Wireshark例题-CTF

原创

H3rmesk1t 于 2021-05-14 23:40:53 发布 1173 收藏 6

分类专栏： 安全学习 文章标签： wireshark ctf 网络安全

本文链接：https://blog.csdn.net/LYJ20010728/article/details/116804672

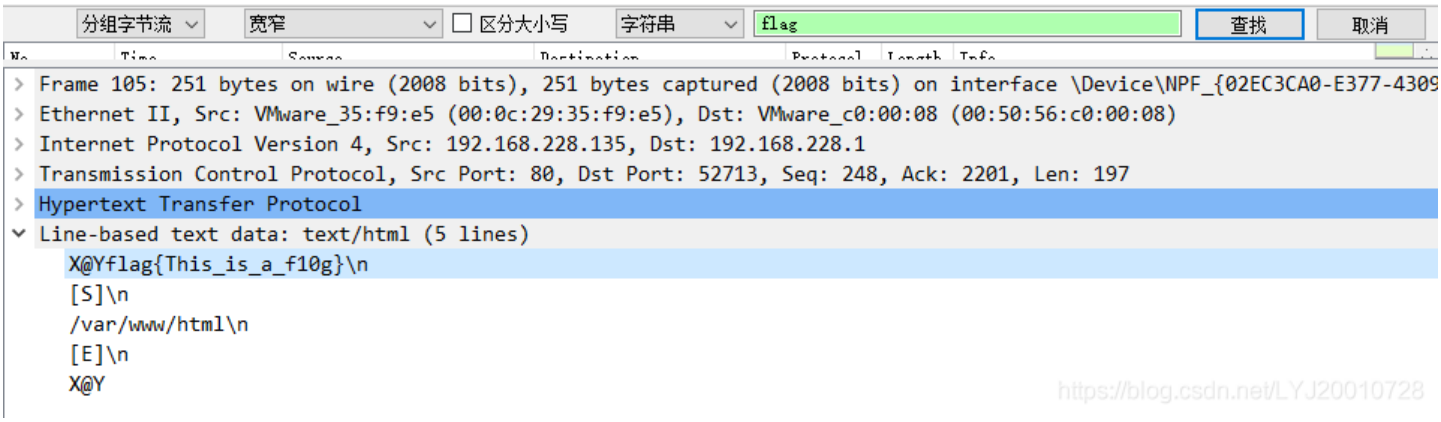
Wireshark例题-CTF

搜索

> 题目文件：key.pcapng

> 题目描述：flag被盗，赶紧溯源！

> 题目题解：
> ①可以只将这个数据包当做文本文件打开，比如用一些notepad++编辑器，然后直接搜索
> ②用Wireshark自带的搜索功能找尝试查找一些关键词（比如key、flag、shell、pass等），然后跟进可疑的数据包，根据数据包特征，很明显看出这是一个菜刀连接一句话木马的数据包，然后往下找，即可看到读取的flag



