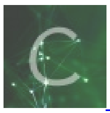
2021绿城杯 RE WP

原创

努力学习的大康 于 2021-09-29 22:31:18 发布 2856 收藏 1
分类专栏： 逆向分析 CTF 文章标签： 安全


逆向分析 同时被 2 个专栏收录
22 篇文章 4 订阅
订阅专栏

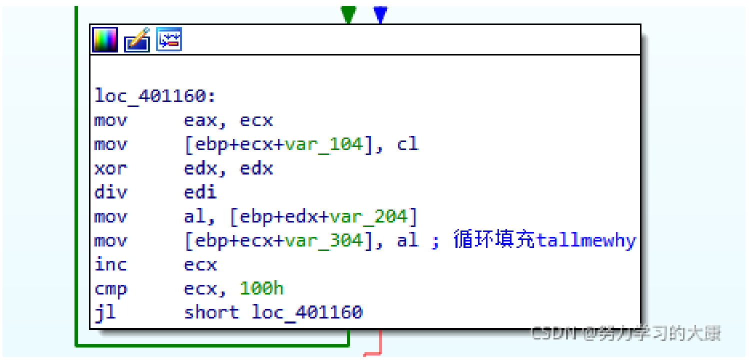
CTF
23 篇文章 2 订阅
订阅专栏

逆向（100分题）

变异RC4，关键点是密钥盒S的生成过程被魔改了。RC4原理参考之前的博客

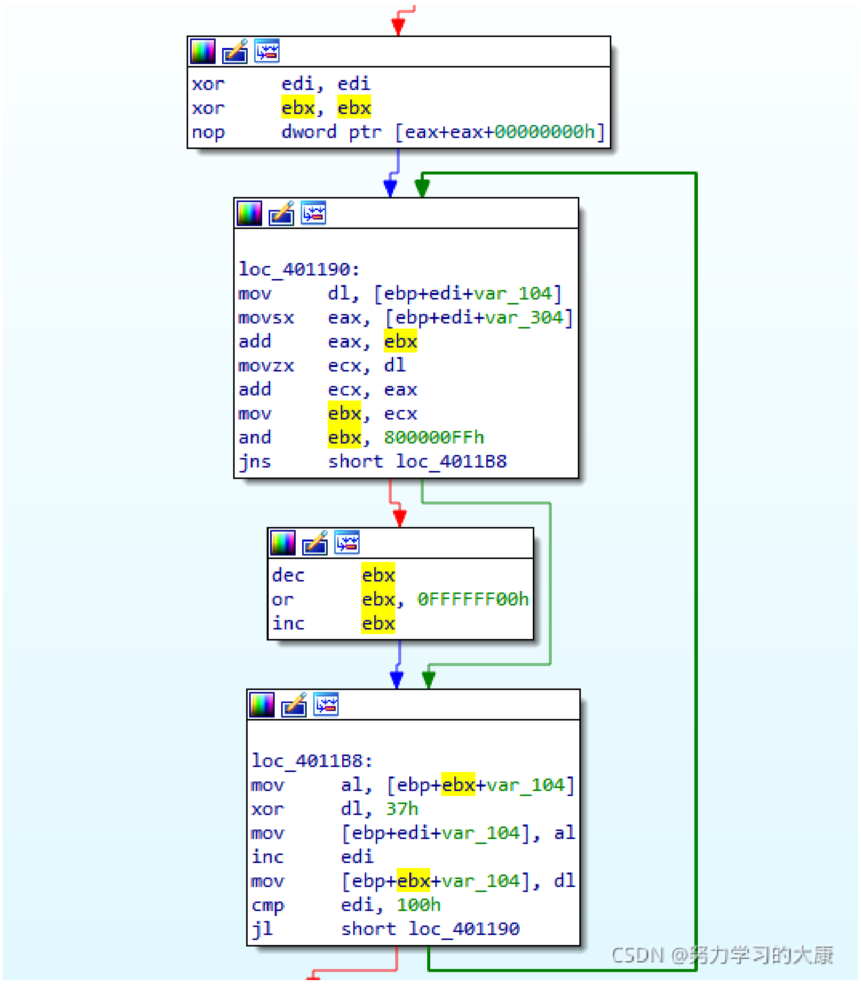
基本流程

1.初始循环填充key，key=“tallmewhy”



```
loc_401160:
mov     eax, ecx
mov     [ebp+ecx+var_104], cl
xor     edx, edx
div     edi
mov     al, [ebp+edx+var_204]
mov     [ebp+ecx+var_304], al ; 循环填充tallmewhy
inc     ecx
cmp     ecx, 100h
jl      short loc_401160
```

