BugkuCTF(old)----流量分析题目Writeup

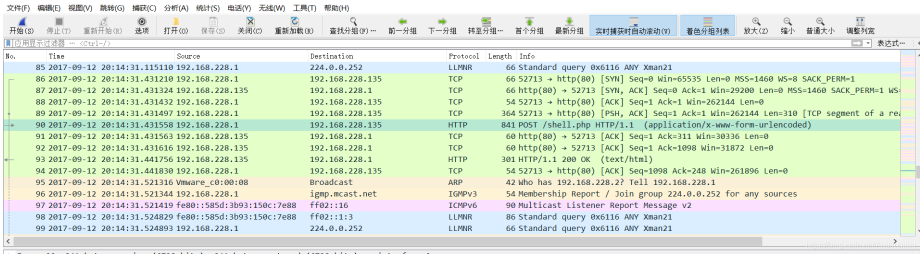
原创

Au.J 于 2019-09-26 17:51:40 发布 1002 收藏 4
分类专栏： CTF

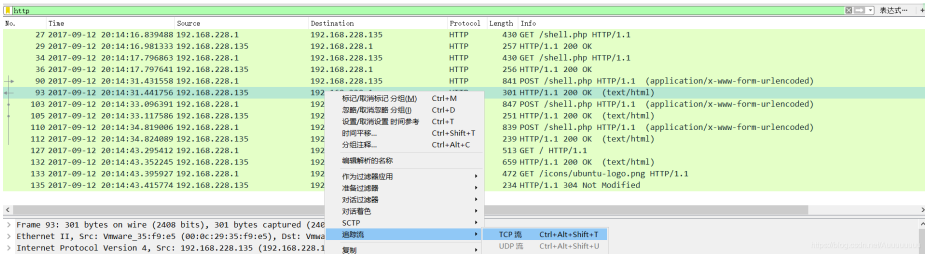
本文链接：https://blog.csdn.net/Auuuuuuuu/article/details/101366456

flag被盗

文件不是很大，粗略看了一下，发现了shell.php字段



筛选为http流量，追踪TCP查看

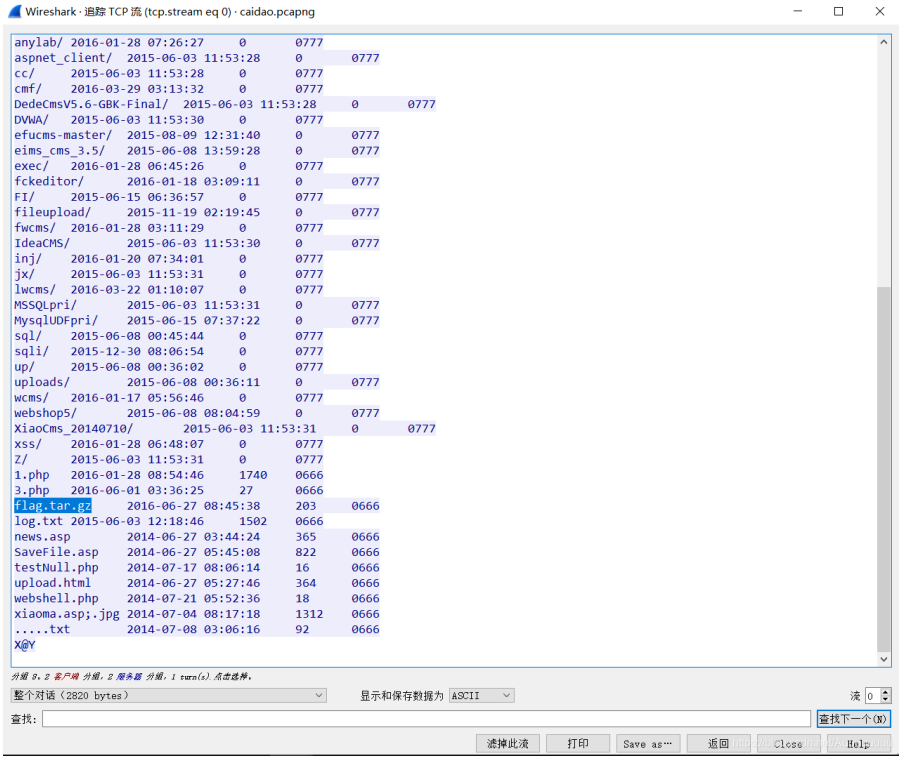


直接得到flag **flag{This_is_a_f10g}**

中国菜刀

下载解压得到数据包，比较小只有7kb

只有TCP和HTTP协议流量，追踪TCP流查看，发现flag.tar.gz压缩包



使用kali集成的 binwalk提取压缩包，

```
binwalk -e caidao.pcapng 文件路径
```

```
root@kali:~# binwalk -e caidao.pcapng

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
7747          0x1E43          gzip compressed data, from Unix, last modified: 2016-06-27 08:44:39
```

