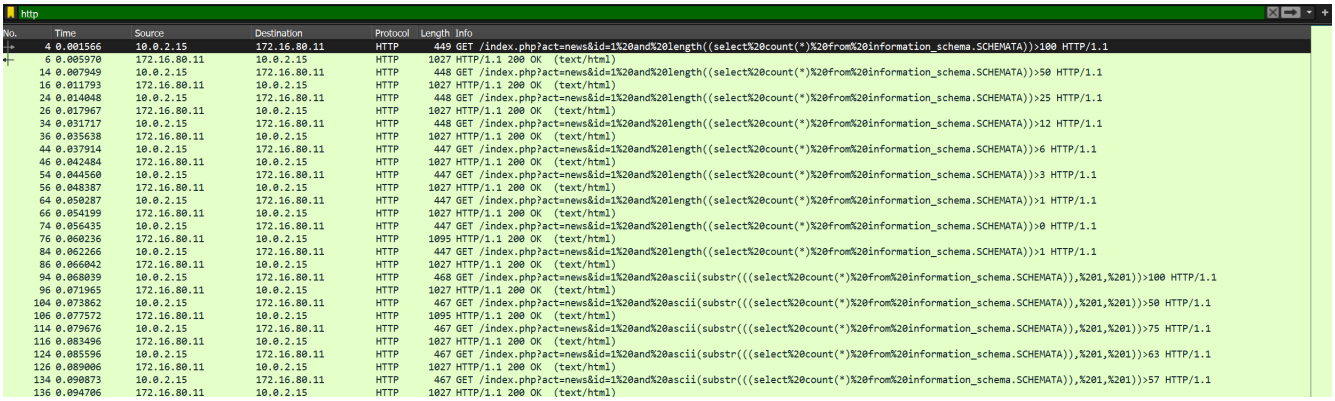
misc-lab3

1 Challenge 1

用 `WireShark` 提高流量包，先筛选 `http` 进行观察，发现类似sql盲注的语句



No.	Time	Source	Destination	Protocol	Length	Info
4	0.001566	10.0.2.15	172.16.80.11	HTTP	449	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>100 HTTP/1.1
6	0.005970	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
14	0.007949	10.0.2.15	172.16.80.11	HTTP	448	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>50 HTTP/1.1
16	0.011793	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
24	0.014048	10.0.2.15	172.16.80.11	HTTP	448	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>25 HTTP/1.1
26	0.017967	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
34	0.031717	10.0.2.15	172.16.80.11	HTTP	448	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>12 HTTP/1.1
36	0.035638	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
44	0.037914	10.0.2.15	172.16.80.11	HTTP	447	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>6 HTTP/1.1
46	0.042484	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
54	0.044560	10.0.2.15	172.16.80.11	HTTP	447	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>3 HTTP/1.1
56	0.048387	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
64	0.050287	10.0.2.15	172.16.80.11	HTTP	447	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>1 HTTP/1.1
66	0.054199	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
74	0.056435	10.0.2.15	172.16.80.11	HTTP	447	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>0 HTTP/1.1
76	0.060236	172.16.80.11	10.0.2.15	HTTP	1095	HTTP/1.1 200 OK (text/html)
84	0.062266	10.0.2.15	172.16.80.11	HTTP	447	GET /index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>1 HTTP/1.1
86	0.066042	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
94	0.068039	10.0.2.15	172.16.80.11	HTTP	468	GET /index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>100 HTTP/1.1
96	0.071965	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
104	0.073862	10.0.2.15	172.16.80.11	HTTP	467	GET /index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>50 HTTP/1.1
106	0.077572	172.16.80.11	10.0.2.15	HTTP	1095	HTTP/1.1 200 OK (text/html)
114	0.079676	10.0.2.15	172.16.80.11	HTTP	467	GET /index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>75 HTTP/1.1
116	0.083496	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
124	0.085596	10.0.2.15	172.16.80.11	HTTP	467	GET /index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>63 HTTP/1.1
126	0.089006	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)
134	0.090873	10.0.2.15	172.16.80.11	HTTP	467	GET /index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>57 HTTP/1.1
136	0.094706	172.16.80.11	10.0.2.15	HTTP	1027	HTTP/1.1 200 OK (text/html)