2.计算密钥盒S，但是不是标准的RC4算法。可以通过dump的方法将密钥和提取



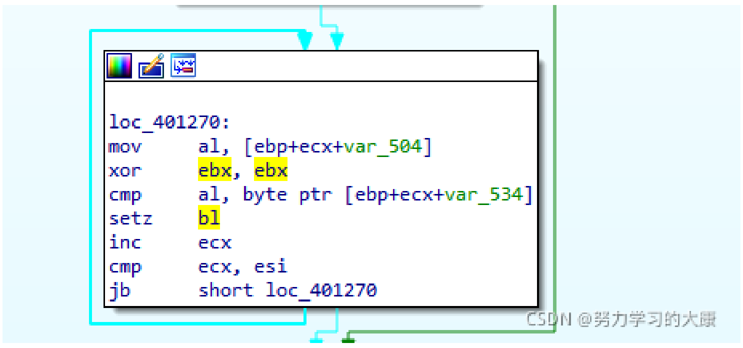
```
xor     edi, edi
xor     ebx, ebx
nop     dword ptr [eax+eax+00000000h]
```

```
loc_401190:
mov     dl, [ebp+edi+var_104]
movsx   eax, [ebp+edi+var_304]
add     eax, ebx
movzx   ecx, dl
add     ecx, eax
mov     ebx, ecx
and     ebx, 800000FFh
jns     short loc_4011B8
```

```
dec     ebx
or      ebx, 0FFFFFF00h
inc     ebx
```

```
loc_4011B8:
mov     al, [ebp+ebx+var_104]
xor     dl, 37h
mov     [ebp+edi+var_104], al
inc     edi
mov     [ebp+ebx+var_104], dl
cmp     edi, 100h
jl      short loc_401190
```

3.ebp+ecx+var_534中是密文



```
loc_401270:
mov     al, [ebp+ecx+var_504]
xor     ebx, ebx
cmp     al, byte ptr [ebp+ecx+var_534]
setz    bl
inc     ecx
cmp     ecx, esi
jb      short loc_401270
```

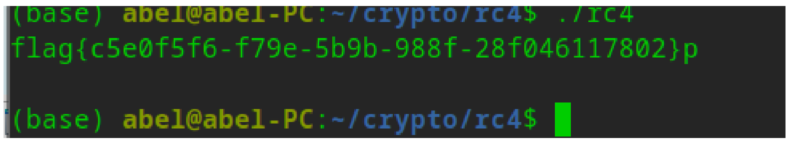
由于不想看魔改的S盒生成算法，所以在S盒生成完成的地方下断点，dump S盒填充就可以了。

