

writeup 2019“新华三杯”中国医疗机构网络安全攻防演练大赛 CTF（复赛）

原创

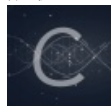
ggzhifeng 于 2019-11-05 21:20:55 发布 2582 收藏 8

分类专栏: [ctf](#) 文章标签: [ctf](#) [writeup](#) [新华三](#)

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/ggzhifeng/article/details/102924798>

版权



[ctf](#) 专栏收录该内容

1 篇文章 0 订阅

订阅专栏

2019年10月31日, 由《中国数字医学》杂志社主办、紫光旗下新华三集团支持的2019“新华三杯”中国医疗机构网络安全攻防演练大赛(复赛), 在武汉光谷科技会展中心火热开赛。来自全国30多个省市的150家医疗机构, 518人参加了本次复赛, 这也是迄今为止中国境内规模最大的医疗行业攻防大赛。

复赛共2小时, 分为理论题和CTF题, 同时进行, CTF共10到题, 并且要在2小时内提交writeup...以下是部分题目的writeup, 少了一题web综合题, 当时没时间去看, 现在也没环境了...

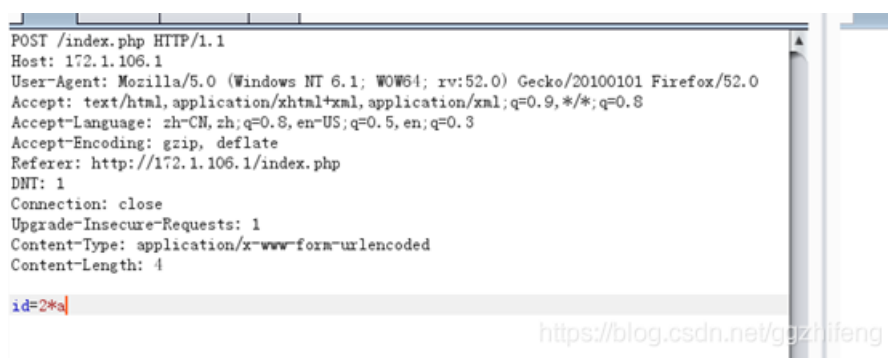
1. 签到题

题目中已显示 flag, 直接填入正确答案。



2. SQL注入

注入点在id



直接使用sqlmap跑, 可以跑出flag

```
[14:18:43] [INFO] Table 'ctftraining.news' dumped to CSV file 'C:\Users\Administrator\AppData\Local\sqlmap\output\172.1.106.1\dump\ctftraining\news.csv'
[14:18:43] [INFO] fetching entries of column(s) 'flag' for table 'flag' in database 'ctftraining'
[14:18:43] [INFO] used SQL query returns 1 entry
[14:18:43] [INFO] fetching number of column(s) 'flag' entries for table 'flag' in database 'ctftraining'
[14:18:43] [INFO] retrieved: 1
[14:18:49] [WARNING] (case) time-based comparison requires reset of statistical model, please wait..... (done)
[14:18:54] [INFO] adjusting time delay to 1 second due to good response times
flag(a0887e70ab)
Database: ctftraining
Table: flag
[1 entry]
-----+
| flag |
-----+
| flag(a0887e70ab) |
-----+
https://blog.csdn.net/ggzhifeng
```

3. 图片隐写

Binwalk发现图片里面藏着其他图片

```
C:\Users\Administrator\Desktop\新建文件夹 (2)\图片隐写_05\binwalk flag10.jpg -e
* suggest: you'd better to input the parameters enclosed in double quotes.
* made by pcat

WARNING: The Python LZMA module could not be found. It is *strongly* recommended that you
install this module for binwalk to provide proper LZMA identification and extraction results.

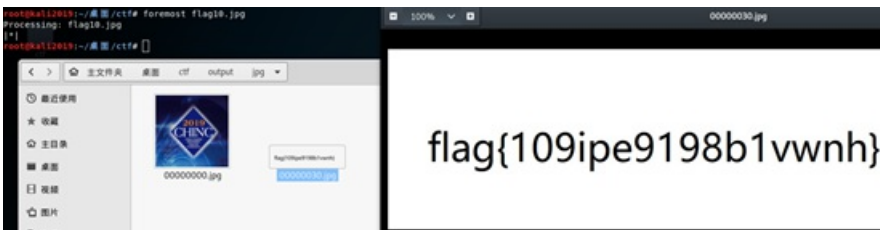
DECIMAL      HEXADECIMAL  DESCRIPTION
-----+-----+-----
0            0x0          JPEG image data, JFIF standard 1.01
15708       0x3D5C       JPEG image data, JFIF standard 1.01
15738       0x3D7A       TIFF image data, big-endian, offset of first image directory
: 8
https://blog.csdn.net/ggzhifeng
```

