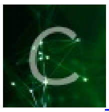
Writeup of NJUPT CTF platform's some easy Reverse

原创

C0ss4ck 于 2017-10-30 21:49:09 发布 996 收藏
分类专栏： Reverse_of_CTF 文章标签： 逆向 CTF wp


Reverse_of_CTF 专栏收录该内容
7 篇文章 0 订阅
订阅专栏

2^{nd}-ReadAsm2

本题主要考查基础的汇编阅读能力

```
int main(int argc, char const *argv[])
{
  char input[] = {0x0,  0x67, 0x6e, 0x62, 0x63, 0x7e, 0x74, 0x62, 0x69, 0x6d,
                  0x55, 0x6a, 0x7f, 0x60, 0x51, 0x66, 0x63, 0x4e, 0x66, 0x7b,
                  0x71, 0x4a, 0x74, 0x76, 0x6b, 0x70, 0x79, 0x66 , 0x1c};
  func(input, 28);
  printf("%s\n",input+1);
  return 0;
}

00000000004004e6 <func>:
  4004e6: 55                    push   rbp                          rbp寄存器，入栈
  4004e7: 48 89 e5              mov    rbp,rsp                      rdp=rsp
  4004ea: 48 89 7d e8           mov    QWORD PTR [rbp-0x18],rdi     rbp中的首地址0x18（考虑到数组与首地址的关系）input=rdi（引入input）
  4004ee: 89 75 e4              mov    DWORD PTR [rbp-0x1c],esi     从esi寄存器引入第二个量28
  4004f1: c7 45 fc 01 00 00 00  mov    DWORD PTR [rbp-0x4],0x1      rbq中0x1地址赋值为0x1
  4004f8: eb 28                 jmp    400522 <func+0x3c>           ------------------------------
  4004fa: 8b 45 fc              mov    eax,DWORD PTR [rbp-0x4]      eax=1
  4004fd: 48 63 d0              movsxd rdx,eax                      rdx=eax=1
  400500: 48 8b 45 e8           mov    rax,QWORD PTR [rbp-0x18]     rax=*input
  400504: 48 01 d0              add    rax,rdx                      rax=rax+rdx=*input+1
  400507: 8b 55 fc              mov    edx,DWORD PTR [rbp-0x4]      edx=1
  40050a: 48 63 ca              movsxd rcx,edx                      rcx=edx=1
  40050d: 48 8b 55 e8           mov    rdx,QWORD PTR [rbp-0x18]     rdx=*input
  400511: 48 01 ca              add    rdx,rcx                      rdx=rdx+rcx=*input+1
  400514: 0f b6 0a              movzx  ecx,BYTE PTR [rdx]           ecx=*input+1
  400517: 8b 55 fc              mov    edx,DWORD PTR [rbp-0x4]      edx=1
  40051a: 31 ca                 xor    edx,ecx                      edx=edx^ecx=1^*input+1='f'
  40051c: 88 10                 mov    BYTE PTR [rax],dl            rax=NULL
  40051e: 83 45 fc 01           add    DWORD PTR [rbp-0x4],0x1      *rbp-ox4+=1=2
  400522: 8b 45 fc              mov    eax,DWORD PTR [rbp-0x4]      eax=2
  400525: 3b 45 e4              cmp    eax,DWORD PTR [rbp-0x1c]     compare 2 with 28
  400528: 7e d0                 jle    4004fa <func+0x14>           ------------------------------
  40052a: 90                    nop
  40052b: 5d                    pop    rbp                                    rbp=null
  40052c: c3                    ret
```

改写成python脚本：
　　详见project：fresher01/cm1.py


fresher01 〉 cm1.py 〉
Project　part of base64.py　fkdflkdfldk.py　ASCII.py　cm.py　cm1.py　numerous
fresher01 C:\Users\
asassas.py
ASCII.py
base.py
base64.txt
cm.py
cm1.py
code.txt

```
#!usr/bin/env python
input = [0x0,  0x67, 0x6e, 0x62, 0x63, 0x7e, 0x74, 0x62, 0x69, 0x6d,
         0x55, 0x6a, 0x7f, 0x60, 0x51, 0x66, 0x63, 0x4e, 0x66, 0x7b,
         0x71, 0x4a, 0x74, 0x76, 0x6b, 0x70, 0x79, 0x66 , 0x1c]
a=''
for i in range(1,29):
    a+=chr(input[i]^i)
```

Run cm1

```
C:\Python27\python.exe C:/Users/76923/PycharmProjects/fresher01/cm1.py
flag{read_asm_is_the_basic}

Process finished with exit code 0
```


