Advertised Edge AI, Implemented Rule Engine: A Black-Box Measurement of the PILHOME Smart-Home Gateway

Daozhendong Yang
School of Intelligent Science and Engineering,
Yunnan Technology and Business University, Kunming, China
hi@yxpil.com

Abstract

Smart-home gateways are security-critical infrastructure with an unusually ambitious feature list: edge inference, self-evolving rules, behaviour profiles, full-home takeover through Home Assistant, an agent-facing tool endpoint, and an audit trail. PILHOME v0.1.0 advertises all of these. This paper measures how much of that surface is enforced by the shipped code, using a harness that runs the real gateway against a mock Home Assistant WebSocket server, a mock MQTT broker, and HTTP/UDP sinks, and that reads back the gateway's own state over its 60 HTTP routes.

Three results stand out. *First*, the released tree does not build: one of the three files the router references was never committed, so the crate fails with `E0583`. Supplying a minimal shim is not enough to use the system, because the authentication middleware never reads its own `auth_enabled` flag — with the documented default of *off*, 12/12 endpoints answer 401 — and the allow-listed `/api/token` hands a 64-character token to any anonymous caller, turning 12/12 endpoints green. *Second*, the "edge AI" is a 4-point nearest-neighbour lookup over 8 hand-written feature vectors plus a detector that returns an empty vector; there are 0 model files and no inference dependency in the lock file, yet the gateway reports 3 models as *ready*. Video severity is computed from detections the *caller* posts: a client can request a `critical` alert, though 0 such alerts reach the event log and so cannot actuate anything. *Third*, what is implemented is solid: the Home Assistant bridge completes a real `auth`/`get_states`/`call_service` exchange (4 entities, 4 mirrored devices, 5 service calls), the automation engine fires in about a second, WOL packets are well formed, and 8 agent tools work. Meanwhile the shipped configuration file can never take effect — three independent defects, each measured — the configured scan profile is ignored, and 8/14 security-relevant actions are audited. We close with the test list that would have caught most of this before release.

Artifact. The gateway is public at `https://github.com/yxpil/PILHOME` (yxpil, 2026) at commit `ed4a576`. This paper's directory contains the harness (`scripts/home_lab.py`, `scripts/measure.py`, `scripts/audit_repo.py`), the build scripts, the shim, the raw JSON artifact, and the generators for every table and figure. No number here was typed by hand.

1 Introduction

A household security gateway sits between sensors, locks, cameras and the internet. NIST's guidance for such devices assumes a small, checkable property set: authenticated interfaces, verified configuration, durable logs, and no undocumented capability (National Institute of Standards and Technology, 2020). PILHOME is a student-built Rust gateway whose README and 13 design documents claim, among other things, "edge AI inference (detection / gesture / video audit)", "self-evolving", "Home Assistant deep integration", "MCP endpoint (agents control the whole house)" and "append-only audit trail".

Those are checkable claims. This paper checks them by running the software: 60 HTTP routes are exercised against mock Home Assistant and MQTT peers, and the resulting behaviour is compared against the documentation and against the source that produces it. The headline is a mismatch of a specific kind. The gateway is not broken — most endpoints work, the integrations are real protocol exchanges, and the rule engine fires — but several advertised *subsystems* are in fact thin surfaces over hand-written rules, and several *controls* (authentication, configuration, audit) are present in the schema and absent from the execution path.

Contributions.

C1. A harness for smart-home gateways that needs no household hardware: a mock Home Assistant WebSocket server, a mock MQTT 3.1.1 broker (Gupta and Miller, 2014), HTTP and UDP sinks, and a raw WebSocket client for the gateway's own event stream (§2).

C2. Evidence that the released artifact is not deployable: a missing source file stops the build (`E0583`), and after supplying it the API is still unreachable because the auth middleware never consults its own enable flag (§3).

C3. A quantified account of an "edge AI" that is a rule engine: 4 templates, 8 dimensions, 0.70 μs of compute, 0 model files, and 0/2 registered model paths that exist on disk (§4).

C4. A security analysis of the control surface: one anonymous `POST /api/token` defeats authentication that is simultaneously always on; a 12-binding QuickJS tool endpoint gives any token holder arbitrary script execution with device, MQTT, WOL and outbound-`httpGet` reach (§3, §6).

C5. A measured list of what does work — the Home Assistant bridge, the rule engine, WOL, MCP, the event stream — so the paper is a characterization as well as a critique (§6).

C6. A regression checklist: twelve assertions, each one command, that would have caught every finding in this paper before release (§8).

Non-goals. This is not a vulnerability disclosure exercise: every observation is behavioural, on a loopback interface, against software the author published. We do not attack the WebUI, do not test the TLS story (there is none), do not evaluate real household traffic, and do not measure radio or device-level security. We report the gateway, not the IoT ecosystem.

2 Method

2.1 The artifact and the shim

Built on axum and tokio (tokio-rs, 2024a,b) with serde-parsed configuration (Tolnay, 2024), PILHOME is a cargo workspace of 11 crates, 8,102 Rust lines in 90 files, plus 1,157 lines of TypeScript front end and 1,242 lines of design prose in 13 documents. The server crate is 62% of the Rust code (Table 2).

