Security & Compliance

Security architecture, RBAC, encryption, GDPR compliance

- Security Architecture
- GDPR & Compliance
- Bilko CIAM abuse-gate fix — checkBefore moved outside SERIALIZABLE tx (MC #104069, root-cause of #103245)

Security Architecture

Bilko — Security Architecture

Status: PLANNED (backend not built yet, security measures documented for implementation)

This document defines the security architecture for Bilko, a financial SaaS handling sensitive accounting data.

Security Principles

1. **Defense in Depth** — Multiple layers of security (network, application, database)
2. **Least Privilege** — Users and services get minimum necessary permissions
3. **Zero Trust** — Verify every request, never assume trust
4. **Encryption Everywhere** — Data encrypted in transit and at rest
5. **Immutable Audit Trail** — All actions logged, tamper-proof

Security Layers Diagram

```
graph TD
    CLIENT["Client Browser / PWA"]

    subgraph NETWORK["Network Layer"]
        CF["Cloudflare\nDDoS Protection\nTLS 1.3 termination\nHSTS"]
    end

    subgraph APP_LAYER["Application Layer"]
        HELMET["Helmet.js\nCSP + X-Frame + HSTS\nno X-Powered-By"]
        CORS["CORS Whitelist\nbilko.io only\nno wildcard *"]
        RATE["Rate Limiter\nexpress-rate-limit\n5 req/min auth\n100 req/min general"]
        AUTH_MW["Auth Middleware\nJWT verify\norg-scope injection"]
        RBAC_MW["RBAC Middleware\nrole check\nowner / admin / accountant / viewer"]
        ZOD["Zod Validation\nall request bodies\ntype-safe parsing"]
    end
```

```
    subgraph DATA_LAYER["Data Layer"]
        PRISMA_ORM["Prisma ORM\nparameterized queries\nno raw SQL for user input\norg-scoped WHERE"]
        PG_ENC["PostgreSQL Railway\nAES-256 disk encryption\nbackup encryption"]
    end

    subgraph AUDIT["Audit Layer"]
        LOG["LoggedAction table\nAPPEND-ONLY\nIP + user + timestamp\nold/new values"]
    end

    CLIENT --> CF --> HELMET --> CORS --> RATE --> AUTH_MW --> RBAC_MW --> ZOD --> PRISMA_ORM --> PG_ENC
    PRISMA_ORM --> LOG
```

Authentication

Strategy: JWT (JSON Web Tokens)

Why JWT?

- Stateless (scales horizontally)
- Works with mobile PWA
- Industry standard

Token Types

Access Token

- **Lifetime:** 15 minutes
- **Storage:** `Authorization: Bearer <token>` header
- **Contains:** User ID, organization ID, role
- **Refresh:** Automatic via refresh token

Refresh Token

- **Lifetime:** 7 days
- **Storage:** httpOnly cookie (not accessible to JavaScript)
- **Purpose:** Obtain new access token
- **Rotation:** New refresh token issued on each refresh
- **Revocation:** Stored in database, can be invalidated

JWT Payload Example

```
{
  "sub": "user-uuid",
  "org": "org-uuid",
  "role": "admin",
  "iat": 1640000000,
  "exp": 1640000900
}
```

Token Flow

```
1. User logs in → POST /api/v1/auth/login
   ← Access token (header) + Refresh token (httpOnly cookie)

2. User makes request → GET /api/v1/invoices (Authorization: Bearer <access>)
   ← Protected resource

3. Access token expires (15 min) → POST /api/v1/auth/refresh (httpOnly cookie)
   ← New access token + New refresh token

4. User logs out → POST /api/v1/auth/logout
   → Delete refresh token from DB
   ← 204 No Content
```

JWT Auth Flow (Sequence)

```
sequenceDiagram
    actor User
    participant FE as Frontend (bilko.io)
    participant API as Express API (api.bilko.io)
    participant DB as PostgreSQL

    User->>FE: Enter email + password
    FE->>API: POST /api/v1/auth/login
    API->>DB: SELECT user WHERE email = ?
    DB-->>API: User record (passwordHash)
    API->>API: bcrypt.compare(password, hash)
```

```
    alt Password valid
        API->>API: jwt.sign({sub, org, role}, JWT_SECRET, 15m)
        API->>API: jwt.sign({sub}, JWT_REFRESH_SECRET, 7d)
        API->>DB: INSERT refreshToken (hashed, expiresAt)
        API-->>FE: 200 { accessToken } + Set-Cookie: refreshToken (httpOnly)
        FE->>FE: Store accessToken in memory
    else Password invalid
        API-->>FE: 401 Unauthorized
    end

    Note over FE,API: 15 minutes later — access token expires
    FE->>API: POST /api/v1/auth/refresh (Cookie: refreshToken)
    API->>DB: SELECT refreshToken WHERE token = ? AND expiresAt > NOW()
    DB-->>API: Valid token record
    API->>API: Rotate: delete old, issue new refresh token
    API->>DB: DELETE old refreshToken, INSERT new refreshToken
    API-->>FE: 200 { newAccessToken } + Set-Cookie: newRefreshToken

    Note over User,DB: User logs out
    User->>FE: Click logout
    FE->>API: POST /api/v1/auth/logout
    API->>DB: DELETE refreshToken WHERE userId = ?
    API-->>FE: 204 No Content
    FE->>FE: Clear accessToken from memory
```

Implementation (Backend)

```
import jwt from 'jsonwebtoken';
import bcrypt from 'bcrypt';

// Generate access token
const accessToken = jwt.sign(
  { sub: user.id, org: user.organizationId, role: user.role },
  process.env.JWT_SECRET!,
  { expiresIn: '15m' }
);

// Generate refresh token
const refreshToken = jwt.sign(
```

```
  { sub: user.id },
  process.env.JWT_REFRESH_SECRET!,
  { expiresIn: '7d' }
);

// Store refresh token in DB (for revocation)
await prisma.refreshToken.create({
  data: {
    token: refreshToken,
    userId: user.id,
    expiresAt: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000),
  },
});
```

Password Security

Hashing: bcrypt

Algorithm: bcrypt with 12 salt rounds

Why bcrypt?

