Mitt Arv Security Bug Bounty:

During comprehensive testing of the Mitt Arv Android app, I conducted both Usage Testing and Vulnerability Testing. Throughout these tests, several security glitches were identified that could adversely affect user experience and compromise application security. The following report documents the findings in detail, along with supporting evidence (refer to the associated video recordings available in my GitHub repository).

1. Testing Methodology:

a. **Usage Testing:**
The application was executed on the Android Studio emulator to identify any glitches experienced during regular user interactions.

b. **Vulnerability Testing:**
A series of tests were employed to uncover potential security vulnerabilities that could be exploited, ensuring comprehensive coverage.

2. Findings from Usage Testing:

2.1. Login Glitch:

- **Issue Description:**
When using the "Sign in with Google" option, upon selecting a Google account, the application automatically navigates to the "Enter Personal Details" page. Despite populating all fields with valid information, the interface consistently returns an error message stating, "Fill all the empty blanks," even when no blanks are left empty.
- **Impact:**
This glitch hampers a smooth user login experience, potentially discouraging proper sign-up and account creation.
- **Recommendation:**
Review and correct the validation logic on the "Enter Personal Details" page to ensure proper detection of filled fields.

2.2. Lockedin Password Reset Questions Error:

- **Issue Description:**
During the password reset procedure, users are prompted to answer personal security questions previously provided during setting of password. When a user enters the correct answers, the application erroneously indicates that the responses are incorrect. Consequently, the system enforces a 24-hour lockout period, preventing the password reset.
- **Impact:**
This represents a critical security flaw. A legitimate user is unable to reset their password due to incorrect validation, potentially leading to loss of access, user frustration, and increased support queries.
- **Recommendation:**
Investigate and rectify the backend logic responsible for verifying answers to security

questions. Ensure that the system correctly matches user input with stored answers, and consider additional logging to monitor for potential abuse or misconfiguration.

3. Findings from vulnerability testing:

I conducted a comprehensive vulnerability assessment of the Mitt Arv APK using MobSF, which identified several security issues requiring prompt remediation.

App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	warning	CWE: CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') OWASP Top 10: M7: Client Code Quality	com/onesignal/session/internal/outcomes/impl/m.java com/pichillilorenzo/flutter_inappwebview_android/credential_database/CredentialDatabaseHelper.java e9/c.java o5/m0.java o5/t0.java u3/c.java

The above is the vulnerability I got after scanning the APK file using MobSf Mobile Security Framework. This tells that the app uses SQLite Database and uses raw SQL queries which might lead to SQL injection and Data breach.

To avoid such incidents, we can use:

i. **Use Parameterized Queries or Prepared Statements:**
- Instead of concatenating user input directly into SQL strings, use parameterized queries that bind user input as parameters.
- For example, if you're using Android's SQLiteDatabase, use methods like query(), update(), or rawQuery() with selection arguments instead of forming queries with string concatenation.

ii. **Input Validation and Sanitization:**
- Validate and sanitize all user inputs before using them in any query.
- Ensure that the input conforms to the expected format (e.g., numbers, emails) to reduce the risk of injecting malicious SQL.

iii. **Encrypt Sensitive Data:**
- Encrypt sensitive information, such as passwords or private data, before saving it into the database.
- Use strong encryption libraries and algorithms (e.g., AES) to secure the data, ensuring that even if the database is compromised, the sensitive information remains protected.

iv. **Use ORMs or Database Libraries (if applicable):**
- Consider using an Object-Relational Mapping (ORM) library that handles parameterization and encryption for you.
- These libraries often provide additional layers of security and reduce the chance of introducing SQL injection bugs.

v. **Least Privilege and Access Control**:
- Restrict the database privileges to only the necessary operations for the application.
- Limit direct access to the database to reduce the attack surface.

4. Conclusion:

In summary, the comprehensive testing of the Mitt Arv Android app has uncovered several critical issues that, if unaddressed, could compromise both user experience and overall application security. The Usage Testing revealed significant glitches in key user flows, such as the login process and password reset mechanism, each of which could deter proper account setup and secure access. The Vulnerability Testing, conducted via the MobSF framework, further exposed the risk posed by the use of raw SQL queries and unencrypted sensitive data, highlighting potential threats from SQL injection and data breaches.

It is imperative that these issues be addressed promptly. By adopting secure coding practices—such as using parameterized queries or prepared statements, enforcing strict input validation, implementing robust encryption, and applying the principle of least privilege—the development team can substantially reduce the risk of exploitation and enhance both data integrity and security. Additionally, continued use of security assessment tools and regular code reviews will be essential to maintaining the resilience of the app.

Implementing the recommended security measures will not only improve the immediate functionality and reliability of the Mitt Arv app but also establish a stronger foundation for long-term user trust and safety.