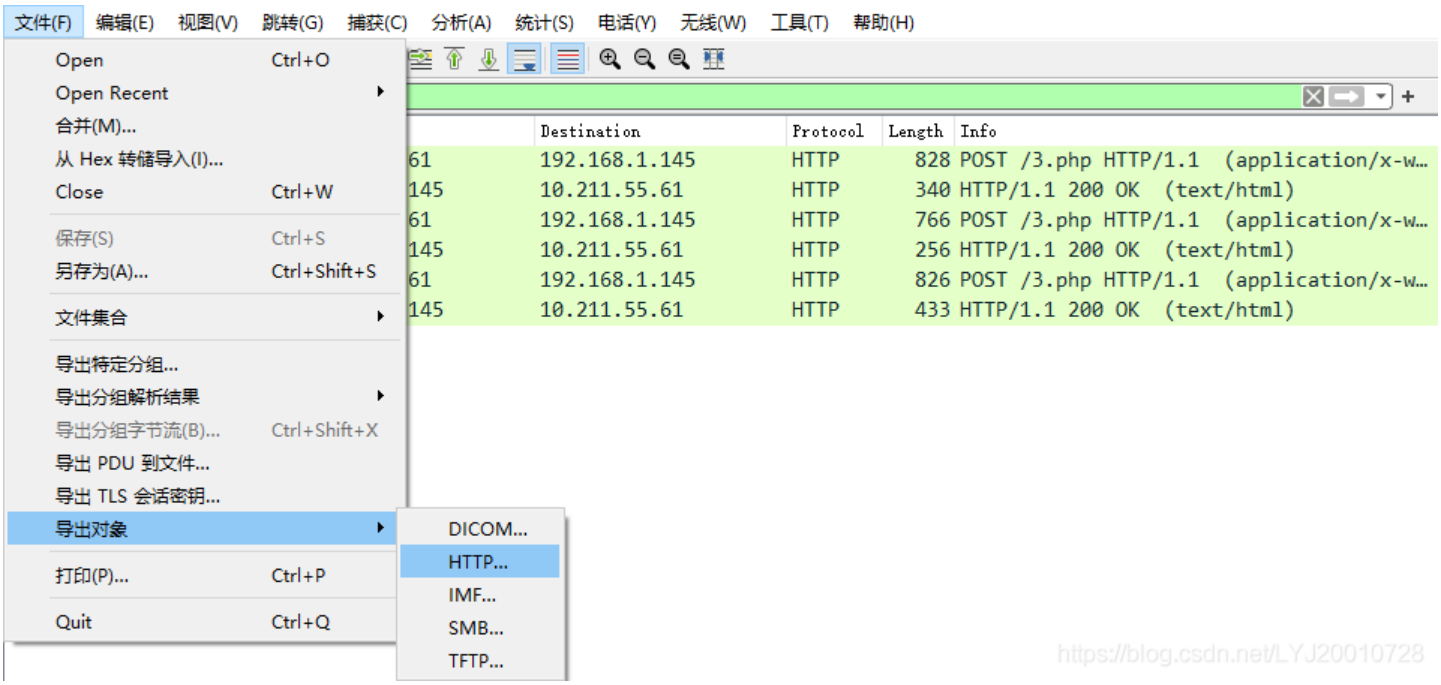
文件提取

例题一

题目文件： caidao.pcapng

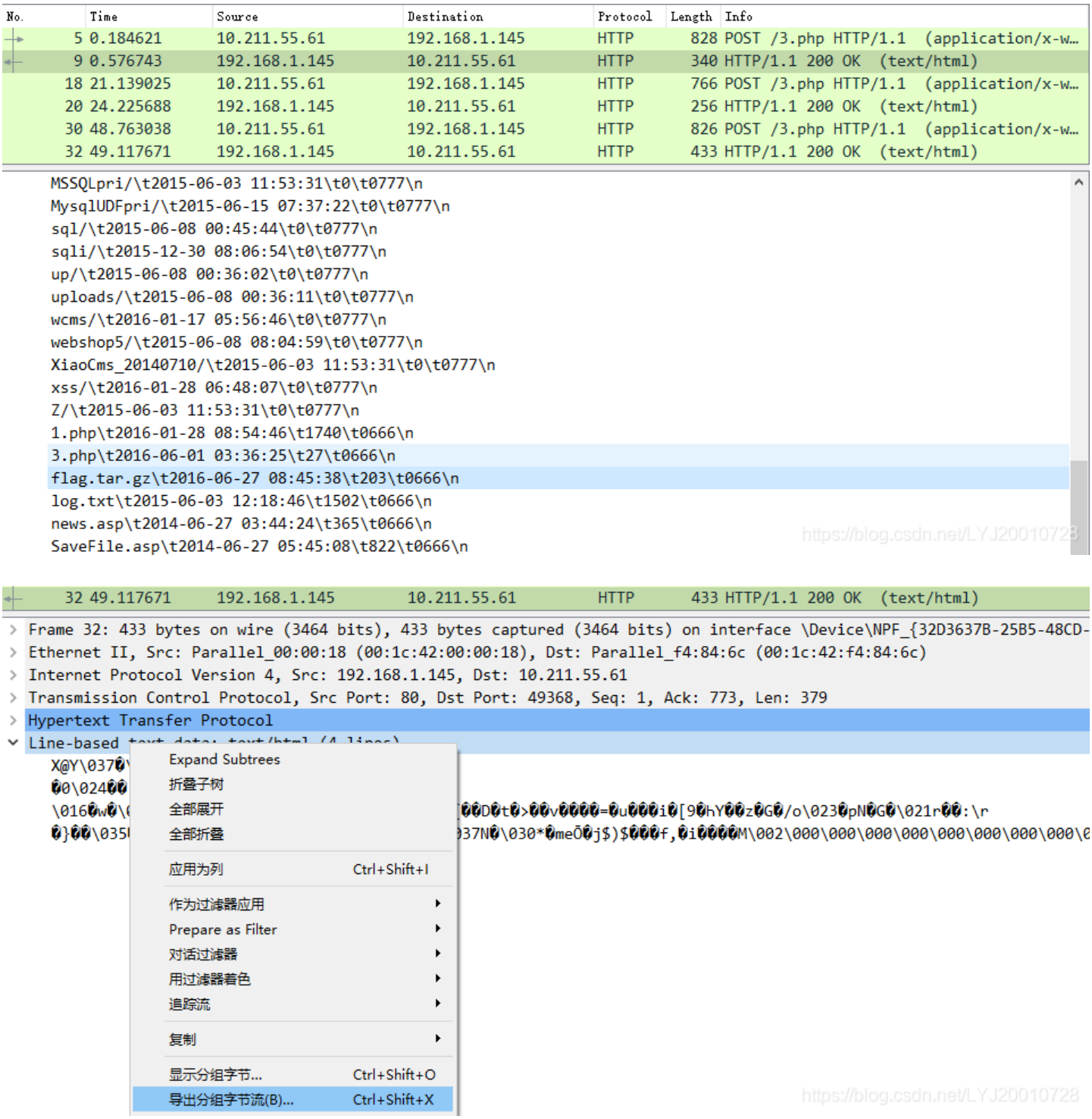
题目描述：有人偷偷下载了文件！

题目题解：
根据题意可能数据包中存在文件传输，尝试直接导出，选择File（文件）–>Export Objexts（导出对象），然后可以看到一些协议，比如选中http就可以看到通过http传输的一些文件，在右下角有导出按钮，可生生成相应的文件。但是本题中无法用此方法直接看到被下载的文件，因为有些文件是直接通过tcp或udp协议传输的，http协议只能看到的访问的链接，但不会看到传输的内容（比如你去访问放一个链接download.php?file=test.rar，通过上述导出对象的方式看不出来下载的文件的内容的）



No.	Time	Source	Destination	Protocol	Length	Info
5	0.184621	10.211.55.61	192.168.1.145	HTTP	828	POST /3.php HTTP/1.1 (application/x-w…
9	0.576743	192.168.1.145	10.211.55.61	HTTP	340	HTTP/1.1 200 OK (text/html)
18	21.139025	10.211.55.61	192.168.1.145	HTTP	766	POST /3.php HTTP/1.1 (application/x-w…
20	24.225688	192.168.1.145	10.211.55.61	HTTP	256	HTTP/1.1 200 OK (text/html)
30	48.763038	10.211.55.61	192.168.1.145	HTTP	826	POST /3.php HTTP/1.1 (application/x-w…
32	49.117671	192.168.1.145	10.211.55.61	HTTP	433	HTTP/1.1 200 OK (text/html)

