A Report on Mobile App-Stores Applications (Apps) Delivery Mechanism Related Good Practices in Order to Enhance the Apps Security

Authors: Zafar Kazmi & Mario Maawad

Version: 6.0

Date: December 2011

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

Mobile devices, such as the smartphones & tablets are playing an increasingly important role in how people communicate, socialise, and carry out other important daily tasks including accessing their banking activities. Newer network technologies as well as the mobile devices will progressively increase security aspects that would contribute to the wider range of mobile applications and services.

Mobile applications stores (App-Stores) are no exception and the recent popularity of different mobile App-Stores such as the Nokia's Ovi store, Google's Android Market, and Apple's App-Store, is a clear evidence to support the above statement. This further indicates that the access to banking through mobile devices will move from browsers to mobile applications (Apps). Therefore, a great part of the security will depend on how secure is the actual App-Stores as that is where a user would download their banking App from, in order to access their mobile banking & other financial activities.

According to an article published on 21st February 2011 by Minda Zetlin, a number of Android users, in the year 2010, downloaded different mobile banking Apps from the Google's Android Market at a cost of $1.50 each. The Apps enabled the users to connect with about 40 major banks, including the Bank of America and the Wells Fargo of the United States. It later appeared that the banks did not upload those Apps and that those Apps were in fact submitted by some unknown fraudsters to different App-Stores, seeking to only make $1.50 from each download. Needless to mention the potential threat of the fraudsters being able to steal the users' banking login credentials, as a result of the users downloading those malicious banking Apps. It was also the reason why many banks asked their customers to actually have their mobile service provider remove those malicious Apps from their mobile devices.

The purpose of this report is to outline a proposal on common policies for mobile Apps management mechanisms for different mobile applications stores (App-Stores). Although, the main audiences of this study include, the Mobile Financial Services (MFS) providers such as the banking and other financial institutions providing mobile banking & mobile banking Apps as a result of that, and other mobile Apps publishers including the third party publishers, this study could also assist the official mobile App-Stores in enhancing the Apps security mechanism, along with any other related elements of the mobile Apps ecosystem, concerned about providing measures against malicious Apps.

This document will therefore best serve different financial institutions providing mobile banking and developing/distributing mobile Apps as a result of that and any other entities interested or concerned about the Apps security. Furthermore, it would be a great achievement and a valuable contribution to the mobile financial ecosystem, if the findings of this document could assist in developing best practices and guidelines for mobile Apps management & monitoring in order to enhance mobile Apps security and ultimately, securing an important aspect of the Mobile Financial Services (MFS).

In order to develop the aforementioned proposal, this study will aim to assess different mobile applications stores in order to establish an understanding of their current Apps delivery/monitoring mechanisms. This includes the current procedures involved in uploading, downloading, and upgrading the Apps to/from an App-Store along with the current reporting mechanisms deployed by different App-Stores for the malicious Apps reporting.

This document will start off by briefly assessing the major mobile App-Stores, namely, Nokia' Ovi store, Android Market, Samsung Apps, Windows Phone Marketplace, Apple App store, Blackberry App World, which will assist in carrying out a comparison of different App-Stores in the following section. Furthermore, a proposal of common policies/guidelines for Apps management will be presented in the third section of this document before concluding this study in the section four of this report. Finally, due to the fast evolving nature this study, this report is intended to be continuously reviewed & updated as and when needed.

2. MOBILE DEVICES APPLICATIONS (Apps) STORES

2.1. Nokia's Ovi Store

According to Nokia, the Symbian Signed is an umbrella process for properly signing application installers during development and for release. In order to make this work, the Symbian Signed root certificate is preinstalled onto Symbian OS based devices by the device manufacturer. Although, the use of Symbian Signed is not required of manufacturers taking advantage of the Symbian OS platform, majority of them do implement it, in order to take advantage of the ecosystem of signed applications. Furthermore, the Symbian Signed program is divided into the following four categories: OpenSigned Online, Open Signed Offline, Express Signed, and Certified Signed.

2.1.1. Applications distribution

Every publisher is passed through a review process prior to their content proceeding through the system. "Once they have been approved, a developer's content passes through a moderation process which looks at each content item and evaluates it against Nokia's content guidelines. After each content item passes the moderation step, it then proceeds through Nokia's quality assurance process, which runs a set of test cases on the targeted devices according to the content type."

Prior to submittal, all application providers are required to have their applications certified by one of the approved testing processes—for example, Symbian applications must be Symbian Signed or Symbian Express Signed. Java applications must be certified by Java Verified, Thawte or VeriSign.

2.1.2. Application QA

A formal quality assurance process of an App is carried out by Nokia once the App is submitted to the store. The App must conform to the general legal, country and language properties which were chosen during the submission process. More importantly, the App is tested on the target devices that were chosen during submission. When choosing the supported devices for an App, it is required that one must choose the level of compatibility of that App on the chosen device. Possible compatibility levels are: Fully tested, Briefly tested, Assumed to work, Might work, Not compatible and Not known.

2.1.3. Ovi Store Apps Signing

Ovi publishers can request to have certain apps signed for free by Nokia, reducing the turnaround time to two weeks and eliminating the signing costs. Qt, Symbian C++, Java, and Adobe Flash Lite apps can be signed by Nokia. The following flowchart outlines the process of Nokia signing Symbian & Qt Apps

