Mobile Governance Framework

Introduction

The adoption of the Digital Oman Strategy in March 2003 was one of the steps towards achieving the vision of His Majesty Sultan Qaboos bin Said to transform the Sultanate of Oman into a knowledge-based economy. The strategy describes the key aspects of developing the Sultanate of Oman into a digital society and the implementation of eGovernment.

The Cabinet of Ministers has also stressed repeatedly on the importance of eGovernment transformation. In its meeting No. 8/2012 dated on February 28, 2012, the Cabinet focused on the evaluation of the ministries and government entities' ability to provide services electronically and enabling them to start implementing the eGovernment Transformation Plan.

Based on these strategic directions to achieve eGovernment, the Information Technology Authority (ITA) has developed the eGovernment Transformation Plan and the corresponding set of eGovernment Transformation Policies that the Cabinet of Ministers has approved in June 2012. All government entities shall comply with the eGovernment Transformation Policies as they implement the various stages and targets defined in the eGovernment Transformation Plan. The first stage of eGovernment Transformation program is to establish ePresence – Government entities should have up-to-date websites with basic information accessible to citizens.

According to statistics provided by NCSI, the total number of mobile subscribers has exceeded the total population of Oman. With so many citizens using smartphones (phones designed for mobile web browsing), it is particularly important to ensure that citizens can access government entities' web content on mobile devices. The first step in this regard is to think about whether website works properly on mobile devices.

A family of four in Oman handles around six mobile phones.

Total Number of Mobile Subscribers: 5,617,426
Total Population of Oman: 3,992,000

Data from NCSI, Mobile Subscribers as of end of 2013, Population data as of end of Feb 2014

This framework was created to be a "starting point" that will help government agencies understand how to methodically use the mobile platform and improve their journey towards eTransformation. It covers:

- Direction for Government Agencies
- Guidelines for Mobile Compatibility
- Guide to Create Mobile Solution Architecture
- Guide to Develop Secure Mobile Application

Mobile Governance Framework

Chapter 1: Direction for Government Agencies

1. Government agencies, at the first stage, should make their websites mobile-compatible, taken into consideration the limitation of mobile devices (e.g., smaller screen, bandwidth limitations, etc).
2. Government agencies should consider W3C Recommendations on One Web Approach while publishing their content on the web.
    a. One Web approach - users shall get the same information and services regardless of which devices they use to access the Web Content.
3. Government agencies should consider providing their public services through mobile applications to the extent feasible on mobile platforms. If any agency choose to develop mobile application to deliver services, it should consider following:
    a. Mobile Solution Architecture should be developed using guidelines provided in Chapter 3.
    b. At least the two most popular mobile platforms (such as Android, and IOS) should be supported by government mobile application in Oman.
    c. Mobile applications shall be developed in accordance with “Mobile Application Secure Development Guidelines” (Chapter 4).
    d. Application development guidelines presented in OeGAF should be used as a general reference.
4. Prior to publishing any government mobile application on any application store:
    a. Government agencies must ensure that an independent security assessment is performed on the mobile application and all critical vulnerabilities are closed.
    b. ITA reserves the right to conduct random security assessment to government mobile applications to ensure that they are vulnerability free.
    c. The mobile application should be digitally singed by an accredited trusted certification authority, before publishing the application, to ensure that it is the official version that was published by the agency itself (utilizing ITA PKI where possible).
    d. The application must undergo comprehensive testing for performance, usability and compatibility.
5. Mobile applications should support ITA Mobile PKI for services requiring authentication (example: to their backend servers, etc.).
6. Mobile Applications should utilize ePayment gateway for services requiring payments.

Chapter 2: Guidelines for Mobile-Compatibility

While many mobile devices can browse entire web pages, the small screen often makes viewing content and using site navigation more difficult than on computers. There are three approaches your agency can explore to optimize your online content for mobiles:

1. Developing a mobile version of website
2. Developing mobile-friendly website
3. Developing mobile apps.

Is mobile optimization for you?

The process of optimizing your website for mobile devices can cost time and money, so it is worth considering whether or not it is going to be a good investment for your agency.

If your agency have limited budget for online projects, then mobile optimization may not be a priority.

Unless your website is complicated, or has the majority of its content presented in Adobe Flash format or MS Silverlight, it is likely that visitors on mobile devices will be able to use your site—even if it has not been optimized for mobile devices.