- Designed for passwords (slow by design, resists brute force)
- Auto-salted (each password has unique salt)
- Adaptive (can increase rounds as hardware improves)

Password Requirements

- **Minimum length:** 8 characters
- **Complexity:** At least one uppercase, one lowercase, one number
- **No common passwords:** Check against list of 10K most common passwords
- **No reuse:** Previous 5 passwords stored (hashed) and blocked

Implementation

```
import bcrypt from 'bcrypt';

// Hash password (registration)
```

```
const passwordHash = await bcrypt.hash(password, 12);

// Verify password (login)
const isValid = await bcrypt.compare(password, user.passwordHash);
```

Two-Factor Authentication (2FA)

Strategy: TOTP (Time-based One-Time Password)

Compatible with:

- Google Authenticator
- Authy
- 1Password
- Microsoft Authenticator

Setup Flow

```
1. User enables 2FA → POST /api/v1/auth/2fa/setup
   ← QR code + secret (base32)

2. User scans QR code in authenticator app
   → Generates 6-digit code

3. User verifies code → POST /api/v1/auth/2fa/verify { code }
   ← 200 OK (2FA enabled)
```

Login Flow with 2FA

```
1. User logs in → POST /api/v1/auth/login { email, password }
   ← 200 OK + { requires2FA: true, tempToken }

2. User enters code → POST /api/v1/auth/2fa/login { tempToken, code }
   ← Access token + Refresh token
```

Backup Codes

Generate 10 single-use backup codes during 2FA setup:

- Stored hashed (bcrypt)
- Used when authenticator unavailable
- Marked as used after redemption

Authorization (RBAC)

RBAC Permission Model

```
classDiagram
    class Owner {
        +createInvoice()
        +editInvoice()
        +deleteInvoice()
        +viewInvoice()
        +approveExpense()
        +generateReport()
        +inviteUser()
        +editOrgSettings()
        +deleteOrg()
    }
    class Admin {
        +createInvoice()
        +editInvoice()
        +viewInvoice()
        +approveExpense()
        +generateReport()
        -deleteInvoice()
        -inviteUser()
        -editOrgSettings()
    }
    class Accountant {
        +viewInvoice()
        +viewExpense()
```

```
        +generateReport()
        -createInvoice()
        -editInvoice()
        -approveExpense()
        -inviteUser()
    }
    class Viewer {
        +viewDashboard()
        +viewReports()
        -createInvoice()
        -editInvoice()
        -generateReport()
        -inviteUser()
    }

    Owner --|> Admin : inherits all Admin permissions
    Admin --|> Accountant : inherits read access
    Accountant --|> Viewer : inherits view access
```

Roles

Role	Permissions
owner	Full access (edit org settings, invite users, delete data)
admin	Manage invoices, expenses, contacts, reports (no org settings)
accountant	Read invoices/expenses, create reports (no edit)
viewer	Read-only access (dashboard, reports)

Permission Matrix

Action	owner	admin	accountant	viewer
Create invoice	□	□	□	□
Edit invoice	□	□	□	□
Delete invoice	□	□	□	□
View invoice	□	□	□	□
Approve expense	□	□	□	□

Action	owner	admin	accountant	viewer
Generate report	✅	✅	✅	✅
Invite user	✅	✅	❌	❌
Edit org settings	✅	✅	❌	❌

Implementation (Middleware)

```
import { Request, Response, NextFunction } from 'express';

function requireRole(roles: string[]) {
  return (req: Request, res: Response, next: NextFunction) => {
    if (!roles.includes(req.user.role)) {
      return res.status(403).json({ error: 'Forbidden' });
    }
    next();
  };
}

// Usage
app.post('/api/v1/invoices', requireRole(['owner', 'admin']), createInvoice);
```

Encryption

In Transit: TLS 1.3

All traffic encrypted via HTTPS:

- Frontend (Vercel): Automatic HTTPS
- Backend (Railway): Automatic HTTPS
- Certificate: Let's Encrypt (auto-renewed)

TLS Configuration:

- Minimum version: TLS 1.3
- Cipher suites: Modern only (no legacy ciphers)
- HSTS enabled (Strict-Transport-Security header)

At Rest: Database Encryption

PostgreSQL (Railway):

- Disk encryption: AES-256 (Railway default)
- Backup encryption: AES-256
- Column-level encryption: Not needed (disk encryption sufficient for accounting data)

Cloudflare R2 (Files):

- Server-side encryption: AES-256 (default)
- No client-side encryption needed (files are receipts/invoices, not PII)

Secrets Management

NEVER commit secrets to git:

- `.env` files in `.gitignore`
- Use platform-provided secrets (Vercel, Railway)
- Rotate JWT secrets quarterly
- Rotate API keys annually

