

HWS WP

Reverse

SEA

刚开始进去有一个迷惑的 md5 比对（而且没有爆破的可能性），然后看见有一个 tryLevel。很自然的想到 SEH，patch 一下 jmp 到对应的 catch 块 IDA 就可以正常反汇编了。

```
rewind(v9);
ms_exc.registration.TryLevel = 0;
strcpy(v13, "e1a93e6635de60d70150a565d68f393b");
strlen(v14);
v4 = v14;
v5 = &v15;
do
{
    v6 = *v4;
    *v5++ = *v4++;
}
while ( v6 );
for ( i = 0; i < 43; ++i )
    aFpeb[i] ^= 0x11u;
puts(aFpeb);
for ( j = 0; j < 34; ++j )
    aD3c[j] ^= 0x13u;
puts(aD3c);
scanf("%s", &v16);
v9 = _iob_func();
rewind(v9);
if ( sub_FA2560((int)&v16) )
{
    for ( k = 0; k < 25; ++k )
        byte_FA4068[k] ^= 0x15u;
    ms_exc.registration.Next = (struct _EH3_EXCEPTION_REGISTRATION *)byte_FA4068;
}
else
{
    for ( m = 0; m < 24; ++m )
        *(&unk_FA4084 + m) ^= 0x17u;
    ms_exc.registration.Next = (struct _EH3_EXCEPTION_REGISTRATION *)&unk_FA4084;
```

进入正确的程序后，前几个字符比对比较简单，都是加减法和异或

```
v20[4] = v2;
```

```
Destination[0] = (Destination[0] + 33) ^ 0x16;
```

```
v20[5] = v3;
```

```
Destination[1] = (Destination[1] ^ 25) - 18;
```

```
Destination[3] = (Destination[3] ^ 25) - 18;
```

```
Destination[2] = (Destination[2] + 33) ^ 0x16;
```

```
Destination[4] = (Destination[4] + 33) ^ 0x16;
```

```
Destination[6] = (Destination[6] + 33) ^ 0x16;
```

```
Destination[5] = (Destination[5] ^ 0x19) - 18;
```

```
Destination[7] = (Destination[7] ^ 0x19) - 18;
```

```
Destination[9] = (Destination[9] ^ 0x19) - 18;
```

```
Destination[8] = (Destination[8] + 33) ^ 0x16;
```

```
v4 = 0;
```

```
while ( *((unsigned __int8 *)v20 + v4) == Destination[v4] )
```

```
{
```

```
    if ( ++v4 >= 10 )
```

```
{
```

```

v19 = v19;
v16 = v6;
keyExpansion(v12);
sub_FA1000(&v17);
sub_FA1320(&v17);
v7 = 9;
do
{
    v3 += 16;
    sub_FA1380();
    sub_FA1470(&v17);
    sub_FA1580(&v17);
    sub_FA1320(&v17);
    --v7;
}
while ( v7 );
sub_FA1380();
sub_FA1470(&v17);
sub_FA1320(&v17);
v8 = 0;
v9 = (_BYTE *) (a1 + 2);
do
{
    *(v9 - 2) = *(&v17 + v8);
    *(v9 - 1) = *((_BYTE *)&v18 + v8 + 3);
    *v9 = *((_BYTE *)&v19 + v8 + 3);
    v9[1] = *((_BYTE *)&v20 + v8++ + 3);
    v9 += 4;
}

