补档：D^3CTF 2021 - Misc - Robust WriteUp

原创

ObjectNF 于 2021-08-15 17:26:44 发布 108 收藏

分类专栏： 网络安全 WriteUp 文章标签： 安全 网络安全

本文链接：https://blog.csdn.net/weixin_44911246/article/details/119716567

网络安全 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏

WriteUp

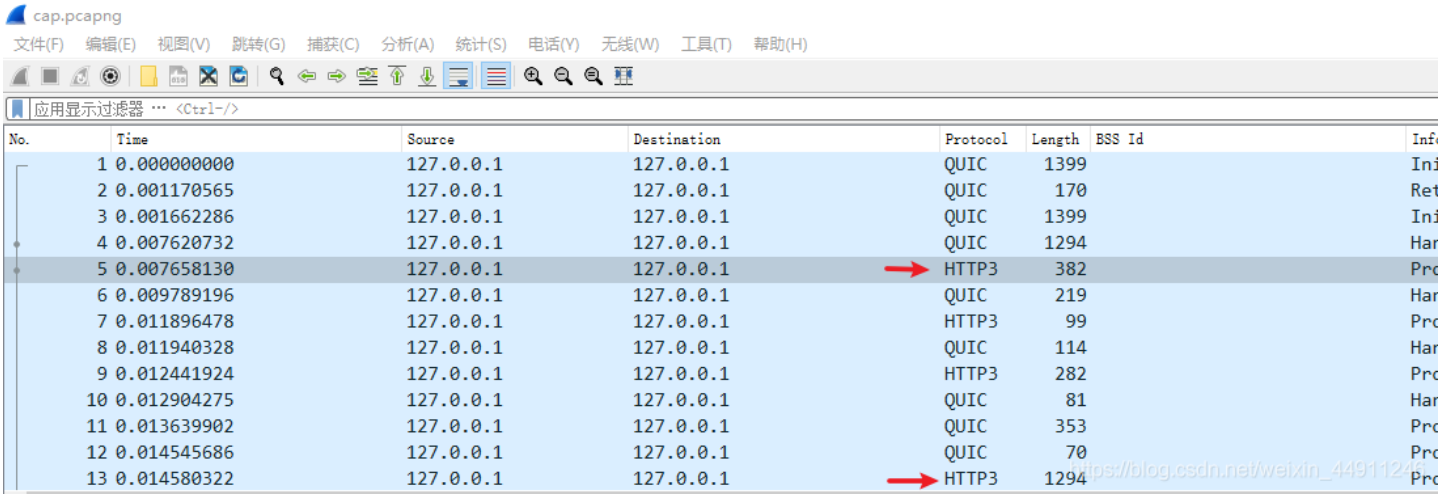
4 篇文章 0 订阅

订阅专栏

“Robust”意为“鲁棒性”。

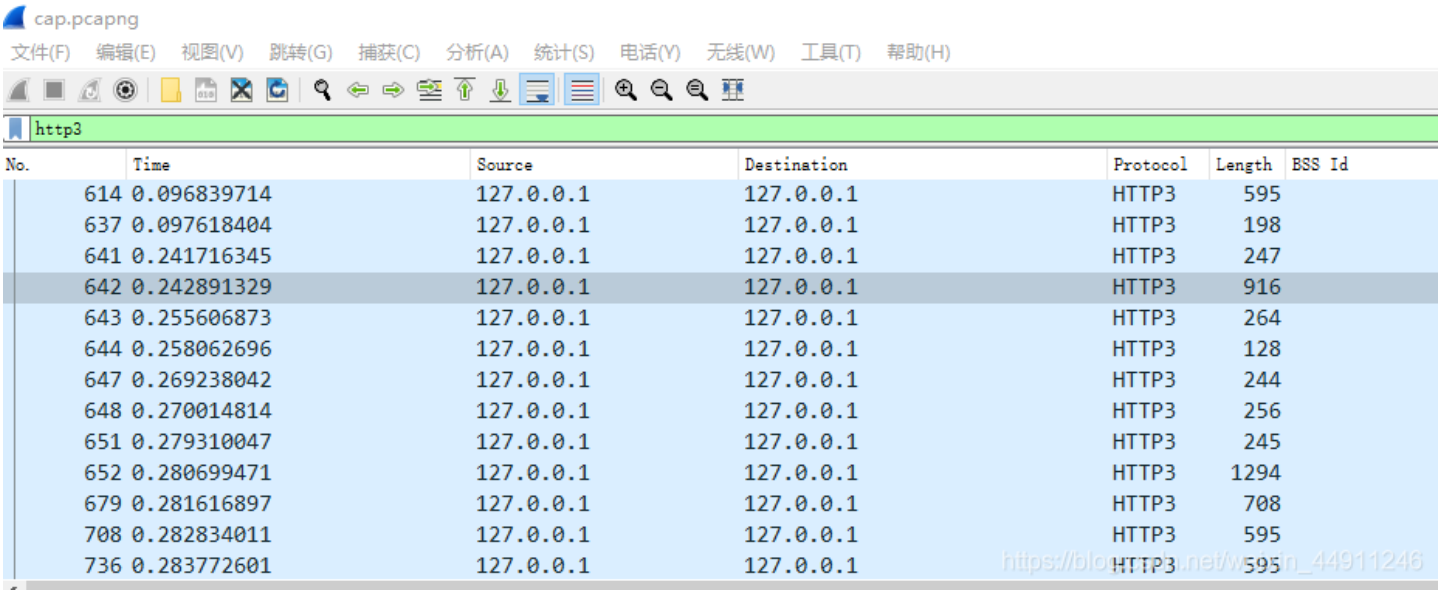
打开cap.pcapng，发现都是QUIC协议的数据包。结合提供的firefox.log（即使用firefox浏览器访问时生成的SSL Key Log）可以想到基于QUIC协议且强制使用TLS 1.3的HTTP3。

