OpenWrt Network Management System Installation Guide

TP-Link Archer C7 v5 Setup

Prerequisites Check

Router Specifications:

- TP-Link Archer C7 v5
- OpenWrt firmware installed
- Minimum 16MB flash storage
- Minimum 128MB RAM
- SSH access enabled

Step 1: Prepare OpenWrt Environment

1.1 Update Package Manager

```bash
# SSH into your router
ssh root@192.168.1.1

# Update package lists
opkg update
```

1.2 Install Required System Packages

```bash
# Essential packages
opkg install python3 python3-pip sqlite3-cli
opkg install iptables-mod-extra kmod-sched-core tc
opkg install nodogsplash
opkg install wireless-tools iw

# Network monitoring tools
opkg install luci-app-statistics
opkg install collectd-mod-interface
opkg install collectd-mod-iwinfo
```

1.3 Increase Available Storage (if needed)

```bash
# Check available space
df -h

# If space is limited, use extroot with USB drive
# Insert USB drive and format
opkg install block-mount kmod-fs-ext4 kmod-usb-storage
mkfs.ext4 /dev/sda1
```

Step 2: Install Python Dependencies

2.1 Install Python Packages

```bash
# Core Flask and database packages
pip3 install flask flask-cors
pip3 install bcrypt
pip3 install sqlite3

# Data analysis and ML packages (lightweight versions)
pip3 install pandas numpy
pip3 install scikit-learn

# Additional utilities
pip3 install requests
pip3 install python-dateutil
```

Step 3: Create Project Directory Structure

3.1 Setup Directory Structure

```bash
# Create main directory
mkdir -p /opt/network-management
cd /opt/network-management

# Create subdirectories
mkdir -p {static,templates,logs,data}
mkdir -p static/{css,js,images}
mkdir -p templates/auth
```

3.2 Set Proper Permissions

```bash
chmod 755 /opt/network-management
chown -R root:root /opt/network-management
```

Step 4: Deploy Application Files

4.1 Upload Core Python Files

Upload these files to `/opt/network-management/`:

- `auth.py`
- `config.py`
- `db_config.py`
- `db_pool.py`
- `device_tracker.py`
- `device_bandwidth_monitor.py`
- `device_connection_monitor.py`
- `firewall_controller.py`
- `error_handling.py`
- `security.py`
- `validation.py`
- `caching.py`
- `ai_network_optimizer.py`
- `automated_incident_response.py`
- `cross_branch_analytics.py`
- `intelligent_guest_system.py`
- `predictive_registration.py`
- `run_device_monitors.py`

4.2 Create Main Application File

Create `/opt/network-management/app.py`:

```python
```

```
#!/usr/bin/env python3
from flask import Flask, jsonify, request
from flask_cors import CORS
import sqlite3
import os
import sys

# Add current directory to Python path
sys.path.append('/opt/network-management')

# Import our modules
from auth import UserManager, require_auth
from device_tracker import DeviceTracker
from firewall_controller import FirewallController
from error_handling import register_error_handlers, logger
from security import require_secure_headers
import config

app = Flask(__name__)
app.config.from_object(config)
CORS(app)

# Initialize components
user_manager = UserManager(config)
device_tracker = DeviceTracker(config)
firewall_controller = FirewallController(config)

# Register error handlers
register_error_handlers(app)

@app.route('/')
@require_secure_headers
def index():
    return jsonify({'message': 'Network Management System API', 'version': '1.0'})

@app.route('/api/login', methods=['POST'])
def login():
    # Login implementation
    pass

@app.route('/api/devices', methods=['GET'])
@require_auth(['Admin', 'NetworkManager', 'Support'])
def get_devices():
    # Device listing implementation
    pass

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

4.3 Initialize Database

bash

```
cd /opt/network-management
sqlite3 data/admin_management.db < db.sql
```

Step 5: Configure System Services

5.1 Create Systemd Service File

Create `/etc/init.d/network-management`:

bash

```
#!/bin/sh /etc/rc.common

START=80
STOP=20

USE_PROCD=1
PROG="/usr/bin/python3"
ARGS="/opt/network-management/app.py"

start_service() {
    procd_open_instance
    procd_set_param command "$PROG" "$ARGS"
    procd_set_param respawn
    procd_set_param stdout 1
    procd_set_param stderr 1
    procd_close_instance
}
```

5.2 Make Service Executable

bash

```
chmod +x /etc/init.d/network-management
/etc/init.d/network-management enable
```

5.3 Create Monitor Service

Create `/etc/init.d/device-monitors`:

```bash
#!/bin/sh /etc/rc.common

START=81
STOP=19

USE_PROCD=1
PROG="/usr/bin/python3"
ARGS="/opt/network-management/run_device_monitors.py"

start_service() {
    procd_open_instance
    procd_set_param command "$PROG" "$ARGS"
    procd_set_param respawn
    procd_set_param stdout 1
    procd_set_param stderr 1
    procd_close_instance
}
```

```bash
chmod +x /etc/init.d/device-monitors
/etc/init.d/device-monitors enable
```

Step 6: Configure Networking

6.1 Configure Firewall Rules

