2018NCTF 部分writeup

原创

aiQG_ 于 2018-11-27 14:29:41 发布 710 收藏
分类专栏： CTF

本文链接：https://blog.csdn.net/qq_42863361/article/details/84565378

> Welcome to http://aiqg.vip/

~~我才不写WP~~

Our 16bit wars

==Difficulty: Easy==

逆向第一步，学好汇编，16位的也要学

// 64位系统运行需要DosBox

==Author: acdxvfsvd==

读十六位汇编

程序很小基本一目了然
运行有打印字符(call了一个打印函数)
有等待输入的函数(call了一个输入函数)
能猜测下一个函数就是check函数，进去

逐个取输入字符进行运算然后对比结果(向上翻一下就能看见)

```
for c in str:
    print(chr((c<<3)+(c>>5),end='')
```

后门后门后门

==Difficulty: Very Easy==

IDA了解一下

==Author: MozhuCY==

Some Boxes

==Difficulty: medium==

这只兔子不会抖脸的

nc ctfgame.acdxvfsvd.net 30010

==Author: MozhuCY==

//给了nc 我还以为是打pwn…
看到有对几个字符的判断 猜测是迷宫之类的东西

cat flag之前判断了两处

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

```
1  __int64 sub_400E58()
2  {
3    __int64 result; // rax@1
4
5    result = unk_6020A0[(16 * dword_6021DC + dword_6021D8)];
6    if ( result == 20 )
7    {
8      result = unk_6020A0[(16 * dword_6021D4 + dword_6021D0)];
9      if ( result == 20 )
10     {
11       puts(&byte_4018EE);
12       system("cat flag");
13       exit(0);
14     }
15   }
16   return result;
17 }
```

然而找到的地图只是一串奇怪的字符

尝试着跑起来看看程序干了什么

发现地图变成了只有8、0和两个0x14的字符串
复制出来整理一下

//黄色的是起始位置, oo 是箱子, x14 是要把箱子推到的地方
在打印兔子脸之前初始化了两个箱子和起始位置

```
1 void sub_400796()
2 {
3   dword_6021B0 = 8;
4   dword_6021B4 = 5;
5   dword_6021D8 = 5;
6   dword_6021DC = 2;
7   dword_6021D0 = 8;
8   dword_6021D4 = 7;
9 }
```

分析每个分支函数的第一句代码可以知道 'W'向上, '4'向左, '5'向下, '0'向右
然后看着地图推箱子就行了

基本操作 \~Reverse Version.~

==Difficulty: Easy++==

This is fundamental.

==Author: acdxvfsvd==

打开发现main函数中有两个strcmp 密文都已经给了

发现转换的函数(sub_400666、sub_4006BE)是两个迭代的函数,都是将input中的字符一个个放进待对比的字符串中
那我们反过来跑一下就好了

```
#include <iostream>
using namespace std;
char s1[]="bcec8d7dcda25d91ed3e0b720cbb6cf202b09fedbc3e017774273ef5d5581794";
char s2[]="7d8dcdcaed592e1dcb07e02c36bcb2f0bf9e0bdcb0e13777237e25fd48515974";
char input1[100]={0};
int i1=0;
unsigned int func1(unsigned int a)
{
    int result;
    int v1=0;
    if(a<=63)
    {
        v1=i1++;
        input1[a]=s1[v1];
        func1(2 * a + 1);
        result = func1(2 * (a + 1));
    }

    return result;
}
int i2=0;
unsigned int func2(int a, int b)
{
    int v2;
    int result;

    if(a<=63)
    {
        func2((2 * a + 1), b);
        v2 = i2++;
        input1[a]=s2[v2];
        result=func2((2 * (a + 1)), b);
    }
    return result;
}
int main()
{
    func1(0);
    cout<<input1<<endl;
    func2(0, 0);
    cout<<input1<<endl;

    return 0;
}
```

