# Pwdreadme

## Purpose

Harden and observe user password integrity in the WattSun backend. This guide sets up:

- \*\*Audit logging\*\* of password changes in SQLite

- \*\*Guard triggers\*\* that \*block\* any rogue password change or delete

- A \*\*canary loop\*\* to detect external DB overwrites

- Quick forensics and runtime checks

## Paths & Assumptions

- DB file: `/volume1/web/wattsun/user-setup/users.db`

- Table uses column: `password\_hash` (not `password`)

Set an env var for convenience:

DB="/volume1/web/wattsun/user-setup/users.db"

## 1) Backup (always do this first)

[ -f "$DB" ] && echo "Found: $DB" || echo "NOT FOUND: $DB"

sudo cp "$DB" "${DB}.$(date +%Y%m%d-%H%M%S).bak"

ls -lh "${DB}."\*.bak | tail -n1

## 2) Create Audit Table + Triggers (log every password write)

Logs inserts, password updates, and deletes to `users\_audit`.

sqlite3 "$DB" <<'SQL'

PRAGMA journal\_mode=WAL;

CREATE TABLE IF NOT EXISTS users\_audit (

id INTEGER PRIMARY KEY AUTOINCREMENT,

user\_id INTEGER,

action TEXT,

col TEXT,

old\_value TEXT,

new\_value TEXT,

changed\_at DATETIME DEFAULT CURRENT\_TIMESTAMP

);

DROP TRIGGER IF EXISTS trg\_users\_insert;

CREATE TRIGGER IF NOT EXISTS trg\_users\_insert

AFTER INSERT ON users

BEGIN

INSERT INTO users\_audit (user\_id, action, col, old\_value, new\_value)

VALUES (NEW.id, 'INSERT', 'password\_hash', NULL, NEW.password\_hash);

END;

DROP TRIGGER IF EXISTS trg\_users\_update\_password;

CREATE TRIGGER IF NOT EXISTS trg\_users\_update\_password

AFTER UPDATE OF password\_hash ON users

WHEN OLD.password\_hash IS NOT NEW.password\_hash

BEGIN

INSERT INTO users\_audit (user\_id, action, col, old\_value, new\_value)

VALUES (NEW.id, 'UPDATE', 'password\_hash', OLD.password\_hash, NEW.password\_hash);

END;

DROP TRIGGER IF EXISTS trg\_users\_delete;

CREATE TRIGGER IF NOT EXISTS trg\_users\_delete

AFTER DELETE ON users

BEGIN

INSERT INTO users\_audit (user\_id, action, col, old\_value, new\_value)

VALUES (OLD.id, 'DELETE', 'password\_hash', OLD.password\_hash, NULL);

END;

SQL

Check:

sqlite3 "$DB" "SELECT name FROM sqlite\_master WHERE type='trigger' AND tbl\_name='users';"

sqlite3 "$DB" "SELECT COUNT(\*) AS audit\_rows FROM users\_audit;"

## 3) Guard Triggers (BLOCK bad changes)

Blocks any password change unless via \*\*valid reset\*\* or \*\*Admin\*\*, and blocks \*all\* deletes.

sqlite3 "$DB" <<'SQL'

DROP TRIGGER IF EXISTS trg\_guard\_password\_update;

CREATE TRIGGER trg\_guard\_password\_update

BEFORE UPDATE OF password\_hash ON users

WHEN NEW.password\_hash IS NOT OLD.password\_hash

AND NOT (

OLD.type = 'Admin'

OR (

COALESCE(LENGTH(OLD.reset\_token),0) > 0

AND CAST(COALESCE(OLD.reset\_expiry,0) AS INTEGER) > CAST(strftime('%s','now') AS INTEGER)

)

)

BEGIN

SELECT RAISE(ABORT, 'Blocked password change: not via valid reset and not Admin');

END;

DROP TRIGGER IF EXISTS trg\_guard\_user\_delete;

CREATE TRIGGER trg\_guard\_user\_delete

BEFORE DELETE ON users

BEGIN

SELECT RAISE(ABORT, 'Blocked delete of users table');

END;

CREATE UNIQUE INDEX IF NOT EXISTS idx\_users\_email ON users(email);

CREATE UNIQUE INDEX IF NOT EXISTS idx\_users\_phone ON users(phone);

SQL

Test (non‑admin `id=2` shown as example):

TEST\_ID=2

sqlite3 "$DB" "UPDATE users SET password\_hash='blocked\_try' WHERE id=$TEST\_ID;" || true

sqlite3 "$DB" "

UPDATE users

SET reset\_token='ok', reset\_expiry = (CAST(strftime('%s','now') AS INTEGER) + 3600)

WHERE id=$TEST\_ID;"

sqlite3 "$DB" "UPDATE users SET password\_hash='ok\_after\_reset' WHERE id=$TEST\_ID;"

sqlite3 "$DB" "UPDATE users SET reset\_token=NULL, reset\_expiry=NULL WHERE id=$TEST\_ID;"

## 4) Canary Loop (detect whole‑file overwrites)

Runs every 10 minutes without cron.

cat > /volume1/web/wattsun/watch-users-hash.sh <<'BASH'

#!/usr/bin/env bash

DB="/volume1/web/wattsun/user-setup/users.db"

OUT="/tmp/users\_dump.txt"

LOG="/tmp/users\_dump.hashlog"

sqlite3 "$DB" "SELECT id,email,substr(password\_hash,1,16) FROM users ORDER BY id;" > "$OUT"

sha256sum "$OUT" >> "$LOG"

echo "$(date '+%F %T') snap done" >> /tmp/users\_dump.timestamps

BASH

sudo chmod +x /volume1/web/wattsun/watch-users-hash.sh

nohup bash -c 'while true; do /volume1/web/wattsun/watch-users-hash.sh; sleep 600; done' >/tmp/watch-users-hash.loop.log 2>&1 &

echo "Canary loop PID: $!"

Check:

tail -n3 /tmp/users\_dump.hashlog

tail -n3 /tmp/users\_dump.timestamps

## 5) What to run when “invalid credentials” happens

sqlite3 "$DB" "

SELECT action, user\_id, col, changed\_at

FROM users\_audit

WHERE changed\_at >= datetime('now','-6 hours')

ORDER BY changed\_at DESC

LIMIT 50;"

tail -n3 /tmp/users\_dump.hashlog

/volume1/web/wattsun/watch-users-hash.sh

tail -n3 /tmp/users\_dump.hashlog

Interpretation:

- Audit shows UPDATE/DELETE → app/script tried it (blocked unless valid reset/Admin).

- Audit empty but hash flipped → external overwrite.

## Minimum vs. Optional

Minimum: Backup, Audit, Guards, Unique Indexes, Canary.

Optional: Route DB‑path logs, Reset-route logging, Soft‑delete pattern.