A Secure Use of Mobile Application with Cloud Service

Kazuaki Nimura, Hidenobu Ito, Yousuke Nakamura, Kouich Yasaki
Fujitsu Laboratories Ltd.
1-1, Kamikodanaka 4-chome, Nakahara-ku,
Kawasaki, Japan
{kazuaki.nimura, itou.hidenobu, nkmr, yasaki.kouichi} @jp.fujitsu.com

ABSTRACT
In today's rapidly expanding world of smartphones and smart tablets, corporations have found commercial applications for these devices in their business model. The use of BYOD (Bring Your Own Device) is an attractive option for corporations as it maximizes cost efficiency. This however raises security issues with the separation of corporate and personal data on the device. In this paper, we propose a mechanism for secure storage of encrypted data and applications on a BYOD smart device. This is achieved by a cloud service sending encrypted applications and data (assets) to the device. The assets can then be executed in a sandbox and with the assurance that sensitive information will not be left unencrypted on the device. This protection by application context is achieved by the use of context specific application groups. A cloud service is used to control critical assets on the device, so a change of context may be used to trigger the erasing of assets. The system proposed is implemented using a HTML5 (Hyper Text Markup Language 5) hybrid application on an Android device. This paper will evaluate the system and demonstrate its feasibility.

Categories and Subject Descriptors
H.3.4 [**Systems and Software**]: Distributed systems

General Terms
Management

Keywords
Smart device, Application management, HTML5, Push, Security

1. INTRODUCTION
The use of Android based smart devices is expanding rapidly. For a corporate device security concerns [1] can be addressed by the use of anti-virus software and a MDM (Mobile Device Management) product to enforce corporate security policy. The BYOD scenario, where employees use their own device for work, is attractive to business because of potential cost savings. Consequently this scenario is being investigated by companies as part of their corporate IT strategy. One of the primary concerns is the mixing of corporate and personal data.

One solution is the use of mobile virtualization technology which uses multiple operating systems on the device to keep corporate and personal assets separate. These however require more device resources in order to function. Typically, using this method, a device will connect to corporate network and use a thin client to process corporate data. However this system cannot be used without network connection and the performance of the system is reliant upon the network speed which may fluctuate. So in certain situations the user could experience significant performance degradation when compared to a system where the data exists locally.

This is the motivation behind enterprises seeking a better BYOD solution which will protect corporate assets whilst maintaining the efficiency of a dedicated device where data is stored locally. In order to assess the available options we need to briefly examine the types of mobile application available.

1.1 Mobile Application Types
Smartphone Applications can be categorized into three basic types: native, web and hybrid mobile applications [2].

- Native applications are commonly used on smart devices, dedicated versions of which are developed for their own particular platforms. The need to develop a separate native application for each platform has the disadvantage of extra development costs in order to support existing corporate assets.
- Web applications are downloaded from the internet and execute in the device browser with no need to install applications on to the device to execute. Web applications have the advantage of already being used on many corporate systems; Set against this is the disadvantage of requiring network access in order to function. Network downtime can prove expensive with employees unable to work efficiently.
- Hybrid mobile applications employ web technologies and offer cross-platform compatibility not available with pure native applications [3]. They can also be distributed via the usual application stores. HTML5 [4] and the ability to use local storage[5] has the advantages of allowing applications to still function when network is unavailable, reducing network traffic and offering better response times than pure web based applications.

  This leads to the conclusion that hybrid applications may be the most suitable choice for a corporation. The system proposed takes the form of a hybrid mobile application.

The proposed system introduces functions for separating each application by encryption and protecting the application by context. The separation of application is performed by encrypting

an application in a cloud service pushing the encrypted application to the device. The application is executed securely by loading to a sandbox; no unencrypted files are visible outside this sandbox. Additionally protection mechanisms for inside and outside the separate environment are also introduced, so that those can mitigate the security issues raised by other applications. The protection by context is achieved by the use of application groups which link a context and applications. These groups are visible on display and will change based on the context of the smart device. In addition, when the context is changed, critical application and data on the smart device are erased. This control over data and applications using cloud service provides maximum protection against information leakage. The proposed system has been implemented as a web service and a hybrid Android application.

