

网络空间安全市赛Misc一道简单的流量分析题

原创

疯狂的1998 于 2021-11-28 12:47:33 发布 4259 收藏 2

分类专栏: [CTF](#) 文章标签: [安全](#)

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/MarkRao/article/details/121589667>

版权



[CTF 专栏收录该内容](#)

16 篇文章 2 订阅

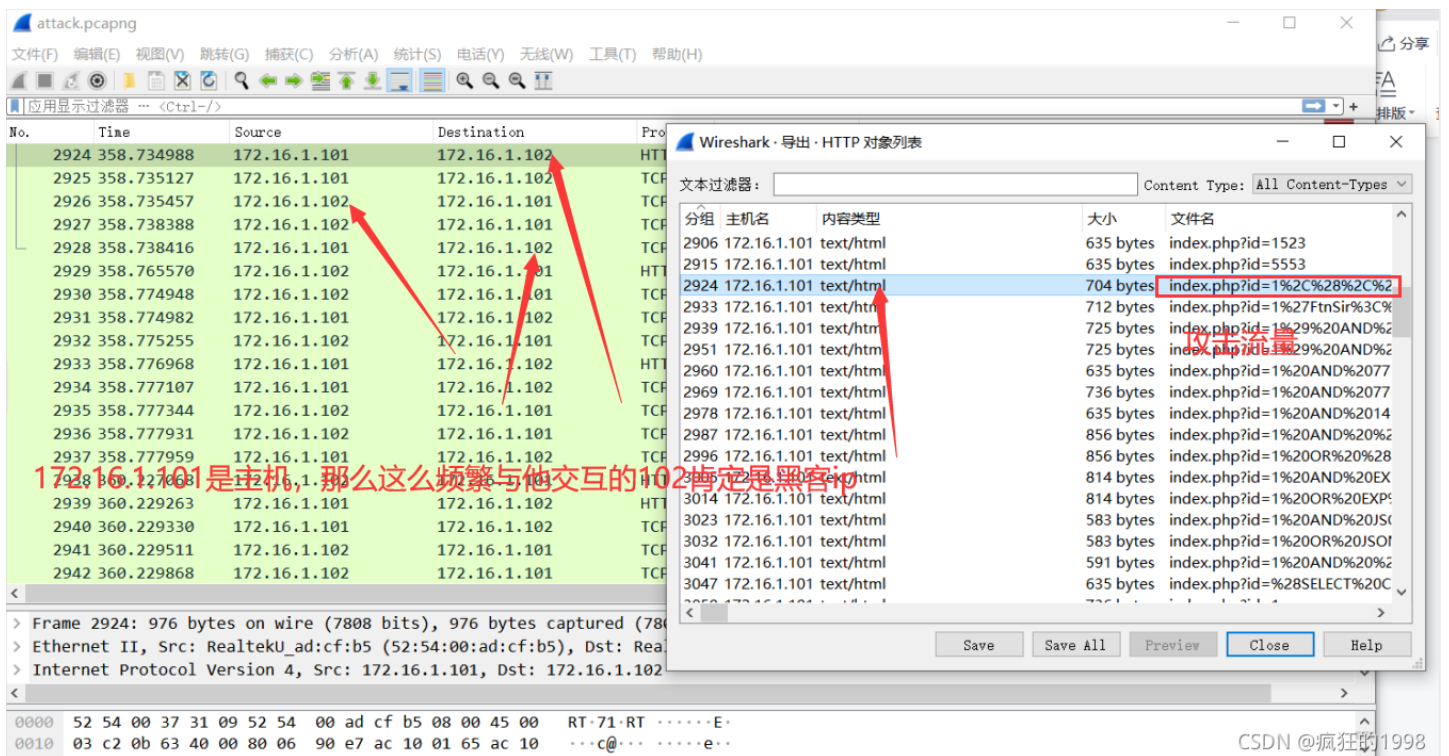
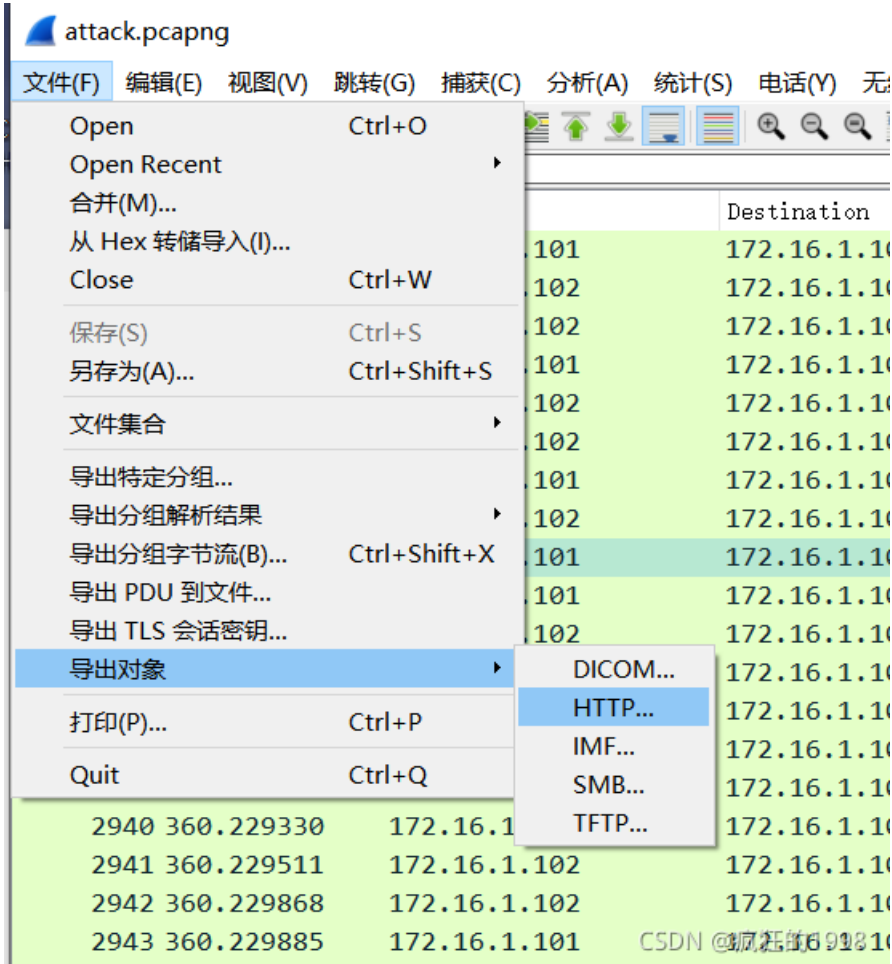
订阅专栏

