[logong] HGAME 2019 week-1 writeup

Web

第一题

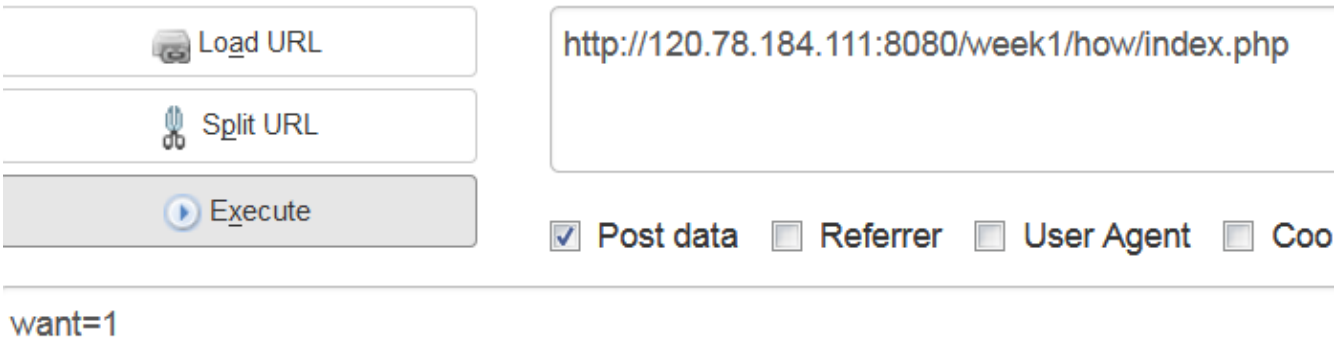
这道题是最后做出来的，因为一直不清楚备份文件应该是什么名字，在得到hint以后确定是源码泄露，之后才试出来。还跟学长申请了一下扫描器扫了扫，也没扫到，应该是字典太过垃圾，这个备份文件是以"·"来开头的，不加这个点就没办法下载文件。这也就提醒了以后要注意这样的情况，联系实际可以想到linux系统下隐藏文件就是要加前缀·来进行区分的。

.index.html.swp 下载以后直接查看文件内容即可发现flag

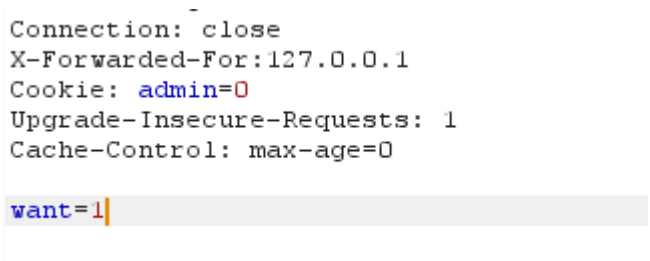
第二题

换头大作战

主要针对协议头的替换操作，第一个提示“I think POST is better” 这个hackbar可以解决

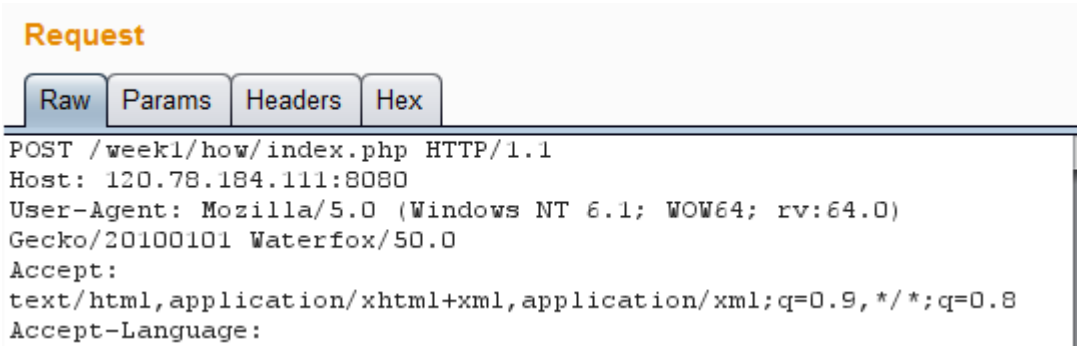


接下来提示“X-Forwarded-For only localhost can get flag”已经是明示了。资料：X-Forwarded-For 就可以burpsuite截取报文添加X-Forwarded-For为本地地址即可。



