Security Measures

Overview

FiredUp implements comprehensive security measures to protect user data and ensure secure integration with Tink. This document details the security controls, encryption methods, access management, and monitoring systems in place.

Data Protection

Encryption in Transit

Protocol	Usage
TLS 1.2+	All API communications
HTTPS	All web traffic (enforced via HSTS)
Certificate	Valid SSL certificate from Let’s Encrypt

All communications between: - Users and FiredUp (frontend/API) - FiredUp backend and Tink API - FiredUp backend and database

…are encrypted using TLS 1.2 or higher.

Encryption at Rest

Data	Encryption Method
Database	PostgreSQL with encrypted storage volume
Tokens	Stored in database (protected by DB access controls)
Backups	Encrypted before storage

No Bank Credential Storage

Critical Security Design: - FiredUp **NEVER** stores, sees, or processes bank login credentials - Bank authentication happens directly on the bank's website via Tink Link - We only receive and store OAuth tokens (access + refresh) - Tokens are scoped and limited (read-only access)

Authentication & Authorization

User Authentication

Method	Technology
Web	NextAuth.js with Google OAuth
Mobile	Google Sign-In + JWT
Biometric	Face ID / Touch ID (mobile)

API Authentication

Layer	Protection
Frontend → Backend	Internal secret header + user ID
Mobile → Backend	JWT Bearer token
Backend → Tink	OAuth 2.0 client credentials + user tokens

Authorization Scopes (Tink)

FiredUp requests **minimal scopes** - only what's needed for read-only access:

Scope	Purpose	Access Type
`user:create`	Create Tink user	Client-level
`authorization:grant`	Generate auth codes	Client-level
`accounts:read`	Read account list	User-level
`transactions:read`	Read transaction history	User-level

Scope	Purpose	Access Type
`credentials:read`	Read credential status	User-level
`credentials:write`	Initial credential setup	User-level
`balances:read`	Read account balances	User-level

Scopes NOT requested: - ❌ `payment:write` (no payment initiation) - ❌ `beneficiaries:write` (no beneficiary management) - ❌ Any administrative scopes

---

CSRF Protection

State Token Implementation

The OAuth flow uses HMAC-SHA256 signed state tokens to prevent CSRF attacks:

```
State Token = base64(random_bytes(32) + HMAC-SHA256(random_bytes, secret))
```

Protection mechanism: 1. Generate cryptographically random state token (32 bytes) 2. Sign with HMAC-SHA256 using server-side secret 3. Store in `tink_pending_auth` table with expiration (15 minutes) 4. Include in Tink Link redirect URL 5. On callback, verify signature and check database 6. Mark token as used (single-use enforcement)

Code Location

- `backend/app/services/tink_service.py` - `generate_state_token()`, `verify_state_token()`

Rate Limiting

Per-User Rate Limits

Endpoint Category	Limit	Window
Auth operations	10 requests	1 minute
Sync operations	5 requests	5 minutes

Endpoint Category	Limit	Window
General API	100 requests	1 minute

Implementation

Rate limiting is implemented using token bucket algorithm per user ID.

Tink API Rate Limits

FiredUp respects Tink's rate limits: - Honors `Retry-After` header - Implements exponential backoff - Caps retry delay at 60 seconds

---

Retry Logic & Error Handling

Exponential Backoff

For transient errors (5xx, 429), FiredUp implements retry with exponential backoff:

```
Delay = min(base_delay × 2^attempt, max_delay) ± jitter
```

Parameter	Value
Base delay	1 second
Max delay	30 seconds
Max attempts	3
Jitter	±25%

Retryable vs Non-Retryable Errors

Status Code	Action
429 (Rate Limited)	Retry with Retry-After header
500, 502, 503, 504	Retry with backoff
400, 401, 403, 404	Fail immediately (client error)

Status Code	Action
422	Fail immediately (validation error)

Code Location

- `backend/app/services/tink_service.py` - `_is_retryable_status()`, `_calculate_backoff_delay()`, `_parse_retry_after()`

Audit Logging

TinkAuditLog Model

All Tink-related operations are logged to the `tink_audit_logs` table:

