TTL隐写

原创

brightendavid 于 2021-04-20 18:51:57 发布 334 收藏 2

分类专栏： CTF 文章标签： 安全 ttl

本文链接：https://blog.csdn.net/brightendavid/article/details/115915512


```
#拿到一个长成这样的文件，里面是意义不明的数字
63
63
63
255
63
63
63
255
63
63
63
255
63
63
63
255
63
63
63
255
63
127
63
255
63
```

转换为2进制，并设置为8位数的2进制，就会有很神奇的事情发生

```
with open('C:/Users/brighten/Desktop/attachment.txt', 'r') as f:
    for line in f:
        num=int(line)
        ss=bin(num)
        while len(ss)<10:
            ss=ss[:2]+'0'+ss[2:]
        print(ss)
```

可能和上面的对不上，但是意思是这个意思。

后面的6位数是一样的，全是**111111**

```
0b00111111
0b11111111
0b01111111
0b01111111
0b00111111
0b11111111
0b00111111
0b00111111
0b00111111
0b11111111
0b01111111
0b00111111
0b01111111
```

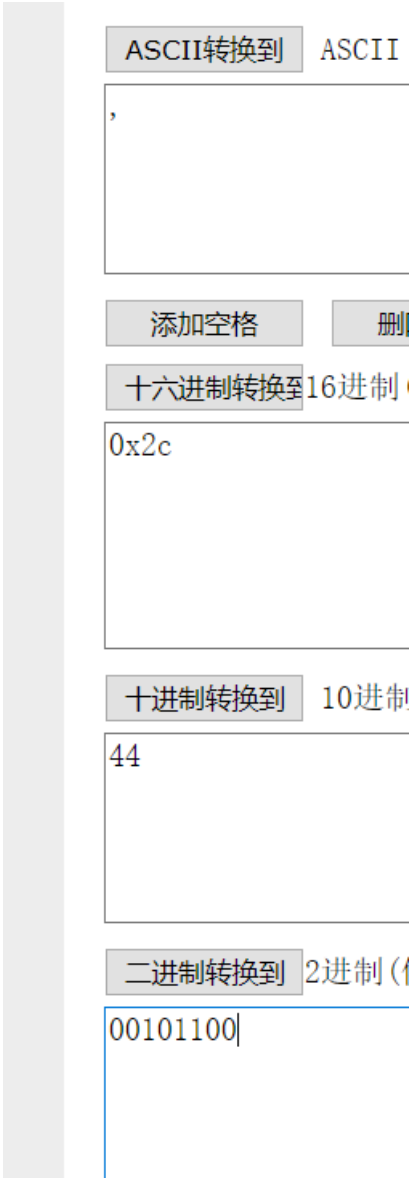
那么现在学习到了一个新的加密方法，就是所谓的TTL隐写。

抄一段百度

> TTL是 Time To Live的缩写，该字段指定IP包被路由器丢弃之前允许通过的最大网段数量。TTL是IPv4报头的一个8 bit字段。注意：TTL与DNS
> TTL有区别。二者都是生存时间，前者指ICMP包的转发次数（跳数），后者指域名解析信息在DNS中的存在时间。

IP报文在路由间穿梭的时候每经过一个路由，TTL就会减1。
这个东西是用来防止数据过多的。计算机网络知识。

大多数情况下通常只需要经过很小的跳数就能完成报文的转发，远远比上限255小得多，所以我们可以用TTL值的前两位来进行传输隐藏数据。
所以加密的方法就是把一个ASCII码分4截，分到4个TTL里面，有点像是最低位像素的隐写。



像是这个逗号，在这个隐写下就是

```
00111111
10111111
11111111
00111111
```

```
count = 0
str=""
with open('C:/Users/brighten/Desktop/attachment.txt', 'r') as f:
    for line in f:
        num = int(line)
        ss = bin(num)
        while len(ss) < 10:
            ss = ss[:2] + '0' + ss[2:]
        #print(ss)
        str=str+ss[2:4]
        count += 1
        if count == 4:
            count = 0
            sum=0
            #print(str)
            for i in range(len(str)):
                if str[i]=='1':
                    sum=sum*2+1
                else:
                    sum=sum*2
            # print(sum)
            print(chr(sum),end="")
            str=""
```

解码程序如上

解出来这个应该是应该zip压缩包吧。