

# BugkuCTF(old)----流量分析题目Writeup

原创

于 2019-09-26 17:51:40 发布 1002 收藏 4

分类专栏: CTF

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) 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/Auuuuuuu/article/details/101366456>

版权



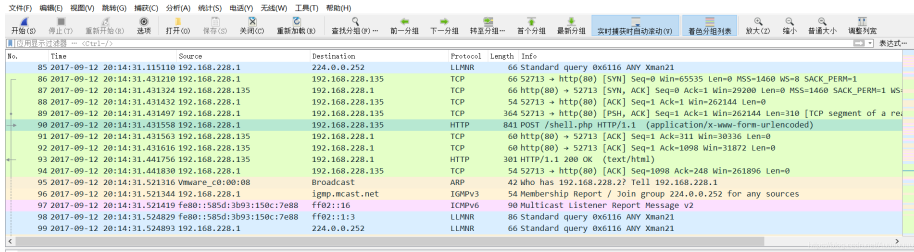
CTF 专栏收录该内容

5 篇文章 0 订阅

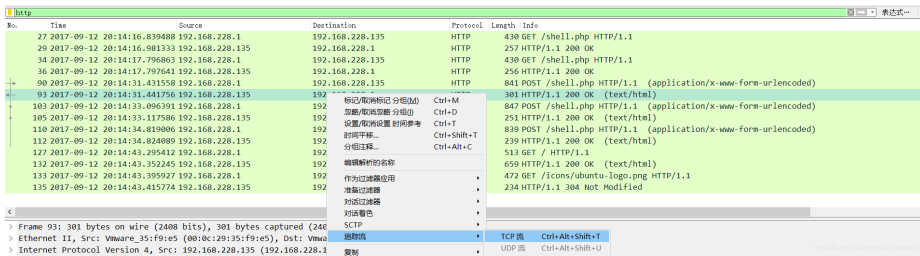
订阅专栏

## flag被盜

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}



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
root@kali:~#
```

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}`

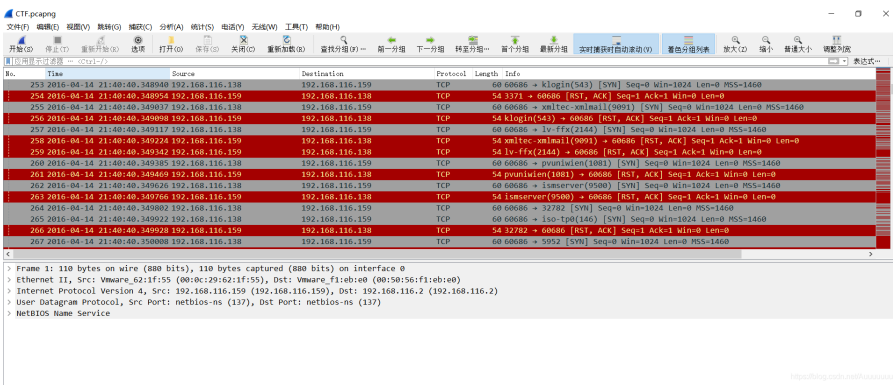
```
root@kali:~/_caidao.pcapng.extracted# ls
1E43
root@kali:~/_caidao.pcapng.extracted#
root@kali:~/_caidao.pcapng.extracted# binwalk -e 1E43
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0          POSIX tar archive
root@kali:~/_caidao.pcapng.extracted# ls
1E43  _1E43.extracted
root@kali:~/_caidao.pcapng.extracted# 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