1. Developing A Mobile Version Of Website

It is possible to build a mobile version of your website that is designed to be automatically displayed when viewed on devices with small screens, such as mobile phones, even when visitors enter your regular domain name into their mobile phone web browser. For example, website has already been developed for a government entity since ages back when the new technology was not adopted to make the website compatible with the mobile devices. Adhoc or interim approach can be adopted to create a mobile version of the website covering the most important details or services.

Mobile versions of websites often share the same basic content as a site designed for a computer screen, presented in a simpler fashion. Even though this content is specifically designed for mobile devices, it is also good practice to offer users the ability to choose to view the standard website from their device.

2. Developing Mobile Friendly Websites

Instead of creating an alternative site for mobile devices, another option is to use a single site and make it mobile-friendly.

There are a few things you can do to make your site more mobile-friendly using writing and design techniques:

- Do not use any more text than you have to. Reading on a small screen can be difficult.
- Avoid using too many images. Internet connections on mobile devices may be slow.
- Use clear, simple navigation.

3. Developing Mobile Apps

There are many thousands of mobile phone applications, or 'apps', now available through online stores, such as Apple's App Store, the Google Android Store and Nokia's Ovi Store. Apps allow people to access news and information and perform countless activities, from accessing information to paying online.

Mobile apps differ from the mobile web in that mobile apps are normally developed specifically for a certain mobile phone operating system such as iOS, Android, Windows Mobile or Symbian.

Most apps are provided to users for low or no cost and can be strong promotional tools to support your business. They can boost your online presence and increase your communication with citizens.

Before deciding to create your own mobile app you need to think about what value it will offer to users and how it will benefit your agency.

There are pros and cons to mobile apps, including:

Pros	Cons
Users may be able to use your mobile app while their mobile device is not connected to the internet.	To provide for all mobile web users, you will have to develop your app for multiple mobile phone operating systems.
You can promote your mobile app through stores like Apple's App Store, the Google Android Store and Nokia's Ovi Store.	You might need to pay an app developer every time you want to update your content (the audiovisual material on a website. It includes text, images, video, audio and animations) which could be very expensive if you need to update online content regularly.
Mobile apps can be designed to take advantage of hardware features in particular devices, like GPS or camera.	Small businesses may not actually need the functions that an app can provide. There could be existing apps that you could choose to list your business on instead.

Mobile Governance Framework

To develop your own mobile app, you may need to commission a software developer. Developing mobile apps can cost more than building a basic website.

Comparison of The Three Approaches

A comparison of the three different approaches you can use to get your web presence ready for mobile devices:

		How easy is it to develop?	How easy is it to update?	How easy is it to use?
1.	**Developing mobile version of website**	Generally **easier** than a mobile app, but **harder** than making a website mobile-friendly.	Generally, neither **easiest**, nor the **hardest**. Updating a site with an alternative mobile version can take longer than it does for other sites because content changes must be applied to the desktop site as well as the mobile site.	Typically **easier** than a mobile-friendly site, but **harder** than a mobile app.
2.	**Developing mobile-friendly website**	**Easier** – does not necessarily require extensive knowledge of markup language if you make use of free plug-ins.	**Easier** – a single update to the content of your site will take effect for all devices at the same time.	**Most difficult** – making modifications to an existing website means you do not have complete control over its presentation and structure.
3.	**Developing a mobile app**	**Most difficult** – there is currently no way to develop an app that will work in all the different mobile operating systems.	**Difficult** – your app may not automatically update on users' mobile devices. Instead, they may have to run the app update process manually.	**Easier** – however apps need to be downloaded and installed before they can be used, which can be a frustrating experience for users who may only use once or twice.

Definitions:

CMS (Content Management System) - Software that enables users to create, present, store, retrieve and manage web content such as text, images, documents and audio/video files.

Content - The audiovisual material on a website. It includes text, images, video, audio and animations.

CSS (Cascading style sheet) - A style sheet language that is often used to change the way that a website looks, based on the specifications of the device that is accessing it. For example, cascading style sheets can make a website display in a certain way on a smartphone and in another way on a computer

Markup Language - Refers to common computing languages such as Hypertext Markup Language (HTML) and Extensible Hypertext Markup Language (XHTML).

