BUUCTF reverse26-30 WP

原创

是Mumuzi 于 2021-04-07 17:50:29 发布 212 收藏 1

分类专栏： ctf reverse buuctf 文章标签： python 信息安全

本文链接：https://blog.csdn.net/qq_42880719/article/details/115439805

ctf 同时被 3 个专栏收录

75 篇文章 28 订阅

订阅专栏

reverse

8 篇文章 4 订阅

订阅专栏

buuctf

15 篇文章 2 订阅

订阅专栏

写脚本技术很菜，所以和其他wp相比看起来比较麻烦

笔记是按照当时的题目排序写的，顺序可能有出入

做题是从0开始做起，所以前面部分也会尽可能写的详细一点

只要能记录并且了解到怎么做即可，所以就没有去在意排版

遇到不会的函数尽可能去百度了解

题目：[SUCTF2019]SignIn、相册、[BJDCTF2020]easy、[ACTF新生赛2020]usualCrypt、[MRCTF2020]Transform

26.[SUCTF2019]SignIn

值制岭 flag 诽匈丐 flag{} 揖弈ザ

查壳，无壳，IDA打开，查看main函数

```
11  v10 = __readfsqword(0x28u);
12  puts("[sign in]");
13  printf("[input your flag]: ", a2);
14  __isoc99_scanf("%99s", &v8);
15  sub_96A(&v8, &v9);
16  __gmpz_init_set_str(&v7, "ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35", 16LL);
17  __gmpz_init_set_str(&v6, &v9, 16LL);
18  __gmpz_init_set_str(&v4, "10346103590081691412139010129904904441395040517371217043416168653987816098454 9", 10LL);
19  __gmpz_init_set_str(&v5, "65537", 10LL);
20  __gmpz_powm(&v6, &v6, &v5, &v4);
21  if ( (unsigned int)__gmpz_cmp(&v6, &v7) )
22    puts("GG!");
23  else
24    puts("TTTTTTTTTql!");
25  return 0LL;
26}
```


稍微看了一下，发现用到Pow就觉得不对劲，然后看到了65537，加上gmpz，就猜想是一道RSA题目。
翻阅https://gmplib.org/manual/发现猜想正确
所以得到c,n,e

```
c = 0xad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35
n = 10346103590081691412139010129904904441395040517371217043416168653987816098454 9
e = 65537
```

首先分解N，用的是http://www.factordb.com
得到

```
q = 28216458745951212484424511395059334827 1
p = 36666910200296685687660566983701422941 9
```

然后写脚本

```
import gmpy2
import binascii

c = 0xad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35
n = 10346103590081691412139010129904904441395040517371217043416168653987816098454 9
e = 65537
q = 28216458745951212484424511395059334827 1
p = 36666910200296685687660566983701422941 9

L = (q-1)*(p-1)
d = gmpy2.invert(e,L)
m = gmpy2.powmod(c,d,n)

print(binascii.unhexlify(hex(m)[2:]))
```

得到flag

```
C:\Users\mumuzi\PycharmProjects
b'suctf{Pwn_@_hundred_years}'
```

> flag{Pwn_@_hundred_years}

27. 盾冒

佑妃= 迟昵丐欧聊传盾牍= 佑盼盼 + 痂毘= 专开讵宏裢制扑杀= 揖娈宅歐邢篾卹乀flag - 泮愕 x 值制皊 flag 诽匈丐 flag{} 揖弈

apk逆向，GDA打开，不会em，以后学了补充。

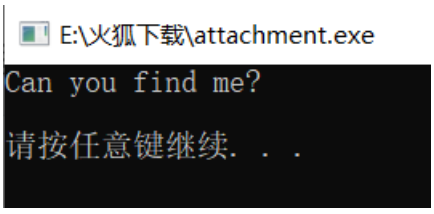
28. [BJDCTF2020]easy

值制皊 flag 诽匈丐 flag{} 揖弈ザ杜準 x https://github.com/BjdsecCA/BJDCTF2020

IDA打开，看main函数

```
int __cdecl main(int argc, const cha
{
  time_t v4; // [esp+10h] [ebp-3F0h]
  struct tm *v5; // [esp+3FCh] [ebp-
  __main();
  time(&v4);
  v5 = localtime(&v4);
  puts("Can you find me?\n");
  system("pause");
  return 0;
}
```