在返回的应答中发现需要使用waterfox这个浏览器，结合/User_agent这个提示，直接在burp suite中修改报文的User agent 达到效果。

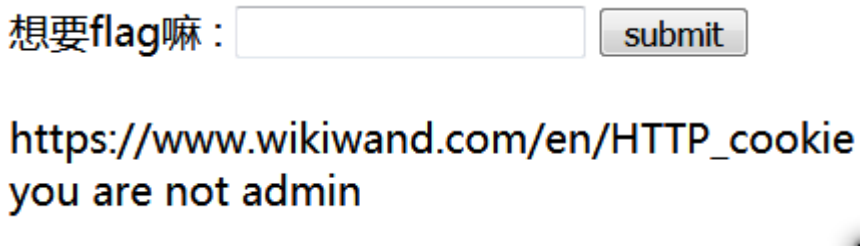
```
    □□flag□ : <input name="want" type="text">
    <input type="submit" value="submit">
</form>
</body>
</html>

<br/>https://www.wikiwand.com/en/User_agent<br/>please
use Waterfox/50.0
```



得到返回报文

```
<br/>https://www.wikiwand.com/en/HTTP_referer<br/>the
requests should referer from www.bilibili.com
```

要求referer应为www.bilibili.com 这里修改了以后出现了问题 burp suite 无法进行下一步操作 这令我很疑惑 之后就放弃了在repeater里操作 而是直接操作每次hackbar发送的报文，经历了同样的操作过后，顺利拿到下一步提示。



在cookie中修改admin为1，发送报文得到flag

第三题

very easy web

urldecode()二次解码

代码审计，一开始就给了php文件 分析一下，关键在于strpos函数，如果没有找到字符串就返回false 不满足if条件就退出 意思是不能出现明文vidar。这代码之后有一个id与urldecode之后的vidar进行比较，据资料$_GET[]会将url编码解码一次，所以将url编码两次，可以使得第一次strpos解码之后还是url编码，vidar!=urlencode(vidar)这样就可以绕过第一次判断，而第二次比较经过了两次的decode，使得后一个if条件可以满足。

```php
<?php
error_reporting(0);
include("flag.php");

if(strpos("vidar",$_GET['id'])!==FALSE)
    die("<p>干巴爹</p>");

$_GET['id'] = urldecode($_GET['id']);
if($_GET['id'] === "vidar")
{
    echo $flag;
}
highlight_file(__FILE__);
?>
```

就给出了flag 这里不知道为什么弄了半天在浏览器上会出各种问题，早知道做题的时候就截图了。

第四题

Can u find me?

出题人太善良了（从hint看出来）

F12以后发现了php文件 点过来发现需要post一个password

> yeah!you find the gate
>
> but can you find the password?
>
> please post password to me! I will open the gate for you!