Plug-ins - A program that provides additional features to a host application when used in conjunction with it, for example to enable a web browser to display different multimedia formats.

Chapter 3: Guide to Create Mobile Solution Architecture

Introduction

The process of creating a mobile solution is very complex. This is because a mobile solution is composed of more than a mobile application or a mobile device management (MDM) tool.

Government agencies must assess business requirements related to mobility, and device a viable mobile strategy to satisfy user needs while balancing security and usability. A methodical approach should be adopted to analyze trade-off across all the options while making the choices.

The risk of failing to consider all of the relevant options is that agencies may make choices that are in conflict with each other and that have unintended consequences (for example, unacceptable risk, excessive cost, poor user experience and unmet business requirements). It is easy for this to happen because the conflicts may be subtle, and oftentimes, no single person or team completely understands the "big picture" of how the mobile architecture is interrelated.

The purpose of this guide is to help government agencies navigate through the complex process of considering all of the relevant options in order to create a mobile solution architecture that satisfies their business requirements and user needs.

How to Create Mobile Solution Architecture

Government agencies should begin creating their mobile architecture by determining their use cases and business requirements. Avoiding this first step may create solutions that do not satisfy business requirements.

Step 1: Determine Use Case and Business Requirements

New opportunities, to engage citizens, are emerging as people rapidly embrace the use of new mobile devices that provide fast wireless communication, run innovative social applications and leverage cloud computing. Unfortunately, IT is burdened by an overwhelming array of regulatory, compliance, security, technology, expense, organizational, policy and legacy technology constraints that limit its ability to quickly adapt to this opportunity. Simply put, the innovation rate for mobility, social software and cloud computing is accelerating faster than the government agency adaptation rate.

Mobility presents agencies with a broad set of interrelated challenges. It is often impossible for an agency to make thoughtful decisions in one area (for example, mobile application development) without considering another area (for example, security and risk management). It is therefore critical for agencies to thoroughly analyze mobile use cases and business needs in order to avoid creating solutions that satisfy neither the citizens nor the business. Agencies must determine which use cases justify investment and then create a mobile architecture that satisfies business requirements. Note that there could be multiple architectures depending on the number of use cases. It is unlikely that one architecture will meet all use cases.

Step 2: Determine Application Architecture Requirements

Application developers have multiple options in terms of how they design their mobile applications, where and how to run the applications, and where they manage application data. The four fundamental dimensions of application architecture include:

1. **Application deployment:** How are applications deployed to the mobile device, and who manages and controls application updates?
2. **Application runtime:** Where do applications run on the mobile device, and how do users launch the application?
3. **Code partitioning:** How is application code partitioned across the mobile device and the server?
4. **Data residency:** Where is the application data?

Each architectural dimension option impacts or constrains other architectural dimensions.

1. Application Deployment

The first mobile application architectural dimension determines how an application gets deployed to a mobile device and who manages and controls application updates. The code deployment options are:

- Static
- Mixed
- Dynamic

Static Deployment

When using the static-deployment option, the application provider publishes and deploys the application and all application updates via public or private application stores or similar distribution channels. Users control initial deployment and updates. The implications associated with static deployment are as follows:

- Application runtime - Native applications that run directly atop the mobile OS require static deployment. Hybrid applications and embedded-container applications may support static or mixed deployment. Applications that run in a browser, hosting container or virtualization server use mixed or dynamic deployment.
- Code partitioning - All or part of the application must execute on the device.
- Data residency - If all code runs on the device, then data must also reside or be cached on the device. Device-resident data may be deployed with the application code via an application store, or it may be deployed or cached during initial setup. If code partitioning is mixed, then data can reside on both the device and the server or just on the server.

- Frameworks - Developers can use native SDKs, cross-compiler frameworks, hybrid frameworks or embedded-container frameworks to implement statically deployed applications.

Mixed Deployment

The mixed deployment option requires initial deployment of the application through an application store, although it supports dynamic updates or customizations via server-side software. The implications associated with mixed deployment are as follows:

- Application runtime -Hybrid applications typically support mixed deployment. These applications must be downloaded from an application store. They run directly atop the mobile OS but can retrieve updates and customizations from their associated Web applications. Applications with embedded containers also may support mixed deployment. They are downloaded from an application store but may be able to receive updates from the associated server-side middleware.
- Code partitioning -All or part of the application must execute on the mobile device.
- Data residency -If all code runs on the device, then the data must also reside or be cached on the device. The data may be deployed statically or dynamically. Mixed code partitioning supports all mixed and server data residency options.
- Frameworks -Developers can use hybrid or embedded-container frameworks to build applications that support mixed deployment.