导入SSL Key Log：



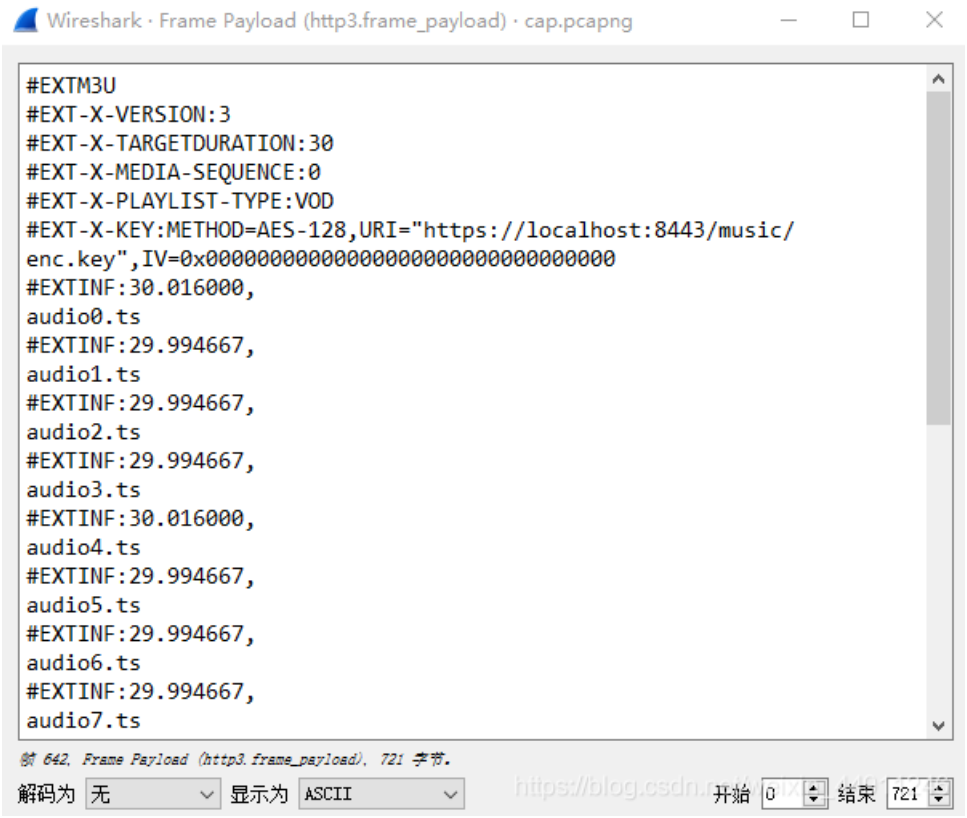
清晰可见HTTP3数据包。利用过滤器过滤出所有HTTP3数据包，然后从头查看：

```
http3
```

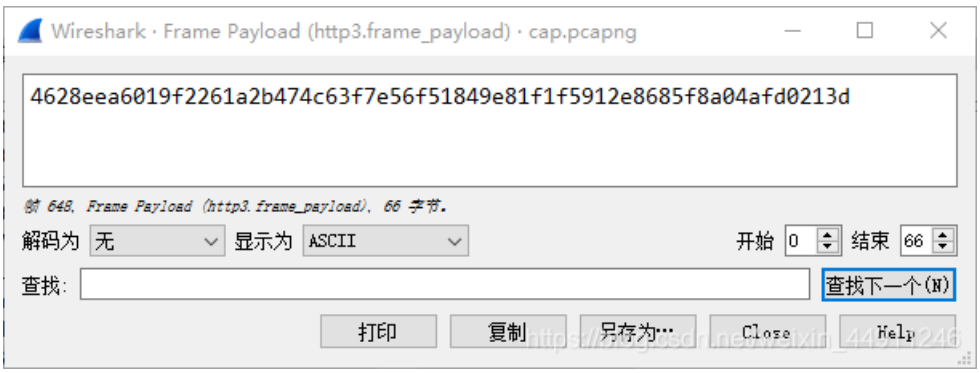


No.	Time	Source	Destination	Protocol	Length	BSS Id
614	0.096839714	127.0.0.1	127.0.0.1	HTTP3	595	
637	0.097618404	127.0.0.1	127.0.0.1	HTTP3	198	
641	0.241716345	127.0.0.1	127.0.0.1	HTTP3	247	
642	0.242891329	127.0.0.1	127.0.0.1	HTTP3	916	
643	0.255606873	127.0.0.1	127.0.0.1	HTTP3	264	
644	0.258062696	127.0.0.1	127.0.0.1	HTTP3	128	
647	0.269238042	127.0.0.1	127.0.0.1	HTTP3	244	
648	0.270014814	127.0.0.1	127.0.0.1	HTTP3	256	
651	0.279310047	127.0.0.1	127.0.0.1	HTTP3	245	
652	0.280699471	127.0.0.1	127.0.0.1	HTTP3	1294	
679	0.281616897	127.0.0.1	127.0.0.1	HTTP3	708	
708	0.282834011	127.0.0.1	127.0.0.1	HTTP3	595	
736	0.283772601	127.0.0.1	127.0.0.1	HTTP3	595	