The workspace does not compile. `Server/src/api/mod.rs:29` declares `pub mod token;` and registers three routes for it, but `Server/src/api/token.rs` has never been committed:

```
error[E0583]: file not found for module `token`
  --> Server/src/api/mod.rs:29:1
```

The repository has 1 commit, so this is not a regression introduced by a later change: the tree as published cannot be built. To measure anything we wrote `scripts/shim/api_token.rs`: three handlers that call the `TokenStore` methods already present in `Server/src/auth.rs` (`issue`, `revoke`, `count`). The shim adds no policy; it is `build_server.py`'s job to copy the tree, drop the file in, and build the copy, leaving the repository untouched. Every measurement in this paper is made against that binary (13,819,904 bytes), and every claim that depends on the missing file is labelled as such.

2.2 Harness

The gateway starts in 0.477 s and idles at 13.29 MB RSS with 509 handles (Table 1). Each scenario is a sequence of `urllib` calls plus, where relevant, a raw WebSocket client; the harness counts what the peer services actually received rather than trusting the gateway's own answers. Three details matter for reproducibility: the MQTT broker must sit on port 1883 because the client's port is hard-coded; the UDP sink binds port 9 to capture magic packets; and the mock HA server keeps idle sockets open, because the bridge treats an idle read as a connection loss.

2.3 Threats to validity

The gateway is measured through one build of one commit on one Windows host; the 32 in-tree tests were not audited for soundness beyond counting them. Mock peers implement the documented subset of each protocol, so a real Home Assistant instance may accept or reject things our mock does; where that distinction matters we say so. Finally, the shim is a confound: the `/api/token` endpoint we measure is our handler over the author's `TokenStore` and the author's middleware allow-list, so the bypass in §3 is a property of the shipped middleware, but its *reachability* depends on supplying the missing file.

3 Access control: always on and open

`auth.rs` documents itself as opt-in: "*enabled by setting `auth_enabled = true` (default off, does not affect local use)*". The middleware it exports does not read that flag. Its body is an allow-list check followed by a Bearer check, and `main.rs` attaches it unconditionally. The measured consequence (Table 3, Figure 2) is that with the default configuration, 12/12 of our probes return 401 and 0 return data: the gateway is not "local use, no auth", it is "unusable without a token".

The second half is worse. The allow-list is `{"/", "/assets/*", "/api/token", "/api/ws"}`, and `/api/token` issues a 64-character hex token to whoever asks. One unauthenticated request therefore turns 12/12 endpoints green, and — because the flag is never read — enabling authentication changes nothing: an attacker who cannot read your device list can still mint the token that reads it. This is the combination of CWE-287 (improper authentication) with CWE-306 (missing authentication for a critical function) (MITRE, 2024b,c): the token endpoint is the authentication mechanism and is also its bypass.

Two related observations from the same probe: `/api/token/status` is *not* allow-listed, so an operator cannot ask how many tokens exist without already having one; and `/` returns 404 because the WebUI bundle is not built in the repository, which is the same class of gap as the missing module — the shipped artifact is a source snapshot, not a release.

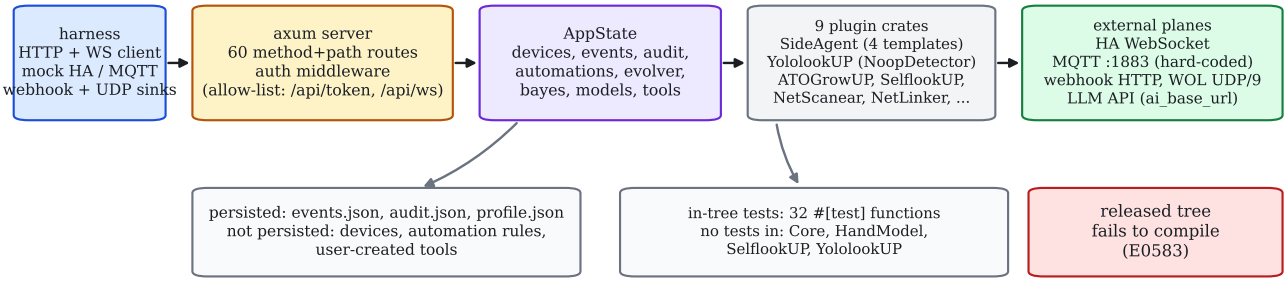


Figure 1: The measured system. The harness drives the HTTP and WebSocket surface and is also the peer at the other end of every integration: a mock Home Assistant WebSocket server, a mock MQTT broker on the port the gateway dials, an HTTP webhook sink and a UDP sink for wake-on-LAN.

Table 1: The measured artifact. The released tree does not compile; the one file needed to build it is `scripts/shim/api_token.rs` in this paper's directory and is described in §2.

Item	Value
Repository commit	`ed4a576` (2026-10-06, 1 commits, worktree clean: yes)
Pristine build	`cargo build --release -p pilhome-server` → exit 101: `error[E0583]: file not found for module 'tok`
Shimmed build	13,819,904 bytes, `scripts/shim/api_token.rs` added
Workspace	90 Rust files, 8,102 lines, 11 crates
Server crate	52 files, 5,020 lines (62% of the tree)
HTTP surface	60 method+path routes in `Server/src/api/mod.rs`
Unit tests	32 `#[test]` functions; crates with none: `Core`, `HandModel`, `SelflookUP`, `YololookUP`
Design docs	13 Markdown files, 1,242 lines
Host	Windows 11 10.0.26300 x64, Python 3.13.14
Mock services	HA WebSocket server, MQTT 3.1.1 broker on port 1883, HTTP webhook sink, UDP sink on port 9
Cold start	0.477 s to a listening socket, 13.29 MB RSS, 509 handles

4 The advertised edge-AI path

4.1 Gesture: four points in eight dimensions

`SideAgent` exposes `POST /api/gesture`, documented as edge inference. What runs is `RuleRuntime`: four hand-written vectors — `fist`, `palm`, `ok`, `peace` — each 8 floats, scored by Euclidean distance with `confidence = clamp(1 - d/2, 0, 1)`. The gateway reports internal inference time of 0.70 μs, which is the honest number for four subtractions.

