moectf逆向题目writeup

原创

NoOneGroup 于 2019-11-17 10:30:26 发布 300 收藏
分类专栏： 逆向 reverse windows

本文链接：https://blog.csdn.net/u010334666/article/details/82975573


逆向 同时被 3 个专栏收录
18 篇文章 3 订阅
订阅专栏


reverse
15 篇文章 0 订阅
订阅专栏


windows
1 篇文章 0 订阅
订阅专栏

博客已经转移

https://noone-hub.github.io/

废话不多说，直接上题解

re1:很简单的题目，用strings命令就查出来flag了

```
greenhand@greenhand-Inspiron-3568:~/moeCTF$ strings re1.exe | grep ctf
moectf{Qidao_by_falcon}
```


re2（xorme):有点小坑，得手动做

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char *Format; // [sp+0h] [bp-98h]@0
  char v5; // [sp+40h] [bp-58h]@1
  char v6; // [sp+41h] [bp-57h]@1
  char v7; // [sp+42h] [bp-56h]@1
  char v8; // [sp+43h] [bp-55h]@1
  char v9; // [sp+44h] [bp-54h]@1
  char v10; // [sp+45h] [bp-53h]@1
  char v11; // [sp+46h] [bp-52h]@1
  char v12; // [sp+47h] [bp-51h]@1
  char v13; // [sp+48h] [bp-50h]@1
  char v14; // [sp+49h] [bp-4Fh]@1
  char v15; // [sp+4Ah] [bp-4Eh]@1
  char v16; // [sp+4Bh] [bp-4Dh]@1
  char v17; // [sp+4Ch] [bp-4Ch]@1
```

```
  char v18; // [sp+4Dh] [bp-4Bh]@1
  char v19; // [sp+4Eh] [bp-4Ah]@1
  char v20; // [sp+4Fh] [bp-49h]@1
  char v21; // [sp+50h] [bp-48h]@1
  char v22; // [sp+51h] [bp-47h]@1
  char v23; // [sp+52h] [bp-46h]@1
  char v24; // [sp+53h] [bp-45h]@1
  char Buf; // [sp+60h] [bp-38h]@1
  int i; // [sp+8Ch] [bp-Ch]@1
  char v27[8]; // [sp+90h] [bp-8h]@3

  _alloca((size_t)Format);
  __main();
  v5 = 24;
  v6 = 53;
  v7 = 21;
  v8 = 54;
  v9 = 53;
  v10 = 21;
  v11 = 87;
  v12 = 32;
  v13 = 21;
  v14 = 42;
  v15 = 79;
  v16 = 32;
  v17 = 76;
  v18 = 54;
  v19 = 122;
  v20 = 54;
  v21 = 32;
  v22 = 21;
  v23 = 65;
  v24 = 123;
  printf("Please enter the flag:");
  fgets(&Buf, 29, __iob);
  for ( i = 0; i <= 19; ++i )
    v27[i - 48] ^= v27[i - 80];
  for ( i = 0; i <= 21 && v27[i - 48] == v27[i - 112]; ++i )
    ;
  if ( i == 22 )
    printf("conglution!");
  gets(&Buf);
  return 0;
}
```

ida的结果很不如愿，数组越界，变量没当成数组，很麻烦，根据偏移自己推算地址

贴上算好后的代码

```
#include <iostream>

using namespace std;

void KeyInit(char *XorKey)
{
    XorKey[0] = 24;
    XorKey[1] = 53;
    XorKey[2] = 21;
    XorKey[3] = 54;
    XorKey[4] = 53;
    XorKey[5] = 21;
    XorKey[6] = 87;
    XorKey[7] = 32;
    XorKey[8] = 21;
    XorKey[9] = 42;
    XorKey[10] = 79;
    XorKey[11] = 32;
    XorKey[12] = 76;
    XorKey[13] = 54;
    XorKey[14] = 122;
    XorKey[15] = 54;
    XorKey[16] = 32;
    XorKey[17] = 21;
    XorKey[18] = 65;
    XorKey[19] = 123;
}you win the game!
wait for a minute,I'm preparing the flag....
moectf{Would_u_Like_Cola?}