得到 1E43文件

```
root@kali:~/_caidao.pcapng.extracted# ls
1E43
```

这是一个压缩包，可以在windows修改后缀为zip直接打开

得到flag **key{8769fe393f2b998fa6a11afe2bfcd65e}**

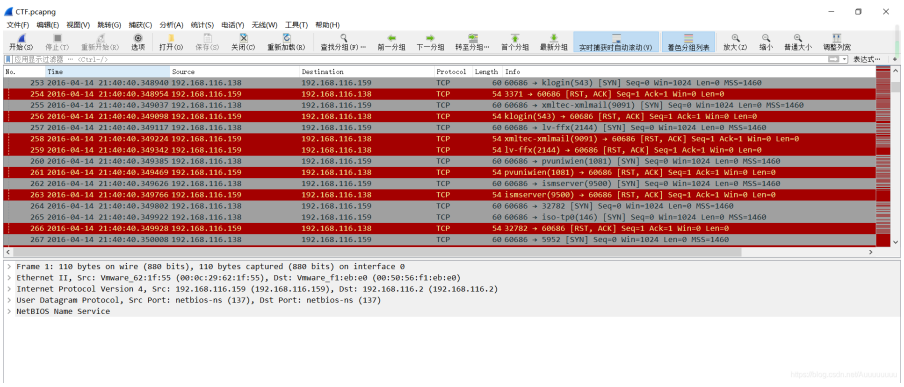
```
root@kali:~/_caidao.pcapng.extracted# ls
1E43
root@kali:~/_caidao.pcapng.extracted#
root@kali:~/_caidao.pcapng.extracted# binwalk  -e 1E43

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             POSIX tar archive

root@kali:~/_caidao.pcapng.extracted# ls
1E43  _1E43.extracted
root@kali:~/_caidao.pcapng.extracted# cd _1E43.extracted/
root@kali:~/_caidao.pcapng.extracted/_1E43.extracted# ls
0.tar  flag
root@kali:~/_caidao.pcapng.extracted/_1E43.extracted# cd flag/
root@kali:~/_caidao.pcapng.extracted/_1E43.extracted/flag# ls
flag.txt
root@kali:~/_caidao.pcapng.extracted/_1E43.extracted/flag# cat flag.txt
key{8769fe393f2b998fa6a11afe2bfcd65e}root@kali:~/_caidao.pcapng.extracted/_1E43.extracted/flag#
```