```

```

a:00FA40A0 00 00 00 00 00 00 00 00 00 00+ ; 00FA40A0: 005E127010
a:00FA40B0 18 79 28 F9 55 99 71 D5 18 EC+byte_FA40E0 db 18h, 79h, 28h, 0F9h, 55h, 99h, 71h, 0D5h, 18h, 0ECh, 0BBh, 0B0h, 95h, 6Fh, 94h, 70h ; sub_FA1910+6B10
a:00FA40C0 BB 80 95 6F 94 70 A3 53 63 AD+ ; DATA XREF: 005E10F41r
a:00FA40D0 54 7B 37 6E C1 DB B1 D7 3D 92+ ; 005E10FE1r
a:00FA40E0 4D D0 14 4C B7 78 62 A0 6A 1A+ ; 005E11181r
a:00FA40F0 CD 00 F1 7D 5E 1E F5 8D 11 65+ ; 005E11251r
a:00FA4100 A9 E2 F2 B9 CA 8C A1 D2 47 AB+ ; 005E11531r
a:00FA4110 7C 66 52 E4 06 77 89 C6 7E B3+ ; 005E115E1r
a:00FA4120 AE E6 B4 8B DF 1D 23 17 EA 3C+ ; 005E11731r
a:00FA4130 90 DC 81 32 A5 AF 50 2D 5D 2D+ ; 005E11801r
a:00FA4140 96 42 35 2E 0A BF ED 8E 38 BA+ ; 005E11B01r
a:00FA4150 61 0B 85 5B 24 6B F0 21 3F CE+ ; 005E11BB1r
a:00FA4160 2B 22 A8 C5 E1 4A 30 74 EF CF+ ; 005E11CD1r
a:00FA4170 A4 D3 C8 D9 EB FB A7 BE 3E 41+ ; 005E11DD1r
a:00FA4180 E0 B5 9F C0 AC 93 9E F8 F7 7F+ ; 005E120E1r
a:00FA4190 DE 3B DA 72 88 0D 56 E8 E7 8A+ ; 005E12191r
a:00FA41A0 F4 91 5A 64 19 67 57 D8 84 FA+ ; 005E122E1r ...
a:00FA41B0 0C 25 9B A2 07 15 04 C4 87 43+db 14h, 4Ch, 0B7h, 78h, 62h, 0A0h, 6Ah, 1Ah, 0CDh, 0, 0F1h, 7Dh, 5Eh, 1Eh, 0F5h, 8Dh, 11h, 65h, 0 ; 00FA41B0: 005E127010
a:00FA41C0 97 B8 60 E3 45 AA 8F 13 FD CB+db 6, 77h, 89h, 0C6h, 7Eh, 0B3h, 0AEh, 0E6h, 0B4h, 8Bh, 0DFh, 1Dh, 23h, 17h, 0EAh, 3Ch, 90h, 0DC ; 00FA41C0: 005E127010
a:00FA41D0 2C A6 1C 3A EE 36 7A E9 D1 09+db 0EDh, 8Eh, 38h, 0BAh, 61h, 0Bh, 85h, 5Bh, 24h, 6Bh, 0F0h, 21h, 3Fh, 0CEh, 2Bh, 22h, 0A8h, 0C5 ; 00FA41D0: 005E127010
a:00FA41E0 39 4E 33 FE 9A 5C 86 6D 16 2F+db 3Eh, 41h, 0E0h, 0B5h, 9Fh, 0C0h, 0ACh, 93h, 9Eh, 0F8h, 0F7h, 7Fh, 0DEh, 3Bh, 0DAh, 72h, 88h, 0 ; 00FA41E0: 005E127010
a:00FA41F0 D4 B2 48 82 5F 68 29 03 C9 02+db 0Ch, 25h, 9Bh, 0A2h, 7, 15h, 4, 0C4h, 87h, 43h, 97h, 0B8h, 60h, 0E3h, 45h, 0AAh, 8Fh, 13h, 0FC ; 00FA41F0: 005E127010
a:00FA4200 80 44 26 BC FF 75 9C 46 2A 27+db 33h, 0FEh, 9Ah, 5Ch, 86h, 6Dh, 16h, 2Fh, 0D4h, 0B2h, 48h, 82h, 5Fh, 68h, 29h, 3, 0C9h, 2, 80h, ; 00FA4200: 005E127010
a:00FA4210 4F C2 9D F6 01 0F 98 40 83 F3+db 1, 0Fh, 98h, 40h, 83h, 0F3h, 31h, 0BDh, 58h, 4Bh, 5, 0B6h, 0D6h, 8, 0C3h, 49h, 1Fh, 59h, 10h, ; 00FA4210: 005E127010
a:00FA41E0 ; int argc
a:00FA41E0 argc dd 0 ; DATA XREF: 005E27731o

```

去网上找了个 AES 的实现，然后换表尝试了一下，就得到了正确的flag

```
//unsigned char S[256] = {
//    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
//    0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
//    0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
//    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB, 0x27, 0xB2, 0x75,
//    0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84,
//    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
//    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50, 0x3C, 0x9F, 0xA8,
//    0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2,
//    0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
//    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE, 0x5E, 0x0B, 0xDB,
//    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79,
//    0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
//    0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B, 0xBD, 0x8B, 0x8A,
//    0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E,
//    0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
//    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
//};

unsigned char S[256] = {
    0x18, 0x79, 0x28, 0xF9, 0x55, 0x99, 0x71, 0xD5, 0x1B, 0xEC,
    0xBB, 0xB0, 0x95, 0x6F, 0x94, 0x70, 0xA3, 0x53, 0x63, 0xAD,
    0xF4, 0x7B, 0x37, 0x6E, 0x61, 0xDB, 0xB1, 0xD7, 0x3D, 0x02,
    0x04, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E, 0x2F, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E, 0x5F, 0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E, 0x6F, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E, 0x7F, 0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E, 0x8F, 0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E, 0x9F, 0xA0, 0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8, 0xA9, 0xAA, 0xAB, 0xAC, 0xAD, 0xAE, 0xAF, 0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE, 0xBF, 0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE, 0xCF, 0xD0, 0xD1, 0xD2, 0xD3, 0xD4, 0xD5, 0xD6, 0xD7, 0xD8, 0xD9, 0xDA, 0xDB, 0xDC, 0xDD, 0xDE, 0xDF, 0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE, 0xEF, 0xF0, 0xF1, 0xF2, 0xF3, 0xF4, 0xF5, 0xF6, 0xF7, 0xF8, 0xF9, 0xFA, 0xFB, 0xFC, 0xFD, 0xFE, 0xFF
};

//unsigned char inv_S[256] = {
//    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81, 0xF3, 0xD7, 0xFB,
//    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB,
//    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
//    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D, 0x8B, 0xD1, 0x25,
//    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92,
//    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
//    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8, 0xB3, 0x45, 0x06,
//    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B,
//    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
//    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C, 0x75, 0xDF, 0x6E,
//    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B,
//    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
//    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xEC, 0x5F,
//    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF,
//    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
//    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0C, 0x7D
//};

unsigned char inv_S[256] = { 0x29, 0xe0, 0xd1, 0xcf, 0xa6, 0xea, 0x40, 0xa4, 0xed, 0xbd, 0x5e, 0x65, 0xa0, 0x91,
```