可以明显看出，642号包之前的部分是在载入网页和JavaScript脚本。在第642号包处可以发现一个m3u8 playlist：



注意到是加密的直播流，因此想到浏览器应该获取到了解密Key。继续向下分析数据包，在第648号数据包处找到解密Key：



复制出来，另存为enc.key。

随后就是找到切片并提取切片了。600余个数据包，肯定不能手动进行处理（除非你有耐心）。因此依旧借助pyshark进行处理。有两种办法：

> 通过HTTP3数据包的类型和长度，判断每个切片的起始位置，再利用数据包内原始的m3u8 playlist和key做解密，随后合并。
>
> 依据MPEG-TS容器格式特性和AES-128-CBC加密方式特性，可以先合并，再解密。

下面以方法二为例解题。

编写脚本将所有的HTTP3 frame payload提取出来，并依次序写入同一个文件中：

```
import pyshark
import os

cap = pyshark.FileCapture("cap.pcapng", override_prefs={"ssl.keylog_file": os.path.abspath("firefox.log")})
fd = open("output.ts", "wb")


for i in range(678, 18706):
    try:
        if int(cap[i].http3.frame_type) == 0:
            fd.write(cap[i].http3.frame_payload.binary_value)
    except Exception:
        continue

fd.close()
```

之后，构造只含一个切片的m3u8 playlist：

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-KEY:METHOD=AES-128,URI="enc.key",IV=0x00000000000000000000000000000000
#EXTINF:10000
output.ts
#EXT-X-ENDLIST
```

EXTINF可以随意给大，解密密钥的URI改为相对路径。随后使用FFMPEG进行解密即可。

```
ffmpeg -allowed_extensions ALL -i index.m3u8 -c copy outdec.ts
```

你也可以使用openssl。这里注意，HLS切片加密时遇到过长的Key会截取前128位作为加密密钥。

