# Base64隐写

原创

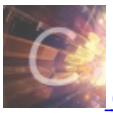
xnightmare 于 2019-12-30 22:06:27 发布 5103 收藏 18

分类专栏： CTF # MISC



本文链接：https://blog.csdn.net/xnightmare/article/details/103774379




CTF 同时被 2 个专栏收录

3 篇文章 0 订阅

订阅专栏


MISC

3 篇文章 0 订阅

订阅专栏

Base64编码是一种常见的编码方式，本文首先介绍了Base64编码的原理，在此基础上介绍了Base64隐写的原理，然后给出了CTF比赛中Base64隐写的一个实例，最后在附录中给出了Base64编码解码的Python代码。

## Base64编码的原理

Base64是一种基于64个可打印字符表示二进制数据的表示方法，其一大特点是能够将不可打印字符编码为可打印字符。

**Base64使用的64个可打印字符及其索引**如下表：

| 索引 | 字符 | 索引 | 字符 | 索引 | 字符 | 索引 | 字符 |
|---|---|---|---|---|---|---|---|
| 0 | A | 16 | Q | 32 | g | 48 | w |
| 1 | B | 17 | R | 33 | h | 49 | x |
| 2 | C | 18 | S | 34 | i | 50 | y |
| 3 | D | 19 | T | 35 | j | 51 | z |
| 4 | E | 20 | U | 36 | k | 52 | 0 |
| 15 | F | 21 | V | 37 | l | 53 | 1 |
| 6 | G | 22 | W | 38 | m | 54 | 2 |
| 7 | H | 23 | X | 39 | n | 55 | 3 |
| 8 | I | 24 | Y | 40 | o | 56 | 4 |
| 9 | J | 25 | Z | 41 | p | 57 | 5 |
| 10 | K | 26 | a | 42 | q | 58 | 6 |
| 11 | L | 27 | b | 43 | r | 59 | 7 |
| 12 | M | 28 | c | 44 | s | 60 | 8 |
| 13 | N | 29 | d | 45 | t | 61 | 9 |
| 14 | O | 30 | e | 46 | u | 62 | + |
| 15 | P | 31 | f | 47 | v | 63 | / |



简单来说，就是A-Za-z0-9+/这64个可打印字符。

编码时，将要编码的内容转换为二进制数据，每6位作为一组，从表中找到对应的字符。因为ASCII编码8位表示一个字符，3个ASCII刚好可以编码成4个字符（3*8=4*6），因此一般以3个ASCII字符为一个编码的基本单位：

| 字符 | h | | | | | | | | e | | | | | | | | l | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII值 | 104 | | | | | | | | 101 | | | | | | | | 108 | | | | | | | |
| 二进制 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 索引 | 26 | | | | | | | | 6 | | | | | | | | 21 | | | | | | | |
| Base64编码 | a | | | | | | | | G | | | | | | | | V | | | | | | | |

但需要编码的文本字节数并不总是3的倍数，不可避免会遇见**最后只剩下2个或1个字符的情况，需要特殊处理**：

%3=2的情况：

| 字符 | h | | | | | | | | e | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII值 | 104 | | | | | | | | 101 | | | | | | | | | | | | | | | |
| 二进制 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | **0** | **0** | | | | | | |
| 索引 | 26 | | | | | | | | 6 | | | | 20 | | | | | | | | | | | |
| Base64编码 | a | | | | | | | | G | | | | U | | | | = | | | | | | | |

%3=1的情况：

| 字符 | h | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ASCII值 | 104 | | | | | | | | | | | | | | | | | | | | | | | |
| 二进制 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | **0** | **0** | **0** | **0** | | | | | | | | | | | | |
| 索引 | 26 | | | | | | | | 0 | | | | | | | | | | | | | | | |
| Base64编码 | a | | | | | | | | A | | | | = | | | | = | | | | | | | |