Figure 3 shows the three properties that follow. Exact template inputs return confidence 1.00 and the right label. Midpoints between two templates return a "top" label that is determined by arithmetic rather than by anything about hands: the fist/palm midpoint is classified `peace`, and an all-`0.5` input — which describes no hand at all — is classified `peace` with confidence 0.46. Noise robustness is correspondingly high (160/160 correct at $\sigma \leq 0.3$), which is what a four-point lookup should deliver; the classifier is simply not a model.

4.2 Video audit: the caller supplies the detections

`YololookUP` is documented as a YOLOv8 pipeline (Redmon et al., 2016; Jocher et al., 2023). The workspace contains 0 model files, and `Cargo.lock` contains no inference runtime — no `ort`, `tract`, `candle`, `tflite`, `ncnn`, `opencv` or `wgpu`. The only `Detector` implementation is `NoopDetector`, whose `detect` returns `Vec::new()`.

The HTTP layer compensates: `POST /api/video/audit` accepts a `detections` array from the caller and, if non-empty, swaps in a `StaticDetector` that replays it. Table 5 walks the consequence. Posting `{label: "person", confidence: 0.9}` with `motion_ratio 0.5` yields `critical`; the same detection with `motion_ratio 0` yields `suspicious`; a detection at confidence `0.50` is filtered by `min_confidence 0.55`; and a `toaster` at `0.99` is ignored because only `person` and `car` are on the watch list. Every level in the table is chosen by the request, not observed by the gateway.

Table 2: Crates by size and test count. The two crates the paper's thesis turns on – `YololookUP` (detection) and `SelflookUP` (self-audit) – carry no tests at all.

Crate	Files	Lines	Tests
`Server`	52	5,020	5
`NetScanear`	6	729	10
`ATOGrowUP`	4	527	7
`NetLinker`	5	403	3
`Core`	4	265	0
`SideAgent`	4	248	2
`AUTOTIME`	3	234	2
`VEScaner`	3	213	3
`HandModel`	3	172	0
`SelflookUP`	3	148	0
`YololookUP`	3	143	0
Total	90	8,102	32

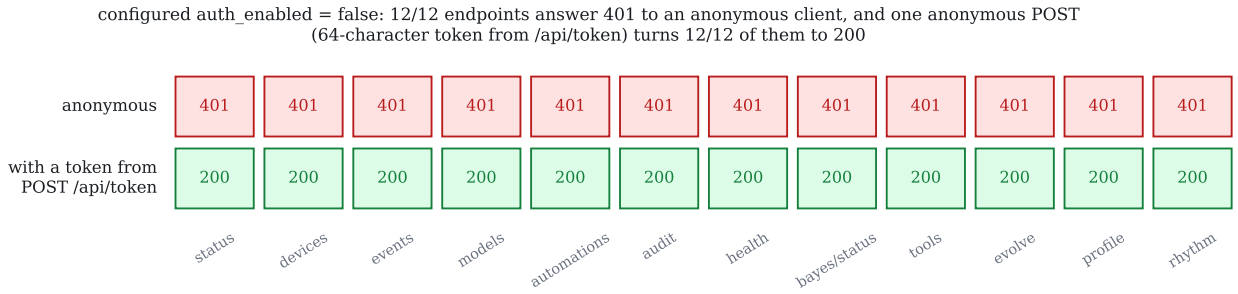


Figure 2: Twelve read endpoints, twice: as an anonymous client, and after one anonymous `POST /api/token`. The middleware allow-lists that endpoint, so the credential gate is a formality rather than a control.

Two mitigating facts, both measured. First, the fabricated event never enters the event log: across all 8 probes the log grew by 0 entries, so a fake `critical` cannot drive the automation engine or the webhook. Second, the request schema is strict — `bbox` is a required 4-tuple, so `POST`ing a detection without a box is rejected with HTTP 422. The finding is therefore not "the alerting pipeline can be spoofed end to end"; it is that the gateway's video severity is a function of its client's input, which is not what "video audit" means.

4.3 Models that are ready without weights

`seed_models` at start-up registers `yolo_v8n_person` with path `models/yolov8n.onnx` and sets `status = Ready` without opening the file; `POST /api/models` does the same for anything a client posts. We registered a deliberately impossible path and read it back as *ready*. The status endpoint consequently reports 3 ready models while 0/2 seeded paths exist on disk. Any dashboard built on `/api/status` will show an inference capability that does not exist.

5 Rules, evolution, habits

5.1 The rule engine is real

Table 6 is the good news. Rules are `{trigger, window, actions}` over an event stream, actions are `log`/`mqtt`/`ha`/`webhook`, and the engine works: a `contact=open` event from a registered device produced the rule's log action within 1,677.3 ms with a 1 s poll, and the webhook action delivered 5 HTTP POSTs to our sink with `level: alert` in the body. Trigger matching is selective: an event with a non-matching value added 1 log entry (the raw event) and fired nothing.

Two sharp edges. Three identical events in a row added 6 entries — there is no dedupe, no rate limit and no loop guard, so a flapping sensor or a rule whose action generates events can amplify. And rules live in an `RwLock<Vec<AutomationRule>>` with no persistence path (§7).