于是利用 `tshark` 提取 `http` 请求到 `data.txt` 中

```
tshark -r sqltest.pcapng -Y "http.request" -T fields -e http.request.full_uri > data.txt
```

```
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>100
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>50
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>25
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>12
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>6
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>3
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>1
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>0
http://172.16.80.11/index.php?act=news&id=1%20and%20length((select%20count(*)%20from%20information_schema.SCHEMATA))>1
http://172.16.80.11/index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>100
http://172.16.80.11/index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>50
http://172.16.80.11/index.php?act=news&id=1%20and%20ascii(substr(((select%20count(*)%20from%20information_schema.SCHEMATA)),%201,%201))>75
```

提取到的 `url` 如下，由于是get,在 `?` 后面的语句，就是sql的查询语句。根据sql的查询的知识，在每个查询位的最后一个查询项，就是条件中止的条件，即正确的字符ascii码，所以提取最后提取 `flag` 数据项进行盲注的语句，在txt文件中发现是628-971行，其中628-711是前9位，剩下是第十位之后（两位数）

因此用python处理 `txt` 文件，写常规的读取和写入的操作

```
def read_file(input_file,start_line,end_line):
    with open(input_file,'r',encoding='utf-16') as input:
        alllines=input.readlines()
        lines=alllines[start_line:end_line]
    print("readed {} to {}".format(start_line,end_line))
    return lines
def write_file(lines,output_file):
    with open(output_file,'w',encoding='utf-16') as output:
        output.writelines(lines)
    print("written")
```

然后通过行号分别读取1-9的数据，10位之后的数据，放在不同的文件里：

```
start_line=627
end_line=711
lines=read_file('D:/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3/data.txt',start_line,end_line)

start_line=711
end_line=972
lines=read_file('D:/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3/data.txt',start_line,end_line)
```

为了搞清楚查询的最后一位，提取查询位前后的差别，

```
In [2]: s='http://172.16.80.11/index.php?act=news&id=1%20and%20ascii(substr(((select%20concat_ws(char(94),%20flag)%20%2
   ...: 0from%20db_flag.tb_flag%20%20limit%200,1)),%20'

In [3]: len(s)
Out[3]: 154
```

在查询位前一共有154个字符，提取 `line[154]` 和之前进行比较，并且留下 `pre_line` 的数据，假设现在的 `line[154]` 开启了新的一位，就取之前的 `pre_line` 的最后一次比较的 `ascii` 码
这样会漏下最后的第九位，需要手动加上去（）十位之后的逻辑也是类似的（）
最后 `flag+=chr(int(pre_line[164:]))` 就能依次连缀 `flag`，获得flag值了

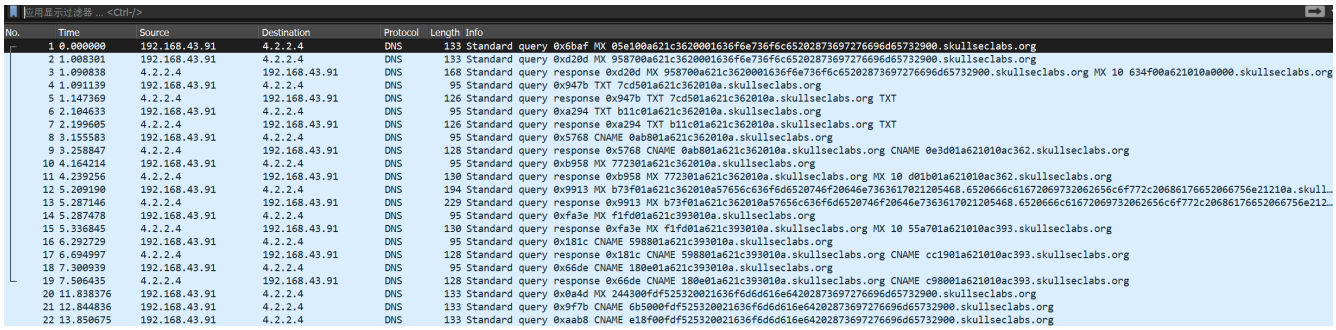
最后的输出如下：

```
PS C:\Users\leexi> & C:/Users/leexi/AppData/Loca
readed 627 to 711
readed 711 to 972
dflag{47edb8300ed5f9b28fc54b0d09ecdef7}
```

