2021年鹤城杯 Writeup

原创

小蓝同学` 已于 2022-04-10 00:53:27 修改 491 收藏 1

分类专栏： 信息安全漏洞 文章标签： CTF WP 鹤城杯

于 2022-04-09 19:56:09 首次发布

鹤城杯 Writeup

MISC

A_MISC

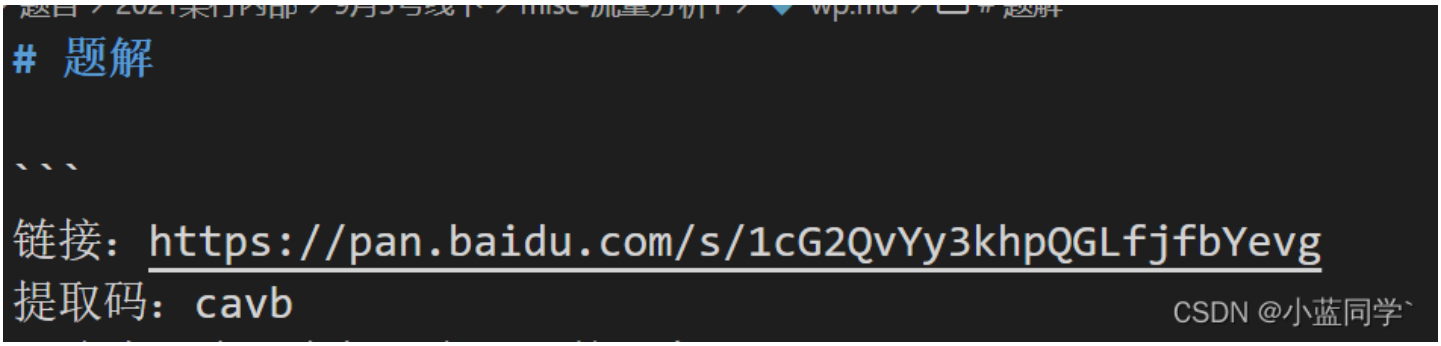
考点：

压缩包加密

基于时间盲注的flag取证

首先拿到一个压缩包，发现无法将其解压出来，使用ARCHPR进行密码爆破，这里爆破一般有几种爆破方式，1.全数字爆破 一般是4到6位 2.全英文爆破大写或小写 4到6位。