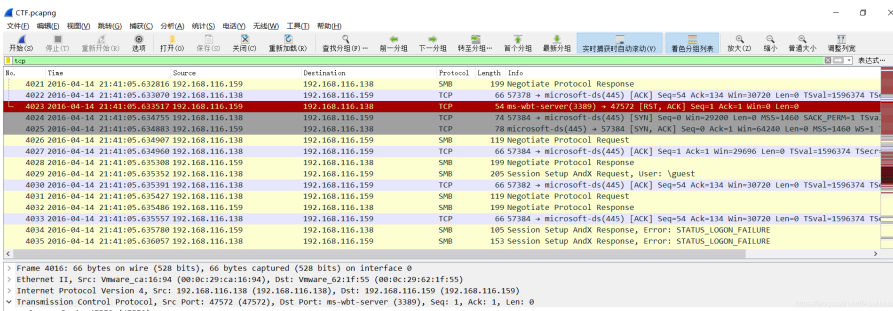
题目提示

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

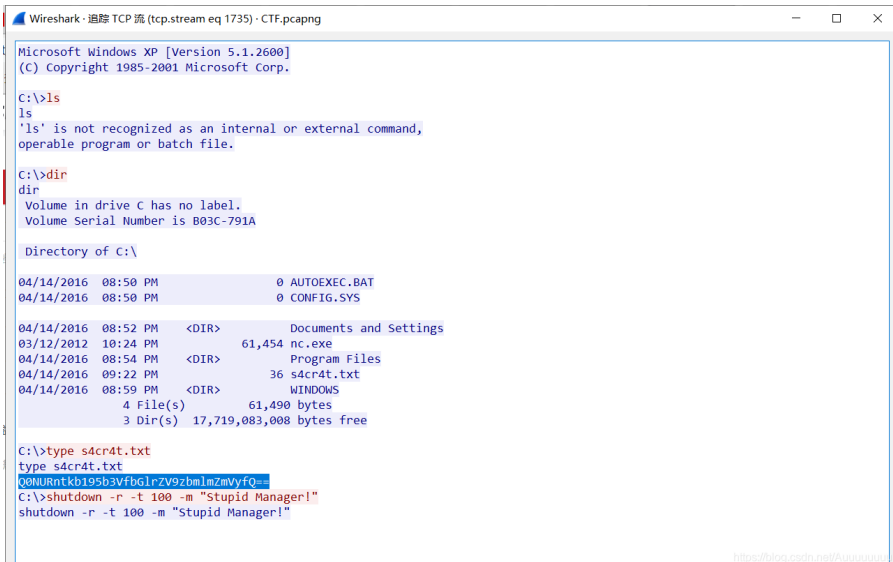
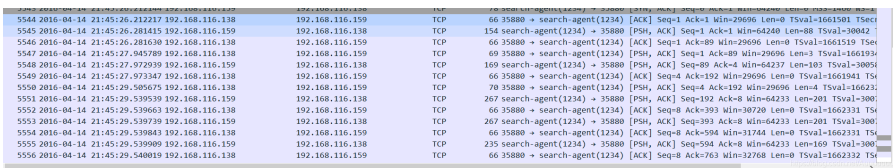


往下看呀看呀看呀，追踪TCP流查看，数据包过多，一无所获

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

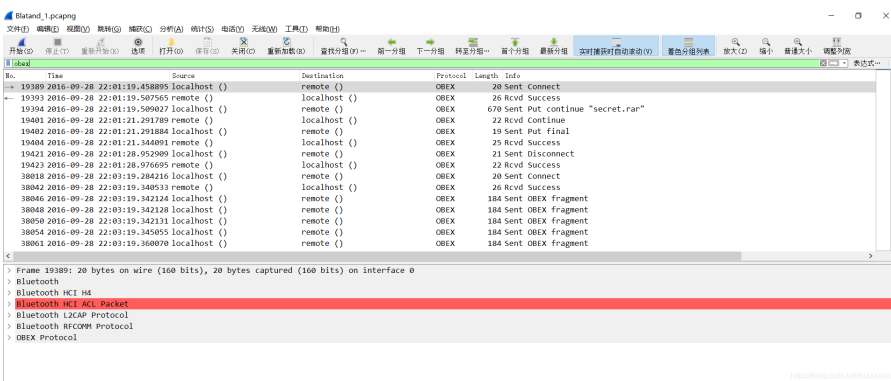
题目描述如下：

httppan.baidu.coms1cwwdVC

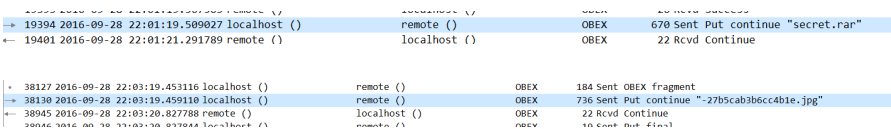
有一天皓宝宝没了流量只好手机来共享，顺便又从手机发了点小秘密到电脑，你能找到它吗？

