SAP
INTEGRATION AND CERTIFICATION CENTER

Mobile Solution– Security Guidelines

.

Version: 2.2
April, 2013

Contents

1 Introduction

Security of enterprise mobile solutions is different from mobile security as enterprise mobile solutions involve Enterprise Information Systems (aka backend system), SAP Mobile Platform and client applications (mobile apps). Due to this security of a mobile solution is dependent on multiple factors which are summarized below.

- Mobile solution security – An enterprise could run multiple mobile solutions on the same infrastructure and each solution must take care of its own security aspects while leveraging the security features of the overall setup.

- Enterprise Infrastructure – Overall infrastructure and security setup of the enterprise in which mobile solution needs to run.

- Device Security – MDM Infrastructure and policies used by the organization to protect enterprise assets on the mobile devices.

- Device Policies – Policy defined by the organization for mobile devices. This would include BYOD, OS upgrade and scrutiny for supported devices.

This document also outlines the generic security vulnerabilities in On-Device apps and how it can be handled in various environments such as the device platform, device and the application boundaries.

2 Characteristic of a secure mobile solutions

A mobile solution is considered secure only when end to end security of the enterprise data / assets is taken care. As mentioned above end to end security is dependent on many factors and some of them are outside the control of solution provider. In the next few sections key factors for building a secure mobile solution have been highlighted.

3 On Device Security

There are many different ways to handle on device security.

3.1 Mobile apps password policy

Client side Policy:

- The user's mobile application password shall never be stored persistently on the device.
- The user's mobile application password shall have a minimum length of 8 characters.
- Mobile application password should kept in working storage (main memory and overwrite when app goes to background).
- Users have to provide the mobile application password again every time the app comes back to foreground.
- By default, any already persistently stored data to which the application-level encryption was applied shall be deleted after 20 unsuccessful attempts to provide the correct mobile application password.

Service side Policy:

- The user's mobile application password has to have minimum length, upper-case and lower-case characters, numbers and special characters as defined by the configured policy.
- The number of unsuccessful attempts to provide the correct password before data will be deleted shall be enforced as configured by the customer.

NOTE: Server side policies are not fully captured in this document considering that there will be client installed on the device.

3.2 Password change and reset functions

- Mobile apps shall provide a function for the user to change the mobile application password
- The old password needs to be successfully verified before a new one is set.
- Critical data encrypted with the old password either has to be re-encrypted or discarded (depends on the amount of data, the time needed for re-encryption and app-specific requirements).
- Mobile apps shall provide a function for the user to change the logon credentials (logon username/logon password) stored on device Ideally, it shall also be possible to apply the change to the server-side system used for authentication (the old logon password needs to be successfully verified before a new one can be set).
- Mobile apps shall provide a function for the user to reset the app to its initial state.
- All data stored on device shall be deleted when that function is called to enable users to leave a clean state before deleting the app.

3.3 Deletion of Data On-device and Platform

Data stored on device has to be deleted according to its assigned data retention policy, this applies to business data, and personal data as well.
Usually, the data retention policy is defined in the business backend system and the app shall enforce it also on-device. If not defined in the business backend, the app has to enforce an appropriate data retention policy on its own. To timely reflect also any authorization changes in the business backend it is recommended to keep the retention period on the device even shorter than in the backend and to fetch the data again from the backend when needed.

Deletion of Data on Device

- Personal data shall be erased after it's originally defined retention period has expired, or after an extended and overruling (mandated by law) retention period has expired.

- Personal data shall be blocked after its originally defined retention period has expired, but additional applicable extended and overruling (mandated by law) retention period is still in place.
- The change log entries for the data erasure action shall not disclose the content of the erased data.

Deletion of Data on Platform

- Personal data removal performed by platforms and applications shall not be accomplished by logical deletion (just marking data as erased) but ensure physically deletion.
- Platforms shall provide a framework for retention management that applications shall leverage.

3.4 Secure storage of data on device, if required

For a business application, data is critical in terms of confidentiality, integrity or privacy. In other words, if these data gets disclosed or compromised by an attacker, the business impact can be serious.

Confidential Data includes:

- Passwords, Pass phrases, Cryptographic keys (except public keys), Certificates, other credentials.
- Business critical data like financial results, sales figures, intellectual property.
- Personal data.
- Data containing credit card numbers, bank account information, social security insurance numbers.
- Logs & Traces.
- System internal data like software version, configuration, etc.

One of the most common vulnerability in On-Device applications where data is stored locally, some scenarios are given below:

- To improve performance Online apps use local storage on device for caching
- Advanced functionalities like offering the user to store logon credentials, such as username and password, to improve usability when working with the app.
- Create and work with a local database (offline line mode due to network outage or network unavailability) which is synchronized with the backend when the app gets online
- Create and work with locally stored documents and data.