这里是全英文的小写的爆破，爆破压缩包密码得到qwer。解压之后发现有一张图片，使用Tweakpng打开图片发现报错，使用脚本修改宽和高，得到完整的图片。

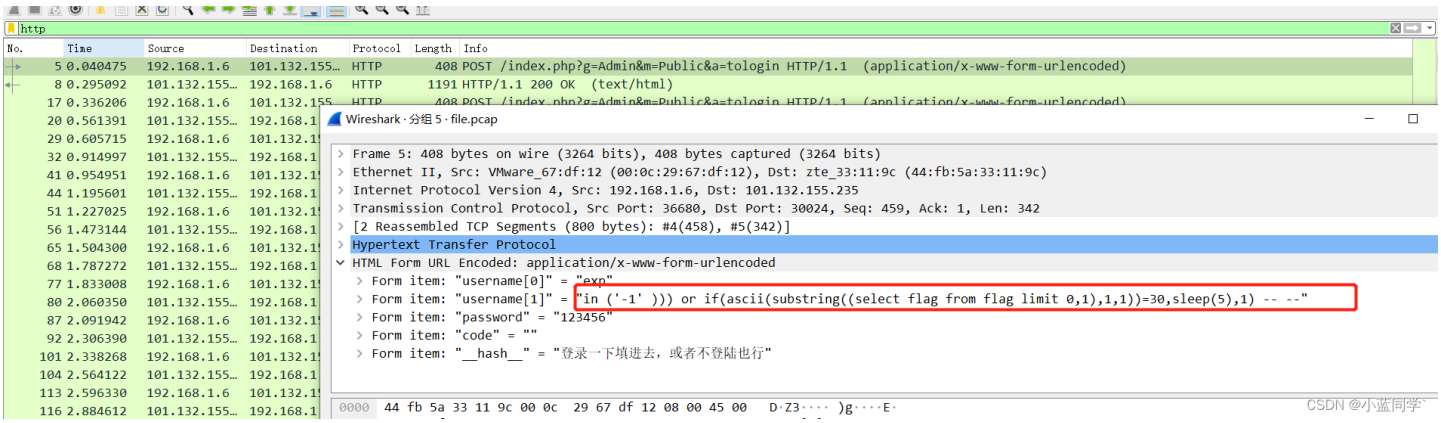


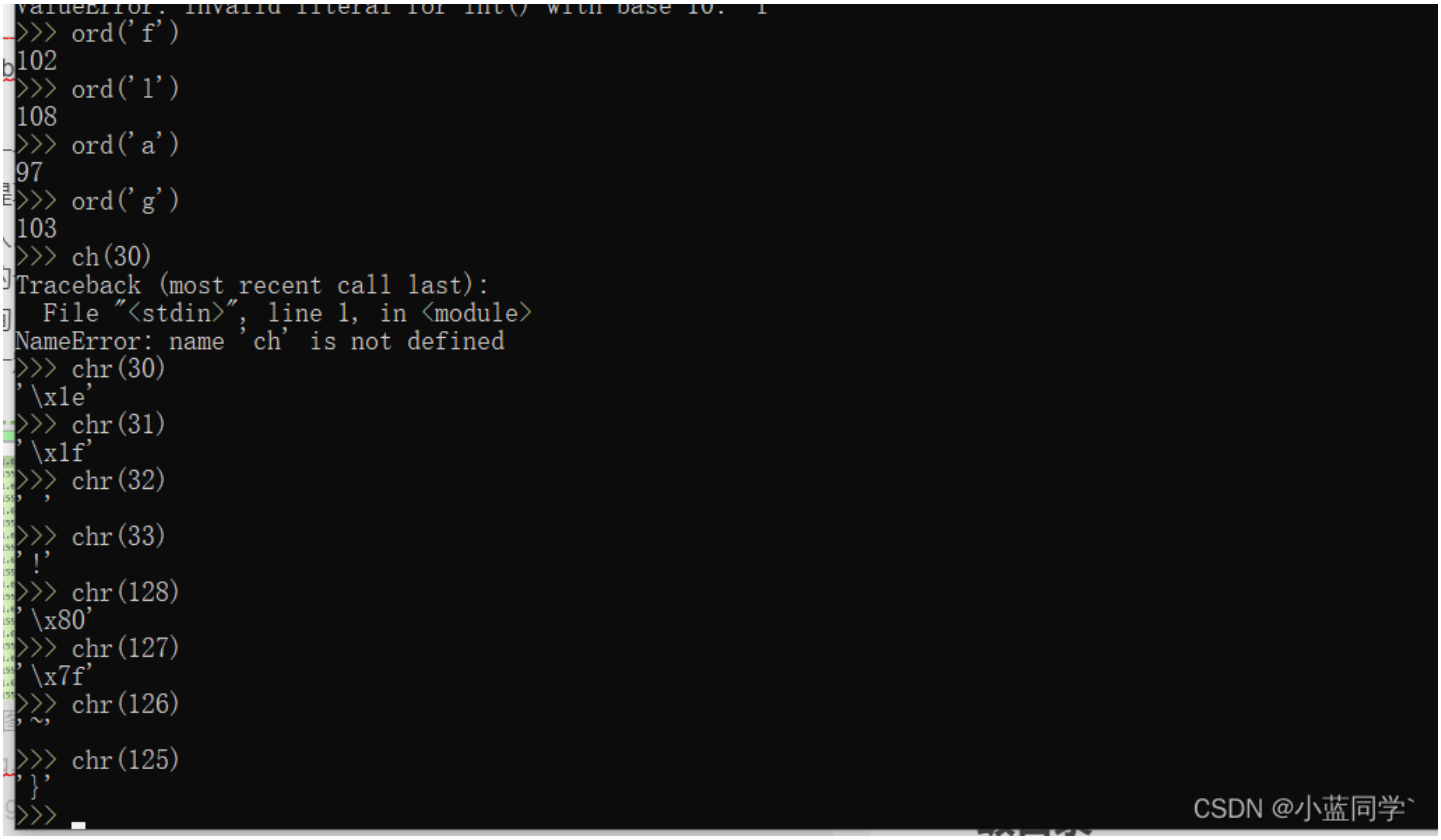
链接：https://pan.baidu.com/s/1cG2QvYy3khpQGLfjfbYevg

提取码：cavb

分析：

提取下来是一个流量包，过滤查看http包，发现是基于时间的盲注，那解法就是要让我们根据流量提取出正确的flag。这个与布尔盲注不一样，不能根据包的长度来判断是否为正确的注入。得通过判断比较得ASCII的值的变化来进行判断，因为盲注一般使用的是基于二分的一个思想进行查询，那么如果本轮的查询已经查到正确的字符就会进行下一轮的查询，而我们注入的范围一般是ASCII值为32 -128之间的字符，注入成功之后查询的值一定是查询32，者这里的查询是从30开始查询。