这道题给了一个流量包，内部分了7个小题

题目介绍:

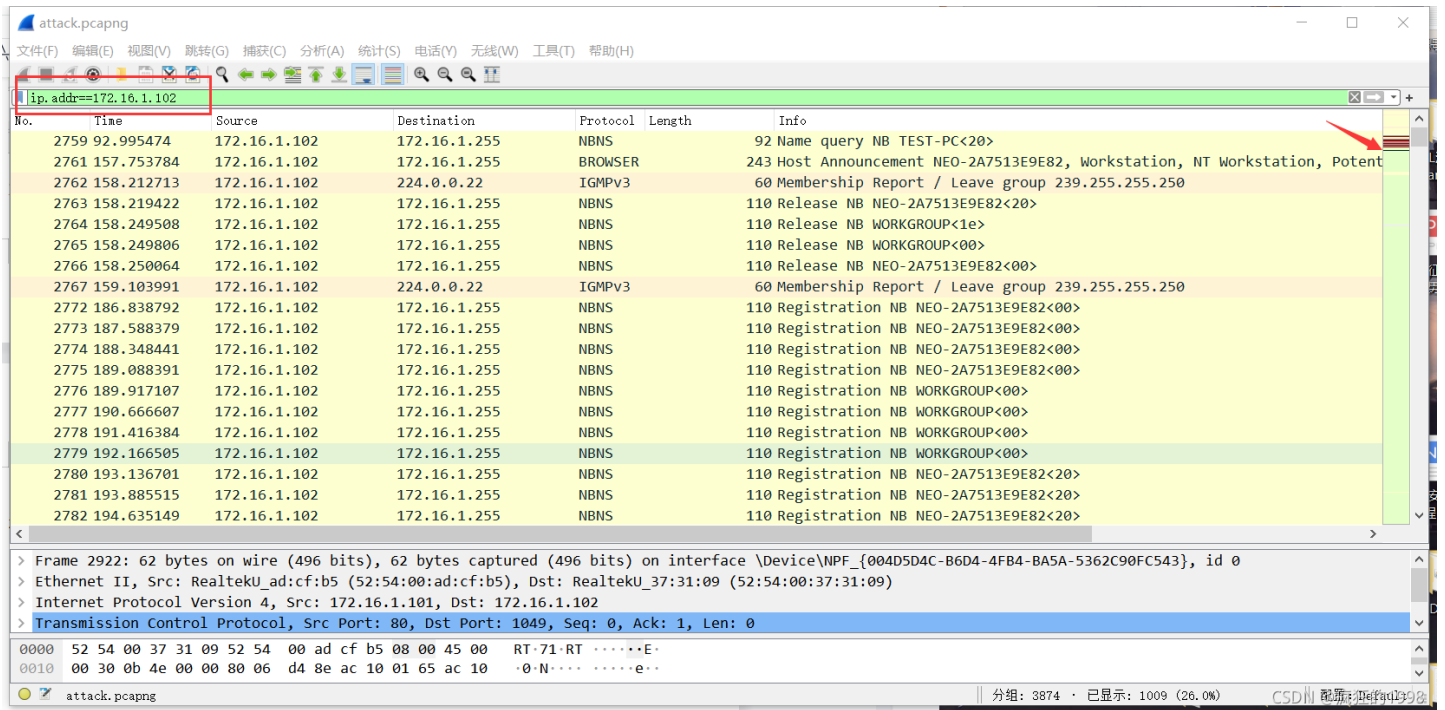
- 1、使用Wireshark查看并分析服务器桌面下的attack.pcapng数据包文件，通过分析数据包attack.pcapng找出黑客的IP地址，并将黑客的IP地址作为FLAG（形式：[IP地址]）提交
- 2、继续查看数据包文件attack.pacapng，分析出黑客扫描了哪些端口，并将全部的端口作为FLAG（形式：[端口名1,端口名2,端口名3...,端口名n]）从低到高提交；
- 3、继续查看数据包文件attack.pacapng分析出黑客最终获得的用户名是什么，并将用户名作为FLAG（形式：[用户名]）提交；
- 4、继续查看数据包文件attack.pacapng分析出黑客最终获得的密码是什么，并将密码作为FLAG（形式：[密码]）提交；
- 5、继续查看数据包文件attack.pacapng分析出黑客连接一句话木马的密码是什么，并将一句话密码作为FLAG（形式：[一句话密码]）提交；
- 6、继续查看数据包文件attack.pacapng分析出黑客下载了什么文件，并将文件名及后缀作为FLAG（形式：[文件名.后缀名]）提交；
- 7、继续查看数据包文件attack.pacapng提取出黑客下载的文件，并将文件里面的内容为FLAG（形式：[文件内容]）提交。

