CTFHUB 流量分析

原创

pain 于 2021-02-09 22:22:31 发布 546 收藏

分类专栏： web做题记录

本文链接：https://blog.csdn.net/qq_51558360/article/details/113775981

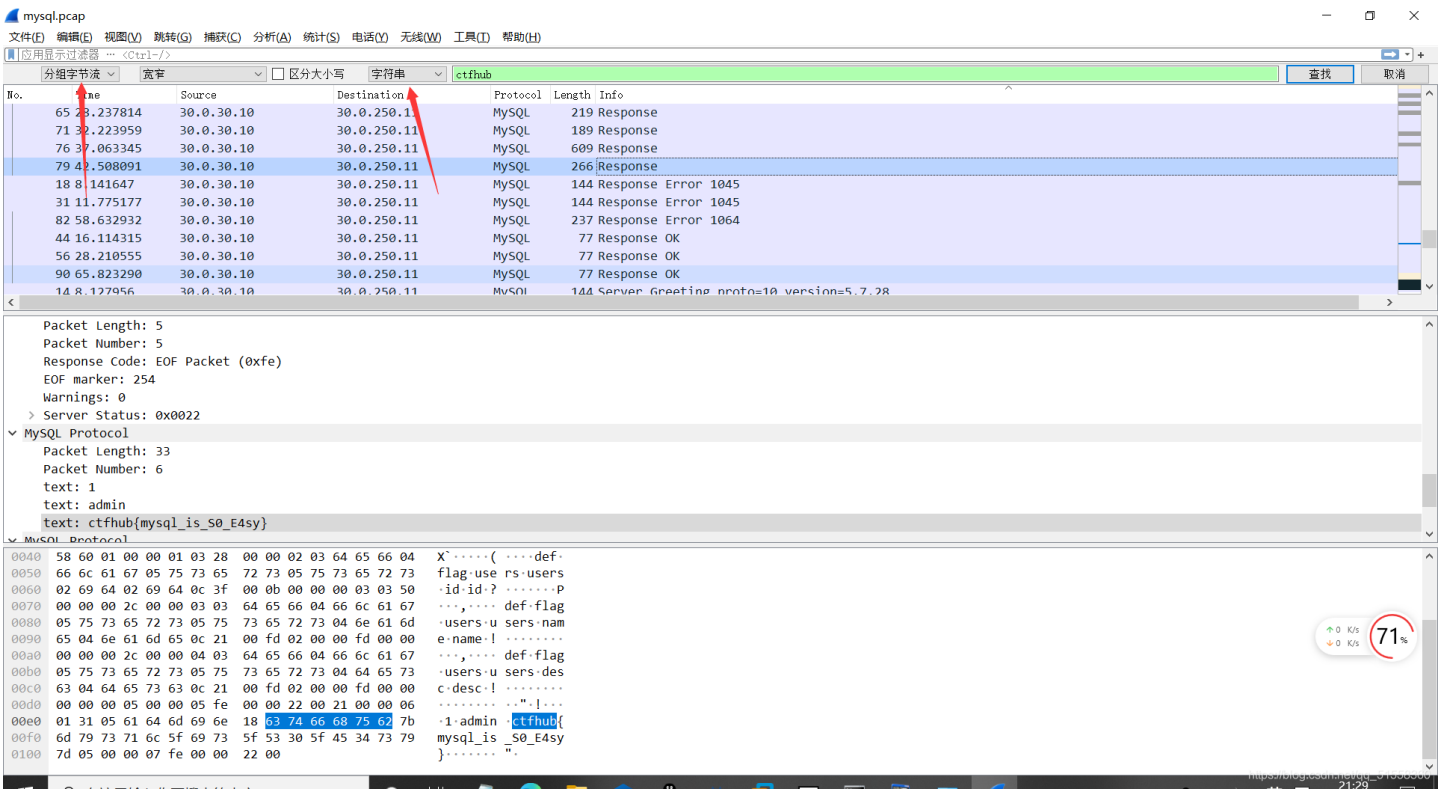
web做题记录 专栏收录该内容

24 篇文章 0 订阅

订阅专栏

MySQL流量

将附件下载之后用wireshark打开过滤搜索ctfhub