得到了 `flag{47edb8300ed5f9b28fc54b0d09ecdef7}`

2 Challenge2

先用 `Wireshark` 打开一下，发现大部分都是DNS协议的流量，是基于UDP协议的，传输的内容应该在频繁出现的 `skullscaps.org` 的前面。



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.43.91	4.2.2.4	DNS	133	Standard query 0x6baf MX 05e100a621c3620001636f6e736f6c6520287369727669 6d65732900.skullseclabs.org
2	1.008301	192.168.43.91	4.2.2.4	DNS	133	Standard query 0xd20d MX 958700a621c3620001636f6e736f6c6520287369727669 6d65732900.skullseclabs.org
3	1.090838	4.2.2.4	192.168.43.91	DNS	168	Standard query response 0xd20d MX 958700a621c3620001636f6e736f6c652028736 9727669 6d65732900.skullseclabs.org MX 10 634f00a621010a0000.skullseclabs.org
4	1.091139	192.168.43.91	4.2.2.4	DNS	95	Standard query 0x947b TXT 7cd501a621c362010a.skullseclabs.org
5	1.147369	4.2.2.4	192.168.43.91	DNS	126	Standard query response 0x947b TXT 7cd501a621c362010a.skullseclabs.org TXT
6	2.104633	192.168.43.91	4.2.2.4	DNS	95	Standard query 0xa294 TXT b11c01a621c362010a.skullseclabs.org
7	2.199605	4.2.2.4	192.168.43.91	DNS	126	Standard query response 0xa294 TXT b11c01a621c362010a.skullseclabs.org TXT
8	3.155583	192.168.43.91	4.2.2.4	DNS	95	Standard query 0x5768 CNAME 0ab801a621c362010a.skullseclabs.org
9	3.258847	4.2.2.4	192.168.43.91	DNS	128	Standard query response 0x5768 CNAME 0ab801a621c362010a.skullseclabs.org CNAME 0e3d01a621010ac362.skullseclabs.org
10	4.164214	192.168.43.91	4.2.2.4	DNS	95	Standard query 0xb958 MX 772301a621c362010a.skullseclabs.org
11	4.239256	4.2.2.4	192.168.43.91	DNS	130	Standard query response 0xb958 MX 772301a621c362010a.skullseclabs.org MX 10 d01b01a621010ac362.skullseclabs.org
12	5.209190	192.168.43.91	4.2.2.4	DNS	194	Standard query 0x9913 MX b73f01a621c362010a57656c636f6d6520746f20646e736361702120546 8.6520666c6167206973206265 6c6f772c20686176652066756e21210a.skull…
13	5.287146	4.2.2.4	192.168.43.91	DNS	229	Standard query response 0x9913 MX b73f01a621c362010a57656c636f6d6520746f20646e736361702120546 8.6520666c6167206973206265 6c6f772c20686176652066756e212…
14	5.287478	192.168.43.91	4.2.2.4	DNS	95	Standard query 0xfa3e MX f1fd01a621c393010a.skullseclabs.org
15	5.336845	4.2.2.4	192.168.43.91	DNS	130	Standard query response 0xfa3e MX f1fd01a621c393010a.skullseclabs.org MX 10 55a701a621010ac393.skullseclabs.org
16	6.292729	192.168.43.91	4.2.2.4	DNS	95	Standard query 0x181c CNAME 598801a621c393010a.skullseclabs.org
17	6.694997	4.2.2.4	192.168.43.91	DNS	128	Standard query response 0x181c CNAME 598801a621c393010a.skullseclabs.org CNAME cc1901a621010ac393.skullseclabs.org
18	7.300939	192.168.43.91	4.2.2.4	DNS	95	Standard query 0x66de CNAME 180e01a621c393010a.skullseclabs.org
19	7.506435	4.2.2.4	192.168.43.91	DNS	128	Standard query response 0x66de CNAME 180e01a621c393010a.skullseclabs.org CNAME c98001a621010ac393.skullseclabs.org
20	11.838376	192.168.43.91	4.2.2.4	DNS	133	Standard query 0x0a4d MX 244300fdf525320021636f6d6d616e6420287369727669 6d5732900.skullseclabs.org
21	12.844836	192.168.43.91	4.2.2.4	DNS	133	Standard query 0x9f7b CNAME 6b5000fdf525320021636f6d6d616e6420287369727669 6d65732900.skullseclabs.org
22	13.850675	192.168.43.91	4.2.2.4	DNS	133	Standard query 0xaab8 CNAME e18f00fdf525320021636f6d6d616e6420287369727669 6d65732900.skullseclabs.org