In addition implementation of this system does not require modifications to Android OS (Operating System) itself, and so the system may be implemented on any standard device with the web application being executed using a standard Android web control (WebView). Also as the system does not rely on an independent browser we are unaffected by the possibility of periodic updates to the devices browser software and consequent maintenance implications.

The remainder of this paper is organized into the following sections: In section 2, we discuss the related topics. In section 3, we discuss the problem and present our architecture. In section 4, we show the details of the prototype implementation. In section 5, we show the results of the system evaluation, In section 6, we state our conclusions.

2. RELATED TOPICS

This section will explain the related topics.

2.1 WAC

WAC (Wholesale Application Community) is an industry standard body which defines specifications on the ecosystem for the hybrid mobile application [6]. It defines a security framework from the execution environment of the application to the security framework of the ecosystem. Aspects of the WAC specification were used on the proposed solution.

2.2 Phonegap

PhoneGap [7] is a development platform that allows a developer to write a web application and have it compiled into a native application for all the major smartphone platforms. The resulting application can then be downloaded and installed from the relevant platforms marketplace as if it was a normal native application. By using PhoneGap we can provide device access capability that is unavailable for web applications. When installing a native application, user will be prompted to allow install and once confirmed it will install the native application on to the device.

To develop a hybrid mobile application, a tool such as PhoneGap can be used to convert a web-based application into a native application. In the case of PhoneGap style approach, the application is converted outside the smart device and is downloaded on to the smart device. Every modification to an application will require recompilation and reinstallation. In addition the installation phase requires user interaction as the user will be prompted for authorization during installation. The ability to install an application without user prompting is preferred, especially when combined with the concept of application push.

2.3 Application Push

Application push provides a method that can send and run HTML5 applications without user interaction [3]. The system uses a dedicated native application on the smart device that acts as a runtime using PhoneGap for the HTML5 web application. The web application is placed on an application server within a cloud service with the download executed automatically. Once downloaded the application is run-in the runtime environment. This in combination with push technology allows us to push and execute an application as required.

This however raises potential issues with security which will be discussed in the following section.

2.4 Security on Smart Device

Security concerns with the Android platform have been raised beforehand [1]. In order to deal with the security threats the combination of the development of antivirus software and the use by Google of the Bouncer malware blocking system have gone some way to address the issue[8]. The ultimate goal would be an environment that does not need to install anti-virus software [9]. Because smart devices can be easily lost or stolen, remote lock and wipe functionality is commonly offered as part of the MDM (Mobile Device Management) [10]. For BYOD, a method that has several trust levels in the OS and the use of isolated domains is proposed to secure a native application [11].

However analysis of the application push system leads to the conclusion that it cannot rely on any of the above security measures for native applications, it becomes necessary to consider new protection mechanisms.

3. ARCHITECTURE

A BYOD system needs to address the issue of corporate data security, the proposed architecture will address the following issues:

- A separation mechanism for corporate environment and its assets. These needs to be kept in a ‘private’ area on the device.
- Protection of application and data. In the event of the theft or loss of a device corporate assets will need to be protected
- Protection for the separation mechanism. This is necessary because the separation mechanism faces security threats due to the open architecture of the Android platform.

A solution must be found for these issues that has the minimum impact on the users experience. One solution is that assets (applications and/or data) are held separately from the rest of the smart device environment and these assets are erased when not required. Additionally a cloud service understands these assets in terms of the context of a smart device. Figure 1 shows (A) an application on the cloud and a connection path between the cloud and a smart device the application in the smart device is separated from the other applications; (B) Assets within the separate environment need to be secure; and (C) Assets held outside the separate environment also need to be secure.