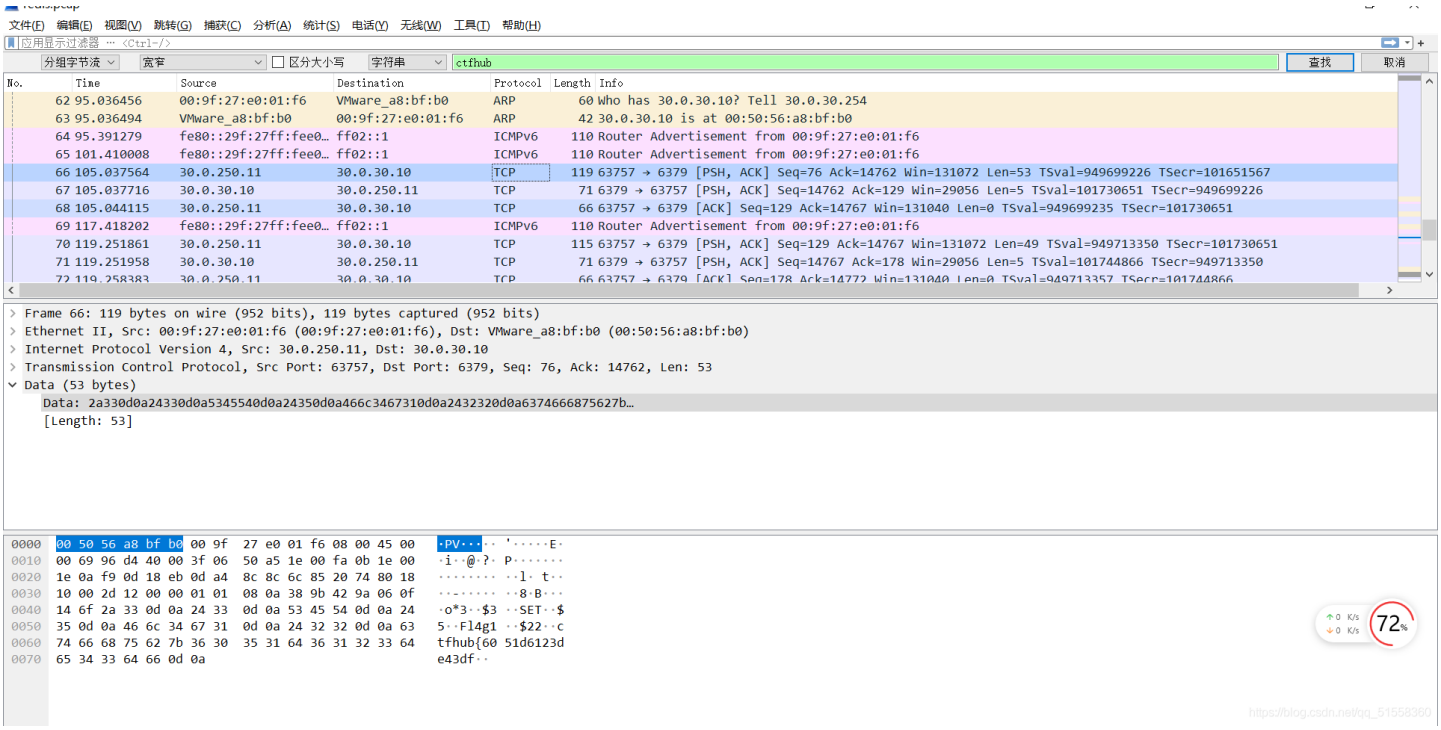
Redis流量

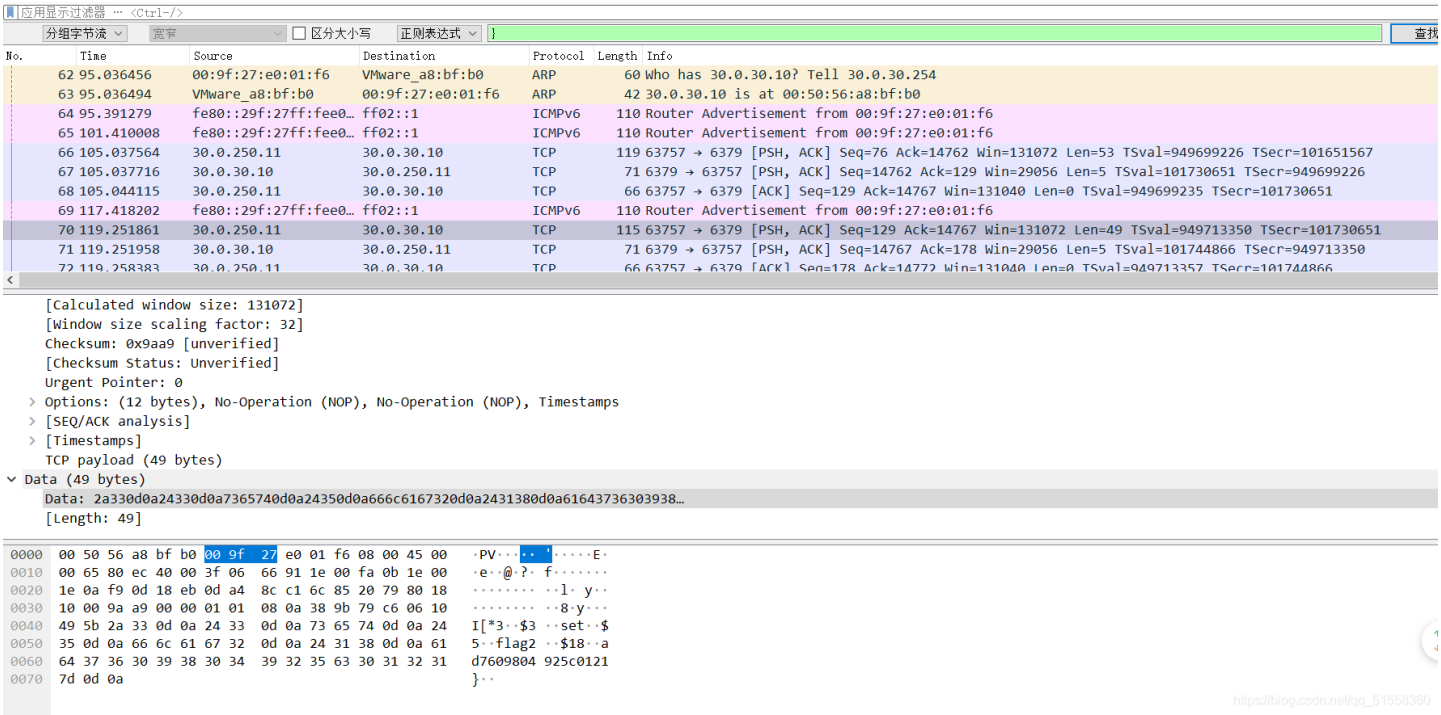
Redis:

- nosql数据库，非关系型数据库
- 支持5大数据类型 (字符串String，列表list、字典hash，集合set，zset)
  - 与之相似的有memcache，但memcache只支持string类型
- 单进程单线程，好处在于不用考虑并发