因此，首先先用 `tshark` 提取一下协议的内容：

```
PS D:\MyRepository\slowist-notebook\docs\Coding\CTF\misc-lec3> tshark -r dnscap.pcap
-Y "http.request" -T fields -e http.request.full_uri > dnscap.txt
```

再利用正则表达式查看一下传输的内容

```
def hex_to_binary_file(hex_data, output_file):
    byte_data = bytes.fromhex(hex_data)
    with open(output_file, 'wb') as bin_file:
        bin_file.write(byte_data)
    print(f"Data written to {output_file}")
find=''
with open('D:/MyRepository/slowist-notebook/docs/Coding/CTF/misc-
lec3/dnscap.txt','r',encoding='utf-16') as f:
    for i in f:
        text = re.findall(r'([\w\.]+)\.skull',i)
        if text:
            find += text[0].replace('.','')
hex_to_binary_file(find,'D:/MyRepository/slowist-notebook/docs/Coding/CTF/misc-
lec3/question')
```

这个时候用十六进制编辑器打开这个文件，先是看到一个提示信息：

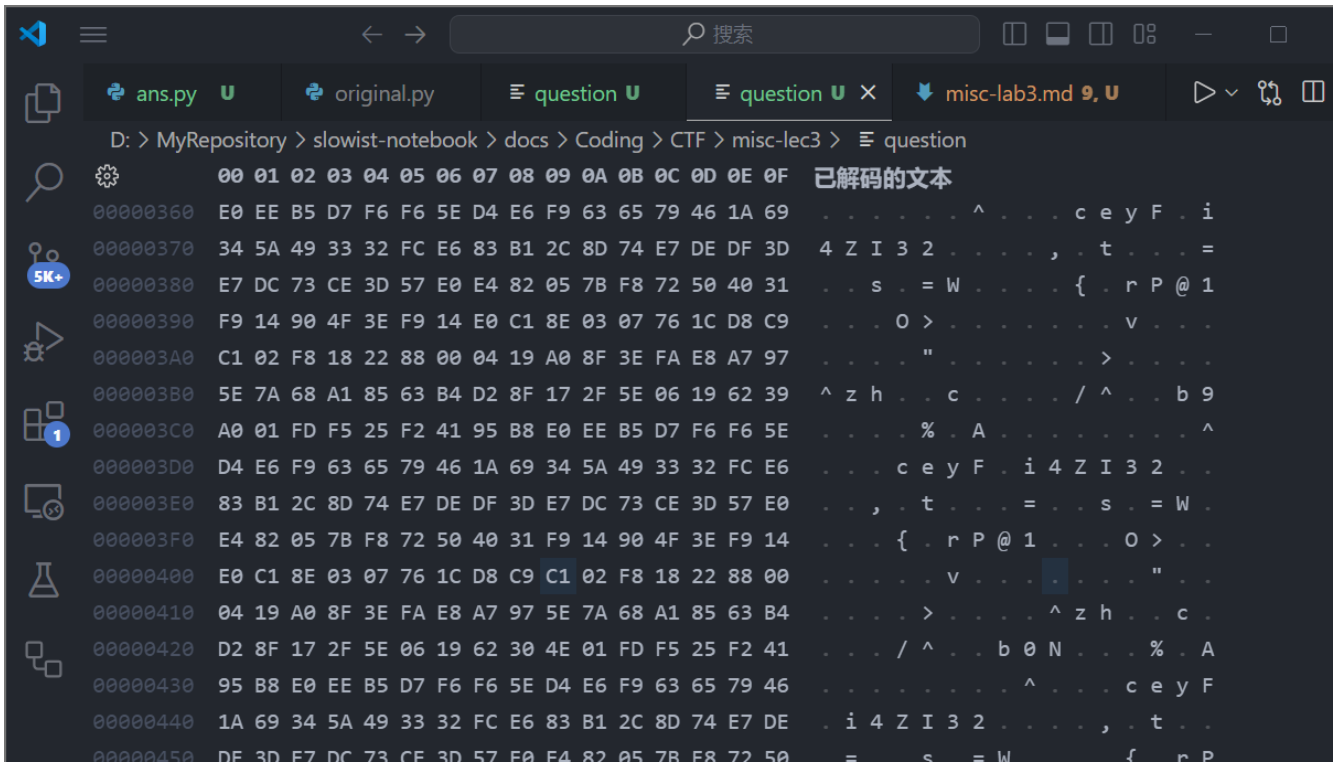
```
21 C3 62 01 0A 57 65 6C 63 6F 6D 65 20 74 6F 20  ! . b . . W e l c o m e   t o
64 6E 73 63 61 70 21 20 54 68 65 20 66 6C 61 67  d n s c a p !   T h e   f l a g
20 69 73 20 62 65 6C 6F 77 2C 20 68 61 76 65 20    i s   b e l o w ,   h a v e
66 75 6E 21 21 0A F1 FD 01 A6 21 C3 93 01 0A 59  f u n ! ! . . . . . ! . . . . Y
88 01 A6 21 C3 93 01 0A 18 0E 01 A6 21 C3 93 01  . . . ! . . . . . ! . . .
```