Mobile apps can use of the below measures to make sure that persistent data is secure and safe.

The DataVault APIs

Provide a secure way to persist and encrypt data on the device. The data vault uses AES-256 symmetric encryption of all its contents. The AES key is computed as a hash of the passcode provided and a "salt" value that can be supplied by the device application developer, or automatically generated through the API. Developers should use data vault APIs for storing all the client side configurations like username, passwords etc.

Device client database (Applicable to Object APIs)

DB.generateEncryptionKey() method in the Object API for MBO packages should always be used during application initialization. It computes a random AES-256 bit encryption key used to encrypt the client database. The encryption key is stored in the data vault.

Persistence APIs (Applicable to oData)

SDMPersistence APIS can persist data in secure and non-secure mode. Developers must make sure the secure mode by using preference PERSISTENCE_SECUREMODE_BOOLEAN. Encryption is done using the secret key provided in SDMPersistence initialization or by using the setEncryptionKey API.

HWC Data Storage APIs

HWC container storage APIs stores the data in encrypted form using device OS databases. Solution developers should use these APIs to store any key-value pair on device for offline usage. One key thing to keep in mind that data stored using SUPStorage APIs is accessible from the java script loaded from an external HTTP server.

Data Security in non-SDK (http) based application

For http based applications client side SDK APIs are available. In such cases developer must use API's of device OS or third party libraries to secure the data stored on device.

Auto removal of data

There are various scenarios in which the app should automatically remove the data stored on devices. Some possible scenarios could be:

- When the application user is switched in cases where multiple users can use the same device. For other cases applications should disable the switching of the user and force reinstall / cleanup of device data before change of user.
- Multiple incorrect attempts of passcode, passwords and detection of inappropriate usage. Any kind of backdoor entry is strongly discouraged.

Other recommendations

- Mobile apps shall not store confidential data on the front-end, whenever possible.
- The user's mobile application password shall never be stored persistently on the device.
- Authentication information shall not be permanently stored on the client
    - Persistent cookies for authentication
    - locally storing previously entered passwords
- Highly critical business data and sensitive personally identifiable information shall only be kept in working storage (main memory and overwrite, if possible, when app goes to background) or transient storage (main memory).
- No confidential data shall be stored on the front-end (including caches) unless triggered by the user (e.g. by downloading business data to the local hard drive).

3.5 Access to Local Data and Resources, Inclusion of Broadly Generic Functions

If mobile app provides interfaces to be called by other apps on-device, for example using Custom URL Schemes on iOS, they shall protect themselves against Cross-Site Request Forgery (CSRF) attacks.

When accessing local resources on-device, such as the Address book, Calendar or geo location services, apps have to prompt the user and ask for user consent before proceeding.

General purpose functions which can be used completely out of context, such as reading or deleting arbitrary data, locking resources on device, adding or executing code etc. shall not be provided by apps.

Mobile apps:

- Shall be protected against Cross-Site Request Forgery (XSRF, CSRF, Session Riding) attacks.
- Running on front-ends shall always prompt the user before processing or exposing local data.
- Shall not include broadly generic functions for which security cannot be enforced.

Note: SAP recommends full device encryption using Afaria; however this is outside the scope of solution development

4 Secure communication

Security of communication between SAP Mobile Platform and device is primarily dependent on the infrastructure setup and configuration. Following would be the guidelines for securing mobile solution while deployment.

- Relay Server is the first line of defense to the platform by acting as a proxy for the device, and facilitating interactions with Unwired Servers installed on the corporate LAN. While deploying make sure to configure Relay Server to use secure protocols, ports, and certificates in Sybase Control Center.

- Securing replication apps (Object API) – Solution developers should use one of the available methods for encrypting the data on network. Developers can choose HTTPS or End to End Encryption (E2EE) or both. Depending on the method chosen following might be required:
  - Client side code changes
    - setNetworkProtocol("https") method of synchronizationProfile.
    - setTrusted_Certificates method of stream parameters.
    - cp.getStreamParams().setE2ee_Type("RSA").
    - cp.getStreamParams().setE2ee_Public_Key(sdcarddirectory+ApplicationName+"_e2eeKey.key").
  - While deploying in production default certificates provided by SMP should be replaced with your own certificates. You can generate these certificates using provided utilities or PKI system of the enterprise.
  - For detailed information on setting up security for replication apps see Encrypting Synchronization for Replication Payloads

- OData Apps - By default network traffic uses HTTP protocol which contains encrypted messages using RSA. For this to work RSA public key needs to be provisioned on device. This can be done either through Afaria or connecting the application through corporate network (Without going through Relay server). Https can be used by calling enableHTTPS() API and making appropriate changes to server settings in SCC.