这么多数据包

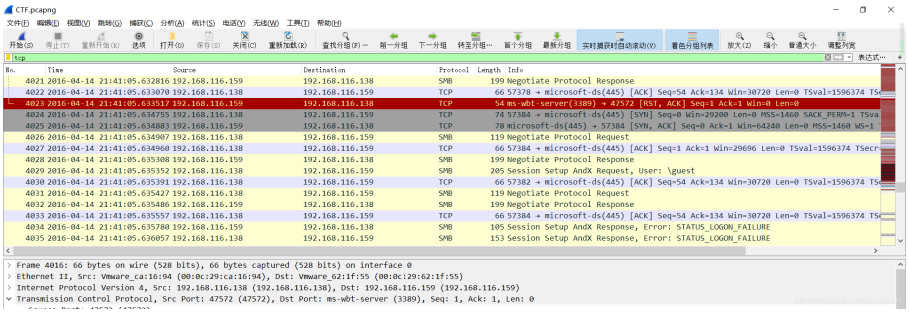
题目提示

> 这么多数据包找找吧，先找到getshell的流

数据包 TCP有大量的标红状态，猜测为攻击机对目标进行的扫描爆破等操作



过滤得到TCP协议数据包，观察发现 **4023**为最后一条标红数据

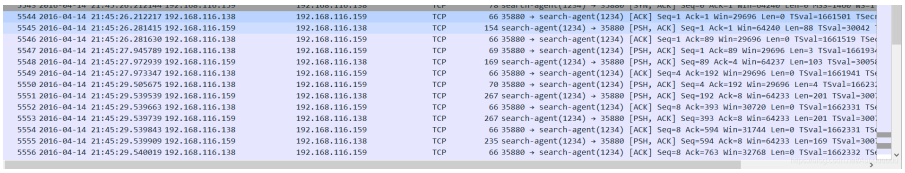


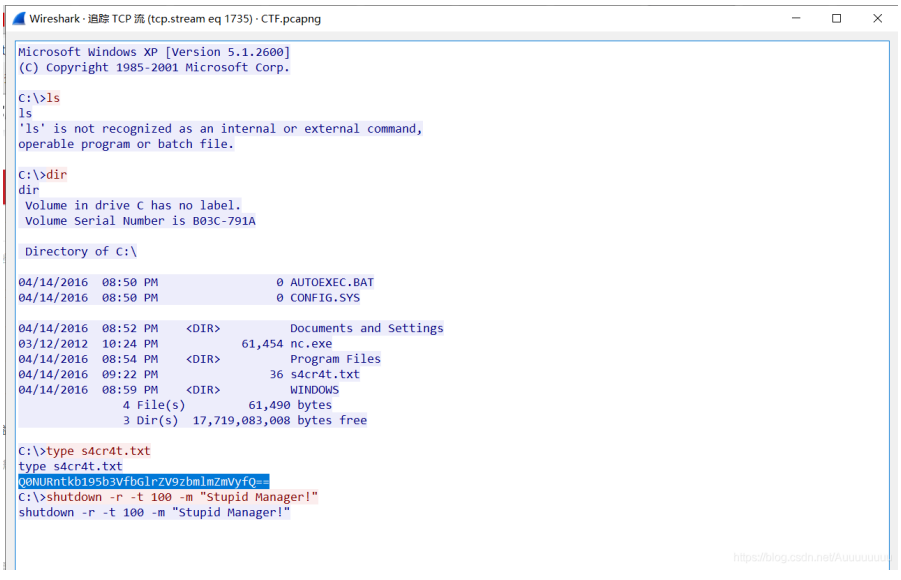
往下看呀看呀看呀，追踪TCP流查看，数据包过多，一无所获

尝试过滤一些数据包，观察发现，此期间大部分是端口4444与端口1040之间的通信，过滤数据包

```
tcp && ! tcp.port eq 4444 or ! tcp.port eq 1040 and ip.addr eq 192.168.116.138
```

发现在最后端口 35880 和 1234 还存在通信，追踪流查看





可疑的base64数据，解码查看，bingo，得到flag **CCTF{do_you_like_sniffer}**

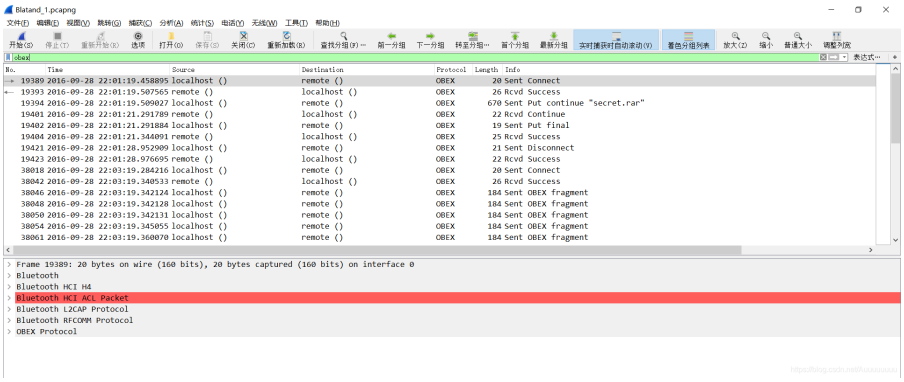
题目提示找getshell的流量，所以应该是拿到shell了，执行操作，所以应该从后往前找又高效~