```
xxd -P enc.key
```

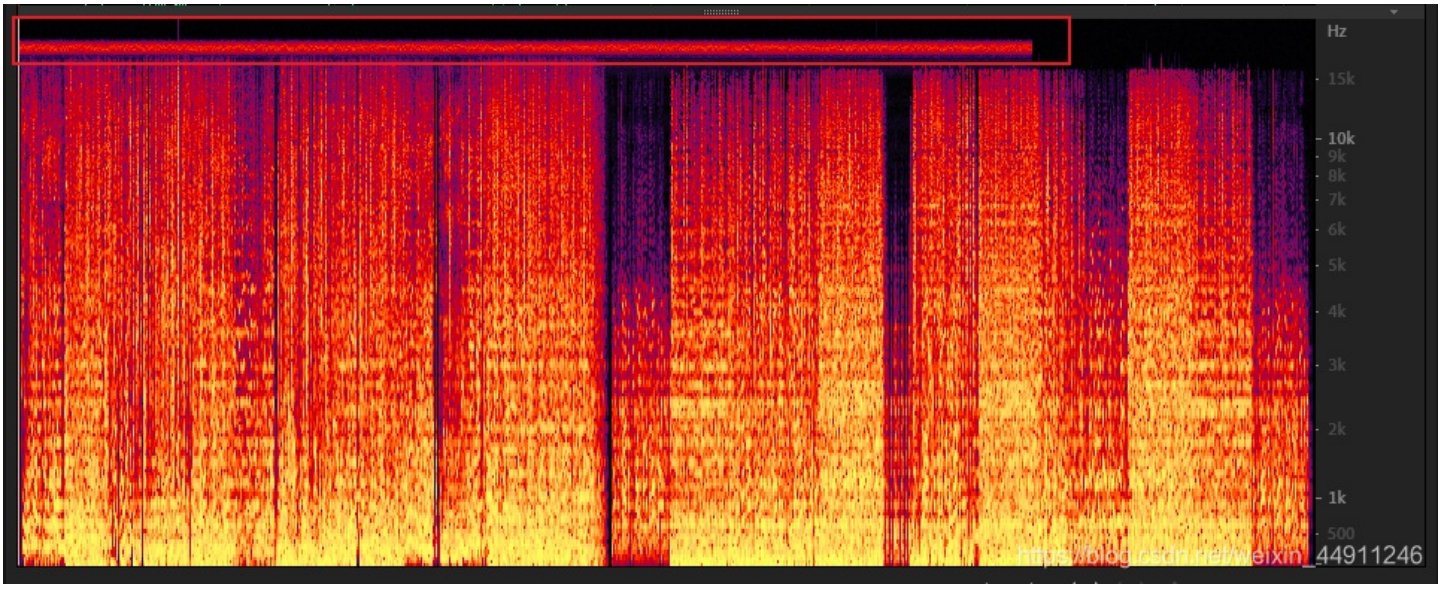
得到如下输出：

```
34363238656561363031396632323631613262343734633633663765353 6
6635313834396538316631663539313265383638356638613034616664 30
323133640d0a
```

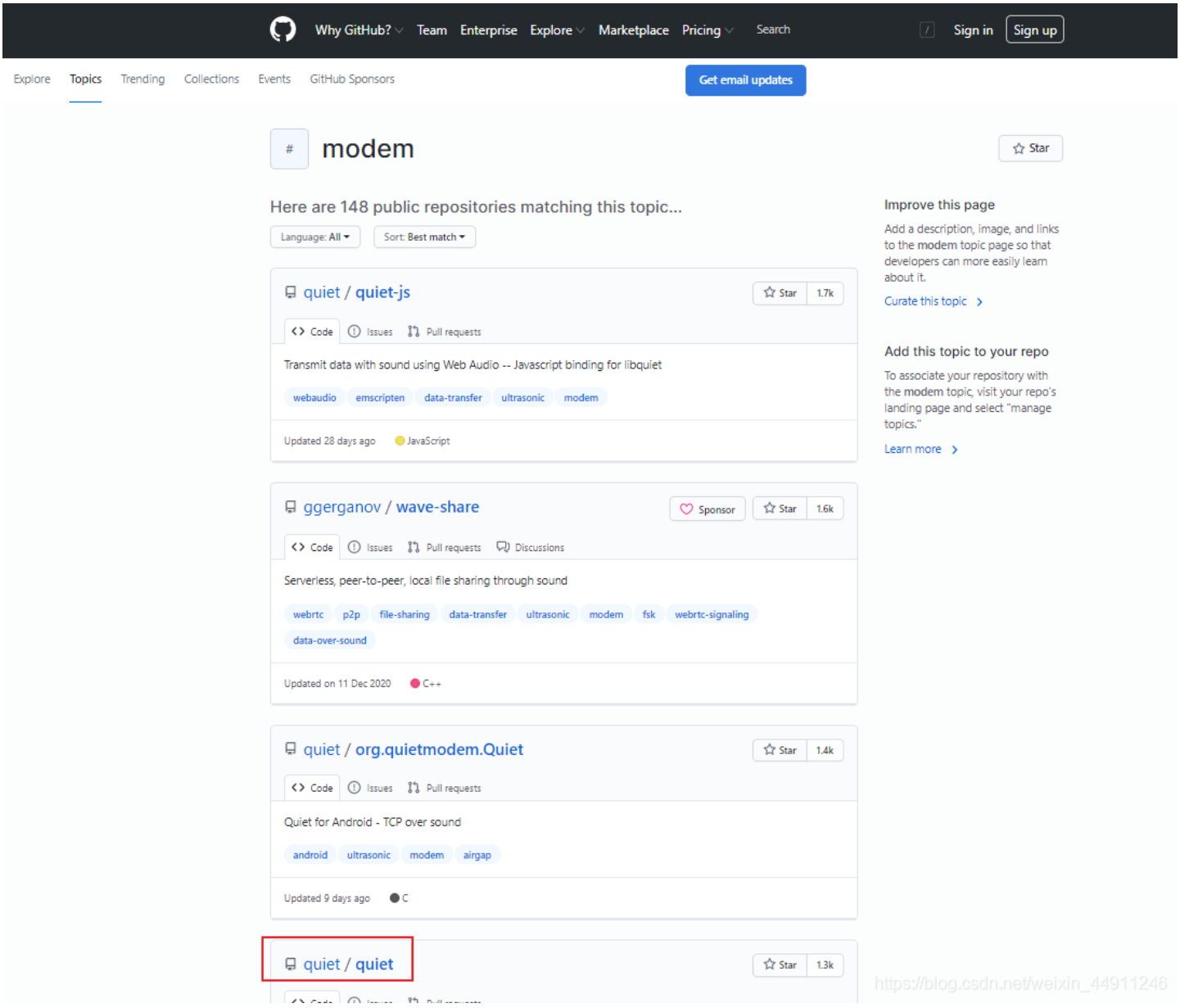
取前128位进行解密即可：