这个时候就需要找到那个执行下载的数据包，找到数据传输的部分再导出，比如下面这个数据包
大概是一个菜刀下载的过程，在最后一个包可以看到下载的文件，直接右键点击“导出分组字节流”，然后保存为.tar.gz文件



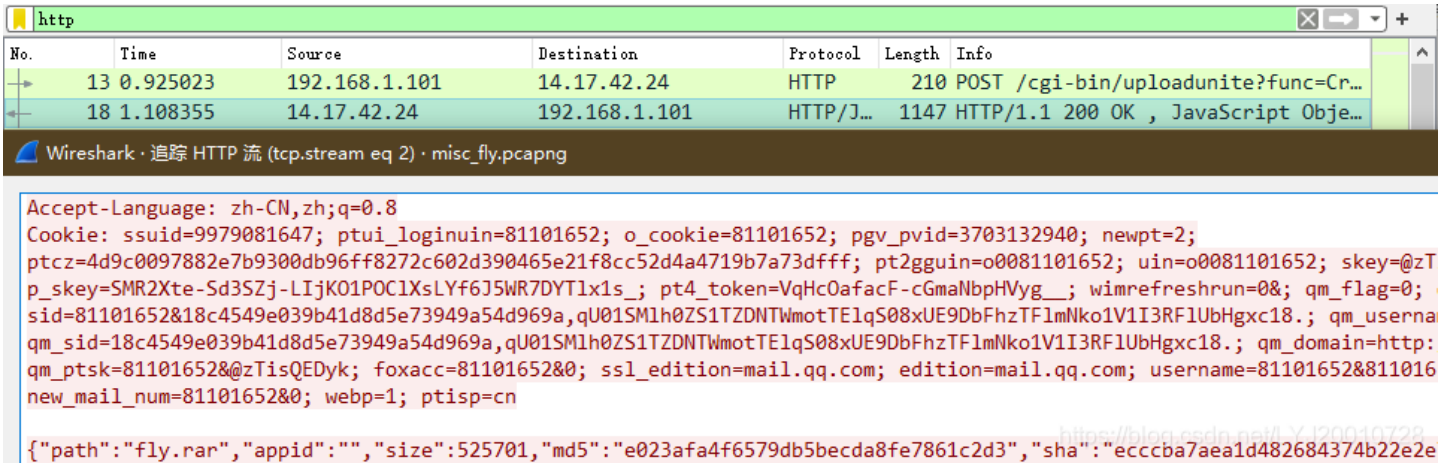
本题中最后还要使用16进制编辑器去除开头和结尾的X@Y字符，这个是菜刀的特征符号，不是文件内容