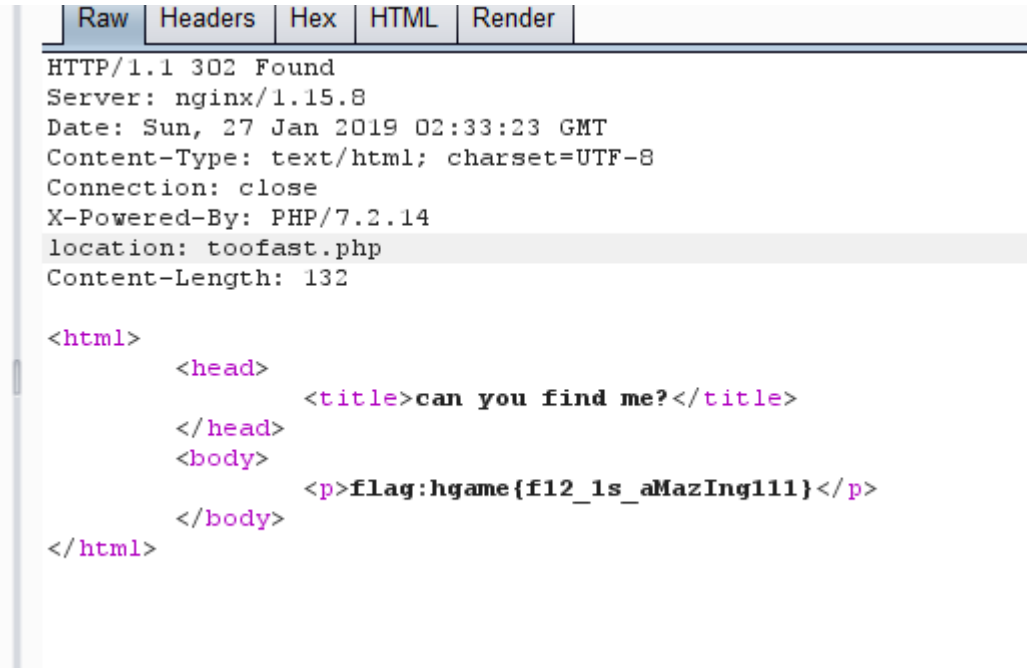
Post一个随机值以后发现提示错误 那就需要在网页中寻找正确的password 在文件中没有 在响应头中找到了一个 ***Password:woyaoflag*** 于是把正确的password发送过去

> yeah!you find the gate
>
> but can you find the password?
>
> please post password to me! I will open the gate for you!
>
> right!
>
> click me to get flag

有了进一步的网页，点进去发现提示

> aoh,your speed is sososo fast,the flag must have been left in somewhere

查看火狐的网络窗口,发现有一个302的重定向，从原先的iamflag.php重定向到了toofast.php 于是bure suite 截取请求报文，扔进reperter里go一下，得到一开始获得的iamflag报文。从中拿到flag



Re

第一题

这个在看了学习资料以后,认真钻研了一下代码规律，实际上就是简化的对内存的操作，把一个数组看成是内存中的一块连续区域，原来想着自己手动思考一下的，然后写了一堆堆py代码解决了。这道题很有启发意义，现在我大概清楚了brainfxxker的原理。

附py代码:

格式分析以后 使用自己并不熟练的python脚本来尝试输出对应的值

In [27]:
```
for a in range(127):
    i=a
    b=10
    while(b):
        a-=10
        b-=1
    if(a+2 == 0):
        print(chr(i))
```

```
b
```

In [28]:
```
for a in range(127):
    i=a
    b=9
    while(b):
        a-=9
        b-=1
```

就这样一步一步输出

第二题

HelloRe

最简单的逆向，扔到IDA里查看一下数据就发现了flag。

第三题

在做完这道题以后，感觉这题也简单。 就是中间矩阵乘法部分比较坑，让没学过线性代数的人怎么做。（虽然我学了）整体思路就是由程序里的两个答案矩阵，逆推出原始矩阵，说起来容易，做起来难，一开始看不太懂函数里的数学运算的意义，后来发现都是矩阵，一个在做加法，一个在做乘法，加和乘的对象都是一个特定的密码矩阵。从IDA里把一个个数据抠出来，换成矩阵，花费了一堆堆时间。整体思路如下:

做完分析以后，就是如何逆向来把flag推出来了。说起来容易，做起来难。原来以为两个加密方式只要用一个矩阵推就可以了，然后网上说这两个操作都会导致数据损失，还必须两个都推出来才能从两个方面推测flag。AB=C的乘法可以用A=C(B逆)来计算,

加密矩阵

8	1	7	1	1	0
4	8	1	2	3	9
3	8	6	6	4	8
3	5	7	8	8	7
0	9	0	2	3	4
2	3	2	5	4	0

v8对应答案矩阵

122	207	140	149	142	168
95	201	122	145	136	167
112	192	127	137	134	147
95	207	110	134	133	173
136	212	160	162	152	179
121	193	126	126	119	147

矩阵乘法

v9对应答案矩阵

16	8	8	14	6	11
5	23	5	10	12	23
14	23	19	7	8	10
4	13	22	17	11	22
6	14	2	11	18	9
5	8	8	10	16	13