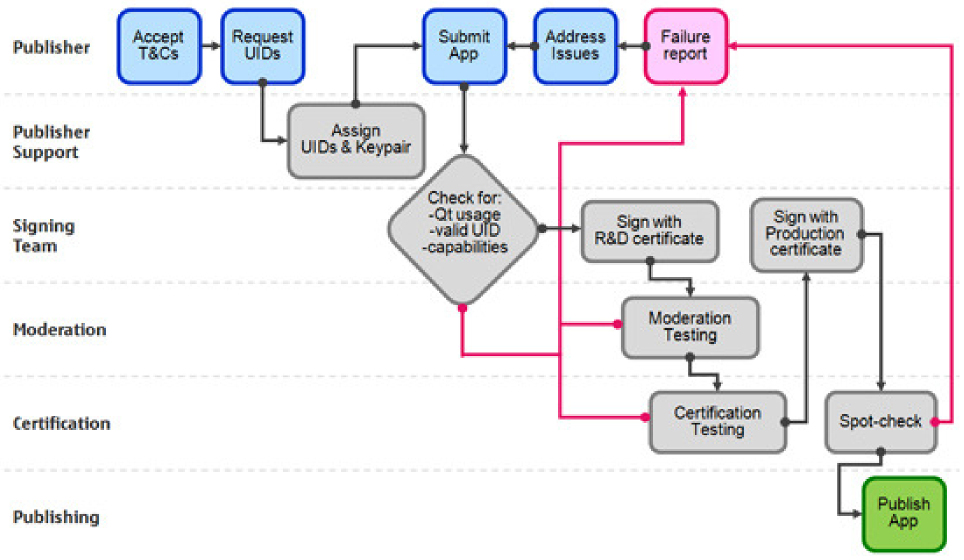

Figure 1 - A Process of Nokia signing Symbian and Qt apps (source: Nokia Developer)

The publication flow illustrated in the above figure is described below (*source: Nokia Developer*):

1. The publisher applies for an Ovi Publish account and accepts the latest Ovi Store Registration and Distribution agreement.
2. The publisher also provides up to five phone IMEI's (International Mobile Equipment Identities) to be used to create a certificate installer for the Publisher's phones to Ovi Publisher Support.
3. Ovi Publisher Support sends the Publisher a list of UID's, a certificate installer, and a developer certificate/key pair for testing a qualified app. Additional UID's can be requested at any time as needed by contacting Ovi Publisher Support.
4. The publisher packages the app's unsigned SIS file using the UID provided and tests it on a phone, being sure to test it against the Symbian Signed Test Criteria.

5. The publisher submits the app (the unsigned SIS file with the UID provided) via the Ovi Publish tool.
6. Ovi Store quality assurance (QA) tests the app based on Nokia content and store guidelines, specific operator guidelines, and the Symbian Signed Test Criteria. If it passes, the app will be express-signed by Nokia and published in Ovi Store.

It's important to note that this arrangement covers only those SIS files that would typically be express-signed. SIS files that require certified signing will not be signed by Nokia; publishers will need to have them signed via third parties.

2.2. Android Market

The Android Market, an online applications store, is developed by Google for Android devices. It gives developers an easy way to distribute applications to Android users. Key features of Android Market include:

- Open: Android Market is open to any developer.
- Simplicity: Join Market in 3 easy steps: Register, upload, and publish.
- Community: Android users can rate and comment on applications.
- Choice: Choose between distributing free and paid applications.
- Manage: Your application portfolio: view downloads, ratings, and comments.

Publishing an application means testing it, packaging it appropriately, and making it available to the users of Android based mobile devices. Before publishing an application, there are several things a developer should do in order to get the application ready. The following figure outlines the process for preparing an application for a successful release:

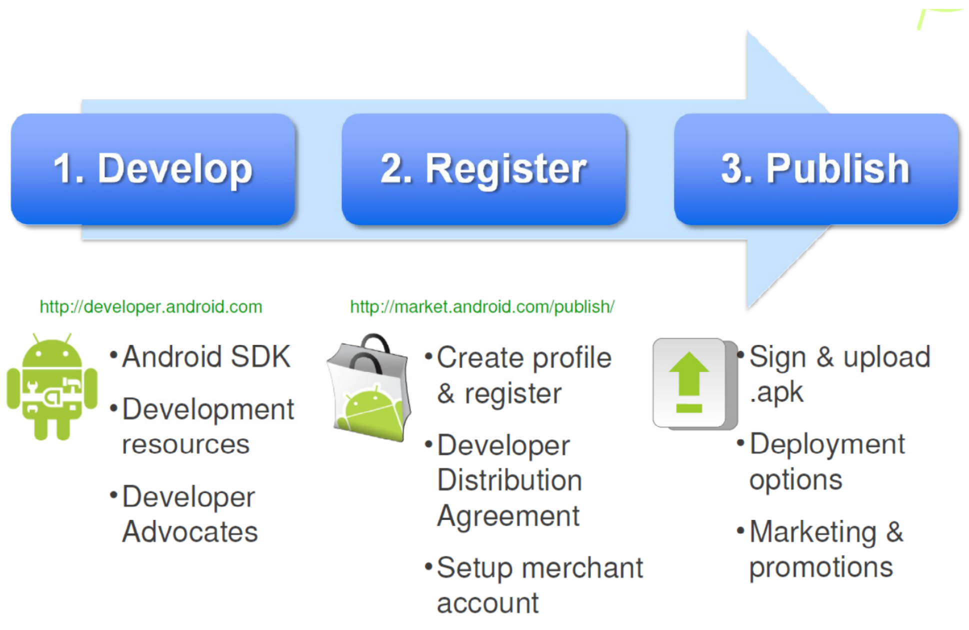

Figure 2 - Android applications development and publishing steps (source: Android developers)

Signing an Android application

The Android system requires that all installed applications be digitally signed with a certificate whose private key is held by the application's developer. The Android system uses the certificate as a means of identifying the author of an application and establishing trust relationships between applications. The certificate is not used to control which applications the user can install. The certificate does not need to be signed by a certificate authority: it is perfectly allowable, and typical, for Android applications to use self-signed certificates. This is true for a developer emulator and personal testing devices, but it’s especially true for programs that a developer wants to publish on the Market.