5.2 Self-evolution does not reach the rules

`ATOGrowUP` implements a genetic loop: `Evolver::evolve` sorts by fitness, keeps half the population, drifts hour windows by ±2, occasionally flips `Arm` to `Disarm`, and decays fitness by 0.9. Measured over 12 `POST /api/evolve/tick` calls, generation advances, the window of one gene moves from `[7,8]` to `[2,3]`, and fitness falls from 0.50 to 0.14 — so the code runs.

It is inert. Seeding ignores its request body and re-seeds three built-in genes, and the rule count is unchanged after all ticks (`automations_before = automations_after`): nothing in `automation.rs`

Table 3: Access control as measured with the documented default configuration.

Probe	Credentials	Result	Reading
GET `/api/status` and 11 more	anonymous	12/12 return 401	configuration says `auth_enabled = false`
POST `/api/token`	anonymous	200, 64-hex token	endpoint is on the middleware's allow-list
Same 12 probes	with that token	12/12 return 200	one unauthenticated request unlocks the surface
`/api/token/status`	anonymous	401	not on the allow-list
`/api/ws`	anonymous	400	allow-listed; the probe was a plain GET
`/`	anonymous	404	the WebUI bundle is not built in the repository

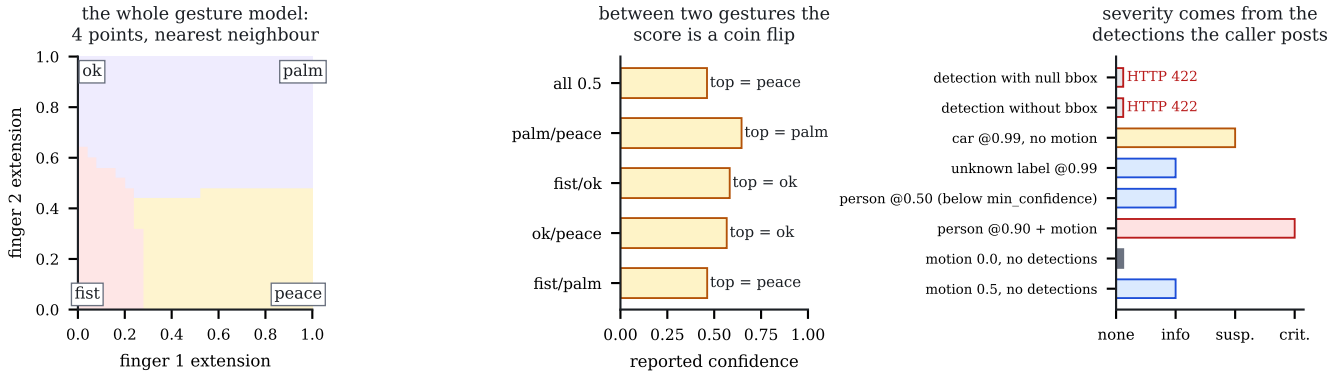


Figure 3: Left: the entire decision surface of the gesture classifier, interpolating two of the eight feature dimensions. Middle: confidence at midpoints between templates — the "winner" is an accident of the template coordinates. Right: the severity produced by `/api/video/audit` for each input; the two grey bars are requests rejected with HTTP 422 because `Detection.bbox` is a required tuple.

reads the evolver. `Evolver::reward`, which would close the fitness loop, has no call site anywhere in the workspace, as has `RuleGene::covers` — textbook dead code, CWE-561 (MITRE, 2024d). The `Core` message bus is the same story: four `broadcast` call sites in `coordinator.rs`, zero readers in the server.

5.3 Habit learning, and its degenerate case

The naive-Bayes classifier (Murphy, 2012) is genuinely implemented and genuinely reachable: 20 positive samples move the predicted probability from 0.50 to 1.00, and the device-event response exposes it as `habit_probability` with a verdict. But with zero negative samples the model answers 1.00 for *any* feature vector, including a pair never trained (Table 7). Laplace smoothing keeps the conditional probabilities finite; it does not stop the prior $\text{pos}/(\text{pos}+\text{neg})$ from saturating. A device that is allowed to learn from confirmations only will recommend allowing everything — and the log line that consumes this value is labelled "automatic execution reference".

6 Integration planes that work

6.1 Home Assistant, MQTT, WOL and MCP

Table 8 is the part of the system that exceeds its own documentation. Over tokio-tungstenite (Vinnik and Butler, 2024), against a mock HA server, the bridge performs a real `auth` → `auth_ok` → `get_states` exchange, mirrors 4 entities into 4 devices in the gateway's own ledger, accepts `POST /api/ha/control` and emits 5 `call_service` frames, and executes the `arm` scene across 3 entities. The MQTT client completes CONNECT and SUBSCRIBE against our broker (1 connection, 1 subscription to `pilhome/+/event`) with rumqttc (rumqttc contributors, 2024). WOL produces a correct magic packet: broadcast prefix, MAC repeated 16 times, captured on port 9. The MCP endpoint speaks JSON-RPC, lists 8 tools, and a tool call through MCP added an automation rule — an agent really can reconfigure the house. The `/api/ws` stream upgraded, pushed 14 frames in six seconds, and delivered a device event 0.1 ms after the POST that created it.

6.2 The scripting endpoint is the sharpest edge

`POST /api/tools` accepts JavaScript, runs it in QuickJS (Bellard, 2024; Shatilo, 2024), and injects 12 host bindings: `listDevices`, `deviceCommand`, `publishMqtt`, `wol`, `bayesPredict`, `httpGet`, `sendEvent`, `tts`, `generateImage`, `getState`, `setState`, `log`. We created a tool that called `sendEvent` — it appeared in the event log — and one that called `httpGet` against the harness sink, which received 1 request. That is arbitrary outbound HTTP from inside the gateway, available to anyone holding a token, which per §3 is anyone who can make

Table 4: The advertised edge-AI path as measured. *Gesture* is a 4-point lookup; *video* severity is computed from client-supplied detections.

