

# BugkuCTF 部分题解(持续更新)

原创

置顶 [z.volcano](#) 已于 2022-04-18 13:34:52 修改 1763 收藏 3

分类专栏: [misc # bugku](#) 文章标签: [ctf bugku](#)

于 2021-12-30 18:37:19 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: [https://blog.csdn.net/weixin\\_45696568/article/details/122243131](https://blog.csdn.net/weixin_45696568/article/details/122243131)

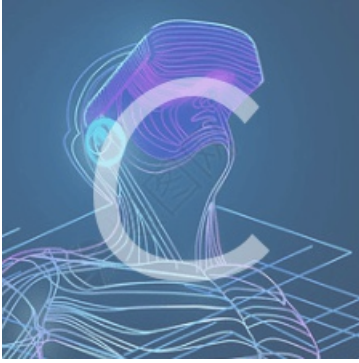
版权



[misc](#) 同时被 2 个专栏收录

1 篇文章 1 订阅

订阅专栏



[bugku](#)

3 篇文章 6 订阅

订阅专栏

之前做的题在[BugkuCTF 部分题解\(一\)](#)

## 佛系更新

4月18日更新了yst的小游戏

4月15日更新了where is flag 5

4月14日更新了图片里的英文

4月12日更新了RSSSSSA、TLS

**bugku**

佛系更新

MISC

yst的小游戏

图片里的英文

TLS

图片里的中文

miko不会javav啊

学不会javav我哭辽

黑客的照片

look

美丽的烟火

disordered\_zip

easypicture

再也没有纯白的灵魂

easy\_python

简单套娃DX

做题要细心-1

简单取证1

南城旧梦

成果狗成果狗

Crypto

RSSSSSA

where is flag 5

## MISC

### yst的小游戏

bmp文件，考虑 `lsb` 或 `wbstego`，后者解出一串hex，转字符得到

```
#coding=utf-8

import os,threading

class user():
    MAX_HP=400
    HP=400
    MP=200
    Danger=30
    Defence=20
class HP():
    name_1='小瓶生命药水'
    HP_1=50
    name_2='大瓶生命药水'
    HP_2=100
    name_3='满血药水'
    HP_3=user.HP
class MP():
```

```

name_1='小瓶魔法药水'
MP_1=50
name_2='大瓶魔法药水'
MP_2=100
name_3='满魔药水'
MP_3=user.MP
class USE():
    name_1='普通攻击'
    MP_1=0
    Danger_1=user.Danger
    name_2='磨刀石'
    MP_2=10
    Danger_2=user.Danger*2
    name_3='鸡汤'
    MP_3=20
    Danger_3=user.Danger*3
    name_4='攻击强化'
    MP_4=50
print('开始游戏'.center(100,'*'))
print('yst的属性值\t生命值%s\t魔法值%s\t攻击力%s\t防御值%s\t'%(user.HP,user.MP,user.Danger,user.Defence))
level=129
class Boss():
    HP=300
    Danger=30
    Defence=20
    HP=HP+20*level
    MP=400
    Danger=Danger+3*level
    Defence=Defence+level*4
user.Danger=user.Danger-Boss.Defence
Boss.Danger=Boss.Danger-user.Defence
if user.Danger <=0:
    user.Danger=1
if Boss.Danger <=0:
    Boss.Danger=1
print('树木的属性值\t生命值%s\t魔法值%s\t攻击力%s\t防御值%s\t'%(Boss.HP,Boss.MP,Boss.Danger,Boss.Defence))
print('战斗开始'.center(100,'*'))
ran=0
while 1:
    ran=ran+1
    print('可用技能\t%s\t攻击力%s\t%s\t攻击力%s\t%s\t攻击力%s\t%s\t攻击力%s\t'%(USE.name_1,user.Danger,USE.name_2,user.D
anger*2,USE.name_3,user.Danger*3,USE.name_4,user.Danger*2))
    print('可用药水\t%s\t回复生命值%s\t%s\t回复生命值%s\t%s\t回复生命值%s\t\n\t\t%s\t回复魔法值%s\t%s\t回复魔法值%s\t%s\t回复
魔法值%s\t'%(HP.name_1,HP.HP_1,HP.name_2,HP.HP_2,HP.name_3,HP.HP_3,MP.name_1,MP.MP_1,MP.name_2,MP.MP_2,MP.name_3
,MP.MP_3))
    use=input('使用:')
    if USE.name_1 in use:
        user.MP=user.MP-USE.MP_1
        if user.MP<USE.MP_1:
            print('魔力不足,自动使用普通攻击')
            Boss.HP=Boss.HP-user.Danger
            print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_1,user.Danger))
        else:
            user.MP=user.MP-USE.MP_1
            print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_1,user.Danger))
            Boss.HP=Boss.HP-user.Danger
    elif USE.name_2 in use:
        if user.MP<USE.MP_2:
            print('魔力不足,自动使用普通攻击')

```

```

    Boss.HP=Boss.HP-user.Danger
    print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_1,user.Danger))
else:
    user.MP=user.MP-USE.MP_2
    print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_2,user.Danger*2))
    Boss.HP=Boss.HP-user.Danger*2
elif USE.name_3 in use:
    if user.MP<USE.MP_3:
        print('魔力不足,自动使用普通攻击')
        Boss.HP=Boss.HP-user.Danger
        print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_1,user.Danger))
    else:
        user.MP=user.MP-USE.MP_3
        print('yst使用了%s 对怪物造成了%s伤害'%(USE.name_3,user.Danger*3))
        Boss.HP=Boss.HP-user.Danger*3
elif USE.name_4 in use:
    if user.MP<USE.MP_4:
        print('魔力不足,自动使用普通攻击')
        Boss.HP=Boss.HP-user.Danger
        print('yst使用了%s 对树木造成了%s伤害'%(USE.name_1,user.Danger))
    else:
        user.MP=user.MP-USE.MP_4
        user.Danger=user.Danger*2
        print('yst使用了%s 对树木造成了%s伤害'%(USE.name_4,user.Danger))
        Boss.HP=Boss.HP-user.Danger
elif HP.name_1 in use:
    user.HP=user.HP+HP.HP_1
    print('yst使用了%s 恢复了%s生命值'%(HP.name_1,HP.HP_1))
elif HP.name_2 in use:
    user.HP=user.HP+HP.HP_2
    print('yst使用了%s 恢复了%s生命值'%(HP.name_2,HP.HP_2))
elif HP.name_3 in use:
    user.HP=user.HP+HP.HP_3
    print('yst使用了%s 恢复了%s生命值'%(HP.name_3,HP.HP_3))
elif MP.name_1 in use:
    user.MP=user.MP+MP.MP_1
    print('yst使用了%s 恢复了%s魔力值'%(MP.name_1,MP.MP_1))
elif MP.name_2 in use:
    user.MP=user.MP+MP.MP_2
    print('yst使用了%s 恢复了%s魔力值'%(MP.name_2,MP.MP_2))
elif MP.name_3 in use:
    user.MP=user.MP+MP.MP_3
    print('yst使用了%s 恢复了%s魔力值'%(MP.name_3,MP.MP_3))
else:
    print('没有该道具 自动使用普通攻击')
    print('yst使用了%s 对树木造成了%s伤害'%(USE.name_1,user.Danger))
    Boss.HP=Boss.HP-user.Danger
user.HP=user.HP-Boss.Danger
print('树木对你造成了%s伤害'%(Boss.Danger))
print('yst\t剩余生命值%s\t攻击力%s\t\t防御力%s\t魔法值%s'%(user.HP,user.Danger,user.Defence,user.MP))
print('树木\t剩余生命值%s\t攻击力%s\t防御力%s\t\t\n'%(Boss.HP,Boss.Danger,Boss.Defence))
print('第%s回合,结束!!!'%ran)
if Boss.HP<=0 and user.HP<=0:
    print('平局!!!')
    break
if Boss.HP <=0:
    print('树木被砍翻了,你赢了!!!\t')
    break
if user.HP <=0:
    print('你死了!!!\t')

```

```
print( "开始游戏" )
break
```

```
*****开始游戏*****
yst的属性值  生命值400    魔法值200    攻击力30  防御值20
树木的属性值  生命值2880   魔法值400    攻击力397  防御值536
*****战斗开始*****
可用技能  普通攻击  攻击力1    磨刀石  攻击力2    鸡汤  攻击力3    攻击强化  攻击力2
可用药水  小瓶生命药水  回复生命值50    大瓶生命药水  回复生命值100    满血药水  回复生命值400
          小瓶魔法药水  回复魔法值50    大瓶魔法药水  回复魔法值100    满魔药水  回复魔法值200
使用 :|
```

