

WEPN home_device Security Audit Report

Date: 2026-05-29 | Branch: dev | Scope: post-2022 pentest

Executive Summary

The codebase shows meaningful hardening since the 2022 7ASecurity pentest: wepn-run correctly uses `SAFE_SNPRINTF` with bounds checking; the local-token-over-HTTP issue is properly resolved with a two-phase loopback design; TLS validation is correct throughout backend-facing code; SQL injection and Python-layer shell injection are cleanly mitigated; the AES-256-GCM E2EE implementation is correct.

One unauthenticated API endpoint (`/api/v1/claim/info`) is the most immediately abusable gap: any LAN device can retrieve the MQTT pairing key and E2EE key from an unclaimed Pod in one HTTP request, enabling silent device hijacking. The status file holding `local_token`, `e2e_key`, and `temporary_key` is written world-readable on every update.

The most systemic risk is a shared reverse-SSH private key (`shared_remote_key.priv`) identical across all Pods — one device breach gives fleet-wide relay access. This requires architectural remediation.

Immediate actions before any new deployment: add `valid_token()` to `claim_info()`, and change `wstatus.py:44` from `0o644` to `0o600`.

Critical & High Findings

H-01 — Unauthenticated `claim_info` exposes pairing and E2EE keys (P0) `usr/local/pproxy/local_server/api.py:112-131`

`claim_info()` has an exposed guard but no `valid_token()` call. For **unclaimed** devices, any LAN caller receives `temporary_key` (MQTT onboarding password) and `e2e_key` (AES-256 key for all E2EE messages). Attack chain: `curl -k https://<pod_ip>:5000/api/v1/claim/info` → POST to backend claim endpoint → device owned. For **claimed** devices, only `serial_number` + "CLAIMED" is returned — lower severity.

Fix: add `if not valid_token(request.args.get('local_token')): return "Not allowed", 401` before line 117.

H-02 — `status.ini` written `0644`; contains `local_token`, `e2e_key`, `temporary_key` (P1) `usr/local/pproxy/wstatus.py:44`

```
os.chmod(tmp_path, 0o644)    # every WStatus.save() call makes status.in
```

The `pi` user (semi-trusted) can `cat /var/local/pproxy/status.ini` and extract all four secrets. `post-install.sh` sets `status.bak` to `0600` — the live file deserves the same.

Fix: change `0o644` → `0o600`.

H-03 — Shared reverse-SSH key across all Pods; single breach → fleet-wide access (P1)

usr/local/pproxy/device.py:957

```
key = "/var/local/pproxy/shared_remote_key.priv" # identical on every
```

Compromising one Pod → extract key → SSH to relay.we-pn.com as remote@ → reach any other Pod's reverse tunnel at 9000 + <device_id>.

Fix: per-device SSH key pairs generated at provisioning; store public keys per-device in backend `authorized_keys` with port restrictions.

H-04 — Backend config overwrite with no signature verification (P1) usr/local/pproxy/device.py:851-870

`fetch_config_from_backend()` fetches from `https://config.we-pn.com` and writes directly to `/etc/pproxy/config.ini` with no cryptographic integrity check. A compromised backend or BGP/DNS hijack can push a malicious config pointing MQTT to an attacker-controlled broker.

Fix: sign configs with Ed25519 at the backend; embed the verification key in firmware.

H-05 — SHADOWSOCKS chain explicitly ACCEPTs 192.168.1.1 before the RFC1918 REJECT (P1) usr/local/sbin/ip-shadow.sh:3,12

```
iptables -t filter -A SHADOWSOCKS -d 192.168.1.1 -j ACCEPT      # line 3:
iptables -t filter -A SHADOWSOCKS -d 192.168.0.0/16 -j REJECT # line 8:
```

Shadowsocks clients can proxy to the home router admin interface. Line 12 is identical dead code after the catch-all.

Fix: remove both `192.168.1.1 ACCEPT` lines.

H-06 — No ip6tables SHADOWSOCKS equivalent; IPv6 traffic bypasses all egress blocks (P1) usr/local/sbin/ip-shadow.sh, usr/local/sbin/iptables-flush.sh

All RFC1918-blocking rules are IPv4-only. On dual-stack networks, Shadowsocks clients can reach IPv6 LAN addresses freely, and port 5000 may be internet-accessible over IPv6 (making H-01 internet-scale). No `disable_ipv6 sysctl` found in the codebase.

Fix: add `ip6tables DROP` rules mirroring the IPv4 chain, or add `net.ipv6.conf.all.disable_ipv6=1` to `/etc/sysctl.d/`.

