CAN

观察 candump.txt 文件，可以看到存在着两段大段的数据传输，这两段传输长度相同，且使用了 ISO-TP 协议，结合题目描述中说明，这段 CAN 数据对应一个文件，经观察与尝试发现，上下段数据存在大量数据异或结果为 0x00 或者 0xFF，且异或结果中包含大量规律数据以及 zip 压缩包头，可以认为上下段数据实际上是使用了一次一密（OTP）的方式进行传输，一次传输为密钥，一次传输为密文。

观察到此时除首帧之外，每个连续帧中第四字节到结束的数据为 zip 包的内容，提取出来后发现 zip 包存在密码。还发现，首帧中最后剩余的有效数据为 8 这一数字，而前八个连续帧中的第二字节为 0x00 ~ 0x07 ，第三字节包含奇怪的数据，之后的连续帧中第二字节均为 0xFF，第三字节均为 0x00。这一部分即对应着压缩包的8位密码。

将前八个连续帧中的第三字节数据提取出来，并表示成 bit 的形式，如下所示

```
1 0 1 0 0 1 0 0
0 0 0 1 1 0 1 1
0 1 0 1 1 1 1 0
1 1 0 0 0 0 1 0
1 0 1 0 1 0 1 1
1 0 1 0 1 1 0 1
1 1 1 1 0 1 1 0
0 0 0 0 0 0 0 0
```

发现这满足 8 * 8 的样式，且最下面一行均为 0，可以发现对于每一列数据，从下至上依次作为数的高位到低位来解读，这些数会落在可打印的ASCII字符范围内。得到的数据为

```
yLqF6e^2
```

即压缩包的密码。

exp 如下：

```
files = [[], []]
index = 0
flag = []

with open('candump.log', 'r') as r:
    for i in range(1):
        line = r.readline()
    line = r.readline().strip()
    while line is not None and len(line) != 0:
        if '7D1#' in line:
            index = 1
            line = r.readline().strip()
            continue
        line = line.split('#')
        files[index].append(line[1])
```

```
        line = r.readline().strip()

for i in range(len(files[0])):
    tmpa = bytes.fromhex(files[0][i])
    tmpb = bytes.fromhex(files[1][i])
    tmpc = b''
    for j in range(len(tmpa)):
        tmpc += bytes.fromhex(hex(tmpa[j] ^ tmpb[j])[2:].rjust(2, '0'))
    flag.append(tmpc)

for i in flag:
    print(i.hex().upper())

passwordList = [0 for _ in range(8)]

with open('flag.zip', 'wb') as w:
    for i in range(1, len(flag)):
        w.write(flag[i][3:])
        num = flag[i][1]
        if num != 0xff:
            tmp = flag[i][2]
            for j in range(8):
                passwordList[j] = passwordList[j] | (((tmp >> (7 - j)) & 1) <<
(num))

print('password:')

for i in passwordList:
    print(chr(i), end = '')

print('\n')
```