# 工具

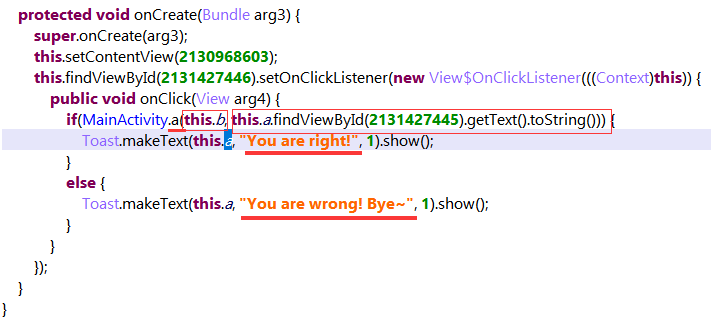
Jeb，ida

# 知识点

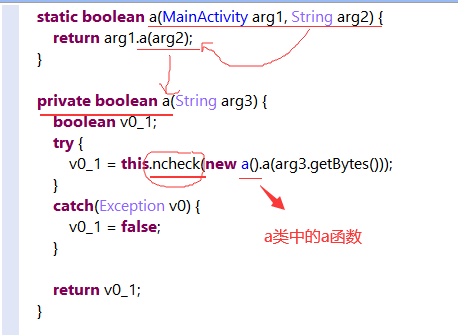
1. .So
2. Base64替换密码表

# 流程分析

1. MainActivity



用MainActivity类中的a函数去验证

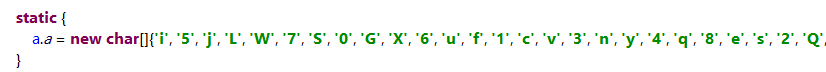


这里的ncheck（）函数是一个native函数，等下去找so库中对应的函数

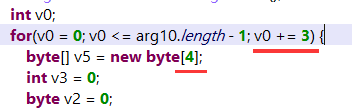
1. a类是一个base64的替换表加密。

几个明显的特征如下：

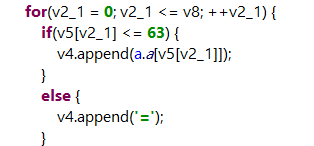
1. 定义了一个密码表



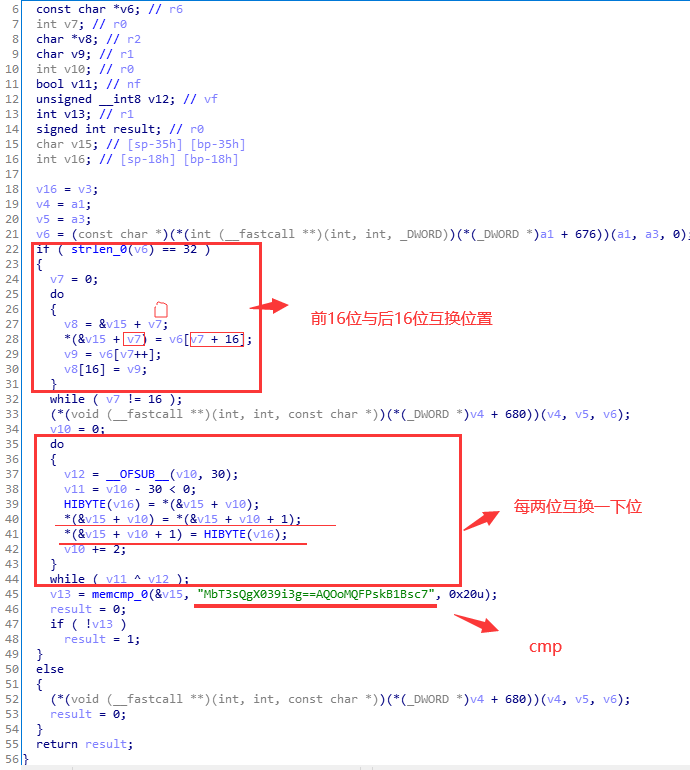
1. 每次循环的长度为3，定义了长度为4的数组



1. 长度不够补充“=”