3.1 Protection of Application

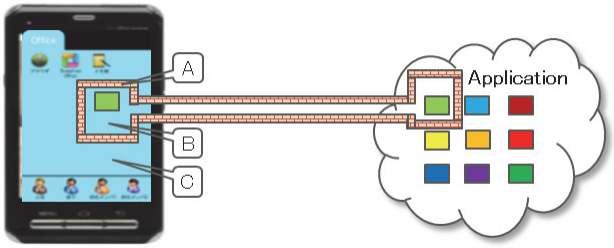

Figure 1. Separation of application environment

An application runtime environment should be separated in order to protect it from other applications (Figure 1. A). this is achieved by the use of two functions: (A-1) separation among applications and (A-2) protection by context.

3.1.1 Separation among Applications

Separation of assets is achieved by using encryption. The encryption is carried out in the cloud service and the encrypted assets are deployed to the application repository. The assets are downloaded on a smart device in encrypted form. When the application is executed it is decrypted on the fly and the decrypted data only held in RAM (Random Access Memory). Therefore the assets are protected and invisible to other applications.

3.1.2 Protection by Context

To avoid complex expressions based on place and time, the concept of a context is introduced, e.g. work and public. Figure 2 shows this idea. The work tabs contain the work environment and public tabs contain the personal and public environment. This simplification is introduced because the security policy is usually dependent on the location of the device i.e. whether it is in or out of company premises. Applications placed on work context should not be used in public context and vice versa. When the context of smart device changes, the access control for the context group will be switched and only the applications tied to current context can be used. It is also plausible to define only 1 context using this approach, e.g. work and then presume the native environment is the personal one.

To summarize, when both (A-1) and (A-2) are combined:

- Application group is linked to context.
- All applications are protected and assigned to an application group or shared among application groups
- Access control is enforced for each application group, and this access control is applied to every application. In other words, a context and an access control are interlocked.
- When context changes, the access control to the previous application group is switched to protected mode and the selected application group is released from the protection and enforcement of access control.

Consider the following scenario: One desktop is assigned to a work context and a second desktop is assigned to all other contexts. The assets that can be used in each context are placed on their desktops, the desktop changes automatically depending on the location of the device. So, for example, if the device moved into a corporate location the work desktop would be displayed and the public desktop hidden. To confirm location information is trusted, it is necessary to check a device ID or a device certificate

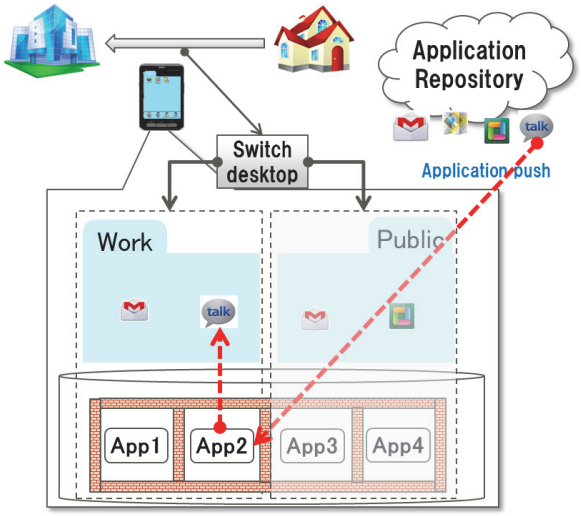

Figure 2. Context and separation

that ties to a location device before sending the information.

3.2 Securing the Separated Environment

Internally the separated environment must be secure (Figure 1. B). this is achieved by the use of three functions:

- Push assets to device only when required, i.e. dependent on context (see B-1).
- Minimal on device storage of assets (see B-2). Assets will be actively deleted when not required. This is an alternative to the lock and wipe function commonly provided by an MDM.
- Protection by resource check at runtime (see B-3).

