Security Architecture

Drop Security Architecture

Last updated: 2026-02-14 **Source:** Security audit (`security/drop-security-rapport.md`), hardening implementation (`security/security-hardening-implementation.md`), source code

> **Architecture model:** Drop uses a PSD2 pass-through model. It never holds customer money — AISP reads bank balances via Open Banking, PISP initiates payments from the user's bank account. Cards are a FUTURE feature, gated behind feature flags (all default to false).

Authentication

JWT Token Management

Library: `jose` ^6.1.3 (well-maintained, no known vulnerabilities) **Algorithm:** HS256 with explicit `setProtectedHeader` **Source:** `src/drop-app/src/lib/auth.ts`

Setting	Value	Source
Algorithm	HS256	`auth.ts`
Expiry	24 hours	`auth.ts` cookie `maxAge: 60 * 60 * 24`
`setIssuedAt()`	Yes	Prevents token reuse
Secret (production)	`JWT_SECRET` env var	Fatal error if missing
Secret (development)	`process.cwd()` hash	Dev fallback for uniqueness

Cookie Configuration

Source: `src/drop-app/src/lib/auth.ts:48-54`

Property	Value	Purpose

`httpOnly`	`true`	Prevents JavaScript access (XSS mitigation)
`secure`	`true` (production)	HTTPS-only cookie transport
`sameSite`	`"strict"`	CSRF prevention
`maxAge`	24 hours	Session lifetime
`path`	`"/"`	Cookie scope

Password Hashing

Library: `bcryptjs` ^3.0.3 (pure JS, no native compilation issues) **Source:** `src/drop-app/src/lib/utils-server.ts:8-16`

Setting	Value
Algorithm	bcrypt
Cost factor	12
SHA-256 fallback	**Removed** (security fix C4)

After hardening, `verifyPassword()` only accepts bcrypt hashes. SHA-256 legacy support has been removed entirely.

Session Management

Source: `src/drop-app/src/lib/auth.ts:45-65`, `src/lib/middleware.ts:42-77`

Sessions are tracked in the `sessions` table:

Column	Type	Purpose
`id`	TEXT PK	Session identifier (format: `ses_<hex16>`)
`user_id`	TEXT FK	References `users.id`
`token_hash`	TEXT	SHA-256 hash of JWT token
`expires_at`	TEXT	Expiration timestamp
`revoked`	INTEGER	0 = active, 1 = revoked

Lifecycle:

1. **Login** -- Session created with token hash (`auth.ts:56-65`)
2. **Each request** -- Session checked for revocation (`middleware.ts:66-74`)
3. **Logout** -- All user sessions revoked server-side (`auth/logout/route.ts:5-14`)

Authorization

IDOR Protection

All data access queries include `AND user_id = ?` to scope data to the authenticated user:

- Recipients: scoped to user
- Cards: scoped to user
- Transactions: scoped to user
- Bank accounts: scoped to user
- Notifications: scoped to user
- Settings: scoped to user

Merchant endpoints verify merchant role and ownership.

Role-Based Access

Roles (from `lib/db.ts` schema `CHECK` constraint):

- `user` -- Standard user
- `merchant` -- Merchant with dashboard access

KYC Status (required for financial operations):

- `pending` -- Default on registration
- `approved` -- Required for remittance
- `rejected` -- Blocked from financial operations

Input Validation

Rate Limiting

Source: `src/drop-app/src/lib/middleware.ts:6-31`

Endpoint Type	Limit	Window
Auth routes (login, register)	10 requests	60 seconds
Transaction routes (remittance, qr-payment)	10 requests	60 seconds

Endpoint Type	Limit	Window
Rate endpoints (/api/rates)	120 requests	60 seconds

Implementation: PostgreSQL-backed persistent rate limiting (survives restarts). Per-IP tracking using `X-Forwarded-For` header. (ADR-014: SQLite removed 2026-03-03)

Rate limit table (`rate_limits`):

- `key` -- IP address (TEXT PK)
- `count` -- Request count (INTEGER)
- `reset_at` -- Window expiry (INTEGER, Unix timestamp)

CSRF Protection

Source: `src/drop-app/src/lib/middleware.ts:44-55`

Origin header validation on all authenticated requests:

- Validates `Origin` header against allowed origins list
- Allowed origins: `NEXT_PUBLIC_APP_URL`, `http://localhost:3000`, `http://localhost:3001`
- Combined with `sameSite: "strict"` cookies