运行一下



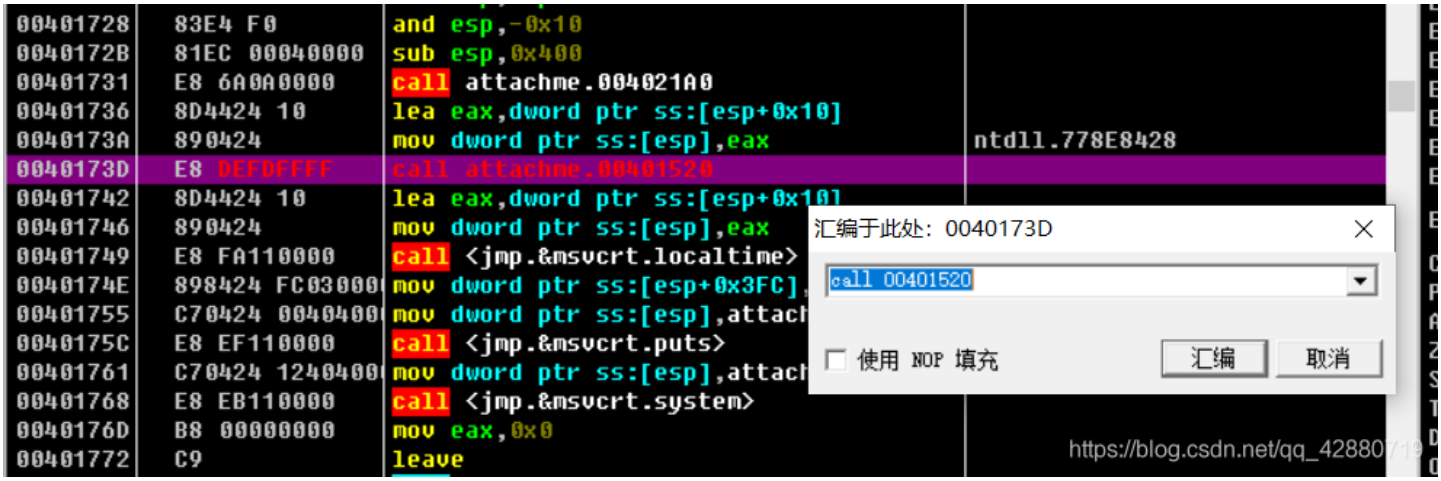
应该在别的函数里，首先看的就是main函数上面的ques，看了之后感觉挺像的，先从这入手

```
int ques()
{
  int v0; // edx
  int result; // eax
  int v2[50]; // [esp+20h] [ebp-128h]
  int v3; // [esp+E8h] [ebp-60h]
  int v4; // [esp+ECh] [ebp-5Ch]
  int v5; // [esp+F0h] [ebp-58h]
  int v6; // [esp+F4h] [ebp-54h]
  int v7; // [esp+F8h] [ebp-50h]
  int v8; // [esp+FCh] [ebp-4Ch]
  int v9; // [esp+100h] [ebp-48h]
  int v10; // [esp+104h] [ebp-44h]
  int v11; // [esp+108h] [ebp-40h]
  int v12; // [esp+10Ch] [ebp-3Ch]
  int j; // [esp+114h] [ebp-34h]
  __int64 v14; // [esp+118h] [ebp-30h]
  int v15; // [esp+124h] [ebp-24h]
  int v16; // [esp+128h] [ebp-20h]
  int i; // [esp+12Ch] [ebp-1Ch]

  v3 = 2147122737;
  v4 = 140540;
  v5 = -2008399303;
```

```
  v6 = 141956;
  v7 = 139457077;
  v8 = 262023;
  v9 = -2008923597;
  v10 = 143749;
  v11 = 2118271985;
  v12 = 143868;
  for ( i = 0; i <= 4; ++i )
  {
    memset(v2, 0, sizeof(v2));
    v16 = 0;
    v15 = 0;
    v0 = *(&v4 + 2 * i);
    LODWORD(v14) = *(&v3 + 2 * i);
    HIDWORD(v14) = v0;
    while ( SHIDWORD(v14) > 0 || v14 >= 0 && (_DWORD)v14 )
    {
      v2[v16++] = ((SHIDWORD(v14) >> 31) ^ (((unsigned __int8)(SHIDWORD(v14) >> 31) ^ (unsigned __int8)v14)
                                          - (unsigned __int8)(SHIDWORD(v14) >> 31)) & 1)
                - (SHIDWORD(v14) >> 31);
      v14 /= 2LL;
    }
    for ( j = 50; j >= 0; --j )
    {
      if ( v2[j] )
      {
        if ( v2[j] == 1 )
        {
          putchar('*');
          ++v15;
        }
      }
      else
      {
        putchar(' ');
        ++v15;
      }
      if ( !(v15 % 5) )
        putchar(' ');
    }
    result = putchar(10);
  }
  return result;
}
```