The important points to understand about signing Android applications are (source: Android developers):

- *All applications must be signed. The system will not install an application that is not signed.*
- *A developer can use self-signed certificates to sign applications. No certificate authority is needed.*
- *When developers are ready to release applications for end-users, they must sign them with a suitable private key. A developer cannot publish an application that is signed with the debug key generated by the SDK tools.*

- *The system tests a signer certificate's expiration date only at install time. If an application's signer certificate expires after the application is installed, the application will continue to function normally.*
- *Developers can use standard tools — Keytool and Jarsigner — to generate keys and sign your application .apk files.*
- *Once you the application have been signed, it is recommended to use the zipalign tool to optimize the final APK package.*

Publishing

The Android Market is a Google-hosted service that can be used for posting android applications. To get started with publishing a developer first need to sign up as a registered developer on the publisher's web site (also known as the Developer Console). There is a small registration fee. As an additional step, if the developer wants to charge for his applications he'll need to sign up with a payment processor. The publisher's web site will instruct developers on how to do that. As of this writing, only Google Checkout is supported, but in the future other processors like PayPal may be supported.

Once ready to upload an application to the market there are certain additional recommendations to follow:

- Turn Copy Protection off. Android's copy protection is completely insecure and performs no useful function other than to irritate your users.
- Unless you have a reason not to, set the Locations option to "All Current and Future Countries." New countries are being added all the time, and this way your application will be available to all of them.
- Do not supply a phone number in the application Contact Information. All Market users will be able to see this number and they will call it when they have problems. Of course, if you have a dedicated number and staff for phone support, then this tip doesn't apply.

Once the publishing process is finished (it just take a few steps), the application will appear immediately on the Android Market. There is no approval process, no waiting period (other than a few seconds to update the download servers), and no limits to what you can do in your programs.

2.2.1. Android Market licensing overview[1]

[1] http://developer.android.com/guide/publishing/licensing.html

Android Market Licensing is a network-based service that lets an application on an Android-powered device query a trusted licensing server, to determine whether the application is licensed to the current device user. After receiving the server response, the application can then allow or disallow further use of the application as needed. In the service, the role of the licensing server is to provide the license status for the current user; the application itself is responsible for querying the server and conditionally granting access to the application.

2.2.1.1. Application, Android Market client, and server

The licensing service is based on the capability of the Android Market server to determine whether a given user is licensed to use a given application. The server considers a user licensed if the user is recorded to have purchased the application, or if the application is available for free. To properly identify the user and determine the license status, the server requires information about the application and user — the application and the Android Market client work together to assemble the information and pass it to the server.

In the licensing service, an application does not query the licensing server directly, but instead calls the Android Market client over remote IPC to initiate a license request. In the license request:

- The application provides its package name and a nonce that is later used to validate any response from the server, as well as a call-back over which the response can be returned asynchronously.
- The Android Market client, which has greater permissions than the application, collects the necessary information about the user and the device, such as the device's primary Google account username, IMSI, and other information. It then sends the license check request to the server on behalf of the application.

The server evaluates the request using all available information, attempting to establish the user's identity to a sufficient level of confidence. The server then checks the user identity against purchase records for the application and returns a license response, which the Android Market client returns to the application over the IPC call-back.
Notice that during a license check, the application does not manage any network connections or use any licensing related APIs in the Android platform.

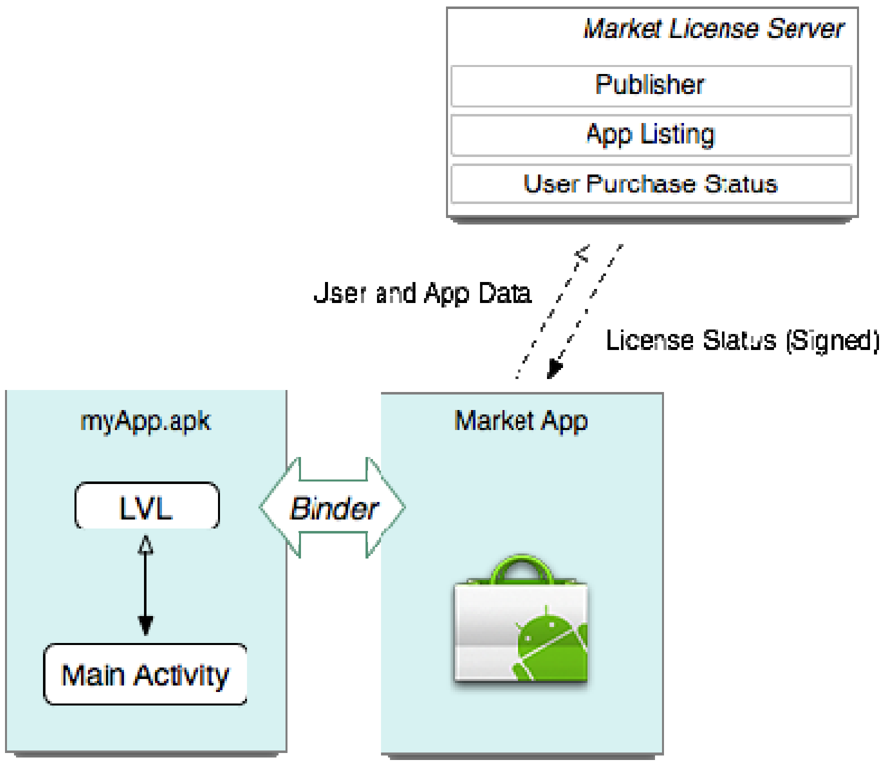

Figure 3 - Application license check through the LVL and the Android Market client (source: Android developers)

2.2.1.2. License responses secured through public key cryptography

To ensure the integrity of each license query, the server signs the license response data using an RSA key pair that is shared exclusively between the server and the application publisher.

The licensing service generates a single licensing key pair for each publisher account and exposes the public key in the account's profile page. The publisher copies the public key and embeds it in the application source code, then compiles and publishes the .apk. The server retains the private key internally and uses it to sign license responses for applications published on that account.

When the application receives a signed response, it uses the embedded public key to verify the data. The use of public key cryptography in the licensing service makes it possible for the application to detect responses that have been tampered with or that are spoofed.