```
openssl aes-128-cbc -d -in .\output.ts -out .\out.ts -iv 00000000000000000000000000000000 -K 3436323865656136303
1396632323631 -nosalt
```

随后，解密出的ts切片就可以直接播放了。利用Adobe Audition查看频谱：



很明显这里包含了信息，需要解码。将数据通过转换变为声信号的过程很容易想到拨号上网时需要用到的的“调制解调器”。于是尝试搜索解码工具：



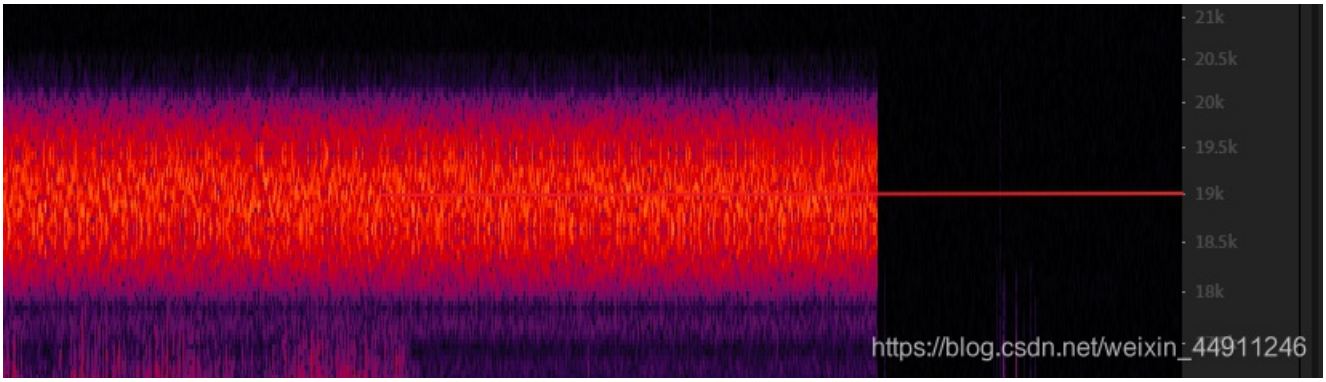
发现quiet工具（https://github.com/quiet/quiet）具有将数据转换为高频声信号（所谓的“ultrasonic”）的功能。

Ultrasonic

The `ultrasonic-` profiles encode data through a very low bitrate, but the audio content lies above 16kHz, which should pass through audio equipment relatively well while being inaudible to the average person. This is a good option for sending data through a channel where you would prefer not to disrupt human listeners.

随后，clone两个repo：quiet/libfec和quiet/quiet，编译即可。编译过程略。

在默认配置文件quiet-profiles.json中，有多个以ultrasonic开头的配置。这时回到Audition，仔细观察频谱频率：



频谱很明显以19KHz为中心，这与ultrasonic配置文件的配置相吻合：

```
"ultrasonic": {
  "mod_scheme": "gmsk",
  "checksum_scheme": "crc32",
  "inner_fec_scheme": "v27",
  "outer_fec_scheme": "none",
  "frame_length": 34,
  "modulation": {
    "center_frequency": 19000,
    "gain": 0.02
  },
  "interpolation": {
    "shape": "rrcos",
    "samples_per_symbol": 14,
    "symbol_delay": 4,
    "excess_bandwidth": 0.35
  },
  "encoder_filters": {
    "dc_filter_alpha": 0.01
  },
  "resampler": {
    "delay": 13,
    "bandwidth": 0.45,
    "attenuation": 60,
    "filter_bank_size": 64
  }
},
```

因此可以确定使用了该配置文件。

随后进行解码。可以直接使用quiet的API，当然也可使用quiet的示例程序。阅读示例程序代码decode_file.c：

```
SNDFILE *wav_open(const char *fname, unsigned int *sample_rate) {
    SF_INFO sfinfo;

    memset(&sfinfo, 0, sizeof(sfinfo));

    SNDFILE *f = sf_open(fname, SFM_READ, &sfinfo);

    *sample_rate = sfinfo.samplerate;

    return f;
}
```



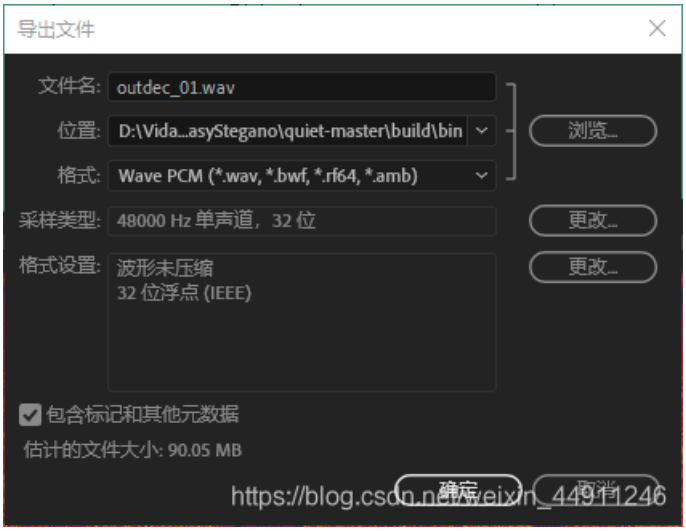
```
        #ifdef QUIET_DEBUG
            decodeopt->is_debug = true;
        #endif

            decode_wav(output, "encoded.wav", decodeopt);

            fclose(output);
            free(decodeopt);

            return 0;
```

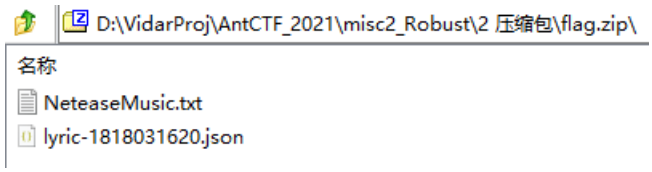

需要将待解码的文件转化为wav格式，且重命名为encoded.wav。联想到题目的“Robust”，意即“鲁棒性”，因此大胆直接转码。但是为了不丢失数据，保险起见，保持原采样率和最高量化位数：



随后解码：