```bash
```

```
# Allow HTTP access to management interface
uci add firewall rule
uci set firewall.@rule[-1].name='Allow-Management-HTTP'
uci set firewall.@rule[-1].src='lan'
uci set firewall.@rule[-1].dest_port='5000'
uci set firewall.@rule[-1].proto='tcp'
uci set firewall.@rule[-1].target='ACCEPT'

uci commit firewall
/etc/init.d/firewall restart
```

6.2 Configure NoDogSplash

```
bash

# Configure captive portal
uci set nodogsplash.@nodogsplash[0].enabled='1'
uci set nodogsplash.@nodogsplash[0].gatewayinterface='br-lan'
uci set nodogsplash.@nodogsplash[0].maxclients='250'
uci set nodogsplash.@nodogsplash[0].authidletimeout='1200'
uci set nodogsplash.@nodogsplash[0].sessiontimeout='3600'

# Set custom splash page
uci set nodogsplash.@nodogsplash[0].splashpage='/opt/network-management/templates/splash.html'

uci commit nodogsplash
```

Step 7: Create Web Interface

7.1 Create Basic HTML Templates

Create `/opt/network-management/templates/index.html`:

```
html
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Network Management</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <h1>Network Management System</h1>
    <div id="app">
        <!-- React/Vue app will be mounted here -->
    </div>
</body>
</html>
```

7.2 Create Splash Page for Captive Portal

Create `/opt/network-management/templates/splash.html`:

```html
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to WiFi</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <div class="splash-container">
        <h1>Welcome to Our Network</h1>
        <form action="$authaction$" method="get">
            <input name="tok" value="$tok$" type="hidden">
            <input name="redir" value="$redir$" type="hidden">
            <button type="submit">Connect to Internet</button>
        </form>
    </div>
</body>
</html>
```

Step 8: Configuration and Security

8.1 Update Configuration File

Edit `/opt/network-management/config.py`:

```python
# Update router-specific settings
ROUTER_IP = '192.168.1.1'  # Your router's IP
DB_PATH = '/opt/network-management/data/admin_management.db'
LOG_FILE = '/opt/network-management/logs/app.log'

# Security settings for production
DEBUG = False
SECRET_KEY = 'your-secure-secret-key-here'
```

8.2 Set Up Log Rotation

Create `/etc/logrotate.d/network-management`:

```
/opt/network-management/logs/*.log {
    daily
    rotate 7
    compress
    missingok
    notifempty
    create 644 root root
}
```

Step 9: Initialize and Start Services

9.1 Create Initial Admin User

```bash
cd /opt/network-management
python3 -c "
from auth import UserManager
import config
um = UserManager(config)
result = um.create_user('admin', 'admin123!', 'Admin', 1)
print(result)
"
```

9.2 Start All Services

```bash
# Start the main application
/etc/init.d/network-management start

# Start device monitors
/etc/init.d/device-monitors start

# Start NoDogSplash
/etc/init.d/nodogsplash start

# Check service status
/etc/init.d/network-management status
logread | tail -20
```

Step 10: Testing and Verification

10.1 Test Web Interface

```bash
# Test local access
curl http://192.168.1.1:5000/

# Test from another device on the network
# Open browser to http://192.168.1.1:5000/
```

10.2 Verify Database

```bash
sqlite3 /opt/network-management/data/admin_management.db "SELECT * FROM users;"
```

10.3 Check Logs

```bash
tail -f /opt/network-management/logs/app.log
logread | grep network-management
```

Troubleshooting

Common Issues:

1. Python Package Installation Fails

```bash
# Free up space first
opkg remove --autoremove luci-app-* (unused apps)
# Use pip with --no-cache-dir
pip3 install --no-cache-dir package_name
```

2. Database Permission Issues

```bash
chown root:root /opt/network-management/data/admin_management.db
chmod 644 /opt/network-management/data/admin_management.db
```

3. Service Won't Start

```bash
# Check Python path
which python3
# Check dependencies
python3 -c "import flask; print('Flask OK')"
# Check logs
logread | grep network-management
```

4. High Memory Usage

```bash
# Monitor memory usage
free -m
# Consider reducing AI features if memory is limited
```

Performance Optimization

For Limited Resources:

1. **Disable AI features** if memory is constrained
2. **Reduce monitoring frequency** in config
3. **Use database cleanup scripts** to manage data size

4. **Consider external database** for multiple branches

Security Considerations

1. **Change default passwords** immediately
2. **Use HTTPS** in production (requires SSL certificate)
3. **Limit access** to management interface
4. **Regular security updates** for OpenWrt and packages
5. **Monitor logs** for suspicious activity

Maintenance

Regular Tasks:

```bash
# Update packages monthly
opkg update && opkg upgrade

# Clean old logs
logrotate -f /etc/logrotate.d/network-management

# Backup database
cp /opt/network-management/data/admin_management.db /tmp/backup_$(date +%Y%m%d).db
```

This installation should give you a fully functional network management system on your TP-Link Archer C7 v5!