还原出来的otherstr

8	7	1	13	5	11
1	15	4	8	9	14
11	15	13	1	4	2
1	8	15	9	3	15
6	5	2	9	15	5
3	5	6	5	12	13

还原出来的原始矩阵

h	g	a	m	e	{
1	_	t	h	i	n
k	_	M	a	t	r
1	x	_	i	s	_
v	e	r	y	_	u
s	e	f	5	l	}

矩阵计算器

https://zh.numberempire.com/matrixcalculator.php

原矩阵

6	6	6	6	6	7
3	5	7	6	6	6
6	5	4	6	7	7
3	7	5	6	7	5
7	6	7	7	5	7
7	6	6	3	6	7

还原出来的原始矩阵

96	96	96	96	96	112
48	80	112	96	96	96
96	80	64	96	112	112
48	112	80	96	112	80
112	96	112	112	80	112
112	96	96	48	96	112

密码矩阵的逆矩阵

0.057	0.156	-0.05	-0.08	-0.11	0.159
0.034	-0.06	0.044	-0.07	0.151	0.007
0.083	-0.18	0.07	0.085	0.108	-0.21
-0.11	0.032	0.245	-0.19	-0.23	0.31
0.048	0.012	-0.35	0.281	0.176	-0.12
-0.06	0.099	0.038	0.032	-0.11	-0.08

```
%
5
E
U
e
u
```

In [42]:
```
for a in range(127):
    if (a&15 == 5):
        print('%c'%a)
```

```
□
□
%
5
E
U
e
u
```

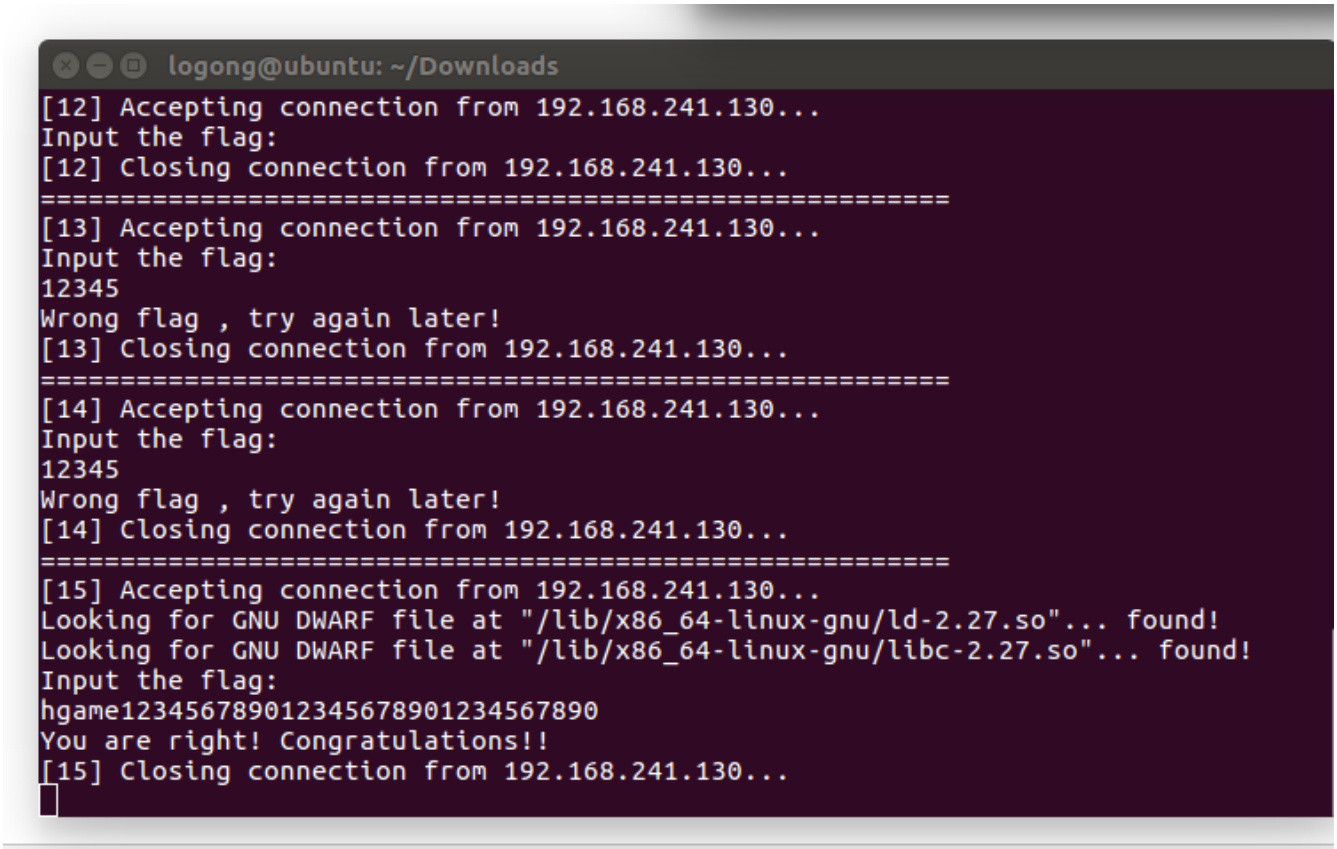
In [43]:
```
for a in range(127):
    if (a&15 == 6):
        print('%c'%a)
```