We make the assumption that the cloud service understands when assets should be pushed (B-1) and deleted and will time this appropriately. For example, the cloud service knows the start and end time of a meeting along with its location. Using this information the cloud can issue the necessary actions, any one of which may trigger the deletion of assets (B-2):

- The cloud detects that the meeting end time has passed or the smart device has left the meeting room.
- When the application transactions complete. This may be because the application workflow has been completed or when the user exits the application. In this case the erase command may come from inside the application. Just before the erase operation any data should be uploaded to the cloud.
- When the context changes. A command is issued from the cloud service or local smart device. For example when the context changes by moving from work to public context an erasure will be performed within the relevant application group. More precisely each application takes one of the three states, i.e., sharing, hiding, or erase. In the case of sharing, even if the context changes, an application will still be available. In the case of hiding, if the context changes, the application will close access and be protected by encryption. The cloud service provides a policy to each application.
- When lifetime expires. If the specified lifetime for assets expires they are deleted regardless of the status of the network

connection. This local enforcement is an effective measure. The timing is specified from the cloud service beforehand or is predefined and kept on the smart device.

- When environment is judged unsafe. When for example there are native applications running other than those on a whitelist, an asset inside the smart device is protected by erasing it.

Protection by resource check at runtime (B-3)

There is also a requirement to protect information inside the separate area because there is a possibility the application itself may be malicious. E.g. an application may reference a malicious URL. In order to detect this, a runtime check is required, as it is difficult to detect malicious code by using static analysis dependent on pattern matching, especially when encryption is used [12]. When a runtime problem is detected the offending code is not executed or the application is terminated, 2 separate whitelists are actually used:

- Application whitelist. Only applications on this whitelist will be granted network access. When network access is requested by an application not on the whitelist the network access is skipped and execution continues to run the next code. This list is generated in the cloud and passed to the device prior to execution of the application.
- Resources Whitelist: This is a set of rules that define the privilege to resources on the device. It lists allowed objects, methods, and parameters and is applicable to all applications. This list is defined by the cloud service which prioritizes the list based on the applications risk of exposure to malicious code injection.

3.3 Ensure Outside Safety of the Separation

External events must not be allowed to affect the separated environment (Figure 1. C) So external environment needs to be evaluated for potential threats. This is necessary as the outside environment may have gained higher privileges due to a security incident (for example a rooted device). We need to have a mechanism to protect the secure environment when such a threat is detected.

It may be possible to detect if a device has been rooted by attempting to perform an operation that should not be possible on a standard device. This approach cannot guarantee to detect 100% of root build though as the specific check being made may pass on some root builds. Some level of security can however be gained by this method especially if an extensive range of checks are made, detecting that a device may be rooted should trigger security measures.

Another security measure is to list the processes running on the device and check this against a list of expected ones, differences in these lists or the presence of a 'blacklisted' process also should trigger security measures.

When a threat is detected the following security measures should be taken:

- Do not allow any new application to run on execution environment
- Terminate any application running on the execution environment.
- Erase any sensitive assets held in the separated environment.
- In addition the security system should generate and send a report about the threat found an the device the threat was found on.

As the types of security threats are constantly changing the security measures will also need to change. The security module will need to have an update mechanism so it is able to detect new threats.

4. IMPLEMENTATION

In this section, we will demonstrate the implementation of the proposed architecture. Figure 3 shows the system structure. For the servers we use FMV-S8350, and for the smart device, we use Fujitsu STYLISTIC M532 [13] running Android 4.0.

The HTML5 application is encrypted by the **packager** using Zip4j [14] (algorithm AES256; Advanced Encryption Standard 256). Symmetric keys are also generated for the application and provided through the key exchange protocol. These keys need to be stored in secure place (i.e. hardware protected storage). In response to a **push gateway** (GW) message the device will execute a download/execute command for an application from the **application repository** (device also needs to respond to erase commands from GW). On the device **push client** downloads the assets and places them in the application repository. The **location sensing** module uses location data (e.g. GPS/Wi-Fi) and sends this information to the **location to context** module on the system server. The **action based on context** module issues a push command when the context matches the service. The **scheduler** may also trigger push commands from the **action based on context** module the issue push commands for example at the start/end times of a meeting. The **desktops** provides context-based user interface with the **application manager** handling application download. The **runtime** is a native Android application previously installed (with full privileges) on the device and implemented using a modified Phonegap [7] and an Android WebView. The downloaded HTML5 application code is processed by the WebView when the application is run. Because the runtime is a native application all HTML5 code runs in the same process space (same user id). In order to secure the downloaded applications assets, the **Secure File System** is used to decrypt / encrypt information on the fly.