Path	Measured	Detail
gesture classifier	4 templates of 8 floats	Euclidean nearest template, `confidence = 1 - d/2`
exact template input	top label correct for all 4	confidence 1.00, internal time 0.70 μs
fist/palm midpoint	top label = `peace`	confidence 0.46, spread over the 4 labels 0.08
all features 0.5	top label = `peace`	confidences 0.46, 0.45, 0.40, 0.36
noisy templates, σ up to 0.30	160/160 correct	4 labels $\times$ 5 noise levels $\times$ 40 samples
video detector	`NoopDetector` returns `Vec::new()`	no ONNX/TFLite dependency, 0 model files in the tree
video audit levels	8 probes, 0 events logged	level is decided by the detections the caller posts

Table 5: Every `/api/video/audit` response. The last two rows show that `Detection.bbox` is a required 4-tuple, so a detection cannot be expressed without a box. The final column is the change in the gateway's event log: a fabricated `critical` alert never reaches the automation engine.

Input (motion, detections)	HTTP	Level	Motion	Dets out	Events Δ
`motion 0.5, no detections`	200	`info`	–	0	0
`motion 0.0, no detections`	200	`(none)`	–	0	0
`person @0.90 + motion`	200	`critical`	–	1	0
`person @0.50 (below min_confidence)`	200	`info`	–	0	0
`unknown label @0.99`	200	`info`	–	1	0
`car @0.99, no motion`	200	`suspicious`	–	1	0
`detection without bbox (contract probe)`	422	`(none)`	–	0	0
`detection with null bbox (contract probe)`	422	`(none)`	–	0	0

one request. The docs describe this as "AI writes tools/flows"; from a security standpoint it is a remote-code surface with capability bindings, and it should be treated as the highest-risk component in the tree (MITRE, 2024e).

One implementation defect, measured because it changes what a user gets: tool ids are derived from the current second only, so two tools created in the same second share an id, and `run` resolves the first match. Our second tool, whose code returns a distinct string, executed the first tool instead (Table 9).

7 Configuration and state

7.1 The shipped configuration file cannot take effect

`config.toml` is the file the deployment guide (`docs/07`) tells an operator to edit. It has three independent defects, and the loader's response to all of them is the same: log one line and fall back to defaults.

D1. Syntax. Line 32 is `webhook_url = "` — an unterminated string, so the file is not valid TOML (Tom Programming Language, 2021).

D2. Schema. Fix only that, and parsing still fails: the `[probe]` table is missing `timeout_ms` and `concurrency`, which `ProbeConfig` requires.

D3. Nesting. Fix both, and the file parses but configures less than it appears to: `webhook_url` and `snapshot_interval_secs` are written *after* the `[probe]` header, so they become `probe.webhook_url` and `probe.snapshot_interval_secs` and are dropped; and the `[ai]` table has no counterpart in the flat `ai_base_url` field, so the advertised LLM endpoint is never loaded.

Figure 4 (left) and Table 10 show the same experiment run three times against the real binary, and Table 11 shows where the keys land once the file parses. The observable consequence of D1–D3 is that the gateway listens on `127.0.0.1:8080` — the compiled-in default — no matter what the operator writes, with a single `ERROR` line in the log as the only signal, and an insecure default that survives precisely because the failure is never fatal (CWE-1188) (MITRE, 2024a).

A fourth, quieter defect is that `cfg.scan` is never passed to `AppState`: the scanner reads `state.scan_cfg`, which is default initialised. Two scans (Table 12) took 19,960.5 and 19,986.5 ms with three loopback listeners open and found 0 alive hosts; the duration matches `254 hosts × 8 default ports` rather than the three configured ports. The same hard-coding appears in the MQTT client, which dials `host:1883` with the port fixed in source — and since the default `host` value is the string `"127.0.0.1:1883"`, the shipped configuration hands a *host:port* string to the client as a hostname, which fails name resolution at run time.

Table 6: The rule engine, which is the part of the system that genuinely works.

Probe	Measured	Reading
event → action latency	1,677.3 ms	engine polls every 1 s in this run
3 duplicate events	6 new log entries	no dedupe, no rate limit, no loop guard
non-matching event	1 (the raw event only)	trigger matching on key and value works
rule toggle	enabled false	rules live in an `RwLock<Vec<_>>` only
webhook action	5 POSTs received	raw HTTP/1.1, `level: alert` in the body

Table 7: Self-evolution and habit learning. The `Evolver::reward` entry point that would feed fitness back has no call site in the workspace.

Probe	Measured	Detail
evolve: seed body honoured	no	POST with a custom gene list yields the 3 built-in seeds
evolve: 12 ticks	generation 12	fitness 0.50 → 0.14
evolve: windows drift	yes	[7, 8] → [21, 22]
evolve: automations changed	2 → 2	the evolved population gates nothing
bayes: before training	0.50	prior with 0 samples
bayes: 20 positive samples	1.00	verdict `allow` (the source string is Chinese)
bayes: unseen feature pair	1.00	still 1.00: with no negatives the model answers 1.0 for anything

7.2 What survives a restart

The persistence layer snapshots events, audit records and behaviour profiles as pretty-printed JSON, and restores them at start-up — that much works, and after a restart we read back 4,096 events and 876 audit records (Figure 5, Table 13). Everything else is gone: 0 devices, 0 automation rules and all user-created tools. An operator who spent an evening writing rules loses them on the next reboot, and the gateway comes back with an empty policy set that still answers `GET /api/automations` with

.