CSDN @z.volcano

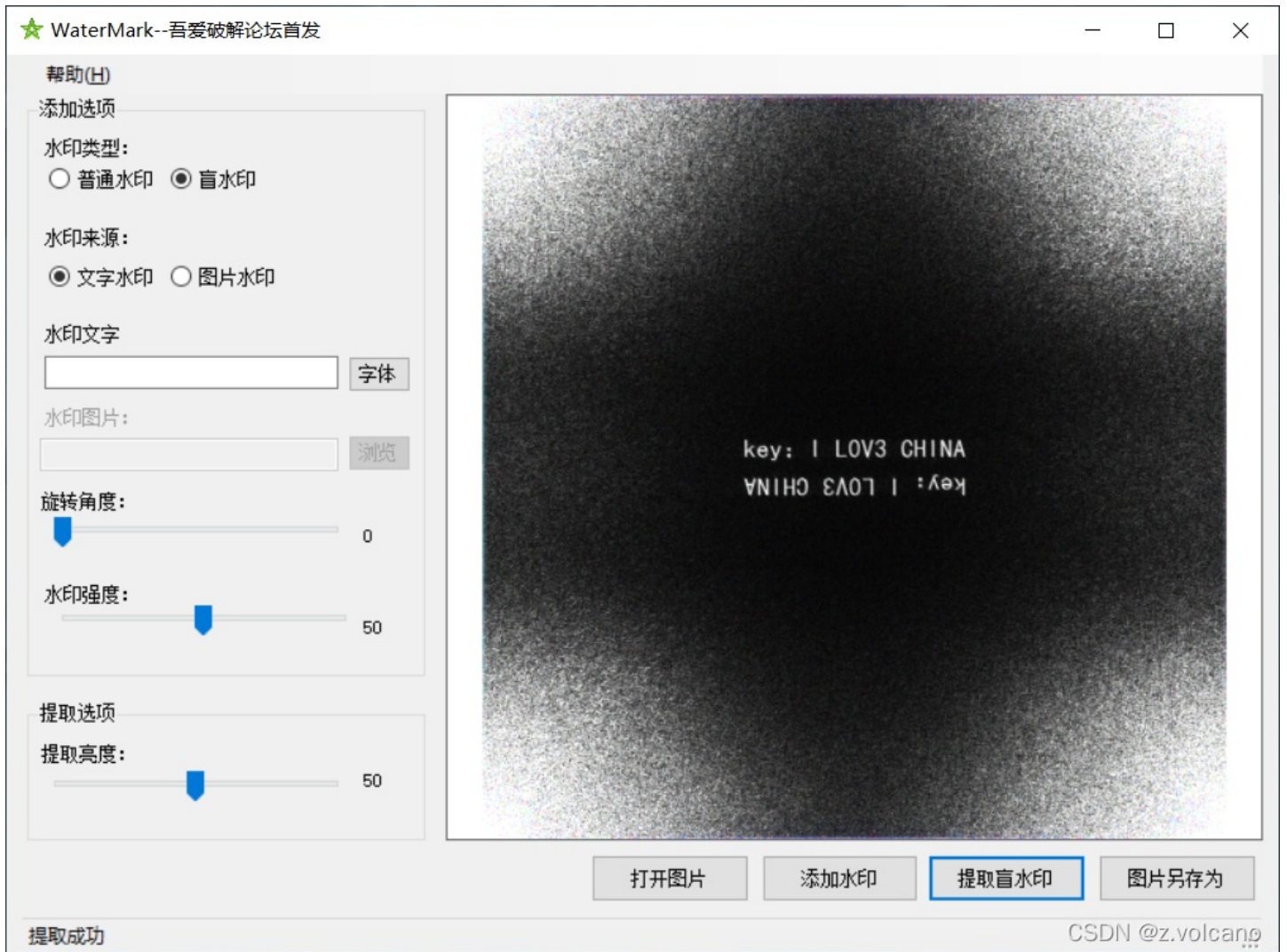
测试一下发现每次攻击强化 耗蓝50，使 普通攻击和技能伤害\*2，同时会造成伤害  
最后鸡汤的伤害会是最高的，用作最后几下补伤害

初始血量400，每次被攻击掉血397，第一回合之后剩3滴血；使用满血药水再被攻击血量变成6，如此往复 使血量超过397，然后  
如果当前蓝量不够，则先补蓝，否则释放攻击强化。每次补蓝或者释放技能之后，都执行多次回血操作。

简单模拟这个过程，就可以得到答案。

## 图片里的英文

先拿到密码 I LOV3 CHINA



这个密码还不知道用到哪里，作者给了文件，直接跳到第二步，有五张图片，结合题目描述，需要从中破解出压缩包的密码

这是游戏sifu里的东西，能认出其中有 **智** 和 **义**，其实是把传统的 **仁义礼智信** 改成了 **人义尊智诚**，这里的顺序是 **人尊智义诚**，不过这五个字所有的排列组合都不是密码。

题目描述是 **而且仇人拥有5种不同能力，你能破解他们吗。**，所以压缩包的密码是 **木火水金土**，对应五个关卡反派的能力



解压之后有两张图片，因为未加密的图片是**bmp**格式，优先试 **LSB** 和 **wbstego**，后者解密出 **key: 虾仁猪心**






解压出 flag.png，是疯狂动物城里的一幕，结合题目意思可以知道需要找出这张图对应的字幕，是 You know you love me



对应题目描述，这句台词大概是密码，应该是某种隐写，不过试了一圈都没解出来...

原来不能直接用图片里的台词，得全换成小写，还得去掉空格，才是正确的密码...

 想养一只粉色的猫 2天前  
最后英文是全小写无空格的

 回复  0

最后一步是有密码的lsb隐写，就很迷惑，空格是可以作为lsb隐写的密码的一部分的，把台词处理过后作为密码，当成一个脑洞点...不理解

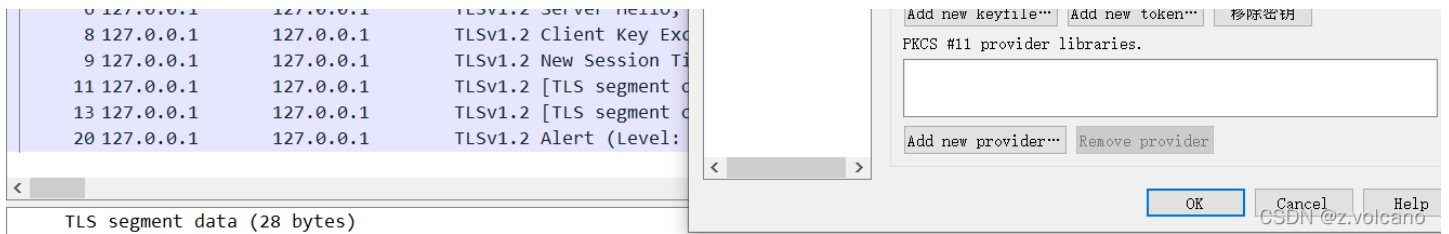
```
python2 lsb.py extract flag.png out.txt youknowyouloveme
```

## TLS

看题目名可以知道本题考察TLS报文的解密，这里用wireshark演示

首先 编辑-> 首选项->RSA密钥，把给的密钥文件选中

No.	Source	Destination	Protocol	Info
2	127.0.0.1	127.0.0.1	TCP	443 → 48238 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=1886
3	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1011772811 Win=0 Len=0
5	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
7	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1011772811 Win=0 Len=0
10	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1011772811 Win=0 Len=0
12	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
14	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
16	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
18	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1011772811 Win=0 Len=0
19	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
21	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1011772811 Win=0 Len=0
22	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1011772811 Win=0 Len=0
4	127.0.0.1	127.0.0.1	TLSv1.2	Client Hello
6	127.0.0.1	127.0.0.1	TLSv1.2	Server Hello



然后得到flag

No.	Source	Destination	Protocol	Info
15	127.0.0.1	127.0.0.1	HTTP	GET /key.html HTTP/1.1
17	127.0.0.1	127.0.0.1	HTTP	HTTP/1.1 200 OK (text/html)
1	127.0.0.1	127.0.0.1	TCP	48238 → 443 [SYN] Seq=0 Win=43690 Len=0
2	127.0.0.1	127.0.0.1	TCP	443 → 48238 [SYN, ACK] Seq=0 Ack=1 Win=
3	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=1 Ack=1 Win=43776
5	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1 Ack=100 Win=437
7	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=100 Ack=1022 Win=
10	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=414 Ack=1276 Win=
12	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1276 Ack=462 Win=
14	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1276 Ack=503 Win=
16	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1276 Ack=529 Win=
18	127.0.0.1	127.0.0.1	TCP	48238 → 443 [ACK] Seq=529 Ack=1580 Win=
19	127.0.0.1	127.0.0.1	TCP	443 → 48238 [FIN, ACK] Seq=1580 Ack=529
21	127.0.0.1	127.0.0.1	TCP	48238 → 443 [FIN, ACK] Seq=556 Ack=1581
22	127.0.0.1	127.0.0.1	TCP	443 → 48238 [ACK] Seq=1581 Ack=557 Win=
4	127.0.0.1	127.0.0.1	TLSv1.2	Client Hello
6	127.0.0.1	127.0.0.1	TLSv1.2	Server Hello, Certificate, Server Hello
8	127.0.0.1	127.0.0.1	TLSv1.2	Client Key Exchange, Change Cipher Spec
9	127.0.0.1	127.0.0.1	TLSv1.2	New Session Ticket, Change Cipher Spec,
11	127.0.0.1	127.0.0.1	TLSv1.2	[TLS segment of a reassembled PDU]
13	127.0.0.1	127.0.0.1	TLSv1.2	[TLS segment of a reassembled PDU]

TLS segment data (28 bytes)

- > [2 Reassembled TLS segments (254 bytes): #17(226), #17(28)]
- > Hypertext Transfer Protocol
- ▼ Line-based text data: text/html (1 lines)
  - The key is 39u7v25n1jxkl123\n

0070	32 30 31 35 20 31 34 3a 31 35 3a 35 34 20 47 4d	2015 14: 15:54 GM
0080	54 0d 0a 45 54 61 67 3a 20 22 31 63 2d 35 32 34	T··ETag: "1c-524
0090	66 39 38 33 37 38 64 34 65 31 22 0d 0a 41 63 63	f98378d4 e1"··Acc
00a0	65 70 74 2d 52 61 6e 67 65 73 3a 20 62 79 74 65	ept-Rang es: byte
00b0	73 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74	s··Conte nt-Lengt
00c0	68 3a 20 32 38 0d 0a 43 6f 6e 74 65 6e 74 2d 54	h: 28··C ontent-T
00d0	79 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 0d 0a	ype: tex t/html··
00e0	0d 0a 54 68 65 20 6b 65 79 20 69 73 20 33 39 75	··The ke y is
00f0	37 76 32 35 6e 31 6a 78 6b 6c 31 32 33 0a	

Frame (370 bytes) | Decrypted TLS (226 bytes) | Decrypted TLS (28 bytes) | Reassembled SSL (254 bytes)

## 图片里的中文

提示 rot47，压缩包注释信息里的密文拿去解密得到 Add 5D honey to me，给密文 MD5加密 一下得到压缩包密码 e950c89788db4042193830f49d273512。我的脑洞仅局限于把5D转为十进制，可能对应某个base家族，结果是93



咱也不懂为啥是MD5，而且不理解为什么提示是 `Add 5D honey to me`，但不是给这句话MD5加密，而是给密文MD5加密

```
Input
p55 ds 9@?6J E@ >6

Output
e950c89788db4042193830f49d273512
CSDN @z.volcano
```

解压得到 `password.txt` 和 `key.png`，前者内容是 `AecvWpF9YinwVc7tvKfCGwt28oYiucpd5MEWhghGaaLWKfgTEyWz5hmRREEi`，这里继续试...