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

```
root@kali:~/_Blatand_1.pcapng.extracted# ls
1019F.gz      48CEA7.gz    78B35B       7A8B87       7F408B.gz    A663D.zlib
10969.gz      48F471.gz    78BDA3.gz    7ABF6C.gz    7F4D93       A665D
317A9F.gz     6042.gz      78CB38.gz    7BA0AA       7F525F.gz    A665D.zlib
36E0BC.gz     635416.rar   790F5F.gz    7BA91B.gz    7F623B.gz    B3FD6
390007        756D27.gz    791813       7BBAA7       7F686F.gz    B3FD6.zlib
390007.zlib   75A488.gz    791FE2       7BC738.gz    7F7693.gz    B4006
398006        764472.gz    79349A.gz    7BDE40.gz    7F7DC1       B4006.zlib
39F91A        765C52       799528       7C0476       7F81D1       B4026
3A0CB3.gz     7660EE       799D84       7C0D13.gz    7FB533.gz    B4026.zlib
3A7DD8.gz     766678.gz    79A55A       7DF9F7.gz    805627.gz    B439D
3A889E        766C19.gz    79AD41.gz    7E052E       8141A3.gz    B439D.zlib
3AC17E        76EBE8.gz    7A3F83.gz    7E0FCF.gz    818AC6       C39482.gz
3BF6C6.gz     76FA73.gz    7A4A6F.gz    7E1BE4.gz    838D12       EB47.gz
3D063E.gz     774FC5.gz    7A52F2.gz    7E21CF.gz    8C4E9E       FB97.gz
40F207.gz     78199E.gz    7A5FA0       7E2A51       8C4E9E.zlib  flag.gif
450CF8.gz     78A617.gz    7A62EE       7F2C25       A06CB.gz
47D8E6.gz     78AEE0       7A7468.gz    7F3676       A663D
```

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

```

Wireshark - 跟踪 HTTP 流 (tcp.stream eq 2) - misc_fly.pcapng
POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1
Host: set2.mail.qq.com
Connection: keep-alive
Content-Length: 143
Cache-Control: no-cache, max-age=0
Origin: http://set2.mail.qq.com
If-Modified-Since: 0
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.146 BIDUBrowser/6.x Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Referer: http://set2.mail.qq.com/zh_CN/html/edition/ajax_proxy.html?mail.qq.com&v=140521
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN, zh;q=0.8
Cookie: ssid=9979081647; ptui_loginuin=81101652; o_cookie=81101652; pgv_pid=3703132940; newpt=2; ptcz=4d9c0097882e7b9300db96ff8272c602d39046e521f8cc52d4a4719b7a73dfff; pt2gguin=00081101652; uin=00081101652; skey=@ZTisQEDyk; p_uin=00081101652; p_skey=SMR2Xte-Sd35Zj-LIJKO1POCLXsLYf635WR7DVTlxIs; pt4_token=VqHC0aFACF-cGmaNbpHVYg; winrefreshrun=0&; qm_flag=0; qqmail_alias=81101652@qq.com; sid=81101652&18c4549e039b41d8d5e73949a54d969a,qu01SMlh0Z51ZDNtWmotELqS08XUE90DFhZTFImNko1V1I3RF1UbHgXC18.; qm_username=81101652; qm_sid=18c4549e039b41d8d5e73949a54d969a,qu01SMlh0Z51ZDNtWmotELqS08XUE90DFhZTFImNko1V1I3RF1UbHgXC18.; qm_domain=http://mail.qq.com; qm_ptsk=81101652&ZTisQEDyk; foxacc=81101652&0; ssl_edition=mail.qq.com; edition=mail.qq.com; username=81101652&81101652; CSHQW=000001; new_mail_num=81101652&0; webp=1; ptisp=en

{"path": "fly.rar", "appid": "", "size": 525701, "md5": "e023afa4f6579db5becda8fe7861c2d3", "sha": "eccbba7aea1d482684374b22e2e7abad2ba86749", "sha3": ""} HTTP/1.1 200 OK
Content-Type: application/json; charset=GB18030
Content-Length: 1003

{"result": {"errCode": 0, "message": ""}, "data": {"sIP": "sz.mail.ftn.qq.com", "nPort": 80, "sKey": "7cb0f705d9067792b5023cb028ba42e65338521c5067213019b0958608c4060b3e756c14212ef37e7f8767c815af7fc0463df85f921a63efbe4b14ee12193de81c69b118b4b50103f688fb3958a0df19bc39f52f435fdae9d0a6705302a25ef71009b87348947f072e8a91a7488fb62ba92cc49c4508ea4870307ad0c06a117c42041f3c550a1f3bc5d28d63206559faf2a3baa7b9fc8850c12a4986a7b28192d07856de86ab51f3e23f9e9c83d2626e9f380bf9e6e6df2392a8443c788df42c8cc766328989417ed1de7aaa577bc9d1c7bfaf1eff7639ace44bb328b8bd9a35b060747fd5d1079dc0a5c9d371fe3508f43c7b7a8e07fa1f80033268e9768efa1e77021e2d8811f0fc24d7133d48f3301ba157403b8bbab91264c5c1e3e46007cb4d3e8a5735", "sFileId": "ypsP1p5Z80dcfesjbotzVc3jXcmQKv1m2fapKkAzByACaKkZ2fyqzfnbe2ZfmgwWwH2f-x0AHBwFjCRFBof7pxgXx97AguigAEezC0gDvHfVTSXmqwFchpxAhZUIhRfdv", "nAppId": "", "nDownCnt": 0, "nCreateTime": 1424837275, "nExpireTime": 0, "nServerSecond": 1424837275, "exist": 0, "sMailId": "ZF0025-Gk7VWwX_mE2g5F1vKH1Y52"}}

