A secure extensible container for hybrid mobile applications

David Jaramillo
IBM CIO Lab – Mobile Innovations
djaramil@us.ibm.com

Robert Smart
IBM Emerging Technology Services
smartrob@uk.ibm.com

Borko Furht
Florida Atlantic University
borko@cse.fau.edu

Ankur Agarwal
Florida Atlantic University
ankur@cse.fau.edu

***Abstract*— An agile, lightweight, but secure extensible hybrid application container and deployment mechanism for mobile devices has the potential to provide improved cross platform support, reduced development lifecycle time and a consistent application security model within large organizations. We demonstrate how such an architecture can provide a set of security and mobile device management interfaces to lightweight applications written in a high level markup and scripting language and also how these applications can be provisioned via push or pull mechanisms to an Enterprise user's device. The use of such a container for hybrid applications widens the potential support and development resource within an organization to employees possessing readily available skill sets, this permits mission critical applications to be developed in a much shorter time frame than with native mobile application deployment methods.**

Keywords—mobile; hybrid; container; deployment; security

I. INTRODUCTION

The hybrid container builds upon the ideas of solutions such as Apache Cordova (Phonegap) but extends this idea such that the container shell application becomes home to not just one single use application but multiple applications, These are presented to the user as an extensible virtual desktop within one application running on the device which we shall refer to as 'the container'. Additional hybrid applications can be dynamically downloaded and installed within, their data and services managed and encapsulated within the application space of the container.

II. HYBRID MOBILE APPLICATIONS

A. Overview

The term *hybrid mobile application (HMA)* describes an application created to run on a mobile device where the application logic is primarily written in JavaScript and the user interface components are defined using HTML5 and CSS3.

All of the current leading mobile operating system platforms allow applications to display and interact with a 'Web View', this web view is an embedded browser documented and contained within each platforms Software Development Kit (SDK).

The HTML and associated JavaScript application logic are contained within a native code 'wrapper'; this allows what would otherwise be a web application to be installed as an application on the mobile devices desktop. The native wrapper exposes device level function to the scripted portion of the application through a set of plug-ins written in the native language of the particular mobile platform, each plug-in has an associated JavaScript API. The plug-ins can expose native function that would otherwise be inaccessible using standard HTML5, for example access to the device calendar, phonebook or in built hardware such as the camera.

The interaction between the JavaScript API and the native code is implemented on each mobile platform using methods made available through that platforms SDK. Calls from JavasSript to native code exploit the ability to intercept navigation events and examine the URI scheme, a URI scheme can be used to indicate that the URI is an encoded method call and should not be passed on to the embedded browser to handle, the hybrid application shell can then process the encoded URI and call native methods within the application shell. Conversly function calls can be made to the embedded browser code by injecting JavaScript at runtime into the currently loaded web page.

The most frequently used hybrid framework is Apache Cordova more commonly known as PhoneGap [1][2]. Cordova is used inside several mobile application platforms including IBM Worklight, Convertigo and ViziApps; Cordova provides an extensible framework allowing developers to create custom plug-ins and make them available to the scripted portion of the HMA [5].

B. Benefits of Hybrid Development

There are several advantages to using a hybrid development approach when creating mobile applications. Using HTML5 and JavaScript increases the portability of the application; the UI and application logic can be reused across multiple platforms with minor changes to presentation format performed using CSS [6]. Another major advantage is the abundance of HTML and JavaScript development resource in comparison to native language knowledge of Objective-C and Java.