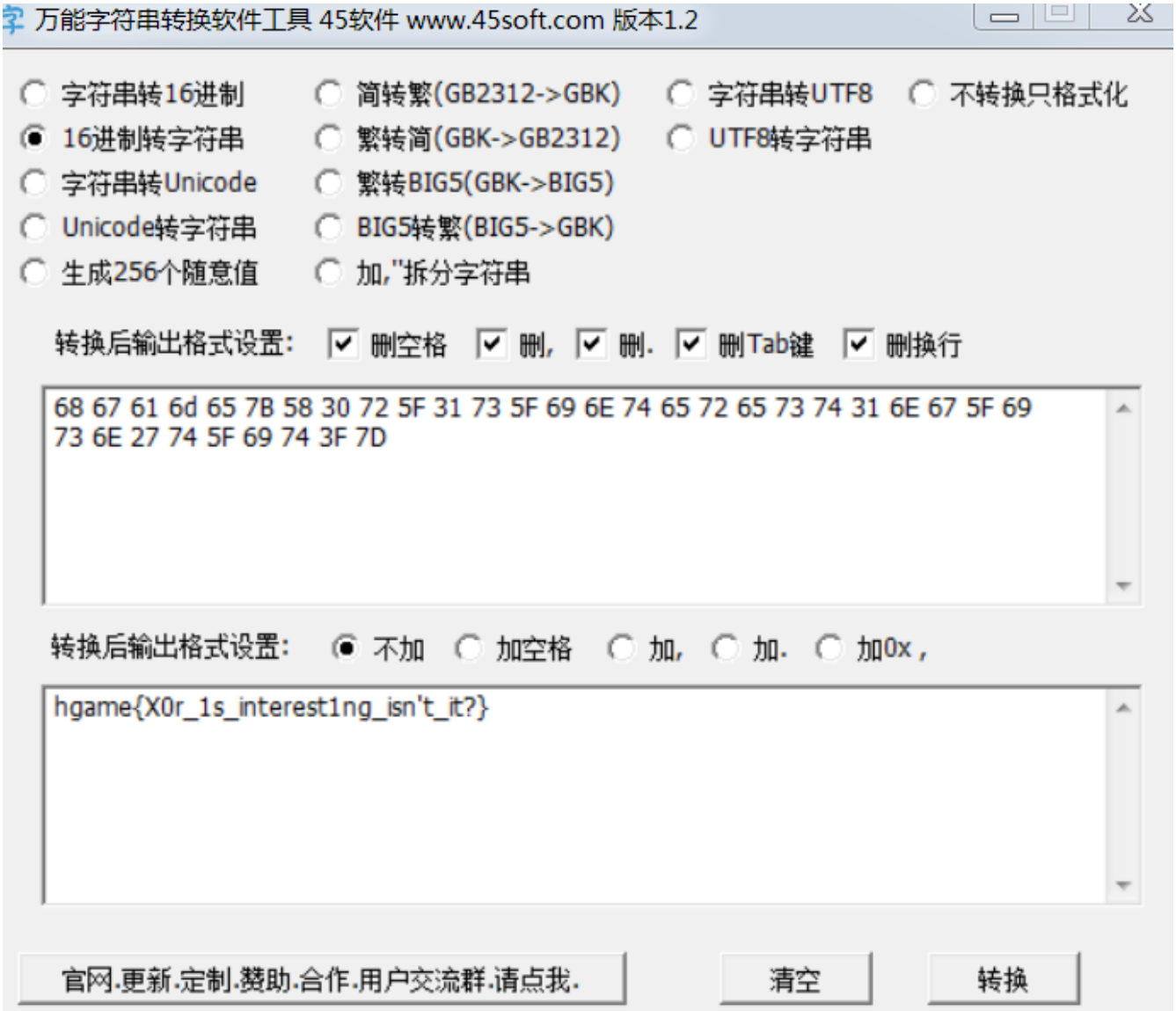
```
□
□
&
6
F
V
f
v
```

第四题

Xor

这道题略逊于base家族 不过让我被逼无奈学会了IDA的动态调试的方法，是真的好用，虽然没有熟悉它的操作，但是一步一步一步的把xor之后的值解出来了，是真的厉害。中间其实用的方法太笨了，我是直接改寄存器里的值来防止跳过的，后来一想有可能可以直接改代码，直接设置成无条件跳转多好，就省去了不停的改寄存器的麻烦。具体过程没截图。。不过有完成的截图。


logong@ubuntu: ~/Downloads
[12] Accepting connection from 192.168.241.130...
Input the flag:
[12] Closing connection from 192.168.241.130...
==========
[13] Accepting connection from 192.168.241.130...
Input the flag:
12345
Wrong flag , try again later!
[13] Closing connection from 192.168.241.130...
==========
[14] Accepting connection from 192.168.241.130...
Input the flag:
12345
Wrong flag , try again later!
[14] Closing connection from 192.168.241.130...
==========