无法binwalk直接解压，使用winhex，查找jpeg的文件头，从FFD8开头，截取后面的文件，导出图片，成功发现flag



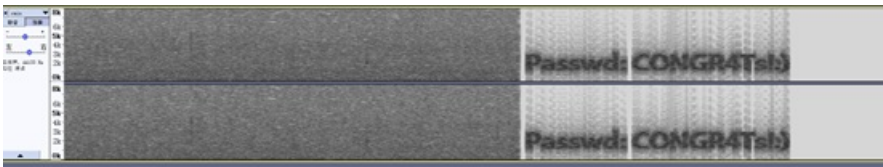
后来听说直接foremost就可以解出来了....果然，好香...

Foremost flag10.jpg



4. 音频隐写

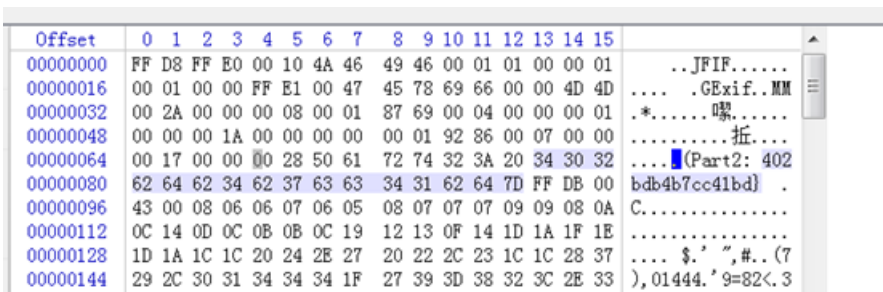
使用音频软件打开，调出频谱图



发现压缩包密码，打开压缩包，获取到part1 flag



使用winhex打开图片，发现flag2



成功获取整个flag

flag{1a6b4d4e0c36581e8256a1a58402bdb4b7cc41bd}

5. 本地文件包含

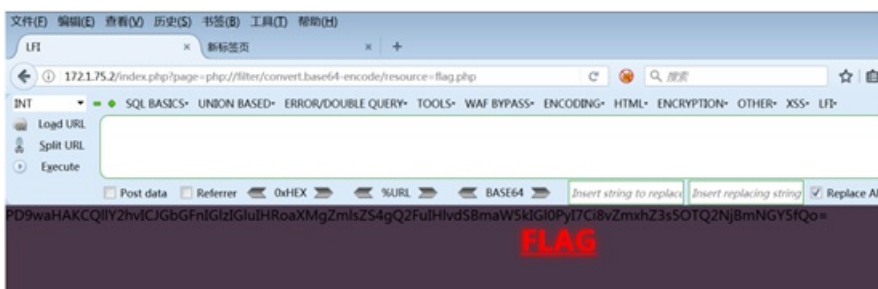
通过目录扫描，发现phpinfo页面，发现

allow_url_fopen on

allow_url_include off

直接包含flag.php发现直接运行了脚本，没有flag，猜测flag藏在注释中，需要读取到php源码才可以，所以只能使用伪协议命令，输出 flag.php 的 base64 编码