所以我们讲所有的查询流量解析为纯文本然后使用正则匹配匹配出对应的字符的ACSII，然后再转为字符即可。
脚本：

```python
import re

#username[1]" = "in ('-1' ))) or if(ascii(substring((select flag from flag limit 0,1),1,1))=30,sleep(5),1)
asciis = []
asciis1 = []
obj = re.compile(r"limit 0,1\),(?P<num>.*),1\)\)=(?P<asciinum>.*)\b,sleep",re.S)
#匹配正则表达式  第一个匹配lit 但是我们需要的只有ascii值  这里用.*贪婪模式  否则匹配失败


def getASCII():
    with open("./test/111.txt","r",encoding="utf-8") as f:
        for line in f.readlines():  #readlines()读取出来为列表所以我们直接迭代 用strip()去除空格
            line = line.strip()
            result = obj.finditer(line) #正则匹配
            for it in result:
                num = it.group("num")  #lit值
                asciinum = it.group("asciinum") #ASCIi值 注意这里提取的都为str类型  所以以下都要做int强转
                asciis.append(asciinum) #将payload的全部ascii值(包括失败的）写入列表

#使用一个循环 时间盲注可知ascii正确后才会进行下一次匹配  所以我们这里直接一个一个循环
# 然后当正确的如102时  下一个又从30开始  所以会比前面小  利用这个特点筛选将正确的ASCIi写入新的列表
def getASCII2():
    a = 1
    while a < len(asciis):
        if int(asciis[a]) < int(asciis[a-1]):
            asciis1.append(int(asciis[a-1]))
        a += 1


def getFLag():  #转换ASCIi
    flag = ""
    for i in asciis1:
        i = chr(i)
        flag = flag + i
        print(flag)

if __name__ == '__main__':
    getASCII()
    getASCII2()
    getFLag()
```

结果:

```
flag{cd2c3e2fea463d
flag{cd2c3e2fea463de
flag{cd2c3e2fea463ded
flag{cd2c3e2fea463ded9
flag{cd2c3e2fea463ded9a
flag{cd2c3e2fea463ded9af
flag{cd2c3e2fea463ded9af8
flag{cd2c3e2fea463ded9af80
flag{cd2c3e2fea463ded9af800
flag{cd2c3e2fea463ded9af800d
flag{cd2c3e2fea463ded9af800d7
flag{cd2c3e2fea463ded9af800d71
flag{cd2c3e2fea463ded9af800d715
flag{cd2c3e2fea463ded9af800d7155
flag{cd2c3e2fea463ded9af800d7155b
flag{cd2c3e2fea463ded9af800d7155be
flag{cd2c3e2fea463ded9af800d7155be7
flag{cd2c3e2fea463ded9af800d7155be7a
flag{cd2c3e2fea463ded9af800d7155be7aq
flag{cd2c3e2fea463ded9af800d7155be7aq}
```


`flag{cd2c3e2fea463ded9af800d7155be7aq}`

或者使用Tshark,直接导入一个文本文件，然后使用脚本直接打。

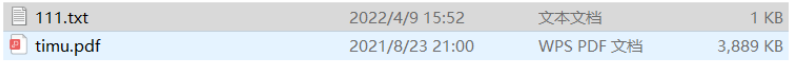
```
tshark -r file.pcap -T fields -e urlencoded-form.value | findstr flag > data.txt
```