将附件下载之后用wireshark打开过滤搜索ctfhub

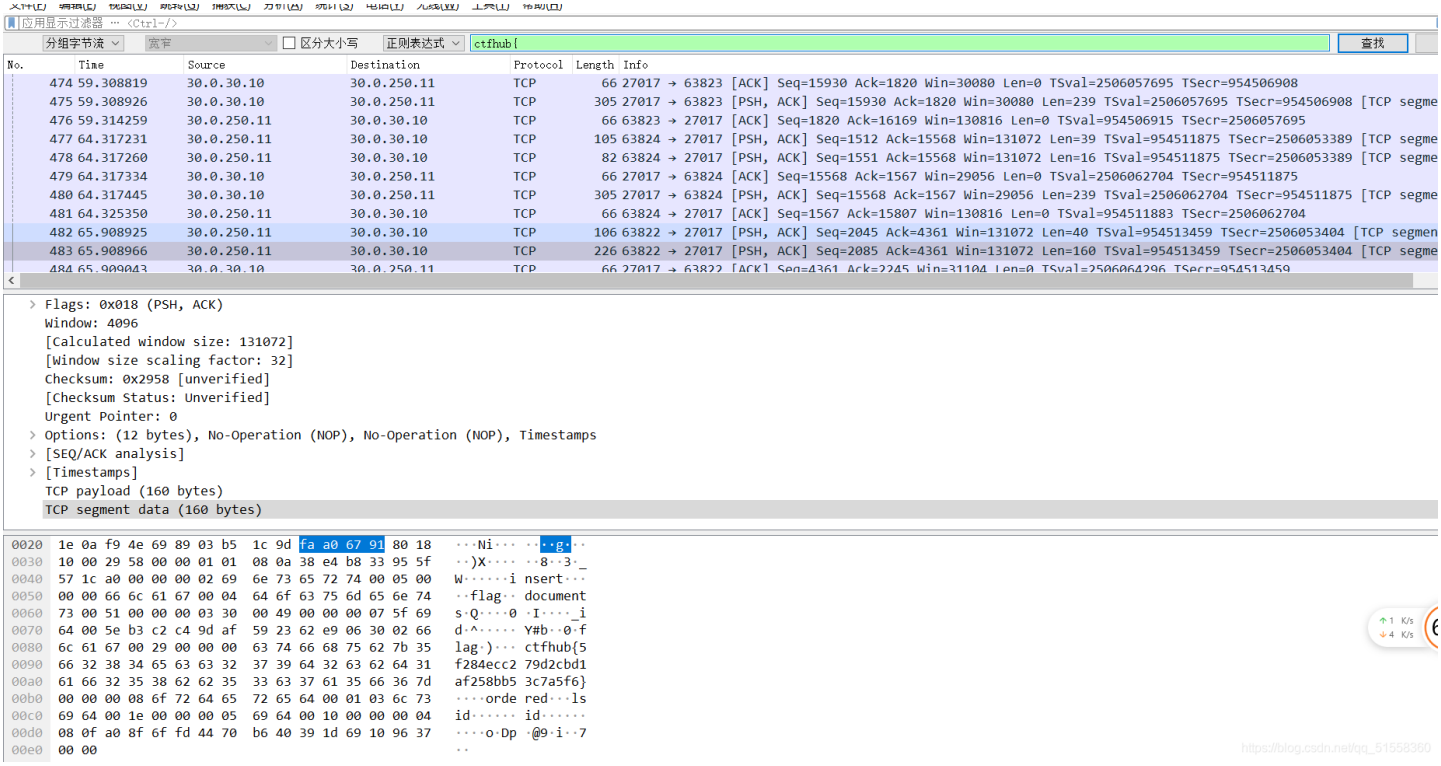
可是我们发现这个flag只有一半所以我们接着找





ctfhub{6051d6123de43dfad7609804925c0121}

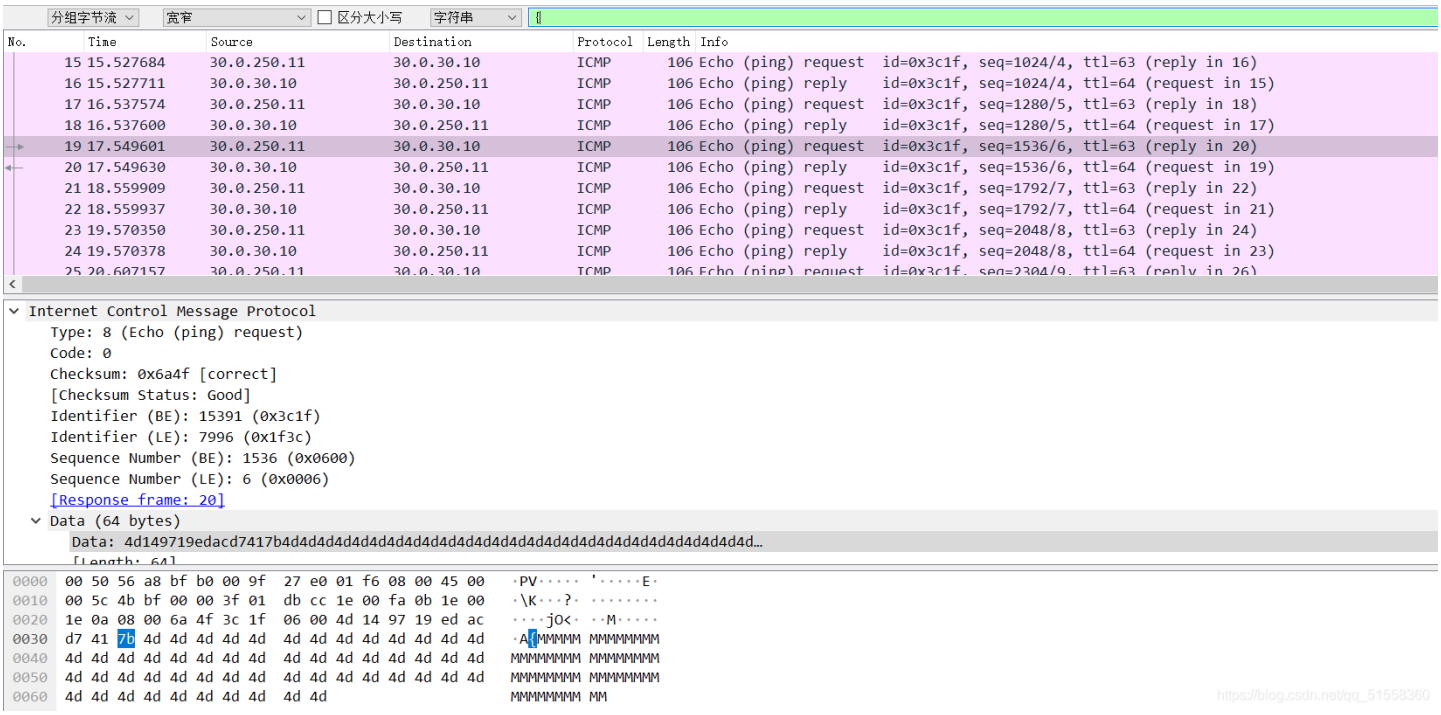
MongoDB流量


应用显示过滤器 … <Ctrl-/>
分组字节流
宽窄
区分大小写
正则表达式
ctfhub{
查找
No. Time Source Destination Protocol Length Info
474 59.308819 30.0.30.10 30.0.250.11 TCP 66 27017 → 63823 [ACK] Seq=15930 Ack=1820 Win=30080 Len=0 TSval=2506057695 TSecr=954506908
475 59.308926 30.0.30.10 30.0.250.11 TCP 305 27017 → 63823 [PSH, ACK] Seq=15930 Ack=1820 Win=30080 Len=239 TSval=2506057695 TSecr=954506908 [TCP segme
476 59.314259 30.0.250.11 30.0.30.10 TCP 66 63823 → 27017 [ACK] Seq=1820 Ack=16169 Win=130816 Len=0 TSval=954506915 TSecr=2506057695
477 64.317231 30.0.250.11 30.0.30.10 TCP 105 63824 → 27017 [PSH, ACK] Seq=1512 Ack=15568 Win=131072 Len=39 TSval=954511875 TSecr=2506053389 [TCP segme
478 64.317260 30.0.250.11 30.0.30.10 TCP 82 63824 → 27017 [PSH, ACK] Seq=1551 Ack=15568 Win=131072 Len=16 TSval=954511875 TSecr=2506053389 [TCP segme