顺利拿到flag

4th-WxyVM

这题目，贼鸡儿烦~

话不多说，IDA pro打开**WxyVM1**，发现是**ELF**文件，找到**main**函数，发现调用了**sub_4005B6**函数，**F5**调出伪代码

```
__int64 sub_4005B6()
{
  unsigned int v0; // ST04_4@3
  __int64 result; // rax@3
  signed int i; // [sp+0h] [bp-10h]@1
  char v3; // [sp+8h] [bp-8h]@3

  for ( i = 0; i <= 14999; i += 3 )
  {
    v0 = byte_6010C0[(signed __int64)i];
    v3 = byte_6010C0[(signed __int64)(i + 2)];
    result = v0;
    switch ( v0 )
    {
      case 1u:
        result = byte_6010C0[(signed __int64)(i + 1)];
        *(&byte_604B80 + result) += v3;
        break;
      case 2u:
        result = byte_6010C0[(signed __int64)(i + 1)];
        *(&byte_604B80 + result) -= v3;
        break;
      case 3u:
        result = byte_6010C0[(signed __int64)(i + 1)];
        *(&byte_604B80 + result) ^= v3;
        break;
      case 4u:
        result = byte_6010C0[(signed __int64)(i + 1)];
        *(&byte_604B80 + result) *= v3;
        break;
      case 5u:
        result = byte_6010C0[(signed __int64)(i + 1)];
        *(&byte_604B80 + result) ^= *(&byte_604B80 + byte_6010C0[(signed __int64)(i + 2)]);
        break;
      default:
        continue;
    }
  }
  return result;
}
```


发现byte_6010C0数组的15000个元素按顺序三个一组分成5000组子序列，记子序列为An[3],n={1,2,3,……,5000},An中的三个元素主导着该函数运行

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char v4; // [sp+Bh] [bp-5h]@1
  signed int i; // [sp+Ch] [bp-4h]@3

  puts("[WxyVM 0.0.1]");
  puts("input your flag:");
  scanf("%s", &byte_604B80);
  v4 = 1;
  sub_4005B6();
  if ( strlen(&byte_604B80) != 24 )
    v4 = 0;
  for ( i = 0; i <= 23; ++i )
  {
    if ( *(&byte_604B80 + i) != dword_601060[i] )
      v4 = 0;
  }
  if ( v4 )
    puts("correct");
  else
    puts("wrong");
  return 0LL;
}
```


不妨认为byte_604B80数组有24个元素，

且发现dword_601060 = [C4,34,22,B1,D3,11,97,7,DB,37,C4,6,1D,FC,5B,ED,98,DF,94,D8,B3,84,CC,8,]

(注意数组各元素的地址逐个间隔4位，因为dword为4字节型，所以不要把多余的FF，00读入)

但是正当我打算写python脚本时，不知道该怎么把byte_6010C0给dump下来……于是疯狂百度，求助IDC（IDA pro自带的脚本）

Shift+F2召唤Execute script怼脚本，借鉴了无数例子（甚至某个巨佬CSDN上的wp）之后终于成功

Please enter script body

```
auto i;
for ( i = 14997; i >= 0; i = i - 3 )
  {
    auto v0 = Byte(0x6010C0+i);
    auto v3 = Byte(0x6010C0+(i + 2));
    auto result = v0;
    if(v0==1){
        result =Byte(0x6010C0+(i + 1));
        PatchByte(0x601060 + result*4,Byte(0x601060 + result*4)-v3);
            }
    if(v0==2){
        result =Byte(0x6010C0+(i + 1));
        PatchByte(0x601060 + result*4,Byte(0x601060 + result*4)+v3);
            }
    if(v0==3){
        result =Byte(0x6010C0+(i + 1));
        PatchByte(0x601060 + result*4,Byte(0x601060 + result*4)^v3);
            }
    if(v0==4){
        result =Byte(0x6010C0+(i + 1));
        PatchByte(0x601060 + result*4,Byte(0x601060 + result*4)/v3);
            }
    if(v0==5){
        result =Byte(0x6010C0+(i + 1));
        PatchByte(0x601060 + result*4,Byte(0x601060 + result*4)^Byte(0x601060+v3*4));
            }
    else
        continue;

  }

for(i=0;i<24;i++)
Message("%c",Byte(0x601060+i*4))
```

Line:31 Column:1

4

Run　Export　Import　Save

运行得结果nctf{Embr4ce_Vm_j0in_R3}

（默默说一句……IDC脚本真的是懒人福利，有些时候就是抄一抄函数~~~）

（emmmmmmmIDApython也是不错的脚本）