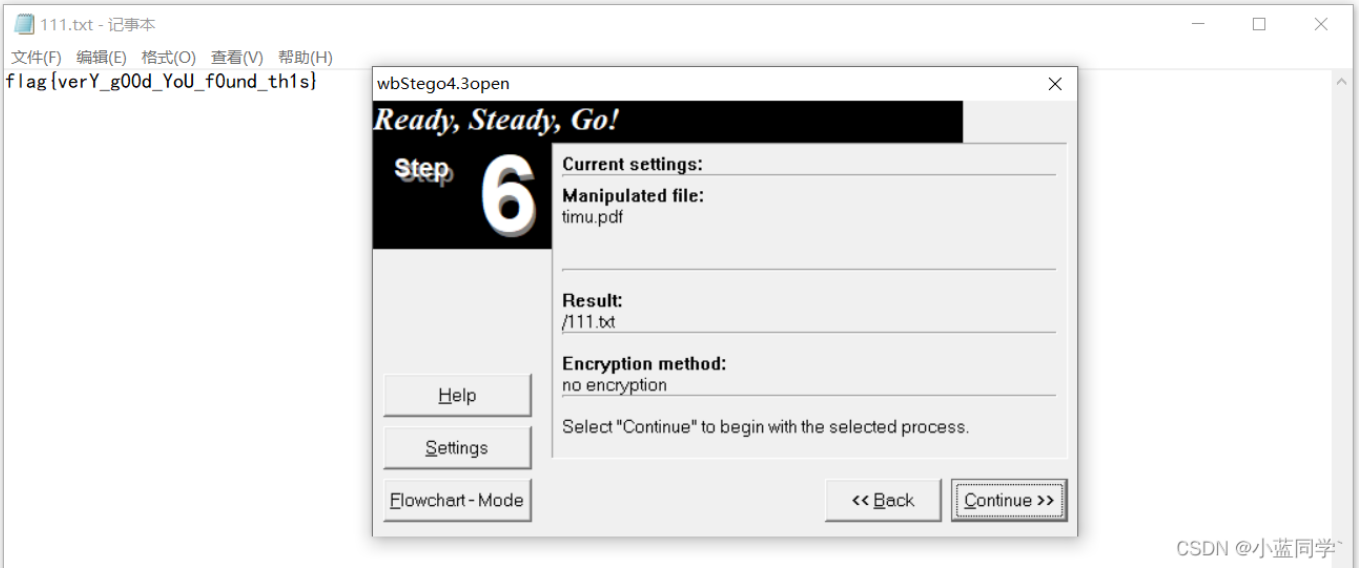
脚本：

```
with open('data.txt', 'r', encoding='UTF-8') as f:
    data = f.read().strip().split('\n')
flag = [0 for i in range(50)]
for i in data:
    flag[int(i[73:i.find(',', 73)])] = int(i[i.find('=') + 1:i.find(',sleep')])
print(bytes(flag).decode())
```