479 64.317334 30.0.30.10 30.0.250.11 TCP 66 27017 → 63824 [ACK] Seq=15568 Ack=1567 Win=29056 Len=0 TSval=2506062704 TSecr=954511875
480 64.317445 30.0.30.10 30.0.250.11 TCP 305 27017 → 63824 [PSH, ACK] Seq=15568 Ack=1567 Win=29056 Len=239 TSval=2506062704 TSecr=954511875 [TCP segme
481 64.325350 30.0.250.11 30.0.30.10 TCP 66 63824 → 27017 [ACK] Seq=1567 Ack=15807 Win=130816 Len=0 TSval=954511883 TSecr=2506062704
482 65.908925 30.0.250.11 30.0.30.10 TCP 106 63822 → 27017 [PSH, ACK] Seq=2045 Ack=4361 Win=131072 Len=40 TSval=954513459 TSecr=2506053404 [TCP segmen
483 65.908966 30.0.250.11 30.0.30.10 TCP 226 63822 → 27017 [PSH, ACK] Seq=2085 Ack=4361 Win=131072 Len=160 TSval=954513459 TSecr=2506053404 [TCP segme
484 65.909043 30.0.30.10 30.0.250.11 TCP 66 27017 → 63822 [ACK] Seq=4361 Ack=2245 Win=31104 Len=0 TSval=2506064296 TSecr=954513459
Flags: 0x018 (PSH, ACK)
Window: 4096
[Calculated window size: 131072]
[Window size scaling factor: 32]
Checksum: 0x2958 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