2.3. Samsung Apps

Samsung Apps is a free to use, open marketplace embedded on every Samsung “smart” device sold. It provides the ecosystem required to actively promote and manage developers’ applications. It was launched in June 2010, and by March 2011 Samsung stated that its app store had already served over 100 million downloads worldwide, less than a year after its launch.

Samsung Apps are being released for Samsung phones and hundreds of application can be found for a variety of platforms, i.e. Windows Mobile, Android, and Samsung’s own platform Bada. In any case, Samsung is a manufacturer of mobile devices that has been supporting a variety of different mobile Operating Systems, from Java ME to Windows Mobile to Android and the Samsung’s own latest development called Bada, to name a few.

The Samsung Android Developer Forum is intended to provide additional technical support and guidance for Android application developers. It provides a single destination for Android developers for Samsung mobile devices, the forum features rich content with news and updates for developers, while lively discussion boards offer unrivalled technical advice and support. The Samsung Android Developer Forum offers development tools to assist developers working on complex projects – including SDK plug-ins for Augmented Reality and Location Based Services.

The service will also offer a Sensor Simulator as well as a Remote Test Lab (RTL), allowing developers to test their applications remotely on a real device controlled by Samsung. According to a statement from Samsung, "*Samsung’s Android Developer Forum enables Android developers all over the world to broaden their scope and achieve even higher levels of quality when designing for Samsung’s Android-based devices, while customers will continue experiencing high-standard Android applications available from Samsung Apps*”.

Bada applications publication process is well documented and although it is not one of the main runners in the mobile OS market as yet, Samsung is pushing strongly into the market to make this system the dominant one, at least in the featured phone market segment. The process through which a third party developer should pass in order to make their App publicly available on the Samsung Apps store in summarized in Figure 5:

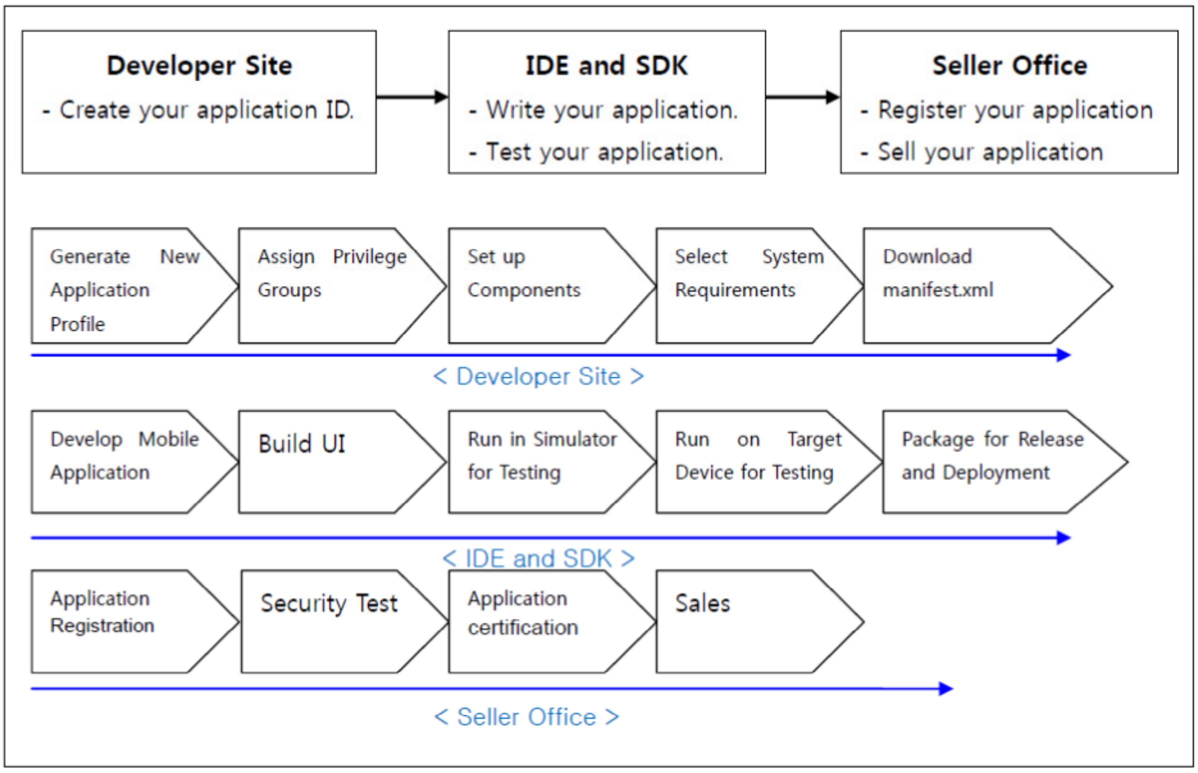
Developer Site
- Create your application ID.
IDE and SDK
- Write your application.
- Test your application.
Seller Office
- Register your application
- Sell your application
Generate New Application Profile
Assign Privilege Groups
Set up Components
Select System Requirements
Download manifest.xml
< Developer Site >
Develop Mobile Application
Build UI
Run in Simulator for Testing
Run on Target Device for Testing
Package for Release and Deployment
< IDE and SDK >
Application Registration
Security Test
Application certification
Sales
< Seller Office >

Figure 4 - Samsung Bada Application Development - Registration Procedures

2.4. Windows Phone Marketplace

Windows Phone Marketplace is a service by Microsoft for its Windows Phone platform that allows users to browse and download applications that have been developed by third-parties. The Windows Phone Marketplace was launched along with Windows Phone 7 in Oct 2010 in some countries. It was reported on October 4, 2010 that the Windows Phone SDK has been downloaded over half a million times. As of July 1, 2011, the Marketplace had more than 25,000 apps available.