[bugku](#)的在线工具、[cyberchef](#)等工具均未解出，最后在某网站成功解 `base58`

解出 `iEeWwc2oGX/DzzdaM1/tdvaWGRodocuXoPMJpAUyMaI=`，应该是AES，下面找密码，这是神剧《让子弹飞》里的名场面，这里的台词应该是 `杀人还要诛心?`，结合题目名可以知道这就是密码。



接着就是到各个在线网站疯狂试，最后在[某网站](#)成功解出

值得一提的是，选项一样的情况下，去其他几个常用的网站会报错(包括作者给的网站)，总之没有技巧，试就完了

A screenshot of an online AES decryption tool interface. The interface is light green and white. At the top, there are several input fields and dropdown menus. The 'AES加密模式' (AES encryption mode) is set to 'ECB'. The '填充' (padding) is set to 'ansix923'. The '密钥长度' (key length) is set to '192位'. The '密钥' (key) is '杀人还要诛心?'. The '输出' (output) is set to 'base64'. Below these fields is a large text area containing the encrypted string: `iEeWwc2oGX/DzzdaM1/tdvaWGRodocuXoPMJpAUyMaI=`. At the bottom, there are four buttons: 'AES加密' (AES encryption), 'AES解密' (AES decryption), '复制结果' (copy result), and '清空所有' (clear all). Below the buttons is another large text area containing the decrypted string: `bugku{[obscured text]}`. In the bottom right corner, there is a small watermark that reads 'CSDN @z.volcano'.

## miko不会jvav啊

第一步仍然是找密码，依旧使用strings命令，把流量包里所有字符串存起来作为字典

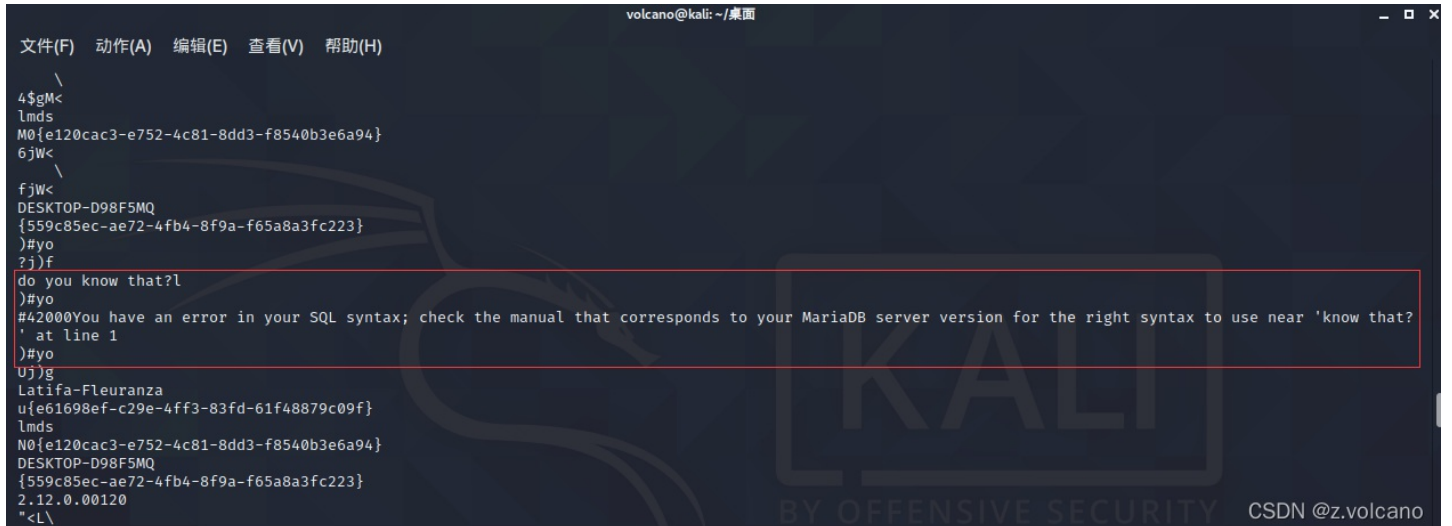
这里是 rar5.0 加密，用Accent RAR Password Recovery进行爆破，得到密码 `Latifa-Fleuranza` (非预期)

剩下步骤同 [学不会jvav我哭辽](#)

## 学不会jvav我哭辽

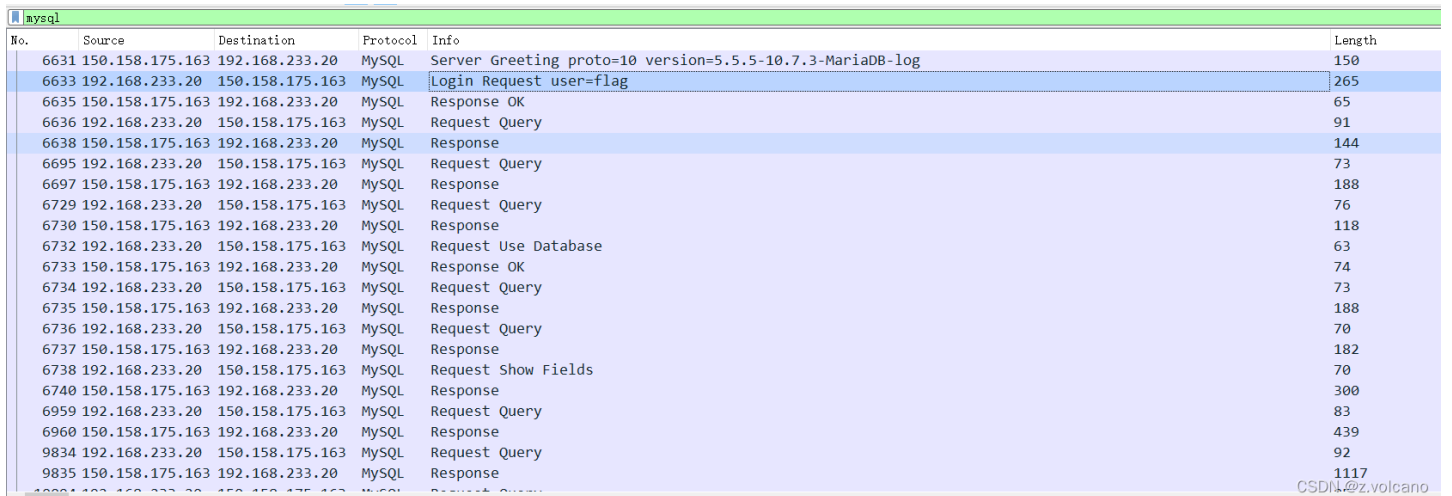
拿到一个流量包和加密的压缩包，先用 `strings` 命令看一下流量包里有那些字符串，挨个试了一下都不是密码...

不过其中有这样一句：



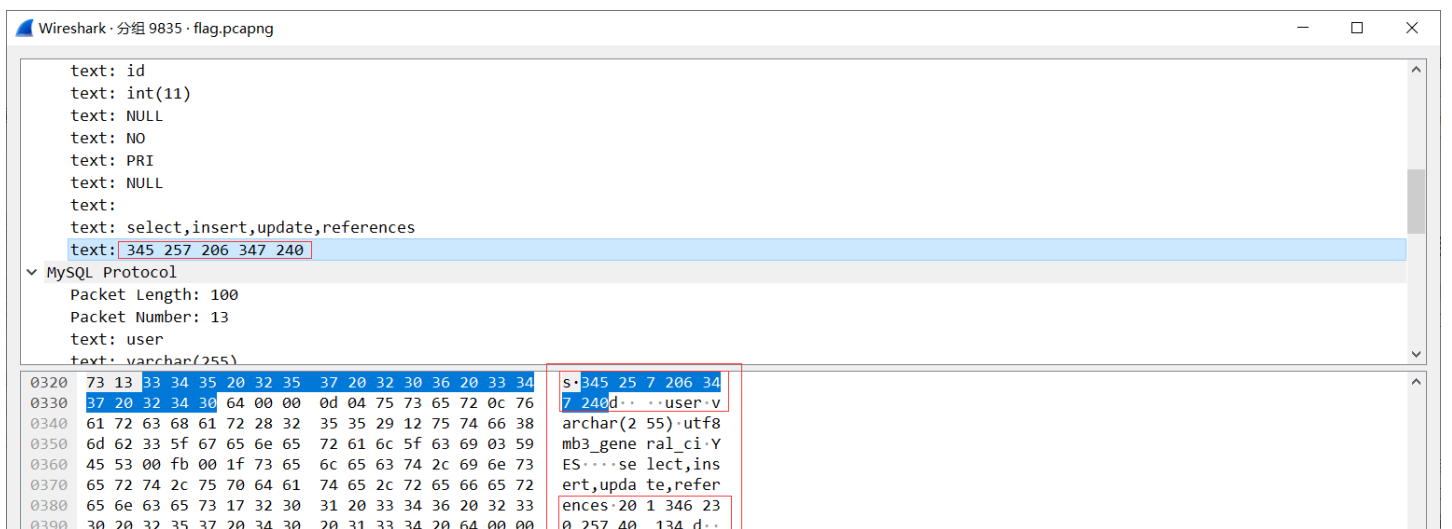
```
volcano@kali: ~/桌面
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
4$gM<
lmds
M0{e120cac3-e752-4c81-8dd3-f8540b3e6a94}
6jW<
fjW<
DESKTOP-D98F5MQ
{559c85ec-ae72-4fb4-8f9a-f65a8a3fc223}
)#yo
?)f
do you know that?l
)#yo
#42000You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'know that?'
' at line 1
)#yo
Uj)g
Latifa-Fleuranza
u{e61698ef-c29e-4ff3-83fd-61f48879c09f}
lmds
N0{e120cac3-e752-4c81-8dd3-f8540b3e6a94}
DESKTOP-D98F5MQ
{559c85ec-ae72-4fb4-8f9a-f65a8a3fc223}
2.12.0.00120
"<L\
BY OFFENSIVE SECURITY CSDN @z.volcano
```