OWASP Top 10 Mitigations

1. Injection (SQL Injection)

Mitigation: Prisma ORM parameterized queries

```
// SAFE — Prisma auto-escapes
await prisma.invoice.findMany({
  where: { customerId: req.params.id }
});

// UNSAFE — Never use raw SQL for user input
await prisma.$queryRaw`SELECT * FROM invoices WHERE customer_id = ${req.params.id}`;
```

---

2. Broken Authentication

Mitigations:

- bcrypt password hashing (12 rounds)
- JWT with short expiry (15 min)
- Refresh token rotation
- 2FA (TOTP)
- Rate limiting on auth endpoints (5 req/min)

---

3. Sensitive Data Exposure

Mitigations:

- TLS 1.3 in transit
- AES-256 at rest
- No PII in JWTs (only user ID)
- No passwords in logs
- No sensitive data in URLs (use POST body)

---

4. XML External Entities (XXE)

Not applicable — Bilko does not parse XML.

---

5. Broken Access Control

Mitigations:

- RBAC enforced on every endpoint
- Organization-scoped queries (middleware)
- No direct object reference (use UUIDs, not auto-increment IDs)

```
// Organization scoping middleware
app.use('/api/v1/*', (req, res, next) => {
  req.prismaWhere = { organizationId: req.user.organizationId };
  next();
});

// Apply to queries
await prisma.invoice.findMany({ where: req.prismaWhere });
```

---

6. Security Misconfiguration

Mitigations:

- Helmet.js security headers
- CORS whitelist (no `*` in production)
- Error messages sanitized (no stack traces in production)
- Disable `X-Powered-By` header

```
import helmet from 'helmet';

app.use(helmet({
  contentSecurityPolicy: {
    directives: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'", "'unsafe-inline'"], // Next.js requires unsafe-inline
      styleSrc: ["'self'", "'unsafe-inline'"],
      imgSrc: ["'self'", "data:", "https:"],
    },
  },
}));
```

---

7. Cross-Site Scripting (XSS)

Mitigations:

- React auto-escapes output (default safe)
- CSP headers (Content-Security-Policy)
- Sanitize user input (Zod validation)
- No `dangerouslySetInnerHTML` without sanitization

```
// SAFE — React escapes by default
<p>{invoice.description}</p>

// UNSAFE — Only use with sanitized HTML
<div dangerouslySetInnerHTML={{ __html: sanitizedHTML }} />
```

---

8. Insecure Deserialization

Not applicable — Bilko does not deserialize untrusted data.

---

9. Using Components with Known Vulnerabilities

Mitigations:

- Dependabot alerts enabled (GitHub)
- Weekly `npm audit` checks
- Automated security updates (Dependabot PRs)
- Lock file committed (`package-lock.json`)

---

10. Insufficient Logging & Monitoring

Mitigations:

- Audit trail (LoggedAction table)
- Error tracking (Sentry recommended)
- Access logs (Railway built-in)
- Failed login attempts logged
- Anomaly detection (future: alert on 10+ failed logins)

Rate Limiting

Rate Limiting Flow

```
flowchart TD
    REQ["Incoming Request"]
    ENDPOINT{{"Endpoint?"}}

    AUTH_CHECK{{"IP count\n≤5 per 15min?"}}
    REG_CHECK{{"IP count\n≤3 per 60min?"}}
    REFRESH_CHECK{{"IP count\n≤10 per 15min?"}}
    REPORT_CHECK{{"User count\n≤10 per 15min?"}}
    GENERAL_CHECK{{"IP count\n≤100 per 15min?"}}

    PASS["Pass to route handler"]
    BLOCK["429 Too Many Requests\n'Try again later'"]

    REQ --> ENDPOINT
    ENDPOINT -->|"/auth/login"| AUTH_CHECK
```

```
    ENDPOINT -->|"/auth/register"| REG_CHECK
    ENDPOINT -->|"/auth/refresh"| REFRESH_CHECK
    ENDPOINT -->|"/reports/*"| REPORT_CHECK
    ENDPOINT -->|"all other /api/*"| GENERAL_CHECK

    AUTH_CHECK -->|Yes| PASS
    AUTH_CHECK -->|No| BLOCK
    REG_CHECK -->|Yes| PASS
    REG_CHECK -->|No| BLOCK
    REFRESH_CHECK -->|Yes| PASS
    REFRESH_CHECK -->|No| BLOCK
    REPORT_CHECK -->|Yes| PASS
    REPORT_CHECK -->|No| BLOCK
    GENERAL_CHECK -->|Yes| PASS
    GENERAL_CHECK -->|No| BLOCK
```

Prevent brute force and abuse:

Endpoint	Limit	Window
`/api/v1/auth/login`	5 requests	15 minutes
`/api/v1/auth/register`	3 requests	60 minutes
`/api/v1/auth/refresh`	10 requests	15 minutes
`/api/v1/*` (general)	100 requests	15 minutes
`/api/v1/reports/*`	10 requests	15 minutes

Implementation

```
import rateLimit from 'express-rate-limit';

const authLimiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 5,
  message: 'Too many login attempts, try again later',
});

app.post('/api/v1/auth/login', authLimiter, loginHandler);
```

Input Validation

All inputs validated with **Zod** schemas:

Example: Invoice Validation

```
import { z } from 'zod';

const createInvoiceSchema = z.object({
  customerId: z.string().uuid(),
  invoiceDate: z.string().regex(/^\d{4}-\d{2}-\d{2}$/),
  dueDate: z.string().regex(/^\d{4}-\d{2}-\d{2}$/),
  currencyCode: z.enum(['EUR', 'RSD', 'BAM', 'HRK']),
  items: z.array(z.object({
    description: z.string().min(1).max(500),
    quantity: z.number().positive(),
    unitPrice: z.number().nonnegative(),
    taxRate: z.number().min(0).max(100),
  })),
});

// Middleware
function validate(schema: z.ZodSchema) {
  return (req, res, next) => {
    try {
      req.body = schema.parse(req.body);
      next();
    } catch (error) {
      res.status(400).json({ error: error.errors });
    }
  };
}

// Usage
app.post('/api/v1/invoices', validate(createInvoiceSchema), createInvoice);
```

File Upload Security

Allowed File Types

- **Receipts:** JPG, PNG, PDF
- **Max size:** 10MB per file

Validation

```
import multer from 'multer';
import path from 'path';

const upload = multer({
  limits: { fileSize: 10 * 1024 * 1024 }, // 10MB
  fileFilter: (req, file, cb) => {
    const allowedTypes = ['.jpg', '.jpeg', '.png', '.pdf'];
    const ext = path.extname(file.originalname).toLowerCase();
    if (allowedTypes.includes(ext)) {
      cb(null, true);
    } else {
      cb(new Error('Invalid file type'));
    }
  },
});
```

Virus Scanning (Planned)

Phase 2: Integrate ClamAV for virus scanning before upload to R2.

Audit Trail

LoggedAction Table (Immutable)

All mutations logged:

- Table name

- Action (INSERT, UPDATE, DELETE)
- User ID
- Timestamp
- Old values (UPDATE/DELETE)
- New values (INSERT/UPDATE)
- Client IP
- SQL query

Example Audit Log Entry

```
{
  "eventId": 12345,
  "tableName": "invoices",
  "action": "UPDATE",
  "userId": "user-uuid",
  "actionTimestamp": "2026-02-20T10:30:00Z",
  "rowData": { "id": "invoice-uuid", "status": "draft" },
  "changedFields": { "status": { "old": "draft", "new": "sent" } },
  "clientIp": "192.168.1.10"
}
```

Audit Queries

```
// Get user activity
await prisma.loggedAction.findMany({
  where: { userId: 'user-uuid' },
  orderBy: { actionTimestamp: 'desc' },
  take: 100,
});

// Get invoice history
await prisma.loggedAction.findMany({
  where: {
    tableName: 'invoices',
    rowData: { path: ['id'], equals: 'invoice-uuid' },
  },
});
```

Data Retention & Deletion

User Data Deletion (GDPR Right to Erasure)

Process:

1. User requests deletion → POST /api/v1/account/delete
2. Soft delete user record (mark `deletedAt`)
3. Anonymize LoggedAction entries (replace user ID with "deleted-user")
4. Delete PII (email, name)
5. **Keep financial records** (required by law, minimum 5 years)

Soft Delete Implementation:

```
await prisma.user.update({
  where: { id: userId },
  data: {
    email: `deleted-${userId}@example.com`,
    fullName: 'Deleted User',
    passwordHash: '',
    deletedAt: new Date(),
  },
});
```

Security Testing

Static Analysis

- **ESLint:** Security rules enabled (no-eval, no-unsafe-regex)
- **TypeScript:** Strict mode (catches type errors)

Dependency Scanning

- **npm audit:** Weekly checks
- **Dependabot:** Automatic PRs for vulnerabilities

Penetration Testing (Future)

- **Phase 2:** Hire security firm for penetration testing
- **Scope:** Auth, API, file uploads, SQL injection
- **Frequency:** Annually

Incident Response Plan

Detection

- Monitor error rates (Sentry)
- Monitor failed login attempts (>10 in 1 hour = alert)
- Railway metrics (CPU spike, memory leak)

Response

1. **Identify:** What is the breach? (data leak, DDoS, unauthorized access)
2. **Contain:** Block attacker IP, revoke compromised tokens
3. **Eradicate:** Fix vulnerability, patch code
4. **Recover:** Restore from backup if needed
5. **Document:** Write post-mortem, update security docs

Notification

- **Internal:** Slack alert to #security channel
- **External:** Email users if PII compromised (GDPR 72h requirement)

Security Checklist (Pre-Launch)

- ☐ JWT secrets generated (32+ chars)
- ☐ HTTPS enforced (no HTTP allowed)
- ☐ CORS whitelist configured (no `*`)
- ☐ Rate limiting enabled (auth endpoints)
- ☐ Helmet.js security headers configured
- ☐ bcrypt password hashing (12 rounds)
- ☐ Prisma queries parameterized (no raw SQL)
- ☐ Input validation (Zod schemas)

- [ ] File upload restrictions (type, size)
- [ ] Audit trail enabled (LoggedAction)
- [ ] Error messages sanitized (no stack traces)
- [ ] Dependabot alerts enabled
- [ ] Backup strategy tested
- [ ] Incident response plan documented
- [ ] Security review completed

Related Documents

- Compliance: COMPLIANCE.md
- Deployment: ../infrastructure/DEPLOYMENT.md
- Testing: ../testing/TESTING-GUIDE.md

Last Updated: 2026-02-20 **Status:** PLANNED — Backend not built yet, security measures to be implemented **Compliance:** OWASP Top 10, GDPR Article 32 (Security of Processing)

GDPR & Compliance

Bilko — Regulatory Compliance

Status: NOT COMPLIANT — Requires legal review and implementation (Phase 2)

This document outlines regulatory compliance requirements for Bilko as a Balkan accounting SaaS.