Microsoft's philosophy is that an app marketplace should balance quality, choice and variety with a great customer experience – which includes easy shopping and discoverability. It is for this reason that Microsoft has defined a whole process through which developers should submit their applications before reaching end user's phones. The main steps of this process are:

- Register as a Windows Phone developer
- Develop and test applications
- Assemble the prerequisites for certification
- Submit applications for certification
- Link applications in the Windows Phone Marketplace catalogue

- Updating application in the Windows Phone marketplace
- Support

When an application is ready for publication, it must go through the certification process before it is eligible for listing in Windows Phone Marketplace. An application does not have to be signed before submission. The certification process involves static validation and automated testing of the application to verify that it meets all the policies and requirements. The following figure is a simplified illustration of the submission and certification process.

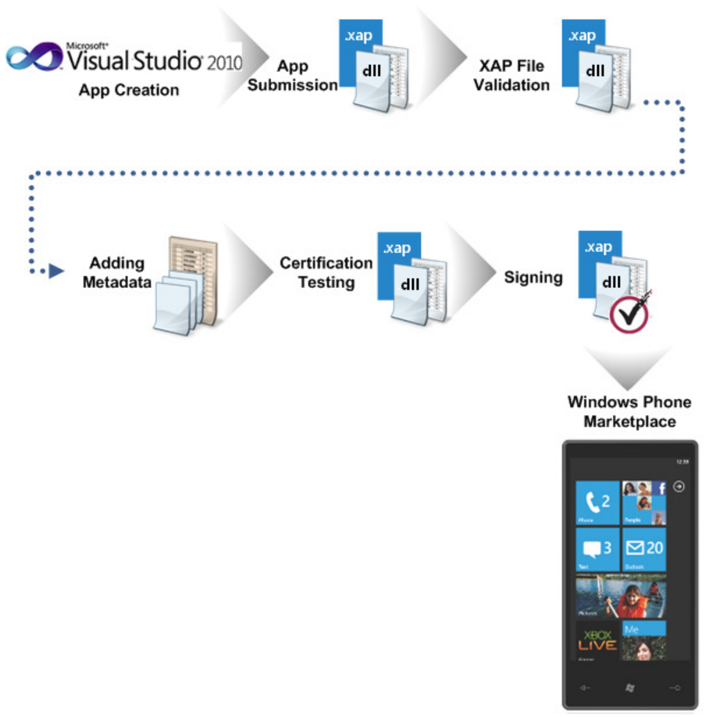

Figure 5 - Windows Phone application submission process (source: Microsoft MSDN)

The following list shows the five major categories of policies and requirements an application should be conformant with, before its final listing in the Windows Phone Marketplace (source: Microsoft MSDN):

- Application Policies
- Content Policies
- Application Submission Requirements
- Technical Certification Requirements
- Additional Requirements for Specific Application Types

2.5. Apple App Store

On July 10, 2008 via an update to iTunes, Apple released an online market place for applications, called “App Store.” It is a service for the iPhone, iPod touch, and now the iPad. By using this app store, iPhone users can download any apps they want through iTunes or directly from their phones to take advantage of all available iPhone features. As of May 4, 2010, iPhone App Store has more than 195,000 third-party applications with over 4 billion total downloads. The Application Store is basically a centralized collection of all different applications. Currently, it has around 20 different categories which help users select and download the exact application they are looking for. The price of these applications, free or paid, is decided by the developer of a particular application. The figure below shows a third-party development model for any online application marketplace.

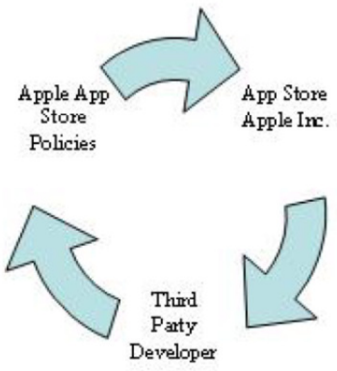

Figure 6 - App Store (Third Party Development Model)

Once the developer decides to build an application for the iPhone, he/she must consider and follow something called the iPhone Developers Program. This program has 3 main contents namely: (1) Develop, (2) Test, and (3) Distribute (“iPhone Developer Program,” n.d.)

- **Develop:** This includes all the development tools and resources needed for building an application using the iPhone’s SDK. (Software Development Kit) Basically, the development tools include: Xcode, iPhone Simulator, Instruments, and Interface Builder. The development resources provide developers an access to the Apple developer forums, getting started videos & documents, iPhone reference library, and coding resources. Development environment preparation (considering also security requirements have been reviewed in the previous section).
- **Test:** This allows developers to test their applications to see how they will perform in the real environment. This program also includes two technical support incidents: (I) Apple engineers provide the developers with code-level assistance and helpful guidance which would help the developers (II) Directions for the issues that developers are facing. This includes test in real-time; test over-the-air and technical Support.

- **Distribute:** Apple has two distribution channels: App Store Distribution and Ad Hoc Distribution. These are the only way users can get access to the applications. Users can download their applications using an iPad, iPhone, or iPod touch via the iTunes store. With App Store Distribution, any app developed and published on the app store can reach millions of customers. While with the Ad Hoc Distribution, the application can install on up to 100 devices without going through the App Store. Developers are forced to use 'Ad Hoc' distribution so that apps can be tested prior to submission to the iTunes App store (Ben, 2008). The Apple method requires pre-registration of beta testers. In order to register them, developers need to know Unique Device Identifier (UDID) for their devices and then create a mobile provisioning file - a security certificate that authorizes that unique device to run the app being tested. (Ben, 2008)

Apple uses the 70/30 model for splitting the revenue generated from the App Store, between the developer and itself. The App Store has helped 3rd party developers to solely focus on developing creative and innovative applications and not worry about its distribution process. The marketing of the applications is handled by the App Store which serves as the distribution channel for these apps. Thus, Apple has created a huge demand for the platform as well as the App Store (Bill, 2007).