int main()
{
    char XorKey[23]={0};
    char JudgeKey[22]={"uZpUAs,IFu9e>O%s`f8Z!"};
    char flag[22]={0};
    KeyInit(XorKey);
    for(int i=0;i<22;i++)
        flag[i]=XorKey[i]^JudgeKey[i];
    cout << flag << "}" << endl;

    return 0;
}
```

第三道python逆向，给了个pyc文件，自己百度可以查到这是什么，如何进行反编译，顺便给上链接了

https://tool.lu/pyc/

```
#!/usr/bin/env python
# visit http://tool.lu/pyc/ for more information
import base64
r = open('flag', 'r')
rflag = 'bX;oY4Tpe4D8Q2;VRW:{U2;IQIP8fR?@'
key = base64.b64encode(r.read())
flag = ''
for i in range(len(key) / 4):
    for j in range(4):
        flag += chr(ord(key[i * 4 + j]) + j)


if rflag == flag:
    print 'You are right.'
```


pyc反编译后的代码，然后读懂逻辑，便可以逆向了

```
#!/usr/bin/env python
# coding=utf-8
import base64
rflag = 'bX;oY4Tpe4D8Q2;VRW:{U2;IQIP8fR?@'

Len=len(rflag)
flag=''
for i in range(Len):
    flag+=chr(ord(rflag[i])-i%4)

flag=base64.b64decode(flag)
print flag
```

第四道，线性代数，考线代，题目代码，

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  bool v4; // eax
  int v6; // [esp+1Bh] [ebp-8Dh]
  char v7; // [esp+1Fh] [ebp-89h]
  int v8; // [esp+8Fh] [ebp-19h]
  char v9; // [esp+93h] [ebp-15h]
  int j; // [esp+94h] [ebp-14h]
  int i; // [esp+98h] [ebp-10h]
  int v12; // [esp+9Ch] [ebp-Ch]

  __main();
  v6 = 0;
  v8 = 0;
  memset(
    (void *)((unsigned int)&v7 & 0xFFFFFFFC),
    0,
    4 * (((unsigned int)((char *)&v6 - ((unsigned int)&v7 & 0xFFFFFFFC) + 120) & 0xFFFFFFFC) >> 2));
  v9 = 0;
  v12 = 0;
  while ( 1 )
  {
    v9 = getchar();
    v4 = v9 != 10 && v12 <= 119;
    if ( !v4 )
      break;
    v3 = v12++;
    *((_BYTE *)&v6 + v3) = v9;
  }
  for ( i = 0; i <= 40; ++i )
  {
    for ( j = 0; j <= 40; ++j )
      Q[i] -= M[j + 41 * i] * *((char *)&v6 + j);
    if ( Q[i] )
      return 0;
  }
  puts("Congratulates!");
  return 0;
}
```

其实原来这道题不会做的，然后自己手动写变量之间关系式就知道了，第一次循环M[0——40】分别跟 input[0——40]进行相乘，input是未知数也就是变量x，然后第二次是M[41------41+41]跟input[0——40]相乘，以此类推可以得到一个41*41的系数矩阵和input矩阵相乘等于Q，所以可推出AB=C,我们B为未知，也就是B=A的逆*C，这是一种思路，不过也可以直接用系数矩阵M，左边，然后右边是数值，具体知识自己补线代吧，不懂问我，这里有个知识点是，数据的处理，我原来自己手动复制数据手动处理了，太麻烦了，原来有现成脚本可以用，lazyida了解下，具体使用我开另一篇文章写

第五道4096，简单的patch，读懂逻辑，强行爆破就好了，od里面也可以改标志位越过，ce也可以修改，od搜索16进制常量1000，然后搞掉他就好了

题目链接: https://pan.baidu.com/s/1DRLgCLmCLshizOAWsTp14A 提取码: 83sq