NJUST CTF 2018 Writeup

原创

frostwing98 于 2018-03-17 22:20:17 发布 1779 收藏

分类专栏： 网络攻防技术与实践 文章标签： 网络攻防

NJUST CTF 2018 Writeup

Supernova, Nanjing University frostwing98，Trap，Arcadia

MISC1 签到

by Arcadia

一路调戏ai之后（误）可以得到一串base64编码的字符串，在线解码得到flag

MISC2 GIF

by frostwing98

下载到gif，发现是不断闪烁的黑/白画面，第一考虑是莫尔斯电码。首先http://tu.sioe.cn/gj/fenjie/分解之。

发现间隔并不均匀，也不按照1：3：5的比例。因此考虑01串的可能性。

后来题目有调整，也分好了每一帧。

我们发现，总共有91个图片，如果是ASCII，应该是8的倍数才对。

我们应该知道，ASCII的可打印字符，高位始终不会大于7，也就是0111.因此最高位其实可以省略。为此我们又发现，91=7×13，进一步验证了这个猜想。

因此将13×7个bit，每7个一组，最高位加上0，就能够构成flag的13个字符。

MISC3 code

by Arcadia

纯编程题，利用在线文字识别工具获取图片中的每个数字，存放在文件中，用算法模拟贪吃蛇移动路线，用辗转相除法计算最大公约数跟最小公倍数，最后得到flag。

C++代码：

```
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>

using namespace std;

int gcd(int a,int b){ return a%b==0?b:gcd(b,a%b);}

int main()
{
    int x=0,y=0;
    int p,q;
    int sum=25*25-1,value=0,temp,ans;
    int state=1;
    ifstream input;
    int map[25][25];
    bool flag[25][25]={0};
    input.open("data.txt",ios::in);
    for (int i=0;i<25;i++)
        for (int j=0;j<25;j++)
            input>>map[i][j];
    input.close();
    p=0;q=1;
    ans=value=map[0][0];
    flag[0][0]=1;
    while (1)
    {

        if (x+p<0 || x+p>=25 || y+q<0 || y+q>=25 || flag[x+p][y+q]) state=(state+1)%4;
        switch(state)
        {
            case 0:p=-1; q=0; break;
            case 1:p=0; q=1; break;
            case 2:p=1; q=0; break;
            case 3:p=0; q=-1; break;
        }
        x+=p; y+=q;
        if (flag[x][y]) break;
        cout<<x<<':'<<y<<' ';

        if (value>=map[x][y])
        {
            temp=gcd(value,map[x][y]);
            ans-=temp;
        }
        else
        {
            temp=gcd(map[x][y],value);
            ans+=value/temp*map[x][y];
```

```
        }
        value=map[x][y];
        flag[x][y]=1;
        sum--;
    }
    cout<<ans;
    return 0;
}
```

运行截图：

Crypto1 Classical Enc

by frostwing98

题干：

Ld hsrakwusyaxr, y hoyjjlhyo hlaxes lj y kraewv hlaxes kxyk zyj tjei xljkwslhyoor gtk dwz xyj vyooed, vws kxe pwjk aysk,ldkw iljtje. Ld hwdksyjk kw pwiesd hsrakwusyaxlh youwslkxpj, pwjk hoyjjlhyohlaxesj hyd ge asyhklhyoor hwpatkei ydi jwofei gr xydi. Xwzefes,WFKT|vGkc `iP5kK2i5p` lnblVJ~ kxer yse yojw tjtyoor fesr jlpaoe kw gseym zlkxpwiesd kehxdwowur. Kxe kesp ldhotiej kxe jlpaoe jrjkepj tjei jldhe Useem ydiSwpyd klpej, kxe eoygwsyke Sedyljjydhe hlaxesj, Zwsoi Zys LL hsrakwusyaxr jthxyj kxe Edlupy pyhxlde ydi gerwdi. Y btlhm gswzd vwq ntpa wfes kxe oycr iwu.

这一道题的确挺“经典加密”的……

首先明眼人一看便知，这个y，多次出现在逗号之前，可能是“a”。于是考虑凯撒密码及其衍生移位密码的可能性。

失败……

后来hint1出现。迅速联想到词频分析。https://quipqiup.com/在这个网站在线进行解密。

阴险的是，其实最后一句话已经给出了答案。A quick brown fox jumpover the lazy dog.是一个著名的句子，因为它不重复的包含了26个字母，可以作为解码的密码表。

于是顺利解出flag所在的句子

OVTU|fBtz `dM5tT2d5m` DjqiFS~

因为|与～的间隔与{与}的间隔同为2，因此这里才是凯撒，offset为1.

decode结果为NUST{eAsy_cL4sS1c4l_CiphER

Crypto2 XOR

by Arcadia

观察代码，发现是进行了5次异或下一位的运算之后得到结果，通过xor的运算规则能推导出第一个字符是由第1,2,5,6个原字符异或而成，第二个是2,3,6,7……以此类推，因为已知flag格式，前5个字符为“NUST{”，故可通过递推得到后面的字符。

代码及运行结果：

RE1 Base64Encode

by Arcadia

这题修改了base64的编码表，并给出了经过编码后的flag值，利用base64编码的原理，将4位的base64编码通过编码表对应到3个字符上，即可获得flag。