Two related measurements. The snapshot interval is floored in code (`max(30)`), so the 2s we configured became 30 s — a silent override of the same kind the retry budget shows in companion work. And when the listen port is occupied, the gateway panics at `main.rs:101` *after* restoring its state and printing its banner, so a port collision takes the whole service down instead of degrading.

7.3 The audit log covers less than documented

the security-design document (`docs/06`) states that device registration, trust changes and scan execution are all written to an append-only audit log. Table 14 probes 14 actions and counts audit entries added. 8/14 are recorded; the six that are not are `device event`, `evolve tick`, `bayes train`, `ha sync`, `run scan` and `discover`. Device events and scans are precisely the two categories an intrusion review would ask for. The log is also not append-only: it is a `VecDeque` of capacity 4096 that evicts from the front, snapshotted by rewriting the whole file — 120,261 bytes per write in our run.

8 Discussion

8.1 The pattern

Read together, the findings are one shape repeated at five layers. A capability is named in prose — `auth_enabled`, `[scan]`, `edge AI`, `self-evolution`, `append-only audit` — and the code implements a nearby but different thing: a flag that is never read, a table that is never passed down, a nearest-neighbour lookup, an evolver with no consumer, a ring buffer. None of these are hard to fix, and none of them are visible from a demo: every one of them survives a manual walkthrough, because the person walking through it knows which parts are real.

This is design by contract in the original sense, inverted (Meyer, 1997): the preconditions and guarantees exist in the documentation and nowhere in the artifact. Consumer-driven contract testing (Pact Foundation, 2024) exists precisely for the integration half of this — the HA bridge and MQTT client are the two components where a mock peer would have caught the port hard-coding and the `host:port` default in an afternoon.

8.2 A twelve-line regression list

Each item below is one assertion against the harness in this paper, and each catches a finding reported above.

R1. `cargo build --release` at the tagged commit succeeds.

R2. With `auth_enabled = false`, `GET /api/status` returns 200.

R3. With `auth_enabled = true`, `POST /api/token` without credentials is refused.

R4. Every key in `config.toml` round-trips: parse it, serialise the `Config`, compare.

Table 8: Integration planes exercised against mock services. Each row is a real protocol exchange, not a stub answer.

Plane	Handshake	Measured
HA bridge	`auth`, `get_states`	4 entities served, 4 mirrored into the device ledger, 5 `call_service` frames observed
HA scene	`scene: arm`	3 entities actuated, HTTP 200
MQTT	1 CONNECT, 1 SUBSCRIBE	subscribes `pilhome/+/event`; 0 PUBLISH recorded in the probe window
WOL	HTTP 200	1 datagram, `ff`×6 prefix, MAC repeated 16 times
MCP	JSON-RPC 200	8 tools; `add_automation` through MCP raised the rule count to 3
JS tools	create and run 200	12 host bindings; `httpGet` reached the harness sink 1 time(s); `sendEvent` reached the event log: yes
WebSocket stream	HTTP/1.1 101 Switching Protocols	14 frames in 6 s, first at 0.1 ms

Table 9: Tool identifiers are derived from the current second `js_{unix\_secs mod 1e8}`, so tools created within the same second collide and the earlier definition shadows the later one.

Probe	IDs	Effect
two tools created in the same second	`js_92564939` = `js_92564939`	running the second id executed the first tool's code: yes

R5. Changing `[scan].prefix` changes the probe set (assert on the report, not the wall clock).

R6. `mqtt.host` with a port suffix is rejected at load time.

R7. `/api/models` never reports `Ready` for a path that does not exist.

R8. `/api/video/audit` rejects caller-supplied detections unless an explicit `allow_injected` flag is set.

R9. Two tools created in the same second get distinct ids.

R10. Restarting preserves devices, rules and tools, or the API refuses to accept them as durable.

R11. Every action listed in `docs/06` as audited adds an audit row.

R12. A second instance exits with a diagnostic, not a panic.

8.3 Advice to an operator of this build

Until the above are fixed: run the gateway behind a reverse proxy that terminates authentication and never expose `/api/token` or `/api/tools`; do not rely on `config.toml`, and verify the effective listen address on start-up; export `/api/automations` to your own storage after every change; and treat gesture and video alerts as telemetry from clients, not as detections by the device.

9 Related Work

Home-automation platforms. Home Assistant (Home Assistant, 2024) is the integration target here and documents the WebSocket protocol we mock (Home Assistant Developers, 2024); its own security model assumes a trusted LAN and explicit long-lived tokens. PILHOME is a re-implementation of a subset of that surface with an AI label attached.

IoT device security. NIST IR 8259 enumerates device-level capabilities — secure interfaces, verified configuration, updatable software, audit logging (National Institute of Standards and Technology, 2020) — which is the checklist our §7 findings map onto directly. The CWE entries cited above are the corresponding weakness classes.

LLM-agent control planes. The MCP specification (Model Context Protocol Project, 2025) defines exactly the tool-endpoint pattern PILHOME implements; the security interest in agent-controlled homes is recent and mostly about prompt injection rather than capability design, so we compare only on mechanism: 8 tools, reachable with a token mintable anonymously.

Silent failure. Gray failure — partial failure that keeps answering requests — is named by Huang et al. (Huang et al., 2017). A gateway that returns `ready` for a model file that does not exist, or `critical` for a detection nobody observed, is a gray failure by construction: the interface is healthy and the semantics are empty.