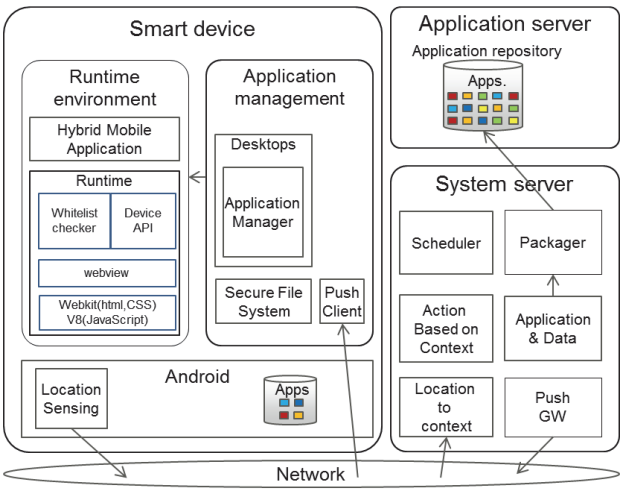

Figure 3. System structure of the implementation

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns="http://www.w3.org/ns/widgets"
    xmlns:wac="http://wacapps.net/ns/widgets"
    id="http://example.org/helloworld" version="1.0 Beta" height="200"
    width="300" viewmodes="floating" wac:min-version="2.0">
  <icon src="icon.png"/>
  <content src="helloworld.html" encoding="UTF-8"/>
  <access origin="http://www.aaabbb.com/*"/>          ⇒ Allows network access
  <feature name="http://wacapps.net/api/camera"/>     ⇒ Allows camera device
  <name short="HelloWorld">Hello World</name>
  <description>Hello World Widget.</description>
  <license href="http://license.example.org/">Example license
    Copyright (c) 2011.</license>
  <author email="myname@host" href="http://foo-
    bar.example.org/">myname</author>
</widget>
```

Figure 4. Example of white list

Access to device APIs and the network is controlled via a whitelist, based on the definition in Widget Packaging and XML Configuration [15] of W3C Working draft and W3C Widget Access Request Policy [16]. Figure 4 shows an example whitelist is shown in this example, the network access to <<http://www.aaabbb.com/>> and camera device access are permitted.

The **whitelist checker** is used by the WebView and links accesses to device APIs and network in the source code.

5. EVALUATION

In this section we evaluate the implemented system. The system consists of some servers and a smart phone, and for the network, 3G or Wi-Fi network is used.

5.1 Operational Example

Evaluation results of (A-1), (A-2), (B-1), and (B-2) are shown here. Figure 5 shows the actual desktop image shown on the device involving context switch and the auto execution of a HTML5 application in response to application push. This confirms that the system works as follows:

For this example, swiping a NFC tag when entering a meeting room triggers an upload of information to a system server. The system server (cloud service) receives this information and interprets it as context change from public to work and in this case decides to issue a context switch command and push a specific application. This is encrypted and packaged, placed on the application server and a push message issued that requests the device switch context and download the application. When the push client on the device receives the push message, it downloads the application at the same time the context changes from outside to work, as shown in upper middle of Figure 5. The runtime environment then executes the application without requiring user interaction, as shown in the right hand side of Figure 5.

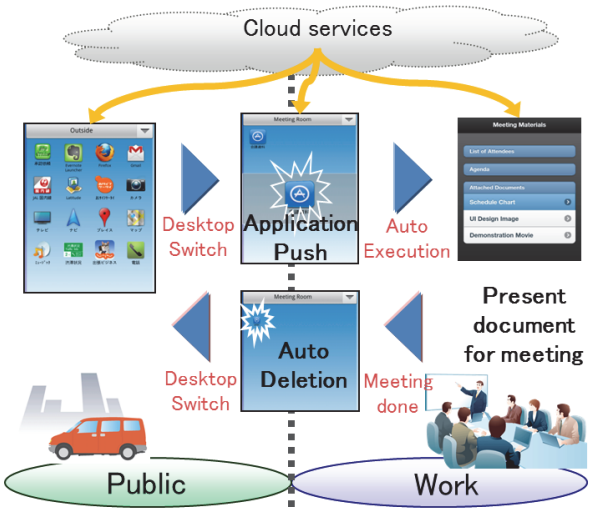

Figure 5. Unity test

When exiting the meeting room a second NFC (Near field communication) tag swipe triggers another upload to the system server. This is interpreted by the server as a context change from work to public and it issues context group switch and erasure command. When the commands are received by the device, context mode is switched to public mode and the application manager erases the application from the encrypted device storage, as shown in the lower middle of Figure 5. If no erasure command was issued and only the context switched to public mode, the application(s) tied to work context will be hidden and the running application is closed. For this example we use NFC for the trigger, alternatively we could have used GPS or some other location detection technology.