```
./quiet_decode_file ultrasonic out.txt
```

得到Base64，解码发现PK头，保存为Zip，打开，发现有密码：



根据文件名称，联想到网易云音乐的歌词。至于是哪首歌的歌词，联想到网易云音乐有歌曲的“Song ID”。因此首先找到歌曲：

https://music.163.com/song?id=1818031620

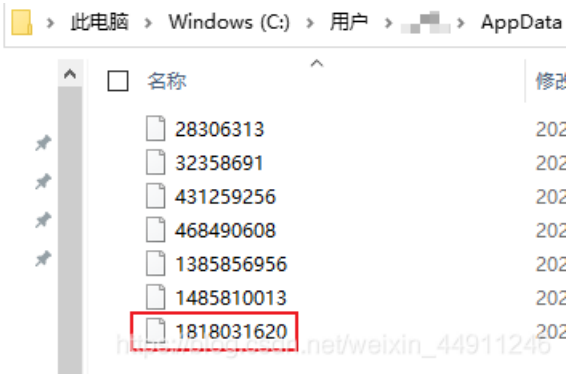
随后提取歌词。

方法一：利用网页API。F12仔细分析即可找到歌词获取接口。例子如下：

```
https://music.163.com/weapi/song/lyric?csrf_token=7ab6599ebc00854e324f6dcf04353358
```

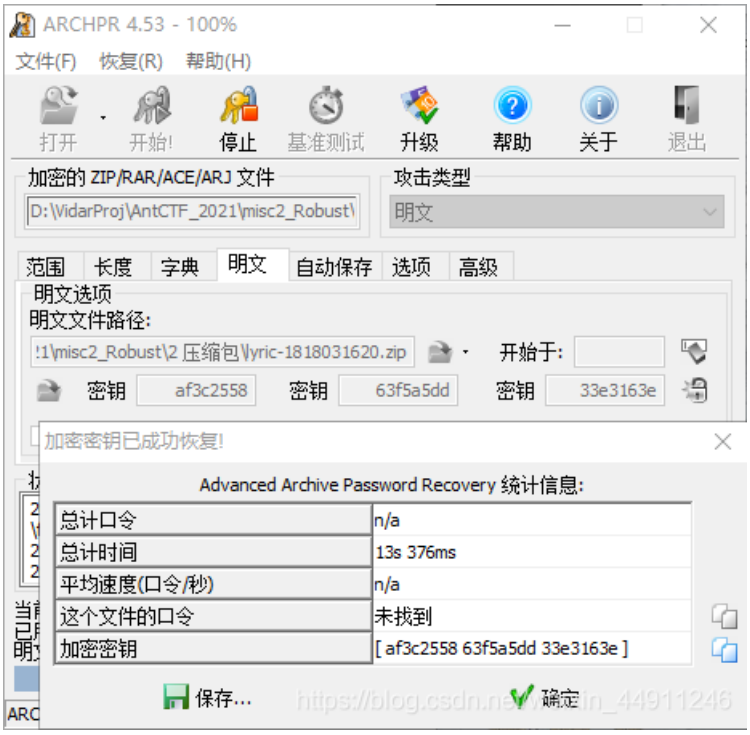
将网页内容保存为lyric-1818031620.json文件即可。

方法二：利用网易云音乐客户端缓存。容易找到缓存歌词的文件夹（以Windows系统为例）：
C:\Users\ObjectNotFound\AppData\Local\Netease\CloudMusic\webdata\lyric

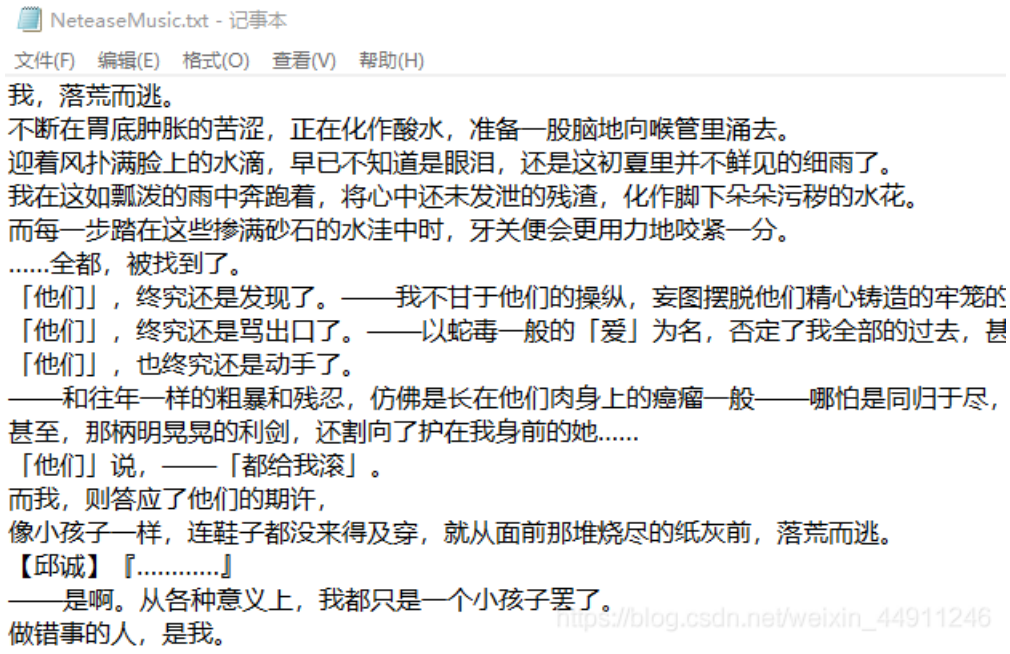


将该文件复制出来改名为lyric-1818031620.json即可。

然后就可以进行明文攻击了。可以使用Advanced Archive Password Recovery，也可以使用pkcrack。注意Zip的压缩算法设置。



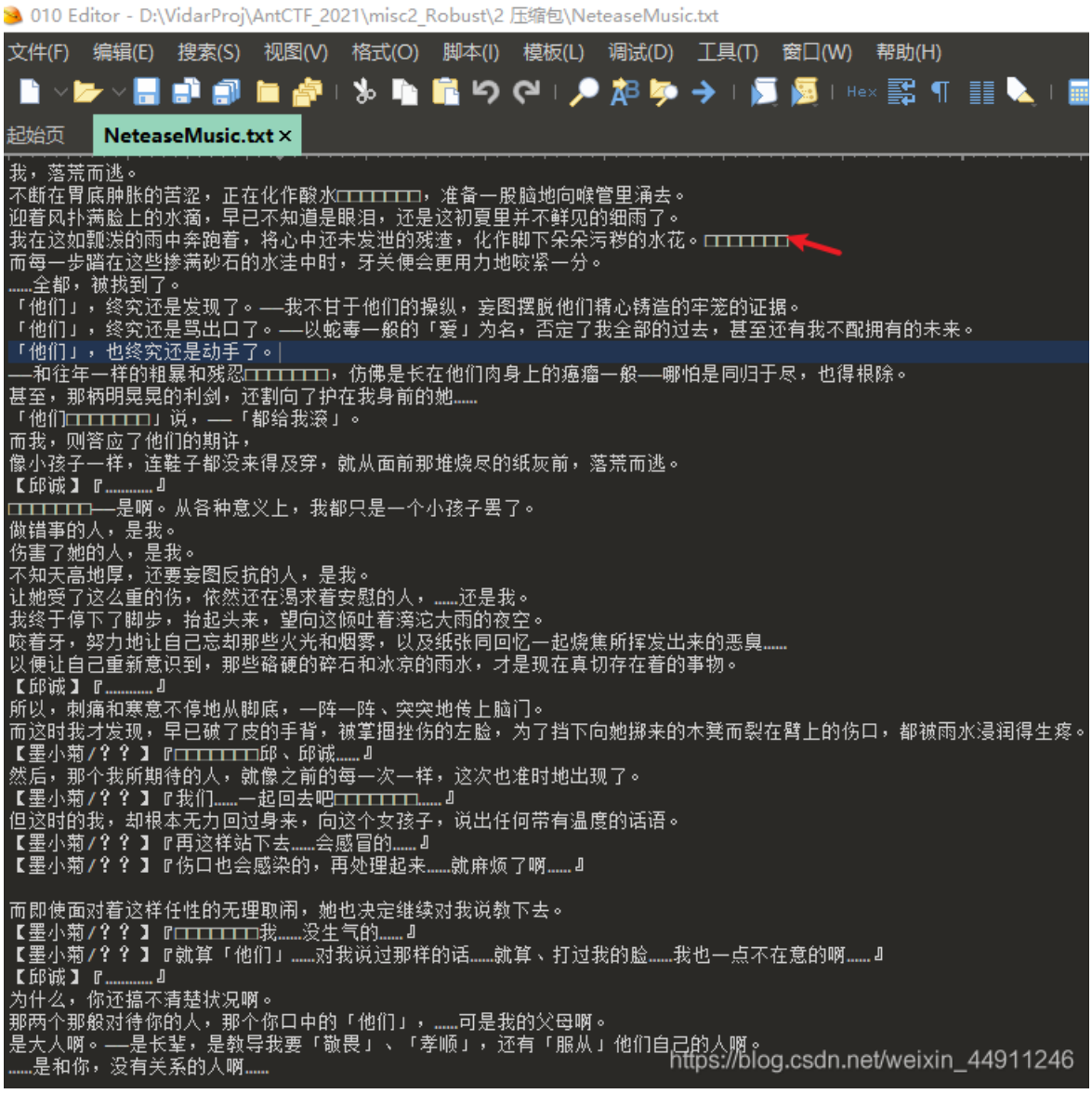
得到txt文档的内容：


NeteaseMusic.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

我，落荒而逃。
不断在胃底肿胀的苦涩，正在化作酸水，准备一股脑地向喉管里涌去。
迎着风扑满脸上的水滴，早已不知道是眼泪，还是这初夏里并不鲜见的细雨了。
我在这如瓢泼的雨中奔跑着，将心中还未发泄的残渣，化作脚下朵朵污秽的水花。