例题二

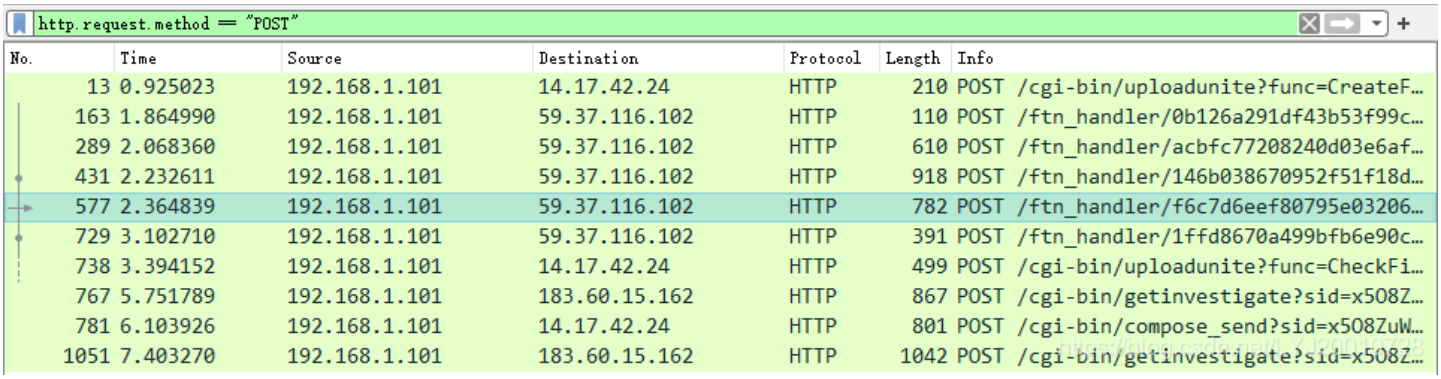
题目文件：misc_fly.p.capng

题目描述：抓到一只苍蝇！

题目题解：
首先用HTTP条件过滤一下；
右键第一个包，追踪流；



可以看到一些基本信息，首先这是一个POST数据包，发送了一些文件相关信息，包括名称（fly.rar）和大小（525701）等。接下来应该就是文件实际上传的数据包，将过滤条件改为：http.request.method==“POST”；



从数据包的结构上看应该就是第二至第六个数据包是数据传输的过程。点开第二个可以看到MediaType的长度为131436=；

```
> [96 Reassembled TCP Segments (132317 bytes): #38(881), #39(1440), #41(1440), #42(1440), #43(1440), #44(1440), #46(144
> Hypertext Transfer Protocol
v Data (131436 bytes)
    Data: abcd9876000003ef000000000002015c01307cb0f705d9067792b5023cb028ba442e6533…
    [Length: 131436]
```