可以发现，最后的时候会在指定位置分别打印空格和*号，根据Can you find me，可以知道我们需要想办法进入ques函数
用OD，改time的地址为ques的地址（因为time没啥用）

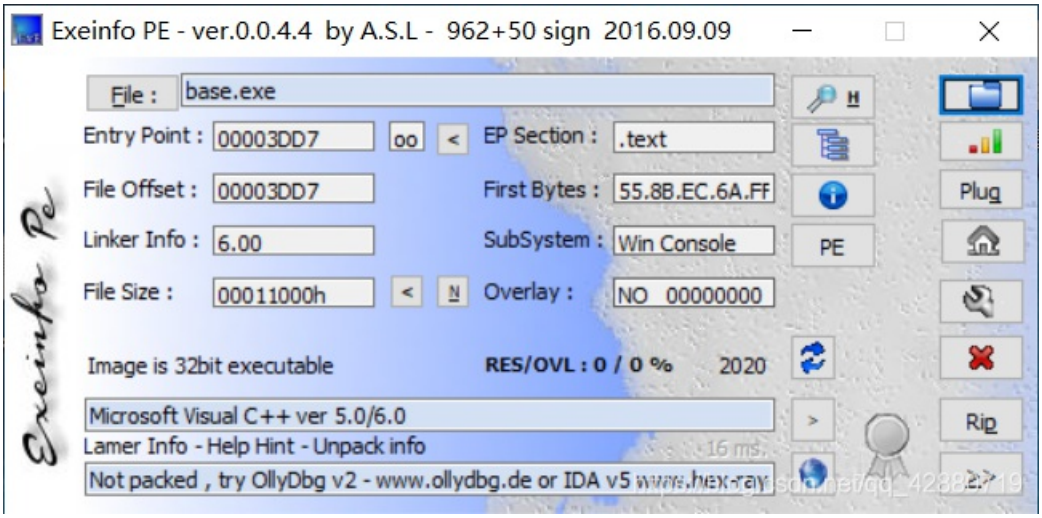


> flag{HACKIT4FUN}

29.[ACTF觸甥贍2020]usualCrypt

值制岭 flag 诽匈丐 flag{} 揖弈ザ

(题目名字是crypt，文件名字叫base，先猜测一波换表base)
查壳，发现无壳



IDA打开，找到main函数，F5大法

```
12  sub_403CF8(&unk_40E140);
13  scanf(aS, &v10);
14  v5 = 0;
15  v6 = 0;
16  v7 = 0;
17  v8 = 0;
18  v9 = 0;
19  sub_401080(&v10, strlen(&v10), &v5);
20  v3 = 0;
21  while ( *((_BYTE *)&v5 + v3) == byte_40E0E4[v3] )
22  {
23    if ( ++v3 > strlen((const char *)&v5) )
24      goto LABEL_6;
25  }
26  sub_403CF8(aError);
27LABEL_6:
28  if ( v3 - 1 == strlen(byte_40E0E4) )
29    result = sub_403CF8(aAreYouHappyYes);
30  else
31    result = sub_403CF8(aAreYouHappyNo);
32  return result;
33}
```