而每一步踏在这些掺满砂石的水洼中时，牙关便会更用力地咬紧一分。
……全都，被找到了。
「他们」，终究还是发现了。——我不甘于他们的操纵，妄图摆脱他们精心铸造的牢笼的
「他们」，终究还是骂出口了。——以蛇毒一般的「爱」为名，否定了我全部的过去，甚
「他们」，也终究还是动手了。
——和往年一样的粗暴和残忍，仿佛是长在他们肉身上的癌瘤一般——哪怕是同归于尽，
甚至，那柄明晃晃的利剑，还割向了护在我身前的她……
「他们」说，——「都给我滚」。
而我，则答应了他们的期许，
像小孩子一样，连鞋子都没来得及穿，就从面前那堆烧尽的纸灰前，落荒而逃。
【邱诚】『……』
——是啊。从各种意义上，我都只是一个小孩子罢了。
做错事的人，是我。

换用其他软件查看，发现存在空白字符隐写：


010 Editor - D:\VidarProj\AntCTF_2021\misc2_Robust\2 压缩包\NeteaseMusic.txt

文件(F) 编辑(E) 搜索(S) 视图(V) 格式(O) 脚本(I) 模板(L) 调试(D) 工具(T) 窗口(W) 帮助(H)

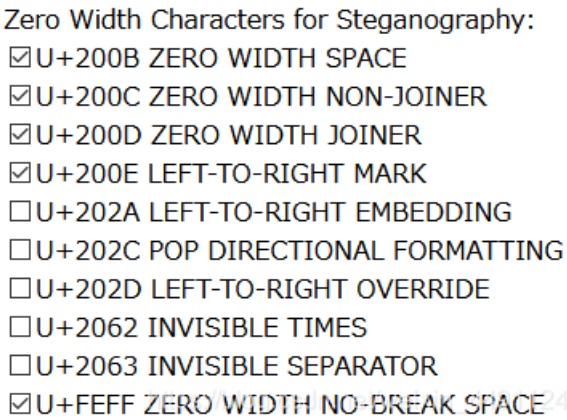
起始页 NeteaseMusic.txt ×

我，落荒而逃。
不断在胃底肿胀的苦涩，正在化作酸水□□□□□□□，准备一股脑地向喉管里涌去。
迎着风扑满脸上的水滴，早已不知道是眼泪，还是这初夏里并不鲜见的细雨了。
我在这如瓢泼的雨中奔跑着，将心中还未发泄的残渣，化作脚下朵朵污秽的水花。□□□□□□□
而每一步踏在这些掺满砂石的水洼中时，牙关便会更用力地咬紧一分。
……全都，被找到了。
「他们」，终究还是发现了。——我不甘于他们的操纵，妄图摆脱他们精心铸造的牢笼的证据。
「他们」，终究还是骂出口了。——以蛇毒一般的「爱」为名，否定了我全部的过去，甚至还有我不配拥有的未来。
「他们」，也终究还是动手了。
——和往年一样的粗暴和残忍□□□□□□□，仿佛是长在他们肉身上的癌瘤一般——哪怕是同归于尽，也得根除。
甚至，那柄明晃晃的利剑，还割向了护在我身前的她……
「他们□□□□□□□」说，——「都给我滚」。
而我，则答应了他们的期许，
像小孩子一样，连鞋子都没来得及穿，就从面前那堆烧尽的纸灰前，落荒而逃。
【邱诚】『……』
