SIL771: SPECIAL TOPIC IN CYBER SECURITY

Assignment 2: Android App Security Testing – InsecureBankv2

We have learnt Android app security, the Android Security Model and performed security testing of a few Android apps. To reinforce the learning, you have to perform security testing of InsecureBankv2 mobile app as assignment 2. **This assignment carries 100 marks and is due on 29 Apr 25.**

Project repository (source & APK): https://github.com/dineshshetty/Android-InsecureBankv2

This assignment uses the deliberately vulnerable mobile-banking application **InsecureBankv2**. Your objective is to discover, exploit, and document the listed vulnerabilities. Each confirmed vulnerability is worth **5 marks**. **Solve any 20 questions.** Submit a report with evidence (screenshots, PoC code, and mitigation advice). The backend server runs in Python 2; ensure it is active on your laptop before testing the APK in an emulator or rooted device. Download the apk from the github link given above.

Default test credentials:

• dinesh / Dinesh@123$
• jack / Jack@123$

NOTE: Confirm emulator ↔ host connectivity before you begin.

Question	Vulnerability & Task	Marks
1	Flawed Broadcast Receivers Identify unprotected broadcast receivers and craft a malicious broadcast that triggers unintended behaviour.	5
2	Intent Sniffing and Injection Capture inter-component intents and inject crafted payloads to access or modify protected resources.	5
3	Weak Authorization Mechanism Bypass or escalate user roles by tampering with tokens, session IDs, or role parameters.	5
4	Local Encryption Issues Locate locally stored encrypted data and demonstrate how weak keys or algorithms allow plaintext recovery.	5
5	Vulnerable Activity Components Launch exported or improperly protected activities to access restricted app functionality.	5
6	Root Detection and Bypass Analyse detection logic and use Magisk / Frida to evade it while keeping root privileges.	5

7	Emulator Detection and Bypass Circumvent anti-emulator checks to run the app in an emulator for dynamic analysis.	5
8	Insecure Content Provider Access Query content providers directly to read or modify sensitive records without proper permissions.	5
9	Insecure WebView Implementation Exploit JavaScript-interface exposure or mixed-content loading to run arbitrary JS in the app context.	5
10	Weak Cryptography Implementation Find cryptographic misuse (e.g., ECB mode, static IV) and decrypt/forge sensitive data.	5
11	Application Patching Generate a patched APK that removes client-side controls and demonstrates impact (e.g., disable SSL pinning).	5
12	Sensitive Information in Memory Use heap dumps or Frida to locate credentials/tokens left in memory after use.	5
13	Insecure Logging Mechanism Search logcat/system logs for leakage of PII or secrets and prove exploitability.	5
14	Android Pasteboard Vulnerability Show how clipboard data can be intercepted or poisoned to steal sensitive values.	5
15	Application Debuggable Flag Enabled Leverage debuggable build to attach a debugger and extract runtime secrets.	5
16	Android Keyboard Cache Issues Demonstrate retrieval of typed secrets from keyboard caches or predictive-text databases.	5
17	Android Backup Vulnerability Back up the app with adb and extract private files that should remain protected.	5
18	Runtime Manipulation (Dynamic Instrumentation) Use Frida/Objection to modify functions at runtime (e.g., force login success).	5
19	Insecure SDCard Storage Locate plaintext sensitive files on external storage and exfiltrate them without root.	5
20	Insecure HTTP Connections Intercept unencrypted traffic, modify server responses, and observe insecure behaviour.	5
21	Parameter Manipulation Tamper with request parameters (e.g., amount, account number) to alter server-side actions.	5
22	Hard-coded Secrets Reverse the APK to extract API keys, cert pins, or credentials and demonstrate misuse.	5

23	Username Enumeration Issue Show how login error messages or timing leaks reveal valid usernames.	5
24	Developer Backdoors Locate hidden features / test endpoints and exploit them for elevated access.	5
25	Weak Change-Password Implementation Exploit logic flaws (e.g., missing old-password check, weak policy) to change another user's password.	5

Total Marks: 100

Submission format:
1. PDF report with methodology & evidence for each task.
2. Patched or instrumented APKs (where applicable).
3. README describing environment and tools used.

Submit a report in pdf format having your roll no and name as filename (e.g. A2_2021JCS2290_AmitSingh). The submission is **due on 29 Apr 25 2359 Hrs**.