5.2 Application Security

An evaluation of full application lifecycle security was performed (see Figure 6). Because the application is fully protected in each stage security can be guaranteed for the full lifecycle.

In summary, after the application has been developed, it is encrypted and passed to the repository using SSL (Secure Sockets Layer). At the next stage the encrypted application is deployed to the device using a context based push (again protected by SSL).

During the storage phase the assets are saved on the device in encrypted form, these assets will only be decrypted on the fly when the HTML5 application is run . When the applications work has finished it will be erased in response to a command from the cloud service. All encrypted files use AES256 encryption and so even if the files can be obtained (for example by using a rooted device) they still cannot be deciphered without the key which is only kept in dedicated hardware storage.

Another advantage of the proposed system is through the usage of application push, we can improve the user experience by eliminating the need for user interaction in order to receive information. User interaction is only required when the runtime is installed. The installation and launch of HTML5-based

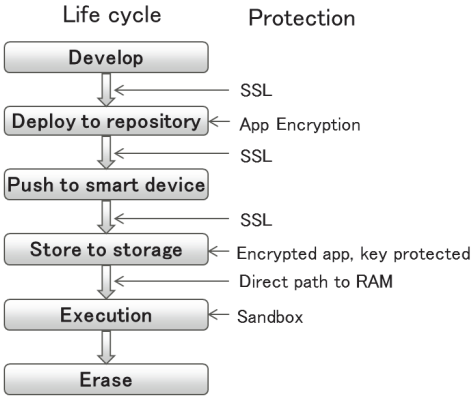

Figure 6. Life cycle of an application

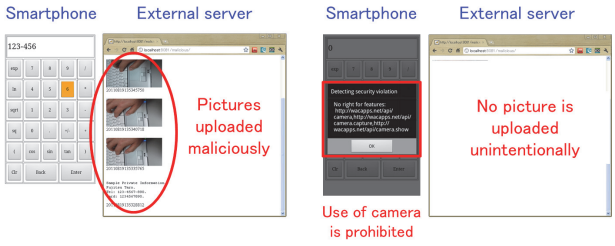

Figure 7. Secure inside of the separation

application does not require any user interaction.

5.3 Loading Application

Metrics were obtained for the load time of applications. Two encrypted HTML5 applications were deployed to a device and kicked for execution. The results were 0.95 second for 156 1KB] size of application and 2.2 seconds for 516 [KB] size of application respectively. These results were fairly slow because of the use of Zip4j as it is. It is believed that this performance could be improved through code optimization and the use of hardware assistance.

5.4 Runtime Protection

An evaluation of, protection by resource check at runtime (B-3) was made. The "Orkut worm [17]" is an inline frame of JavaScript that performs malicious code injection via an external URL. In theory with the proposed system it is possible to enforce the access control to the external URL (Uniform resource locator) using the whitelist checker because the URL is specified statically in the JavaScript.

To demonstrate this, a HTML5 application was developed which included the malware code. The application takes a photo by a camera on the Android smart device and uploads the photo to an external server. Figure 7 show the results. The left hand shows the results without the URL runtime access check. The right hand side shows the results with the runtime check. After pushing the ok button on the warning window, it was possible to continue to use the application because only the code detected as malicious was prevented from working.