往下翻，接着看到了 `PNG` 开始的标志：

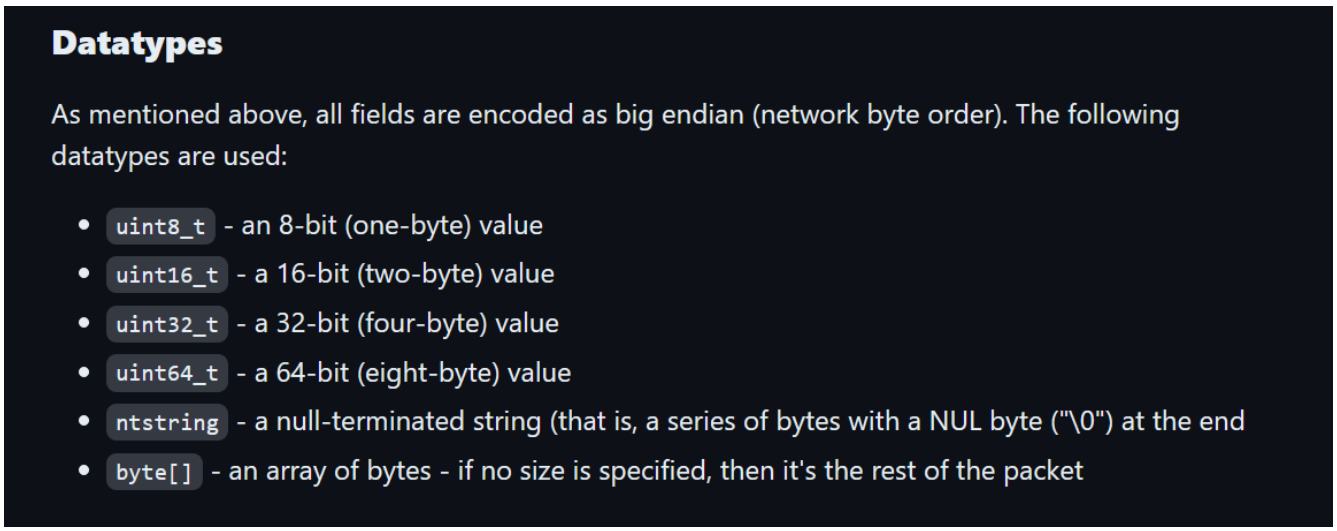
```
C0  01 00 03 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49  . . . . P N G . . . . . . . I
D0  48 44 52 00 00 01 00 00 00 01 00 08 04 00 00 00  H D R . . . . . . . . . . . .
E0  F6 7B 60 ED 00 00 00 04 67 41 4D 41 00 01 86 A0  . { ` . . . . g A M A . . . .
F0  31 E8 96 5F 00 00 00 02 62 4B 47 44 00 FF 87 8F  1 . . _ . . . . b K G D . . . .
00  CC BF 00 00 00 09 70 48 59 73 00 00 0B 13 00 00  . . . . . . p H Y s . . . . . .
10  0B 13 01 00 9A 9C 18 00 00 00 07 CE 22 01 FD F5  . . . . . . . . . . . . " . . .
20  25 92 41 95 74 49 4D 45 07 E1 02 02 05 0D 35 24  % . A . t I M E . . . . . 5 $
30  D3 81 E9 00 00 2C 08 49 44 41 54 78 DA ED 9D 77  . . . . . , . I D A T x . . . w
40  9C 1B D5 D5 F7 BF A3 AE 95 56 DB AB D7 DE 5D 7B  . . . . . V . . . . ] {
```

但是直接改成 `.png` 来看，就会错误

于是再往下翻，就会发现不太对，发现有很多数据块发生重复，比如下面的`i4ZI32`的部分，至少重复了三遍。



翻看dns的知识，发现udp在传输的时候会发生重复，不能单靠连接来构成这个文件，所以还是要先去重，再得到对应的png文件。
主要是看data的结构，真正传输的部分是从后面开始的，我们提取的也应该是data的部分，并且对这部分去重。因此前面也会有很多冗余信息，需要删除，所以取`find.append(tmp[18:])`



Datatypes

As mentioned above, all fields are encoded as big endian (network byte order). The following datatypes are used:

- `uint8_t` - an 8-bit (one-byte) value
- `uint16_t` - a 16-bit (two-byte) value
- `uint32_t` - a 32-bit (four-byte) value
- `uint64_t` - a 64-bit (eight-byte) value
- `ntstring` - a null-terminated string (that is, a series of bytes with a NUL byte ("\0") at the end
- `byte[]` - an array of bytes - if no size is specified, then it's the rest of the packet

在提取这部分数据块之后，依次存储列表，每次和之前的数据块进行比较：

```
last=[]
for i in find:
    if i not in last:
        last.append(i)
```

最后得到最新的文件`output`

```
hex_data = last_string
output_file = "D:/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3/output.bin"
hex_to_binary_file(hex_data, output_file)
print("written to output.bin")
```