第二到第五个都是一样的长度，第六个为1777，应该是剩余的最后一部分数据。但是131436*4+1777=527521！=525701，再看下第一个数据包；

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000030  C4 06 0B 3E 75 6C 14 21 2E F3 7E 7F 87 6F 7C 81  Ä..>ul.!.ó~.‡o|.
00000040  5A F7 FC 04 63 DF 85 F9 21 A6 3E FB E4 B4 F1 4E  Z÷ü.cß…ù!¦>ûä´ñN
00000050  E1 21 93 DE 81 CB 69 BE 1B 8B 4B 50 10 3F 68 AF  á!"Þ.Ëi¾.‹KP.?h¯
00000060  BD 39 58 A0 DF 19 BC 39 FB 52 F4 35 FD 4E 9D D0  ½9X ß.¼9ûRô5ýN.Ð
00000070  A6 70 53 02 A2 5E F7 10 09 B8 73 48 94 7F 07 2E  ¦pS.¢^÷..¸sH"...
00000080  8A 91 A7 48 8F B6 2B A9 2C C4 9C 4D 50 8E A4 87  Š‘§H.¶+©,ÄœMPŽ¤‡
00000090  03 0D 7A 0D C0 6E A1 17 C4 20 41 FC 55 0A 1F 3B  ..z.Àn¡.Ä AüU..;
000000A0  C5 D2 8D 63 20 65 59 FA F2 A3 BA A7 B9 FC 88 50  ÅÒ.c eYúò£º§¹ü^P
000000B0  C1 2A 49 86 A7 B2 81 92 D0 78 56 DE 86 AB 51 F3  Á*I†§².'ÐxVÞ†«Qó
000000C0  1E 23 F9 E9 C8 3D 26 26 E0 9F 38 0B F9 6E 60 DD  .#ùéÈ=&&à8.ùn`Ý
000000D0  F2 39 2A 84 43 C7 88 DF D2 C8 CC 76 63 28 89 04  ò9*„CÇˆßÒÈÌvc(‰.
000000E0  17 ED 1D E7 AA AA 57 7B CB 9D 1C FB FA 1E FF 76  .í.çªªW{Ë..ûú.ÿv
000000F0  39 AC E4 4B BB 32 B8 B0 DB D9 A3 5B 06 07 47 FD  9¬äK»2¸°ÛÙ£[..Gý
00000100  5D 10 79 C0 CA 5C 9D 37 1F E3 50 B8 F4 3C 7B 7A  ].yÀÊ\.7.ãP¸ô<{z
00000110  8E 07 FA 1F 80 03 32 68 69 76 8E FA 1E 77 02 1E  Ž.ú.€.2hivŽú.w..
00000120  2D 88 11 F0 FC 24 D7 13 3D 48 F3 30 1B A1 57 40  -ˆ.ðü$×.=Hó0.¡W@
00000130  3B 8B BA B9 12 64 C5 C1 E3 E4 60 07 CB 4D 3E 8A  ;‹º¹.dÅÁãä`.ËM>Š
00000140  57 35 00 14 EC CC BA 7A EA 1D 48 26 84 37 4B 22  W5..ìÌºzê.H&„7K"
00000150  E2 E7 AB AD 2B A8 67 49 00 08 05 85 00 00 00 00  âç«.+¨gI.....….....
00000160  00 02 00 00 00 00 00 00 00 00 00 00 52 61 72 21  ............Rar!
```

都知道rar文件头应该是Rar，但是选中的数据部分前面却多出了很多，简单计算一下一共多出了364，且364*5+525701=527521，所以多出的也许是某种校验数据，在导出的时候将其忽略；

每个包都做同样的操作即可得出5个文件，再将这个文件按顺序拼接即可。拼接的话可以使用16进制编辑器手动拼接，也可以使用linux下cat命令，比如“cat 1 2 3 4 5 > fly.rar”；’这道题还设置了伪加密，需要修改加密位，将0x84位置改为0x80即可；

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  52 61 72 21 1A 07 00 CF 90 73 00 00 0D 00 00 00  Rar!...Ï.s......
00000010  00 00 00 00 F9 81 74 80 90 2D 00 3D 05 08 00 86  ....ù.t€.-.=...†
00000020  22 0F 00 02 25 2D ED F1 09 5C 59 46 1D 35 08 00  "...%-íñ.\YF.5..
00000030  20 00 00 00 66 6C 61 67 2E 74 78 74 00 F0 79 03   ...flag.txt.ðy.
00000040  4C 18 1E 15 15 0C 89 15 DC 16 1D EF A3 72 4B 90  L.....‰.Ü..ï£rK.
00000050  B0 90 08 24 3A 52 23 05 22 02 C8 41 C4 84 82 40  °..$:R#."ÈAÄ„‚@
00000060  9D 04 3A EC 24 87 44 3A 58 A1 18 08 81 92 6F 60  ..:ì$‡D:X¡...'o`
```