Field	Description
`user_id`	User performing the action
`tink_connection_id`	Related connection (if applicable)
`action_type`	Type of operation
`result`	success / failure / partial
`request_method`	HTTP method
`request_path`	API endpoint
`status_code`	HTTP response code
`ip_address`	Client IP (IPv6 compatible)
`user_agent`	Client user agent
`metadata`	Additional context (sanitized)
`created_at`	Timestamp

Action Types

Action Type	Trigger
`connect_initiated`	User starts bank connection
`connection_created`	Successful connection
`connection_failed`	Connection error
`connection_disconnected`	User disconnects bank
`token_refreshed`	Access token refreshed
`transactions_synced`	Transactions fetched
`transaction_reviewed`	User reviews transaction
`debug_access`	Admin debug access
`data_refreshed`	Data refresh operation

Data Sanitization

Audit logs **NEVER** contain: - ❌ Access tokens - ❌ Refresh tokens - ❌ Bank credentials - ❌ Full transaction details - ❌ Personal financial data

Logs only contain: - ✅ Counts (e.g., “synced 25 transactions”) - ✅ Error categories (e.g., “rate_limited”) - ✅ Anonymized identifiers

---

Monitoring & Alerting

Error Monitoring (Sentry)

Feature	Configuration
Platform	Sentry (cloud)
Environments	Production, Staging
Alert Threshold	>5% error rate
Notification	Email + Slack

Captured Data

- Exception stack traces
- Request context (sanitized)
- User ID (for debugging)
- Environment information

Excluded from Sentry

- ❌ Access tokens
- ❌ Financial data
- ❌ Personal information

Metrics Monitoring

TinkMetricsService tracks: - Request counts by endpoint - Response times (latency percentiles) - Error rates by error type - Token refresh frequency - Sync operation success rates

---

Access Control

Database Access

Role	Access Level
Application	Read/Write (via connection string)
Admin	SSH required + MFA
Backups	Automated, encrypted

Server Access

Method	Requirement
SSH	Key-based authentication only
Root	Sudo required
Firewall	Only ports 22, 80, 443 open

Code Access

Repository	Protection
GitHub	Private repo, branch protection
Secrets	Environment variables (not in code)
CI/CD	GitHub Actions with secrets

Incident Response

Severity Levels

Level	Description	Response Time
Critical	Data breach, service down	< 1 hour
High	Security vulnerability	< 4 hours
Medium	Functionality impacted	< 24 hours
Low	Minor issues	< 72 hours

Response Procedures

1. **Detection** - Automated alerts or user report
2. **Triage** - Assess severity and impact
3. **Containment** - Isolate affected systems
4. **Investigation** - Root cause analysis
5. **Remediation** - Fix and deploy
6. **Communication** - Notify affected users (if required)
7. **Post-mortem** - Document lessons learned

Tink-Specific Incidents

For Tink-related security incidents: 1. Revoke affected user tokens immediately 2. Notify Tink support 3. Review audit logs for scope 4. Notify affected users

---

Security Checklist

Implementation Status

Control	Status	Evidence
HTTPS everywhere	✅ Implemented	SSL Labs A+ rating
Token encryption	✅ Implemented	Database storage with access controls
No credential storage	✅ Implemented	Code review confirmed
CSRF protection	✅ Implemented	HMAC-SHA256 state tokens
Rate limiting	✅ Implemented	Per-user limits
Retry logic	✅ Implemented	Exponential backoff
Audit logging	✅ Implemented	TinkAuditLog table
Error monitoring	✅ Implemented	Sentry integration
Access controls	✅ Implemented	SSH keys, firewalls

Compliance

Standards Alignment

Standard	Status
OWASP Top 10	Mitigations in place
GDPR	Compliant (see Data Handling doc)
PSD2	Via Tink (licensed AISP)

Tink’s Certifications (our provider)

Tink holds: - SOC 2 Type II - ISO 27001 - PCI DSS (where applicable)

Document Revision

Version	Date	Changes
1.0	February 2026	Initial version