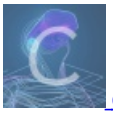
re学习笔记（75）BUUCTF - re - [ACTF新生赛2020]Splendid_MineCraft

原创

Forgo7ten 于 2021-06-06 12:09:17 发布 152 收藏

分类专栏： ctf小白成长ing # reverse 文章标签： python smc Reverse CTF

本文链接：https://blog.csdn.net/Palmer9/article/details/117622987



ctf小白成长ing 同时被 2 个专栏收录

112 篇文章 6 订阅

订阅专栏

reverse

113 篇文章 6 订阅

订阅专栏

[ACTF新生赛2020]Splendid_MineCraft

main函数

strtok是用分隔符划分字符串，根据+5 +9差4个，再加上+9强转为WORD，得到每组字符是4+2=6个

flag格式为 `ACTF{xxxxxx_xxxxxx_xxxxxx}`

```
18
19  sub_971020("%s\n", (char)aWelcomeToActfS);
20  sub_971050("%s", (char)Str1);
21  if ( &Str1[strlen(Str1) + 1] - &Str1[1] == 26 && !strncmp(Str1, "ACTF{", 5u) && Str1[25] == '}' )
22  {
23    Str1[25] = 0;
24    v3 = strtok(Str1, "_");
25    v15 = *(_DWORD *)(v3 + 5);
26    v16 = *(_WORD *)(v3 + 9);
27    v17 = *(_DWORD *)(v3 + 5);
28    v18 = *(_WORD *)(v3 + 9);
29    v4 = strtok(0, "_");
30    v11 = *(_DWORD *)v4;
31    v12 = *((_WORD *)v4 + 2);
32    v8 = strtok(0, "_");
33    *(_DWORD *)v13 = *(_DWORD *)v8;
34    v14 = *((_WORD *)v8 + 2);
35    dword_973354 = (int)sub_9751D8;
36    if ( ((int (__cdecl *)(int *))sub_9751D8)(&v15) )
37    {
38      v9 = BYTE2(v17) ^ HIBYTE(v18) ^ v17 ^ HIBYTE(v17) ^ BYTE1(v17) ^ v18;
39      for ( i = 256; i < 496; ++i )
40        byte_975018[i] ^= v9;
41      __asm { jmp     eax }
42    }
43  }
44  sub_971020("Wrong\n", v6);
45  return 0;
```


36行跳转到数据段

先是对数据段进行异或0x72h解密

```
.data:009751D8 call    $+5
.data:009751DD pop     esi
.data:009751DE push    edi
.data:009751DF xor     edi, edi
.data:009751E1
.data:009751E1 loc_9751E1:                             ; CODE XREF: sub_9751D8+1D↓j
.data:009751E1 cmp     edi, 337
.data:009751E7 jg      short loc_9751FB
.data:009751E9 mov     bl, [esi+edi+1Fh]
.data:009751ED xor     bl, 72h
.data:009751F0 mov     [esi+edi+1Fh],
.data:009751F4 inc     edi
.data:009751F5 jmp     short loc_9751E
.data:009751F7 ; ---------------------
.data:009751F7 dec     eax
.data:009751F8 db      65h
.data:009751F8 jns     short near ptr
.data:009751FB
.data:009751FB loc_9751FB:
.data:009751FB pop     edi
```

```
[esi+edi+1Fh]=[sub_9751FC]

; =============== S U B R O U T I N E =======================

; Attributes: bp-based frame

; BOOL __cdecl sub_9751FC(char *a1)
sub_9751FC      proc far
                daa
                stc
                sahf
```


解密后，F5得到

Debug View | IDA View-EIP | Pseudocode-C | Pseudocode-B

```
1  void __cdecl sub_9751FC(char *a1)
2  {
3    char v1[20]; // [esp+0h] [ebp-2Ch] BYREF
4    char v2[8]; // [esp+14h] [ebp-18h] BYREF
5    int v3; // [esp+20h] [ebp-Ch]
6    int v4; // [esp+24h] [ebp-8h]
7    int i; // [esp+28h] [ebp-4h]
8
9    v4 = 0;
10   qmemcpy(v2, "3@1b;b", 6);
11   strcpy(v1, "Welcome ");
12   v3 = 8;
13   for ( i = 0; i < 6; ++i )
14   {
15     v1[i + 12] = (v1[i + 1] ^ v2[i]) + 35;
16     if ( v1[i + 12] == a1[i] )
17       ++v4;
18   }
19 }
```


两个数异或+35与第一段输入进行比较

第二段也是一个smc自解密
第二段主要代码。

```
.data:00975158 loc_975158:                             ; CODE XREF: .data:00975196↓j
.data:00975158 cmp     edi, 6
.data:0097515B jge     short near ptr unk_975198
.data:0097515D xor     ecx, ecx
.data:0097515F mov     cl, [esi+edi]
.data:00975162 and     cl, 0FFh
.data:00975165 sub     eax, 100h
.data:0097516A xor     ebx, ebx
.data:0097516C mov     bl, cl
.data:0097516E mov     ecx, edi
.data:00975170 add     ecx, 83h
.data:00975176 xor     ebx, ecx
.data:00975178 mov     bl, [eax+ebx]
.data:0097517B jmp     short loc_975185
.data:0097517B ; ---------------------------------------------------------------------------
.data:0097517D db    0
.data:0097517E db 30h, 4, 4, 3, 30h, 43h
.data:00975184 db  90h
.data:00975185 ; ---------------------------------------------------------------------------
.data:00975185
.data:00975185 loc_975185:                             ; CODE XREF: .data:0097517B↑j
.data:00975185 mov     cl, [eax+edi+166h]
.data:0097518C cmp     bl, cl
.data:0097518E jnz     short loc_9751A4
.data:00975190 inc     edi
.data:00975191 add     eax, 100h
.data:00975196 jmp     short loc_975158
.data:00975196 ; ---------------------------------------------------------------------------
.data:00975198 unk_975198 db  5Bh ; [                  ; CODE XREF: .data:0097515B↑j
.data:00975199 pop     ecx
```