```

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
```

```

185 2011-02-25 12:07:57.000000 192.168.1.101 81.46.111.116:80 HTTP 138 POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1
200 2011-02-25 12:07:57.017379 192.168.1.101 81.46.111.116:80 HTTP 610 POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1
432 2011-02-25 12:07:58.000000 192.168.1.101 81.46.111.116:80 HTTP 182 POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1
517 2011-02-25 12:07:58.000000 192.168.1.101 81.46.111.116:80 HTTP 182 POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1
720 2011-02-25 12:07:58.000000 192.168.1.101 81.46.111.116:80 HTTP 182 POST /cgi-bin/uploadunite?func=CreateFile&input=json&output=json&sid=x508ZuWv5p9yXfgM HTTP/1.1

```

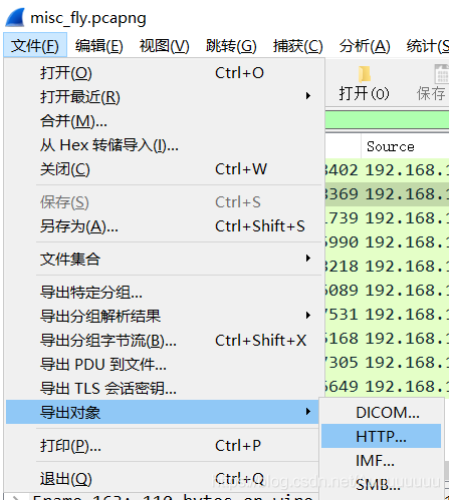
> Hypertext Transfer Protocol  
Data (131436 bytes)  
Data: abcd9876000003ef00000000002015c01307cb0f705d906...  
[Length: 131436]

> Hypertext Transfer Protocol  
Data (1777 bytes)  
Data: abcd9876000003ef000000000006e101307cb0f705d906...  
[Length: 1777]

但是 $131436 * 4 + 1777 = 527521 \neq 525701$ ，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
```

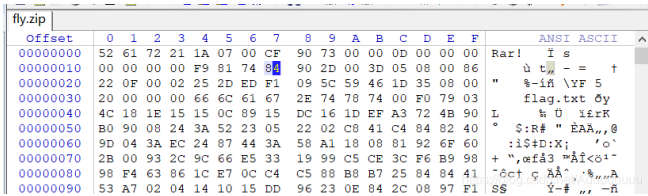
```

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



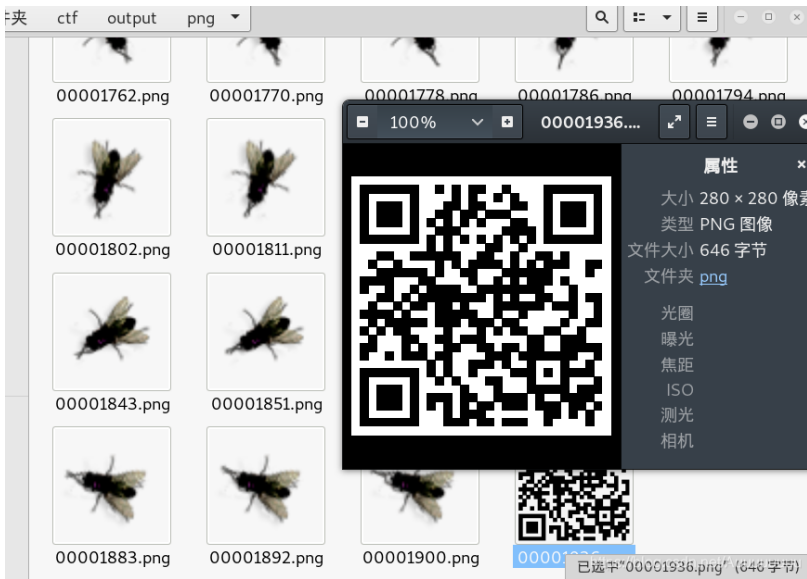
使用kali 集成的binwalk 查看，发现可疑图片