□□□□□□□——是啊。从各种意义上，我都只是一个小孩子罢了。
做错事的人，是我。
伤害了她的人，是我。
不知天高地厚，还要妄图反抗的人，是我。
让她受了这么重的伤，依然还在渴求着安慰的人，……还是我。
我终于停下了脚步，抬起头来，望向这倾吐着滂沱大雨的夜空。
咬着牙，努力地让自己忘却那些火光和烟雾，以及纸张同回忆一起烧焦所挥发出来的恶臭……
以便让自己重新意识到，那些硌硬的碎石和冰凉的雨水，才是现在真切存在着的事物。
【邱诚】『……』
所以，刺痛和寒意不停地从脚底，一阵一阵、突突地传上脑门。
而这时我才发现，早已破了皮的手背，被掌掴挫伤的左脸，为了挡下向她掷来的木凳而裂在臂上的伤口，都被雨水浸润得生疼。
【墨小菊/？？？】『□□□□□□□邱、邱诚……』
然后，那个我所期待的人，就像之前的每一次一样，这次也准时地出现了。
【墨小菊/？？？】『我们……一起回去吧□□□□□□□……』
但这时的我，却根本无力回过身来，向这个女孩子，说出任何带有温度的话语。
【墨小菊/？？？】『再这样站下去……会感冒的……』
【墨小菊/？？？】『伤口也会感染的，再处理起来……就麻烦了啊……』

而即使面对着这样任性的无理取闹，她也决定继续对我说教下去。
【墨小菊/？？？】『□□□□□□□我……没生气的……』
【墨小菊/？？？】『就算「他们」……对我说过那样的话……就算、打过我的脸……我也一点不在意的啊……』
【邱诚】『……』
为什么，你还搞不清楚状况啊。
那两个那般对待你的人，那个你口中的「他们」，……可是我的父母啊。
是大人啊。——是长辈，是教导我要「敬畏」、「孝顺」，还有「服从」他们自己的人啊。
……是和你，没有关系的人啊……

利用工具解密即可。注意选择正确的解码设置。可以使用十六进制编辑器的查找功能确定使用的编码字符。

https://330k.github.io/misc_tools/unicode_steganography.html



得到解码结果：

```
<~A2@_;ApZ7(GA0]MC.i&:F%'t#:JXSd=tj-$>'EtK0m.1t0i38~>
```

由定界符<~ ~>易知其为Base85编码。解码可得Flag：

```
d3ctf{1IwiKUjKcUsEn0OOJZZ0ZsZwUX1uiC1P}
```