exp

```c
#include<stdio.h>
#include<string.h>

struct rc4_state
{
    int x, y, m[256];
}rc4_state;

void rc4_setup( struct rc4_state *s, unsigned char *key,  int length );
void rc4_crypt( struct rc4_state *s, unsigned char *data, int length );

void rc4_setup( struct rc4_state *s, unsigned char *key,  int length )
{
    int i, j, k, *m, a;

    s->x = 0;
    s->y = 0;
    m = s->m;

    for( i = 0; i < 256; i++ )
    {
        m[i] = i;
    }

    j = k = 0;

    for( i = 0; i < 256; i++ )
    {
        a = m[i];
        j = (unsigned char) ( j + a + key[k] );
        m[i] = m[j]; m[j] = a;
        if( ++k >= length ) k = 0;
    }
}

void rc4_crypt( struct rc4_state *s, unsigned char *data, int length )
{
    int i, x, y, *m, a, b;

    x = s->x;
    y = s->y;
    m = s->m;

    for( i = 0; i < length; i++ )
    {
        x = (unsigned char) ( x + 1 );
        a = m[x];
        y = (unsigned char) ( y + a );
        m[x] = b = m[y];
        m[y] = a;
        data[i] ^= m[(unsigned char) ( a + b )];
    }

    s->x = x;
    s->y = y;
}
```

```
int main()
{
    struct rc4_state rc4_ctx;
    char* key = "tallmewhy";
    unsigned char content[256] = "aaaabbbbccccdddd";
    unsigned char encrpyt[256] = {
        0xF5,0x8C,0x8D,0xE4,0x9F,0xA5,0x28,0x65,0x30,0xF4,0xEB,0xD3,0x24,0xA9,0x91,0x1A,
        0x6F,0xD4,0x6A,0xD7,0x0B,0x8D,0xE8,0xB8,0x83,0x4A,0x5A,0x6E,0xBE,0xCB,0xF4,0x4B,
        0x99,0xD6,0xE6,0x54,0x7A,0x4F,0x50,0x14,0xE5,0xEC,0x76,0x00,0x20,0xC9,0xD1,0x00
    };
    unsigned char mtrix[256] = {
        0x74,0x1B,0xD8,0xAC,0x9E,0xB5,0x0B,0x7A,0xFB,0x10,0x8A,0xAB,0x3A,0x72,0x15,0x19
        ,0x5B,0x18,0x00,0x67,0xE7,0xAA,0x75,0x24,0xB1,0xF4,0xE3,0x89,0x49,0x9F,0x84,0xB5
        ,0x10,0xD2,0x5C,0x67,0x2A,0x6D,0xCA,0x4A,0x52,0x89,0x4A,0x6A,0x46,0xC5,0x76,0x6D
        ,0x8A,0xCE,0xE2,0x8C,0x60,0x36,0x06,0xF6,0x0B,0x61,0x39,0xCF,0x62,0xDD,0x7A,0x47
        ,0xA8,0x95,0x43,0x12,0x41,0xF3,0xFE,0x7F,0x56,0x20,0x0E,0xD3,0x04,0x94,0xB3,0xBA
        ,0xA0,0xA1,0x3F,0xFA,0x30,0xBC,0xF7,0x53,0xF1,0xC5,0x42,0xF5,0x62,0xA1,0x2F,0x64
        ,0x91,0xAF,0x44,0x92,0xD8,0x03,0xFA,0x1E,0xA0,0xC7,0xC0,0x71,0x85,0xD0,0xAA,0x6F
        ,0x05,0x02,0x17,0x47,0xF3,0x30,0x32,0x6B,0xB8,0x04,0x5E,0x83,0xA7,0xA6,0xEB,0x0F
        ,0xF0,0x2B,0x8E,0xC8,0x08,0x06,0x8B,0xB1,0xF5,0xEA,0xB6,0x2C,0x16,0x38,0x8F,0x99
        ,0x95,0x4B,0xF4,0x25,0x94,0x68,0x5B,0x35,0xB4,0x58,0xF9,0x79,0x59,0xCB,0x00,0x0A
        ,0x7E,0x87,0xDF,0xEE,0x93,0xAB,0xC8,0xD4,0xA4,0x24,0xBA,0x98,0x44,0x2E,0x69,0x03
        ,0x5D,0x77,0xFE,0xD1,0xD7,0xCA,0xEC,0x3E,0xF6,0xAD,0xDB,0x0C,0xD9,0x2D,0x36,0x45
        ,0xE0,0x23,0xC6,0x77,0x92,0x29,0xE2,0x26,0x09,0x9B,0xED,0x63,0x57,0x78,0xFC,0x79
        ,0x54,0x3B,0x1A,0x7B,0x2B,0x22,0x80,0x3A,0xE4,0xCC,0x7B,0xBF,0xDE,0x7F,0x68,0x51
        ,0xAD,0x3D,0x65,0xDA,0x6E,0x1D,0xBF,0xD6,0xF8,0xDE,0x99,0xDC,0xB0,0x78,0xB9,0x85
        ,0x26,0x1F,0x23,0xE8,0x16,0x28,0xA7,0x66,0xBB,0x4B,0xE1,0xB6,0xE5,0xB7,0xA9,0xD1
    };
    memset(&rc4_ctx,0,sizeof(rc4_state));
    rc4_setup(&rc4_ctx,key,strlen(key));
    for(int i=0;i<256;i++)//用dump的密钥盒S修复
    {
        rc4_ctx.m[i]=mtrix[i];
    }
    rc4_crypt(&rc4_ctx,encrpyt,strlen(encrpyt));
    printf("%s\n");
    printf("\n");
}
```