手机热点

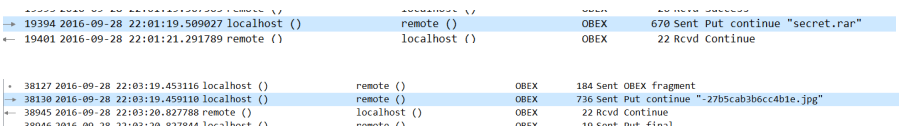
题目描述如下：

> httppan.baidu.coms1cwwdVC
> 有一天皓宝宝没了流量只好手机来共享，顺便又从手机发了点小秘密到电脑，你能找到它吗？

手机共享
推断为obex协议（蓝牙传输协议） 过滤



发现了一个压缩包，和图片



使用kali集成的 binwalk 分离文件

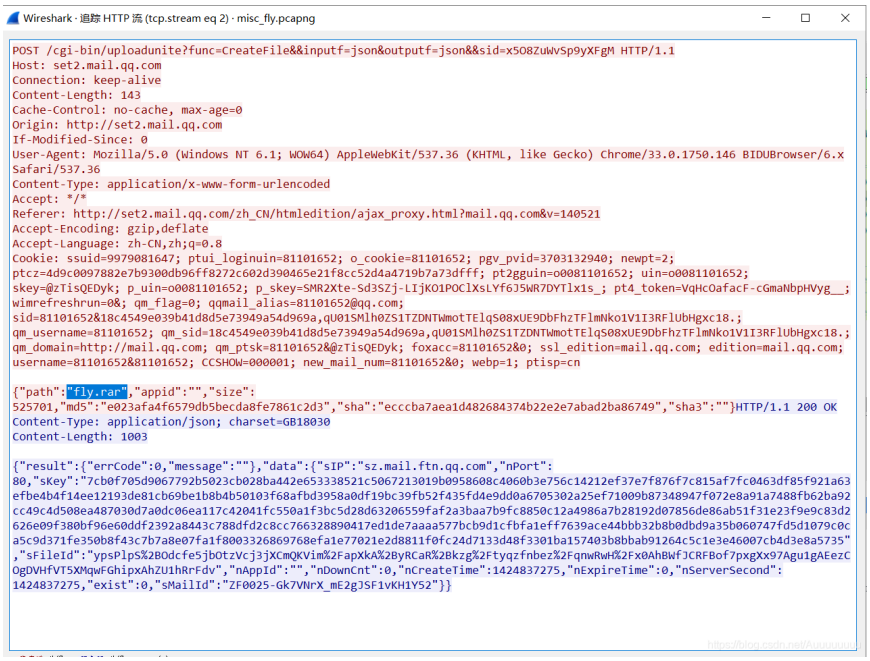
```
binwalk -e Blatand_1.pcapng
```

得到如下

打开 右下角的 flag.gif 得到 flag **SYC{this_is_bluetooth}**

抓到一只苍蝇

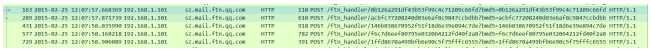
过滤得到HTTP流量，追踪HTTP流，发现一个用户通过qq邮箱上传了一个fly.rar

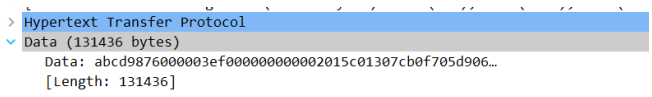


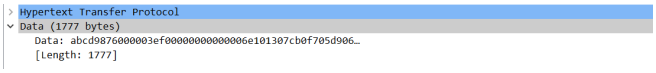
压缩包大小通过 size字段可知为 **525701**

继续过滤，从数据包的结构上下图五个数据包是数据传输的过程。点开可以看到第一个到第四个的长度为**131436**，最后一个为**1777**，应该是剩余的最后一部分数据

```
http && http.request.method== POST
```





