# PKU Geek Game Genesis

## Q群签到题

群公告中提供了一个`base64`编码的字符串，解码得`synt{J3ypbzr gb 0gu CXH ThThTh, rawbl gur tnzr!}`。已知前四位明文为`flag`，容易看出是`ROT-13`。

## 不含`main`的C程序

```
#define decode(m,s,u,t) m##s##u##t
#define begin decode(m,a,i,n)
```

## 本群已和谷歌达成战略合作

> 理科一号楼计算中心机房的门牌号

见 https://its.pku.edu.cn/pcroom.jsp

> 讲得好、作业少、考试水、给分高的课

见 https://courses.pinzhixiaoyuan.com/

> `HTCPCP-TEA` 协议

`418` 是 `I'm a teapot` ，`504` 才是 `Temporarily unavailable`。

> 在 2013 年 5 月 4 日，全世界共有多少可用的顶级域名（TLD）？

根据 http://web.archive.org/web/20130512163835/https://data.iana.org/TLD/tlds-alpha-by-domain.txt 确定数字，结合 https://en.wikipedia.org/w/index.php?date-range-to=2013-05-15&tagfilter=&title=List_of_Internet_top-level_domains&action=history 确认当周增删变化，确认答案是`317`。

## 图种

打开WinHex，在提供的GIF文件尾部，注意到`504B0304`和后面的strings，即获得压缩包。

GIF图片有八个关键帧，其中第二、四、六、八帧有灰白像素色块，一定是编码了信息。二四可以左右拼起来，骑缝的色块正好对上，六八也可以。然后再将二四和六八上下拼接，得到一个33x33的`+`形二维码。四个角上分别是原风景照的四个角的片段，占据了8x8的空间，左上角是风景照的右下角，提示应该把图片旋转180°。

看到缺了四个角，结合之前问答题cue汉信码，然后开始按汉信码的规范填四个角，然后找到古董扫码App去扫，未果，寻病终。绝望中试了试改成填入 QR Code 的回字（右下角留空），拿com.xiaomi.scanner没扫出来，拿微信给扫出来`https://www.pku.edu.cn/#hint=zip_password_is_fm2jbn2z6tOgl5le`，解压后即被颁发了Steganography学硕士学位。

## 2038银行

简单的算术分析告诉我们，手里547元资金（每个银行可以免息借19元），最多收支相抵，无法实现财富增值，只能走非法手段。

2038顾名思义就是把`MAXINT`溢出到负数。贷款额度给的是在本行净资产的十倍，利用三家银行轮流加杠杆，发现贷款上限是2,000,000,000元，并不能直接溢出到负数。不过，计息以后就可以溢出到负数了。多等几天，等到系统提示净资产够买flag了就平仓跑路。

# 假IAAA

心灵鸡汤故事告诉我们，公主选婿的最佳策略是放跑前`1/e`(~37%)个候选者，记下它们中的最大值，然后后面只要遇到超过前面的就下手。但是这个策略显然不足以保证题目要求的`60%`正确率，于是只能多玩几轮才能通过……

然后是破解假冒的IAAA，token是`JWT`的，使用`HS256`签名，目标是把里面编码的`student`改成`teacher`。试图把头部改成`"alg": None`发现过不去，那看来密钥是必须的。又用cracker跑brute force，也没跑出来。

题面里说他抄了份代码，没有研究并设置任何的可配置参数，于是去GitHub搜`jwt+teacher+hs256+student+Werkzeug`，未果，寻病终。在 https://jwt.io 演示站摆弄琢磨过程中，忽然发现，诶，怎么没输密钥他居然说verified？一看密钥真的就是空字符串。于是签好交了。汗颜……

# Flask

开始还以为是条件竞争什么的，想不太出来。然后每行查下来，发现`app.run`的时候忘记关闭debug模式，大喜，这不就题里说的"上线前的疏漏"么！试着让系统出错（填非法payload，比如`'''flag'''`），结果发现返回的是啥料都没有的4XX页面，一看哦套了`nginx`。

然后打开`/console`，发现有PIN，敲了个123-456-789进去发现已经太多错误尝试被锁掉了。开始搜PIN的生成原理，甚至试图`nbtstat`获得服务器的`hostname`和MAC地址（但是失败了，众所周知摆大早已禁`ping`）（而且套了nginx的话是不是搞到了也没有用）。

然后开始对session的签名跑字典，跑不出来。

然后就很绝望了，在那儿瞎摆弄琢磨，忽然发现flag取消输入（POST的json里传一个null）会冒500，而且能直接把debugger弹在响应里！定睛一看，debugger会返回前后各三四行的源码，于是祸水东引，POST了`{"action": null}`，成功用debugger套出了签名密钥，把篡改的session签了出来。

# 未来汇编语言

不难，但是汇编比较长，要花时间转写成伪代码（结果见`/mirai/mirai.c`），转写完就容易了，加密算法完全是可逆的，看图说话即可，见`/mirai/mirai.py`。（这算是让大一同学提前体验bomblab吗

# emoji排序

代码见`/emoji/emoji.py`。不会什么算法，就胡写一个奖惩算法，六十几个emoji而已，再怎么O(N^2)也能跑出来。

# 猜字典

不会什么算法，就brute-force了，密文从头至尾尝试译回，枚举遇到生词时的选择的词长。试译结束时字典里应当恰好有16个词，并且词频要和题中所给条件一样。

拿到题时先估算了一下，密文的平均词长在3出头。想着明文是一堆ASCII，感觉这个数其实比较低（我当时没有意识到并不是任意字符，只有0-9a-f！还以为是任意的字母，所以是和字节长度比的，一想这压缩好高效）。然后就福至心灵，把枚举词长上限设在8，觉得既然是压缩那总不能比原来长（？）

代码见`/translation/translation.c`。拿c写的（循环层数太多，实在不敢python），也没弄并行，单核跑三五小时居然真跑出来了。，运行得到字典（按在密文中出现先后顺序）是`['10000', '011', '10011', '0100', '101', '10001', '11', '100100', '01011', '000', '0011', '1001011', '10010100', '01010', '0010', '10010101']`。回头一看，居然真的把八位词长用完了，不禁暗暗为自己捏了把汗。然后翻译回明文，然后对Huffman肃然起敬。

# ZIP不动点

阅读提供的ZIP格式文档，知道被mask掉的四处里，第一处和第四处位于payload外，是被压缩文件（也就是`quine.zip`）的CRC32校验码。压缩方式是normal/deflating，没有加密什么的。校验码前后编码的是文件时间戳、以及压缩后大小等信息。第二处和第三处在压缩后的payload里，定睛一看，被马掉的内容前后仍然是文件时间戳和压缩后大小，和文件头里是一样的，相当于这块在deflating时完全untouched。因此大胆猜测文件被马掉的内容也是untouched，是和第一处第四处原样相同的CRC32。

那就很简单了，只需对这32位进行枚举。代码见`/crc32/crc32.c`。注意大小端问题，CRC32填入文件时是倒着填的，交flag时也是倒着交。