Dynamic Deployment

When using the dynamic deployment option, the application provider publishes and deploys applications and application updates via server-side software such as a Web server or server-side mobile middleware. Users retrieve the application by accessing the server-side software through an associated mobile application. For example, users access an MWA using a browser. Managed-container systems that supply a hosting-container RMA also support dynamic deployment. In this case, users launch the hosting-container RMA to access applications. When updating applications, developers deploy the revised code to the server-side software, and the server-side software pushes the updates to the mobile device dynamically and automatically the next time the user launches the application. Virtualization systems always use dynamic deployment. The implications associated with dynamic deployment are as follows:

- Application runtime -Browser and virtualization runtimes support dynamic deployment. Hosting containers support dynamic deployment. Mobile OS and embedded-container applications cannot be deployed dynamically.
- Code partitioning -Browser applications can be cached on the device, or the code can be mixed across the device and the server. Hosting containers also support device-resident or mixed code partitioning. When using virtualization, all code must run on the server.

- Data residency -If all code runs on the device, then data must also reside or be cached on the device, and the data must be deployed or cached dynamically. Mixed code partitioning supports all mixed and server data residency options. Virtualization requires server-side data.
- Frameworks -Developers can use mobile Web frameworks and hosting-container frameworks to implement applications that support dynamic deployment.

2. Application Runtime

The second architectural dimension determines how a user launches an application and the environment in which the application executes on the mobile device. The application runtime options are:

- Mobile OS runtime
- Browser runtime
- Managed-container runtime
- Virtualization

Mobile OS Runtime

Native and hybrid RMAs execute directly atop the mobile OS runtime option. The application must be installed on the device, and the user must launch it via a dedicated icon. The implications associated with using the mobile OS runtime are as follows:

- Application deployment -Applications that run directly atop the mobile OS must initially be statically deployed to the mobile device using an application store or similar distribution channel. Hybrid applications can support the mixed deployment option.
- Code partitioning -All or part of the application must execute on the mobile device.
- Data residency -If all code runs on the device, then the data must also reside on the device. Mixed code partitioning supports all mixed and server data residency options.
- Frameworks -Developers can use native, cross-compilation and hybrid frameworks to implement applications that run directly atop the mobile OS.

Browser Runtime

MWAs use the browser runtime option. An MWA can be cached on the device, in which case users can launch it via a dedicated icon. Otherwise, users must launch the browser and then access the application by typing in a URL or using search to find it. The implications associated with using a browser runtime are as follows:

- Application deployment -Applications that run in a browser support dynamic deployment.

- Code partitioning -The application code can run entirely in the browser on the mobile device, or it can be mixed across the device and the server. All or part of the application can be cached on the mobile device to support disconnected processing.
- Data residency -If the application code runs entirely on the device, then data must also be cached on the device. However, that data is read-only data. Mixed code partitioning supports all mixed and server data residency options. Server-based code requires server-based data, and vice versa.
- Frameworks -Developers use mobile Web frameworks to implement applications that run in a browser.

Managed-Container Runtime

A managed container provides runtime and management services to RMAs that are developed using a container framework. A container typically supports mobile device management features, such as dynamic deployment, data encryption and remote-wipe capabilities, although the exact set of features varies depending on the container framework being used.

The container separates the managed applications and their data from other applications on the mobile device. Information managed by the container is typically protected by authentication and encryption. Most container frameworks provide associated server-side software that administrators use to manage policies, configurations and updates. If the container supports remote-wipe capabilities, administrators can disable applications and remove information held within the container without affecting other information or applications on the mobile device.

