FIGI FINANCIAL INCLUSION GLOBAL INITIATIVE

FIGI Symposium
18 May - 24 June, 2021

Security audit of various DFS applications

Results

Philippe Oechslin, Objectif Securite

figi.itu.int
#financialinclusion

Organized by

ITU · THE WORLD BANK IBRD • IDA · Committee on Payments and Market Infrastructures · BIS

3. Results

- All three applications have been tested with the same method
- Only failed tests are described below

3.1 App1

Payment app, backed by a bank account, credit card or prepaid

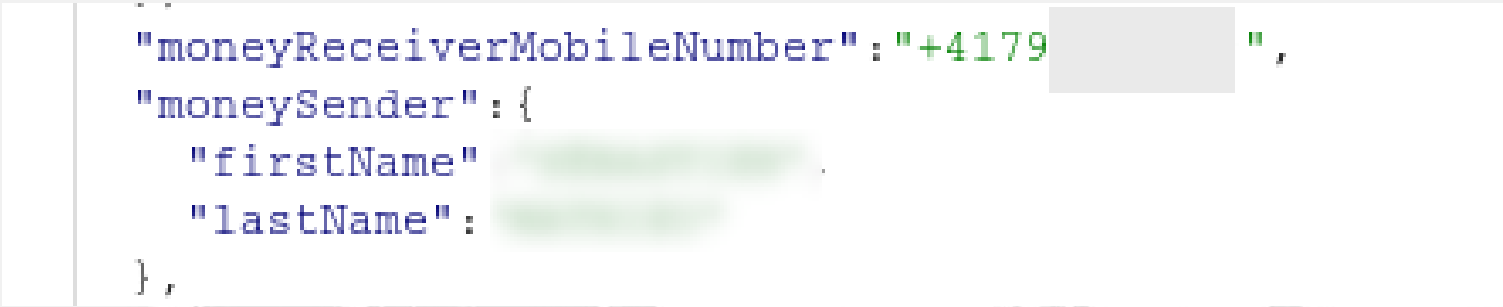
Insecure Data Storage:

- ✗ T2.1 The application requires the "android.permission.WRITE_EXTERNAL_STORAGE" permission.
  - Note that this does not imply that the app actually writes data on external storage and, if it did, that this data is sensible.

App1

Insufficient cryptography:

- T5.1 The application uses the weak MD5 and SHA-1 hashing algorithms as well as the weak ECB mode of encryption.
- Interception of data shows names, phone numbers and amounts transmitted in clear inside HTTPS connection



App1

Code Tampering:

- ✗ T8.1 App1 runs on rooted devices
  - Most financial apps refuse to run on rooted phones

3.2 App2

App2 is provided by a mobile network operator that provides digital financial services in areas in which they operate across Africa

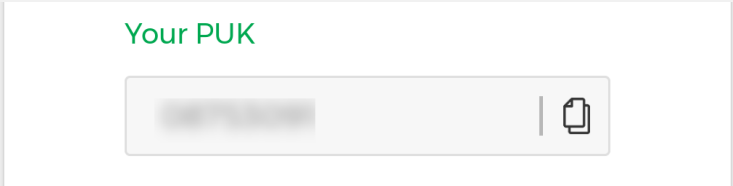
Insecure communication:

- ✗ T3.4 Android:usesClear textTraffic is set to true in the manifest
  - We observed cleartext traffic sent when starting the app
  - It did not include sensitive information

App2

Insecure Authentication:

- T4.1 The application does not require a PIN or fingerprint every time it is started. A PUK can be accessed



Insufficient Cryptography:

- T5.1 The application uses the weak SHA-1 hashing algorithm as well as a weak random number generator

3.3 App3

App3 is also provided a mobile operator and in several countries across Africa and Asia. The app makes it possible for users to send money to contacts, pay for goods and services

Insecure data storage:

- ✗ T2.2 While the app is running, screenshot is not disabled.

Insecure authentication:

- ✗ T3.3 The app accepts to establish an HTTPS connection to a proxy with a trusted certificate
- ✗ T3.4 Android:usesClear textTraffic is set to true in the manifest

App3

Insecure Authentication:

- ✗ T4.1 The application does not require a PIN or fingerprint every time it is started
  - One can see the balance of the account

Insufficient Cryptography:

- ✗ T5.1 The application uses the weak MD5 and SHA-1 hashing algorithms as well as a weak random number generator.

