攻防世界stegobase64详解

原创

入山梵行 于 2020-09-10 16:34:50 发布 632 收藏 3

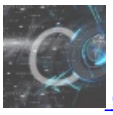
分类专栏： 初学 CTF misc 文章标签： python 算法 密码学

本文链接：https://blog.csdn.net/Mrs_H/article/details/108514352


初学 同时被 3 个专栏收录

8 篇文章 0 订阅

订阅专栏


CTF

32 篇文章 0 订阅

订阅专栏


misc

1 篇文章 0 订阅

订阅专栏

base64stego与base64隐写的那些事

今天在做攻防世界的时候，偶然间遇到一道misc的题，名字叫base64stego,题目原型是俄罗斯Olympic_ctf 的一道2014年的杂项。从题目中可以了解到这是一道考察base64隐写的一道题。但是首先这道题目考察的是对伪加密的应用。

伪加密解决方法

winhex处理

首先先说一下有关winhex的相关操作吧。

压缩源文件数据区：

50 4B 03 04：这是头文件标记

14 00：解压文件所需 pkware 版本

00 00：全局方式位标记（有无加密）

08 00：压缩方式

5A 7E：最后修改文件时间

F7 46：最后修改文件日期

压缩源文件目录区：

50 4B 01 02：目录中文件文件头标记

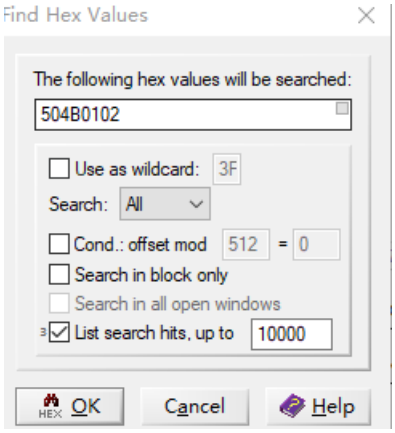
3F 00：压缩使用的 pkware 版本

14 00：解压文件所需 pkware 版本

00 00：全局方式位标记（有无加密，这个更改这里进行伪加密，改为09 00打开就会提示有密码了）

08 00：压缩方式

简单了解一下上面的相关16进制数据，咱们知道要搜索50 4B 01 02，所以操作一下



然后就，

```
36 7A C7 00 3A B1 B6 F5  2F AD E8 CC FC DB F8 0F
50 4B 01 02 3F 03 14 03  09 00 08 00 68 BF 9B 48
FE 32 7D 4B E9 0D 00 00  B5 1B 00 00 09 00 24 00
00 00 00 00 00 00 20 80  ED 81 00 00 00 00 73 74
```

刚刚上面说过了，伪加密的原理是将目录区的全方位标记的00 00改成了00 09所以咱们把他改回来，咱们可以看到14 03 后有一个非常明显的00 09，那就是他了，改完之后然后就可以进去拿文件了。

民间处理方法

额。。。怎么说呢。其实很高兴你能从上面那些奇奇怪怪的东西看到这里，想要学明白上面那些东西是要耗费一些时间的，但是呢，再后来我又看到了一个很简单的办法。简单到什么程度呢？一句话来说，有手就行。那废话不多说，上图（额。。。应该没有人不知道Winrar吧，不会吧不会吧）

打开WinRAR然后选择你从攻防世界下载的压缩文件（单击，选中即可，不用点开）然后按下上图中的修复按钮。于是乎，你会得到一个WinRAR的rebuilt文件，这时候你就可以直接进rebuilt文件中，拿到你想要的东西了。（有没有感觉真就右手就行。

base64隐写解密

其实在写这篇在写文章的之前，我去查了很多很多的大佬博客，但是不知道为什么，他们的代码大多数不能在我的电脑上运行。这让我很苦恼，所以干脆我就直接去学了原理，自己写一个。

