Security Testing Policy

Security Testing Policy

> **Organization:** Bilko — Balkan Accounting SaaS **Policy Number:** POL-SEC-TEST-001 **Version:** 1.0 **Date:** 2026-02-23 **Author:** CTO / Security Engineer **Status:** Draft **Reviewers:** Engineering Lead, DPO **Classification:** Confidential

Document History

Version	Date	Author	Changes
0.1	2026-02-23	CTO	Initial security testing policy for Bilko

---

1. Purpose & Scope

This policy defines the security testing requirements, tools, schedule, and acceptance criteria for the Bilko platform. Bilko handles regulated financial data (tax IDs, IBAN, accounting records) across three jurisdictions. Security testing is mandatory, not optional.

Scope: All Bilko applications — Express API (`apps/api/`), Next.js frontend (`apps/web/`), database layer (Prisma + PostgreSQL), and external integrations (SEF, HR-FISK).

---

2. Security Testing Pyramid

```
graph TD
    subgraph AUTOMATED["Automated (runs every CI pipeline)"]
        SAST["SAST\nESLint Security Rules\nTypeScript strict mode\nnpm audit\nSnyk SCA"]
        UNIT["Security Unit Tests\nVitest\nRBAC matrix tests\nOrg isolation tests\nEncryption tests\nVAT accuracy tests"]
```

```
        INT["Integration Tests\nVitest + Supertest\nAuth flow tests\nJWT validation\nRate limiting tests"]
    end

    subgraph PERIODIC["Periodic (scheduled)"]
        DAST["DAST\nOWASP ZAP\nMonthly + pre-release"]
        E2E["Security E2E\nPlaywright\nCross-tenant boundary tests\nPrivilege escalation tests"]
    end

    subgraph MANUAL["Manual (scheduled)"]
        PENTEST["Penetration Test\nExternal vendor\nAnnual"]
        REVIEW["Security Code Review\nPre-merge (security-sensitive PRs)\nArchitecture review quarterly"]
    end

    UNIT --> INT --> DAST --> PENTEST
```

---

3. Automated Security Testing (CI/CD)

Every push to `main` and every pull request triggers:

3.1 Static Analysis (SAST)

Tool	What It Checks	Failure Threshold
ESLint + `eslint-plugin-security`	Common JS security patterns (eval, RegExp DoS, object injection)	Any `error` level finding blocks merge
TypeScript strict mode	Type safety prevents implicit `any` that could bypass validation	Build failure blocks merge
`npm audit --audit-level=high`	Known vulnerabilities in dependencies	HIGH or CRITICAL CVEs block merge
Snyk (optional Phase 2)	Deeper SCA including license compliance	CRITICAL blocks merge

3.2 Security Unit Tests (Vitest)

Location: `apps/api/src/__tests__/security/`

Required test suites:

RBAC Matrix Tests

```
// Every permission combination must be explicitly tested
describe("RBAC — Invoice access", () => {
  const roles = ["owner", "admin", "accountant", "viewer"];

  test.each([
    ["owner", "create", true],
    ["admin", "create", true],
    ["accountant", "create", true],
    ["viewer", "create", false],
    ["owner", "delete", true],
    ["admin", "delete", true],
    ["accountant", "delete", false],
    ["viewer", "delete", false],
  ])("role=%s action=%s expected=%s", async (role, action, expected) => {
    const token = signTestJWT({ role, org: "org-1" });
    const res = await request(app)
      .post(`/api/invoices`)
      .set("Authorization", `Bearer ${token}`);
    // check response matches expected
  });
});
```

Organization Isolation Tests (Multi-Tenant Critical)

```
describe("Org isolation — no cross-tenant data leak", () => {
  let org1Token: string;
  let org2InvoiceId: string;

  beforeAll(async () => {
    // Setup two orgs with data
    org1Token = signTestJWT({ org: "org-1", role: "owner" });
    const org2Token = signTestJWT({ org: "org-2", role: "owner" });

    // Create invoice in org-2
    const res = await request(app)
      .post("/api/invoices")
      .set("Authorization", `Bearer ${org2Token}`)
      .send(validInvoicePayload);
```

```
    org2InvoiceId = res.body.id;
  });

  test("org-1 cannot read org-2 invoice", async () => {
    const res = await request(app)
      .get(`/api/invoices/${org2InvoiceId}`)
      .set("Authorization", `Bearer ${org1Token}`);
    expect(res.status).toBe(404); // NOT 403 — don't reveal existence
  });

  test("org-1 list does not include org-2 data", async () => {
    const res = await request(app)
      .get("/api/invoices")
      .set("Authorization", `Bearer ${org1Token}`);
    const ids = res.body.data.map((i: any) => i.id);
    expect(ids).not.toContain(org2InvoiceId);
  });
});
```