Managed containers come in two basic flavors:

- **Embedded container:** In this variant, the developer uses the container framework to embed container capabilities within an RMA. From outward appearances, the RMA looks and feels very much like an RMA that runs on the mobile OS, except that the embedded container intercepts API calls to the mobile OS and injects various behaviors, such as data encryption. The container also exposes an API that enables server-side software to manage the application. RMAs with an embedded container support static, or sometimes mixed, deployment. They must be downloaded from an application store. In some cases, the server-side software can deploy updates dynamically.
- **Hosting container:** In this variant, the framework provides an RMA through which users access managed applications. The hosting-container RMA is somewhat like a browser or a virtualization viewer. The user launches the hosting-container RMA and then accesses applications via a menu in the container application. All containerized applications run within the hosting container. The hosting-container RMA must be statically deployed, but all managed applications are deployed dynamically.

The implications associated with using managed-container runtimes are as follows:

- Application deployment -Applications with an embedded container support static, and sometimes mixed, deployment options. Applications that run in a hosting container support dynamic deployment.
- Code partitioning -All or part of the application must execute on the mobile device.
- Data residency -If the application code runs entirely on the device, then data must also be resident or cached on the device. Mixed code partitioning supports all mixed and server data residency options. Server-based code requires server-based data, and vice versa.
- Frameworks -Developers use a container framework to implement applications that run in a container.

Virtualization

A virtualization system enables mobile users to access applications running on a remote system. The user launches a virtualization viewer RMA on the mobile device, which communicates with the virtualization server-side software using a proprietary protocol. The viewer RMA displays the remote virtual desktop, and the user can then launch applications available on the remote system. The implications associated with using a virtualization runtime are as follows:

- Application deployment -The virtualization viewer application must be statically deployed on the mobile device, but the individual applications are not deployed to the mobile device. Individual applications are accessed dynamically.
- Code partitioning -The applications always run on the server.
- Data residency -The data always lives on the server.
- Frameworks -No mobile frameworks are required to support virtualization.

3. Code Partitioning

The third architectural dimension determines how you partition the application code between the mobile device and the server. Server-side software comes in many flavors: It may be written specifically for the application in question, it may comprise one or more shared services that enable access to various back-end applications, it may be server-side middleware supplied by a mobile platform vendor, or it may be some type of cloud service — also known as back end as a service (BaaS).

Code partitioning and data residency decisions are closely related. In most scenarios, it's more efficient to process data where the data lives than to move data back and forth between the mobile device and the server. If you maintain any data on the server, then an application must have server-side software to process the data. The code partitioning options are:

- Device

- Mixed
- Server

Device-Resident Code

In this code partitioning model, all application code runs on the mobile device. When all application code runs on the device, then data must also be resident or cached on the device. Both RMAs and MWAs can execute entirely on the device. Device-resident code and data enables the application to work when disconnected from the network. The implications associated with running all application code on the mobile device are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment.
- Application runtime -RMAs can run directly atop the mobile OS or in a container (either embedded or hosting). MWAs that run in a browser can be cached on the device for disconnected processing. Virtualized applications cannot run on the mobile device.
- Data residency -Applications that run entirely on the device must maintain data on the device. That data may be stand-alone data that lives only on the mobile device, or it may be cached from a server for read-only access. If the device-based code needs to make updates to the data and those updates must be propagated to the server, a mixed code partitioning solution is required. At a minimum, the application requires a server-based service to synchronize the device-resident data with server-based data.
- Frameworks -Developers can use any type of mobile framework to implement device-resident applications.

Mixed Partitioned Code

In this code-partitioning model, part of the application code runs on the mobile device, and another part runs on the server. The server-side code may be a Web application, Web service, server-side middleware, BaaS or any back-end application. Mobile server-side middleware provides technologies that enable mobile applications to integrate with back-end applications. The client and server code typically communicate using HTTP, although some server-side middleware systems use proprietary middleware. The implications associated with mixed code partitioning are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment.
- Application runtime -The device-based code can run in the mobile OS, a browser or a container (hosting or embedded). Server-based code may be implemented as a Web service, Web application or BaaS, or the mobile client can invoke server-side code through server-side middleware. Virtualized applications cannot run on the mobile device.

- Data residency -Applications with some code on the device and some code on the server can support all mixed or server-based data residency options.
- Frameworks -Developers can use any type of mobile framework to implement mixed code-partitioning applications.

Server-Resident Code

In this code-partitioning model, all application code runs on the server, and likewise, all data resides on the server. Virtualization is the only runtime solution that supports all code on the server. The implications associated with server-resident code are as follows:

- Application deployment -The virtualization viewer application must be statically deployed on the mobile device, but the individual applications are not deployed to the mobile device. Individual applications are accessed dynamically.
- Application runtime -Virtualization is the only solution that supports all code on the server.
- Data residency -If all code is on the server, then all data must also be on the server.
- Frameworks -No mobile frameworks are required to support virtualization.

