Out-of-Band External Captive Portal

External Captive Portal Architecture (Dynamic, IP-based Design)

Overview

This architecture supports multiple access devices (e.g., Aruba, FortiGate) through a **single entry point** (`index.php`) that dynamically renders vendor-specific captive portal forms with customized styles. It relies on a backend database that defines:

- **Vendor profiles** (form field mappings, POST targets)
- **Access device mappings** (IP ➔ vendor profile)
- **Style templates** (colors, logos, layout definitions)

System Flow

1. **User connects** to guest network.
2. **Access device** redirects user to a fixed URL (e.g., `http://nac.example.com/guest/index.php?...`).
3. `index.php` determines access device IP (from `$_SERVER['REMOTE_ADDR']` or GET params).
4. **System queries DB**:
   - Which access device is this?
   - What vendor profile is mapped to it?
   - What style profile is associated with the portal?
5. `index.php` dynamically:
   - Loads the form template for the vendor (e.g., POST action to Aruba/FortiGate)
   - Injects dynamic form fields (from DB)
   - Loads CSS/styling
6. User fills out credentials.
7. **Form POSTs directly** to the access device's session endpoint (e.g., `http://<device>/swarm.cgi`).

Directory Structure

```
/var/www/html/guest/
├── index.php               # Central logic for rendering login page
├── templates/
│   ├── aruba_form.php        # Form skeleton for Aruba
│   └── fortigate_form.php     # Form skeleton for FortiGate
├── styles/
│   ├── default.css          # Default guest portal style
│   └── style_*.css           # Dynamic CSS files
└── utils/
    ├── validate_switch.php  # API to check if switch IP is trusted
    └── db.php               # DB connector

```

Dynamic Resolution Logic (`index.php`)

1. **Get source IP** (either from `$_SERVER` or GET `switchip` param).
2. **Query DB**:
   - Match IP to access device.
   - Resolve to vendor_profile and style_id.
3. **Load template** (`require "templates/{$vendor}_form.php"`).
4. **Inject form fields**:
   - From `form_fields` table.
   - Field names map to vendor POST requirements.
5. **Apply style**:
   - Load appropriate CSS (inline or via `<link>`).

Security Measures

- Only render form if switch IP is whitelisted.
- Escape all GET/POST inputs.
- Use HTTPS where possible.
- Avoid sending credentials to NAC; post directly to access device.

Advantages

- Single entry point for all devices.
- Dynamic form rendering based on database mappings.
- Centralized management of styles and logic.
- Easy addition/removal of access devices or portal variations.