得到最后的flag：flag{c5e0f5f6-f79e-5b9b-988f-28f046117802}



```
(base) abel@abel-PC:~/crypto/rc4$ ./rc4
flag{c5e0f5f6-f79e-5b9b-988f-28f046117802}p

(base) abel@abel-PC:~/crypto/rc4$
```

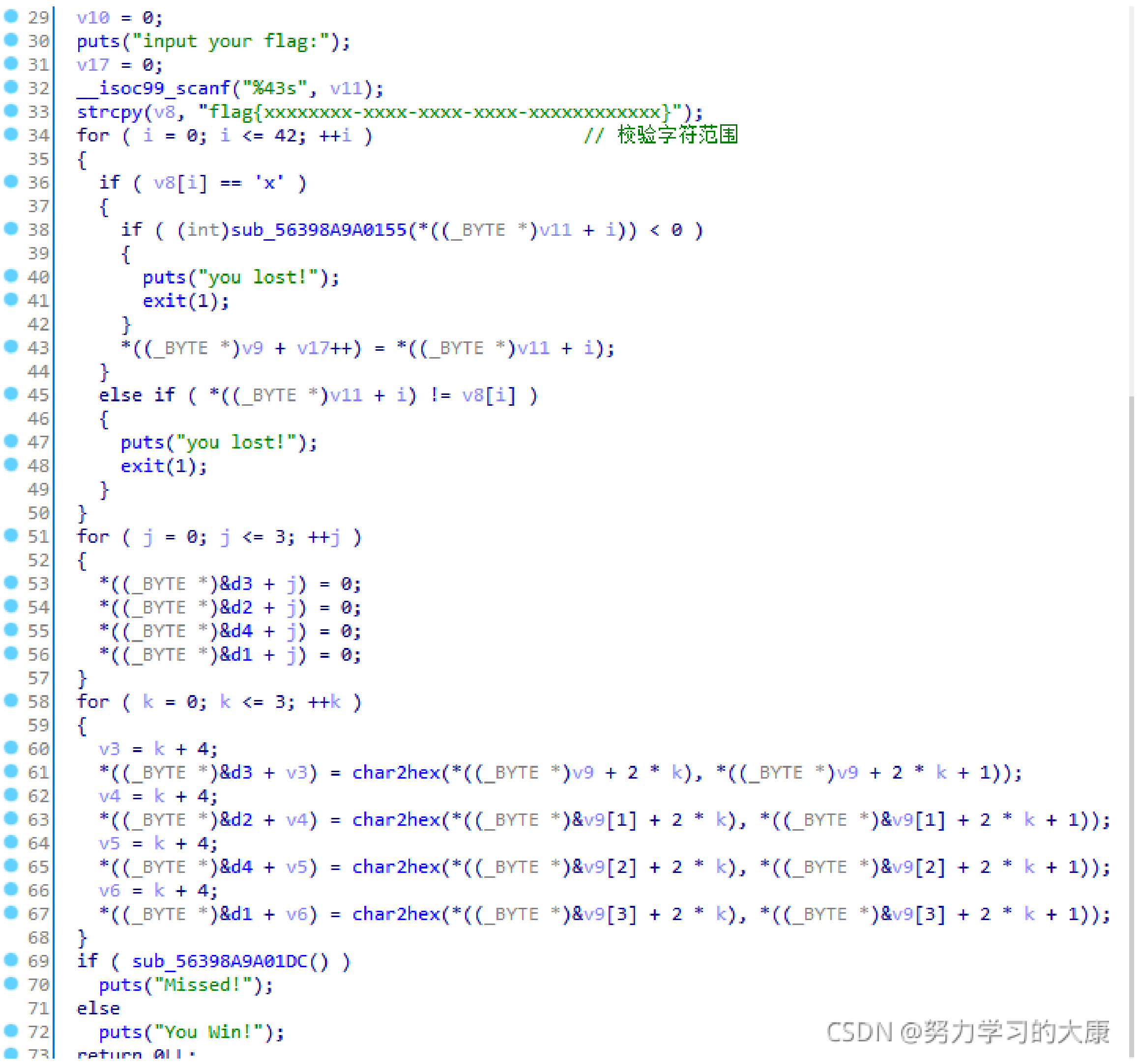
投石机

思路是通过爆破得到4个int32的值。一开始一直不对的原因是大端小端填充的问题。

基本流程

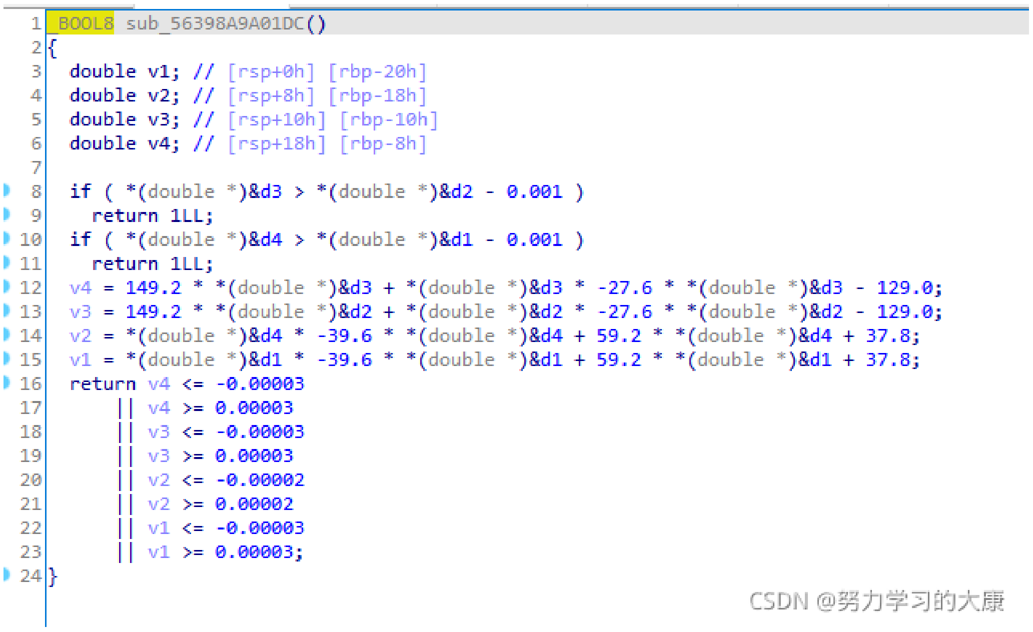
flag的格式为flag{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}

之后将char2hex进行转换



```
v10 = 0;
puts("input your flag:");
v17 = 0;
__isoc99_scanf("%43s", v11);
strcpy(v8, "flag{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}");
for ( i = 0; i <= 42; ++i )                        // 校验字符范围
{
  if ( v8[i] == 'x' )
  {
    if ( (int)sub_56398A9A0155(*((_BYTE *)v11 + i)) < 0 )
    {
      puts("you lost!");
      exit(1);
    }
    *((_BYTE *)v9 + v17++) = *((_BYTE *)v11 + i);