Embedded inference. Real edge detection on constrained hardware means an ONNX (ONNX Open Community, 2024) or TFLite runtime and a trained graph (Redmon et al., 2016); PILHOME has neither, and says it does.

10 Limitations

We measured one commit (`ed4a576`, 1 commit in the public history) with a shim we wrote; the `/api/token` behaviour is therefore "reachable once the missing file is supplied", and the shipped tree's behaviour at that endpoint is: does not build. Our mock peers implement documented subsets of HA and MQTT; a real broker or HA instance may behave differently

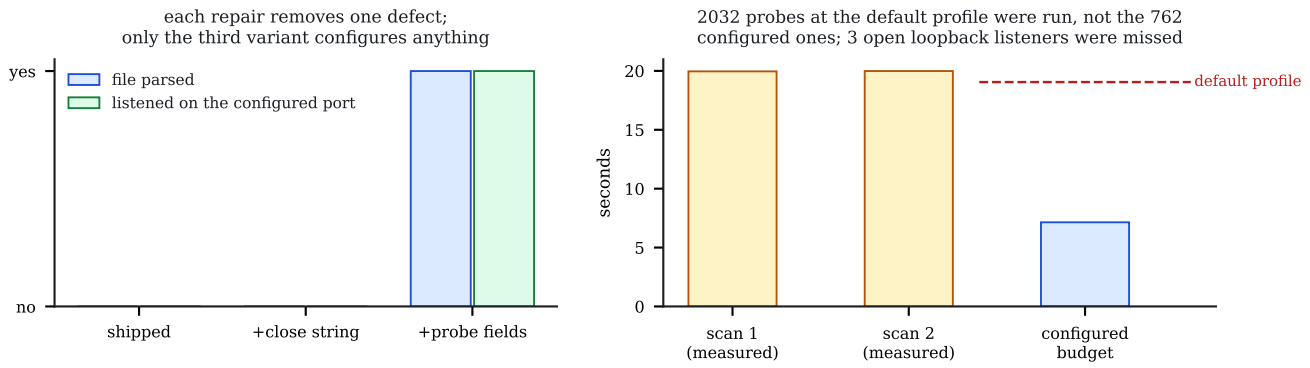


Figure 4: Left: three variants of the shipped `config.toml`, each starting the real binary. Only the third parses, and only then does the configured listen address apply. Right: two scans with three loopback listeners open; the wall time matches the 2,032-probe default profile, not the 762-probe configured one.

Table 10: A/B of the shipped `config.toml`. The file can never configure the gateway: it has a syntax defect, then a schema defect, and the run silently falls back to defaults with an `ERROR` log line.

Variant	Parses	Logged error	Effective listen
shipped file	no	`2026-10-21T06`	listening on `127.0.0.1:8080` (the built-in default)
unterminated string closed	no	`missing field 'timeout_ms'`	still the default port
plus the two missing `[probe]` fields	yes	–	listening on `127.0.0.1:18777` as written

in ways that would change the MQTT publish counts, though not the hard-coded port. Timing measurements come from a Python client on the same host and bound differences, not absolutes. We did not test the WebUI, the SQLite/MySQL paths, the AI chat proxy (whose base URL is never loaded, per D3), concurrency, or firmware/radio security. Finally, the "160/160 correct under noise" result is a property of a four-point lookup in an eight-dimensional space where the templates are far apart; a different template set would behave differently, and we did not attempt to optimise it.

11 Conclusion

PILHOME is a working rule engine, a working Home Assistant bridge, a working MQTT/WOL/MCP/WebSocket surface, and a working naive-Bayes habit model, wrapped in a release that does not compile, behind an authentication flag that is never read and a token endpoint that is its own bypass, configured by a file that cannot load, with an "edge AI" that is four vectors and a stub detector, a self-evolution loop that feeds nothing, and an audit log that misses device events and scans. Every one of those statements is a single command to reproduce, and the twelve assertions in Section 8 would turn this from a post-mortem into a release gate. For a security device, the gap between the advertised and the enforced capability *is* the vulnerability: not because the AI is missing, but because someone reading the documentation would configure the gateway as if it were there.

References

F. Bellard. QuickJS JavaScript engine. https://bellard.org/quickjs/, 2024.

A. Gupta and R. Miller. MQTT version 3.1.1. OASIS Standard, 2014. URL https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.

Home Assistant. Home assistant documentation. https://www.home-assistant.io/docs/, 2024.

Home Assistant Developers. Websocket API. https://developers.home-assistant.io/docs/api/websocket/, 2024.

P. Huang, C. Guo, L. Zhou, et al. Gray failure: The Achilles' heel of cloud-scale systems. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems (HotOS '17)*. ACM, 2017. doi: 10.1145/3102980.3103005. URL https://www.usenix.org/conference/hotos17/presentation/huang-peng.

G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLOv8. https://github.com/ultralytics/ultralytics, 2023.

B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 1997. Design by contract.

Table 11: Even after both parse defects are repaired, keys written at the bottom of the shipped file land inside the preceding `[probe]` table, where the deserialiser ignores them. `effective_webhook_url` in the parsed tree is absent.

Key	Top level	Under `[probe]`	Consequence
`webhook_url`	no	yes	`probe.webhook_url`
`snapshot_interval_secs`	no	yes	`probe.snapshot_interval_secs`
`[ai] base_url`	no	n/a	the Rust field is `ai_base_url` at top level

Table 12: Two scans with three loopback listeners open. Both take about 20.0 s, which matches the 2,032-probe default profile rather than the 762-probe configured one: `cfg.scan` is never handed to `AppState`, so the configured prefix and ports are inert.