于是找到对应的mysql数据流，一个个看



No.	Source	Destination	Protocol	Info	Length
6631	150.158.175.163	192.168.233.20	MySQL	Server Greeting proto=10 version=5.5.5-10.7.3-MariaDB-log	150
6633	192.168.233.20	150.158.175.163	MySQL	Login Request user=flag	265
6635	150.158.175.163	192.168.233.20	MySQL	Response OK	65
6636	192.168.233.20	150.158.175.163	MySQL	Request Query	91
6638	150.158.175.163	192.168.233.20	MySQL	Response	144
6695	192.168.233.20	150.158.175.163	MySQL	Request Query	73
6697	150.158.175.163	192.168.233.20	MySQL	Response	188
6729	192.168.233.20	150.158.175.163	MySQL	Request Query	76
6730	150.158.175.163	192.168.233.20	MySQL	Response	118
6732	192.168.233.20	150.158.175.163	MySQL	Request Use Database	63
6733	150.158.175.163	192.168.233.20	MySQL	Response OK	74
6734	192.168.233.20	150.158.175.163	MySQL	Request Query	73
6735	150.158.175.163	192.168.233.20	MySQL	Response	188
6736	192.168.233.20	150.158.175.163	MySQL	Request Query	70
6737	150.158.175.163	192.168.233.20	MySQL	Response	182
6738	192.168.233.20	150.158.175.163	MySQL	Request Show Fields	70
6740	150.158.175.163	192.168.233.20	MySQL	Response	300
6959	192.168.233.20	150.158.175.163	MySQL	Request Query	83
6960	150.158.175.163	192.168.233.20	MySQL	Response	439
9834	192.168.233.20	150.158.175.163	MySQL	Request Query	92
9835	150.158.175.163	192.168.233.20	MySQL	Response	1117

发现多组可疑的数字



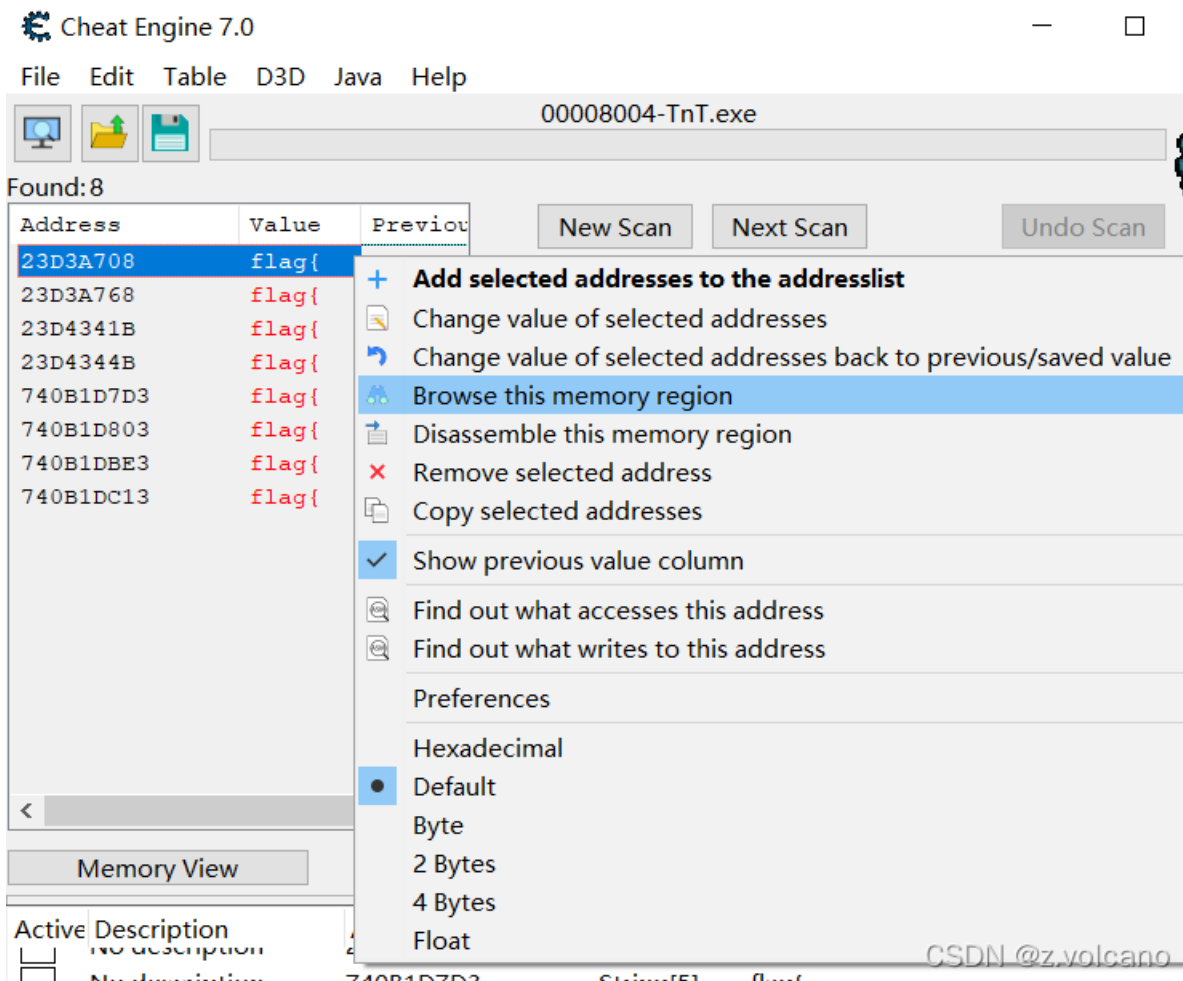
```
Wireshark · 分组 9835 · flag.pcapng
text: id
text: int(11)
text: NULL
text: NO
text: PRI
text: NULL
text:
text: select,insert,update,references
text: 345 257 206 347 240
MySQL Protocol
Packet Length: 100
Packet Number: 13
text: user
text: varchar(255)
0320 73 13 33 34 35 20 32 35 37 20 32 30 36 20 33 34 s:345 25 7 206 34
0330 37 20 32 34 30 64 00 00 0d 04 75 73 65 72 0c 76 7 240d...-user-v
0340 61 72 63 68 61 72 28 32 35 35 29 12 75 74 66 38 archar(2 55)-utf8
0350 6d 62 33 5f 67 65 6e 65 72 61 6c 5f 63 69 03 59 mb3_gene ral_ci-Y
0360 45 53 00 fb 00 1f 73 65 6c 65 63 74 2c 69 6e 73 ES...se lect,ins
0370 65 72 74 2c 75 70 64 61 74 65 2c 72 65 66 65 72 ert,upda te,refer
0380 65 6e 63 65 73 17 32 30 31 20 33 34 36 20 32 33 ences-20 1 346 23
0390 30 20 32 35 37 20 34 30 20 31 33 34 20 64 00 00 0 257 40 134 d...
```

```
03a0 0e 06 70 61 73 73 77 64 0c 76 61 72 63 68 61 72 ..passwd varchar
03b0 28 32 35 35 29 12 75 74 66 38 6d 62 33 5f 67 65 (255)ut f8mb3_ge
03c0 6e 65 72 61 6c 5f 63 69 03 59 45 53 00 fb 00 1f neral_ci YES...
03d0 73 65 6c 65 63 74 2c 69 6e 73 65 72 74 2c 75 70 select,insert,up
03e0 64 61 74 65 2c 72 65 66 65 72 65 6e 63 65 73 15 date,references:
03f0 35 36 20 35 36 20 31 35 35 20 31 35 31 20 31 35 56 56 15 5 151 15
0400 33 20 31 35 37 4b 00 00 0f 04 64 61 74 65 08 64 3 157K...date-d
0410 61 74 65 74 69 6d 65 fb 03 59 45 53 00 fb 00 1f atetime YES...
0420 73 65 6c 65 63 74 2c 69 6e 73 65 72 74 2c 75 70 select,insert,up
```

看着像八进制的数，转字符得到密码

The screenshot shows a 'Recipe' application interface. On the left, there is a 'From Octal' section with a 'Delimiter Space' button. The main area is split into 'Input' and 'Output' sections. The 'Input' section shows a single line of 84 octal characters: 345 257 206 347 240 201 346 230 257 40 134 56 56 155 151 153 157 111 154 61 56 56 57. The 'Output' section shows the result: 密码是 \..mikoIl1../. The application also displays metadata like 'length: 84', 'lines: 1', 'start: 84', 'end: 84', and 'length: 0' for the input, and 'time: 0ms', 'length: 17', 'lines: 1' for the output.