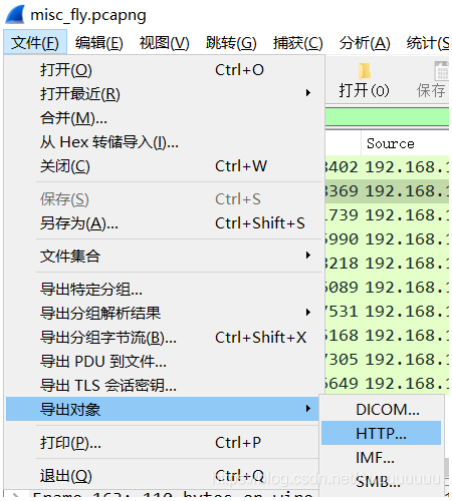


但是131436*4+1777=527521 ！ = 525701，这是由于分块传输多带了TCP的文件头，每个文件头大小规范一致

所以文件头的大小就是

131436*4+1777=527571-525701=1820/5=364 需要每个文件去掉其364字节的文件头。

提取数据，选中对应数据包，按顺序重命名为 12345



linux下使用 dd 命令 合成文件

```
语法：dd [选项]

if =输入文件（或设备名称）。

of =输出文件（或设备名称）。

ibs = bytes 一次读取bytes字节，即读入缓冲区的字节数。

skip = blocks 跳过读入缓冲区开头的ibs*blocks块。

obs = bytes 一次写入bytes字节，即写入缓冲区的字节数。

bs = bytes 同时设置读/写缓冲区的字节数（等于设置ibs和obs）。
即：（dd if=文件名 bs=输入输出块的大小 skip=偏移量 of=新的文件名）
```

```
dd if=1 bs=1 skip=364 of=1.1

dd if=2 bs=1 skip=364 of=2.1

dd if=3 bs=1 skip=364 of=3.1

dd if=4 bs=1 skip=364 of=4.1

dd if=5 bs=1 skip=364 of=5.1
```

```
root@kali:~/ctf# ls
1  2  3  4  5
root@kali:~/ctf#
root@kali:~/ctf# dd if=1 bs=1 skip=364 of=1.1
记录了131072+0 的读入
记录了131072+0 的写出
131072 bytes (131 kB, 128 KiB) copied, 0.228129 s, 575 kB/s
root@kali:~/ctf# dd if=2 bs=1 skip=364 of=2.1
记录了131072+0 的读入
记录了131072+0 的写出
131072 bytes (131 kB, 128 KiB) copied, 0.228951 s, 572 kB/s
root@kali:~/ctf# dd if=3 bs=1 skip=364 of=3.1
记录了131072+0 的读入
记录了131072+0 的写出
131072 bytes (131 kB, 128 KiB) copied, 0.224865 s, 583 kB/s
root@kali:~/ctf# dd if=4 bs=1 skip=364 of=4.1
记录了131072+0 的读入
记录了131072+0 的写出
131072 bytes (131 kB, 128 KiB) copied, 0.233962 s, 560 kB/s
root@kali:~/ctf# dd if=5 bs=1 skip=364 of=5.1
记录了1413+0 的读入
记录了1413+0 的写出
1413 bytes (1.4 kB, 1.4 KiB) copied, 0.00266204 s, 531 kB/s
root@kali:~/ctf# ls
1  1.1  2  2.1  3  3.1  4  4.1  5  5.1
root@kali:~/ctf#
```

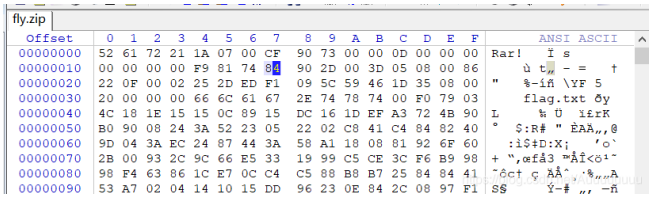

cat拼接得到压缩包 fly.rar

```
cat  1.1  2.1  3.1  4.1  5.1  >  fly.rar
```

```
root@kali:~/ctf# cat  1.1  2.1  3.1  4.1  5.1  >  fly.rar
root@kali:~/ctf# ls
1  1.1  2  2.1  3  3.1  4  4.1  5  5.1  fly.rar
root@kali:~/ctf#
```