首先查看第一个while，双击byte_40E0E4

```
.data:0040E0A0 byte_40E0A0     db 'A'                  ; DATA XREF: sub_401000:loc_401005↑r
.data:0040E0A0                                         ; sub_401000+17↑w ...
.data:0040E0A1                 db  42h ; B
.data:0040E0A2                 db  43h ; C
.data:0040E0A3                 db  44h ; D
.data:0040E0A4                 db  45h ; E
.data:0040E0A5                 db  46h ; F
.data:0040E0A6                 db  47h ; G
.data:0040E0A7                 db  48h ; H
.data:0040E0A8                 db  49h ; I
.data:0040E0A9                 db  4Ah ; J
.data:0040E0AA byte_40E0AA     db 'K'                  ; DATA XREF: sub_401000+B↑r
.data:0040E0AA                                         ; sub_401000+11↑w
.data:0040E0AB aLmnopqrstuvwxy db 'LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/',0
.data:0040E0E1                 align 4
.data:0040E0E4 ; char byte_40E0E4[]
.data:0040E0E4 byte_40E0E4     db 'z'                  ; DATA XREF: _main+5C↑r
.data:0040E0E4                                         ; _main:loc_401238↑o
.data:0040E0E5 aMxhz3tignxlxjh db 'MXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9',0
.data:0040E109                 align 4
.data:0040E10C aAreYouHappyNo  db 'Are you happy?No!',0Ah,0
.data:0040E10C                                         ; DATA XREF: _main:loc_40125F↑o
.data:0040E11F                 align 10h
.data:0040E120 aAreYouHappyYes db 'Are you happy?yes!',0Ah,0
```


……结果显而易见了，换表base和密文都给了，尝试解码，但是发现解不出来（啊这）
去研究一下main函数，之前进入的函数是判断函数，所以这次进入sub_401080加密函数

```
int __cdecl sub_401080(int a1, int a2, int a3)
{
  int v3; // edi
  int v4; // esi
  int v5; // edx
```

```
  int v6; // eax
  int v7; // ecx
  int v8; // esi
  int v9; // esi
  int v10; // esi
  int v11; // esi
  _BYTE *v12; // ecx
  int v13; // esi
  int v15; // [esp+18h] [ebp+8h]

  v3 = 0;
  v4 = 0;
  sub_401000();
  v5 = a2 % 3;
  v6 = a1;
  v7 = a2 - a2 % 3;
  v15 = a2 % 3;
  if ( v7 > 0 )
  {
    do
    {
      LOBYTE(v5) = *(_BYTE *)(a1 + v3);
      v3 += 3;
      v8 = v4 + 1;
      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[(v5 >> 2) & 0x3F];
      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(a1 + v3 - 3) & 3)
                                            + (((signed int)*(unsigned __int8 *)(a1 + v3 - 2) >> 4) & 0xF)];
      *(_BYTE *)(v8 + a3 - 1) = byte_40E0A0[4 * (*(_BYTE *)(a1 + v3 - 2) & 0xF)
                                          + (((signed int)*(unsigned __int8 *)(a1 + v3 - 1) >> 6) & 3)];
      v5 = *(_BYTE *)(a1 + v3 - 1) & 0x3F;
      v4 = v8 + 1;
      *(_BYTE *)(v4 + a3 - 1) = byte_40E0A0[v5];
    }
    while ( v3 < v7 );
    v5 = v15;
  }
  if ( v5 == 1 )
  {
    LOBYTE(v7) = *(_BYTE *)(v3 + a1);
    v9 = v4 + 1;
    *(_BYTE *)(v9 + a3 - 1) = byte_40E0A0[(v7 >> 2) & 0x3F];
    v10 = v9 + 1;
    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3)];
    *(_BYTE *)(v10 + a3) = 61;
LABEL_8:
    v13 = v10 + 1;
    *(_BYTE *)(v13 + a3) = 61;
    v4 = v13 + 1;
    goto LABEL_9;
  }
  if ( v5 == 2 )
  {
    v11 = v4 + 1;
    *(_BYTE *)(v11 + a3 - 1) = byte_40E0A0[((signed int)*(unsigned __int8 *)(v3 + a1) >> 2) & 0x3F];
    v12 = (_BYTE *)(v3 + a1 + 1);
    LOBYTE(v6) = *v12;
    v10 = v11 + 1;
    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3) + ((v6 >> 4) & 0xF)];
    *(_BYTE *)(v10 + a3) = byte_40E0A0[4 * (*v12 & 0xF)];
    goto LABEL_8;
```

```
  }
LABEL_9:
  *(_BYTE *)(v4 + a3) = 0;
  return sub_401030((const char *)a3);
}
```