4. Data Residency

The fourth architectural dimension determines where you maintain an application's data. Many factors will influence your data residency decision, such as data volume, network connectivity, bandwidth and data sensitivity.

Data residency and code partitioning decisions are closely related. If all code runs on the device, you must maintain data — either resident or cached — on the device. Likewise, if you maintain any data on the server, then the application must have server-side software to process the data. The data residency options are:

- Device
- Mixed cached
- Mixed unsynchronized
- Mixed synchronized
- Server

Device-Resident Data

In this data residency model, all data lives on the mobile device, and likewise, all code runs on the mobile device. This data residency model only works with RMAs running directly atop the OS or running in a container. Device-resident code and data enables the application to work when disconnected from the network. The implications associated with maintaining all data on the mobile device are as follows:

- Application deployment -Only mobile OS and container applications can support all data on the device. Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment. Initial data may be installed during deployment or during initial setup.
- Application runtime -Mobile OS and container runtimes support all data on the device.
- Code partitioning -Applications that maintain all data on the device must run all code on the device.
- Frameworks -Developers can use any type of mobile framework that implements RMAs to implement device-resident applications with device-resident data.

Mixed Cached Data

In this data residency model, data lives on the server, and all or part of the data is cached on the mobile device to support better performance and offline processing. Cached data supports read-only processing. If updates must be propagated to the server, the application should use mixed synchronized or server-based data residency models. The implications associated with using mixed cached data are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment. Device-resident data is typically cached dynamically.
- Application runtime -All runtime options except virtualization support the mixed cached data residency option.
- Code partitioning -Applications that maintain data on the server and cache the data on the mobile device can run all code on the device or run code on both the device and the server.
- Frameworks -Developers can use any type of mobile framework to implement applications with mixed cached data.

Mixed Unsynchronized Data

In this data residency model, the application maintains two separate datasets — one on the device and one on the server — and the datasets are not synchronized. The implications associated with using mixed unsynchronized data are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment. Initial data may be installed during deployment or during initial setup or may be downloaded dynamically.
- Application runtime -Mobile OS and container runtimes support unsynchronized datasets on the mobile device and the server.

- Code partitioning -Applications that maintain separate unsynchronized datasets on the mobile device and the server must have code running in both locations in order to process the separate datasets.
- Frameworks -Developers can use any type of RMA mobile framework to implement applications with mixed unsynchronized data.

Mixed Synchronized Data

In this data residency model, the application maintains duplicate synchronized datasets on the mobile device and the server. This option works with any type of RMA. It does not work with MWAs. The device-resident dataset enables applications to work in offline mode, and updates can be propagated to the server upon reconnection. Data synchronization can be accomplished using a variety of methods, such as database synchronization (using server-side middleware), service invocation or email. The implications associated with using mixed synchronized data are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment. Initial data may be installed during deployment or during initial setup or may be downloaded dynamically.
- Application runtime -Mobile OS and container runtimes support synchronized data on the mobile device and the server.
- Code partitioning -Applications that maintain duplicate synchronized datasets on the mobile device and the server must have code running in both locations. Server-side software must support the data synchronization processing.
- Frameworks -Developers can use any type of RMA mobile framework to implement applications with mixed synchronized data.

Server-Resident Data

In this data residency option, all data lives on the server. In this scenario, some aspect of the application code must run on the server. The implications associated with using server-resident data are as follows:

- Application deployment -Mobile OS and embedded-container applications require static or mixed deployment. Applications that run in a browser or hosting container support dynamic deployment. Virtualization applications also use dynamic deployment.
- Application runtime -All runtime options support server-resident data.
- Code partitioning -All or part of the application code must run on the server.
- Frameworks -Developers can use any type of mobile framework to implement applications with server-resident data.