解压出一个exe文件，按照题目提示需要再一步逆向，但是没找到exe转jar的方法，于是尝试使用逆向工具，ida没搞出来，ce可以



## 黑客的照片

文件尾部有一段flag: `flag{m1s}`，和压缩包的数据

formost提取出两个压缩包，其中一个提示，简单rsa拿到密码 `I_love_mumuz1!`

```

from Crypto.Util.number import *
import gmpy2

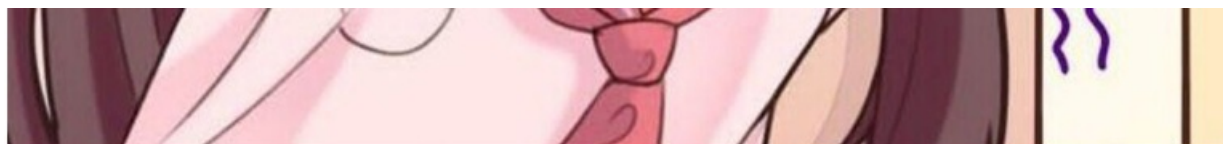
e = 65537
p = 164350308907712452504716893470938822086802561377251841485619431897833167640001783092159677313093192408910634
1515872177745304247807992106067884232351611457183384462784125948755770305853482416773991153515948843417300309677
75904826577379710370821510596437921027155767780096652437826492144775541221209701657278949
q = 107494571486621948612091613779149137205875732174969005765729543731117585892506950289230919634697561179755186
3116175246603288365808686169586869876116142330130777055195289464907210650023428684035570701767520157672062631303
91554820965931893485236727415230333736176351392882266005356897538286240946151616799180309
c = 172105717681128595126067638716024320302580099226540889895663287273811908496845134751248133647780512006509440
8516038736820519009411424847079555046641194088992338301424669862452475743116313384445191004980498535902165589356
4081185136250014784383020061202277758202995568045817822133418748737332056585115499621035958182697568687907469775
3020762718244695640255050646928845249911237037919339069501704346276031543633275347903359600551999999423621526762
4007913422491101327287356171052279416368093831172045432519727958991865338637874300446408807155286060630237859502
490924209652484068178676906868066093033640022862042786586612

n= p*q
d = int(gmpy2.invert(e , (p-1) * (q-1)))
m = pow(c ,d ,n)
#passwd:I_love_mumuz1!
print(long_to_bytes(m))

```

另一个压缩包需要手动提取，解压后得到 `flag2.jpg` 和 `flag3.png`

`flag2.jpg`修改图片高度



# 3\_1s\_eas3\_bu1\_mi

`flag3.png`是stegpy隐写

```
(volcano@kali)-[~/桌面/stegpy]
└─$ stegpy ../flag3.png
sc_d0n't_love_me}
```

## look

后缀改为zip，解压出look.bmp，stegsolve打开，分别在R0、G0、B0通道发现隐写痕迹，提取得到flag，zsteg应该也能一把梭

## 美丽的烟火

先解zip伪加密

```
java -jar ZipCenOp.jar r 美丽的烟火.zip
```



**Recipe**

**From Base64**

Alphabet  
A-Za-z0-9+/=

Remove non-alphabet chars

**From Base58**

Alphabet  
123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefgh...

Remove non-alphabet chars

**Input**

am5QwDVwNVp0ZkRKdw14U3NFSw==

**Output**

t\_hp1ass\_s1wsd

CSDN @z.volcano

- 栅栏密码
- 凯撒密码
- 凯撒移位(中文版)
- 维吉尼亚密码
- 摩斯电码
- 百度/Google/网页字符
- MD5
- 置换密码
- 替代密码

清空 拼音 频率 去空格 每隔 2 个字符 加空格 横/竖 大写 小写 倒序 词倒序

替换 计算 十进制 > 十六进制

## 栅栏密码

在下面的文本框输入明文或密文，点加密或解密，文本框中即可出现所得结果

加密 解密 列举加密 列举解密 栏数: 2  只列举完整匹配的

密文框:

```
t_hp1ass_s1wsd  
2栏:  
ts__hsp1lwassd  
7栏:  
th1s_1s_passwd
```

图片的文件尾有 `stegpy:shumu`，所以用stegpy解密，密码是shumu，解出一串字符 `aZgs8ImPpQ0zO3CVA/wIUvq/M7X8C33ptNZSW2Blenc=`

最后找一个在线网站解密就行

个别网站不太行，可以试试下面这两个

```
http://tool.chacuo.net/cryptaes/  
https://the-x.cn/zh-cn/cryptography/Aes.aspx
```

## disordered\_zip

binwalk看一下可以发现有两文件，foremost只能梭出来png

```
(volcano@kali)-[~/桌面]  
└─$ binwalk -e disordered_zip.zip
```

DECIMAL	HEXADECIMAL	DESCRIPTION
290475	0x46EAB	End of Zip archive, footer length: 22
290497	0x46EC1	PNG image, 50 x 50, 8-bit/color RGBA, non-interlaced

png图片是一个缺了两定位符的二维码，简单补全扫码得到 `psw:yeQFPDS6uTaRasdvwLkoEZq00iUk`，这个地方很有迷惑性，psw会被认为是password的简写，从而认为后面这个就是密码，其实还得解一手栅栏(在线网站)

- 栅栏密码
- 凯撒密码
- 凯撒移位(中文版)
- 维吉尼亚密码
- 摩斯电码
- 百度/Google/网页字符
- MD5
- 置换密码
- 替代密码

清空 拼音 频率 去空格 每隔 2 个字符 加空格 横/竖 大写 小写 倒序 词倒序

替换 计算 十进制 > 十六进制 转换

## 栅栏密码

在下面的文本框输入明文或密文，点加密或解密，文本框中即可出现所得结果

加密 解密 列举加密 列举解密 栏数: 2  只列举完整匹配的

密文框:

```
psw:yeQFPDS6uTaRasdvwLkoEZq00iUk  
2栏:  
passwd:vyweLQKFoPEDZSq60uOTiaURk  
4栏:  
pPaEsDsZwSdq:6v0yuwOeTLiQaKUFROk  
8栏:  
pyPuawEOseDTsLZiwQSadKqU:F6Rvo0k  
16栏:  
pwyQPSuaadwKEqOUs:eFD6TRsvLoZ0ik
```

CSDN @z.volcano

得到了密码 `vyweLQKFoPEDZSq60uOTiaURk`，回头把zip拿出来。这里很明显是加密的zip文件的特征(可以找一个zip加密文件对照

着看)

```
4:6E20h: 4A 5A 16 08 87 7D 97 69 03 87 88 08 2F CD 93 81 02..+}~1.g .71 a
4:6E30h: A8 F4 25 A5 78 60 50 48 5A 8C 50 4B 07 08 BB 74 ..ô%¥x`PHZEPK..»t
4:6E40h: C8 1C 0F 6E 04 00 D6 89 04 00 50 4B 01 02 1F 00 È..n..Ö%..PK....
4:6E50h: 14 00 09 00 63 00 5B 98 8A 53 BB 74 C8 1C 0F 6E ....c.[~ŠS»tÈ..n
4:6E60h: 04 00 D6 89 04 00 04 00 2F 00 00 00 00 00 00 00 ..Ö%..../.....
4:6E70h: 20 00 00 00 00 00 00 00 66 6C 61 67 0A 00 20 00 .....flag..
4:6E80h: 00 00 00 00 01 00 18 00 CC 43 18 7B B5 ED D7 01 .....îC.{µí×.
4:6E90h: 42 A9 62 7D B5 ED D7 01 95 75 AC 5A B5 ED D7 01 B@b}µí×.•u-Zµí×.
4:6EA0h: 01 99 07 00 01 00 41 45 03 08 00 50 4B 05 06 00 .™....AE...PK...
4:6EB0h: 00 00 00 01 00 01 00 61 00 00 00 4C 6E 04 00 00 .....a....Ln...
4:6EC0h: 00 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 .CSDN.@z.volcarrn
```

