Vula Mock-GUI: Rapid Front-End Prototyping

Run the GUI without DBus or kernel dependencies

April 23, 2025

Mursel Khan, Sander Dedding | Vula Project

Motivation

- **Isolate** the front end from system services (DBus, WireGuard, …).
- Enable **UI development**, **bug reproduction** and **CI tests** on any machine.
- Provide deterministic dummy data via a `MockDataProvider`.

Key Components

MockDataProvider Returns empty or sample structures for `get_peers()`, `get_prefs()`, …

Patching Functions `patch_dataprovider()` and `patch_constants()` monkey-patch the real modules before the UI loads.

Tk App After patching, `vula.frontend.ui.App` starts as usual—no code changes inside the UI layer.

One-Time Container Setup

Executed automatically via `postCreateCommand`

```
# Install GTK and other build dependencies
sudo apt-get update && sudo apt-get install -y gir1.2-gtk-3.0 \
  girepository-2.0 libgirepository1.0-dev libcairo2-dev libffi-dev pkg-config

# Create and activate virtual environment
python3 -m venv venv
source venv/bin/activate

# Install Vula in editable mode and Python deps
pip install --upgrade pip setuptools wheel
pip install -e .

pip install pydbus PyGObject PyYAML qrcode Pillow pynacl cryptography schema
    click
```

Start the Mock-GUI

1. Open browser: `http://localhost:6080` (noVNC desktop)
2. In terminal:

```
cd /workspaces/vula
source venv/bin/activate
python tools/mock_gui_main.py
```

3. The GUI window appears; interact freely—backend is mocked.

Developer Benefits

- **Styling tweaks**: edit `vula/frontend/ui/*.py`, reload.
- **Debugging**: set breakpoints without touching system services.
- **CI**: headless launch, capture screenshots, run UI tests.

Take-aways

- Mock-GUI *decouples* front-end work from privileged backend.
- Perfect for fast iteration, demonstrations, and automated tests.
- Easily extended—add more fake data or test scenarios as needed.