Run	Wall time	Alive hosts	Probe budget
scan run 1	19,960.5 ms	0 alive	configured: prefix `127.0.0`, 3 ports (762 probes)
scan run 2	19,986.5 ms	0 alive	default profile: prefix `192.168.1`, 8 ports (2,032 probes)

MITRE. CWE-1188: Insecure default initialization of resource. https://cwe.mitre.org/data/definitions/1188.html, 2024a.

MITRE. CWE-287: Improper authentication. https://cwe.mitre.org/data/definitions/287.html, 2024b.

MITRE. CWE-306: Missing authentication for critical function. https://cwe.mitre.org/data/definitions/306.html, 2024c.

MITRE. CWE-561: Dead code. https://cwe.mitre.org/data/definitions/561.html, 2024d.

MITRE. CWE-94: Improper control of generation of code (“Code Injection”). https://cwe.mitre.org/data/definitions/94.html, 2024e.

Model Context Protocol Project. Model Context Protocol specification. https://modelcontextprotocol.io/specification/, 2025.

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012. ISBN 978-0-262-01802-9.

National Institute of Standards and Technology. Fundamentals of enterprise IoT device security and management. Technical Report NISTIR 8259, NIST, Oct. 2020. URL https://csrc.nist.gov/pubs/ir/8259/final.

ONNX Open Community. Open Neural Network Exchange (ONNX). https://onnx.ai/, 2024.

Pact Foundation. Consumer-driven contract testing for microservices. https://docs.pact.io/, 2024.

J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. doi: 10.1109/CVPR.2016.91. URL https://arxiv.org/abs/1506.02640.

rumqttc contributors. rumqttc: MQTT client for Rust. https://github.com/bytecodealliance/rumqttc, 2024.

S. Shatilo. rquickjs: High-level bindings for QuickJS. https://github.com/DelSkayn/rquickjs, 2024.

tokio-rs. axum: Web framework that Rust saw comes true. https://github.com/tokio-rs/axum, 2024a.

tokio-rs. Tokio: An asynchronous Rust runtime for writing network applications. https://github.com/tokio-rs/tokio, 2024b.

D. Tolnay. serde: A generic serialization framework for Rust. https://github.com/serde-rs/serde, 2024.

Tom Programming Language. TOML v1.0.0. https://toml.io/en/v1.0.0, 2021.

S. Vinnik and A. Butler. tokio-tungstenite: Websocket library for Tokio. https://github.com/snapview/tokio-tungstenite, 2024.

yxpil. PILHOME: Smart home security monitoring gateway (version 0.1.0). https://github.com/yxpil/PILHOME, 2026.

Table 13: What survives a restart, and how the boot sequence ends when the listen port is taken.

State	Before	After restart	Reading
events	4,096 in memory	4,096 after restart	survives
audit records	876 in memory	876 after restart	survives
devices	0 after restart	–	no snapshot path
automation rules	0	–	no snapshot path
user-created tools	11 (built-ins only)	–	lost
snapshot interval	configured 2 s	effective 30 s	`max(30)` floor
port already in use	panic at `main.rs:101`	after restoring 3 lines of state	the gateway exits instead of degrading

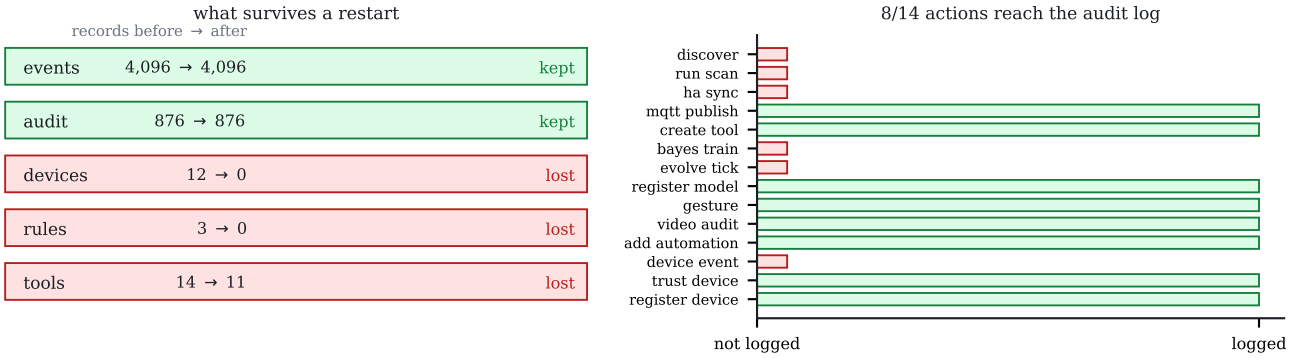


Figure 5: Left: records present before and after a restart of the same binary on the same data directory. Right: which security-relevant actions append to the audit log.

Table 14: 8 of 14 security-relevant actions append to the audit log. The unrecorded ones are exactly the actions a security review would want (the design document in `docs/06` lists them): device events, scans, discovery, HA control requests and evolution ticks.

Action	HTTP	Audit entries added	Recorded
`register device`	200	1	yes
`trust device`	200	1	yes
`device event`	200	0	*no*
`add automation`	200	1	yes
`video audit`	200	1	yes
`gesture`	200	1	yes
`register model`	200	1	yes
`evolve tick`	200	0	*no*
`bayes train`	200	0	*no*
`create tool`	200	1	yes
`mqtt publish`	200	1	yes
`ha sync`	200	0	*no*
`run scan`	200	0	*no*
`discover`	200	0	*no*