6. CONCLUSION

Use of smart devices within the BYOD model is under investigation for the corporate IT sector. We have proposed a system which will mitigate security concerns with the leakage of corporate data by using the following concepts:

- Intelligent cloud service will push and erase applications to and from the device securely according to context.
- Always held in encrypted format, decryption to RAM occurs 'on the fly'.
- The concept of context, and the linkage of context to application groups.
- The use of a separated environment for isolation of application execution.

We implemented the proposed architecture on Android-based smart devices and evaluated in real environment. Since the proposed technique did not need to modify the OS, it was easy to deploy. From the evaluation results, it was shown that the proposed method was feasible to protect from a malicious activity. The future works would be implementing the protection for the outside separation.

7. REFERENCES

[1] Asaf Shabtai, Yuval Fledel, Uri Kanonov, Yuval Elovici, Shlomi Dolev, and Chanan Glezer Ben-Gurion, Google Android: A Comprehensive Security Assessment, IEEE Security & Privacy, March-April 2010, Volume:8 Issue:2, page:35-44

[2] Christopher Mims, Rise of the "Hybrid" Mobile App, http://www.technologyreview.com/computing/37831/page1/, technology review, June 17, 2011

[3] Hidenobu Ito, Kazuaki Nimura, Yousuke Nakamura, Akira Shiba, and Nobutsugu Fujino, Application Push & Play – Proposal on Dynamic Execution Environment Combined with Personal Devices and Cloud Computing, IWIN (International Workshop on Informatics) 2011

[4] HTML5, W3C, http://dev.w3.org/html5/spec/Overview.html.

[5] Web Storage, W3C, November 2011, http://dev.w3.org/html5/webstorage/

[6] Wholesale Applications Community, http://www.wacapps.net/

[7] PhoneGap, http://www.phonegap.com/

[8] Hiroshi Lockheimer, Android and Security, http://googlemobile.blogspot.com/2012/02/android-and-security.html, Feb. 2, 2012

[9] Chris DiBona, https://plus.google.com/114765095157367281222/posts/ZqPvFwdDLPv#114765095157367281222/posts/ZqPvFwdDLPv

[10] Device Administration, http://developer.android.com/guide/topics/admin/device-admin.html

[11] Sven Bugiely, Lucas Daviy, Alexandra Dmitrienkoz, Stephan Heuserz, Ahmad-Reza Sadeghiy,z, Bhargava Shastry, Practical and Lightweight Domain Isolation on Android, SPSM'11, October 17, 2011

[12] David Brumley , Cody Hartwig , Min Gyung Kang , Zhenkai Liang , James Newsome , Pongsin Poosankam , Dawn Song , Heng Yin, "BitScope: Automatically dissecting malicious binaries", CMU-CS-07-133, 2007

[13] Fujitsu STYLISTIC M532, http://www.fujitsu.com/fts/products/computing/pc/notebooks-tablets/all-round/stylistic-m532/

[14] Zip4j, http://www.lingala.net/zip4j/

[15] Widget Packaging and XML Configuration, W3C Working Draft(2011) http://www.w3.org/TR/widgets/

[16] Widget Access Request Policy, W3C Candidate Recommendation(2010) http://www.w3.org/TR/widgets-access/

[17] The Orkut Worm, http://www.symantec.com/connect/blogs/orkut-worm-has-landed