[15] Accepting connection from 192.168.241.130...
Looking for GNU DWARF file at "/lib/x86_64-linux-gnu/ld-2.27.so"... found!
Looking for GNU DWARF file at "/lib/x86_64-linux-gnu/libc-2.27.so"... found!
Input the flag:
hgame12345678901234567890123456789O
You are right! Congratulations!!
[15] Closing connection from 192.168.241.130...



上面这个是记录下来的EAX中的值。

第五题

Pro的python教室

这道题只要会py就能看得出来 直接拼接给出的字符串就可以了，注意中间的字符串经过了一次base64加密

PWN

只会做aaaaaaaaaa

这道题比较简单 扔到ida里分析一下发现只要超出了99个a就可以okk，打开kali输入命令，再输入n多的a 就可以发现列出来了文件目录 发现一个flag文件 cat flag就可以得到flag

---

MISC

第一题

直接拖到stegsolve解决 手打flag

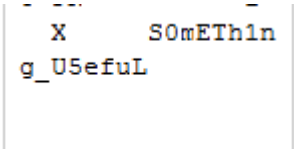
第二题

打字机

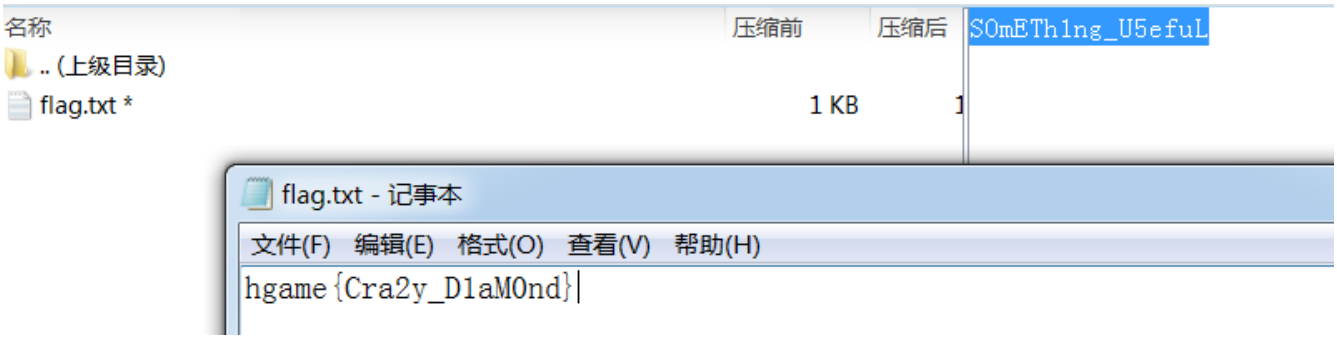
谷歌识图发现这是紫罗兰永恒花园中的打字机，知乎上有具体的图，其中一个回答有对比两张图，然后就是一个一个对比着往上写了，手打flag。没啥难度，但是大小写识别和一些比较难认的字符比较费事。