Compliance Scope

Bilko operates in a highly regulated space:

Region	Regulations
EU/EEA	GDPR (General Data Protection Regulation)
Serbia	Zakon o računovodstvu, SEF (Sistem E-Faktura)
Bosnia & Herzegovina	Zakon o PDV-u, Electronic bookkeeping requirements
Croatia	Zakon o fiskalizaciji, eRačun (public sector invoicing)

Current Status: MVP focuses on GDPR compliance. Balkan-specific regulations deferred to Phase 2.

Compliance Roadmap by Phase

```
graph LR
    subgraph P1["Phase 1 — MVP (pre-launch)"]
        GDPR["GDPR\nData minimization\nEncryption TLS+AES-256\nUser rights endpoints\nDPA with processors"]
    end

    subgraph P2["Phase 2 — Serbia Launch (3-6mo)"]
        RS_COA["Serbian CoA\n(Kontni plan template)"]
        RS_VAT["VAT 20%\nReporting"]
        RS_REP["Financial Reports\nBilans stanja\nBilans uspeha"]
        RS_SEF["SEF Integration\nB2G e-invoicing\n(optional at MVP)"]
```

```
    end

    subgraph P3["Phase 3 — Regional (12-18mo)"]
        BIH["BiH\nVAT 17%\nPDV prijava"]
        HR["Croatia\nVAT 25%\nFiskalizacija 2.0\neRačun B2G\nDigital signature"]
    end

    P1 --> P2 --> P3
```

GDPR (General Data Protection Regulation)

Applicability

- **Applies to:** All EU/EEA users (regardless of where Bilko is hosted)
- **Scope:** Personal data of natural persons (name, email, IP address)
- **Penalties:** Up to €20M or 4% of global turnover (whichever is higher)

Data We Collect

Data Type	Purpose	Legal Basis	Retention
Email	Account authentication	Contract performance	Until account deletion
Full name	User identification	Contract performance	Until account deletion
IP address	Security audit trail	Legitimate interest	30 days
Password (hashed)	Authentication	Contract performance	Until account deletion
Organization name	Service delivery	Contract performance	5 years (accounting law)
Financial records	Service delivery	Legal obligation	5-10 years (varies by country)

GDPR Principles Compliance

1. Lawfulness, Fairness, Transparency (Article 5(1)(a))

Implementation:

- Privacy policy visible before registration

- Terms of Service linked during signup
- Clear explanation of data usage
- No hidden data collection

Status: PLANNED — Privacy policy to be drafted

---

2. Purpose Limitation (Article 5(1)(b))

Implementation:

- Data used only for stated purposes (accounting, invoicing)
- No data selling to third parties
- No marketing emails without explicit consent

Status: COMPLIANT (by design)

---

3. Data Minimization (Article 5(1)(c))

Implementation:

- Only collect necessary data (email, name)
- No tracking cookies
- No analytics beyond server logs

Status: COMPLIANT (by design)

---

4. Accuracy (Article 5(1)(d))

Implementation:

- Users can update profile (email, name)
- Users can correct financial data (invoices, expenses)

Status: COMPLIANT (by design)

---

5. Storage Limitation (Article 5(1)(e))

Implementation:

- User data deleted on request (soft delete)
- Financial records retained 5 years (legal requirement overrides GDPR Article 17)
- Audit logs kept 30 days

Status: PLANNED — Deletion workflow to be implemented

6. Integrity & Confidentiality (Article 5(1)(f))

Implementation:

- TLS 1.3 encryption in transit
- AES-256 encryption at rest
- bcrypt password hashing
- Access controls (RBAC)

Status: PLANNED — See SECURITY-ARCHITECTURE.md

GDPR Data Flow Diagram

```
flowchart TD
    USER["User (Data Subject)"]
    REG["Registration\nPOST /api/v1/auth/register"]
    STORE_PII["Store PII\nRailway EU West (Frankfurt)\nAES-256 at rest\nemail, name,
passwordHash (bcrypt)"]
    PROCESS["Service Processing\nInvoices, Expenses, Reports\nOrganization data"]
    LOG["Audit Trail\nLoggedAction table\nIP + timestamp (30 days)"]
    FILE["File Storage\nCloudflare R2 EU\nReceipts + PDFs\nAES-256"]

    subgraph THIRD["Third-Party Processors (DPA Required)"]
        RAILWAY["Railway EU West\n(DB hosting)"]
        VERCEL["Vercel\n(Frontend hosting)"]
        CF_R2["Cloudflare R2 EU\n(File storage)"]
        SG["SendGrid\n(Transactional email)"]
    end

    USER -->|"Consent via ToS"| REG --> STORE_PII
    STORE_PII --> PROCESS --> LOG
    PROCESS --> FILE

    STORE_PII --> RAILWAY
    REG --> VERCEL
    FILE --> CF_R2
    PROCESS -->|"Invoice emails"| SG
```

GDPR Rights (Articles 12-22)

Right to Access (Article 15)

User can request:

- Copy of all personal data
- Purpose of processing
- Data retention period

Implementation:

```
// Endpoint: GET /api/v1/account/data
await prisma.user.findUnique({
  where: { id: userId },
  include: { organization: true, auditLogs: true },
});
```