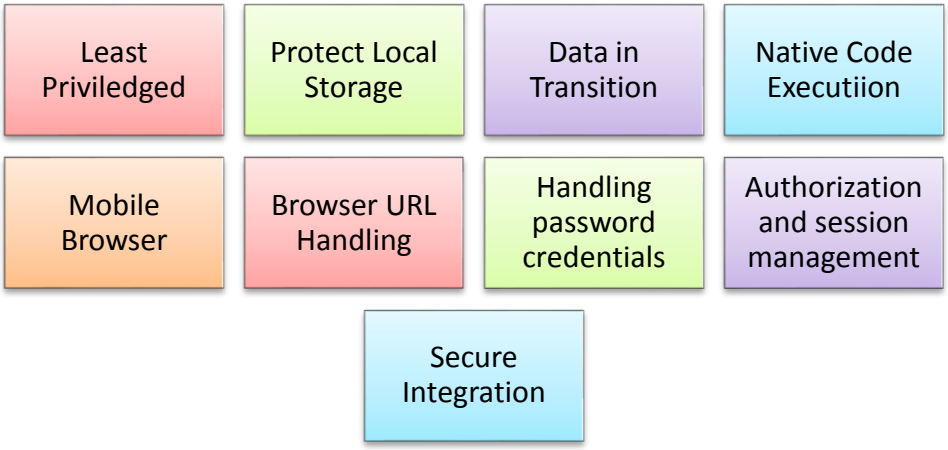
Note: For further guidance and support, agencies are encouraged to contact ITA.

Chapter 4: Guide to Develop Secure Mobile Application

Mobile and smartphone applications have very different threat models than their web-based counterparts. Developers need to both understand the capabilities of their chosen development platform(s) as well as understand how to design and build applications to securely take advantage of mobile capabilities without exposing their government agencies or end user (general public and employees) to unnecessary risks.

Every mobile application platform has different characteristics and so developers must be familiar with the specifics for their platform if they are going to begin developing applications. Developer should know about the characteristics of different platforms (i.e. iphone, Android, etc.) for example their security features, capabilities, and weakness.

Developers should consider application development guidelines presented in OeGAF as a general reference and should focus on following Mobile Apps controls and design principles.



1. **Applying least privilege security principle**

Mobile devices contain sensitive information such as email messages, user contacts and the device owner's current location. In addition, mobile devices have access to sensitive capabilities such as the ability to make phone calls and send SMS messages. Developers should be careful to only ask for permissions that are required for them to accomplish their specific application goals in order to potentially limit damage if their applications are compromised.

2. **Protect local storage**

Mobile devices have the ability to store information in files, databases and other constructs. Because devices can be lost or transferred to other users without being wiped, application developers should be very careful about storing sensitive information locally on the device. Avoiding storing sensitive information on the device is preferable because then the risk of compromise is minimized. If sensitive data must be stored on the device, it should be encrypted to prevent disclosure.

Developer should consider following points:

- In the design phase, classify data storage according to sensitivity and apply controls accordingly (e.g. passwords, personal data, location, error logs, etc.). Process, store and use data according to its classification. Validate the security of API calls applied to sensitive data.
- Store sensitive data on the server instead of the client-end device.
- When storing data on the device, use a file encryption API provided by the OS or other trusted source.
- Do not store/cache sensitive data (including keys) unless they are encrypted and if possible stored in a tamper-proof area
- Do not store historical GPS/tracking or other sensitive information on the device beyond the period required by the application.
- Be aware of caches and temporary storage as a possible leakage channel, when shared with other apps.
- Be aware of public shared storage such as address book, media gallery and audio files as a possible leakage channel. For example storing images with location metadata in the media-gallery allows that information to be shared in unintended ways.
- Do not store temp/cached data in a world readable directory.
- For sensitive personal data, deletion should be scheduled according to a maximum retention period, (to prevent e.g. data remaining in caches indefinitely).
- There is currently no standard secure deletion procedure for flash memory (unless wiping the entire medium/card).
- Consider the security of the whole data lifecycle in writing your application (collection over the wire, temporary storage, caching, backup, deletion etc.)
- Apply the principle of minimal disclosure - only collect and disclose data which is required for business use of the application. Identify in the design phase what data is needed, its sensitivity and whether it is appropriate to collect, store and use each data type.

3. Data in transition

Mobile platforms offer a variety of networking options including provider networks, WiFi and others. Network spoofing attacks, surveillance. The majority of smartphones are capable of

using multiple network mechanisms including Wi-Fi, provider network (3G, 4G, GSM), Bluetooth etc. Sensitive data passing through insecure channels could be intercepted.