1: 首先可以在导出HTTP中查看攻击流量

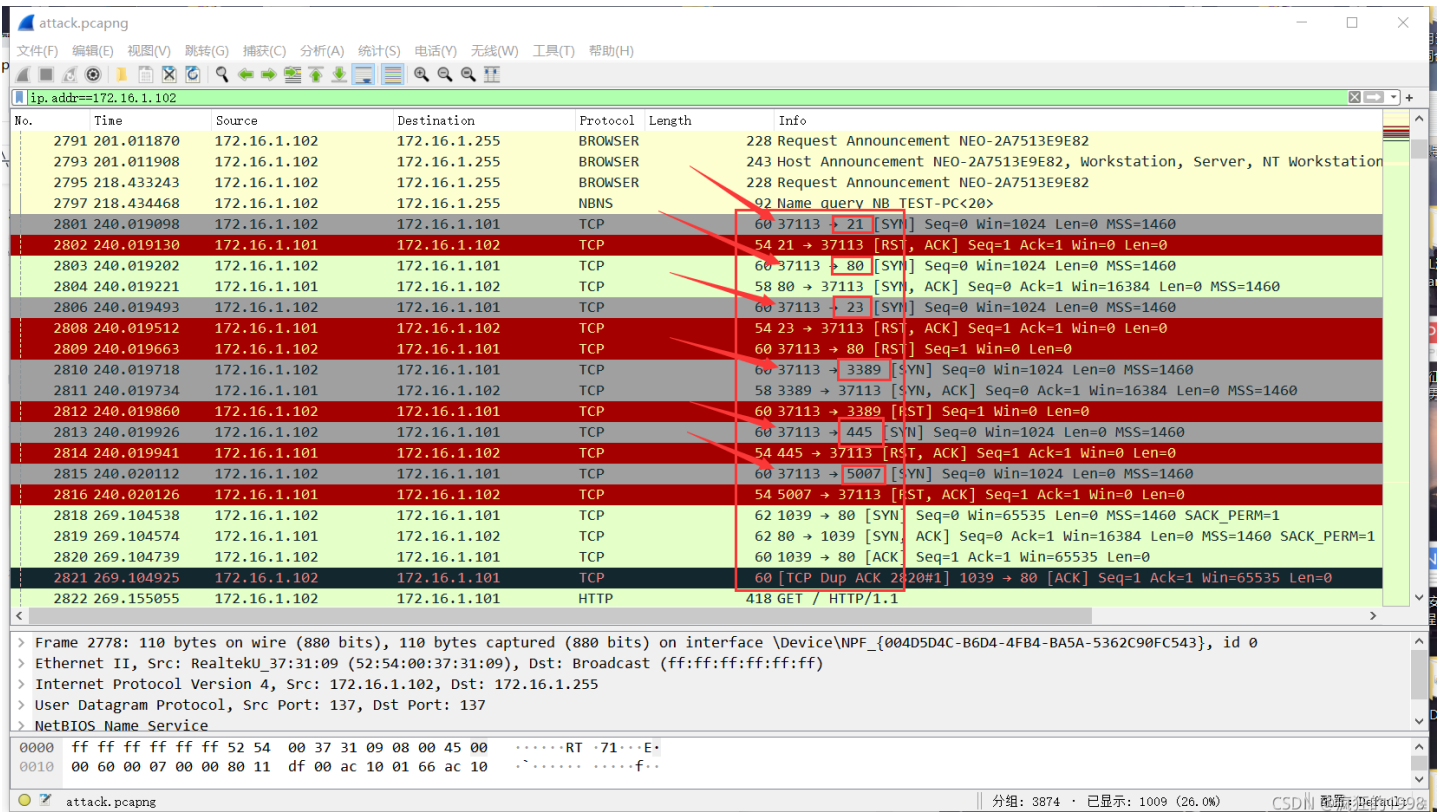


第一题的flag答案便是: [172.16.1.101]

2、查端口，我们找到黑客ip后便过滤ip，输入ip.addr==172.16.1.102



往下分析你会发现很集中的有端口扫描现象，黑客是通过37113来扫描的，你只需要看这些便可，便能得到flag，所以第二题flag为[21,23,80,445,3389,5007]



3/4两题、要分析出黑客最终获得的用户名就得找到黑客什么时候登录的，攻击点在哪里，那就用同样的方法，导出http处查看，发现有登录login请求，点击它便会自动跳转到那条流量，关闭导出框，对3747这条流量进行TCP追踪，这样便看到了用户名和密码，

第三题flag便是 [Lancelot]

第四题flag便是 [12369874]