978-1-4799-0053-4/13/$31.00 ©2013 IEEE

III. MOBILE APPLICATION CONTAINERS

Increasingly large organizations are allowing employees to use their own mobile devices for work purposes, this has created a set of Mobile Device Management (MDM) challenges for the IT teams within these organizations. A major challenge is to allow the employee freedom to use the phone for personal tasks unencumbered by lengthy passwords and IT Management enforced policy, yet still protect enterprise data and applications. In order to do this some type of separation mechanism needs to be introduced in order to ensure that there are no data leakages or security intrusions. Protection is delivered in the form of a virtual container that applications can run inside, their data is containerizing at the application level, this is accomplished by wrapping a layer of protection around the enterprise deployed apps, which separates the corporate data from the employee's private information and consumer applications. Further security is provided by some solutions by using specific SDKs that intercept all data related APIs and store the data in secure/encrypted file stores. An additional security measure is to monitor the application using Mobile Device Management and provide functions that give an IT administrator the power to do corporate wipes of the enterprise container contents. A 'time bomb' mechanism maybe employed that will self wipe the container and data if the device has not checked in within a period of time defined by a device security policy. Other solutions like micro-containers have been done which use cloud based services to provide mobile service-containers for hosting service-based applications [3].

IV. REQUIREMENTS FOR A HYBRID APPLICATION CONTAINER

We start by examining several scenarios that illustrate how different sets of users might benefit from such a device container and how the container fits within a broader application ecosystem within a large organization or enterprise.

A. Example Use Cases/Application scenarios

1) Scenario - Project team

A development team within the organization has been charged with creating a mobile time keeping application that uses an existing legacy server system to house the company time records. They have to support 3 mobile platforms Android, iOS and BlackBerry to give the majority of workers access. They also need to read up on and adhere to all of the companies security and privacy requirements, provision external facing servers and obtain the blessing of the Chief Security Officer (CSO). The team don't have native mobile development skills but do have a lot of experience in writing web applications.

2) Scenario - Employee

Susan is an employee who works in Sales, she had a mobile phone given to her by the company but no longer likes to use it as it is out of date and clunky. She has a personal iPhone which she uses for work purposes from time to time, she would like to get at sales data when she travels but currently there is no VPN access and the work email system requires that she enter a long password whenever she wants to unlock her phone. The couple of work applications she has installed involved connecting her phone to a laptop and installing via iTunes which she felt was cumbersome, she hasn't updated the apps even though she received emails indicating updates are available.

3) Scenario - IT Management

Several different groups within the company have started developing mobile applications independently and are distributing them to employees. Other groups within the organization are constantly requesting new applications from the IT Management team who cannot keep up.

The groups who have implemented applications are using differing levels of security in their applications and the CSO is attempting to educate them to meet certain standards and adhere to best practice. Deployment methods for the applications are adhoc and inconsistent, the organisation does not have a clear picture of which applications are proving popular and do not know if the devices they are installed upon are registered with their MDM system.

The above scenarios illustrate why a managed container solution would be useful and dictate that certain properties and capabilities are present within a containerized mobile application solution in order to meet the needs of three distinct groups within an organization. The needs of these groups can be summarized as follows:

a) Project teams

- Utilization of existing technical skill sets.
- Ease of development.
- Ease of deploying initial application and updates.

b) Employees

- Separation of personal and business applications and data on a self purchased device – aka Bring your Own Device (BYOD).
- Ease of installation and management of work related applications.
- Access to work related data and service whilst not in the office.

c) IT Management

- The need to enforce privacy and security policy.
- Be able to have an organisation level view of what applications employees are using and what devices they have.
- Ability to foster a vibrant developer ecosystem.
- Ability to control what applications can access the organizations internal networks.

B. Application creation

Creation of a hybrid application should not require a developer to possess platform specific skills but rather have a grasp of common web development methods including knowledge in HTML5, CSS3 and JavaScript.

The container itself consists of a native application code shell and provides all the necessary services and APIs that an application team would need to build applications. This allows the creators of hybrid applications to focus on the presentation and business logic in a platform independent manner without having to learn new high-level languages such as Objective-C or Java. Web development skills are more widely available in an organization, as a result the pool of developers who can contribute custom applications will in most cases be substantially larger than the set of developers who possess knowledge of the high level languages used to create applications on platforms such as iOS and Android.