Field Encryption Tests

```
describe("Field encryption — L4 Restricted", () => {
  test("PIB stored encrypted in DB", async () => {
    const testPIB = "123456789"; // fake PIB
    // Create contact with PIB
    await request(app)
      .post("/api/contacts")
      .set("Authorization", `Bearer ${ownerToken}`)
      .send({ name: "Test", taxId: testPIB, type: "RS" });

    // Read raw DB value — should not be plaintext
    const raw = await prisma.$queryRaw`
      SELECT "taxId" FROM "Contact" WHERE name = 'Test'
    `;
    expect(raw[0].taxId).not.toBe(testPIB);
    expect(raw[0].taxId).toMatch(/^[A-Za-z0-9+/]+=*:[A-Za-z0-9+/]+=*:[A-Za-z0-9+/]+=*$/);
    // Should be base64:base64:base64 format (iv:authTag:ciphertext)
  });

  test("decrypted PIB matches original on read", async () => {
```

```
    const res = await request(app)
      .get("/api/contacts")
      .set("Authorization", `Bearer ${ownerToken}`);
    const contact = res.body.data.find((c: any) => c.name === "Test");
    expect(contact.taxId).toBe("123456789");
  });
});
```

VAT Accuracy Tests (Financial Compliance)

```
describe("VAT calculation accuracy", () => {
  test("RS: VAT 20% on standard goods (NUMERIC precision)", () => {
    const net = new Decimal("100.00");
    const vatAmount = net.mul("0.20");
    const gross = net.plus(vatAmount);
    expect(vatAmount.toString()).toBe("20.00");
    expect(gross.toString()).toBe("120.00");
  });

  test("HR: VAT 25% (EUR since Jan 2024)", () => {
    const net = new Decimal("100.00");
    const gross = net.mul("1.25");
    expect(gross.toString()).toBe("125.00");
  });

  test("BA: VAT 17% (UIO standard)", () => {
    const net = new Decimal("100.00");
    const gross = net.mul("1.17");
    expect(gross.toString()).toBe("117.00");
  });

  test("No float drift on invoice totals", () => {
    // Known JS float bug: 0.1 + 0.2 !== 0.3
    const line1 = new Decimal("0.10");
    const line2 = new Decimal("0.20");
    expect(line1.plus(line2).toString()).toBe("0.30");
    // Contrast: expect(0.1 + 0.2).toBe(0.3) would FAIL
  });
});
```

Authentication Tests

```
describe("Auth — JWT security", () => {
  test("expired access token returns 401", async () => {
    const expiredToken = signTestJWT({ exp: Math.floor(Date.now()/1000) - 1 });
    const res = await request(app)
      .get("/api/invoices")
      .set("Authorization", `Bearer ${expiredToken}`);
    expect(res.status).toBe(401);
  });

  test("tampered token returns 401", async () => {
    const validToken = signTestJWT({ role: "viewer" });
    // Tamper: change role claim in payload
    const parts = validToken.split(".");
    const payload = JSON.parse(Buffer.from(parts[1], "base64url").toString());
    payload.role = "owner"; // attempt privilege escalation
    parts[1] = Buffer.from(JSON.stringify(payload)).toString("base64url");
    const tamperedToken = parts.join(".");
    const res = await request(app)
      .delete("/api/invoices/any-id")
      .set("Authorization", `Bearer ${tamperedToken}`);
    expect(res.status).toBe(401);
  });

  test("rate limiting: 6th auth attempt in 15min returns 429", async () => {
    for (let i = 0; i < 5; i++) {
      await request(app).post("/api/auth/login").send({ email: "x", password: "wrong" });
    }
    const res = await request(app).post("/api/auth/login").send({ email: "x", password:
"wrong" });
    expect(res.status).toBe(429);
  });
});
```

3.3 Dependency Scanning

```
# .github/workflows/security.yml
- name: Audit dependencies
  run: npm audit --audit-level=high
  # HIGH or CRITICAL CVEs fail the build
```

```
- name: Check for secrets in code
  uses: trufflesecurity/trufflehog@main
  # Scans for committed credentials, API keys
```

Dependency update policy:

- CRITICAL CVE: patch within 24 hours
- HIGH CVE: patch within 7 days
- MEDIUM CVE: patch within 30 days
- LOW CVE: patch at next sprint boundary