第三题

打开文件发现是一个损坏的压缩文件,Winhex查看一下文件内容发现有flag.txt这个文件，文件最后还有一串字符



原来不清楚这个是什么，加框flag试一试发现不正确,回头研究损坏的问题，网上查了查zip的文件头，发现文件头有问题，修改正确以后就可以打开了。但是打开的时候提示输入密码，这就比较清晰了，把刚刚得到的字符输入，得到flag。



第四题

Try

这道题特别麻烦，给了一个pcapng后缀的文件，以前也了解过，是wireshark的保存的文件，用wireshark打开，有一堆堆的报文，原先尝试着自己从报文里提取文件数据出来，但是在网上查了一些资料以后说wireshark可以直接帮助提取文件，于是直接用现有的功能提取出了文件： open-it这个压缩文件和一个password的txt文件 打开txt发现密码格式是由hgame开头的13位密码，这一步是最坑的，折腾了很久很久，最后觉得无字天书应该没有字母，用数字暴力破解，因为已经给出了5位的密码，后8位的数字比较好进行遍历。最后成功的暴力破解出密码。

输入密码，解压以后是一张图片，并没有flag的明确信息(真好看)，扔进winhax在最后发现了1.docx，更改后缀发现打开还是乱码，这条路不行，之后扔进binwalk分析了一下文件发现是一个压缩包，改成zip打开，发现了一堆堆的xml文件，在document里发现了hgame结合大括号就可以判断出内容，得到flag。

```
<w:t>hgame</w:t>
</w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:rFonts w:hint="eastAsia" />
    <w:vanish />
  </w:rPr>
  <w:t>{</w:t>
</w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:vanish />
  </w:rPr>
  <w:t>59d28413e36019861498e823f3f41406</w:t>
</w:r>
- <w:r w:rsidRPr="004066DE">
  - <w:rPr>
    <w:rFonts w:hint="eastAsia" />
    <w:vanish />
  </w:rPr>
  <w:t>}</w:t>
</w:r>
```

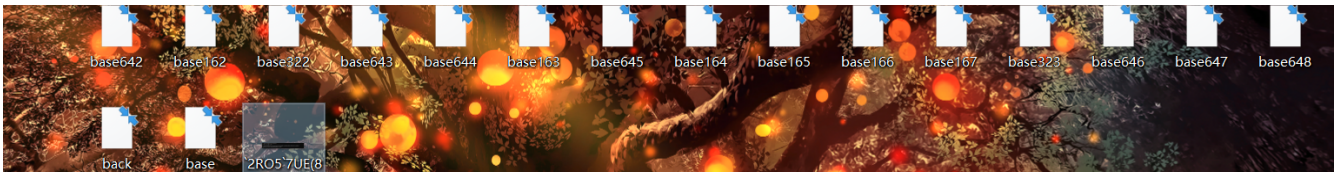
CRIPTO

第一题

这道题比较复合,里面有很多加密,hgame2018应该也有相似的题目 先是摩斯密码 16进制转ascll码 然后栅栏密码 栏数为9 然后凯撒密码 凯撒密码之后就没有别的了 但是这里注意栅栏密码的大小写 我因为大小写问题而错误了

第三题

最坑的题。 没有之一，后悔自己没有好好学python以下为血泪史。总体就是base16，base32，base64的不停decode。



最后还有一个base58 查了以后发现有一个py库可以直接出结果，就直接用了。

得到flag。