这样我们得到的文件`output`就是去重之后的文件，但是`PNG`藏在文件内部，所以想要提取这个文件，想起了工具binwalk

```
(base) slowist@Slowist:~$ cd /mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3
(base) slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3$ binwalk output.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
95            0x5F            PNG image, 256 x 256, 8-bit gray+alpha, non-interlaced
206           0xCE            Zlib compressed data, best compression

(base) slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3$ dd if=output.bin of=extracted bs=
1 skip=95
11607+0 records in
11607+0 records out
11607 bytes (12 kB, 11 KiB) copied, 2.35701 s, 4.9 kB/s
```

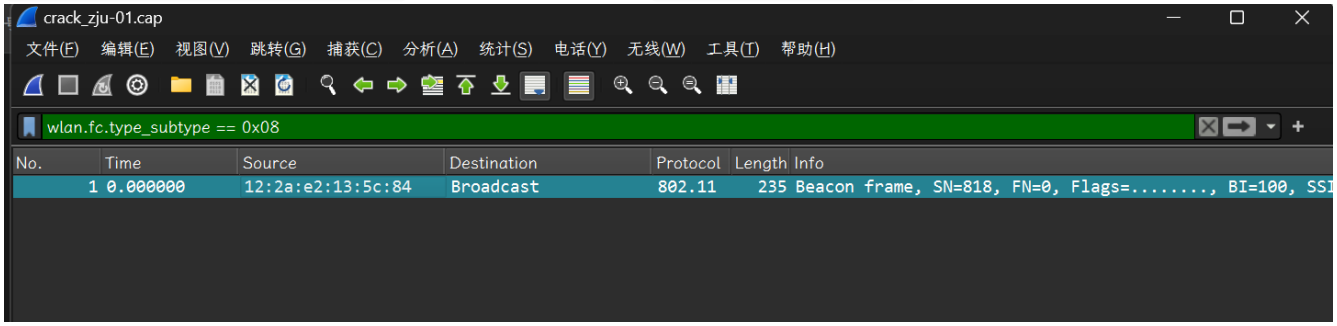
因此得到了文件：

提交的flag就是`flag{b91011fc}`

3 Challenge 3

先使用`binwalk-e`解压`godspeed.img`,得到一个`.password.txt.swp`和一个`cap`流量包
因此，想要恢复`.password.txt.swp`这个交换文件，使用`vim -r .password.txt.swp`，用`:w password.txt`保存恢复文件到新的`password.txt`

接着用`airodump-ng`破解wifi密码，先查找BSSID，用wireshark打开，查找`wlan.fc.type_subtype==0x08`，找到了Source是`12:2a:e2:13:5c:84`就是BSSID。



接着爆破：

```
aircrack-ng -w password.txt -b 00:11:22:33:44:55 crack_zju-01.cap
```

```
slowist@Slowist:/mnt/d/MyRepository/slowist-notebook/docs/Coding/CTF/misc-lec3/godspeed/_godspeed.img.extracted/ext-root$ aircrack-ng -w password.txt -b 12:2a:e2:13:5c:84 cra
_zju-01.cap
Reading packets, please wait...
Opening crack_zju-01.cap
Read 6716 packets.

1 potential targets

                               Aircrack-ng 1.6

      [00:00:00] 357/401 keys tested (6474.61 k/s)

      Time left: 0 seconds                                      89.03%

                           KEY FOUND! [ 0YcWPeLMBp ]


      Master Key     : 2A 0A 56 2C 6E 44 73 96 60 DB 3B F2 D5 76 9F 1A
                       E4 CD 5B C1 9A 08 62 FA EF 0F 65 E2 34 B4 D1 ED

      Transient Key  : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 28 A0 12 09 B3 E9 32 1F E7 4A E3 3E 05 5A C9 77
```

最后爆破出了密码 `0YcWPeLMBp`， 所以提交的就是 `AAA{0YcWPeLMBp}`

4 Challenge A