DFS Assurance Framework

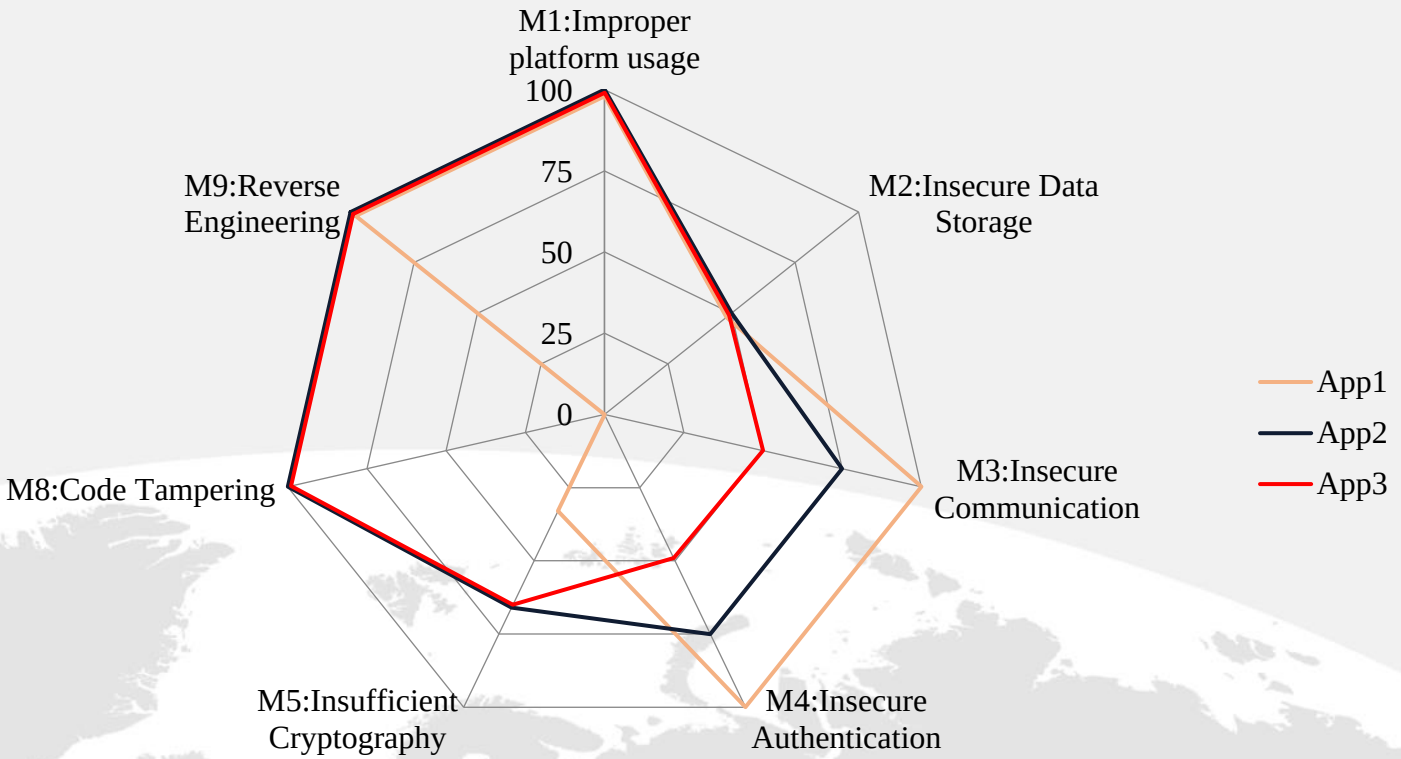
TEMPLATE FOR APPLICATION SECURITY BEST PRACTICES	Corresponding tests
9.1 Device integrity	T1.2 Android:debuggable T1.4 Dangerous permissions T8.1 The application should refuse to run on a rooted device
9.2 Communication Security and Certificate Handling	T3.1 Application should only use HTTPS connections T3.2 Application should detect Machine-in-the-Middle attacks with untrusted certificates T3.3 Application should detect Machine-in-the-Middle attacks with trusted certificates T3.4 App manifest should not allow clear text traffic T5.1 The app should not use unsafe crypto primitives T5.2 The HTTPS connections should be configured according to best practices T5.3 The app should encrypt sensitive data that is sent over HTTPS
9.3 User authentication	T4.1 Authentication required before accessing sensitive information T4.2 The application should have an inactivity timeout T4.3 If a fingerprint is added, authentication with fingerprints should be disabled T4.4 It should not be possible to replay intercepted requests

DFS Assurance Framework

TEMPLATE FOR APPLICATION SECURITY BEST PRACTICES	Corresponding tests
9.4 Secure Data Handling	T1.1 Android:allowBackup T1.3 Android:installLocation T2.1 Android.permission.WRITE_EXTERNAL_STORAGE T2.2 Disabling screenshots
9.5 Secure Application Development	T9.1 The code of the app should be obfuscated

Summary of results

- No critical vulnerabilities were detected but
  - App1 has no application-level encryption
  - App2 displays PUK without requiring PIN



4 Conclusions

- The tests allow an independent evaluation of the security of DFS apps
- Impact of failed tests is difficult to estimate, as the logic of the apps was not analyzed
  - This would require reverse engineering the obfuscated code
  - It may also require interacting with the server and authorization by the owner of the app
- Still, all tests correspond to best practices that should be followed by DFS