1. 将apk解压拿到so库函数，将libnative.so用ida打开



# 解题思路

1. 从字符串首位开始每两位交换一次位置
2. 前16位和后16位交换位置
3. 变异的base64解密

脚本

|  |
| --- |
| import base64  *# 每两位交换一次位置*  str1 = list("MbT3sQgX039i3g==AQOoMQFPskB1Bsc7")  str\_twoChange = ""  for i in range(0,len(str1),2):      str\_twoChange += str1[i+1] + str1[i]  *# 前16位与后16位交换位置*  str\_F16\_B16\_Change = ""  str2 = list(str\_twoChange)  str\_F16\_B16\_Change = ''.join(str2[i] for i in range(16,32))+''.join(str2[i] for i in range(0,16))  *# 变异base64解密*  base\_new = ['i', '5', 'j', 'L', 'W', '7', 'S', '0', 'G', 'X',   '6', 'u', 'f', '1', 'c', 'v', '3', 'n', 'y', '4', 'q', '8',   'e', 's', '2', 'Q', '+', 'b', 'd', 'k', 'Y', 'g', 'K', 'O',   'I', 'T', '/', 't', 'A', 'x', 'U', 'r', 'F', 'l', 'V', 'P',   'z', 'h', 'm', 'o', 'w', '9', 'B', 'H', 'C', 'M', 'D', 'p',    'E', 'a', 'J', 'R', 'Z', 'N']  base\_new\_str = ''.join(i for i in base\_new)  base\_original\_str = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'  flag = base64.b64decode(str\_F16\_B16\_Change.translate(str.maketrans(base\_new\_str,base\_original\_str)))  print(flag) |

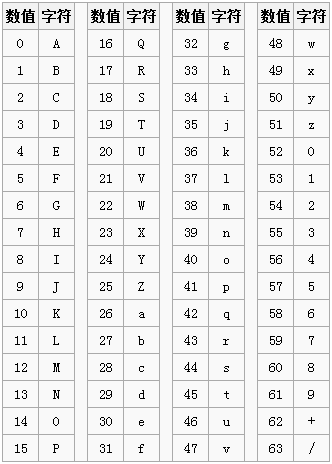
# Base64编码的补充介绍

## base64编码的作用和诞生

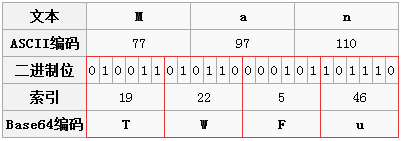
为什么会有Base64编码呢？因为有些网络传送渠道并不支持所有的字节，例如传统的邮件只支持可见字符的传送，像ASCII码的控制字符就 不能通过邮件传送。这样用途就受到了很大的限制，比如图片二进制流的每个字节不可能全部是可见字符，所以就传送不了。最好的方法就是在不改变传统协议的情 况下，做一种扩展方案来支持二进制文件的传送。把不可打印的字符也能用可打印字符来表示，问题就解决了。Base64编码应运而生，Base64就是一种 基于64个可打印字符来表示二进制数据的表示方法。

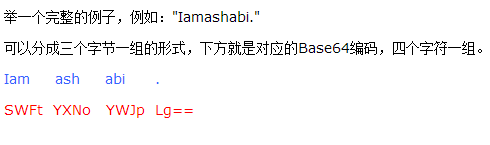
## Base64编码原理

看一下Base64的索引表，字符选用了"A-Z、a-z、0-9、+、/" 64个可打印字符。数值代表字符的索引，这个是标准Base64协议规定的，不能更改。64个字符用6个bit位就可以全部表示，一个字节有8个bit 位，剩下两个bit就浪费掉了，这样就不得不牺牲一部分空间了。这里需要弄明白的就是一个Base64字符是8个bit，但是有效部分只有右边的6个 bit，左边两个永远是0。



那么怎么用6个有效bit来表示传统字符的8个bit呢？8和6的最小公倍数 是24，也就是说3个传统字节可以由4个Base64字符来表示，保证有效位数是一样的，这样就多了1/3的字节数来弥补Base64只有6个有效bit 的不足。你也可以说用两个Base64字符也能表示一个传统字符，但是采用最小公倍数的方案其实是最减少浪费的。结合下边的图比较容易理解。Man是三个 字符，一共24个有效bit，只好用4个Base64字符来凑齐24个有效位。红框表示的是对应的Base64，6个有效位转化成相应的索引值再对应 Base64字符表，查出"Man"对应的Base64字符是"TWFU"。说到这里有个原则不知道你发现了没有，要转换成Base64的最小单位就是三个字节，对一个字符串来说每次都是三个字节三个字节的转换，对应的是Base64的四个字节。这个搞清楚了其实就差不多了。





但是转换到最后你发现不够三个字节了怎么办呢？愿望终于实现了，我们可以用两 个Base64来表示一个字符或用三个Base64表示两个字符，像下图的A对应的第二个Base64的二进制位只有两个，把后边的四个补0就是了。所以 A对应的Base64字符就是QQ。上边已经说过了，原则是Base64字符的最小单位是四个字符一组，那这才两个字 符，后边补两个"="吧。其实不用"="也不耽误解码，之所以用"="，可能是考虑到多段编码后的Base64字符串拼起来也不会引起混淆。由此可见 Base64字符串只可能最后出现一个或两个"="，中间是不可能出现"="的。下图中字符"BC"的编码过程也是一样的。

