buu逆向刷题（六）

原创

北风~ 于 2020-12-31 17:44:02 发布 6430 收藏

分类专栏： CTF 逆向与保护 文章标签： CTF

本文链接：https://blog.csdn.net/weixin_45055269/article/details/109038886

CTF 同时被 2 个专栏收录

31 篇文章 3 订阅

订阅专栏

逆向与保护

65 篇文章 4 订阅

订阅专栏

1.[GWCTF 2019]re3

又快又准的判断算法。

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

```
1 void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
2 {
3   signed int i; // [rsp+8h] [rbp-48h]
4   char s; // [rsp+20h] [rbp-30h]
5   unsigned __int64 v5; // [rsp+48h] [rbp-8h]
6
7   v5 = __readfsqword(0x28u);
8   __isoc99_scanf("%39s", &s, a3);
9   if ( (unsigned int)strlen(&s) != 32 )
10  {
11    puts("Wrong!");
12    exit(0);
13  }
14  mprotect(&dword_400000, 0xF000uLL, 7);
15  for ( i = 0; i <= 223; ++i )
16    *((_BYTE *)sub_402219 + i) ^= 0x99u;
17  sub_40207B(&unk_603170, 61440LL);
18  sub_402219(&s);
19 }
```


输入判断长度32位，mprotect将400000位置的代码修改为可读可改可执行代码，402219的代码异或0x99自修改（SMC自修改）。修改后的代码形成函数sub_402219，参数为输入。动调也可，这里写静态idc。

```
auto addr = 0x402219;
    auto i;
    for(i = 0; i <= 223; ++i){
        PatchByte(addr+i,Byte(addr+i)^0x99);
    }
```

402219处的代码被修改，按C数据转代码。最上面有个push ebp，按D转数据后，再按C转代码，这样就是一个push ebp也到了下面，选中红色区域按P创建函数。

sub_40207B加密unk_603170，sub_401CF9标准的MD5加密（通过四个数组识别）

IDA… | Pseu… | Pseu… | Findcrypt… | Pseu…

```
1 unsigned __int64 __fastcall sub_40207B(__int64 a1)
2 {
3   char v2; // [rsp+10h] [rbp-50h]
4   __int64 v3; // [rsp+20h] [rbp-40h]
5   __int64 v4; // [rsp+30h] [rbp-30h]
6   __int64 v5; // [rsp+40h] [rbp-20h]
7   unsigned __int64 v6; // [rsp+58h] [rbp-8h]
8
9   v6 = __readfsqword(0x28u);
10  sub_401CF9(&BASE64_table_603120, 64LL, &v2);
11  sub_401CF9(&CRC32_table_603100, 20LL, &v3);
12  sub_401CF9(&Prime_Constants_char_6030C0, 53LL, &v4);
13  sub_401CF9(&MD5_Constants_4025C0, 256LL, &v5);
14  sub_401CF9(&v2, 64LL, a1);
15  return __readfsqword(0x28u) ^ v6;
16 }
```


md5加密的输入512bit分组，正好base64的码表是64位字符（512bit），最终生成32位（128bit）的密文，上述函数可以看到base64的码表经过两次md5加密，得到a1，这里没有输入，所以动调可出两次md5的结果。

```
CB8D493521B47A4CC1AE7E62229266
```

sub_402219，经过分析可知AES加密，由外向里看，没有传入iv，而且传入的输入并没有异或操作，所以这是一个ECB模式的AES加密。

```
__int64 __fastcall sub_402219(__int64 a1)
{
  unsigned int v2; // [rsp+18h] [rbp-D8h]
  signed int i; // [rsp+1Ch] [rbp-D4h]
  char v4; // [rsp+20h] [rbp-D0h]
  unsigned __int64 v5; // [rsp+E8h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  sub_400A71((__int64)&v4, (__int64)&unk_603170);// 轮秘钥扩展
  sub_40196E((__int64)&v4, a1);                 // AES加密
  sub_40196E((__int64)&v4, a1 + 16);            // AES加密
  v2 = 1;
  for ( i = 0; i <= 31; ++i )
  {
    if ( *(_BYTE *)(i + a1) != byte_6030A0[i] ) // 与标准数组对比
      v2 = 0;
  }
  return v2;
}
```



```
__int64 __fastcall sub_401828(__int64 a1, __int64 a2)
{
  unsigned __int8 i; // [rsp+1Fh] [rbp-1h]

  sub_400B0A(0LL, a1, a2);
  for ( i = 1; i <= 9u; ++i )                   // 9次组合操作
  {
    sub_400BAC(a1);                             // S盒替换
    sub_400C1F(a1);                             // 行移位
    sub_400D27(a1);                             // 列混合
    sub_400B0A(i, a1, a2);                      // 轮秘钥加
  }
  sub_400BAC(a1);                               // 第十次组合操作（除列混合）
  sub_400C1F(a1);
  return sub_400B0A(10LL, a1, a2);
```


