00414AAC loc_414AAC: ; CODE XREF: sub_414A84+32↓j
.data:00414AAC mov byte ptr [ebp+eax+var_64], dl
.data:00414AB0 inc dl
.data:00414AB2 inc eax
.data:00414AB3 cmp dl, 39h ; '9'
.data:00414AB6 jle short loc_414AAC
.data:00414AB8 push ebx
.data:00414AB9 push esi
CSDN @流浪打浪

转化成伪c代码

```
char v23[16]; // [esp+54h] [ebp-10h] BYREF

v2 = 65;
v3 = 0;
do
  *((_BYTE *)v20 + v3++) = v2++;
while ( v2 <= 90 );
for ( i = 97; i <= 122; ++i )
  *((_BYTE *)v20 + v3++) = i;
for ( j = 48; j <= 57; ++j )
  *((_BYTE *)v20 + v3++) = j;
*(__int16 *)((char *)v20 + v3) = 12075;
*((_BYTE *)&v20[1] + v3) = 0;
v6 = &v21;
v21 = 0;
v22[0] = 0;
strcpy(v23, "cmVhbEN0Rl8=");                    // 传入了固定字符串"cmVhbEN0Rl8="推测为flag的第10到17位
v22[1] = 0;
v22[2] = 0;
v7 = 0;
do
{
  v8 = *(_BYTE *)(a1 + v7 + 1);
  v9 = *(_BYTE *)(a1 + v7 + 2);
  v10 = *(_BYTE *)(a1 + v7) & 3;
  v11 = *(unsigned __int8 *)(a1 + v7) >> 2;
  v7 += 3;
  *v6 = *((_BYTE *)v20 + v11);
  v6[1] = *((_BYTE *)v20 + ((v8 >> 4) | (16 * v10)));
  v6[2] = *((_BYTE *)v20 + ((v9 >> 6) | (4 * (v8 & 0xF))));
  v6[3] = *((_BYTE *)v20 + (v9 & 0x3F));
  v6 += 4;
}
while ( v7 < 6 );
if ( v7 < 8 )
{
  v12 = *(_BYTE *)(v7 + a1);
  *v6 = *((_BYTE *)v20 + (v12 >> 2));
  v13 = v12 & 3;
  if ( v7 == 7 )
  {
    v6[1] = v20[8 * v13];
    v6[2] = 61;
```

00014AEB sub_414A84:41 (414AEB)


```
57  if ( v7 < 8 )
58  {
59    v12 = *(_BYTE *)(v7 + a1);
60    *v6 = *((_BYTE *)v20 + (v12 >> 2));
61    v13 = v12 & 3;
62    if ( v7 == 7 )
63    {
64      v6[1] = v20[8 * v13];
65      v6[2] = 61;
66    }
67    else
68    {
69      v14 = *(_BYTE *)(v7 + a1 + 1);
70      v6[1] = *((_BYTE *)v20 + ((16 * v13) | (v14 >> 4)));
71      v6[2] = v20[2 * (v14 & 0xF)];
72    }
73    v15 = v6 + 3;
74    *v15 = 61;
75    v6 = v15 + 1;
76  }
77  v16 = 0;
78  *v6 = 0;
79  v17 = 0;
80  v18 = v21;
81  while ( v18 == v23[v17] )                     // 与前面base64 进行比较
82  {
83    v18 = *((_BYTE *)v22 + v17++);
84    if ( !v18 )
85    {
86      do
87        *((_BYTE *)a2 + v16++) ^= 0x4Du;
88      while ( v16 < 83 );
89      a2(a1 + 8);
90      return 0;
91    }
92  }
93  return 0;
94}
```

base64解密

进入第四层解密，解密脚本如下：

```
static xor_setp4(){
    auto addr = 0x00414a30;   //这里填入要解密字节串的起始地址
    auto i = 0;
    for(i=0;addr+i<0x00414a30 + 83;i++)   //循环结束的条件为字节串的结束地址
    {
        PatchByte(addr+i,Byte(addr+i)^0x4D);   //异或的数字根据情况修改
    }
}
```