ebx为用户输入与(0x83+i)异或得到的下标
根据eax进行索引查表。得到的bl与第二段输入cl进行比较

```
.text:009712C9 loc_9712C9:                             ; CODE XREF: _main+238↑j
.text:009712C9 push    6                               ; MaxCount
.text:009712CB push    offset a5mcsm                   ; "5mcsM<"
.text:009712D0 lea     ecx, [ebp+var_1C]
.text:009712D3 push    ecx                             ; Str1
.text:009712D4 call    ds:strncmp
.text:009712DA add     esp, 0Ch
.text:009712DD test    eax, eax
.text:009712DF jz      short loc_9712F0
```

第三段直接是一个字符串比较

python脚本

```
s1 = "3@1b;b"
s2 = "elcome "
flag1 = ""
for i in range(6):
    flag1+=chr((ord(s1[i])^ord(s2[i]))+35)
print(flag1)
table = [0xF6, 0xA3, 0x5B, 0x9D, 0xE0, 0x95, 0x98, 0x68, 0x8C, 0x65, 0xBB, 0x76, 0x89, 0xD4, 0x09, 0xFD, 0xF3, 0
x5C, 0x3C, 0x4C, 0x36, 0x8E, 0x4D, 0xC4, 0x80, 0x44, 0xD6, 0xA9, 0x01, 0x32, 0x77, 0x29, 0x90, 0xBC, 0xC0, 0xA8,
 0xD8, 0xF9, 0xE1, 0x1D, 0xE4, 0x67, 0x7D, 0x2A, 0x2C, 0x59, 0x9E, 0x3D, 0x7A, 0x34, 0x11, 0x43, 0x74, 0xD1, 0x6
2, 0x60, 0x02, 0x4B, 0xAE, 0x99, 0x57, 0xC6, 0x73, 0xB0, 0x33, 0x18, 0x2B, 0xFE, 0xB9, 0x85, 0xB6, 0xD9, 0xDE, 0
x7B, 0xCF, 0x4F, 0xB3, 0xD5, 0x08, 0x7C, 0x0A, 0x71, 0x12, 0x06, 0x37, 0xFF, 0x7F, 0xB7, 0x46, 0x42, 0x25, 0xC9,
 0xD0, 0x50, 0x52, 0xCE, 0xBD, 0x6C, 0xE5, 0x6F, 0xA5, 0x15, 0xED, 0x64, 0xF0, 0x23, 0x35, 0xE7, 0x0C, 0x61, 0xA
4, 0xD7, 0x51, 0x75, 0x9A, 0xF2, 0x1E, 0xEB, 0x58, 0xF1, 0x94, 0xC3, 0x2F, 0x56, 0xF7, 0xE6, 0x86, 0x47, 0xFB, 0
x83, 0x5E, 0xCC, 0x21, 0x4A, 0x24, 0x07, 0x1C, 0x8A, 0x5A, 0x17, 0x1B, 0xDA, 0xEC, 0x38, 0x0E, 0x7E, 0xB4, 0x48,
 0x88, 0xF4, 0xB8, 0x27, 0x91, 0x00, 0x13, 0x97, 0xBE, 0x53, 0xC2, 0xE8, 0xEA, 0x1A, 0xE9, 0x2D, 0x14, 0x0B, 0xB
F, 0xB5, 0x40, 0x79, 0xD2, 0x3E, 0x19, 0x5D, 0xF8, 0x69, 0x39, 0x5F, 0xDB, 0xFA, 0xB2, 0x8B, 0x6E, 0xA2, 0xDF, 0
x16, 0xE2, 0x63, 0xB1, 0x20, 0xCB, 0xBA, 0xEE, 0x8D, 0xAA, 0xC8, 0xC7, 0xC5, 0x05, 0x66, 0x6D, 0x3A, 0x45, 0x72,
 0x0D, 0xCA, 0x84, 0x4E, 0xF5, 0x31, 0x6B, 0x92, 0xDC, 0xDD, 0x9C, 0x3F, 0x55, 0x96, 0xA1, 0x9F, 0xCD, 0x9B, 0xE
3, 0xA0, 0xA7, 0xFC, 0xC1, 0x78, 0x10, 0x2E, 0x82, 0x8F, 0x30, 0x54, 0x04, 0xAC, 0x41, 0x93, 0xD3, 0x3B, 0xEF, 0
x03, 0x81, 0x70, 0xA6, 0x1F, 0x22, 0x26, 0x28, 0x6A, 0xAB, 0x87, 0xAD, 0x49, 0x0F, 0xAF]
data=[0x30, 0x04, 0x04, 0x03, 0x30, 0x63]
flag2 = ""
for i in range(6):
    flag2+=chr(table.index(data[i])^(0x83+i))
print(flag2)
flag3="5mcsM<"

print("ACTF{%s_%s_%s}"%(flag1,flag2,flag3))
```

得到flag为`ACTF{yOu0y*_knowo3_5mcsM<}`