📝 Scope of Work – Integrate manage_dnsmasq.sh into Flask Control Panel

Project Context

This Flask app runs under the cornerpins user and must programmatically start/stop dnsmasq based on the status of LAN (eno1) and WAN (enp5s0f0/enp5s0f1) interfaces. The backend logic for determining this is already handled in a script:

bash

CopyEdit

/home/cornerpins/portal/manage_dnsmasq.sh

The goal is to expose this logic via the Flask app interface so the user can invoke it on demand.

🔧 Requirements

1. Flask Trigger Endpoint

Implement a Flask route such as:

http

CopyEdit

GET /network/refresh-dnsmasq

This endpoint must:

- **Run /home/cornerpins/portal/manage_dnsmasq.sh**
- **Return 200 OK and "dnsmasq service updated" if successful**
- **Return 500 with error message if the script fails**

2. Execution Privileges

The script must run as the cornerpins user and does not require sudo. Ensure:

- **File has execute permission:**

bash

CopyEdit

chmod +x /home/cornerpins/portal/manage_dnsmasq.sh

- **Flask calls it using subprocess.run without sudo**

3. Sample Flask Code

Provide the following in routes/network.py or similar:

python

CopyEdit

import subprocess

from flask import Blueprint, jsonify

network_bp = Blueprint('network', __name__)

@network_bp.route('/network/refresh-dnsmasq', methods=['GET'])

def refresh_dnsmasq():

try:

subprocess.run(["/home/cornerpins/portal/manage_dnsmasq.sh"], check=True)

return jsonify({"status": "success", "message": "dnsmasq service updated"}), 200

except subprocess.CalledProcessError as e:

return jsonify({"status": "error", "message": str(e)}), 500

4. Frontend (Optional)

If using a web UI, add a button that calls /network/refresh-dnsmasq via AJAX or a simple HTML form.

5. Logging (Optional but Ideal)

Write a simple log file:

python

CopyEdit

with open("/home/cornerpins/portal/logs/dnsmasq.log", "a") as log:

log.write(f"[{datetime.now()}] dnsmasq script run\n")