---

4. Dynamic Application Security Testing (DAST)

OWASP ZAP

Schedule: Monthly + before every major release

Scope (in-scope for ZAP):

- `https://staging.bilko.io` (staging environment only — NEVER production)
- All API endpoints under `/api/`
- Authentication flows
- File upload endpoints (if any)

Out of scope:

- SEF portal, FINA portal (external systems)
- Railway infrastructure
- Cloudflare WAF (managed by Cloudflare)

ZAP Configuration:

```
# zap-baseline.yaml
env:
  contexts:
    - name: Bilko API
      urls:
        - https://staging.bilko.io/api/
```

```
    authentication:
      method: script
      # ZAP script to authenticate and get JWT
  rules:
    - id: 10202  # Absence of Anti-CSRF tokens — note (cookies are httpOnly)
      threshold: LOW
    - id: 10096  # Timestamp Disclosure — ignore (timestamps are public)
      threshold: OFF
```

Required ZAP findings threshold (before release):

- CRITICAL / HIGH: 0 allowed
- MEDIUM: must be assessed — known acceptable risks documented
- LOW / INFORMATIONAL: document and prioritize

---

5. Penetration Testing

Frequency: Annual (or after significant architecture change) **Provider:** External certified pentest firm (OSCP/CREST certified)

Scope:

- Web application (app.bilko.io)
- API endpoints
- Authentication & session management
- Multi-tenant isolation (primary focus — org isolation must be tested)
- Business logic flaws (VAT calculation, invoice numbering)
- Third-party integrations (SEF API, HR-FISK)

Rules of engagement:

- Staging environment only — no production testing without explicit CEO approval
- No DoS / DDoS testing
- No social engineering of employees
- Penetration test agreement signed before engagement begins

Remediation SLAs (post-pentest findings):

Severity	Fix Deadline
CRITICAL	48 hours
HIGH	7 days

Severity	Fix Deadline
MEDIUM	30 days
LOW	Next quarter

---

6. Security Code Review

When required (mandatory pre-merge):

- Changes to authentication or authorization code
- Changes to encryption utilities (`encryptField`, `decryptField`)
- Changes to Prisma query patterns (potential org isolation bypass)
- New external API integrations (SEF, FINA, etc.)
- Changes to RBAC middleware or permission matrices

Who reviews: CTO or designated Senior Engineer with security background.

Checklist for security-sensitive PRs:

- ☐ No secrets or credentials in code or config
- ☐ All new Prisma queries include `organizationId` in WHERE clause
- ☐ New endpoints have RBAC decorator applied
- ☐ New user inputs validated with Zod schema
- ☐ L4 Restricted fields encrypted before write, decrypted after read
- ☐ LoggedAction entry created for all write operations
- ☐ Rate limiting applied to new auth-adjacent endpoints

---

7. CI/CD Security Gates

```
flowchart LR
    PR["Pull Request"] --> LINT["ESLint Security\nRules"]
    LINT -->|"PASS"| AUDIT["npm audit\n--audit-level=high"]
    AUDIT -->|"PASS"| TEST["Vitest\nSecurity Test Suite"]
    TEST -->|"PASS"| SECRETS["TruffleHog\nSecret Scan"]
    SECRETS -->|"PASS"| MERGE["Merge Allowed"]
    LINT -->|"FAIL"| BLOCK["PR Blocked"]
    AUDIT -->|"FAIL"| BLOCK
```

```
    TEST -->|"FAIL"| BLOCK
    SECRETS -->|"FAIL"| BLOCK
```

Non-negotiable gates (cannot be bypassed with `--no-verify` or `--force`):

1. ESLint security rules: zero `error` findings
2. npm audit: zero HIGH/CRITICAL CVEs
3. Vitest security tests: 100% pass — especially org isolation and RBAC tests
4. TypeScript strict: zero type errors

---

8. Vulnerability Disclosure

Process:

1. Security researchers may report vulnerabilities to security@bilko.io
2. Acknowledgment within 24 hours
3. Investigation and severity assessment within 5 business days
4. Remediation per SLA in Section 5
5. Responsible disclosure: researcher notified when fix is deployed

---

Approval

Role	Name	Signature	Date
Author	CTO / Security Engineer		2026-02-23
Reviewer (Engineering Lead)			
Reviewer (DPO)			
Approver	CEO		

Revision #3
Created 2026-02-24 22:50:54 UTC by John
Updated 2026-05-31 20:04:00 UTC by John