WPA2 Security Auditor - Project Structure & Usage Guide

Overview

WPA2 security auditing tool for NXP i.MX93-11x11-LPDDR4X-EVK. Captures WPA2 handshakes and performs offline dictionary attacks.

Platform: NXP i.MX93 EVK
WiFi Adapter: Atheros AR9271 (ath9k_htc)
Language: C++ with Python OLED helper

File Structure

```
wpa2_auditor/
├── main.cpp                  # Main application logic
├── hardware.h/cpp            # LED, Button, OLED interfaces
├── wifi.h/cpp                # WiFi & packet processing
├── cracker.h/cpp             # WPA2 cracking engine
├── oled_display.py           # Python OLED helper
├── Makefile                  # Build configuration
├── dictionary_short.txt      # User provided
├── dictionary_medium.txt     # User provided
├── dictionary_large.txt      # User provided
└── wpa2-attack-info.txt      # Auto-generated log
```

Component Details

1. `main.cpp` - Main Application

State Flow:

```
Startup Menu → Scanning (3s) → Network List (auto-rescan 15s) →
SSID Options → [Deauth: Client Selection] → Capturing Handshake →
Dictionary Selection → Cracking → Success/Failure
```

Key Functions:

- `scan_for_networks()` - Scans channels 1,3,5,7,9,11 for WPA2 beacons
- `find_clients()` - Discovers associated clients
- `capture_handshake()` - Captures 4-way handshake with optional deauth

---

2. `hardware.h/cpp` - Hardware Control

LEDController

- `set_solid(LED, on)` - On/off
- `set_blink(LED, delay_ms)` - Blink with timing
- **RED:** Idle/Cracking/Failed | **YELLOW:** Scanning | **GREEN:** Success | **BLUE:** Capturing

ButtonController

- `get_press()` - Returns UP/DOWN/SELECT/ESC/NONE
- `check_double_esc()` - True if ESC pressed twice within 1s

OLEDDisplay (Python wrapper)

- `show_menu(items, idx)` - Menu with cursor
- `show_message(title, lines)` - Title + message
- `show_scanning(count)` - Scan status
- `show_capturing(mode, ssid)` - Capture status
- `show_cracking(progress, total)` - Progress bar
- `show_success(password)` - Result display

---

3. `wifi.h/cpp` - WiFi & Packet Processing

WiFiNetwork Structure

```cpp
uint8_t bssid[6], ap_mac[6], client_mac[6];
uint8_t anonce[32], snonce[32], mic[16];
std::string ssid;
int channel;
std::vector<uint8_t> eapol_msg1, eapol_msg2;
```

WiFiController

- `setup_monitor_mode()` - Enable monitor mode
- `set_channel(int)` - Set WiFi channel
- `open_capture(filter)` - Open pcap with BPF

PacketProcessor (static methods)

- `is_beacon_frame()`, `parse_beacon()` - Extract SSID, check WPA2
- `is_eapol_frame()`, `get_eapol_message_type()` - Identify handshake messages
- `parse_eapol_msg1()` - Extract ANonce
- `parse_eapol_msg2()` - Extract SNonce, MIC
- `craft_deauth_frame()` - Create deauth packet

---

4. `cracker.h/cpp` - WPA2 Cracker

Methods:

- `log_handshake_info()` - Write to `wpa2-attack-info.txt`
- `crack(net, dict_file, callback)` - Dictionary attack with progress

Algorithm:

```
For each password:
  PSK = PBKDF2-HMAC-SHA1(password, SSID, 4096)
  PTK = PRF-512(PSK, "Pairwise key expansion", ...)
  KCK = PTK[0:16]
  MIC = HMAC-SHA1(KCK, EAPOL_with_zeroed_MIC)
  If MIC matches → Found!
```

Performance: 100-1000 pwd/sec on i.MX93

---

5. `oled_display.py` - OLED Helper

Commands called by C++:

```bash
./oled_display.py clear
./oled_display.py menu <idx> <item1> <item2> ...
./oled_display.py message <title> <line1> ...
./oled_display.py scanning <count>
./oled_display.py capturing <mode> <ssid>
./oled_display.py cracking <percent> <total>
./oled_display.py success <password>
./oled_display.py failure
```

Building & Running

Prerequisites

```bash
# System packages
sudo apt-get install build-essential libpcap-dev libssl-dev

# Python packages
pip3 install luma.oled evdev
```

Dictionaries: Place password dictionaries in same directory as executable:

```bash
# Short (~10K)
wget https://github.com/danielmiessler/SecLists/raw/master/Passwords/Common-Credentials/10-mi

# Medium (~100K)
wget https://github.com/danielmiessler/SecLists/raw/master/Passwords/Common-Credentials/10-mi

# Large (rockyou ~14M)
wget https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt -O dictio
```

Build & Run

```bash
make
chmod +x oled_display.py
sudo ./wpa2_auditor
```

Log Files

`wpa2-attack-info.txt`

Auto-generated with handshake details:

```
==================== Handshake Capture =====================
Timestamp: 1701504000
SSID: MyWiFi-5G
BSSID: AA:BB:CC:DD:EE:FF
AP MAC: AA:BB:CC:DD:EE:FF
Client MAC: 11:22:33:44:55:66
Channel: 6

ANonce: <64 hex>
SNonce: <64 hex>
MIC: <32 hex>

EAPOL Message 1 (121 bytes): <hex dump>
EAPOL Message 2 (121 bytes): <hex dump>

[CRACKED]
Password: mypassword123
Attempts: 45678
Time: 123 seconds
============================================================
```

Troubleshooting

Monitor mode fails:

```bash
sudo ifconfig wlan0 down
sudo iwconfig wlan0 mode monitor
sudo ifconfig wlan0 up
iwconfig wlan0  # Verify Mode:Monitor
```

OLED not working:

```bash
i2cdetect -y 0  # Should show 0x3C
./oled_display.py message "Test" "Line1"
```

LEDs not working:

```bash
ls /sys/class/leds/
echo 255 > /sys/class/leds/wpa2:red:status/brightness
```

Buttons not working:

```bash
ls /dev/input/by-path/*kbd*
evtest /dev/input/event1
```

No packets captured:

```bash
iwconfig wlan0  # Verify monitor mode
sudo tcpdump -i wlan0 -c 10
```

Security & Legal Notice

⚠️ EDUCATIONAL USE ONLY

Legal Uses:

- ✅ Your own networks
- ✅ Lab with written permission
- ✅ Networks you own/authorized to test

Illegal:

- ❌ Unauthorized networks
- ❌ Public WiFi without permission
- ❌ Unauthorized access with captured credentials

By using this tool, you accept full legal responsibility.