[SEQ/ACK analysis]
[Timestamps]
TCP payload (160 bytes)
TCP segment data (160 bytes)
0020 1e 0a f9 4e 69 89 03 b5 1c 9d fa a0 67 91 80 18 ···Ni··· ····g···
0030 10 00 29 58 00 00 01 01 08 0a 38 e4 b8 33 95 5f ··)X···· ··8··3·_
0040 57 1c a0 00 00 00 02 69 6e 73 65 72 74 00 05 00 W·····i nsert···
0050 00 00 66 6c 61 67 00 04 64 6f 63 75 6d 65 6e 74 ··flag·· document
0060 73 00 51 00 00 00 03 30 00 49 00 00 00 07 5f 69 s·Q····0 ·I····_i
0070 64 00 5e b3 c2 c4 9d af 59 23 62 e9 06 30 02 66 d·^····· Y#b··0·f
0080 6c 61 67 00 29 00 00 00 63 74 66 68 75 62 7b 35 lag·)··· ctfhub{5
0090 66 32 38 34 65 63 63 32 37 39 64 32 63 62 64 31 f284ecc2 79d2cbd1
00a0 61 66 32 35 38 62 62 35 33 63 37 61 35 66 36 7d af258bb5 3c7a5f6}
00b0 00 00 00 08 6f 72 64 65 72 65 64 00 01 03 6c 73 ····orde red···ls
00c0 69 64 00 1e 00 00 00 05 69 64 00 10 00 00 00 04 id····· id·····
00d0 08 0f a0 8f 6f fd 44 70 b6 40 39 1d 69 10 96 37 ····o·Dp ·@9·i··7
00e0 00 00 ··
1 K/s
4 K/s
https://blog.csdn.net/qq_51558360