?page=php://filter/convert.base64-encode/resource=flag.php

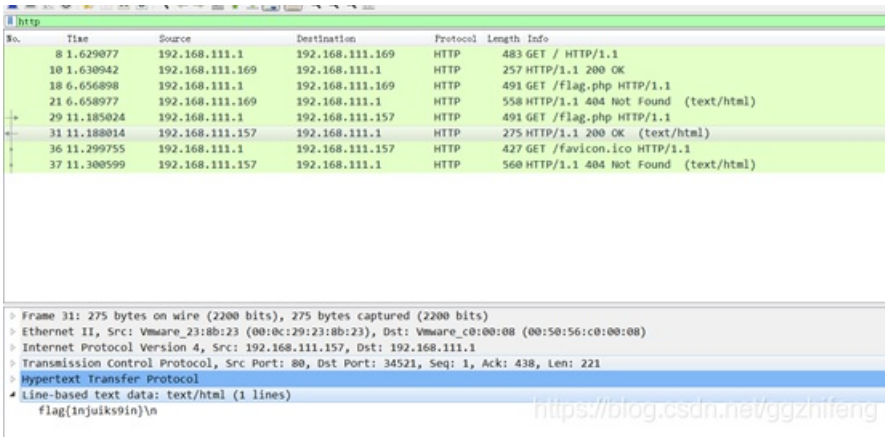


Base64解码发现flag藏在注释中



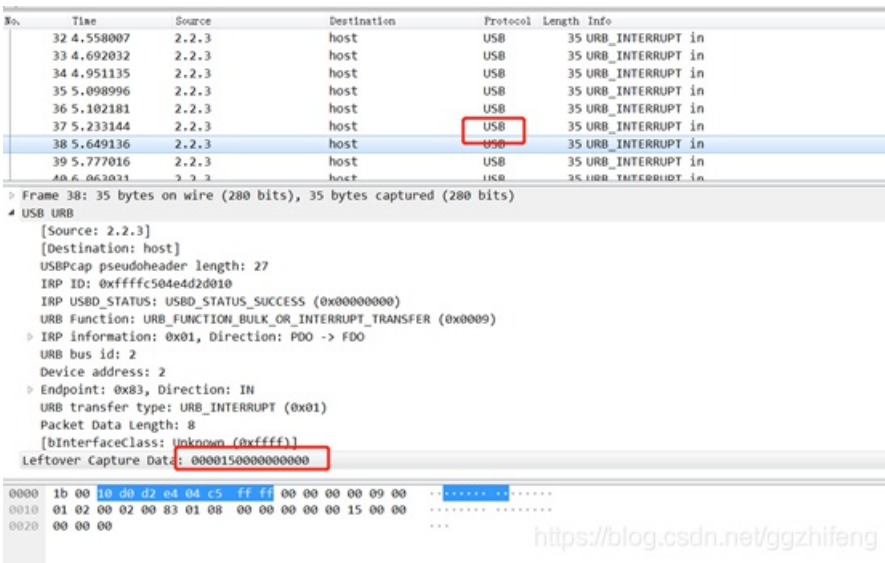
6. 数据包分析1

过滤http数据，直接发现flag



7. 数据包分析2

下载文件回来，发现一个压缩文档，需要密码，不是伪加密，另外有个pcap流量文件，打开发现是usb流量，USB协议数据部分在Leftover Capture Data域中，键盘数据包的数据长度为八个字节。其中键盘击键信息集中在第三个字节中。数据如下图所示：。



具体的键位映射关系可参考：《USB键盘协议中键码》中的HID Usage ID，链接：https://www.shsu.edu/csc_tjm/fall2000/cs272/scan_codes.html

所以压缩文档的密码在这个流量包的usb键盘流量中

1. 使用kali linux中的tshark 命令把cap data提取出来：

```
tshark -r usb.pcap -T fields -e usb.capdata > usbdata.txt
```

1. 根据《USB键盘协议中键码》中的HID Usage ID将数据还原成键位，github上有脚本，但不知道为什么，一直没跑出来....

<https://github.com/WangYihang/UsbKeyboardDataHacker>

<https://ctf-wiki.github.io/ctf-wiki/misc/traffic/protocols/USB-zh/>

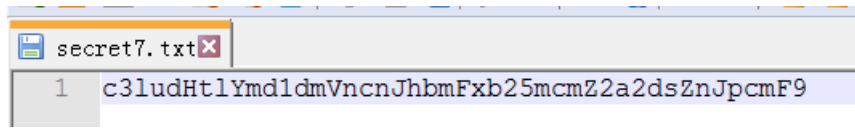
[跑出来也密码不正确](#)

```
tthiisimmypasswordl
'D:\3.Coding\python\test>python keyboardstest.py password.txt
```

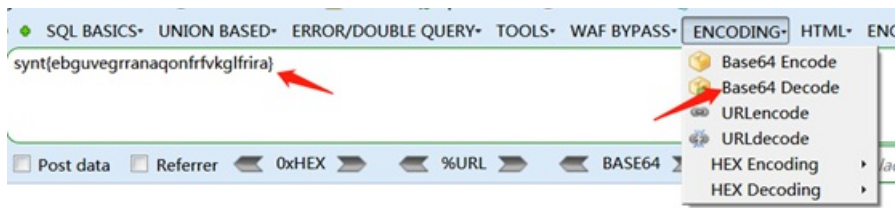
8.EasyCrypto

源码

```
c3ludHtlYmd1dmVncnJhbmFxb25mcmZ2a2dsZnJpcmF9
```



经过Base64解码后如下：



```
synt{ebguvegrranaqonfrfvkglfira}
```

发现是flag格式，说明字母进行了转换，与flag{xxx}对比，发现是字母assic码往后减13，直接转了下，发现存在特殊字符，提交flag提示错误，猜想如果减完后的值小于a的assic码97，则加上26，使其还原成字母范围，在上面字符串去掉{}后转为assic码，写了个python脚本，成功跑出flag（因为现场没有网络，忘记python中string和assic码如何转换了，所以手工使用工具进行转换然后再python脚本简单的加减和判断）。

```
temp1=[115,121,110,116,101,98,103,117,118,101,103,114,114,97,110,97,113,111,110,102,114,102,111,107,103,108,102,114,105,114,97]
for i in range(len(temp1)):
    temp2=temp1[i]-13
    if temp2<97:
        temp2=temp2+26
    print temp2
```