Status: PLANNED

Right to Rectification (Article 16)

User can:

- Update email, name
- Correct invoices, expenses

Implementation:

```
// Endpoint: PATCH /api/v1/account/profile
await prisma.user.update({
  where: { id: userId },
  data: { email, fullName },
});
```

Status: PLANNED

Right to Erasure (Article 17)

Exceptions:

- Financial records must be kept 5 years (legal obligation overrides)
- Audit logs anonymized (user ID replaced with "deleted-user")

Implementation:

```
// Endpoint: DELETE /api/v1/account
await prisma.user.update({
  where: { id: userId },
  data: {
    email: `deleted-${userId}@example.com`,
    fullName: 'Deleted User',
    passwordHash: '',
    deletedAt: new Date(),
  },
});
```

Status: PLANNED

Right to Data Portability (Article 20)

User can:

- Export all data in JSON format

Implementation:

```
// Endpoint: GET /api/v1/account/export
const data = {
  user: await prisma.user.findUnique({ where: { id: userId } }),
  invoices: await prisma.invoice.findMany({ where: { organizationId } }),
  expenses: await prisma.expense.findMany({ where: { organizationId } }),
};
res.json(data);
```

Status: PLANNED

Right to Object (Article 21)

Not applicable — Bilko does not use profiling or automated decision-making.

Data Processing Agreement (DPA)

Required when Bilko processes customer data on behalf of organizations.

Third-Party Processors:

Service	Purpose	DPA Available?	GDPR Compliant?
Railway	Database hosting	Yes	Yes (EU region)
Vercel	Frontend hosting	Yes	Yes
Cloudflare	R2 storage, DNS	Yes	Yes
SendGrid	Transactional email	Yes	Yes

Action Required: Sign DPAs with all processors before launch.

Status: PENDING

Data Breach Notification (Article 33)

Requirement:

- Notify supervisory authority within 72 hours of breach
- Notify affected users if high risk to rights and freedoms

Process:

1. Detect breach (monitoring, user report)
2. Assess impact (how many users, what data)
3. Contain breach (block attacker, revoke tokens)
4. Notify authority (within 72h)
5. Notify users (if high risk)
6. Document incident (post-mortem)

Breach Notification Flow

```
sequenceDiagram
    participant MON as Monitoring (Sentry / Railway)
    participant JOHN as John (AI Director)
    participant ALEM as Alem (CEO)
    participant AUTH as Supervisory Authority
    participant USERS as Affected Users

    MON->>JOHN: Alert: anomaly detected
    JOHN->>JOHN: Assess impact\n(data type, user count)
    JOHN->>JOHN: Contain: revoke tokens\nblock attacker IP
```

```
    JOHN->>ALEM: Breach report + impact summary

    alt High risk to users
        ALEM->>AUTH: Notify within 72h (GDPR Art. 33)
        ALEM->>USERS: Email notification\n(nature of breach, data affected,\nsteps taken)
    else Low risk
        ALEM->>AUTH: Optional notification
        Note over USERS: No user notification required
    end

    JOHN->>JOHN: Post-mortem\nUpdate security docs\nPatch vulnerability
```

Status: PLANNED — Incident response plan documented in SECURITY-ARCHITECTURE.md

---

Data Protection Officer (DPO)

Required? No — Bilko does not meet GDPR Article 37 criteria:

- Not a public authority
- Not large-scale systematic monitoring
- Not large-scale processing of sensitive data

Threshold: DPO required if >250 employees or large-scale processing. Bilko is small startup.

Status: NOT REQUIRED (as of 2026-02-20)

Data Residency

Requirement: Store EU user data within EU/EEA (GDPR Article 44-50)

Implementation:

- Railway: EU West region (Frankfurt or Paris)
- Vercel: Edge network (serves from EU for EU users)
- Cloudflare R2: EU region

Status: PLANNED — Configure Railway to EU region on deployment

---

Serbia — Zakon o ra?unovodstvu (Accounting Law)

Applicability

- **Applies to:** All legal entities in Serbia
- **Scope:** Financial record-keeping, reporting, retention

Requirements

1. Chart of Accounts

Regulation: Companies must use standardized chart of accounts (Kontni plan)

Implementation:

- Bilko allows custom chart of accounts
- Provide Serbian CoA template (predefined accounts)

Status: PLANNED — Create Serbian CoA seed data

---

2. Double-Entry Bookkeeping

Regulation: All transactions must use double-entry (debit + credit)

Implementation:

- Prisma schema enforces double-entry (`debitAccountId` + `creditAccountId`)
- Backend validates debit = credit

Status: COMPLIANT (by design)

---

3. Financial Reporting

Required reports:

- Bilans stanja (Balance Sheet)
- Bilans uspeha (Income Statement)
- Izvještaj o novčanim tokovima (Cash Flow Statement)

Implementation:

- Bilko generates P&L, Balance Sheet, Cash Flow
- Export to PDF (Serbian language support)

Status: PLANNED — Backend report generation

---

4. Data Retention

Regulation: Financial records must be kept minimum 5 years

Implementation:

- Soft delete (never hard delete financial data)
- Backup retention: 30 days (Railway automatic backups)

Status: PLANNED

SEF (Sistem E-Faktura) — Electronic Invoicing

Requirement: B2G (business-to-government) invoices must be submitted electronically via SEF portal.