明文比较的结果

```
bc0aadc0147c5ecce0b140bc9c51d52b46b2b9434de5324bad7fb4b39cdb4b5b
```

写出解密脚本

```
from Crypto.Cipher import AES
from Crypto.Util.number import *
key = long_to_bytes(0xcb8d493521b47a4cc1ae7e62229266ce)
mi = long_to_bytes(0xbc0aadc0147c5ecce0b140bc9c51d52b46b2b9434de5324bad7fb4b39cdb4b5b)
lun = AES.new(key, mode=AES.MODE_ECB)
flag = lun.decrypt(mi)
print(flag)
```

2.[GUET-CTF2019]number_game

iDA打开，主体函数逻辑清楚。

```
  char v9; // [rsp+2Ah] [rbp-16h]
  unsigned __int64 v10; // [rsp+38h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  v5 = 0LL;
  v6 = 0;
  v7 = 0LL;
  v8 = 0;
  v9 = 0;
  __isoc99_scanf("%s", &v5, a3);
  if ( (unsigned int)sub_4006D6((const char *)&v5) )
  {
    v3 = sub_400758((__int64)&v5, 0, 10);        // 字符变换 v5->v3
    sub_400807((__int64)v3, (__int64)&v7);       // 字符变换 v3->v7
    v9 = 0;
    sub_400881((char *)&v7);                     // 向数独中填数据
    if ( (unsigned int)sub_400917() )            // 数独
    {
      puts("TQL!");
      printf("flag{", &v7);
      printf("%s", &v5);
      puts("}");
    }
    else
    {
      puts("your are cxk!!");
    }
  }
  return __readfsqword(0x28u) ^ v10;
}
00000A84 main:23 (400A84)
```


sub_400881和sub_400917函数利用数独给出最后的比饺字符。

```
__int64 __fastcall sub_400881(char *a1)
{
  __int64 result; // rax

  byte_601062 = *a1;                             // 0
  byte_601067 = a1[1];                           // 4
  byte_601069 = a1[2];                           // 2
  byte_60106B = a1[3];                           // 1
  byte_60106E = a1[4];                           // 4
  byte_60106F = a1[5];                           // 2
  byte_601071 = a1[6];                           // 1
  byte_601072 = a1[7];                           // 4
  byte_601076 = a1[8];                           // 3
  result = (unsigned __int8)a1[9];
  byte_601077 = a1[9];                           // 0
  return result;
}
```



```
__int64 sub_400917()
{
  unsigned int v1; // [rsp+0h] [rbp-10h]
  signed int i; // [rsp+4h] [rbp-Ch]
  signed int j; // [rsp+8h] [rbp-8h]
  int k; // [rsp+Ch] [rbp-4h]
```

```
  int k; // [rsp+Ch] [rbp-4h]

  v1 = 1;
  for ( i = 0; i <= 4; ++i )                    // 若按照先行后列
  {
    for ( j = 0; j <= 4; ++j )
    {
      for ( k = j + 1; k <= 4; ++k )
      {
        if ( *((_BYTE *)&unk_601060 + 5 * i + j) == *((_BYTE *)&unk_601060 + 5 * i + k) )// 行
          v1 = 0;
        if ( *((_BYTE *)&unk_601060 + 5 * j + i) == *((_BYTE *)&unk_601060 + 5 * k + i) )// 列
          v1 = 0;
      }
    }
  }
  return v1;
}
```


数独比较简单结果是

```
0421421430
```

接着向上看，两个函数通过递归操作进行数组位置的替换，可以输入0123456789，10个数字作位置，动调看最后各数字的位置，来得知数组变换的结果。sub_4006D6限制输入长度为10，输入字符只能为‘0’ ‘1’ ‘2’ ‘3’ ‘4‘，考虑这个限制，所以输入0123456789需要改下跳转。

输入0123456789动调改跳转，可得v7结果

```
[stack]:00007FFCEAB39D5E db    0
[stack]:00007FFCEAB39D5F db    0
[stack]:00007FFCEAB39D60 db  37h ; 7
[stack]:00007FFCEAB39D61 db  33h ; 3
[stack]:00007FFCEAB39D62 db  38h ; 8
[stack]:00007FFCEAB39D63 db  31h ; 1
[stack]:00007FFCEAB39D64 db  39h ; 9
[stack]:00007FFCEAB39D65 db  34h ; 4
[stack]:00007FFCEAB39D66 db  30h ; 0
[stack]:00007FFCEAB39D67 db  35h ; 5
[stack]:00007FFCEAB39D68 db  32h ; 2
[stack]:00007FFCEAB39D69 db  36h ; 6
[stack]:00007FFCEAB39D6A db    0
[stack]:00007FFCEAB39D6B db    0
[stack]:00007FFCEAB39D6C db    0
```