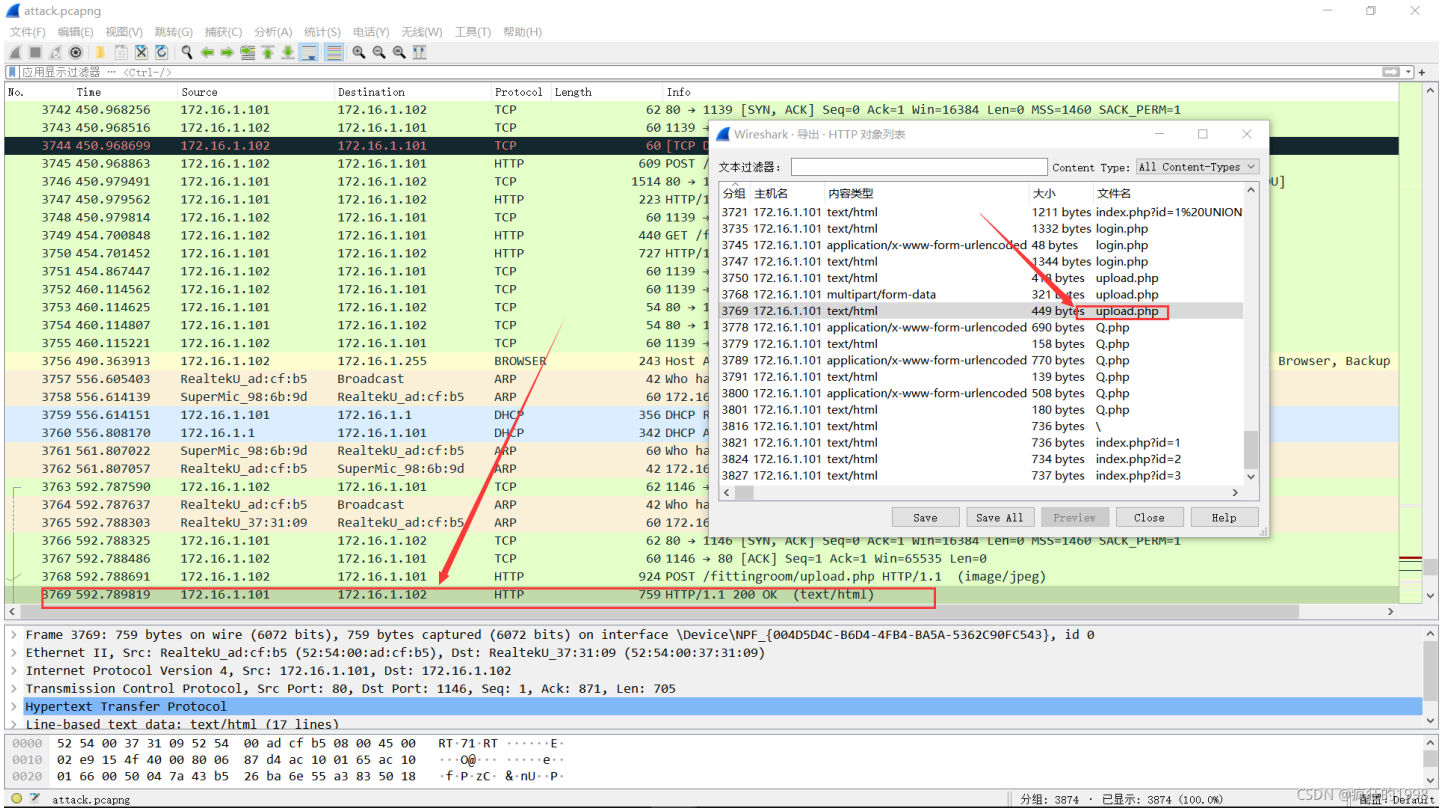
The screenshot shows a Wireshark capture of network traffic. The packet list pane displays several packets, with packet 3744 highlighted. The details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP response status is 200 OK. The content type is application/x-www-form-urlencoded. The response body contains HTML code for a login form, including a submit button labeled 'submit=LoginHTTP/1.1 200 OK'.

No.	Time	Source	Destination	Protocol	Length	Info
3725	406.513099	172.16.1.101	172.16.1.102	TCP	54	80 → 1134 [ACK] Seq=1432 Ack=445 Win=65092 Len=0
3726	435.902504	172.16.1.102	172.16.1.101	TCP	60	1137 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3727	435.902627	172.16.1.102	172.16.1.101	TCP	60	1137 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3728	435.902702	172.16.1.102	172.16.1.101	TCP	60	1137 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3729	442.986155	172.16.1.102	172.16.1.101	TCP	62	1138 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3730	442.986199	172.16.1.102	172.16.1.101	TCP	62	1138 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3731	442.986375	172.16.1.102	172.16.1.101	TCP	60	1138 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3732	442.986598	172.16.1.102	172.16.1.101	TCP	60	1138 → 80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
3733	442.986919	172.16.1.102	172.16.1.101	HTTP	484	GET /
3734	442.987721	172.16.1.101	172.16.1.102	TCP	1514	80 → 1138 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3735	442.987875	172.16.1.101	172.16.1.102	HTTP	182	HTTP/1.1 200 OK (text/html)
3736	442.988121	172.16.1.102	172.16.1.101	TCP	60	1138 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3737	448.509898	172.16.1.101	172.16.1.102	TCP	54	80 → 1138 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3738	448.510217	172.16.1.102	172.16.1.101	TCP	60	1138 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3739	448.510467	172.16.1.102	172.16.1.101	TCP	60	1138 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3740	448.510509	172.16.1.101	172.16.1.102	TCP	54	80 → 1138 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3741	450.968209	172.16.1.102	172.16.1.101	TCP	62	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3742	450.968256	172.16.1.101	172.16.1.102	TCP	62	80 → 1139 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3743	450.968516	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3744	450.968699	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3745	450.968863	172.16.1.102	172.16.1.101	HTTP	609	POST /fittingroom/login.php
3746	450.979491	172.16.1.101	172.16.1.102	TCP	1514	80 → 1139 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3747	450.979562	172.16.1.101	172.16.1.102	HTTP	223	HTTP/1.1 200 OK (text/html)
3748	450.979814	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3749	454.700848	172.16.1.102	172.16.1.101	HTTP	440	GET /fittingroom/upload.php
3750	454.701452	172.16.1.102	172.16.1.101	HTTP	727	HTTP/1.1 200 OK (text/html)
3751	454.867447	172.16.1.101	172.16.1.102	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3752	460.114562	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [FIN, ACK] Seq=942 Ack=2303 Win=64862 Len=0