C. Application deployment and updates

Application deployment as well as application updates are essential in a container environment and they need to be simple and secure. Application deployment can be performed in two ways. The first by pushing the application(s) during initial installation of the hybrid container, the second deployment is via user requested installation from an app store. The configuration of the applications to be installed is controlled by the backend system that manages or communicates with the hybrid container. In the case of application updates, there are two types that applications are sensitive to. One is the core native shell where plug-ins reside and the second is the internal HTML/CSS code. Depending on the complexity of the application, there may be few updates to the shell and frequent changes to the UI. In the case of updating the shell, this requires a complete re-installation of the application including both the shell plug-in as well as the core UI files. Careful consideration needs to be taken to make sure that any files or settings from previous installations do not conflict or cause adverse effects with the new installation.

D. Appstore model

Hybrid applications by nature are very dynamic and are continuously being updated, new applications may be released that users will want to install after the initial setup of the hybrid container. In order to address this scenario, the hybrid container can be architected to connect to an application store backend that will allow the user to select from a catalog of applications that they are entitled to download and install.

The IT organization can also have a managed set of required applications that would be pushed to the hybrid container ensuring that users always have a set of pre-defined applications.

E. Container API

The container itself must be written in the native language used by each targeted mobile platform. It will provide a consistent set of APIs to application developers; it will also contain the mechanisms to allow hybrid applications to be installed dynamically within the container.

1) Security and Storage

Security is of utmost importance within a large organization, data may have varying levels of sensitivity but the determination of what is or isn't sensitive should not be left purely down to the individual application developer. The best policy for application development is therefore to provide secure storage through strong encryption for all application data. Allowing the container to handle this behind a simple storage API reduces the burden on the application developer and allows IT management to be confident that standards and policy are being adhered to.

Due to the nature of the hybrid container application, data separation must be enforced via strict name spacing to prevent one application reading another's data. It may however be useful to allow sharing of data in certain circumstances e.g. allowing an instant messaging application access to the database of a contacts application.

2) Access to the organizations internal network

Providing access to data and services that are located within an organizations internal network from a mobile device can be problematic and a source of much debate amongst security teams. The option to provide device level Virtual Private Network (VPN) exists but opens the organization up to attacks by malware that may have been unwittingly installed on an employee's personal device [4]. A second option is to provide access on an application-by-application basis often through secure reverse proxy mechanisms.

This is much more secure but can be a lot of effort for the IT management part of the organization. The ideal option is to allow the container to manage access providing a common interface for applications that reside within it.

3) Notifications

Applications in the most part act in a request response manner, however there are occasions where it is necessary to push information to an application. In a mobile context this usually takes the form of a 'push notification' sent from a messaging provider service usually run by the creator of that particular mobile operating system. Apple provides the Apple Push Notification Service (APNS) whilst Google provide the Google Cloud Messaging (GCM) service. It is necessary to abstract the use of these messaging providers away from application developers both for client and server side development.

4) Email/Calendar

Common applications such as email and calendar could be provided with the container rather than on an organization-by-organization basis. This function is required to reside within the container as the information used is of a highly sensitive nature. The protocols used for email and calendar are very well documented and standardized so implementing them as common components is justifiable.

5) Updates

An update mechanism must be present in the container logic providing two distinct functions. Firstly the ability to check for code updates to the container itself, these may be frequent as the demand for new native plug-in components is driven by the arrival of different applications. Secondly to check for new versions of previously installed applications within the container. Both methods of update will require communication with a server based management component.

F. Container security

The container should provide an authentication mechanism that secures access to the applications and data residing within. The credentials used by the container will be determined at install time and may be governed by a policy which enforces a password change at a configurable time interval.

V. CREATING A HYBRID APPLICATION CONTAINER

The decision was made to use the open source Apache Cordova project rather than create the hybrid framework of the container from scratch, Cordova already has a clean extensible plug-in architecture and a large community of developers creating plug-ins to offer additional functionality on top of the base set.