Applicability:

- Mandatory for government contracts
- Optional for B2B (as of 2026)

Implementation (Phase 2):

- SEF XML export format
- API integration with SEF portal
- Digital signature (qualified certificate)

Status: NOT IMPLEMENTED — Deferred to Phase 2

Bosnia & Herzegovina — Zakon o PDV-u (VAT Law)

VAT Rates

- **Standard:** 17%
- **Reduced:** 0% (exports, specific goods)

Requirements

1. VAT Calculation

Implementation:

- Bilko supports configurable tax rates per invoice item
- Default tax rate: 17% for BiH organizations

Status: COMPLIANT (by design)

---

2. VAT Reporting

Required report:

- PDV prijava (VAT return) — monthly or quarterly

Implementation:

- Bilko generates VAT report (sales, purchases, net VAT)
- Export to PDF

Status: PLANNED — Backend report generation

---

3. Electronic Bookkeeping

Regulation: Companies with revenue >50,000 BAM must maintain electronic records.

Implementation:

- Bilko is cloud-based (electronic by default)
- Data export to XML (future integration with tax authority)

Status: PLANNED (Phase 2)

Croatia — Zakon o fiskalizaciji (Fiscalization Law)

Applicability

- **Applies to:** All businesses with cash transactions (retail, hospitality, services)

Requirements

1. Fiscalization (Fiskalizacija 2.0)

Regulation: All invoices must be registered with tax authority in real-time.

Implementation (Phase 2):

- API integration with Porezna uprava (tax authority)
- Digital signature (qualified certificate)
- Unique invoice identifier (JIR) from tax authority
- QR code on invoice (links to tax authority verification)

Status: NOT IMPLEMENTED — Deferred to Phase 2

---

2. eRa?un (Public Sector Invoicing)

Requirement: B2G invoices must be submitted via eRačun system.

Implementation (Phase 2):

- UBL XML format
- Integration with eRačun portal

Status: NOT IMPLEMENTED — Deferred to Phase 2

Multi-Country Compliance Matrix

Requirement	Serbia	BiH	Croatia	Implementation Status
Double-entry bookkeeping	Required	Required	Required	Compliant (Prisma schema)
VAT calculation	20%	17%	25%	Compliant (configurable)
VAT reporting	Required	Required	Required	Planned
Financial reports	Required	Required	Required	Planned

Requirement	Serbia	BiH	Croatia	Implementation Status
Data retention (5 years)	Required	Required	Required	Planned
Electronic invoicing (B2G)	SEF	Optional	eRačun	Phase 2
Real-time fiscalization	Not required	Not required	Required	Phase 2
Digital signature	Not required	Not required	Required	Phase 2

Data Retention Lifecycle

```
stateDiagram-v2
    [*] --> Active : User registers

    Active --> DeletionRequested : POST /api/v1/account/delete
    Active --> Active : Normal usage\n(invoices, expenses, reports)

    DeletionRequested --> SoftDeleted : Anonymize PII\nemail → deleted-
{uuid}@example.com\nname → "Deleted User"\npasswordHash → ""

    SoftDeleted --> AuditAnonymized : Replace userId\nin LoggedAction\nwith "deleted-user"

    AuditAnonymized --> FinancialRetained : Financial records\nKEPT for 5 years\n(legal
obligation Serbia/BiH/HR)

    FinancialRetained --> PermanentDelete : After 5-year\nretention period

    PermanentDelete --> [*]

    note right of Active
        IP logs: 30 days
        Audit trail: 30 days
        Financial data: indefinite (legal)
    end note

    note right of FinancialRetained
```

```
        Invoices, expenses,\ntransactions, reports\nretained per Zakon o računovodstvu
        User PII already anonymized
    end note
```

Compliance Roadmap

Phase 1 (MVP) — GDPR Only

- ☐ Privacy policy drafted
- ☐ Terms of Service drafted
- ☐ Data minimization (by design)
- ☐ Encryption (TLS + AES-256)
- ☐ User data deletion workflow
- ☐ Data export (JSON)
- ☐ Sign DPAs with processors

Timeline: Pre-launch (before first customer)

Phase 2 (Serbia Launch)

- ☐ Serbian CoA template
- ☐ VAT reporting (20%)
- ☐ Financial reports (Balance Sheet, P&L, Cash Flow)
- ☐ SEF integration (B2G invoicing)
- ☐ Legal review by Serbian lawyer

Timeline: 3-6 months after MVP

Phase 3 (Regional Expansion)

- ☐ BiH VAT support (17%)
- ☐ Croatian VAT support (25%)
- ☐ Croatian fiscalization (real-time)
- ☐ eRačun integration (Croatia)
- ☐ Multi-language support (SR, BS, HR)

Timeline: 12-18 months after MVP

Compliance Checklist (Pre-Launch)

GDPR

- ☐ Privacy policy published
- ☐ Terms of Service published
- ☐ Cookie banner (if using cookies)
- ☐ User consent mechanism
- ☐ Data deletion workflow
- ☐ Data export endpoint
- ☐ DPAs signed (Railway, Vercel, Cloudflare, SendGrid)
- ☐ Railway EU region configured
- ☐ Breach notification process documented

Serbia (Phase 2)

- ☐ Legal review (Serbian accounting law)
- ☐ Serbian CoA template
- ☐ VAT calculation (20%)
- ☐ Financial reports (Serbian format)
- ☐ SEF integration (optional for MVP)