现在查了下，发现很简单，number = ord(char), char = chr(number)，最后的脚本如下

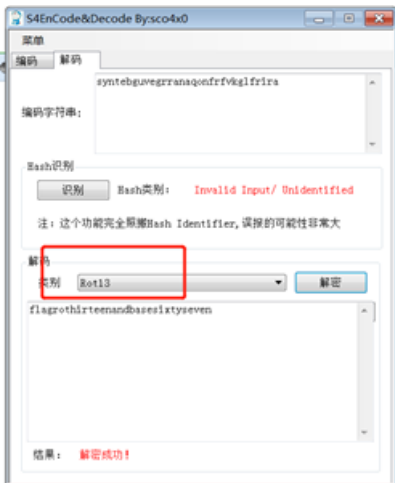
```
1  temp1="syntebguvegrranagofrfvkglfrrira"
2  flag=""
3  for i in range(len(temp1)):
4      temp2=ord(temp1[i])-13
5      if temp2<97:
6          temp2=temp2+26
7      flag=flag+chr(temp2)
8  print flag

for i in range(len(temp1))
Run: test x
D:\program\Python27\python.exe D:/3.Coding/python/test/php/test.py
flagrothirteenandbasesixtyseven
Process finished with exit code 0
https://blog.csdn.net/ggzhibfeng
```

前后添加{}就是最终的flag

Flag{rothirteenandbasesixtyseven}

后来听说是rot13加密方式，直接使用工具就可以解开了.....



9. Easy RSA

当时不懂RSA并且没有网络无法安装相应的Crypto库，运行不了脚本，所以没有做

打开压缩包，发现存在3个文件，通过看py脚本，发现cipher.txt是flag加密后的密文，pubkey.txt是RSA参数中的e和N

.. (上级目录)	
cipher.txt	1 KB
encrypt.py	1 KB
pubkey.txt	1 KB

所以情况很明了，分解N，算出p和q，然后计算d，就可以解析出明文了（当然也可以直接运行脚本，穷举p和q，然后判断是否得出的N等于原始给的N，这样也可以获取P和Q）

但是发现N特别的大，

如果n小于256bit，可以使用本地工具进行暴力分解，例如windwods平台的RSATool，可以在数分钟之内完成256bit的n的分解。

如果n大于768bit，可以尝试利用在线网站http://factordb.com，这一类在线网站的原理是储存了部分n分解成功的值。

这里的N有617位，又没有网络，所以最终比赛结束，都没有人算出来.....

337741676001996910724704248988429281685705599403627707860606993209895468516951064669241638 Factorize! (?)

Result:		
status (?)	digits	number
FF	617 (show)	3377416760_89<617> = 1779934618_83<309> * 1897494843_83<309>

E=65537

N=33774167600199691072470424898842928168570559940362770786060699320989546851695106466924

P=17799346181607540824086675222721031931682557429100037672752399131508609760506383756334

Q=18974948436644986163073648262203020422960007493673339722966873858660589597981182399402

使用RSATool，得到D

D=11264411788839355592444856301614488363956471904061056255881635805090094375457400203763

理论上应该是这样就能够接触来了，可是不知道为什么不对.....

1st Prime
198316766645675423523317234436261068493854277438996689406667510384024633202469661
72505094817799671009070311486253645420435061175078660441183

2nd Prime
94836992026767113435286273497842956368997116036170165393912022560935791934662695453
87084602431291560404860521941014042046916379779129644454583

Modulus (N) [R]
Exact size: 0 Bits
337741676001996910724704248988429281685705599403627707860606993209895468516951064669241638

Private Exponent
11264411788839355592444856301614488363956471904061056255881635805090094375457400203763

Factoring info (Prime)
PRIME FACTOR: 13574881
PRIME FACTOR: 23781539

Decryption M=C*D MOD N failed! Check for M < N

附件:

本次比赛能下载的题目，已上传百度云

链接: https://pan.baidu.com/s/14SR1m-vKEUaO_cMDjnQsUw

提取码: 9fbv