2.5.1. Application distribution

Apple's Application Store was the first user experience that made acquiring mobile applications a simple task. Apple exercises total control over the content of the Application Store; all iPhone applications must be approved prior to distribution, and can be revoked at Apple's discretion. The rules for forbidden applications are something of a moving target, and therefore are difficult to enumerate. However, with the advent of iPhone OS 3.0, Apple seems to be making moves to loosen these restrictions.

Although there are many disadvantages to this approach, it does mean that the App Store serves as a security boundary. Programmatic security mechanisms for applications running on the iPhone OS are fairly lax, but Apple can rein in developers behaving in a deceptive or malicious fashion by refusing to publish their applications or revoking them from the store.

In the event actively malicious software does make it through the App Store, a second approach, albeit one that has yet to be used by Apple, is a "kill switch" allowing the blacklisting of applications after install. Although there were some initial worries that Apple would be using this to actively disable software it didn't like, this has not been the case to date. Thus far, using Apple itself as the gatekeeper and security boundary for iPhone applications has had good results—except for developers, some of whom have had wait times of up to six months just to get an application accepted. With the sheer number of new iPhone applications appearing on a daily basis, it's not unlikely that more OS-based security features will be used more actively in the future to smooth out the application-vetting process.

2.6. Blackberry App World

Just like all other mobile devices manufactures, BlackBerry has also launched its own Applications Distribution Service which is called the Balckberry App World. It *in over 70* countries and supports 21 currencies and six languages, and comes pre-loaded on new BlackBerry smartphones. BlackBerry App World helps developers to:

- Increase monetization opportunities with in-app advertising and the ability to sell digital goods
- Understand the real-time status of applications submissions by providing a transparent submission process
- Increase the discoverability of apps with a range of 'Top 25' lists that end users can search on both the mobile and desktop storefront versions of BlackBerry App World
- Make it easy for end users to purchase apps and digital goods by providing popular payment methods like PayPal, credit card and direct to carrier billing (carrier dependent)

The architecture developed for RIM for its App Store, and distribution process is illustrated in the following figure:

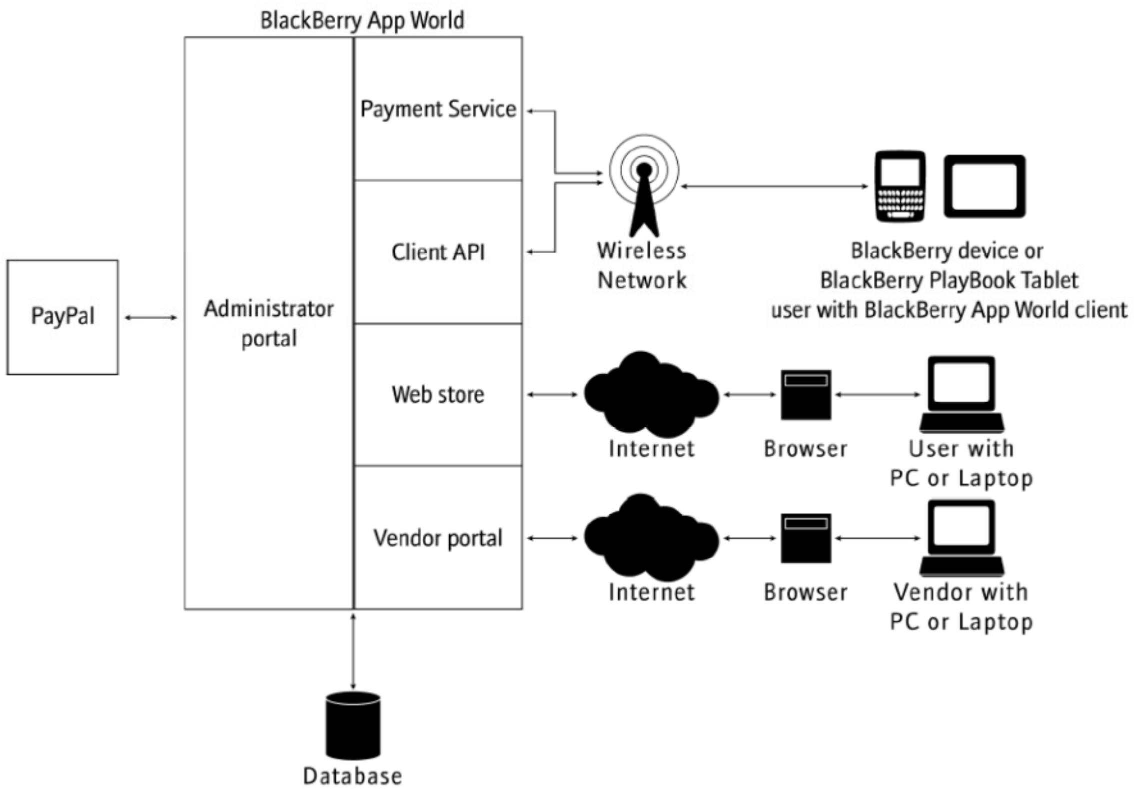

Figure 7 - Blackberry App World Architecture (source: Blackberry App World Distribution)

2.6.1. Application distribution

To submit an app or theme to BlackBerry App World, a developer will first need to create a vendor account. Once the account credentials have been confirmed, a developer will receive instructions on how to begin submitting apps or themes. When submitting an app, it is required to include the following:

- Application name, description and icon or logo
- Category in which your app should be placed
- License type (free, paid or try and buy)
- Wireless service providers your app will be available to
- Countries where your app should be distributed
- Releases and file bundles
- Screenshots

2.6.1.1. Process flow: Applying for membership

1. A vendor visits a sign-up to the BlackBerry App World™ storefront through a web portal link (for example, at www.blackberry.com/developers/appworld)

2. A vendor should provide PayPal account details. PayPal validates the PayPal account information.
3. RIM accepts or rejects the membership request and notifies vendor. If RIM accepts the membership request, the vendor can start adding products to BlackBerry App World. If

2.6.1.2. Process flow: Adding a product to the App World

To add a product and a digital good to the BlackBerry App World™ storefront, a vendor logs in to the vendor web site for BlackBerry App World and clicks Manage products.