  }
  else if ( *((_BYTE *)v11 + i) != v8[i] )
  {
    puts("you lost!");
    exit(1);
  }
}
for ( j = 0; j <= 3; ++j )
{
  *((_BYTE *)&d3 + j) = 0;
  *((_BYTE *)&d2 + j) = 0;
  *((_BYTE *)&d4 + j) = 0;
  *((_BYTE *)&d1 + j) = 0;
}
for ( k = 0; k <= 3; ++k )
{
  v3 = k + 4;
  *((_BYTE *)&d3 + v3) = char2hex(*((_BYTE *)v9 + 2 * k), *((_BYTE *)v9 + 2 * k + 1));
  v4 = k + 4;
  *((_BYTE *)&d2 + v4) = char2hex(*((_BYTE *)&v9[1] + 2 * k), *((_BYTE *)&v9[1] + 2 * k + 1));
  v5 = k + 4;
  *((_BYTE *)&d4 + v5) = char2hex(*((_BYTE *)&v9[2] + 2 * k), *((_BYTE *)&v9[2] + 2 * k + 1));
  v6 = k + 4;
  *((_BYTE *)&d1 + v6) = char2hex(*((_BYTE *)&v9[3] + 2 * k), *((_BYTE *)&v9[3] + 2 * k + 1));
}
if ( sub_56398A9A01DC() )
  puts("Missed!");
else
  puts("You Win!");
return 0LL;
```