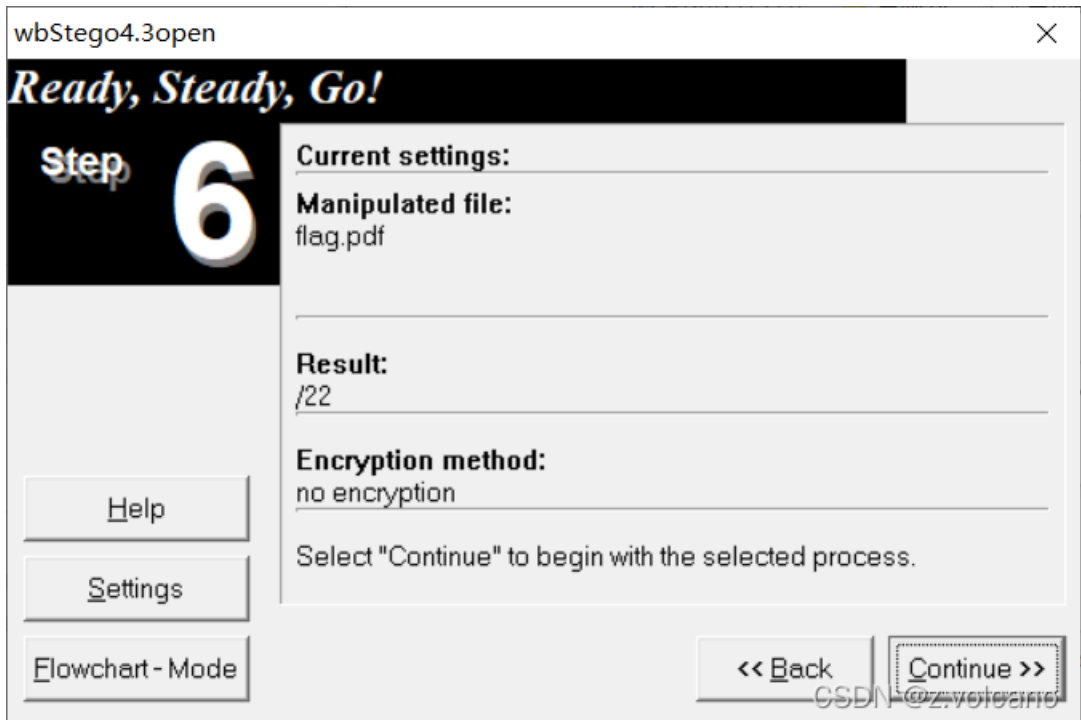
这里是zip加密文件的文件头位置，在最前面补上 504B 即可。

```
编辑方式: 十六进制(H) 运行脚本 运行模板
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 03 04 14 00 09 00 63 00 5B 98 8A 53 BB 74 C8 1C .....c.[~ŠS»tÈ.
0010h: 0F 6E 04 00 D6 89 04 00 04 00 0B 00 66 6C 61 67 .n..Ö%....flag
0020h: 01 99 07 00 01 00 41 45 03 08 00 79 5C 31 C5 CB .™....AE...y\lÄÈ
0030h: 41 87 06 56 E0 EF 91 92 64 2C 38 9B 43 3B 81 B5 A+.Vàì' 'd,8>C;.µ
```

输入密码，解压出的flag其实是个pdf文件，修复文件头后补上后缀

```
编辑方式: 十六进制(H) 运行脚本 运行模板
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 25 50 44 46 2D 31 2E 35 20 20 09 20 20 09 20 20 %PDF-1.5 . .
0010h: 0D 0A 25 B5 B5 B5 B5 20 20 20 20 20 20 20 0D %uuuu
```

这里是pdf文件，相关隐写很容易想到 [webstego4open](#)，解密得到flag



## easypicture

图片尾有额外数据，提取出加密的压缩包，爆破得到密码 0707

得到俩图片和 [hint:你喝过咖啡吗?你见过彩色的图片吗?](#) , 这里其实提示的是 [stegsolve](#)

先看ciphertext.png, 文件尾有个提示 [AES,ECB](#) , 用stegsolve简单翻找没有找到, 直接用 [ztseg](#)

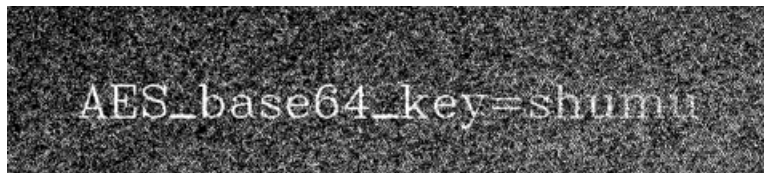
```
zsteg -a ciphertext.png
```

结合提示找到一串密文 [rcPr04H36Qj+7vt9YFA2KbRyD0yEg+y7mTAQHC82CBM=](#)

接着找key.jpg中藏着的密钥, 挨个试工具, 最后发现是 [java盲水印](#)

项目地址: <https://github.com/ww23/BlindWaterMark/releases>

使用命令: [java -jar BlindWatermark.jar decode -c bingbing.jpg decode.jpg](#)



最后到[在线网站](#)解密即可

## 再也没有纯白的灵魂

题目名即为提示, 后一句是 [自人类堕落为半兽人](#) , 直接观察密文也能看出来这是兽音译者

~呜嗷

```
BBBUBGUUBUUKBBKGBGUBBBKBKBUUBUGBBBUGUKGUBUUKBUUGBGUBBUBKBKUBBKBBBBUGUKUUBUUKBKKBGUBBBBKBUU  
BKUBBBUUBBBUBUUKUGGGBGUBBBKKBKUUBGBBBBUUBBBUBUUKUGGGBGUBBBKKBKUUBKGBBBUGUGBUBUUGBUBGBGUBUKU  
KBKUUBUBBBBUGBBUUBUUKUUB啊
```

这种加密由 [嗷呜啊~](#) 四个字符组成, 这里替换成了 [BUGK](#) , 找出规律替换回去

```
B-嗷 U-呜 G-啊 K--~
```

不过解出来的是乱码，和正常加密的密文对比一下



可以发现这个位置少了一个啊，加上之后就能解出来了

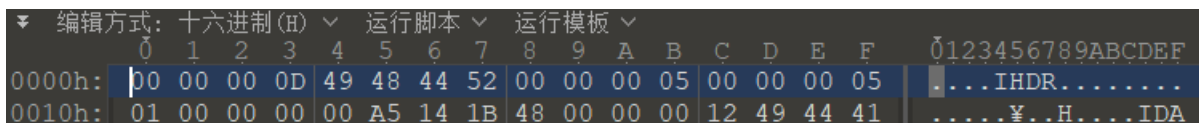
## easy\_python

每次运行 `game.py`，先会读取 `.level` 文件中的内容，即玩家现在的等级，每次战斗后玩家会升一级，攻击力和防御力提升，而怪物的血量和防御力是不会变的，大概算一下就行。

不想算的话就修改 `.level` 文件中的内容，然后运行 `game.py`，最后的答案应该是 `56390`，base64编码后套上flag{}即可

## 简单套娃DX

下载得到8080个无后缀文件，随使用010editor打开几个，发现都是png文件，不过都存在一些问题，先写脚本给这些文件加上png后缀



```
import os

path = r'E:\desktop\XD' # 文件夹所在地址
for i in os.listdir('./XD'): # 文件夹名字
    oldname = os.path.join(path,i)
    newname = os.path.join(path,i+'.png')
    os.rename(oldname,newname)
```

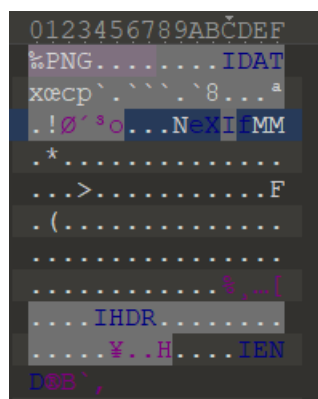
加上后缀后只有一部分文件可以正常打开，剩下的就需要修改，前面发现部分文件 **缺失了8字节文件头**，写脚本对这种文件进行修复，成功修复了 `1046` 个文件，说明还有很多图片是其他种类的错误，需要一一找出

```
import os

out = 'output' #存放修复后的文件的文件夹
if not os.path.exists(out):
    os.makedirs(out)
for i in os.listdir('./XD'): #文件夹名字
    file = open('./XD/'+i, 'rb').read()
    if file[1:4] != b'PNG':
        file = int(0x89504E470D0A1A0A).to_bytes(8, 'big') + file

    f = open('./output/' + i + '.png','wb')
    f.write(file)
    f.close()
```

比如说有一些文件是IDAT块跑到IHDR块前面去了



这个过程最好是拿一个正常的png文件过来进行比对，同时使用010editor的模板功能，可以节省很多时间，最后发现有如下几类错误

- 1.文件头缺失8字节(89504E470D0A1A0A) 如aAnLOyuK文件
- 2.IDAT块跑到IHDR块前面。如aAWyDGMo文件
- 3.IHDR块的长度位和标记位替换为00。如aAFRoNjh文件
- 4.图片的宽高被修改了(正确的应该是5\*5) 如abFdgotH文件
- 5.Color type被修改成不是0的数。如ABmgLXTn文件
- 6.IDAT块的长度被改为0。如aAWrlqiP文件
- 7.IDAT块标记位被删除。如abQZgrjo文件

八神师傅还是很贴心的，每个文件最多一种错误，而且翻了前十个文件就能找出所有种类的错误，写出最终脚本。



```

import os

out = 'output' #存放修复后的文件的文件夹
if not os.path.exists(out):
    os.makedirs(out)
for i in os.listdir('./XD'): #文件夹名字
    file = open('./XD/'+i, 'rb').read()
    if file[1:4] != b'PNG': #文件头缺失
        file = int(0x89504E470D0A1A0A).to_bytes(8, 'big') + file
    elif file[8:16] == b'\x00\x00\x00\x00\x00\x00\x00\x00': #IHDR块的长度位和标记位替换为00
        file = file[:8] + int(0x0000000D49484452).to_bytes(8, 'big') + file[16:]
    elif file[33:37] == b'\x00\x00\x00\x00' and file[37:41] == b'IDAT': #IDAT块的长度被改为0
        len_idat = file.index(b'eXIf') - 49
        file = file[:33] + int(len_idat).to_bytes(4, 'big') + file[37:]
    elif file[12:16] != b'IHDR': #IHDR块位置错误
        file = file[:8] + file[-37:-12] + file[8:-37] + file[-12:]
    elif file[16:24] != b'\x00\x00\x00\x05\x00\x00\x00\x05': #图片宽高被修改
        file = file[:16] + int(0x0000000500000005).to_bytes(8, 'big') + file[24:]
    elif file[1:4] == b'PNG' and file[12:16] == b'IHDR' and file[37:41] != b'IDAT': #IDAT块标记位被删除
        file = file[:37] + b'IDAT' + file[37:]
    elif file[1:4] == b'PNG' and file[12:16] == b'IHDR' and file[25] != b'\x00': #Color type被修改
        file = file[:25] + b'\x00' + file[26:]
    f = open('./output/' + i, 'wb')
    f.write(file)
    f.close()