首先查看sub_401000函数

```
{
  signed int result; // eax
  char v1; // cl

  result = 6;
  do
  {
    v1 = byte_40E0AA[result];
    byte_40E0AA[result] = byte_40E0A0[result];
    byte_40E0A0[result++] = v1;
  }
  while ( result < 15 );
  return result;
}
```


其中，byte_40E0AA是

> KLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

byte_40E0A0是

> ABCDEFGHIJ

研究此循环，发现是将表中QRSTUVWXY和GHIJKLMNO进行了交换
（因为可以将此表拼接起来成为ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/ 然后又分开看，可以发现40E0AA是从K开始，40E0A0是从A开始，所以是这样换的表）
手动替换一下

> ABCDEFQRSTUVWXYPGHIJKLMNOZabcdefghijklmnopqrstuvwxyz0123456789+/

然后吧，中间就是base过程，但是最后有个函数（sub_401030）

```
  v2 = a1[HIDWORD(v1)];
  if ( v2 < 97 || v2 > 122 )
  {
    if ( v2 < 65 || v2 > 90 )
      goto LABEL_9;
    LOBYTE(v1) = v2 + 32;
  }
  else
  {
    LOBYTE(v1) = v2 - 32;
  }
  a1[HIDWORD(v1)] = v1;
```

明显的大小写转换，在线转一下

```
string = 'zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9'.swapcase() #sub_401030()
print (string)
```

```
1 string = 'zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9'.swapcase() #sub_401030()
2 print (string)
```

ZmxhZ3tiGNXlXjHfaDTzN2FfK3LycRTpc2L9

得到之后再在线换表转一下

1、输入 编码的任意文本:

ZmxhZ3tiGNXlXjHfaDTzN2FfK3LycRTpc2L9

2、输入自定义映射字符 自定义base64

ABCDEFQRSTUVWXYPGHIJKLMNOZabcdefghijklmnopqrstuvwxyz0123456789+/

Base类型: base64 编码: gb2312编码（简体 填充: =

编码 解密

编解码数据:

flag{bAse64_h2s_a_Surprise}

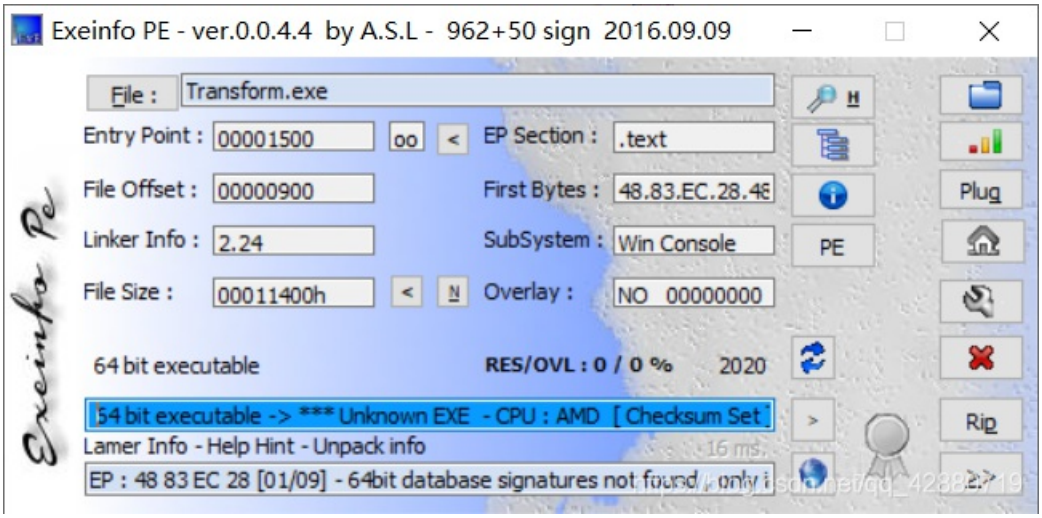
> flag{bAse64_h2s_a_Surprise}

30.[MRCTF2020]Transform

值制岭 flag 诽匈丐 flag{} 揖弯ザ

恻谨夯瑾戛陬侕颞ザ

夯瑾戛陬幹叶 xhttp://ctf.merak.codes/



IDA64

```
10  sub_402230(argc, argv, envp);
11  sub_40E640(argc, (__int64)argv, v3, (__int64)"Give me your code:\n");
12  sub_40E5F0(argc, (__int64)argv, (__int64)v6, (__int64)"%s");
13  if ( strlen(*(const char **)&argc) != 33 )
14  {
15    sub_40E640(argc, (__int64)argv, v4, (__int64)"Wrong!\n");
16    system(*(const char **)&argc);
17    exit(argc);
18  }
19  for ( i = 0; i <= 32; ++i )
20  {
21    byte_414040[i] = v6[dword_40F040[i]];
22    v4 = i;
23    byte_414040[i] ^= LOBYTE(dword_40F040[i]);
24  }
25  for ( j = 0; j <= 32; ++j )
26  {
27    v4 = j;
28    if ( byte_40F0E0[j] != byte_414040[j] )
29    {
30      sub_40E640(argc, (__int64)argv, j, (__int64)"Wrong!\n");
31      system(*(const char **)&argc);
32      exit(argc);
33    }
34  }
35  sub_40E640(argc, (__int64)argv, v4, (__int64)"Right!Good Job!\n");
36  sub_40E640(argc, (__int64)argv, (__int64)v6, (__int64)"Here is your flag: %s\n");
37  system(*(const char **)&argc);
38  return 0;
39}
```


要求长度33，然后进入循环，最后进行判断

判断是将byte_40E640与byte_414040进行判断，v6是我们输入的flag，在循环里会将其打乱之后赋值给byte_414040，然后byte_414040与我们输入的flag进行异或，现在已经给出了异或之后的值(byte_40F0E0中)，可以写脚本了

```python
enc = [103, 121, 123, 127, 117,  43,  60,  82,  83, 121, 87,  94,  93,  66, 123,  45,  42, 102,  66, 126, 76,  8
7, 121,  65, 107, 126, 101,  60,  92,  69, 111,  98,  77]#3F dup(0)是3F是0的意思
chg = [9, 10, 15, 23, 7, 24, 12, 6, 1, 16, 3, 17, 32, 29, 11,30, 27, 22, 4, 13, 19, 20, 21, 2, 25, 5, 31, 8, 18,
26, 28, 14,0] #同理8 dup(0)是8个0的意思
tmp = ''
flag = ['']*len(enc)
for i in range(len(enc)):
    tmp += chr(enc[i]^chg[i]) #byte_414040
#print(tmp)
for i in range(len(tmp)):
    flag[chg[i]] += chr(ord(tmp[i]))
print(''.join(flag))
```

```
C:\Users\mumuzi\PycharmProjects\pythonProject\venv\Sc
MRCTF{Tr4nsp0sltiON_Clph3r_1s_3z}
```

> flag{Tr4nsp0sltiON_Clph3r_1s_3z}