给了几个double的限制条件，爆破一下就好了



```
_BOOL8 sub_56398A9A01DC()
{
  double v1; // [rsp+0h] [rbp-20h]
  double v2; // [rsp+8h] [rbp-18h]
  double v3; // [rsp+10h] [rbp-10h]
  double v4; // [rsp+18h] [rbp-8h]

  if ( *(double *)&d3 > *(double *)&d2 - 0.001 )
    return 1LL;
  if ( *(double *)&d4 > *(double *)&d1 - 0.001 )
    return 1LL;
  v4 = 149.2 * *(double *)&d3 + *(double *)&d3 * -27.6 * *(double *)&d3 - 129.0;
  v3 = 149.2 * *(double *)&d2 + *(double *)&d2 * -27.6 * *(double *)&d2 - 129.0;
  v2 = *(double *)&d4 * -39.6 * *(double *)&d4 + 59.2 * *(double *)&d4 + 37.8;
  v1 = *(double *)&d1 * -39.6 * *(double *)&d1 + 59.2 * *(double *)&d1 + 37.8;
  return v4 <= -0.00003
      || v4 >= 0.00003
      || v3 <= -0.00003
      || v3 >= 0.00003
      || v2 <= -0.00002
      || v2 >= 0.00002
      || v1 <= -0.00003
      || v1 >= 0.00003;
}
```

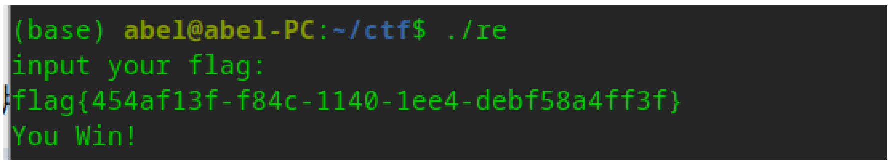
exp

```c
#include <stdio.h>

int main()
{

    for(unsigned long long int i=0;i<0xffffffff;i++)
    {
        unsigned long long int tmp = i<<32;
        double tmp2 = *(double*)(&tmp);
        double v4 = 149.2 * tmp2 + tmp2 * (-27.6) * tmp2 - 129.0;
        double v2 = tmp2 * (-39.6) * tmp2 + (59.2) * tmp2 + (37.8);
        double v1 = tmp2 * (-39.6) * tmp2 + (59.2) * tmp2 + (37.8);
        if( v4>-0.00003 && v4<0.00003)
        {
            printf("v4:%p\n",i);
        }
        if(v2>-0.00002 && v2<0.00002)
        {
            printf("v2:%p\n",i);
        }
        if(v1>-0.00003 && v1<0.00003)
        {
            printf("v1:%p\n",i);
        }
    }
    return 0;
}
```

output

```
v4:0x3ff14a45
v1:0x3fffa458
v4:0x40114cf8
v1:0xbfdee41d
v2:0xbfdee41e
v1:0xbfdee41e
v1:0xbfdee41f
```

遍历一下或者根据前面的限制条件得到最后的flag
flag{454af13f-f84c-1140-1ee4-debf58a4ff3f}



```
(base) abel@abel-PC:~/ctf$ ./re
input your flag:
flag{454af13f-f84c-1140-1ee4-debf58a4ff3f}
You Win!
```

Vxworks

没做出来，球球师傅教教我