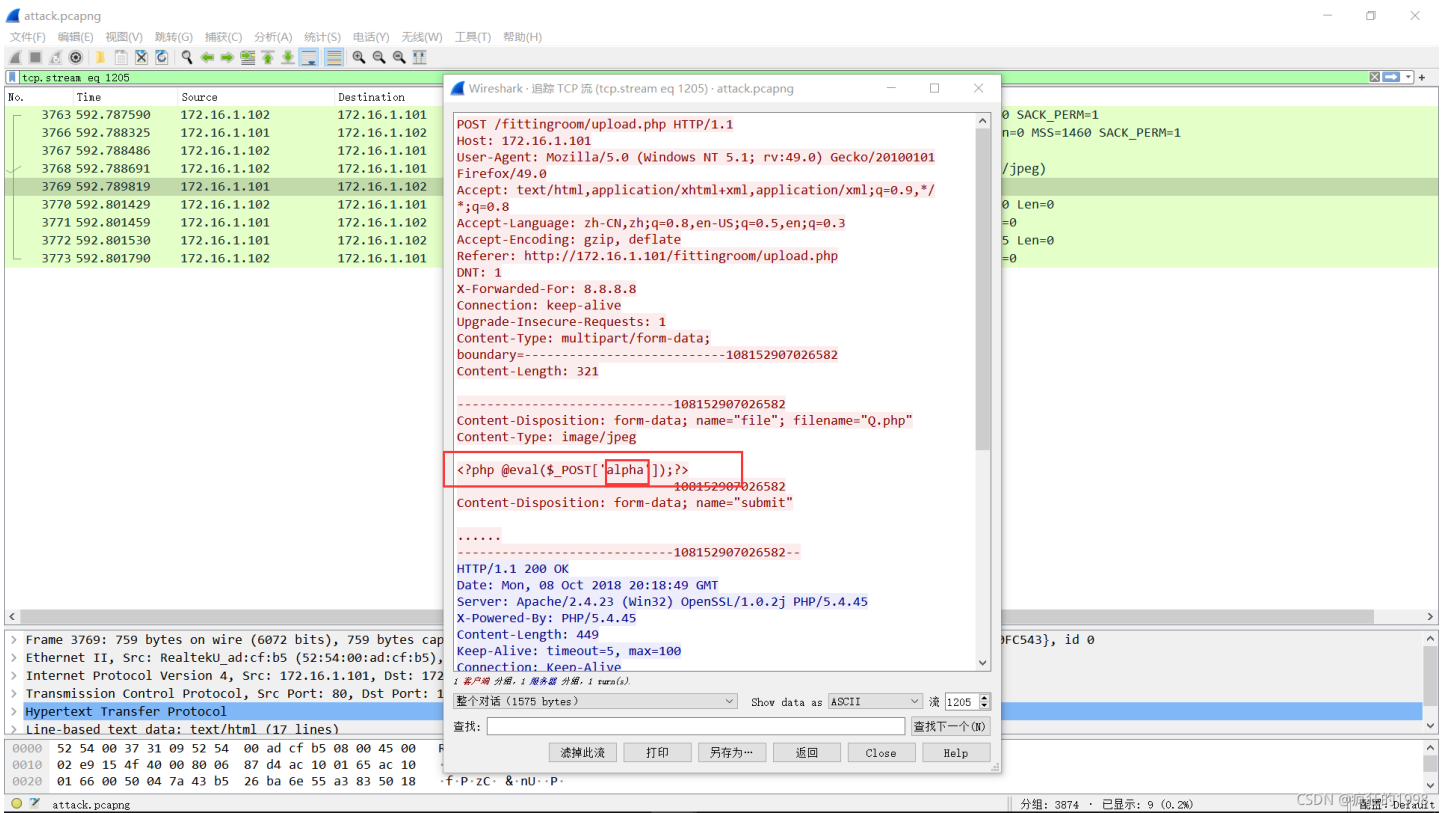
The screenshot shows a detailed view of an HTTP POST request in Wireshark. The packet list pane shows packet 3744 selected. The details pane shows the structure of the selected packet, including Ethernet II, Internet Protocol Version 4, Transmission Control Protocol, and Hypertext Transfer Protocol. The HTTP request method is POST. The content type is application/x-www-form-urlencoded. The request body contains HTML code for a login form, including a submit button labeled 'submit=LoginHTTP/1.1 200 OK'. The response status is 200 OK. The content type is text/html. The response body contains HTML code for a login form, including a submit button labeled 'submit=LoginHTTP/1.1 200 OK'.

No.	Time	Source	Destination	Protocol	Length	Info
3741	450.968209	172.16.1.102	172.16.1.101	TCP	54	80 → 1138 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3742	450.968256	172.16.1.101	172.16.1.102	TCP	62	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3743	450.968516	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3744	450.968699	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3745	450.968863	172.16.1.102	172.16.1.101	HTTP	609	POST /fittingroom/login.php
3746	450.979491	172.16.1.101	172.16.1.102	TCP	1514	80 → 1139 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3747	450.979562	172.16.1.101	172.16.1.102	HTTP	223	HTTP/1.1 200 OK (text/html)
3748	450.979814	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3749	454.700848	172.16.1.102	172.16.1.101	HTTP	440	GET /fittingroom/upload.php
3750	454.701452	172.16.1.102	172.16.1.101	HTTP	727	HTTP/1.1 200 OK (text/html)
3751	454.867447	172.16.1.101	172.16.1.102	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3752	460.114562	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [FIN, ACK] Seq=942 Ack=2303 Win=64862 Len=0
3753	460.114625	172.16.1.101	172.16.1.102	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3754	460.114807	172.16.1.101	172.16.1.102	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0
3755	460.115221	172.16.1.102	172.16.1.101	TCP	60	1139 → 80 [ACK] Seq=942 Ack=2303 Win=64862 Len=0