```
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

```

477 172.17.0.1 - - [03/May/2018:02:50:55 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 200 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
478 172.17.0.1 - - [03/May/2018:02:50:55 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 404 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
479 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 404 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
480 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 200 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
481 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 200 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
482 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 404 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
483 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 404 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"
484 172.17.0.1 - - [03/May/2018:02:50:56 +0800] "GET /vulnabilities/all_blind?id=2' AND OR(MD5(SELECT IPURL(CAST(1765 AS CHAR),0x29 FROM
vwa_flag_is_base OR(SB BY LIMIT 0,1) AND 'HCM'='HCGMSubsit=Subsit HTTP/1.1' 404 1765 "
https://172.17.0.1:8000/vulnabilities/all_blind?id=2&id=Subsit" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/69.0.3497.100 Safari/537.36"

```

## 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，后部署war包上传shell得到权限

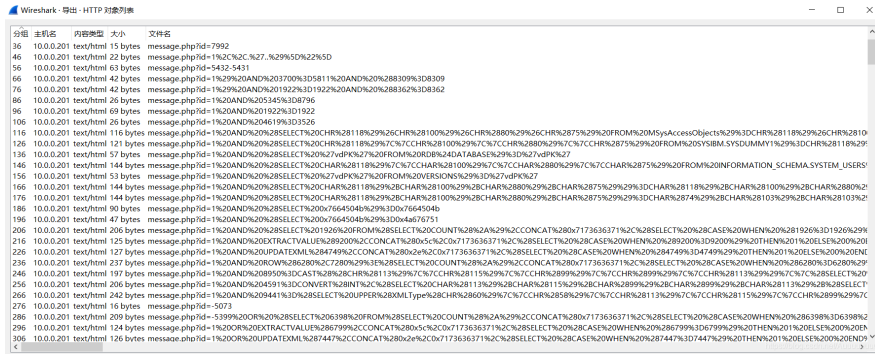


# 信息提取

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
mach1 = re.compile(r"LIMIT 0,1\),(\d*?),1\)\)>(\d*?) HTTP/1.1")
mach2 = 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 = mach1.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 = mach2.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

255	2018-10-12	16:42:40.501671	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
256	2018-10-12	16:42:40.501810	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
257	2018-10-12	16:42:40.501920	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
258	2018-10-12	16:42:40.502109	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
259	2018-10-12	16:42:40.502232	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
260	2018-10-12	16:42:40.502479	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
261	2018-10-12	16:42:40.502695	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
262	2018-10-12	16:42:40.502722	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
263	2018-10-12	16:42:40.502881	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
264	2018-10-12	16:42:40.503008	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
265	2018-10-12	16:42:40.503143	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
266	2018-10-12	16:42:40.503345	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
267	2018-10-12	16:42:40.503510	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
268	2018-10-12	16:42:40.503636	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
269	2018-10-12	16:42:40.503762	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
270	2018-10-12	16:42:40.503880	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)
271	2018-10-12	16:42:40.504012	192.168.238.138	123.123.123.123	ICMP	55 Echo (ping) request	id=0x0001, seq=0/0, ttl=64 (no response found)

```

> Frame 256: 55 bytes on wire (440 bits), 55 bytes captured (440 bits)
> Ethernet II, Src: VMware_39:59:9a (00:0c:29:39:59:9a), Dst: VMware_77:f5:6a (00:50:56:f7:f5:6a)
> Internet Protocol Version 4, Src: 192.168.238.138 (192.168.238.138), Dst: 123.123.123.123 (123.123.123.123)
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total length: 41
    Identification: 0x25af (9647)
    > Flags: 0x4000, Don't Fragment
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x6efb [validation disabled]
    [Header checksum status: Unverified]
  .....
```

查看第一个数据包，发现提示

```

·PV·j·);9·E·
·)%·@·@·n·...{{
{{·...·flagis
here·...
```

继续观察，发现数据包依次存在单个字符

```

·PV·j·);9·E·      ·PV·j·);9·E·      ·PV·j·);9·E·      ·PV·j·);9·E·      ·PV·j·);9·E·
·)%·@·@·n·...{{    ·)%·@·@·n·...{{    ·)%·@·@·n·...{{    ·)%·@·@·n·...{{    ·)%·@·@·n·...{{
{{·...·f·...       {{·...·l·...       {{·...·a·...       {{·...·g·...       {{·...·{·...
·...·              ·...·              ·...·              ·...·              ·...·
```

拼接得到 flag `flag{icmp_backdoor_can_transfer-some_infomation}`