1.16集训总结

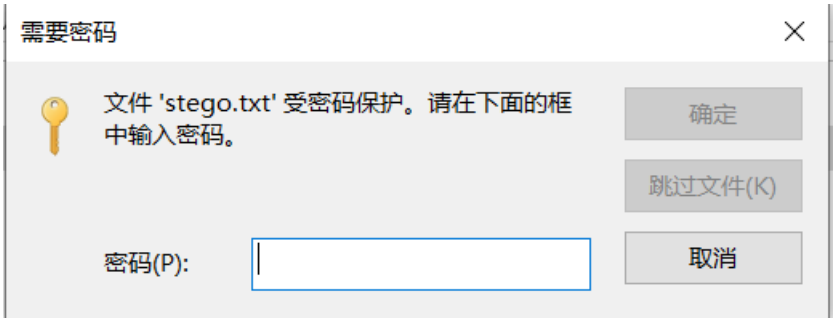
原创

keai1025 于 2020-01-17 08:19:02 发布 68 收藏
版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/keai1025/article/details/103998770
版权

攻防世界

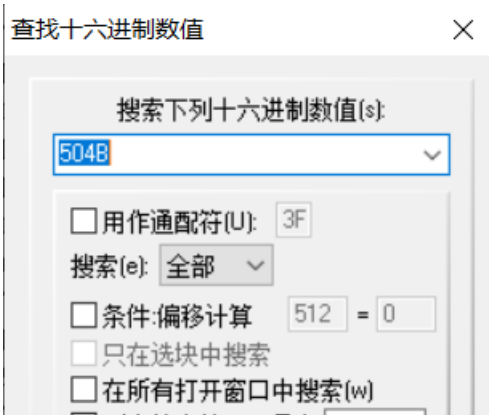
一、 base64stego小白题解

这是昨晚做的一个题，emmm...在我这样的小菜鸟看来，还是好难的。
首先，文件打不开，显示需要密码，第一次遇到这样的题，憨憨的我还真的有理有据的猜了两个密码（base64和rot13），没猜对，我就找了writeup，才知道这是**伪加密**



writeup上面说可以用啥啥软件破解，但是我莫得那个软件，也懒得下载，就用了另一种更粗暴的方式
找到一个介绍，可以晓得如何分辨加密伪加密介绍
1.用WinHex（就是一个很有用的软件，做题时经常压缩包损坏啊，就用这个修复）打开这个压缩包，然后搜索**504B**（为啥么要搜索504B呢，上面那个介绍里有的）
如何搜索呢
（1）就是我圈起来的这两个望远镜啦，左边是搜索文本，右面是搜索16进制数的，这里用右面的

（2）如图所示，>>确定（记得，给列出搜索结果打√！）



然后就找到三处，发现果然是伪加密，把框起来的那个09改为00，保存，发现文件可以打开啦

```
 0  1  2  3  4  5  6  7
50 4B 03 04 14 03 00 00
```

```
 0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
50 4B 01 02 3F 03 14 03  09 00 08 00 68 BF 9B 48
FE 32 7D 4B E9 0D 00 00  B5 1B 00 00 09 00 24 00
00 00 00 00 00 00 20 80  ED 81 00 00 00 00 73 74
65 67 6F 2E 74 78 74 0A  00 20 00 00 00 00 00 01
00 18 00 80 0B 49 BF 9D  A0 D1 01 80 A7 42 38 B7
2F D4 01 00 11 AA 37 B7  2F D4 01 50 4B 05 06 00
00 00 00 01 00 01 00 5B  00 00 00 10 0E 00 00 00
```

2.打开发现是好长好长的字符串，厉害的人可能一下子就知道解题方法了，我只能用base64解密看看（至于为什么我会想用base64解密，我忘记了，可能感觉比较像叭，还有题目说如来十三掌最后一步嘛，那不就是base64解密的）

Base64 编码或解码的结果：

Steganography is the art and science of writing hidden messages in such a way that no one

解码出来后是这样一句话，翻译后就是：隐写术是一门艺术和科学，它以一种无人能及的方式书写隐藏的信息。
然后百度base64隐写，就有好多文章啦，不过我还是懵懵懂懂，这里附上base64隐写介绍及解题方法
复制了大神的python脚本运行，就得到了结果（框里改为自己的文件名字，还要把文件放在python的那个文件夹里才行）