fly.rar打开报错，存在加密，winhex打开

伪加密，修改加密位，将0x84位置改为0x80即可



解压得到flag.txt文件，打开乱码，修改后缀为 exe 运行发现.....苍蝇

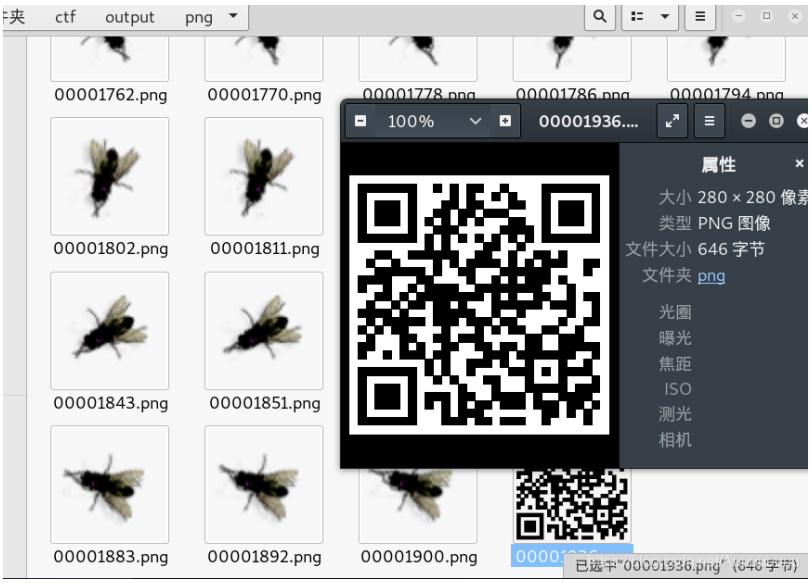
使用kali 集成的binwalk 查看，发现可疑图片

```
binwalk flag.exe
```

```
969698        0xECBE2         Zlib compressed data, best compression
973060        0xED904         PNG image, 60 x 60, 8-bit/color RGBA, non-interlaced
974018        0xEDCC2         Zlib compressed data, best compression
979947        0xEF3EB         PC bitmap, Windows 3.x format,, 49 x 23 x 8
990196        0xF1BF4         XML document, version: "1.0"
991232        0xF2000         PNG image, 280 x 280, 1-bit colormap, non-interlaced
root@kali:~/ctf#
```

使用 foremost 分离图片，在一堆苍蝇图片底部中找到一个二维码

```
foremost -v -i flag.exe
```

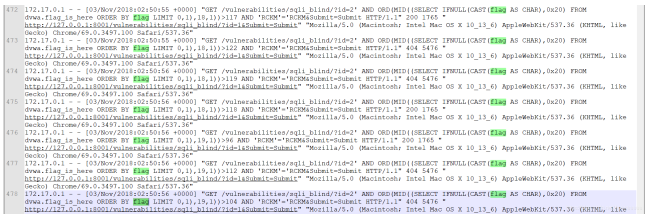


扫描得到flag **flag{m1Sc_oxO2_Fly}**

日志审计

下载得到 Access.log文件，Unicode解码发现是 **dvwa靶场的sql盲注过程**

搜索flag字段得到如下，sqlmap通过二分法注入爆破对应字段



网上copy的脚本，提取字符

```
# coding:utf-8
import re
import urllib

f = open('C:\\access.log','r')
lines = f.readlines()
datas = []
for line in lines:
    t = urllib.unquote(line)
    if '1765' in t and 'flag' in t:  # 过滤出与flag相关，正确的猜解
        datas.append(t)

flag_ascii = {}
for data in datas:
    matchObj = re.search( r'LIMIT 0,1\),(.*?),1\)\)>(.*?) AND', data)
    if matchObj:
        key = int(matchObj.group(1))
        value = int(matchObj.group(2))+1
        flag_ascii[key] = value     # 使用字典，保存最后一次猜解正确的ascii码

flag = ''
for value in flag_ascii.values():
    flag += chr(value)

print flag
```