flag为: flag{ETYt3eKw4nmMPfByjAQiqhCIDT}

Re

一开始就知道flag长度42

进入第一个函数可以知道flag开头为flag{, 末尾为}, 且中间几处有-

```
Str1 = 0;
v6 = 0;
v7 = 0;
Destination = 0;
v4 = 0;
v2 = 13;
strncpy(&Str1, Source, 5u);
strncpy(&Destination, Source + 41, 1u);
v7 = 0;
v4 = 0;
if ( strcmp(&Str1, "flag{") || (result = Destination, Destination != '}') )
    sub_401028();
while ( v2 <= 28 )
{
    if ( !sub_40104B(Source[v2]) )
        sub_401028();
    result = v2 + 5;
    v2 += 5;
}
return result;
}
```

中间有一处花指令混淆, 没有用处

```
1  int16 sub_4029D0()
2  {
3      return 15;
4  }
```

去-

```
int __cdecl sub_402C40(char *a1, char a2)
{
    int result; // eax
    int v3; // [esp+4Ch] [ebp-8h]
    int i; // [esp+50h] [ebp-4h]

    v3 = 0;
    for ( i = 0; ; ++i )
    {
        result = a1[i];
        if ( !a1[i] )
            break;
        if ( a1[i] != a2 )
            a1[v3++] = a1[i];
    }
    a1[v3] = 0;
    return result;
}
```

flag{后的几个字符，通过异或加密，很简单

```
v3[0] = 102;
v3[1] = 52;
v3[2] = 51;
v3[3] = 49;
v3[4] = 52;
v3[5] = 57;
v3[6] = 96;
v3[7] = 60;
v3[8] = 61;
v3[9] = 34;
v3[10] = 104;
v3[11] = 33;
v3[12] = 56;
for ( i = 0; i < 13; ++i )
{
    if ( i % 2 )
    {
        if ( (char)((2 * i) ^ *(_BYTE *)(i + a1)) != v3[i] )
            sub_401028();
    }
    else if ( (char)(i ^ *(_BYTE *)(i + a1)) != v3[i] )
    {
        sub_401028();
    }
    result = i + 1;
}
return result;
}
```

后面用md5加密，需要结合后面的 base64 比对，就可以暴力解出来

```
v5[0] = 63;
v5[1] = -70;
v5[2] = -60;
v5[3] = -111;
v5[4] = -60;
v5[5] = 116;
v5[6] = 2;
v5[7] = 38;
v5[8] = -20;
v5[9] = -110;
v5[10] = 56;
v5[11] = -62;
v5[12] = 11;
v5[13] = 109;
v5[14] = 39;
v5[15] = -45;
sub_40102D(v3);
v1 = strlen(Str);
sub_401041((int)v3, Str, v1);
result = sub_40105F(v3, v4);
for ( i = 0; i < 16; ++i )
{
    if ( v5[i] != v4[i] )
        sub_401028();
    result = i + 1;
}
return result;
}

- - - - -

strcpy(Str1, "ZjQ3ODEzYzI2NTk0YzA=");
Str[14] = 0;
sub_401055(Str, (int)Str2);
result = strcmp(Str1, Str2);
if ( result )
    sub_401028();
return result;
}
```

最后因为要除0触发异常才有正确提示，所以是'0'（导致a1的值为0）

```
1 int __cdecl sub_402890(int a1)
2 {
3     int v2[22]; // [esp+50h] [ebp-84h] BYREF
4     char v3[16]; // [esp+A8h] [ebp-2Ch] BYREF
5     int v4; // [esp+B8h] [ebp-1Ch]
6     CPPEH_RECORD ms_exc; // [esp+BCh] [ebp-18h]
7
8     ms_exc.registration.TryLevel = 0;
9     v4 = 0;
10    sub_40102D(v2);
11    sub_401041((int)v2, "flag", 4);
12    sub_40105F(v2, v3);
13    return 1 / a1;
14 }
```

所以最后flag为：

flag{f61703f2-50b7-4f47-813c-26594c0e5810}

Crypto

ezRsa

由题目描述知道，Flag为c模p的平方根，据此求出两个，其中一个即为flag

```
from sympy.ntheory import legendre_symbol, sqrt_mod

# Given values
p =
13107939563507459774616204141253747489232063336204173944123263284507604328885680072
478669016969428366667381358004059204207134817952620014738665450753147857
c =
41248207997371072363088370085243973551077869504147699961813243335569501542069800594
06402767327725312238673053581148641438494212320157665395208337575556385

# Finding square roots modulo p
sqrt_c_mod_p = sqrt_mod(c, p, all_roots=True)
```