1. The vendor provides information about the product and the digital good.
2. Research In Motion verifies that the product conforms to RIM standards and performs testing on the product.
3. RIM accepts or rejects the product and notifies the vendor.
4. The vendor makes the product available on BlackBerry App World.

2.6.1.3. Process flow: Adding a release

To add a release for a product to the BlackBerry App World storefront, a vendor logs in to the vendor web site for BlackBerry App World and clicks Manage products.

1. The vendor provides information about the release.
2. Research in Motion verifies that the release conforms to RIM standards and performs testing on the release.
3. RIM accepts or rejects the release. If RIM accepts the release, it is included with the product.

3. A REVIEW OF DIFFERENT APP-STORES Apps MANAGEMENT MECHANISMS

There are a number of unique challenges and pieces to application security that most devices currently, are not able to provide. Similarly, all major mobile platforms bear different Apps delivery mechanisms & hence the associated risks also differ to a certain extent. One of the common major tasks however, for all mobile Apps platforms/providers is to ensure the Apps authenticity and a secure management/delivery mechanism for all Apps available on their App-Stores.

The differences between the capabilities of platforms have a significant impact on the management of their security. Managing the Apps delivery mechanism is all about the capabilities of a mobile devices platform/App-Store to monitor/authenticate the legality of an App's source. Being able to keep a track/record of where an App came from can be vital to deploy appropriate countermeasures against a malicious or a fraudulent App.

Furthermore, the rapid changes in this largely consumer driven mobile devices market mean that code is quickly written, deployed and replaced or even upgraded. Development platforms that support the writing of a secure code are currently lacking for the mobile devices. This is particularly case for the Operating Systems which are often written in either "C" or other native languages leaving security totally at the discretion of the developer.

Therefore, balancing the restrictions imposed by application delivery mechanisms while ensuring an acceptable level of versatility and usability of the smart phone is a challenge for the Apps providers/vendors. Some vendors provide for encoded signatures on applications and some restrict applications to a single controlled source while others have no restrictions on the source of an application.

The following chart outlines the Apps delivery security mechanisms which are put in place by different mobile platforms:

Platform	Signing	Revocation	Approval
Symbian	Signed by Vendor	Yes	Quality
Android	Anonymous, self-signed	Yes	No
iOS	Signed by Vendor	Yes	Policy & Quality
Windows Phone	Signed by Vendor	Yes	Policy, Quality & Security
Blackberry	Signed with Vendor issued key	Yes	No

Table 1 - A Chart Outlining the Apps Signing, Revocation, & Approval Procedures for Different Mobile App-stores (Source: Veracode)

As it can be gathered from the above chat, all mobile platforms have some sort of App signing mechanism in place. The mobile OS will not allow Apps that are not signed to execute with an exception of the “jail-broken” or “rooted” OS’s which are in fact “jail-broken or rooted” by the users specifically to allow unsigned apps to be executed. Furthermore, the above chart also confirms that all mobile platforms support “revocation” in order to remove malicious Apps, once detected.

Depending on the implementation of the signing mechanism, it can be a huge achievement in terms of improved security. For instance, if the App is signed by the developer with a self-generated key, there is little security gain but if the application is signed by a key issued by the platform provider then there will be a security benefit based on the policies the platform provider adheres to, for approving Apps. Moreover, it is important to stress that the mobile jail-breaking removes the security benefits of the platform signing mechanism altogether.

Android's Market App store for instance, is probably not up to the challenge when it comes to malicious Apps publishing and distribution. This is evident in the ease with which malicious Apps can be uploaded & distributed on the Android Market. Those malicious Apps can access the sensitive OS resources such as the text messages, mobile device location via GPS, camera, voice recording, to name a few. It can therefore be stated that developing, publishing & distributing a powerful fraudulent Android App which could steal one’s personal information including their financial information is almost trivial as the current security measures deployed around the App submission process by Android are inadequate to identify and prevent submission of malicious applications to the Android Market.

The Apple App store on the other hand, appears to be “walled garden” and it does employ an approval process but the details of its approval process are not well documented or publicised. It is therefore difficult to establish exactly what sort of details the iOS Security Team at the Apple App store takes into consideration, when it comes to an App approval or screening. Having said that, it is quite clear, based on the deployment of the aforementioned approval process, that the Apple App store is putting a lot of effort in ensuring the user experiences and the compliant to its policies.

According to a recent report from Symantec, Apple and Google are very different when it comes to mobile security, "creating distinct potential vulnerabilities for enterprises embracing devices running these operating systems." Apple employs an “Application Provenance” strategy which basically involves identifying, certifying and vetting an App before it is published on to the Apple App store. For Android Market on the other hand, the course of action is completely different as there is no vetting process as there are far more self-signed applications and the Apps can be uploaded from just about anywhere on the internet. So although these user-friendly Apps are continuously getting more and more popular amongst the users, these are also bringing new security threats and breaches with them.

One of the proposed solutions could be that the Android apps are not only self-signed but are also signed by certified keys issued by the trusted authorities. This strategy on its own may not prevent malicious Apps but it would certainly assist in deploying a tracing mechanism which would allow tracking down the fraudulent App owner/publisher.

As far as the malicious/fraudulent App reporting procedures are concerned, these differ from each app-store and the most common issue is that the reporting procedures are not clearly defined by the App-stores. The same goes for the removal/take-down procedures of a malicious App. The Google’s Android Market for instance, does have an option to report a malicious/fraudulent App on its website, which can be used to request Android Market to review and remove an inappropriate App from its store but it could become a difficult task for an end user to find the “reporting option” on Android Market website.

Furthermore, there have been a number of instances whereby a malicious App was reported but it took a while until that App was taken down by the Android Market. Therefore, it can be stated that the process of identifying/reporting and removing/take-down a malicious App from the Android Market requires serious considerations and perhaps improvements as a result of those considerations.

