代理服务器封禁分析

我在我的 Shadowsocks 实例上运行了一个简单的服务器，代码如下：

```
from flask import Flask, jsonify
from flask_cors import CORS
import subprocess

app = Flask(__name__)
CORS(app)  # 为所有路由启用 CORS

@app.route('/bandwidth', methods=['GET'])
def get_bandwidth():
    # 运行 vnstat 命令获取 eth0 的 5 分钟间隔流量统计
    result = subprocess.run(['vnstat', '-i', 'eth0', '-5', '--json'], capture_output=True, text=True)
    data = result.stdout

    # 将捕获的数据作为 JSON 响应返回
    return jsonify(data)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

我使用 nginx 来服务 443 端口，如下所示：

```
server {
    listen 443 ssl;
    server_name www.some-domain.xyz;

    ssl_certificate /etc/letsencrypt/live/www.some-domain.xyz/fullchain.pem; # 由 Let's Encrypt 管理
    # ...
    location / {

        proxy_pass http://127.0.0.1:5000/;
        # ...
    }
}
```

这个服务器程序提供网络数据，我将其作为我的代理服务器，利用网络数据在博客上展示我的在

线状态。

有趣的是，这个服务器已经好几天没有被防火长城（GFW）或其他网络控制系统封禁了。通常，我设置的代理服务器会在一两天内被封禁。该服务器运行 Shadowsocks 程序，端口像 51939，因此它同时处理 Shadowsocks 流量和常规 API 流量。这种混合似乎让 GFW 认为该服务器不是专用的代理服务器，而是一个正常的服务器，从而避免了封禁该 IP。

这一观察结果令人感兴趣。看起来，GFW 使用特定的逻辑来区分代理流量和常规流量。尽管像 Twitter 和 YouTube 等网站在中国被封锁，但许多外国网站——如国际大学和公司的网站——仍然可以访问。

这表明，GFW 可能根据规则区分正常的 HTTP/HTTPS 流量和代理相关流量。处理两种类型流量的服务器似乎能够避免封禁，而仅处理代理流量的服务器更容易被封锁。