H-07 — device_key logged unconditionally at INFO in mark_msg_read() (P1) usr/local/pproxy/messages.py:73

```
data_json = json.dumps({"device_key": ..., "serial_number": ..., "is_re
self.logger.info(data_json) # device_key in plaintext, not gated by D
```

Fix: remove the `logger.info` call or log only non-sensitive fields.

Medium Findings

ID	File	Line(s)	Description
M-01	periodic/recovery.py	12	Hardcoded logging-debug.ini path — recovery cron silently crashes on prod Pods lacking the debug config. Fix: from constants import LOG_CONFIG.
M-02	local_server/api.py	210–227	Error log readable without auth on unclaimed devices. Fix: require valid_token() unconditionally.
M-03	usr/local/sbin/update-system.sh	11	wget apt-key add uses deprecated global keyring; should use signed-by= scoped keyring already used in scommands[18].
M-04	setup/setuid.c	82,92	Any wepn-web member can mount hardcoded USB paths as root via wepn-run 1 11/20. Remove from service-accessible command set.
M-05	setup/setuid.c	89–90	Any wepn-web member can enable beta apt repo (wepn-run 1 18) then install packages (wepn-run 1 3) → root code execution via supply chain.
M-06	local_server/api.py	32–33, 233	tempfile.mkdtemp() at module load; shutil.rmtree only in __main__ (never reached under uWSGI) → accumulates directories in /tmp.
M-07	run.py	58	exec(open(UPDATE_SCRIPT).read()) — if update_config.py is writable by any non-pproxy user, code executes in pproxy context at boot. Replace with subprocess.run().
M-08	device.py	910–914	USB setup_mod.py imported without signature verification — arbitrary code execution with pproxy privileges from a crafted USB drive.

Low Findings

ID	File	Line(s)	Description
L-01	iptables-flush.sh	9	iptables -A OUTPUT -p icmp -j REJECT appended (not replaced) on every flush; duplicate rules accumulate.
L-02	local_server/api.py	56–57	valid_token() uses isalnum() — would silently break if token format ever adds non-alnum chars.
L-03	local_server/api.py	122–131	Manual JSON concatenation in claim_info() without escaping. Use json.dumps().
L-04	usr/local/sbin/update-pproxy.sh	8,10	TOCTOU on predictable /tmp/update-out; chown root:root \$LOG follows symlinks. Use mktemp.
L-05	device.py	960	StrictHostKeyChecking=accept-new — MITM window on first connection to relay. Embed host key in firmware.

Regression Status

2022 Finding	Status
IP resolution over HTTP	Fixed — <code>https://ip.we-pn.com</code> with cert validation
<code>local_token</code> sent to spoofable IP	Fixed — 2-phase loopback design; token only goes to <code>127.0.0.1</code>
Passwordless <code>sudo</code>	Fixed — replaced by <code>wepn-run</code> with <code>SAFE_SNPRINTF</code> and indexed commands
Dangerous <code>cron-upnp.sh</code>	Fixed — dangerous commands commented out
<code>iptables RFC1918</code> blocking	Improved/Partial — chain exists but H-05 (explicit <code>192.168.1.1</code> <code>ACCEPT</code>) and H-06 (no IPv6) remain

Prioritized Remediation Roadmap

Immediate (this week):

1. `api.py` — add `valid_token()` to `claim_info()` (H-01)
2. `wstatus.py:44` — `0o644` → `0o600` (H-02)
3. `messages.py:73` — remove `logger.info(data_json)` (H-07)
4. `recovery.py:12` — from `constants` import `LOG_CONFIG` (M-01)

Short-term (this sprint): 5. `iptables-flush.sh` + `ip-shadow.sh` — add `ip6tables` rules or disable IPv6 (H-06) 6. `ip-shadow.sh` — remove `192.168.1.1` `ACCEPT` lines (H-05) 7. `api.py:210` — require `valid_token()` unconditionally on error log endpoint (M-02) 8. `update-system.sh:11` — migrate to `signed-by=scoped` keyring (M-03) 9. `run.py:58` — replace `exec()` with `subprocess.run()` (M-07) 10. `iptables-flush.sh:9` — flush `OUTPUT` chain before appending (L-01) 11. `update-pproxy.sh:8` — use `mktemp` for log path (L-04)

Architectural (next quarter): 12. Per-device SSH key pairs for reverse tunnel (H-03) 13. Cryptographic signature on `config.we-pn.com` responses (H-04) 14. Remove or signature-gate USB provisioning code path (M-08) 15. Remove USB mount commands from service-accessible `wepn-run` set (M-04) 16. Gate beta apt repo commands behind separate auth (M-05)