Input Sanitization

Source: `src/drop-app/src/lib/middleware/validation.ts:149-203`

- `sanitizeText()` -- Removes HTML tags, control characters, trims whitespace, enforces max length
- `validateName()` -- Rejects XSS payloads, script tags, numbers-only names
- `validateEmail()` -- Regex validation for email format
- `validatePhone()` -- International format validation
- `validateAmount()` -- Positive, finite, max 2 decimal places
- `validateIBAN()` -- Format and checksum validation
- `validatePIN()` -- Exactly 4 digits
- `validateCurrency()` -- Whitelist: EUR, USD, GBP, BAM, CHF, PLN, NOK, RSD, TRY, PKR
- `validateLanguage()` -- Whitelist: nb, en, bs, sq

Applied to: recipients, merchants, settings, notifications.

Amount Validation

Endpoint	Min	Max	Source

Remittance	100 NOK	50,000 NOK	`transactions/remittance/route.ts`
QR Payment	1 NOK	100,000 NOK	`transactions/qr-payment/route.ts`

Additional checks: `Number.isFinite()` to prevent NaN/Infinity injection. Pagination limited to max 50 per page.

SQL Injection Prevention

All 24 API endpoints use **parameterized queries** exclusively (`?` placeholders). No string concatenation in SQL statements.

Example from `transactions/route.ts`:

```
// Dynamic WHERE clauses use parameter arrays
const conditions: string[] = [];
const params: unknown[] = [];
if (type) { conditions.push("type = ?"); params.push(type); }
```

Merchant dashboard uses strict whitelist for `period` parameter.

Security Headers

Source: `src/drop-app/next.config.ts:6-46`

Header	Value
Content-Security-Policy	`default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; ...`
X-Frame-Options	`DENY`
X-Content-Type-Options	`nosniff`
Referrer-Policy	`strict-origin-when-cross-origin`
Permissions-Policy	`camera=(self), microphone=(), geolocation=(self)`
Strict-Transport-Security	`max-age=63072000; includeSubDomains; preload`

Known limitation: CSP includes `unsafe-inline` and `unsafe-eval` (required for Next.js dev mode). Should be tightened with nonce-based CSP for production.

API Response Masking

- **Card numbers:** Masked as `**** **** **** XXXX` in responses (`cards/[id]/route.ts:25-35`)
- **CVV:** Displayed as `***` in responses
- **Bank accounts:** Only last 4 digits visible (`utils-server.ts:23-26`)

Transaction Integrity

Source: `transactions/remittance/route.ts`, `transactions/qr-payment/route.ts`

- Atomic balance deduction using PostgreSQL transactions (Drizzle ORM)
- `WHERE balance >= $1` prevents overdraw
- PostgreSQL MVCC with explicit `FOR UPDATE` row locking
- Fee calculated and included in deduction

Feature Flags

Source: `src/drop-app/src/lib/feature-flags.ts`

Gated features (disabled by default):

Flag	Env Var	Default
`virtualCards`	`NEXT_PUBLIC_FF_VIRTUAL_CARDS`	`false`
`physicalCards`	`NEXT_PUBLIC_FF_PHYSICAL_CARDS`	`false`
`cardDetails`	`NEXT_PUBLIC_FF_CARD_DETAILS`	`false`
`cardFreeze`	`NEXT_PUBLIC_FF_CARD_FREEZE`	`false`
`cardPin`	`NEXT_PUBLIC_FF_CARD_PIN`	`false`
`spendingLimits`	`NEXT_PUBLIC_FF_SPENDING_LIMITS`	`false`
`notifications`	`NEXT_PUBLIC_FF_NOTIFICATIONS`	`true`
`merchantDashboard`	`NEXT_PUBLIC_FF_MERCHANT_DASHBOARD`	`true`

API routes check feature flags and return 404 when disabled.

Dependency Security

Package	Version	Risk Assessment
`jose`	^6.1.3	Low -- Well-maintained JWT library
`bcryptjs`	^3.0.3	Low -- Pure JS bcrypt
`drizzle-orm`	latest	Low -- Type-safe ORM, parameterized queries
`next`	16.1.6	Low -- Recent version
`react`	19.2.3	Low -- Latest major
`radix-ui`	^1.4.3	Low -- UI components only

No known vulnerable dependencies identified (from `security/drop-security-rapport.md:400-411`).

Revision #12
Created 2026-02-23 11:29:01 UTC by John
Updated 2026-06-28 20:02:25 UTC by John