Secure Coding Review – Recommendations and Remediation

Recommendations and Best Practices for Secure Coding

1. Use Parameterized Queries for Database Access

Always use parameterized queries to prevent SQL Injection attacks. This ensures user input is treated strictly as data and not executable SQL.

2. Never Store or Compare Plaintext Passwords

Passwords must be securely hashed using strong algorithms such as bcrypt, Argon2, or PBKDF2 to protect user credentials in case of data breaches.

3. Validate and Sanitize User Input

All user input should be validated on the server side to prevent injection attacks and ensure application stability.

4. Use Static Analysis Tools Regularly

Tools such as Bandit should be integrated into the development lifecycle to detect vulnerabilities early.

5. Handle Errors Securely

Avoid exposing detailed error messages to users. Instead, log errors internally and show generic messages to users.

6. Avoid Hardcoded Secrets

Secrets such as database credentials should never be hardcoded. Use environment variables or secure configuration files.

7. Apply the Principle of Least Privilege

Applications should run with the minimum permissions required to reduce potential damage from security breaches.

Documented Findings and Remediation Steps

Finding 1: **SQL Injection Vulnerability**

Location: app.py, **line 7**

Tool Used: **Bandit (B608)**

Risk Level: **Medium**

CWE**: CWE-89**

Issue Description:

User input was directly concatenated into an SQL query, allowing attackers to manipulate database queries.

Remediation:

String-based SQL queries were replaced with parameterized queries and verified using Bandit.

Finding 2: Insecure Password Handling

Risk Level: **High**

Issue Description:

Passwords were handled and compared in plaintext.

Remediation:

Secure password hashing using **bcrypt** was implemented, eliminating plaintext password comparison.

Verification of Remediation

After applying all remediation steps, the secured application (fixedapp.py) was re-scanned using Bandit.

Result: No issues identified.

Conclusion

The secure coding review successfully identified and remediated critical vulnerabilities. Applying secure coding best practices significantly improves application security and resilience.