DIVA Android App

Security Assessment Report

Date: 2024-11-30

DIVA Android Security Assessment Report

Executive Summary

This report presents the findings of a security assessment conducted on the DIVA (Damn Insecure and Vulnerable App) Android application. DIVA is deliberately built to be vulnerable for learning and testing purposes. The assessment identified multiple high-risk vulnerabilities that could compromise user data and system security.

1. Insecure Logging

Severity: High
Risk: The application logs sensitive user credentials in plain text to the Android system log.

Technical Details:
- Location: Insecure Logging Activity
- Issue: User credentials (username and password) are logged using Android's Log.d()
- Impact: Any application with READ_LOGS permission can access user credentials
- Proof of Concept:
  1. Enter credentials in the login form
  2. Check logcat output using 'adb logcat'
  3. Credentials appear in plain text

Recommendation:
1. Remove all debug logging in production builds
2. Never log sensitive information
3. Implement proper logging levels (ERROR, WARN, INFO, DEBUG)

2. Hardcoding Issues

Severity: High
Risk: Critical secrets and API keys are hardcoded in the application code.

Technical Details:
- Location: Hardcoding Issues Activity
- Issue: API keys and credentials stored directly in source code
- Impact: Reverse engineering can easily expose sensitive credentials
- Proof of Concept:
  1. Decompile APK using tools like jadx
  2. Search for hardcoded strings
  3. Extract sensitive information

Recommendation:
1. Use Android Keystore System for storing sensitive keys
2. Implement proper key management systems
3. Use encryption for storing sensitive data
4. Consider using remote configuration for API endpoints

3. Insecure Data Storage

Severity: High
Risk: Sensitive user data stored without encryption in shared preferences and SQLite database.

Technical Details:
- Location: Insecure Data Storage Activities
- Issues Found:
  a) Plain text storage in SharedPreferences
  b) Unencrypted SQLite database
  c) External storage without proper permissions
- Impact: Local attacks can extract sensitive user data
- Proof of Concept:
  1. Access app's data directory using adb
  2. Extract shared_prefs and databases
  3. Read data without requiring decryption

Recommendation:
1. Use EncryptedSharedPreferences
2. Implement SQLCipher for database encryption
3. Use Android Keystore for key management
4. Avoid storing sensitive data in external storage

4. Access Control Issues

Severity: High
Risk: Insufficient access controls allow unauthorized access to protected functionality.

Technical Details:
- Location: Access Control Issue Activities
- Issues Found:
  a) Missing permission checks
  b) Implicit intents exposing sensitive activities
  c) Weak activity access controls
- Impact: Unauthorized users can access restricted features
- Proof of Concept:
  1. Launch activities directly using adb
  2. Bypass authentication checks
  3. Access protected features without proper authorization

Recommendation:
1. Implement proper permission checks
2. Use explicit intents for internal components
3. Add runtime permission validation
4. Implement proper authentication checks

5. SQL Injection

Severity: Critical
Risk: SQL injection vulnerabilities allow unauthorized database access and manipulation.

Technical Details:
- Location: SQL Injection Activity
- Issue: Raw SQL queries with unvalidated user input
- Impact: Attackers can:
  a) Extract unauthorized data
  b) Modify database contents
  c) Execute arbitrary SQL commands
- Proof of Concept:
  1. Input: ' OR '1'='1
  2. Input: '; DROP TABLE users;--
  3. Successfully bypass authentication

Recommendation:
1. Use parameterized queries
2. Implement input validation
3. Use ORM frameworks
4. Limit database user privileges

Overall Recommendations

1. Implement Secure Coding Practices:
- Follow OWASP Mobile Security Testing Guide
- Regular security training for developers
- Code review processes focusing on security

2. Enhance Data Protection:
- Implement encryption for all sensitive data
- Secure key management using Android Keystore
- Regular security assessments

3. Improve Access Controls:
- Proper authentication mechanisms
- Role-based access control
- Input validation and sanitization

4. Security Testing:
- Regular penetration testing
- Automated security scanning
- Vulnerability assessments

5. Monitoring and Logging:
- Implement secure logging practices
- Monitor for security incidents
- Regular security audits