BiH (Phase 3)

- ☐ Legal review (BiH VAT law)
- ☐ VAT calculation (17%)
- ☐ PDV prijava report

Croatia (Phase 3)

- ☐ Legal review (Croatian fiscalization law)
- ☐ VAT calculation (25%)
- ☐ Fiscalization integration (mandatory)
- ☐ Qualified digital certificate
- ☐ eRačun integration

Risk Assessment

Risk	Likelihood	Impact	Mitigation
GDPR fine	Low (if compliant)	High (€20M)	Implement all GDPR requirements pre-launch
Data breach	Medium	High	Encryption, rate limiting, security audit
Serbian non-compliance	Medium	Medium	Hire local accountant as advisor
Croatian fiscalization failure	Low (Phase 3)	High	Partner with Croatian accounting firm
User data loss	Low	High	Daily backups, test restore process

Legal Disclaimer

IMPORTANT: This document is for internal planning only. It is NOT legal advice.

Before launch:

- Consult GDPR lawyer (EU compliance)
- Consult Serbian lawyer (accounting law)
- Consult BiH/Croatian lawyers (Phase 2/3)
- Review Privacy Policy with lawyer
- Review Terms of Service with lawyer

Recommended Lawyers:

- GDPR: Find lawyer specialized in EU data protection
- Serbia: Find lawyer specialized in računovodstvo (accounting law)

Related Documents

- Security Architecture: SECURITY-ARCHITECTURE.md
- Deployment Guide: ../infrastructure/DEPLOYMENT.md
- Privacy Policy: Privacy Policy *(not yet created)* (to be created)

- Terms of Service: Terms of Service *(not yet created)* (to be created)

Last Updated: 2026-02-20 **Status:** NOT COMPLIANT — Requires implementation and legal review **Next Review:** Before first paying customer **Compliance Officer:** TBD (hire accounting advisor in Phase 2)

Bilko CIAM abuse-gate fix — checkBefore moved outside SERIALIZABLE tx (MC #104069, root-cause of #103245)

Bilko CIAM abuse-gate fix — checkBefore moved outside SERIALIZABLE tx

MC #104069 | Root-cause of **MC #103245** | Fixed 2026-06-20

1. THE BUG (root cause)

MC #103245 [H1-PRE-PUBLIC-LAUNCH] CIAM abuse gate was marked done 2026-06-09 16:35, but the actual fix was committed 17:27 and **NEVER merged**. In `origin/main`, `CiamAbuseGate.checkBefore()` ran ONLY inside `UserProvisioningService.kt` (called from inside the SERIALIZABLE `transaction{}` block in `AuthService.createSessionFromEntraIdToken`, line 334).

Exposed's SERIALIZABLE transaction retry handler caught/swallowed `DisposableEmailException` and `TooManyRequestsException` → disposable-email accounts (e.g. guerrillamailblock.com) and over-rate-limit requests got provisioned with HTTP 200 despite the gate. **The disposable-email + rate-limit abuse gate was effectively defeated on the JIT/Entra provisioning path.**

2. THE FIX

Commit: `a862355a` → rebased `deb1621d`
Branch: `fix/abuse-gate-tx-swallow-103245`
PR: #3

Changes:

- **FIX1:** `AuthService.kt` (~line 329) — `CiamAbuseGate.checkBefore(email)` now called **BEFORE** the SERIALIZABLE `transaction{}` opens; exceptions propagate directly to StatusPages with no retry/swallow.
- **FIX2:** `routes/AuthRoutes.kt` (lines 379-388) — explicit re-throw catches for `DisposableEmailException` and `TooManyRequestsException` before the broad `catch(Exception)`.
- `StatusPages.kt` (111-129): `DisposableEmailException` → HTTP 422 (VAL_002); `TooManyRequestsException` → HTTP 429 (INFRA_002).
- **New test:** `CiamAbuseGateTransactionPathTest.kt` (+223 lines) — TX1/TX2/TX3 exercising the full `createSessionFromEntraIdToken` path through the SERIALIZABLE tx wrapper (the gap prior unit tests missed).

3. VERIFICATION

- **Manual branch build #44** (Azure DevOps Bilko-CI-CD): `CI_Gates 8/8 PASS` + `Build` + `Flyway` + `Deploy_Stage` SUCCEEDED on commit `deb1621d`.

  URL: https://dev.azure.com/alai-holding/Bilko/_build/results?buildId=44
- **Proveo integration tests** (real PostgreSQL/Testcontainers): `CiamAbuseGateTransactionPathTest 3/3 PASS`, `CiamAbuseGateTest 4/4 PASS`.
- **Live stage** confirmed serving the fix; gate sits behind Entra JWT signature verification (security boundary — a fully external HTTP 422 probe is not reachable without a tenant-signed Entra token, which is itself a positive security property).
- **Evidence bundle:** `/tmp/evidence-104069/` (proveo-abuse-probe/VERDICT.md, test XMLs, probe captures; build-43/44 logs).

4. PROCESS LESSON

A parent MC was closed before its fix was merged → the fix sat unmerged in a worktree for ~11 days.

Lesson: Do not mark a security MC done until the fix is verified merged on main. Link this to the broader "no fake DONE" rule.

References

- MC #104069 (parent security fix task)
- MC #103245 (original abuse gate task)
- Evidence bundle: `/tmp/evidence-104069/`
- PR #3: `fix/abuse-gate-tx-swallow-103245`