```

随后得到了8080个5\*5的小方块，挨个查看它们的exif信息，可以发现这两个值均不相同，写脚本挨个输出，发现x对应的最大值是449，y对应的是19，应该是坐标

## IFD0

X分辨率	166
Y分辨率	18

写脚本画图

```

import os
from PIL import Image

img = Image.new('RGB', (450*5, 20*5), (255, 255, 255)) #乘以5是因为一张小图是5*5
for i in os.listdir('./output'): #文件夹名字
    file = open('./output/'+i, 'rb').read().hex()
    x = int(file[-64:-56], 16) #x坐标
    y = int(file[-48:-40], 16) #y坐标
    x*=5
    y*=5
    im = Image.open('./output/'+i)
    img.paste(im, (x,y,x+5,y+5))

img.save('out.png')

```

bugku{Hjpp0Po7om0n5tRos3s9Uipped4LjopHo6i4}

## 做题要细心-1

这个位置有一串二进制，拿出来

注意到file.gif文件尾有额外数据，看特征是 文件头不全的gif文件

名称	值	开始	大小	颜色
> struct GIFHEADER GifHeader		0h	6h	Fg: B
> struct LOGICALSCREENDESCRIPTOR LogicalScreenDescriptor		6h	7h	Fg: B
> struct GLOBALCOLORTABLE GlobalColorTable		Dh	180h	Fg: B
> struct DATA Data		18Dh	71650h	Fg: B
> struct TRAILER Trailer		717DDh	1h	Fg: B

补全文件头，010editor打开，在类似的位置看到另一串二进制

```

0000h: 47 49 46 38 39 61 SE 01 SE 01 E6 00 00 00 00 00 GIF89a^..æ....
0010h: FF FF FF C6 82 84 48 30 31 9A 28 2F 6A 4E 52 B5 ŸŸŸE,,H01š(/jNRµ
0020h: B1 B2 FA F1 F4 64 5F 61 84 00 52 93 8D 91 B4 6D ±²úñód_a,,.R".´m
0030h: A2 4A 33 47 DA B0 D6 52 00 52 24 11 24 70 6A 70 çJ3GÚ°ÖR.R$.şpjp
0040h: 84 82 84 34 22 35 32 11 38 52 00 84 50 4C 53 DF ,,,"4"52.8R.,,PLSß
0050h: DC E8 84 82 C6 00 00 84 00 00 52 00 00 12 11 11 Ÿè,,.E.,.,.,.R.,.,.
0060h: 23 22 22 35 44 44 57 BB BB BC 61 66 94 30 33 46 #"5DDW»»»af"03F
0070h: 50 54 69 17 24 57 B4 BD CF 2B 55 99 A5 C7 FF 00 PTi.şW´½İ+U™¥ÇŸ.
0080h: 55 A5 84 C7 FF 52 A6 E7 00 82 C6 1F 51 68 A5 E3 U¥,,ÇŸR|ç.,.E.Qh¥ă
0090h: FF 65 93 A1 30 45 4B 00 11 15 10 22 26 33 B3 C5 Ÿe";0EK...."&3³Ă
00A0h: 21 33 33 C6 FF FF E7 FF FF 11 22 11 C6 FF C6 33 !33EŸŸçŸŸŸ.".EŸE3
00B0h: 44 32 76 88 61 2E 33 21 52 55 00 FF FF C6 FF FF D2v^a.3!RU.ŸŸEŸŸ
00C0h: E7 FF FF EE 44 44 42 A4 A4 A3 84 82 52 D8 C8 40 çŸŸîDDB™™£,,.RØE@
00D0h: 9D 91 67 28 22 10 FA F1 D7 E4 D3 A7 FF E3 A5 C6 .`g("úñ×ăóşŸŸăŸE
00E0h: 82 00 54 47 2D E7 A6 52 FF C7 84 A5 55 00 F4 D3 ,.TG-ç|RŸÇ,,¥U.ôó
00F0h: B5 C6 82 52 E4 9A 69 F4 DB D2 7A 4B 3D CF B8 B1 µE,RăšiôŸŸZK=İ,±
0100h: DF A7 99 CD 3C 23 B1 94 90 E1 6B 5D 8E 72 6F 27 ß$™İ<#±".ák}žro'
0110h: 0F 0D A3 5F 5B 84 00 00 52 00 00 5F 1A 1A 11 06 ..£ [,,.R.,.,.
0120h: 06 37 22 22 55 44 44 D8 D0 D0 EE EE EE DD DD DD .7""UDDØĐİİİŸŸŸ
0130h: CC CC CC AA AA AA 99 99 99 88 88 88 77 77 77 66 İİİ^a a a™™™™^w^wwf
0140h: 66 66 55 55 55 33 33 33 22 22 22 11 11 11 FF FF ffUUU333""...ŸŸ
0150h: FF 01 01 11 11 01 10 00 10 01 11 01 01 01 00 01 Ÿ.....
0160h: 11 01 11 11 01 AA AA 00 00 00 00 00 00 00 00 00 ..... a a .....
0170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0180h: 00 00 00 00 00 00 00 00 00 00 00 00 00 21 FF 0B ..CSDN@z.volcano
0190h: 4E 45 54 53 43 41 50 45 32 2E 30 03 01 00 00 00 NETSCAPE2.0

```

连在一起，转字符得到 `Nnb77_buG}`，查看第二个gif的exif信息，发现一个字符串，base64解码得到 `{Ax_S`

### GIF

GIFVersion	89a
ImageWidth	350
ImageHeight	350
HasColorMap	Yes
ColorResolutionDepth	7
BitsPerPixel	7
BackgroundColor	0
AnimationIterations	Infinite
FrameCount	40
Duration	2.00 s

### XMP-x

XMP工具kit	Image::ExifTool 12.34
----------	-----------------------

### XMP-tiff

Artist	e0F4X1M=
--------	----------

第二个图片的第20帧有一串字符，md5解密得到 `bugku`，至此flag找全了



## 简单取证1

下载得到windows系统下一个目录，获取用户名和密码需要用 `SAM` 和 `system` 两个文件。

把SAM和SYSTEM文件放到Win32文件夹下，运行mimikatz，执行命令

```
mimikatz 2.1.1 x86 (oe.eo)
version - Display some version informations
cd - Change or display current directory
localtime - Displays system local date and time (OJ command)
hostname - Displays system local hostname

mimikatz # lsadump::sam /sam:SAM /system:SYSTEM
Domain : DESKTOP-VBBTMVS
SysKey : 37f0b72b8ef4052d5a305281c2c8905c
Local SID : S-1-5-21-2468344919-3152572563-1510310172

SAMKey : 6bbc5d51b8b9609e241bba9a2558048e

RID : 000001f4 (500)
User : Administrator

RID : 000001f5 (501)
User : Guest

RID : 000001f7 (503)
User : DefaultAccount

RID : 000001f8 (504)
User : WDAGUtilityAccount
Hash NTLM: 8adf83b531e1cdadc8d16b206d87a4d5

RID : 000003e8 (1000)
User : administrator-QQAazz
Hash NTLM: 5f9469a1db6c8f0dfd98af5c0768e0cd

mimikatz #
```