解密前：

```
.data:00414A2F                 db    0
.data:00414A30 unk_414A30      db  18h                 ; DATA XREF: sub_414BE0+47↓o
.data:00414A31                 db 0C6h
.data:00414A32                 db 0A1h
.data:00414A33                 db 0CEh
.data:00414A34                 db 0A1h
.data:00414A35                 db  41h ; A
.data:00414A36                 db  1Eh
.data:00414A37                 db  7Eh ; ~
.data:00414A38                 db  96h
.data:00414A39                 db  8Ah
.data:00414A3A                 db    8
.data:00414A3B                 db 0B9h
.data:00414A3C                 db  26h ; &
.data:00414A3D                 db  3Bh ; ;
.data:00414A3E                 db  39h ; 9
.data:00414A3F                 db  38h ; 8
.data:00414A40                 db  1Bh
.data:00414A41                 db  1Ah
.data:00414A42                 db  8Ah
.data:00414A43                 db    8
.data:00414A44                 db 0B5h
.data:00414A45                 db  2Dh ; -
.data:00414A46                 db  0Eh
.data:00414A47                 db  79h ; y
.data:00414A48                 db  25h ; %
.data:00414A49                 db 0CEh
.data:00414A4A                 db  87h
.data:00414A4B                 db 0B2h
.data:00414A4C                 db  2Bh ; +
.data:00414A4D                 db  8Ah
.data:00414A4E                 db    8
.data:00414A4F                 db 0B1h
.data:00414A50                 db  6Fh ; o
.data:00414A51                 db  22h ; "
.data:00414A52                 db 0C5h
.data:00414A53                 db  10h
.data:00414A54                 db 0B3h
.data:00414A55                 db  0Fh
.data:00414A56                 db  75h ; u
.data:00414A57                 db  11h
```


解密后：

```
.data:00414A2F                   db    0
.data:00414A30 unk_414A30        db  55h ; U             ; DATA XREF: sub_414BE0+47↓o
.data:00414A31                   db  8Bh
.data:00414A32                   db 0ECh
.data:00414A33                   db  83h
.data:00414A34                   db 0ECh
.data:00414A35                   db  0Ch
.data:00414A36                   db  53h ; S
.data:00414A37                   db  33h ; 3
.data:00414A38                   db 0DBh
.data:00414A39                   db 0C7h
.data:00414A3A                   db  45h ; E
.data:00414A3B                   db 0F4h
.data:00414A3C                   db  6Bh ; k
.data:00414A3D                   db  76h ; v
.data:00414A3E                   db  74h ; t
.data:00414A3F                   db  75h ; u
.data:00414A40                   db  56h ; V
.data:00414A41                   db  57h ; W
.data:00414A42                   db 0C7h
.data:00414A43                   db  45h ; E
.data:00414A44                   db 0F8h
.data:00414A45                   db  60h ; `
.data:00414A46                   db  43h ; C
.data:00414A47                   db  34h ; 4
.data:00414A48                   db  68h ; h
.data:00414A49                   db  83h
.data:00414A4A                   db 0CAh
.data:00414A4B                   db 0FFh
.data:00414A4C                   db  66h ; f
.data:00414A4D                   db 0C7h
.data:00414A4E                   db  45h ; E
.data:00414A4F                   db 0FCh
.data:00414A50                   db  22h ; "
.data:00414A51                   db  6Fh ; o
.data:00414A52                   db  88h
.data:00414A53                   db  5Dh ; ]
.data:00414A54                   db 0FEh
.data:00414A55                   db  42h ; B
.data:00414A56                   db  38h ; 8
.data:00414A57                   db  5Ch ; \
```