整体流程;

输入10位0 1 2 3 4

数组变换

变换后的数组满足数独结果

```
tmp=[7,3,8,1,9,4,0,5,2,6]
result=['0','4','2','1','4','2','1','4','3','0']
flag=[0 for i in range(10)]
for i in range(10):
 flag[tmp[i]]=result[i]
print(''.join(flag))
#1134240024
```

3.[ACTF新生赛2020]Oruga

```
  char v5; // [rsp+9h] [rbp-37h]
  char s2[4]; // [rsp+Ah] [rbp-36h]
```

```
  char s[40]; // [rsp+10h] [rbp-30h]
  unsigned __int64 v8; // [rsp+38h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  memset(s, 0, 0x19uLL);
  printf("Tell me the flag:", 0LL);
  scanf("%s", s);
  strcpy(s2, "actf{");
  LODWORD(v4) = 0;                                  // v4共8个字节，低四个字节存放索引，高四个存放数据
  while ( (signed int)v4 <= 4 )
  {
    *((_BYTE *)&v4 + (signed int)v4 + 4) = s[(signed int)v4];
    LODWORD(v4) = v4 + 1;
  }
  v5 = 0;
  if ( !strcmp((const char *)&v4 + 4, s2) )         // 输入的前四个字节是actf{
  {
    if ( (unsigned __int8)sub_78A((__int64)s) ) // 主函数sub_78A
      printf("That's True Flag!", s2, v4);
    else
      printf("don't stop trying...", s2, v4);
    result = 0LL;
  }
  else
  {
    printf("Format false!", s2, v4);
    result = 0LL;
```


v4 8个字节，低四字节存索引，高四字节存数据

```
_BOOL8 __fastcall sub_78A(__int64 a1)
{
  int v2; // [rsp+Ch] [rbp-Ch]
  signed int v3; // [rsp+10h] [rbp-8h]
  signed int v4; // [rsp+14h] [rbp-4h]

  v2 = 0;
  v3 = 5;
  v4 = 0;
  while ( byte_201020[v2] != 0x21 )                 // 终点是0x21
  {
    v2 -= v4;                                       // v2目前的位置，v2减走到非0数字所走的路程
    if ( *(_BYTE *)(v3 + a1) != 'W' || v4 == -16 )
    {
      if ( *(_BYTE *)(v3 + a1) != 'E' || v4 == 1 )
      {
        if ( *(_BYTE *)(v3 + a1) != 'M' || v4 == 16 )
        {
          if ( *(_BYTE *)(v3 + a1) != 'J' || v4 == -1 )
            return 0LL;
          v4 = -1;                                  // J 左移
        }
        else
        {
          v4 = 16;                                  // M 下移
        }
      }
      else
      {
```