SAP Gateway scenario (mobile apps developed using SAP Gateway only)

Connection between the app (on-device) and the target servers shall be strongly encrypted. Wherever possible, mobile apps shall make use of the HTTPS protocol support as provided by the mobile platform.
Mobile solution shall support the use of encrypted communication connections .

- Mobile apps shall make use of the HTTPS protocol support as provided by the mobile platform.
- Proper encryption of communication connections (Cryptographic Functions).
- Mobile apps shall not pass locally managed trusted certificates into this verification call for server certificates, but always rely on trusted root CA certificates as available in the device's default root store.
- For software supporting multi-tenancy make sure that each tenant is able to maintain their own certificates for establishing secure communication.

Mobile solution shall avoid unencrypted storage of confidential data during transfer. Mobile apps:

- Shall avoid unencrypted storage of confidential data during transfer.
- Shall not store confidential data on the front-end.
- Push notifications shall never contain any confidential data, but merely links to further information or workflows to be executed, where these links enforce user authentication and authorization when being used.

NOTE: Security features related to communication with device depends on overall server configuration (ports, protocols and certificates) and security configuration defined for the solution. SAP recommends solution developers to evaluate various security related features and implement the relevant one while developing and deploying any solution.

Multi-Layered Defense (Defense in Depth): Server Side

This is a server-side requirement and belongs to Network and Communication security.

- Mobile solution shall support to be deployed and run securely in segmented networks (including firewall systems at network and application level).

For enhanced security SAP recommends usage of mutually authenticated connections which requires provisioning of certificates on the device. This can be done either through Afaria or connecting the application through corporate network for first time use.

5 User Onboarding Access control

This is one of the critical aspects of overall solution security and interlinked with overall server setup in the enterprise landscape. One key thing to understand is SMP does not provide proprietary security systems for storing and maintaining user credentials (except for development setups) and access control rules, but delegates these functions to the enterprise's existing security solutions which needs to be integrated with Common Security Infrastructure (CSI) component of SMP.

- **User Onboarding** – There are 2 kinds of onboarding automatic and manual. Manual onboarding requires whitelisting of the user and requires usage of one time password to complete the registration process. Automatic onboarding does not require any additional steps for onboarding / registration process and shall be used where devices and client application distribution is managed through Afaria.

- **Authentication Mechanism and Provider** – A user can be authenticated using user credential provided through the application UI or can be handled using certificates. Depending on the type of the mechanism required a solution developer needs to choose a suitable authentication provider (e.g. LDAP) for the solution and create a security configuration in the server for the same. For full list of supported providers and their details see Authentication Provider.

- **Single Sign-on for SAP** – SMP supports Single Sign-on credential support for connecting to SAP Backend systems in following ways:
  - **X.509 certificates** – use the CertificateAuthenticationLoginModule provider to implement X.509 authentication. . At runtime, the mobile client selects the certificate signed by a trusted CA, which is authenticated by the SAP EIS.
  - **SAP single sign-on (SSO2) tokens** – use the HttpAuthenticationLoginModule provider for both basic HTTP authentication and to implement SSO2.
  - **Implementing Single Sign-on (SSO)** - Implementing single sign-on requires setting up SAP environment, creation of security configuration, setting up a connection template and security profile setup. More details of the same can be found at SSO for SAP Backends

- **Http based application** – Pure Http clients can authenticate themselves using any of the four mechanism as outlines in this section Http client authentication

Client side provisioning for security

Most of the security features requires client side provisioning in addition to usage of certain APIs in code. These are listed here for quick reference -

- **Encryption keys** – Keys that encrypts the data on device and connection to Relay Server / SMP
- **Security profiles** – Profiles saved to the device and contain sensitive information.
- **SSO and other certificates** – Certificates that identify the user and required for mutual authentication
- **Credentials & Passcodes** – Device user names, one time passwords and passcodes etc.
- **Connection properties** – Settings for values of server, host, port, farm etc. and preventing any connection properties changes.

Cryptography & Secure Configuration

Mobile solution

- Shall use approved cryptographic functions, protocols, certificate and key management solutions only.
- Shall not use static or hard-coded cryptographic keys, passwords and other authentication credentials.
- Shall be delivered with a secure default configuration.

Solution developers should pay careful attention to these artifacts as they are vital for building and running a secure solution

SAP recommends usage of Afaria for automatic onboarding for enhanced security. For SAP back ends SAP encourages usage of SSO for authentication.

6 Additional considerations (TBD)

Code Signing
Signing and verification of the client binary code is important to make sure only the trusted applications are accessing the enterprise systems.

Client Event Tracking / Logging
Implementing mechanisms to track and log various security related events like login failure

Data distribution
Validate implementation of data distribution logic to prevent data leaks within enterprise users

Copyright