5、第五题要找木马密码，同样的思路，跟第3、4题思路一样，找出http看木马可能会在upload上传点，果然可以看到有个upload.php



一样的道理，追踪TCP流得到木马密码，flag便是 [alpha]



6、取出黑客下载的文件名字以及后缀，跟前面一样找攻击点，但是这次在导出并没有分析出有用的东西，这边我的方法有些另类，因为这种题肯定是按照流程出的，所以后面的flag必定在前面的基础之上的，所以我依旧追踪http看upload并对他进行TCP流追踪，然后关闭追踪选项，在上面过滤处tcp.stream eq 1205 往后进行过滤并且进行大致的TCP追踪（就看第一条或后面几条流量），发现tcp.stream eq 1207有猫腻，看到flag.zip估计是答案了，便得到flag为[flag.zip]

Wireshark - 跟踪 TCP 流 (tcp.stream eq 1207) · attack.pcapng

POST /fittingroom/upload/Q.php HTTP/1.1
 Cache-Control: no-cache
 X-Forwarded-For: 101.159.98.98
 Referer: http://172.16.1.101
 Content-Type: application/x-www-form-urlencoded
 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
 Host: 172.16.1.101
 Content-Length: 770
 Connection: close

alpha=%40eval%01%28base64_decode%28%24_POST%5B%05%29%29%3B&0=QGlu
 aV9zZXQoImRpc3B5SXYlFzXjY3ZjY3ZiwiMCIp00BzZXRFdGltZV9saWpdcGwKTAC2V0
 X21hZ2ljX3F1b3Rlc19ydW50aWw1KDAP02VjaG8oIi0%2BfcIpozskRD1iYXNlNjRfZG
 Vjb2RlKCRFUE9TVF5ieSjEiXSk7JEY9QG9wZW5kaXIoIEQpO2ImKCRPT10VXUMKXtY2
 hvKjFjUjJwYjovLWY0YXRoIE5vdCBGbzVuzCBPciB0byBQZjJtaXNaw9uISip031lBh
 NleyRNPUSVTEw7JEw9TlVMTDT3aG1sZSgkTj1AcMvhZGRpcigRikRikpeyRPSREliViIi
 4KTjksVDIAGf0ZSgins1tLWQsDpponMILEBmaXlxbRpbWUoJFApKTAEJ9U93vic3
 RYKGJhc2VfY29udmVydChAZmlzZXBlcm1zKCR0KSwKcW4KSwTnCK7FI91Lx0i4Kvc
 4iXHQilKBMaXlc2l2ZSgkucUk1x0Ii4KR54iCiI7awY0QGlzX2RpcigkUCkPJE0uPS
 ROLiViIi4kUjtlbHNlCRMLj0k1t4Kujt9ZWNobyAKTS4kTdtAY2xvc2VkaXIoIEYpO2
 07ZNobygifiDwtIk7ZGllKck78z1=QzpcXB0cFNd0R5XFxxV1dCXGZpdHRpbmdyb2
 9tXFx1cGxvYmRcXk%3D%3DHTTP/1.1 200 OK
 Date: Mon, 08 Oct 2018 20:19:30 GMT
 Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
 X-Powered-By: PHP/5.4.45
 Content-Length: 139
 Connection: close
 Content-Type: text/html

flag.zip 2017-10-09 04:12:00 174 0666
 Q.php 2018-10-08 20:18:49 31 0666

(其实还有一种做法就是搜索flag,分组字节流搜flag刚好会显示那条流量也是跟上面追踪TCP得到的结果一样)