When a Cordova application starts up the native shell code loads the HTML/JavaScript UI application logic from a fixed www root folder within the application file system.

This root folder is read only once the native application is installed so it is necessary to subclass the Cordova class that loads from the root folder, this allows the UI application logic to be loaded from a different writable folder that we can install downloaded hybrid applications within.

A. System architecture

Figure 1 shows a typical deployment configuration for the container architecture and required backend services. A mobile client running the container connects over the Internet to a secure reverse proxy that can perform device authentication and authorization to establish a secure session. Within this secure session the device can contact a variety of services. These services are located behind a firewall in a demilitarized zone (DMZ).

The App Store service allows a user to browse and download new applications to the container, it also provides version management services to the container. Individual applications will need specific backend services made available to them, these can all be accessed via an application service located inside the DMZ. Requests to the service will be distributed to many pre-existing service APIs located within the organizations intranet.

A Security service also resides within the DMZ which provides applications with the service APIs that allow various authentication and authorization tasks to be performed. This service layer also accesses a variety of pre-existing intranet based applications. These can include employee directory servers, Mobile Device Management (MDM) solutions or authentication services.

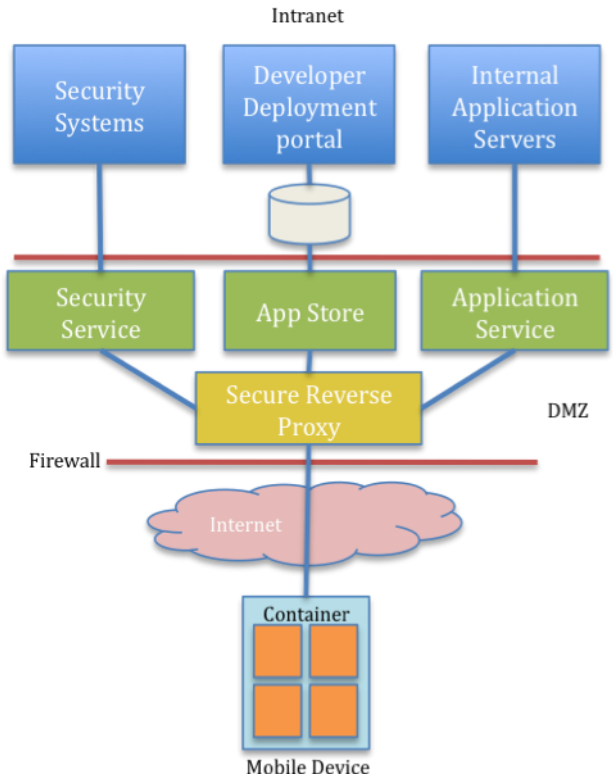

Fig. 1. An example system architecture for a hybrid container deployment

B. Container components

1) Container Application

As illustrated in Figure 2 the container application is made from a variety of layered components, the application itself is written in the native language of the mobile device i.e. Objective-C for iOS, Java for Android. The container contains bootstrap code to load the initial Application Desktop user interface web application in a Web View when the application starts.

An application executing in the Web View can call native code in the form of plug-ins, each plug-in has a JavaScript API and a corresponding native language class implementation.

The desktop contains icons which when selected by the user trigger a call to the Application Manager (AM) which then loads the selected application into the Web View components.

2) Plug-ins

A base set of plug-ins are included with the container, however more may be added over time as the container receives updates. Plug-ins are written to conform to the Apache Cordova interface and many of the standard Cordova plug-ins are included. Each plug-in has a corresponding JavaScript API so that it may be called from a hybrid application.

a) Security

This set of plug-ins provides authentication and authorization components used when accessing services hosted on the organization's intranet, this includes a filtered VPN service.