In the case of Nokia’s Ovi Store, there is a revocation process in place for malicious Apps. Revocation provides a final sanction to handle applications which may pose a threat to the Nokia users, its network or the Nokia’s Ovi community as a whole to ensure those malicious Apps do not spread any further.

Based on the findings of this study, the next section of this document will aim to propose common policies on Apps management in order to improve security of the mobile Apps ecosystem. These policies therefore, could be considered or even implemented by different Mobile Financial Services (MFS) providers such as the banking and other financial institutions providing mobile banking & mobile banking Apps as a result of that, and other mobile Apps publishers including the major App-Stores & third party publishers.

4. A PROPOSAL OF COMMON POLICIES ON MOBILE Apps MANAGEMENT

1. While conducting a security review of the submitted Apps, the App-Stores should consider a number of different known threats associated to the mobile Apps. These threats could include but are not limited to:

    - Activity monitoring and data retrieval
    - Unauthorized premium rates dialling & premium rates SMS
    - Mobile banking & mobile payments related frauds (since the mobile banking Apps are increasingly being uploaded & downloaded)
    - Unauthorized network connectivity
    - UI (Unique Identifier) impersonation
    - Sensitive data leakage
    - Unsafe sensitive data transmission

2. The App-Stores should increase the user awareness regarding malicious Apps & the consequences of downloading such Apps by introducing a number of different measures including an informative message, clearly visible on the App-Stores' websites. This should also be the case when it comes to the reporting procedures so that a user could follow simple steps in order to report malicious Apps to the relevant App-Store.

3. An introduction of a warning message for the Apps developers/authors could also be introduced, stating that a malicious App upload will lead to their accounts being deleted permanently and possibility of their details being passed onto the law enforcement agencies, in case of a criminal activity as a result of their App being downloaded by the users.

4. All App-Stores should look into placing a quality framework in the form of a series of standard security tests which could issue health certificates for different Apps in order to combat the growing threat of fraudulent Apps.

5. Apps Isolation - A strong security mechanism for ensuring mobile Apps access permissions could be introduced by the App-Stores, since it would not be sufficient to

allow an App access to certain functions as the functionality needs to be further restricted by type, by time and conditions.

6. Banking and other Mobile Financial Services related Apps should only be allowed to be published by the banks & related financial institutions, ensuring their credentials are fully verified prior to submission of those Apps.

7. In order to improve the security, the App-Stores could carry out a pre-screening of the Apps and could also deploy a continuous monitoring procedure to ensure that the Apps which have been upgraded could also be screened/monitored.

8. A comprehensive vetting mechanism for Apps developers/authors could be deployed for all app-stores to ensure a successful tracking-down of the owner of the App in case of an App turning out to be a malicious App. For instance, all Apps publishers must have an account with the appropriate app store before they could submit an App to the app-store. Also, the app-stores should verify those publishers' accounts by emails, postal addresses, telephone numbers, and also through their credit card details.

9. All App-Stores should consider deploying some sort of Apps and Apps developers/authors reviewing procedures including the review counters, which would automatically raise the alarm in case of a negative review. Perhaps an Apps auto-suspension procedure could also be deployed in case of reaching so many numbers of negative reviews (e.g. 5), until the App is investigated by the app-store security team. A serious consideration should also be given to monitor how those feedbacks were being submitted in order to ensure that the fraudsters could not abuse the review process.

10. An App review rating from one App-Store could also be used by other App-Stores through some sort of a shared reviewing mechanism implemented jointly by all app-stores. One of the most serious concerns here could be that the most users would rate Apps for their actual functionality and not for their security aspects. It would therefore be important to have two separate categories for the App reviewing mechanism, one for security and privacy issues, which could include an App asking for excessive privileges at install, and the other for the general functionality issues of

an App, which could include information such as the App worked as it was supposed to, etc.

11. A continuous monitoring of the Apps developers/authors in a way that the prior Apps contribution should not be taken into account and each new submitted App is considered on its own basis.

12. The malicious Apps reporting procedures should be well defined and easily visible (highlighted) & accessible to the users of an App-Store. One of the options could be a simple reporting form in order to submit a complaint which should be dealt with and resolved, as quickly as possible. Another suggestion could be a live chat channel in place on all App-Stores so that a user who wishes to report a malicious App could do so, as and when needed.

13. A simple but effective & appropriate, "take-down" or "removal" procedure for all different App-Stores could be implemented. Perhaps a unified "shared" take-down mechanism could be employed throughout different App-Stores in order to ensure the malicious App publisher gets barred from all stores.

14. All App-Stores should have a kill-switch to remotely kill a malicious App upon reporting/discovery. A generic policy & procedure on "remote wipe" could also be implemented in case of a malicious App disaster ensuring that the users are fully aware of those procedures.

15. All App-Stores should consider priority vetting for updates of existing Apps in order to enable the Apps developers/publishers to patch up the vulnerabilities quickly and effectively.

16. A special emphasis could be given to the free Apps and perhaps all App-Stores could jointly create and publish a list of all "free" Apps that are submitted to their stores.
17. A knowledge share mechanism between different App-Stores could be deployed in order to report and monitor a fraudulent developer's/author's activities.

18. A generic (for all App-Stores), mobile devices Apps downloading & user's best practices could be developed in order to ensure an increased awareness for the end users.

5. CONCLUSIONS

Mobile Financial Services related Applications such as the banking Apps, are becoming the wave of the future and these must be developed, managed & monitored with a great consideration, in order to ensure the Apps security. It is therefore, a huge responsibility of different Apps publishers & the App-Stores to develop and maintain an up-to-date Apps delivery security mechanism.

Furthermore, when developing a mobile banking application, the Apps developers should not only consider the application's functionality and usability but must also take into account the security aspects of their application.

Similarly, the App-Stores must also ensure they take into account, the current security issues including the known threats and vulnerabilities when permitting an App to be published on their stores.