## Base64隐写的原理

从Base64编码的原理我们很自然推出Base64解码的过程：

1. 丢掉末尾的所有‘=’；
2. 每个字符查表转换为对应的6位索引，得到一串二进制字符串；
3. 从头开始，每次取8位转换为对应的ASCII字符，**如果不足8位则丢弃。**

在解码的第3步中，会有部分数据被丢弃（即不会影响解码结果），这些数据正是我们在编码过程中补的0（本文第三第四张图中加粗的部分）。也就是说，如果我们在编码过程中不全用0填充，而是用其他的数据填充，仍然可以正常编码解码，因此这些位置可以用于隐写。

解开隐写的方法就是将这些不影响解码结果的位提取出来组成二进制串，然后转换成ASCII字符串。

## Base64隐写实例

在GXYCTF2019中出现了一道Base64隐写的题目，题目网址为：https://buuoj.cn/challenges#[GXYCTF2019]SXMgdGhpcyBiYXNlPw==
题目文件中每一行为一串Base64编码后的字符串，解题思路大致如下：

1. 依次读取每行，从中提取出隐写位。
    1. 如果最后没有‘=’，说明没有隐写位，跳过。
    2. 如果最后是一个‘=’，说明有两位隐写位，将倒数第二个字符转化为对应的二进制索引，然后取后两位。
    3. 如果最后是两个‘=’，说明有四位隐写位，将倒数第三个字符转化为对应的二进制索引，然后取后四位。
2. 将每行提取出的隐写位依次连接起来，每8位为一组转换为ASCII字符，最后不足8位的丢弃。

解题脚本如下（flag.txt是在Windows下生成的文本文件，换行为\r\n，如果要在Linux下运行脚本，需要先把flag.txt每行末尾的\r\n替换为\n）：

```
def inttobin(a, n):
    ret = bin(a)[2:]
    while len(ret) < n:
        ret = '0' + ret
    return ret

table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'

f = open("flag.txt", "r")
tmpbin = ''
res = ''
line = f.readline()
while line:
    if line[-2] == '=':
        if line[-3] == '=':
            tmpbin += inttobin(table.index(line[-4]), 6)[2:]
        else:
            tmpbin += inttobin(table.index(line[-3]), 6)[4:]
    line = f.readline()
quotient = len(tmpbin)/8
for i in range(quotient):
    res += chr(int(tmpbin[8*i:8*i+8], 2))
print res
```

## 附录

**Base64编码和解码脚本：**

```
table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'

def inttobin(a, n):
    res = bin(a)[2:]
    while len(res) < n:
        res = '0' + res
    return res

def b64encode(strin):
    res = ''
    quotient = len(strin)/3
    remainder = len(strin)%3
    for i in range(quotient):
        tmp = strin[i*3:i*3+3]
        tmpbin = inttobin(ord(tmp[0]), 8) + inttobin(ord(tmp[1]), 8) + inttobin(ord(tmp[2]), 8)
        print tmpbin
        for j in range(4):
            index = int(tmpbin[6*j:6*j+6], 2)
            res += table[index]
    if remainder == 1:
        tmpbin = inttobin(ord(strin[-1]), 8) + '0000'
        res += table[int(tmpbin[:6], 2)]
        res += table[int(tmpbin[6:], 2)]
        res += '=='
    elif remainder == 2:
        tmpbin = inttobin(ord(strin[-2]), 8) + inttobin(ord(strin[-1]), 8) + '00'
        res += table[int(tmpbin[:6], 2)]
        res += table[int(tmpbin[6:12], 2)]
        res += table[int(tmpbin[12:], 2)]
        res += '='
    return res

def b64decode(strin):
    res = ''
    while strin[-1] == '=':
        strin = strin[:-1]
    tmpbin = ''
    for i in range(len(strin)):
        tmpbin += inttobin(table.index(strin[i]), 6)
    remainder = len(tmpbin) / 8
    for i in range(remainder):
        res += chr(int(tmpbin[i*8:i*8+8], 2))
    return res
```