

base64隐写

转载

[aminzhuo3701](#) 于 2017-04-18 00:50:00 发布 66 收藏

文章标签: [python](#)

原文链接: <http://www.cnblogs.com/Elope/p/6725824.html>

版权

由NJCTF2017一道misc题引出的问题

先来看一下base64的概念

Base64要求把每三个8Bit的字节转换为四个6Bit的字节 ($3 \times 8 = 4 \times 6 = 24$),然后把6Bit再添两位高位0,组成四个8Bit的字节,也就是说,转换后的字符串理论上将要比原来的长1/3。

规则

关于这个编码的规则:

- 1.把3个字符变成4个字符
- 2.每76个字符加一个换行符
- 3.最后的结束符也要处理
- 4.10个数字,26个大写字母,26个小写字母,1个+,一个/刚好64个字符

例子

为方便理解,举下例子

转换前 11111011, 11101111, 10111110 (二进制)

首先按每6位分开 111110, 111110, 111110, 111110

再到最高位添加两个0

转换后 00111110, 00111110, 00111110, 00111110 (二进制)

上面的三个字节是原文,下面的四个字节是转换后的Base64编码,其前两位均为0。

转换后,我们用一个码表来得到我们想要的字符串(也就是最终的Base64编码),这个表是这样的:

	索引 对应字符
1	
2	0 A
3	1 B
4	2 C
5	3 D
6	4 E
7	5 F
8	6 G
9	7 H
10	8 I
11	9 J
12	10 K
13	11 L
14	12 M
15	13 N
16	14 O
17	15 P
18	16 Q
19	17 R

20	18 S
21	19 T
22	20 U
23	21 V
24	22 W
25	23 X
26	24 Y
27	25 Z
28	26 a
29	27 b
30	28 c
31	29 d
32	30 e
33	31 f
34	32 g
35	33 h
36	34 i
37	35 j
38	36 k
39	37 l
40	38 m
41	39 n
42	40 o
43	41 p
44	42 q
45	43 r
46	44 s
47	45 t
48	46 u
49	47 v
50	48 w
51	49 x
52	50 y
53	51 z
54	52 0
55	53 1
56	54 2
57	55 3
58	56 4
59	57 5
60	58 6
61	59 7
62	60 8
63	61 9
64	62 +
65	63 /

虽然python解base64很方便
最好还是先写个脚本加深对base64的理解
文末我会贴出我的调试脚本

那么base64隐写到底是什么东西呢？

关键是base64解码的时候

- 1.检查base64编码后面有几个等于号
- 2.把字符串按照base64表转换成46的倍数位数二进制
- 3.删除等于号的个数8的bit_(等于号不在base64规定的范围内,只是为了补足长度,所以解码时要删除)
- 4.按照6个bit一组转成字符

此处的关键就是,解码的时候,会删除等于号的个数8的bit
且我们只用6个bit表示一个等于号(xxxxxx)
那么,意思就是我们可以控制__等于号个数2bit__的字符

NJCTF2017有一道misc是base64隐写
从里面抽出一条来

```
1 import base64
2 s = "QnkgcmVhc29uIG9mIGhpcyBmYWxsZW4gZGI2aW5pdHm="
3 print base64.b64decode(s)
4 print s
5 print base64.b64encode(base64.b64decode(s))
```

会发现返回

```
1 By reason of his fallen divinity
2 QnkgcmVhc29uIG9mIGhpcyBmYWxsZW4gZGI2aW5pdHm=
3 QnkgcmVhc29uIG9mIGhpcyBmYWxsZW4gZGI2aW5pdHk=
```

会发现有什么区别,但是并不影响正常显示
来分析一下为啥,就拿我的昵称举例
首先是把每位字符转化成ascii的二进制形式

```
1 c | 0 | 1 | 1 | 4
2 01100011 | 00110000 | 00110001 | 00110100
```

进行base64编码时,按每6位分开,如果不能被6整除要用0补齐

```
1 011000 11|0011 0000|00 110001 | 001101 00
2 011000 | 110011 | 000000 | 110001 | 001101 | 000000
```

并且所有高位补两个0至8位

```
1 011000 | 110011 | 000000 | 110001 | 001101 | 000000
2 00011000 | 00110011 | 00000000 | 00110001 | 00001101 | 00000000
```

这样才能刚好用64个不同形式表示

1	00011000 00110011 00000000 00110001 00001101 00000000
2	Y z A x N A

但是会发现字符只有6位数不够被4整除

所以到字符后面添加2个等号

1	YzAxNA
2	YzAxNA==

解密时

先找出YzAxNA==每位字符在base64码表中的位置

由于等号没有我就不表示了

1	Y z A x N A
2	011000 110011 000000 110001 001101 000000

将二进制连接起来并按每八位划分

1	011000110011000000110001001101000000
2	01100011 00110000 00110001 00110100 0000

会发现末尾多了等号数*2bit的0

这些0在解密时是要被删除的

所以base64隐写的精髓就是这几bit可控

如

1	YzAxNA==
2	YzAxNB==
3	YzAxNC==
4
5	YzAxNO==

解密的到的都是一样的结果

调试脚本

1	
2	
3	
4	
5	
6	

```

7
8
9
10
11
12
13
14
15
16
17
18
19
20 # -*- coding:utf-8 -*-
21
22
23 base = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
24
25 def encode(str):
26     s = str
27     s_bin = ""
28     for i in range(0,len(s)):
29         s_bin = s_bin + bin(ord(s[i])).replace("0b","").rjust(8,"0") #把每个字符二进制化
30     print s_bin
31
32     if len(s_bin) % 6 != 0: #判断二进制长度能否被6整除
33         for j in range(1,6):
34             if (len(s_bin) + j) % 6 == 0:
35                 bin_length = len(s_bin) + j #获取二进制长度
36                 break
37         s_bin = s_bin.ljust(bin_length, "0") #把二进制长度补到能被6整除
38     print s_bin
39
40     encode = ""
41     for k in range(0,len(s_bin)/6):
42         index = k * 6
43         each_bin = s_bin[index:index+ 6].zfill(8) #把6Bit再添两位高位0
44         print each_bin
45         each_enc = int(each_bin, 2) #转化为10进制
46         print each_enc
47         encode = encode + base[each_enc]
48
49     # print encode
50     if len(encode) % 4 != 0: #判断解密后字符串是否能被4整除,如果不能,要在末尾加=
51         for l in range(1,4):
52             if (len(encode) + l) % 4 == 0:
53                 encode_length = len(encode) + l
54                 print encode.ljust(encode_length,"=")
55
56     def raw(str): #获取每位字符的二进制形式
57         s = str
58         for i in range(0,len(s)):
59             print s[i],bin(ord(s[i])).replace("0b","").rjust(8,"0")
60
61
62
63
64
65

```

66
67
68
69
70
71
72
73
74
75
76
77
78

转载于:<https://www.cnblogs.com/Elope/p/6725824.html>