

# User Management Security Audit Report

---

**Date:** January 15, 2026 **Auditor:** Automated Security Review **Scope:** Complete  
User Management System (Database, API, Frontend)

---

# Executive Summary

---

A comprehensive security audit of the User Management system was conducted, covering database models, API endpoints, authentication/authorization flows, and frontend components.

**Key Finding:** A critical **authorization bypass vulnerability** was discovered and fixed. The caching middleware was serving admin-only responses to non-admin users.

**Test Results:** 12/12 tests passing after fixes.

---

# Critical Issue Fixed

---

## CVE-Like: Authorization Bypass Through Response Caching

**Severity:** HIGH **OWASP Category:** A01:2021 - Broken Access Control

**Description:** The caching middleware in `/src/backend/src/index.ts` created cache keys based solely on the request path (`api:cache:${req.path}`). When an admin fetched `/api/users`, the response was cached. Subsequent requests from ANY authenticated user received the cached admin response, completely bypassing the controller's role-based authorization check.

**Impact:** - Non-admin users could view all user accounts including admin accounts  
- Exposed user emails, usernames, roles, and login history - Could be used for account enumeration and targeted attacks

### Fix Applied:

```
// Before: /api/users was cached
const skipRoutes = ['/api/auth', '/api/health', '/api/users/me'];

// After: All user management endpoints excluded from cache
const skipRoutes = [
  '/api/auth',
  '/api/health',
  '/api/users' // All user management endpoints - role-based access control
];
```

**File Modified:** `src/backend/src/index.ts:529-533`

---

# Security Strengths Found

---

## 1. Password Security (OWASP Compliant)

- **Bcrypt hashing** with 12 rounds (configurable)
- **Password validation** with:
  - Minimum 12 characters (production), 8 (development)
  - Character diversity requirements (3+ character classes)
  - Common password blocklist
  - Pattern detection (sequential, repeated, keyboard patterns)
  - Personal info detection (username/email in password)

## 2. Authentication Security

- **Timing attack prevention:** Constant 100ms delay on all login attempts
- **User enumeration prevention:** Generic "Invalid email or password" message
- **Dummy hash computation:** Bcrypt compare runs even for non-existent users
- **Account lockout:** 10 failed attempts = 15-minute lockout

## 3. Token Security

- **JWT tokens** with configurable expiration
- **Refresh tokens** for session extension
- **Proper token validation** middleware

## 4. Rate Limiting

- **Auth endpoint:** 10 requests/15 minutes per IP+email combination
- **API endpoints:** 20 requests/minute
- **Upload endpoints:** 10 uploads/5 minutes
- **Script operations:** Rate limited

## 5. Input Security

- **Request sanitization** middleware

- **SQL injection protection** via Sequelize ORM
  - **XSS protection** via input sanitization
  - **Request size limits** (50MB)
-

# Remaining Security Recommendations

---

## High Priority

1. **Token Storage (XSS Vulnerability)**
2. **Current:** Tokens stored in localStorage
3. **Risk:** XSS attacks can steal tokens
4. **Recommendation:** Use httpOnly cookies for token storage
5. **CSRF Protection**
6. **Current:** No CSRF tokens implemented
7. **Risk:** Cross-site request forgery attacks possible
8. **Recommendation:** Implement CSRF tokens for state-changing operations
9. **Token Refresh Logic (Frontend)**
10. **Current:** No automatic token refresh in frontend
11. **Risk:** Users get logged out unexpectedly
12. **Recommendation:** Implement automatic token refresh before expiration

## Medium Priority

1. **Email Verification**
2. **Current:** No email verification on registration
3. **Risk:** Fake accounts, spam
4. **Recommendation:** Add email verification flow
5. **Password Reset Flow**
6. **Current:** Admin-only password reset
7. **Risk:** Users cannot self-recover accounts
8. **Recommendation:** Implement self-service password reset with email
9. **Token Revocation**

10. **Current:** No token blacklist on logout
11. **Risk:** Tokens remain valid after logout
12. **Recommendation:** Implement token blacklist or use short-lived tokens

## Low Priority

1. **Multi-Factor Authentication (MFA)**
  2. **Current:** Single-factor (password only)
  3. **Recommendation:** Add optional TOTP/SMS 2FA
  4. **Session Management**
  5. **Current:** Basic JWT sessions
  6. **Recommendation:** Add "active sessions" view and remote logout
-

# Test Coverage Summary

## Registration Tests

Test	Status
Register new user	✓ PASS
Reject duplicate email	✓ PASS
Reject duplicate username	✓ PASS
Reject weak password	✓ PASS
Reject invalid email format	✓ PASS

## Authentication Tests




Test	Status
Login with valid credentials	✓ PASS
Reject wrong password	✓ PASS
Generic error for non-existent user	✓ PASS

## Token Tests



Test	Status
Get user info with valid token	✓ PASS
Reject missing token	✓ PASS
Reject invalid token	✓ PASS
Refresh token	✓ PASS



## Authorization Tests

Test	Status
Non-admin blocked from user list	 PASS
Admin can access user list	 PASS
Admin CRUD operations	 PASS

## Security Tests

Test	Status
Reject SQL injection	 PASS
Handle XSS in input	 PASS

---

## Files Reviewed

---

- `src/backend/src/models/User.ts` - User model with password hashing
  - `src/backend/src/routes/auth.ts` - Authentication endpoints
  - `src/backend/src/routes/users.ts` - User management endpoints
  - `src/backend/src/controllers/UserController.ts` - User CRUD logic
  - `src/backend/src/middleware/authMiddleware.ts` - JWT authentication
  - `src/backend/src/middleware/security.ts` - Security middleware
  - `src/backend/src/utils/passwordValidation.ts` - Password strength validation
  - `src/backend/src/index.ts` - Main app configuration
  - `src/frontend/src/utils/errorUtils.ts` - Error handling utilities
  - `src/frontend/src/pages/Settings/UserManagement.tsx` - Admin UI
-

# Conclusion

---

The user management system demonstrates strong security practices in authentication and password handling. The critical authorization bypass vulnerability was identified and fixed. The system now passes all security tests.

**Recommended Next Steps:** 1. Migrate token storage from localStorage to httpOnly cookies 2. Implement CSRF protection 3. Add frontend token refresh logic 4. Consider email verification for new registrations

---

*Report generated by comprehensive security testing suite.*

Generated 2026-01-16 21:23 UTC