StrangeLab — CamUtil Cockpit ↔ Vault Architecture (v2)

Generated: 2026-02-02 23:39

This document is the consolidated “source-of-truth” for the CamUtil work stream discussed in this chat: what the app does today, how it fits into the StrangeLab architecture, where artifacts/logs go, and the concrete build/run commands.

1. Current Lab Architecture

Committed blueprint: Mac is Tier 0 “Brain/Cockpit” (UI + orchestration), with two headless Raspberry Pis as Tier 1: Pi■Aux (aggregator/normalizer + extra radios) and Pi■Logger (append■only vault/logger backed by the 8TB SSD).

- Tier 0 (Mac): CamUtil “scanner/cockpit” app for discovery, operator actions, and client■friendly reporting.
- Tier 1a (Pi■Aux): sensor/radio expansion node (Wi■Fi/BLE dongles, ESP32 neighbors) + event forwarder to vault; also a local ‘node view’ in the Mac cockpit.
- Tier 1b (Pi■Logger/Vault): append■only storage on the 8TB SSD; receives scan artifacts + node event streams; later feeds analysis/graphing.
- Edge nodes: ESP32 / CYD / other sensors produce event■based telemetry (not continuous media).

2. CamUtil: What It Does Now

CamUtil is a legitimate, work■usable network + BLE inventory/mapping tool with strong guardrails: scope controls, consent UX, safe■mode gating, and audit■grade exports. It focuses on discovery + evidence capture; “active” interactions are explicitly gated and credential■driven.

2.1 Network Discovery & Evidence

- Scan scope controls: CIDR + optional IP range override; default “Only scan my local subnet” ON.
- Safe Mode default ON: blocks active interactions beyond discovery/read■only metadata (e.g., blocks ONVIF RTSP fetch / hard probe when enabled).
- Host scan produces: alive/no■response, open ports, hostname (reverse DNS), basic HTTP title fingerprinting when available.
- ARP integration: parses /usr/sbin/arp -an to map IP → MAC; runs after scan; manual ‘Refresh ARP’ button.
- OUI vendor lookup: importable OUI text file + bundled starter mapping; maps MAC prefix → vendor name.
- Exports: CSV + JSON snapshots with IP/status/ports/hostname/MAC/vendor/ONVIF flags/RTSP URI (when known).

2.2 BLE Discovery, Mapping, and Find/Radar

- Passive CoreBluetooth scanning (no connections by default).
- BLE fingerprints include: name, connectable flag, TX power (if present), service UUIDs, manufacturer data (company ID + prefix), beacon hints (iBeacon/Eddystone).
- BLE knowledge tables: Company IDs + Assigned Numbers + Service UUID tables loaded locally (no downloads inside app).
- Readable layer: display 16■bit/32■bit UUIDs alongside normalized 128■bit Bluetooth base UUID form.
- Find mode: pick a device and focus on it; RSSI smoothing (EMA) + trend and proximity buckets; radar view to help ‘walk it down’ physically.
- BLE evidence is included in audit + report exports.

2.3 Confidence Scoring

- Deterministic confidence scoring for hosts and BLE peripherals (level + numeric score + reasons).
- Used for sorting (e.g., Cameras/Interesting sorted by confidence then IP) and filtering (e.g., High confidence only).
- Included in exports and readable reports.

2.4 Operator Actions + Hard Probe

- Unified selection by IP; action panel available for any selected host (not only Interesting).
- Actions: Open Web UI, Open in VLC, Copy IP/RTSP (and/or export equivalents depending on current UI state).
- Hard Probe button (Interesting/Cameras only): uses known RTSP URI, runs reachability diagnostics, launches multiple VLC variants, and writes a per■probe folder with logs + a status/report JSON (even when blocked).
- Safe Mode blocks active steps; UI should show why an action is disabled.

3. Storage, Exports, and Data Flow

Local operator artifacts are written under ~/StrangeLab/scans/ in clearly named subfolders. These artifacts should be shipped to the Pi■Logger vault (append■only) for long■term retention and downstream analysis. The Mac remains the cockpit/report generator; the vault is the durable truth store.

Category	Folder (Mac)
Runtime logs	~/StrangeLab/scans/logs-raw/ and logs-readable/
Audit exports	~/StrangeLab/scans/audit-raw/ and audit-readable/
Evidence per host	~/StrangeLab/scans/evidence-raw/ and evidence-readable/
BLE fingerprint exports	~/StrangeLab/scans/ble-raw/ and ble-readable/
Device reports	~/StrangeLab/scans/device-report-raw/ and device-report-readable/

Scan reports	~/StrangeLab/scans/scan-report-raw/ and scan-report-readable/
RTSP hard probe runs	~/StrangeLab/scans/rtsp-hard-probe/<ip>-<iso>/

3.1 Shipping to Pi■Logger Vault

Recommended: a simple ‘shipper’ job (rsync or HTTP ingest) that pushes Mac artifacts to the vault. Keep the vault append■only: never mutate prior records; write new versions/snapshots with timestamps. The Mac can keep a rolling window or prune old artifacts locally if disk becomes tight.

4. Pi■Aux Integration Status

You reported: Pi■Logger is online with the 8TB SSD attached and logging; Pi■Aux is on Wi■Fi with BLE/Wi■Fi dongles and is ready to produce/forward events. The next step is to make Pi■Aux emit events into the Mac cockpit (for operator view) and simultaneously forward them to the vault.

- Mac cockpit should show a tab per active node (starting with Pi■Aux), with last■seen time, health, and recent events.
- Artifacts produced on Mac are client■facing; the same artifacts (raw+readable) should be shipped to the vault for retention/analysis.
- Design principle: replaceable modules—swap Mac/Pi/node without re■architecting the whole system.

5. Build & Run Commands

Task	Command
Build (Debug)	xcodebuild -project /Users/letsdev/CamUtil/CamUtil.xcodeproj -scheme CamUtil -configuration Debug -des
Open built app (DerivedData)	open ~/Library/Developer/Xcode/DerivedData/CamUtil-*/Build/Products/Debug/CamUtil.app
Find actual build output path	xcodebuild -project /Users/letsdev/CamUtil/CamUtil.xcodeproj -scheme CamUtil -configuration Debug -sho
Run binary directly	~/Library/Developer/Xcode/DerivedData/CamUtil-*/Build/Products/Debug/CamUtil.app/Contents/MacOS/C

6. Known Gotchas & Fixes

- DerivedData vs repo build folder: the real app Info.plist and binaries live under ~/Library/Developer/Xcode/DerivedData/... not ./build/Build/Products unless you explicitly set a custom build dir.
- macOS Bluetooth permission requires the correct NSBluetooth* usage string in the app bundle Info.plist; verify with PlistBuddy against the DerivedData-built app.
- tccutil Bluetooth resets can fail depending on macOS version; best test is bump bundle id or delete app + reset privacy settings from System Settings.
- SSDP/Bonjour results can legitimately be zero depending on network/device settings; ARP + port scans are often more reliable for ‘what is here’ mapping.

7. Next Steps

- Make CamUtil exports ship to Pi■Logger automatically (append■only): rsync/SSH or a tiny HTTP ingest service on the vault.
- Replace the tiny BLE mapping tables with full datasets (still local-first) and keep them versioned in Resources or an import path.
- Finish UUID normalization display everywhere BLE UUIDs appear (list/detail + readable reports).
- Add a DMG packaging script for local distribution across your Macs (no developer fees required).
- After cockpit is stable: start the separate forensic analysis/graphing app that reads from the vault and correlates across Wi■Fi/BLE/node sensors.