首先可以在IDA中直接看到编码后的flag值：

然后在base64dncode函数中可以发现一个数组，存放着编码信息：

然后编个代码decode就搞定了：

RE2 Maze Game

by Trap

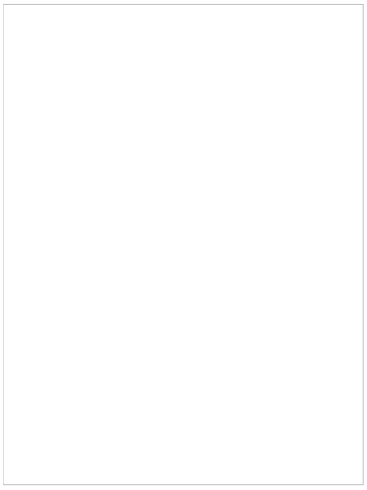
这道re比较简单，用IDA反编译很方便，既没有删节头又没有加壳，可以说是非常良心了。

这里要注意的只有三个地方，第一处是输入的字符串必须为22个（这也是让人容易掉进陷阱的地方，你构造的路径必须是正好22步），第二处是迷宫的验证，第三处是生成flag。

Maze_verify如下：

可以看出，就像走迷宫一样用wsad移动，只是这里的上下是相反的，可以先构造再替换字符。然后如果在某一步里面出了边界或者迷宫地图上不是1，也不算。看看迷宫的地图：

结合上面的v3 v2的判断条件，容易联想到是8*8的迷宫，大概长这个样：



Payload为wwdwdsssddddwwwaawwddw

Pwn1 babystack

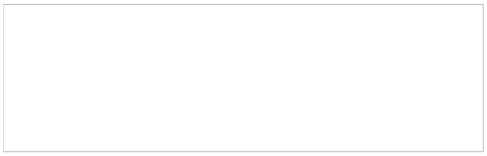
by Arcadia

利用栈可执行，将shellcode插在返回地址之后，返回到给出的栈地址，利用jmp指令跳转到shellcode所在地址执行，拿到权限，cat flag得到flag

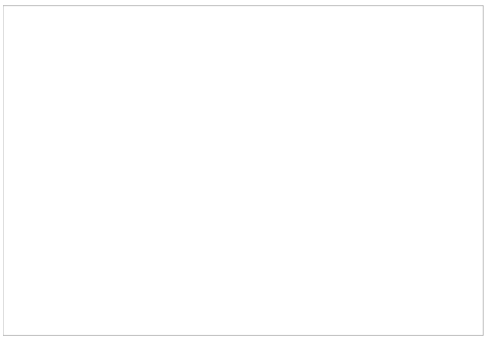
直接通过gets的函数漏洞修改返回地址为之前泄露的字符地址：

指向栈上的jmp汇编代码：

跳转执行shellcode：



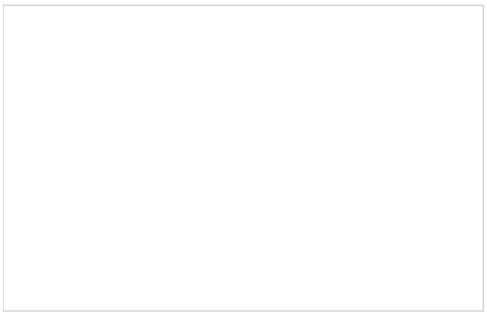
就可以进入服务器获取flag了：



Web1 PHP Audit

by Trap

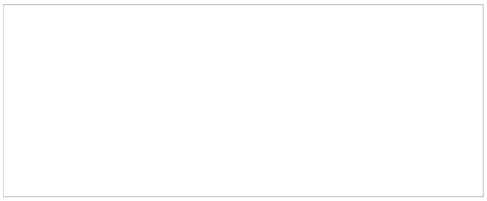
开始的状态是这样的：



这里提示我们输入两个参数key1和key2，其中，key1是一个文件名，将其打开后与“Hello hacker!”一致。容易想到PHP协议流data://text/plain;base64,SGVsbG8gaGFja2VyIQ==，这里的第二个参数是“Hello hacker!”经过base64编码后的结果，于是就得到第二部分：

这里要求一个经过MD5以后的字符串要大于66666*66666，这里的结果不能以字母开头，不能以0e和1e开头，我在MD5网上随便试了一些，得到一个字符串vwer ， 其32位MD5为4e32e71b2e1aaa761a3c27eff3f7c257，PHP会将其识别为4^32。

整理一下是这样的：



过k3需要用到intval的一个漏洞：他不能识别除了数字和字母e以外的字符，遇到了就会停止。（小数点、负号等除外）因此这里用十六进制编码666 =0x29a，可以让intval的输出不等于666，而在字符串与666比较时，会被看做16进制数转化成666.这样就来到了k4

经过与主办方交流，预期解似乎是665.999999 intval会将小数点截断，然后比较的时候会四舍五入(?)

接下来到了这道题最诡异的部分了。关于k4的取值，理论上只要传给intval的值超过临界值214743647，那么将会溢出，返回一个负数。这样我们可以构造一个k4=2147483647的payload。这个payload在我的机子上测试通过(php version 5.6.25, 服务器version 5.6.28)但就是拿不到flag。然后，如果你在后面疯狂地加0，你会发现仍然没有flag……

经过我的不严谨的测试，我发现，服务器端是否溢出只跟位数有关而与大小无关。最后我们构造了一个21位的payload 214748364900000000000 拿到的flag。但经过我的测试，发现只要是21位，不管是什么数都可以拿到flag。但往下减位数，比如20 19 18 17位都不行。而往上找呢？只有24 36位可以用。各位有兴趣可以去尝试一下服务器的这个神奇的设定。

最终payload：