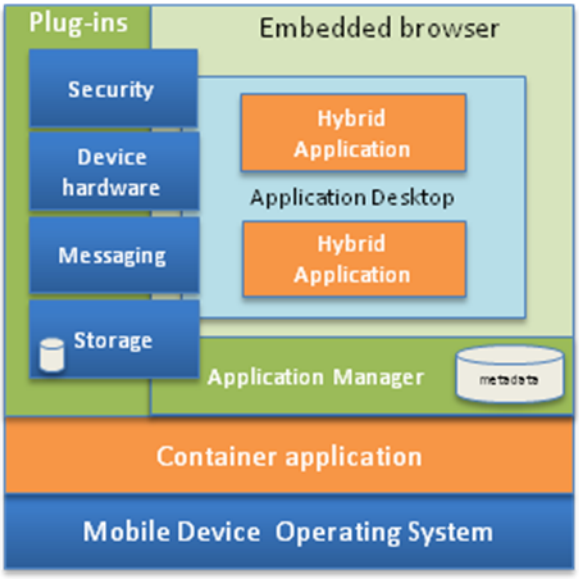

Fig. 2. Representation of the hybrid mobile application container

b) Device Hardware

Access to hardware-based functions of the device such as GPS, Camera and Microphone can be obtained and interacted with through this set of APIs.

c) Messaging

The messaging plug-in set has responsibility for processing incoming notifications and ensuring that the message is routed to the correct hybrid application. The AM is called if the application is not currently running and an appropriate alert can be shown allowing the user to switch to the application that the push notification targeted.

d) Storage

In order to maintain security standards with regards to storage any write operations performed by a hybrid application should go through the storage API this allows the data to be encrypted before it is written. If the application requires access to a database then an application specific sqlite instance is created. The entire database is encrypted using the container password created when the container is first installed.

3) Application Manager

The Application Manager shown in Figure 2 has several responsibilities including loading the desktop on the first instantiation of the container. When applications other than the desktop are started the container stores the state of the current application then reads the new applications descriptor. Any plugins required by the application are instantiated and the web resources loaded into the web view component, if a previous state was stored the state is reloaded before the application displays.

The AM handles the installation of new applications within the container and also periodic version checks for both installed applications and the container itself. Version checks are performed by making an HTTP request to a server based version component within the appstore, applications can be updated dynamically and restarted within the container. In order to update the container itself the full application must be reinstalled.

Once an application has been developed the application resource files are added to an archive file and zip compressed. Along with the resources in the archive are an application descriptor and a desktop icon used to display the application within the container.

Fig. 3. The contents of a hybrid application wrapped for deployment to the container.

The application descriptor contains metadata such as the application name, version, the plug-ins used by the application. Installed application details are maintained within a local database by the container and queried at various points in the applications execution cycle.

VI. CONCLUSIONS

We presented an extensible hybrid mobile application container solution. The container solution offers benefits to three distinct groups within a large organization the IT management group, the application developers and the organizations general employees. We discussed the general properties that such a solution requires and outlined the benefits these properties bring to the organization in terms of adherence to data security policies, device and application management and encouragement of the mobile development ecosystem within an organization.

REFERENCES

[1] Rohit Ghatol, Yogesh Patel, "Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5" Apress.

[2] Palmieri, M.; Singh, I.; Cicchetti, A.; , "Comparison of cross-platform mobile development tools," Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on , vol., no., pp.179-186, 8-11 Oct. 2012.

[3] Omezzine, A.; Sami Yangui; Bellamine, N.; Tata, S.; , "Mobile Service Micro-containers for Cloud Environments," Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012 IEEE 21st International Workshop on, vol., no., pp.154-160, 25-27 June 2012.

[4] Lawton, G., "Is It Finally Time to Worry about Mobile Malware?," Computer , vol.41, no.5, pp.12-14, May 2008.

[5] PhoneGap. http://phonegap.com, 2012.

[6] V. G. Sarah Allen and L. Lundrigan. "Pro Smartphone Cross-Platform Development" Apress, 2010.