得到flag **flag{sqlm4p_15_p0werful}**

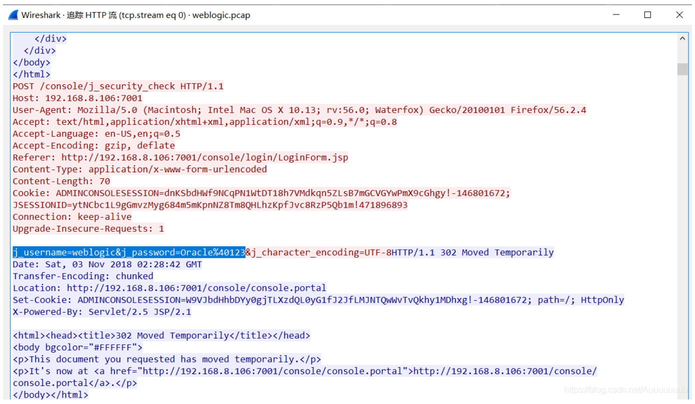
```
flag{sqlm4p_15_p0werful}
```

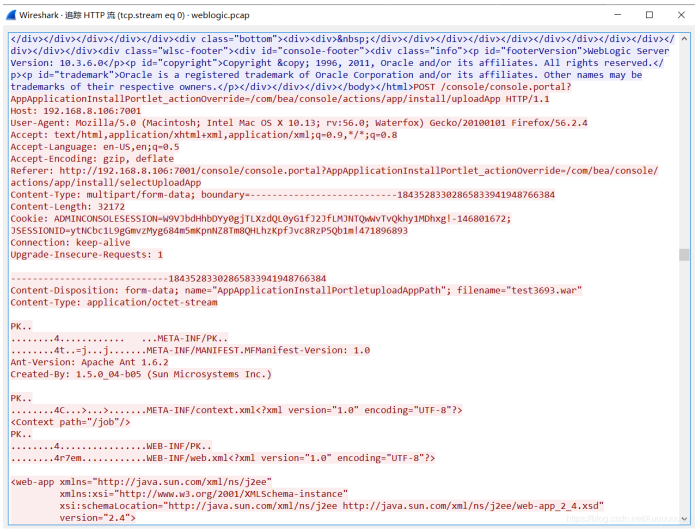
Weblogic

题目描述：

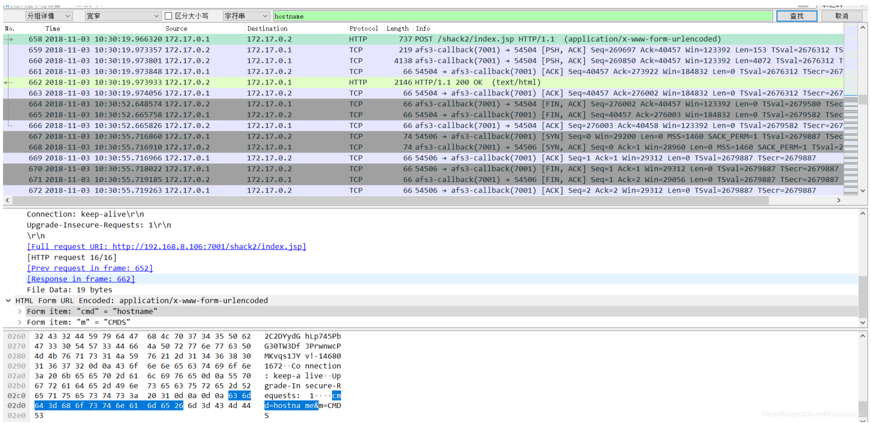
黑客攻击了Weblogic应用，请分析攻击过程，找出Weblogic的主机名。Tip：主机名为十六进制。