Data

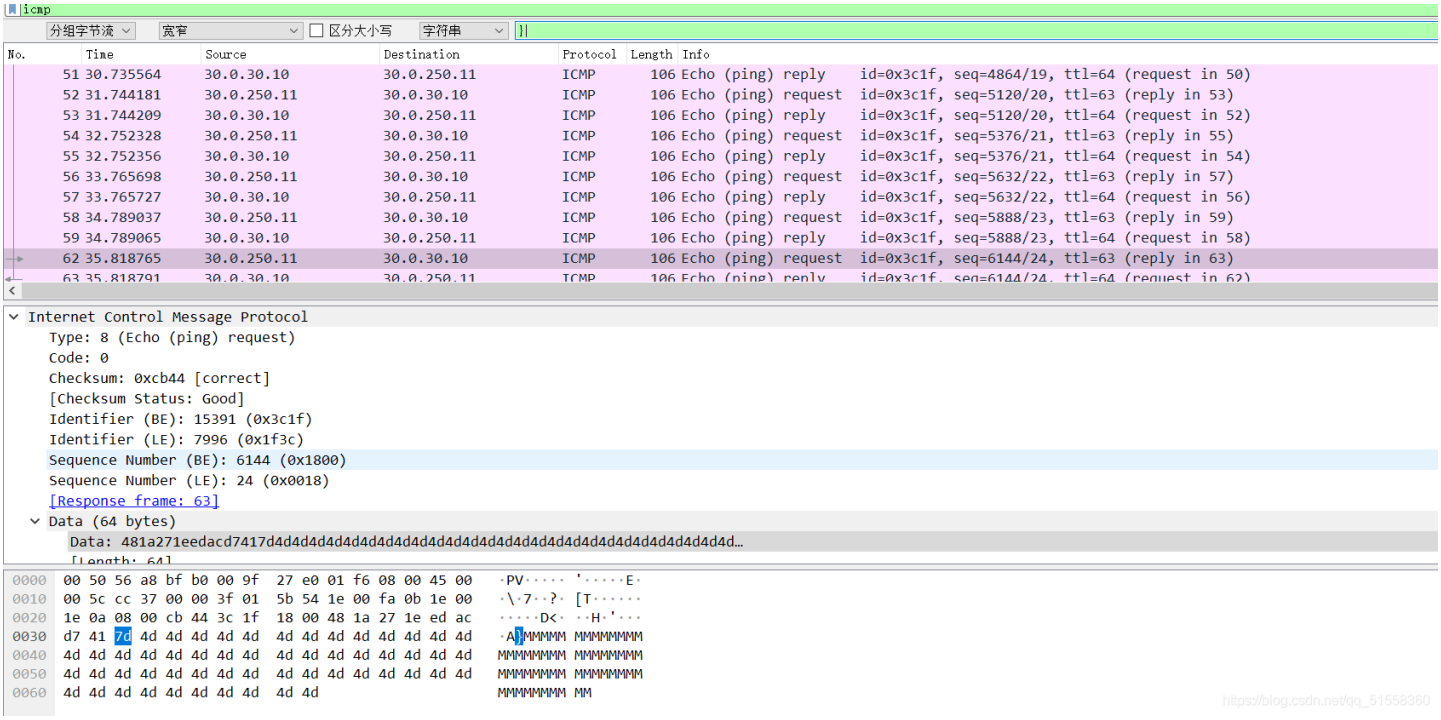
过滤icmp包

观察




```
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x6a4f [correct]
    [Checksum Status: Good]
    Identifier (BE): 15391 (0x3c1f)
    Identifier (LE): 7996 (0x1f3c)
    Sequence Number (BE): 1536 (0x0600)
    Sequence Number (LE): 6 (0x0006)
    [Response frame: 20]
  Data (64 bytes)
      Data: 4d149719edacd7417b4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d4d…
      [Length: 64]
```