Developers should take note not to send sensitive data over unencrypted connections because it might be intercepted by attackers. All communication should be performed through SSL and HTTPS web requests. Care should be taken to force SSL connections to use appropriately strong encryption and to properly verify the identity of the connected server.

Developer should enforce the use of an end-to-end secure channel (such as SSL/TLS) when sending sensitive information over the wire/air

- Use strong and well-known encryption algorithms (e.g. AES) and appropriate key lengths
- Use certificates signed by PKI (Mobile PKI)
- The user interface should make it as easy as possible for the user to find out if a certificate is valid.
- SMS, MMS or notifications should not be used to send sensitive data to or from mobile end-points.

4. Native Code Execution

Running native code opens up opportunities for the introduction of whole classes of vulnerabilities such as buffer overflows and format string attacks. Whenever possible, developers should utilize managed code because it typically provides automatic memory management and array bounds checking. Also, some platforms offer access to buffer overflow protection technologies such as non-executable stacks and address space layout randomization. If it is required to run native code these protections should be used.

5. Mobile Browser

Mobile platforms rely on platform-provided browsers to access the web. These are either used as standalone applications, or are embedded into custom applications in order to provide access to content and functionality. Many attacks on mobile platforms have used the browser as a vector, so developers need to understand how their mobile platform makes use of the included browser.

6. Browser URL Handling

Most mobile platforms allow developers to register their applications to handle requests and content initially handled by the device web browser. This allows developers to provide a richer experience than a web browser on its own could provide, but also opens up avenues for attackers to try and subvert application behavior by seeding malicious websites with specially-crafted links intended to execute a target application, but with malicious parameters. Developers should understand the situations under which their applications might be executed

and be sure to properly validate incoming data and request appropriate confirmation from application users before performing sensitive actions.

7. Handle password credentials securely on the device

Spyware, surveillance, financial malware. A user's credentials, if stolen, not only provide unauthorized access to the mobile backend service, they also potentially compromise many other services and accounts used by the user. The risk is increased by the widespread of reuse of passwords across different services. Instead of passwords consider using longer term authorization tokens that can be securely stored on the device (as per the OAuth model). Encrypt the tokens in transit (using SSL/TLS). Tokens can be issued by the backend service after verification.

8. Implement user authentication, authorization and session management correctly

Unauthorized individuals may obtain access to sensitive data or systems by circumventing authentication systems (logins) or by reusing valid tokens or cookies.

Developer should consider following points:

- Require appropriate strength user authentication to the application. It may be useful to provide feedback on the strength of the password when it is being entered for the first time. The strength of the authentication mechanism used depends on the sensitivity of the data being processed by the application and its access to valuable resources (e.g. costing money).
- It is important to ensure that the session management is handled correctly after the initial authentication, using appropriate secure protocols. For example, require authentication credentials or tokens to be passed with any subsequent request (especially those granting privileged access or modification).
- Where possible, consider using additional authentication factors for applications giving access to sensitive data or interfaces where possible - e.g. voice, fingerprint (if available), who-you-know, behavioral etc.
- Use authentication that ties back to the end user identity (rather than the device identity).

9. Secure data integration with third party services and applications

Data leakage. Users may install applications that may be malicious and can transmit personal data (or other sensitive stored data) for malicious purposes.

Developer should consider following points:

- Vet the security/authenticity of any third party code/libraries used in your mobile application (e.g. making sure they come from a reliable source, with maintenance supported, no backend Trojans)

- Track all third party frameworks/APIs used in the mobile application for security patches. A corresponding security update must be done for the mobile applications using these third party APIs/frameworks.
- Pay particular attention to validating all data received from and sent to non-trusted third party apps (e.g. ad network software) before processing within the application.

References:

For Solution Architecture

- OeGAF http://www.ita.gov.om/OeGAFWeb/Login.aspx
- GARTNER - Mobile Application Architecture http://www.gartner.com/document/2119218?ref=readershipHistory

For Security Guidelines

- OWASP - Secure Mobile Development Guidelines https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Secure_Mobile_Development
- Cloud Security Alliance - Security Guidance for Critical Areas of Mobile Computing https://cloudsecurityalliance.org/download/security-guidance-for-critical-areas-of-mobile-computing/