直接过滤得到http数据包，追踪HTTP流，观察发现爆破口令，后台部署war包上传shell得到权限

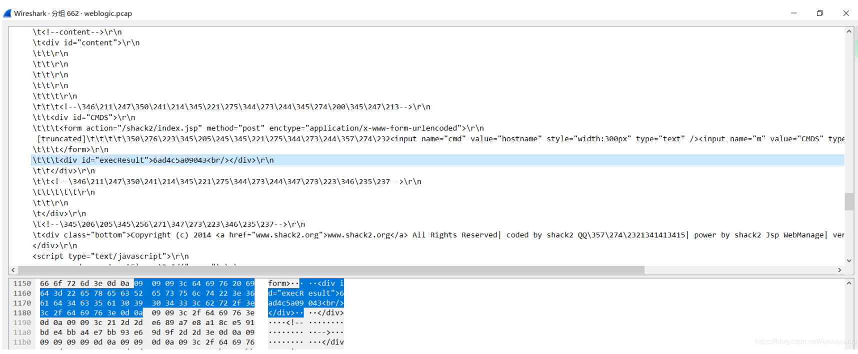




使用wireshark 搜索 hostname字符串关键字，数据包658 662 存在该关键字，发现命令执行操作



如图所示，得到16进制flag，**flag{6ad4c5a09043}**

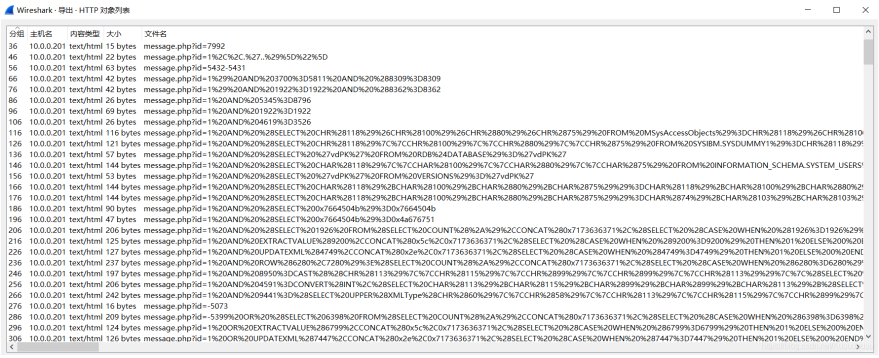


信息提取

题目描述如下

> sqlmap用过吗

打开流量包，看到很多注入语句，判断出sqlmap使用二分法进行搜索布尔盲注



网上的脚本，copy使用 需要先导出分组解析结果 为 CSV文件格式

```
import re
import urllib.parse

# 更改为自己从wireshark提取出的csv文件地址
f = open(r"D:\temp\sqlmap.csv")
lines = f.readlines()
datas = []
# 转码，保存进datas
for line in lines:
    datas.append(urllib.parse.unquote(line))
lines = []  # 懒得改，就复用一下，这个lines保存注入flag的url
for i in range(len(datas)):  # 提取出注入flag的url
    if datas[i].find("isg.flags ORDER BY `value` LIMIT 0,1),1,1))>64") > 0:
        lines = datas[i:]
        break
flag = {}
# 用正则匹配
macth1 = re.compile(r"LIMIT 0,1\),(\d*?),1\)\)>(\d*?) HTTP/1.1")
macth2 = re.compile(r'"HTTP","(\d*?)","HTTP/1.1 200 OK')
for i in range(0, len(lines), 2):  # 因为有返回响应，所以步长为2
    get1 = macth1.search(lines[i])
    if get1:
        key = int(get1.group(1))  # key保存字符的位置
        value = int(get1.group(2))  # value保存字符的ascii编码
        get2 = macth2.search(lines[i + 1])
        if get2:
            if int(get2.group(1)) > 450:
                value += 1
        flag[key] = value  # 用字典保存flag
f.close()
result = ''
for value in flag.values():
    result += chr(value)
print(result)

# ISG{BLind_SQl_InJEcTi0N_DeTEcTEd}
```

最后得到flag **ISG{BLind_SQl_InJEcTi0N_DeTEcTEd}**

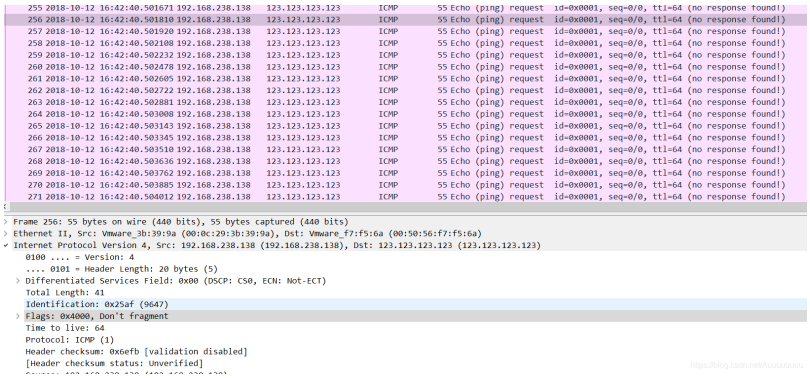
特殊后门

题目描述：

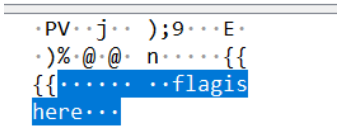
从通信方式的角度看，后门可分为http/https型、irc型、dns型、icmp型等等。安全人员抓到一份可疑的流量包，请从中分析出利用某种特殊协议传输的数据。

某种特殊的协议，尝试得知为 **icmp协议** 过滤

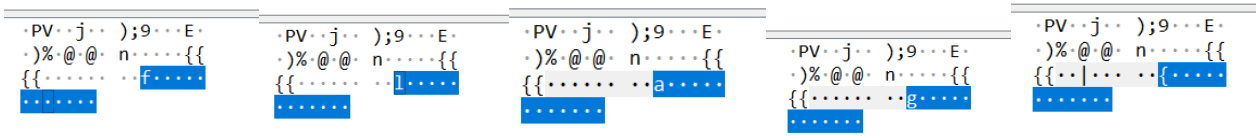
搜索flag，得到下图效果



查看第一个数据包，发现提示



继续观察，发现数据包依次存在单个字符



拼接得到 flag **flag{lcmp_backdoor_can_transfer-some_infomation}**