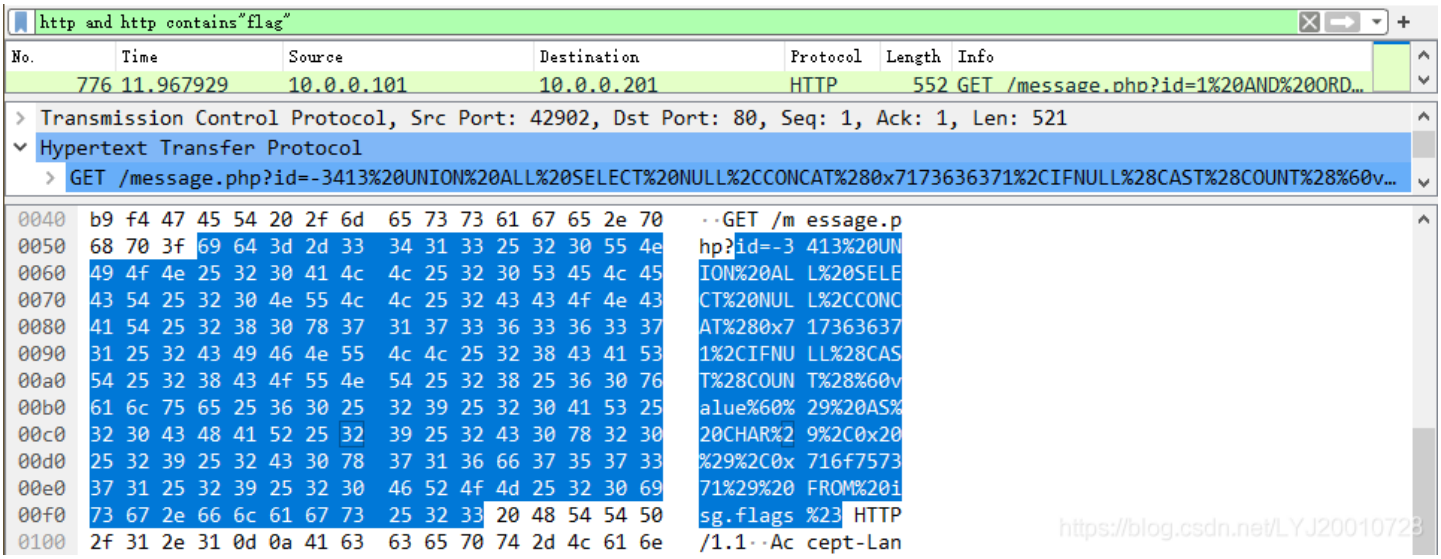
解压出来后是一个exe可执行文件，里面隐藏了一个png图片，是个二维码，扫描即可得到flag

信息提取

> 题目文件： sqlmap.pcap

> 题目描述：数据库中的flag被偷走了，好在全过程我们都有记录

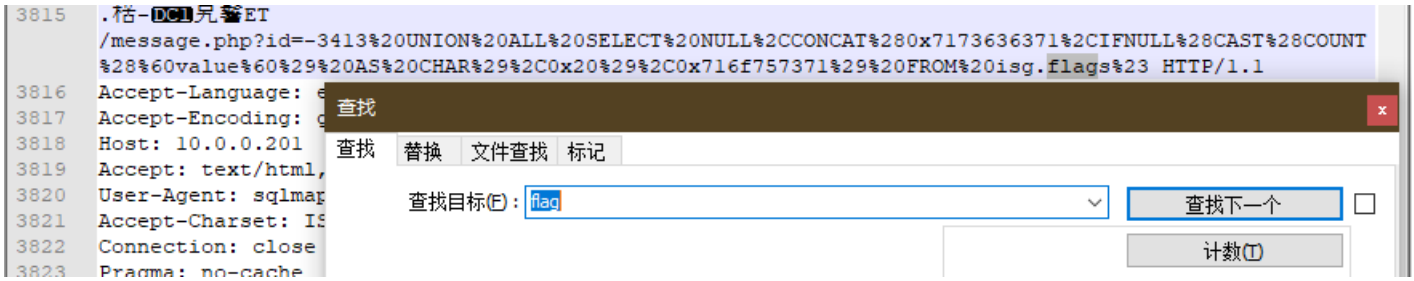
> 题目题解：
> 数据包记录的是sqlmap获取flag的过程，使用http && http contains"flag"过滤一下



将其payload解码一下是这样的，判断其ascii码是否大于64

id=1 AND ORD(MID((SELECTIFNULL(CAST(`value` AS CHAR),0x20) FROM isg.flags ORDER BY `value` LIMIT0,1),1,1))>64

然后一直到836个包判断第一位ascii码值大于72，然后开始从高到低递减，判断其ascii码不大于73，则第一位的ascii码值是73，对应的字符为I。以此类推，其flag为ISG{BLind_SQl_InJEcTi0N_DeTEcTEd}。本题需要一定的耐心和SQL注入基础。但是这么做可能有些繁琐，其实pcap数据包可以直接用文本编辑器打开，就可以看到其中的http请求



所以可以使用字符串搜索的方式直接去查找其中的语句，然后判断flag，首先将原数据包中的http请求导出来，另存为sqli.pcap