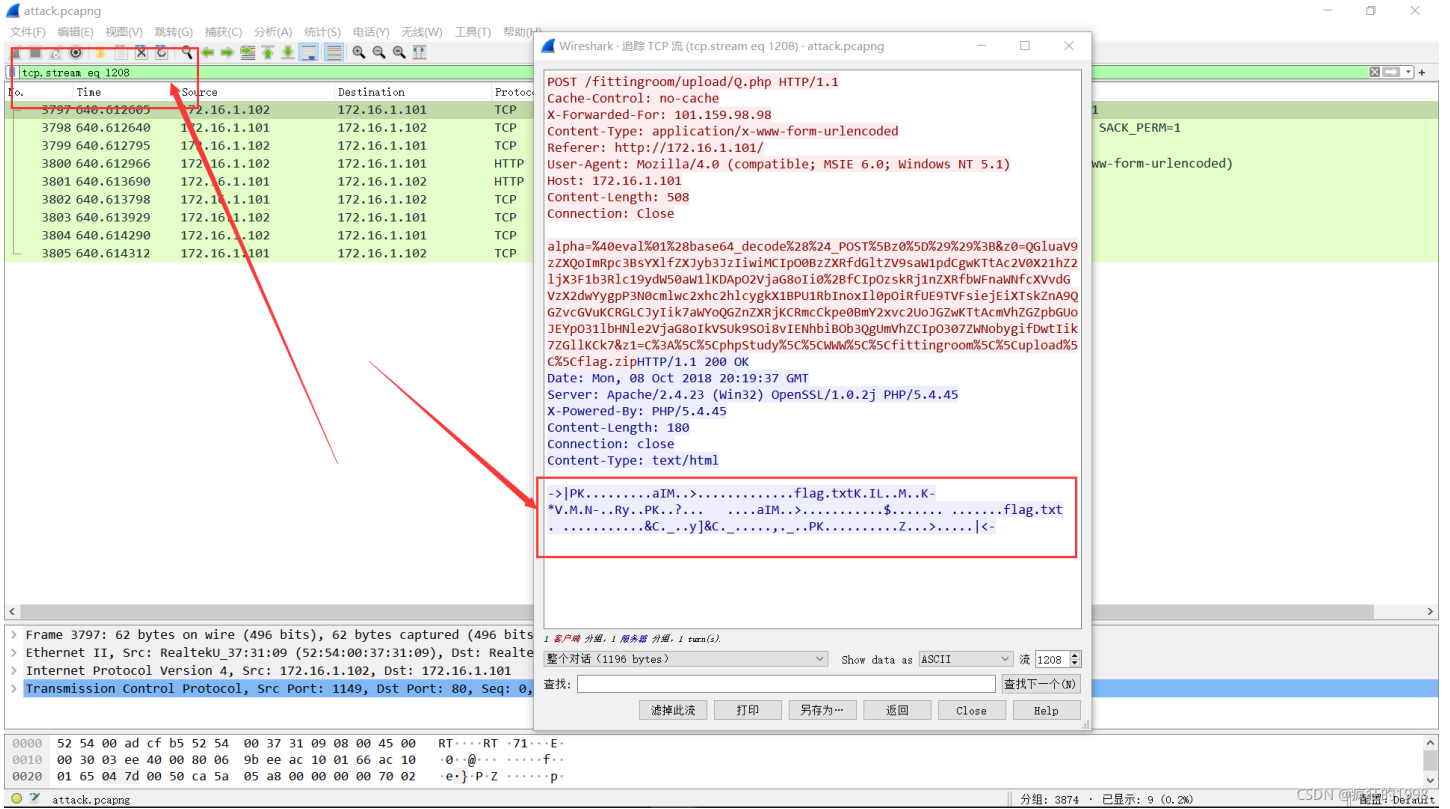
应用显示过滤器: flag

No.	Time	Source	Destination	Protocol	Length	Info
3775	632.783563	172.16.1.102	172.16.1.101	TCP	62	1147 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3776	632.783600	172.16.1.101	172.16.1.102	TCP	62	80 → 1147 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 SACK_PERM=1
3777	632.783902	172.16.1.101	172.16.1.102	TCP	60	1147 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
3778	632.784053	172.16.1.102	172.16.1.101	HTTP	1047	POST /fittingroom/upload/Q.php HTTP/1.1 (application/x-www-form-urlencoded)
3779	632.791854	172.16.1.101	172.16.1.102	HTTP	416	HTTP/1.1 200 OK (text/html)
3780	632.791951	172.16.1.101	172.16.1.102	TCP	54	80 → 1147 [FIN, ACK] Seq=363 Ack=994 Win=64542 Len=0
3781	632.801183	172.16.1.102	172.16.1.255	NBNS	92	Name query NB C.WMwCD.TOP<00>
3782	632.801265	172.16.1.102	172.16.1.101	TCP	60	1147 → 80 [ACK] Seq=994 Ack=364 Win=65173 Len=0
3783	632.802620	172.16.1.102	172.16.1.101	TCP	60	1147 → 80 [FIN, ACK] Seq=994 Ack=364 Win=65173 Len=0
3784	632.802642	172.16.1.101	172.16.1.102	TCP	54	80 → 1147 [ACK] Seq=364 Ack=995 Win=64542 Len=0
3785	633.142431	172.16.1.102	172.16.1.101	TCP	62	1148 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3786	633.142477	172.16.1.101	172.16.1.102	TCP	62	80 → 1148 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 SACK_PERM=1
3787	633.142649	172.16.1.102	172.16.1.101	TCP	60	1148 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
3788	633.142837	172.16.1.102	172.16.1.101	TCP	357	1148 → 80 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=303 [TCP segment of a reassembled PDU]
3789	633.142849	172.16.1.102	172.16.1.101	HTTP	824	POST /fittingroom/upload/Q.php HTTP/1.1 (application/x-www-form-urlencoded)
3790	633.142866	172.16.1.101	172.16.1.102	TCP	54	80 → 1148 [ACK] Seq=1 Ack=1074 Win=64462 Len=0
3791	633.144641	172.16.1.101	172.16.1.102	HTTP	397	HTTP/1.1 200 OK (text/html)
3792	633.144811	172.16.1.102	172.16.1.102	TCP	54	80 → 1148 [FIN, ACK] Seq=344 Ack=1074 Win=64462 Len=0
3793	633.144941	172.16.1.102	172.16.1.101	TCP	60	1148 → 80 [ACK] Seq=1074 Ack=345 Win=65192 Len=0
3794	633.145052	172.16.1.102	172.16.1.101	TCP	60	1148 → 80 [FIN, ACK] Seq=1074 Ack=345 Win=65192 Len=0
3795	633.145070	172.16.1.101	172.16.1.102	TCP	54	80 → 1148 [ACK] Seq=345 Ack=1075 Win=64462 Len=0
3796	633.546236	172.16.1.102	172.16.1.255	NBNS	92	Name query NB C.WMwCD.TOP<00>
3797	640.612605	172.16.1.102	172.16.1.101	TCP	62	1149 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1
3798	640.612640	172.16.1.101	172.16.1.102	TCP	62	80 → 1149 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 SACK_PERM=1
3799	640.612795	172.16.1.102	172.16.1.101	TCP	60	1149 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
3800	640.612966	172.16.1.102	172.16.1.101	HTTP	866	POST /fittingroom/upload/Q.php HTTP/1.1 (application/x-www-form-urlencoded)
3801	640.613690	172.16.1.101	172.16.1.102	HTTP	438	HTTP/1.1 200 OK (text/html)