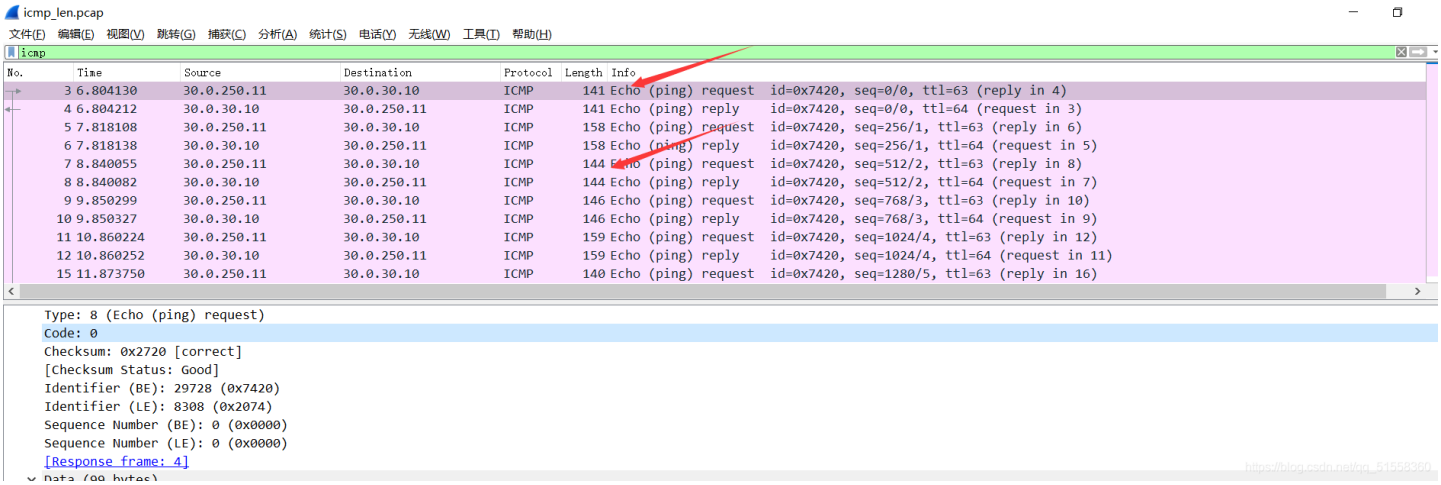
```
0000  00 50 56 a8 bf b0 00 9f  27 e0 01 f6 08 00 45 00   ·PV····· '·····E·
0010  00 5c 4b bf 00 00 3f 01  db cc 1e 00 fa 0b 1e 00   ·\K···?· ········
0020  1e 0a 08 00 6a 4f 3c 1f  06 00 4d 14 97 19 ed ac   ····jO<· ··M·····
0030  d7 41 7b 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d   ·A{MMMMM MMMMMMMM
0040  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d   MMMMMMMM MMMMMMMM
0050  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d   MMMMMMMM MMMMMMMM
0060  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d                     MMMMMMMM MM
```



从上面那个包开始有“{”，到第二张的包后又有”}“，猜测可能是”{“
到”}“之间相对应的位置的字母数字再加上ctfhub可能就是flag
可以发现有重复情况
所以再观察记录一下得
ctfhub{c87eb99796406ac0b}

Length

首先通过关键字过滤ICMP
之后，从题目Length猜出可能与长度有关，我们知道这个平台默认flag开头是ctfhub{}，发现c与f相差3个ascii码值，查看数据包也是一样的



上面因为重复了，就当没看见吧
将所有ICMP数据包的Length字段取出来，转换为ASCII码值，即可得到flag
99 116 102 104 117 98 123 97 99 98 54 53 57 102 48 50 51 125
ctfhub{acb659f023}