CSDN @z.volcano

5f9469a1db6c8f0dfd98af5c0768e0cd

解密

ntlm

forensics

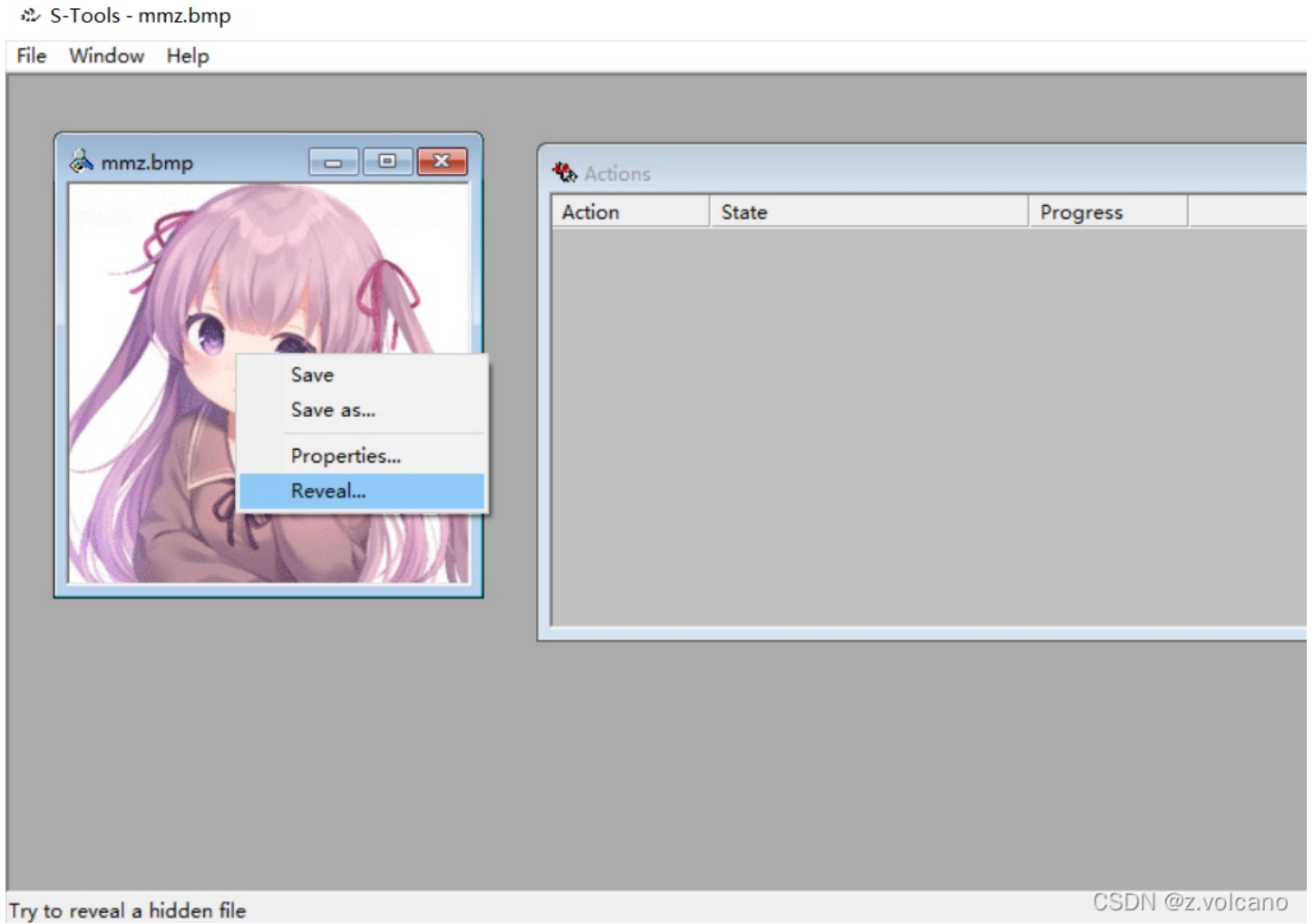
CSDN @z.volcano

所以 `flag{administrator-QQAazz_forensics}`

## 南城旧梦

mmz.bmp文件尾有一段 `DE@@=<6J:DB625K4`，rot47解码后得到 `stoolkeyisqeadzc`

意思是使用工具 `stool`，密码是 `qeadzc`，这个工具使用的话要直接把图片拖进去，解密的话是这个选项，输入密码就可以解出 `Key.txt`，得到把酒言欢.rar的密码 `mmzbudaiwowuwu~~~##`



倾听.txt是 `SNOW`隐写，解密可以得到密码表

(1) 1 2 3 4 5 6 ! @ # \$ % ^	(2) 1 2 3 4 5 6 p f f d e h	(3) 1 2 3 4 5 6 v f s a e y
(4) 1 2 3 4 5 6 / * - + . ?	(5) 1 2 3 4 5 6 F G S F H H	(6) 1 2 3 4 5 6 " " / \   :
(7) 1 2 3 4 5 6 r o k u g s	(8) 1 2 3 4 5 6 f J L Y G W	(9) 1 2 3 4 5 6 J O P N G W
(10) 1 2 3 4 5 6 F S A S D T		



游戏规则在 [九菊.docx](#) 中有描述

你玩过骰子吗？这是酒局必不可少的。

MMZ 和路人乙是两姐妹花，一天，乙突然嘴馋想喝酒，乙和 MMZ 说要不要去喝酒，MMZ 说就你那样，看我喝不死你。

下面分别是 MMZ 和乙的骰子图：



(MMZ)



(乙)

游戏开始：

MMZ: 三个二

MMZ: .....

MMZ: XXXXX

MMZ: 给爷喝！

问: XXXXX 中 MMZ 应该说什么？

乙: .....

乙: 四个四

乙: .....

Hint: "....."没有特殊意义，问题对应密码表即为压缩包密码。想要获取密码表快去[倾听叭](#)！

看不懂规则，所以用密码表中所有的字符组成字符串，用 [Accent RAR Password Recovery](#) 爆破

```
!@#%^&*pfdehvfdsaey/*-+.?FGSFHH/\|:rokugsfJLYGWJOPNGWFSASDT
```

最后跑出密码是 `%ee.`，解压出 `key` 和 `gpg` 两个无后缀文件。

key中的内容是

```
;3%5;3%5;3%5;2 ;5 ; ;%4;3%5;4 ;5
;2 ;open _,$0;2 ; ;5 ; ;3;2;1%2;4 ; ;5 ;3 ;5
;2 ;7 ;5 ; ;while (<_>) { ;5 ; ;5 ;1%6;2
;3%5;2%7; ;7 ;2;s;% ;g;3 ;1%7;2; ;5
;8 ; ;s'([; ](\d)' $1x$2 ; 'ge;; ; s'S';'g; ;4;
s; ;#;g; ; ;5 ; ;5 ; ;5 ; ;5 ; s;;; ;g; ;3 ;
;3%5;3%5;3%5;3%5;; print ;}%5;5%3;2
```

这个是混淆后的perl代码，直接运行得到 `secunet`

```
(volcano@kali)-[~/桌面]
└─$ perl key
#####  #####  #####  #   #   #####  #####  #
#   #   #   #   #   #   #   #   #   #   #
#   #   #   #   #   #   #   #   #   #####
#####  #####  #   #   #   #   #####  #
#   #   #   #   #   #   #   #   #   #
#   #   #   #   #   #   #   #   #   #   #
#####  #####  #####  #####  #   #   #####  ###
```

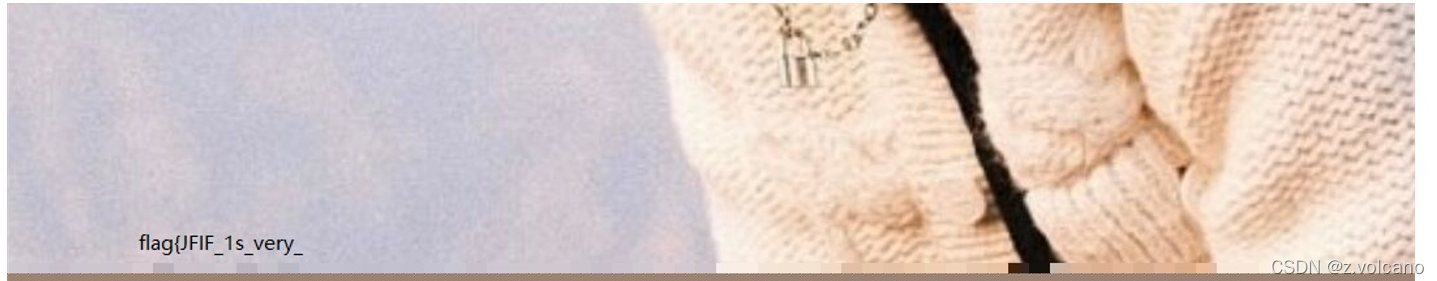
另一个gpg的话可以在kali下直接运行，输入密码 `secunet` 即可

```
(volcano@kali)-[~/桌面]
└─$ gpg -d gdg
gpg: AES encrypted data
gpg: 以 1 个密码加密
bugku{mmz_Won@nshEn_Kuailai_Jiji@n$%!!}gpg: 签名建立于 2021年03月31日 星期三 21时37分49秒 CST
gpg: 使用 RSA 密钥 0113DEE65D4D7969B1801A64A985FD02A220FC1F
gpg: 无法检查签名: No public key
```

## 成果狗成果狗

图片文件尾有额外数据，是 `一段base64+一个jpg文件的数据+另一段base64`，这里按照先后命名为 `图片1`、`图片2`、`bs1`、`bs2`。

两段base64解完再hex解密，没有什么特征...结合题目描述，把 `bs1` 经过base64解码后得到数据放到 `图片1文件尾部(插入FFD9之前)`，接着图片高度改高。



另一段同理。

## Crypto

### RSSSSSA

看特征很明显是低加密指数广播攻击，关于原理的话可以看翅膀师傅的博客，出题脚本和解题脚本可参考这篇。

### where is flag 5

base64解码后转hex得

```
1b 1f 04 00 0f 12 08 12 1f 1d 00 11 0b 13 14 1f 0b 00 1d 1c 11 1e 1f 01 11 0c 1a 14 14 1f 1b 03 18 08 16 11 1f 1
e 12 05 14 1d 19 1f 07 01 0d 10 15
```

因为每一组的第一位都是0或1

```
1100010111010111001111110101111101011111010111110101111101111100011
```

每七位转字符得bugku{c

不过找不到另一段flag了，转变思路，回过头base64解码后 **直接转二进制**

Recipe

From Base64

Alphabet  
A-Za-z0-9+/=

Remove non-alphabet chars

To Binary

Delimiter  
Space

Byte Length  
8

Input

GX8EAA8SCBI fHQARCxMUHwsAHRwRHh8BEQwaFBQfGwMYCByRHx4SBRQdGR8HAQ0QFQ==

Output

```
00011011 00011111 00000100 00000000 00001111 00010010 00001000 00010010 00011111 00011101 00000000 00010001
00010001 00001011 00010011 00010100 00011111 00001011 00000000 00011101 00011100 00010001 00011110
00011111 00000001 00010001 00001100 00011010 00010100 00010100 00011111 00011011 00000011 00011000
00001000 00010110 00010001 00011111 00011110 00010010 00000101 00010100 00011101 00011001 00011111
00000111 00000001 00001101 00010000 00010101
```

CSDN @z.volcano

取每一组二进制的 **后五位**，因为这里共有 **49组** 二进制，再结合前面转hex时得到一部分flag的过程，很容易想到：**竖着选取，7个一组**

```
s = '00011011 00011111 00000100 00000000 00001111 00010010 00001000 00010010 00011111 00011101 00000000 00010001
00001011 00010011 00010100 00011111 00001011 00000000 00011101 00011100 00010001 00011110 00011111 00000001 000
10001 00001100 00011010 00010100 00010100 00011111 00011011 00000011 00011000 00001000 00010110 00010001 0001111
1 00011110 00010010 00000101 00010100 00011101 00011001 00011111 00000111 00000001 00001101 00010000 00010101'
lt = [i[3:] for i in list(s.split())]

ind = 0

out = [''] * 35
for i in range(5):
    for j in range(len(lt)):
        if len(out[ind]) == 7:
            ind += 1
            out[ind] += lt[j][i]

for i in out:
    print(chr(int(i,2)),end="")
```