base64编码原理

咱们用一个经典的例子来解释它的原理（摘自百度百科）

在电脑里，内存1个字节占8位

转前： s 1 3

先转成ascii：对应 115 49 51

然后转成8位2进制： 01110011 00110001 00110011

然后拼接在一起 011100110011000100110011

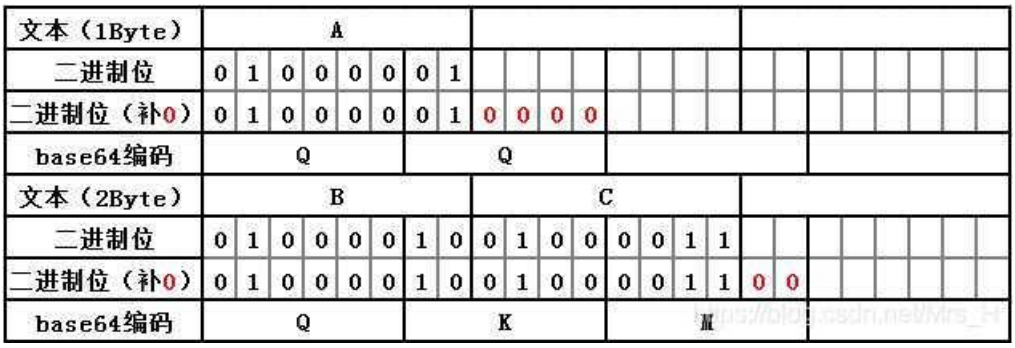
分成6各个一组 011100 110011 000100 110011

然后再将上面的二进制数字算出来

得到 28 51 4 51

查对下照表 c z E z

看不太懂的话，可以再来个图稍微了解一下。



上面的图中所有空白的二进制位都会用0进行补位然后下面出来一到两个等号。

base64解码原理

方便起见咱们以刚才图片的第二个例子举例

把 Base64 字符串去掉等号, 转为二进制数(QKM= -> QKM -> 010000100100001100).

从左到右, 8 个位一组, 多余位的扔掉, 转为对应的 ASCII 码（01000010 01000011 00 -> 扔掉最后2位 -> 1000010 01000011-> BC)

有没有发现什么？

对了，最后两个标红的0在解密中并没有起到任何的作用。但是咱们不能这两个标红零的后面，进行隐写，因为那样会影响到等号的数量，说白了就是破坏了原先的结构。好，从现在开始我们的任务只有一个，提取和咱们刚才所说过的标红的0地位相当的二进制数字，并将他们转化为字符。

base64隐写解密

先上代码

```
import base64
b64chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'#参考注释1
with open('1.txt', 'rb') as f:
    bin_str = ''
    for line in f.readlines():
        stegb64 = str(line, "utf-8").strip("\r\n")#参考注释2
        rowb64 = str(base64.b64encode(base64.b64decode(stegb64)), "utf-8").strip("\r\n")#参考注释3
        equalnum = stegb64.count('=')
        if equalnum:#参考注释4
            offset = abs(b64chars.index(stegb64.replace('=','')[-1])-b64chars.index(rowb64.replace('=','')[-1]))
#参考注释5
            bin_str += bin(offset)[2:].zfill(equalnum * 2)
        print(''.join([chr(int(bin_str[i:i + 8], 2)) for i in range(0, len(bin_str), 8)]))
```

注释正文：

1.在这一行我们的工作就是简单的做一个base64的参照表，后面会解释作用。

2.这里我想说一下我踩的坑，我第一次写这段代码的时候，并没有写`'\r'`我只写了`’\n‘`但是后来我发现输出的时候好像还是有回车，于是就又加了一个（但是据说好像用默认，两个就都删？（混乱

3.要加“utf-8”不加会报错

4.刚刚从我们的分析过程中可以看到，如果在一串base64编码的字符串中，没有等号是不可能隐藏数据的，所以这里要判断一下

5.这里要算一下偏移量，可能就这么说偏移量大家可能不是很懂，我就用简单通俗的话来讲吧。如果咱们不去隐写数据的话，最后补位的一定全是零，那么用咱们得到的隐写数据的二进制减去那些补位是零的二进制数据（他们两个的头部都是从前面继承的），那么就会得到咱们隐写进去的非零二进制数据。

好啦，我想说的就是这些，如有不足欢迎指正。