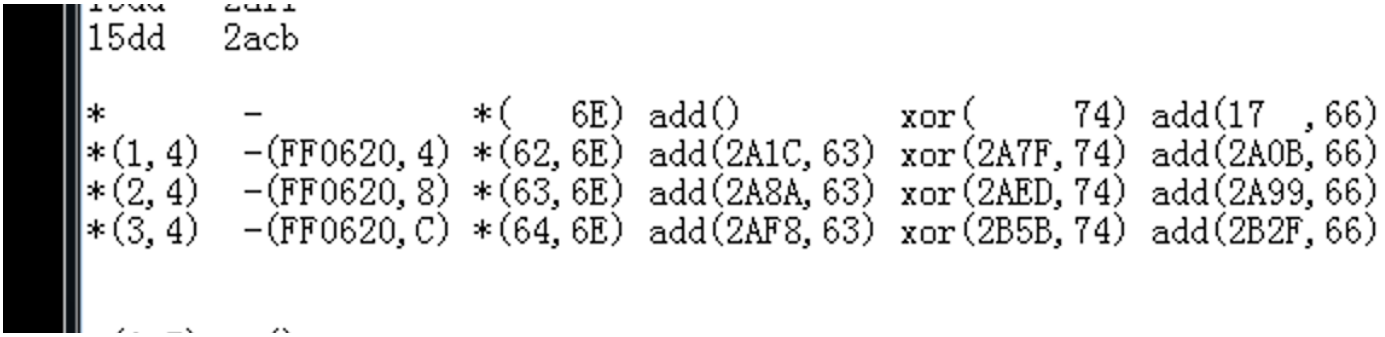
WcyVM

==Difficulty: Hard==

==Author: MozhuCY==

很经典的vm
指令集在 sub_400DAB中
运算指令在下半部分

至于它会怎么运算...
随便输入一串字符开始调试
断点下在add sub xor or and mul 和check处
得到调用运算的顺序与参数



> input *0x6E +0x63 ^0x74 +0x66从左到右计算得到密文

密文就是dword_6020A0处的数组
脚本反着跑一遍就好

Havefun

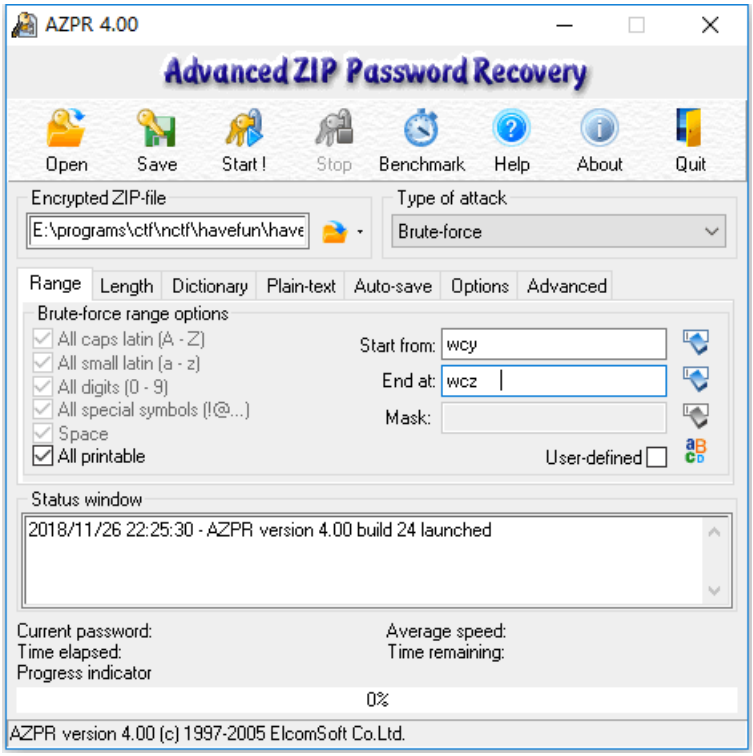
==DIfficulty: medium==

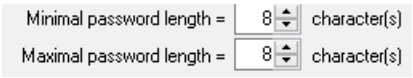
郁离歌！郁离歌！

flag是一串有意义的字符

==Author: ccc==

一个压缩包, 有密码里面有一个txt, 提示了密码是wcy_____ (后面是5个字节)





(cchan: 开始结束后面都要填5个空格占位)

压缩包打开后是一百张看似一样的gif //要是全都一样还藏什么flag
对比文件的md5

> 文件md5对比

挑出了十五张不同的图片

图片每帧的速度显然不同
用identify 查看每帧时间

```
aiqg@ubuntu:~/Desktop/havefun/havefun % identify -format "%T\n" 77.gif
10
20
20
10
10
20
20
10
```

将20作为二进制的1
将10作为二进制的0
正好八位一个字节
转换得到flag

```
import os

result = ''

for root,dirs,files in os.walk('.'):
    for file in files:
        if file[-4:] == '.gif':
            v = 0
            for i in os.popen('identify -format "%T\n" ' + file).read()[:-1].split('\n'):
                v <<= 1
                if i == '0':
                    break
                v |= ord(i[0]) - 49
            if v != 0:
                result += chr(v)

print result
#脚本by cchan
```