NEW_MISC

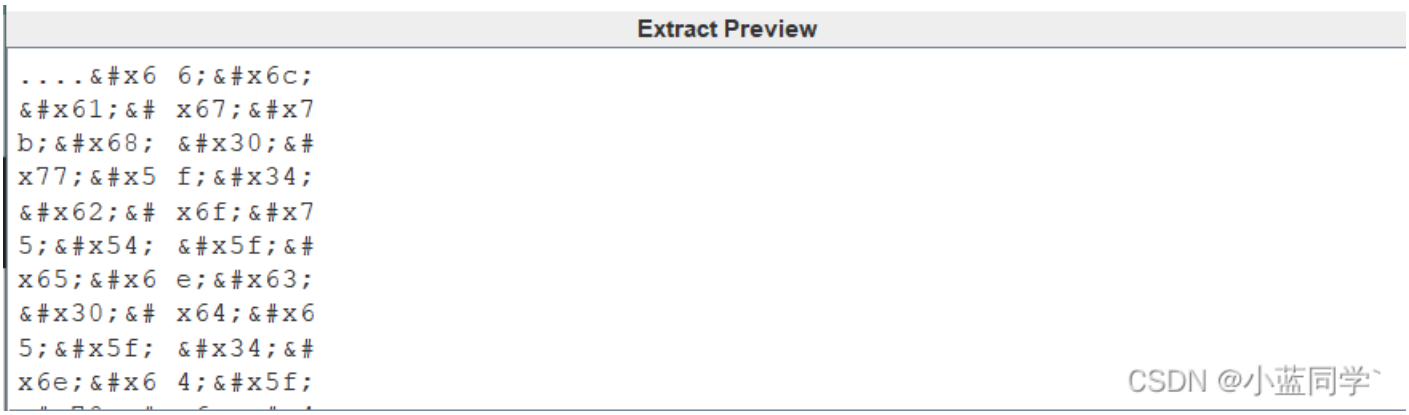
给了一个PDF直接WebStego隐写，拿flag，还是无密钥的。





MISC2

LSB，直接工具出来。

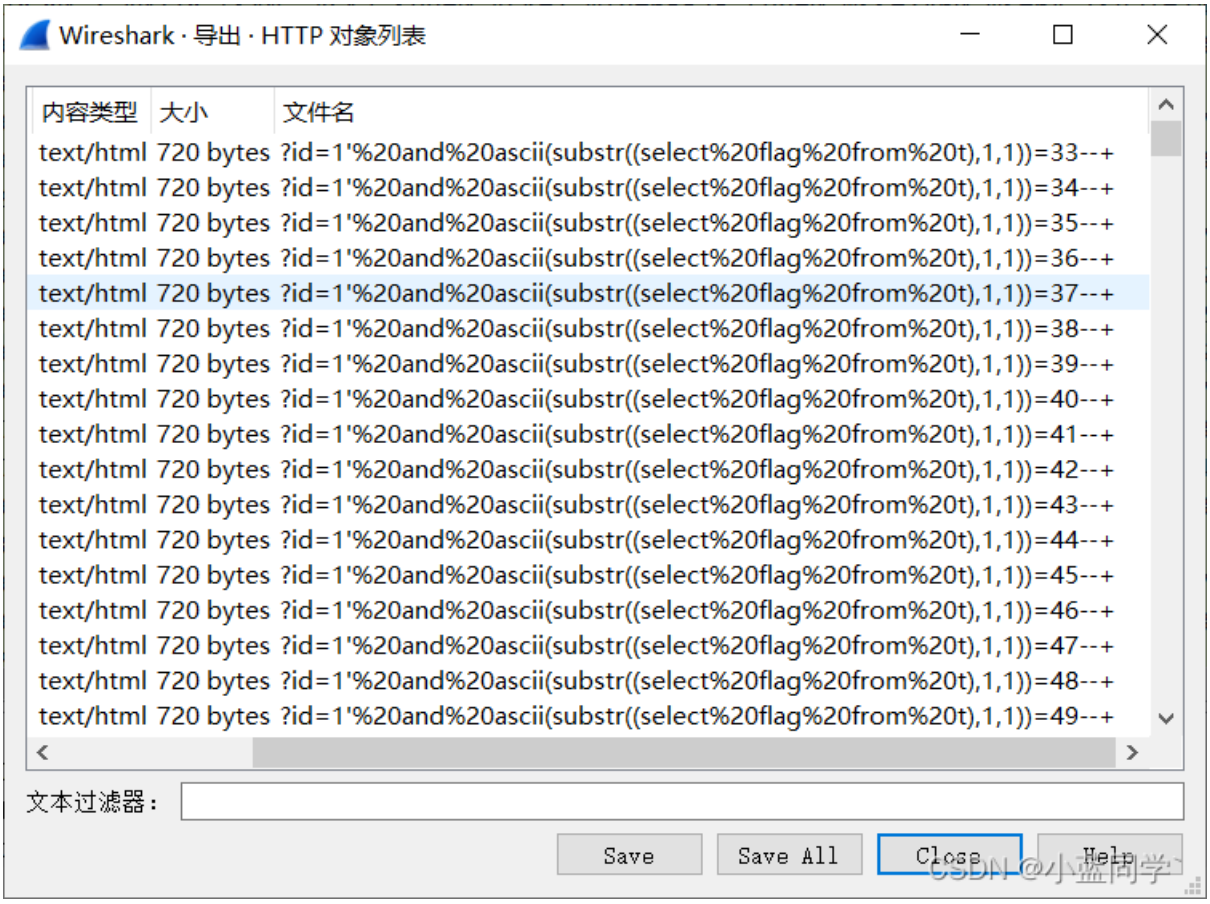


html 实体解码

`flag{h0w_4bouT_enc0de_4nd_pnG}`

流量分析

http对象，好明显的注入流量。



和上面那个差不多直接，用Tshark读取即可。

`http and frame.len==978`，直接完全地把成功注入的语句全部拿出来。

```
import re
s = r"from%20t\),([0-9]*),1\)\)=([0-9]*)"
pat = re.compile(s)
f = open("timu.pcapng","rb")
st = f.read().decode("utf-8","ignore")
lis = pat.findall(st)
flag = ['' for i in range(1000)]
for t in lis:
    flag[int(t[0])] = chr(int(t[1]))


for i in flag:
    print(i,end="")
```

```
s\好的先生\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\
achments_r2I3t6a\exp.py'
flag{w1reshARK_ez_1sntit}~~~~<
```

`flag{w1reshARK_ez_1sntit}`

MI—量子加密

另外一种ZIP爆破

> 以往进行ZIP已知明文攻击，通常需要一个完整的明文文件。 而本文讨论的攻击方式只需要知道加密压缩包内容的12个字节，即可进行攻击破解降低了已知明文的攻击难度。同时，结合各类已知的文件格式，更扩宽了ZIP已知明文攻击的攻击面。

需要使用的工具：bkcrack.

安装过程

> 先下载工具
> cmake 安装：cmake 安装/更新
> cmake .
> make //在src下生成bkcrack文件
> cp bkcrack /usr/sbin/bkcrack //作为系统命令使用

注意：如果安装过程中出现Openssl未安装，使用指令自动安装

```
sudo apt install libssl-dev libcurl4-openssl-dev
```