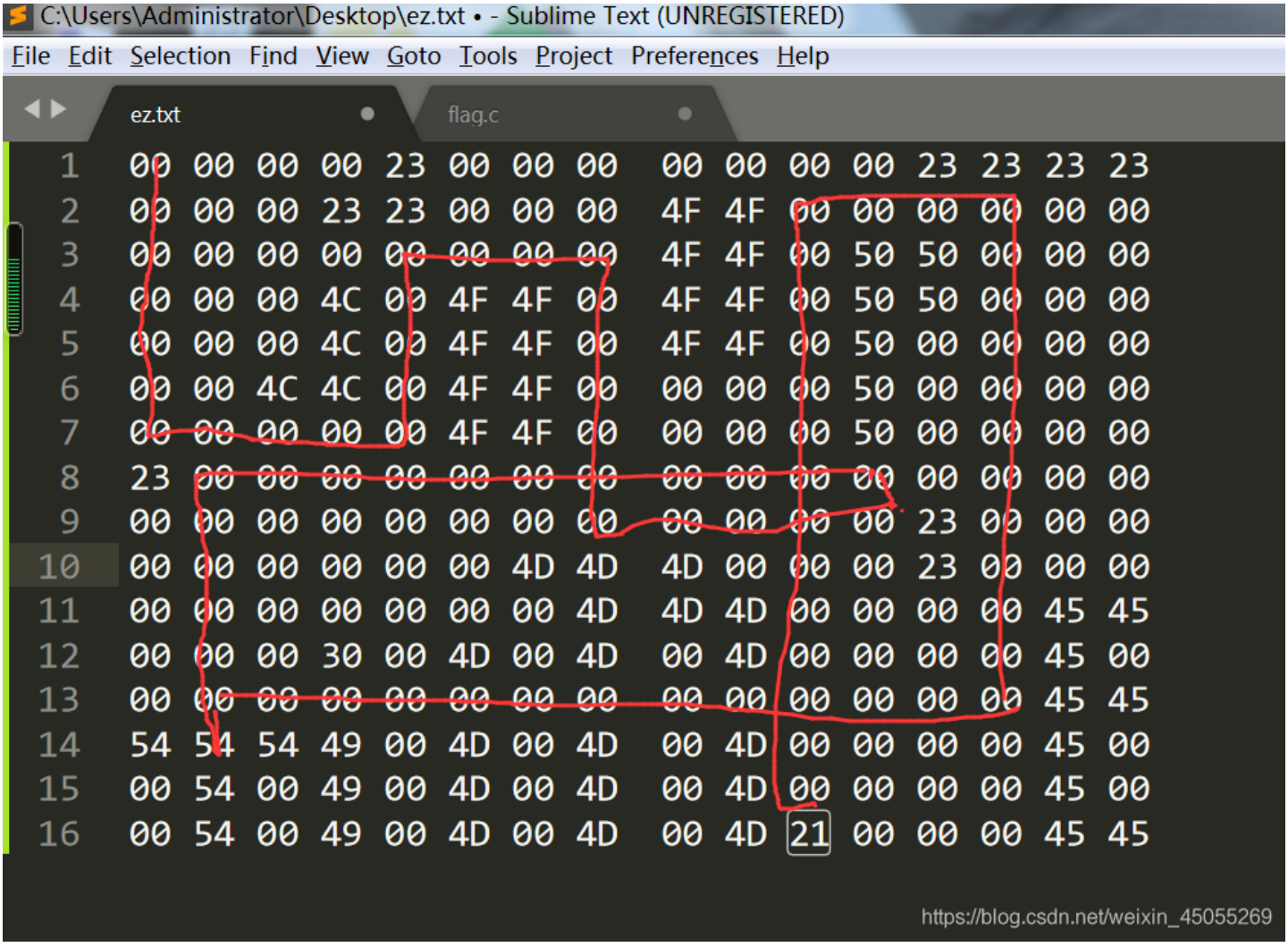


```
        {
          v4 = 16;                                  // M 下移
        }
      }
      else
      {
        v4 = 1;                                     // E 右移
      }
```

```
    }
    else
    {
      v4 = -16;                                 // W 上移
    }
    ++v3;
    while ( !byte_201020[v2] )                  // v2处字符为0，意思是只允许走0
    {
      if ( v4 == -1 && !(v2 & 0xF) )            // 最左列不能向左移
        return 0LL;
      if ( v4 == 1 && v2 % 16 == 15 )           // 最右列不能右移
        return 0LL;
      if ( v4 == 16 && (unsigned int)(v2 - 0xF0) <= 0xF )// 最后一行不能下移
        return 0LL;
      if ( v4 == -16 && (unsigned int)(v2 + 0xF) <= 0x1E )// 第一行不能上移
        return 0LL;
      v2 += v4;                                 // 在循环中，v2一直走，直到v2位置的字符非0，
    }                                           // 进行判断跳出循环，但是已经加上了v4，所以开头v2减v4还原v2的位置
  }
  return *(_BYTE *)(v3 + a1) == '}';
}
```


&运算结果为0的情况表示位置相当于取摸

地图题，亮点是地图的走法，类似于象棋里的'车'，可以一直走直到遇到障碍物。



flag{MEWEMEWJMEWJM}

4.[网鼎杯 2020 青龙组]singal

网鼎杯WP

5.[Zer0pts2020]easy strcmp

这种题我梦到过，简单的加减法，但要考虑小端存储。

关键函数sub_6EA()

```
__int64 __fastcall sub_6EA(__int64 a1, __int64 a2)
{
  int i; // [rsp+18h] [rbp-8h]
  int v4; // [rsp+18h] [rbp-8h]
  int j; // [rsp+1Ch] [rbp-4h]

  for ( i = 0; *(_BYTE *)(i + a1); ++i )
    ;
  v4 = (i >> 3) + 1;
  for ( j = 0; j < v4; ++j )
    *(_QWORD *)(8 * j + a1) -= qword_201060[j];
  return qword_201090(a1, a2);
}
```


每八位组成的数字相减

```
enc = "********CENSORED********"
m = [0x410A4335494A0942, 0x0B0EF2F50BE619F0, 0x4F0A3A064A35282B]

import binascii

flag = b''
for i in range(3):
    p = enc[i*8:(i+1)*8]
    a = binascii.b2a_hex(p.encode('ascii')[::-1])
    b = binascii.a2b_hex(hex(int(a,16) + m[i])[2:])[::-1]
    flag += b
print (flag)
```