Line-based text data: text/html (5 lines)
 ->|./\t2018-10-08 20:18:49\t0\t0777\n
 ..\t2018-10-08 19:23:49\t0\t0777\n
 flag.zip\t2017-10-09 04:12:00\t174\t0666\n
 Q.php\t2018-10-08 20:18:49\t31\t0666\n
 |<

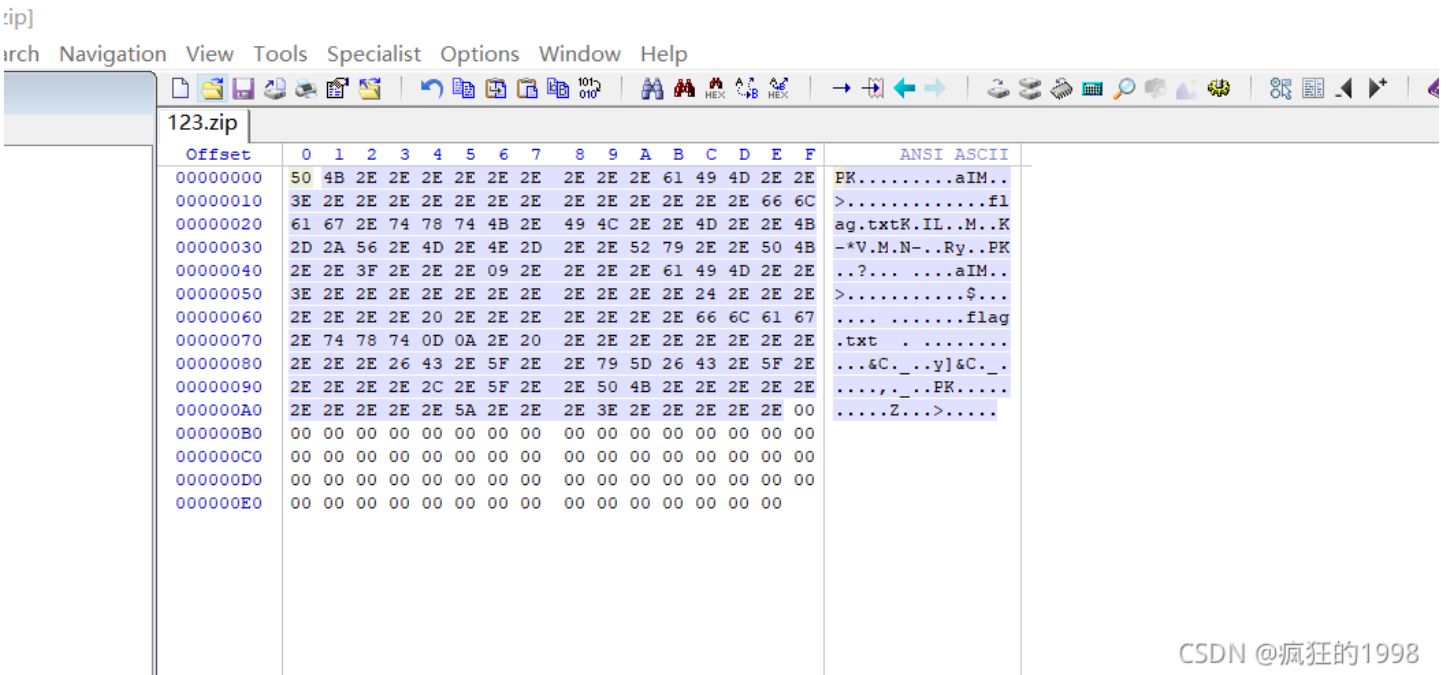
0140 37 0a 66 6c 61 67 2e 7a 69 70 09 32 30 31 37 2d 7-flag.z ip:2017-
 0150 31 30 2d 30 39 20 30 34 3a 31 32 3a 30 30 09 31 10-09 04 :12:00:1
 0160 37 34 09 30 36 36 0a 51 2e 70 68 70 09 32 30 74-0666-Q.php:20

7、最后一题提取出黑客下载的文件(这边思路弯了),并将文件里面的内容为FLAG,这题有问题跟第6题思路一样,换过滤号,往后面分析 tcp.stream eq 1208 并且对第一条进行TCP流追踪,发现zip格式的ascii,将其提取放入hex保存

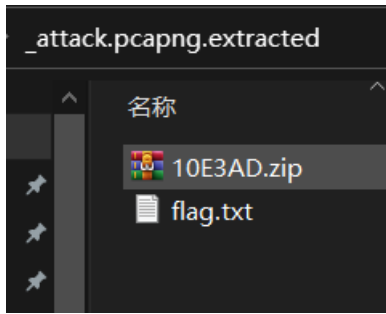


得到的ascii:

```
PK...aIM...>...flag.txtK.L...M...K-*V.M.N-...Ry...PK...?...aIM...>...$. ...flag.txt
...&C...y]&C...,_...PK...Z...>...
```



其实很明显zip开头是504B0304这显然有问题，压缩包恢复不了，直接凉凉，到这里发现是思路问题将其放入kali中进行分离流量包，binwalk -e分离可得



其中打开第一个压缩包(存在伪加密，直接用360压缩打开)便是flag.txt一样的文件
flag{Manners maketh man}



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)