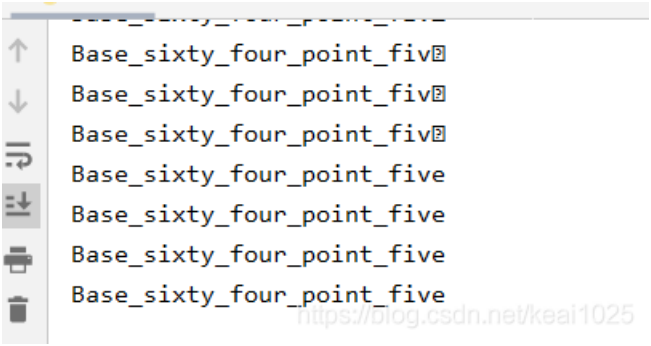
```python
# -*- coding: cp936 -*-
import base64
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = str(line, "utf-8").strip("\n")
        rowb64 =  str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\
        offset = abs(b64chars.index(stegb64.replace('=','')[-1])-b64chars.index(rowb
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
        print(''.join([chr(int(bin_str[i:i + 8], 2)) for i in range(0, len(bin_str),
```

https://blog.csdn.net/keai1025

这里附上解密脚本嗷

```
# -*- coding: cp936 -*-
import base64
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
with open('stego.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = str(line, "utf-8").strip("\n")
        rowb64 =  str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\n")
        offset = abs(b64chars.index(stegb64.replace('=','')[-1])-b64chars.index(rowb64.replace('=','')[-1]))
        equalnum = stegb64.count('=') #no equalnum no offset
        if equalnum:
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
        print(''.join([chr(int(bin_str[i:i + 8], 2)) for i in range(0, len(bin_str), 8)]))
```

这就是运行结果啦，也就是答案



到这里我觉得我真的真的是憨憨，我以为这个不是答案，因为之前做的题，都是flag{}这种的，然后这个题他没有，呜呜，我就去看题解，发现他们就说到这里，放了张截图，我就想是不是这就是答案，复制上去，不对！！直到看了一个超级贴心的大哥把flag放了出来，我才知道这个是给这个结果加一个flag就可以，也就是**flag{Base_sixty_four_point_five}**，我竟然因为最后这里多浪费一个多小时！！还是太年轻，莫得经验

二、ext3

这是大神写的writeup

题目上说是一个Linux系统光盘，把文件拖到我的虚拟机里，emm…为了方便，我把下载的附件名字，改为1，然后我发现了一个神奇的事，这个东东它没有扩！展！名！（与题目无关，纯粹是我觉得神奇）

用vim打开之后是这样的

看不懂欸，题目上也没有啥么线索了，这时候就要用到一个万能的解题方法了，看writeup…

（大神说下载附件知道这是一个二进制文件，但是怎么看出是二进制文件呢，我也不知道，对解题没有影响不管他）

用string命令来搜索flag相关信息看看（strings 文件名|grep 想要搜的关键字）这里就搜strings 1|grep flag

```
root@kali:~/keai# strings 1|grep flag
.flag.txt.swp
flag.txtt.swx
~root/Desktop/file/O7avZhikgKgbF/flag.txt
.flag.txt.swp
flag.txtt.swx
.flag.txt.swp
flag.txtt.swx
```

据经验，flag的信息很有可能就在flag.txt里面，这时候就要用到分离文件的命令啦

（binwalk -e 文件名）分离后，打开flag.txt文件，是一串加密的字符

ZmxhZ3tzYWpiY2lienNrampjbmJoc2J2Y2pianN6Y3N6Ymt6an0=，用base64解密就得到flag啦

（emmm…我发现后面有=的，用base64解密就没错，还有熟知Linux的命令真的很重要）

今天还学到了python的新知识

try：写程序不确定是否正确可以用，如果正确，运行try，如果不正确，就会根据except后面的错误类型运行后面的程序

(Exception包含了全部的错误类型）

```
a = 'xiaoding'
try:
    print(a[10])
except Exception:
    print('F')
```

```
11
G:\python\venv\Scripts\python.exe G:/python/11.py
F

Process finished with exit code 0
```

https://blog.csdn.net/keai1025