转为汇编

```
.data:00414A30 ; ---------------------------------------------------------------------------
.data:00414A30
.data:00414A30 loc_414A30:                             ; DATA XREF: sub_414BE0+47↓o
.data:00414A30                 push    ebp
.data:00414A31                 mov     ebp, esp
.data:00414A33                 sub     esp, 0Ch
.data:00414A36                 push    ebx
.data:00414A37                 xor     ebx, ebx
.data:00414A39                 mov     dword ptr [ebp-0Ch], 7574766Bh
.data:00414A40                 push    esi
.data:00414A41                 push    edi
.data:00414A42                 mov     dword ptr [ebp-8], 68344360h
.data:00414A49                 or      edx, 0FFFFFFFFh
.data:00414A4C                 mov     word ptr [ebp-4], 6F22h
.data:00414A52                 mov     [ebp-2], bl
.data:00414A55
.data:00414A55 loc_414A55:                             ; CODE XREF: .data:00414A5A↓j
.data:00414A55                 inc     edx
.data:00414A56                 cmp     [ebp+edx-0Ch], bl
.data:00414A5A                 jnz     short loc_414A55
.data:00414A5C                 test    edx, edx
.data:00414A5E                 jle     short loc_414A7A
.data:00414A60                 mov     esi, [ebp+8]
.data:00414A63                 lea     edi, [ebp-0Ch]
.data:00414A66                 sub     edi, esi
.data:00414A68
.data:00414A68 loc_414A68:                             ; CODE XREF: .data:00414A78↓j
.data:00414A68                 movsx   ecx, byte ptr [edi+esi]
.data:00414A6C                 movsx   eax, byte ptr [esi]
.data:00414A6F                 dec     ecx
.data:00414A70                 cmp     eax, ecx
.data:00414A72                 jnz     short loc_414A7A
.data:00414A74                 inc     ebx
.data:00414A75                 inc     esi
.data:00414A76                 cmp     ebx, edx
.data:00414A78                 jl      short loc_414A68
.data:00414A7A
.data:00414A7A loc_414A7A:                             ; CODE XREF: .data:00414A5E↑j
.data:00414A7A                                         ; .data:00414A72↑j
.data:00414A7A                 pop     edi
```

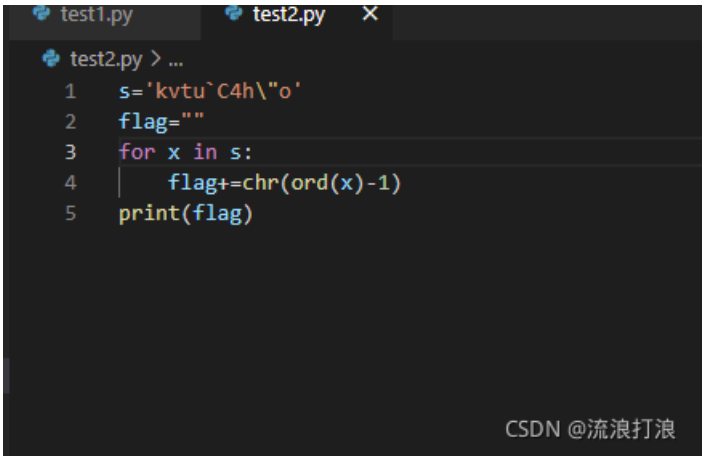

伪c代码为：

```
1  int __cdecl sub_414A30(char *a1)
2  {
3    int v1; // ebx
4    int v2; // edx
5    char *v3; // esi
6    char v5[12]; // [esp+Ch] [ebp-Ch] BYREF
7
8    v1 = 0;
9    strcpy(v5, "kvtu`C4h\"o");
10   v2 = -1;
11   do
12     ++v2;
13   while ( v5[v2] );
14   if ( v2 > 0 )
15   {
16     v3 = a1;
17     do
18     {
19       if ( *v3 != v3[v5 - a1] - 1 )
20         break;
21       ++v1;
22       ++v3;
23     }
24     while ( v1 < v2 );
25   }
26   return 0;
27 }
```

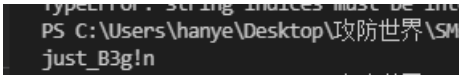
```
int __cdecl sub_414A30(char *a1)
{
  int v1; // ebx
  int v2; // edx
  char *v3; // esi
  char v5[12]; // [esp+Ch] [ebp-Ch] BYREF

  v1 = 0;
  strcpy(v5, "kvtu`C4h\"o");                    // 字符串为"kvtu`C4h\"o"
                                                //
  v2 = -1;
  do
    ++v2;
  while ( v5[v2] );
  if ( v2 > 0 )
  {
    v3 = a1;
    do
    {
      if ( *v3 != v3[v5 - a1] - 1 )             // 将参数地址所有的字符减1
        break;
      ++v1;
      ++v3;
    }
    while ( v1 < v2 );                          // 字符比较完后就退出
  }
  return 0;
}
```


python脚本如下：



结果为：



综上flag为 ：flag{The_realCtF_just_B3g!n}

参考：SMC代码自修改逆向分析（仅针对汇编语言分析）_Y1seco的博客-CSDN博客