# Role-based Privilege Isolation: A Novel Authorization Model for Android Smart Devices

Batsayan Das
TCS Innovation Labs, India
Email: batsayan.das@tcs.com

Lakshmipadmaja Maddali
TCS Innovation Labs, India
Email: lakshmipadmaja.maddali@tcs.com

Harshita Vani Nallagonda
TCS Innovation Labs, India
Email: harshita.n@tcs.com

***Abstract*—Data ex-filtration is a major security concern in smart devices as they often store private and confidential data. Data ex-filtration can potentially lead to identity theft, financial and non-financial risks, and reputation damage for individuals and organizations. In Android smart devices, sandbox mechanism is not flexible enough to allow an application, such as webbrowser, to protect its own data against attacks such as cross-site request forgery, session or cookie hijacking that exploit application or platform vulnerabilities. These attacks in turn can lead to severe sensitive, private and confidential data ex-filtration. In this paper, we propose a novel authorization model for Android smart devices called Role Based Privilege Isolation (RBPI) which intends to mitigate data ex-filtration. This model achieves fine-grained privilege separation by creating roles based on application usage categories. By using roles, different instances of an application can be made to run with different data access privileges. Thus, the model protects sensitive data even in case where other instances of the same application are compromised. RBPI acts as an additional data security layer on top of the existing Android's security model without any performance overhead. Our proposed model is also applicable on any end-user computing system.**

**Keywords:** Android Sandbox, Data ex-filtration, Mobile Applications, Authorization model, Role, Smart devices

## I. Introduction

The ever increasing use of Android operating system (OS) based smart phones and tablets[1][2] together referred as Android devices, has gave raise to security concerns and challenges. Android has a good security model mainly in terms of application sandbox[3] and permission model mechanisms. With the presence of kernel-level application sandbox, each application (app) runs under its own unique user identity (UID) and owns its private storage space, which is not accessible to other apps for reading and writing. By default, apps cannot interact with each other and they have limited access to the OS. With the presence of application-level permission model, Android restricts apps' access to user data such as SMS and call log, and device features such as camera and microphone and the app requested permissions are shown to the user during app install time. The application sandbox mechanism is mainly focused on protecting app specific data stored on internal storage from other apps. The permission mechanism is static and coarse-grained, and its main purpose is to protect device resources and app data access from unauthorized access by other apps.

In spite of above security mechanisms in place, data ex-filtration can occur in Android devices due to cross-site request forgery, session or cookie hijacking attacks. These attacks exploit app or platform vulnerabilities when coupled with malicious code coming from a website or an email attachment. For example, buffer overflows and improper input validation can be treated as app or platform vulnerabilities. Android apps were shown to be vulnerable to several attacks. For instance, a webbrowser app when used to browse a malicious site accidentally, can attempt to exploit an already logged in session (by stealing cookies, session tokens) of a banking website or access browser's sensitive data (such as browser history) [4][5][7][11]. Similarly, a PDF reader used to view a maliciously crafted PDF can be subverted to steal (by executing arbitrary code) other confidential documents (like bank statements, payslips) that happen to have been read via the same PDF reader[12]. In all the above cases, upon execution of the malicious code, an attacker gains access to everything and anything the compromised app has access to. This includes app specific data stored on internal as well as external storage spaces. If the app has read access to external storage, malicious code even gets access to other apps' data present on external storage as Android OS treats the external storage as world-readable. Further, data exposed to the attacker may be removed from the smart devices by the attacker which leads to data loss. Such ex-filtration could have been prevented to some extent, if Android notified the user about data usage at run-time or allowed the user to control data sharing. Unfortunately, Android has no mechanism for granting permissions dynamically (at run-time) because it complicates the user experience to the detriment of security. Android 4.3 included a hidden feature called App Ops, which gave an end user the ability to selectively disable permissions of an app. With such a feature, a user could have disabled external storage and Internet access to suspicious apps and avoided data ex-filtration. However, such disabling could impact functionality of apps and can even crash them. Eventually the App Ops feature was removed in Android 4.4.2.

In this paper we address above data security risks in Android devices by introducing a *fine-grained privilege separation access* mechanism for Android devices. We call this model "Role-based privilege isolation" (RBPI). This authorization model separates and uses roles based on app usage category or functional purpose. For example, a webbrowser instance used for random browsing (unknown or arbitrary websites, probably malicious) should be separated from sensitive browsing (bank-

ing website, corporate portal) that might expose sensitive data. That is, data pertaining to one role is isolated from all other roles. Hence, RBPI achieves fine-grained privilege separation and thus addresses the challenge of securing potential data from loss or theft in case malicious code or an adversary takes control over an app. We introduce a design framework for RBPI. It has three major components: (a) Role Management module to manage roles (b) Application integration module to integrate apps with Role Management module and (c) Data security module to manage data accumulation pertaining to various roles. A prototype architecture has been developed to realize the RBPI design framework. We also present an adversarial model against which the provision of RBPI on Android devices mitigates data ex-filtration. Finally, we present limitations of RBPI model.

The paper is organized as follows. Section II presents related work. Section III describes all the aspects of RBPI. It has the following sub sections which explain RBPI architecture III-B, logical components III-C, implementation III-D, adversarial model III-E, limitations III-F and finally highlights security with and without RBPI solution III-G. Section IV presents our conclusion. Finally, Section V suggests the directions for future work.

## II. Related work

From our study we found that existing solutions do not address data security risks such as data ex-filtration and data loss completely and moreover, these solutions encounter adaptability issues (need changes to Android OS or existing apps) and manageability issues (implemented for older Android versions and require technical skills to use the customized version). One solution[6] addresses threats from malicious URL attacks that steal data. Very few solutions address protection mechanisms for external data storage (SD card). Researchers suggested some generic data security solutions to protect data on Android smart devices using various approaches. These approaches include imposing access restrictions by using role based access control [9][10][14], mandatory access control mechanism [13], provision of enhanced encryption mechanisms, digital rights management policy and organizing apps and data into logical domains [8]. Role based access control solutions works by limiting permissions of apps based on roles, thus leading to app crash or functionality loss. Mandatory access control mechanism and digital rights management policy usage may need specialized skills for proper policies' setup, which might be difficult for most users who are not generally security-aware. Research solutions prevent privilege escalation attacks from affecting other domains and unauthorized data access to other domains (by segregating and placing apps under different logical domains) however they do not address the threats coming from outside world which target a single app or service within a single domain. RBPI solution aims to provide multiple private mode sessions dubbed as role for third-party applications which is to some extent similar to PrivateDroid [15], which provides single private mode for third-party applications.

## III. Role Based Privilege Isolation (RBPI)

The goal of RBPI solution is to effectively and efficiently address the data security issues, which includes mitigating data theft and reducing data loss as explained in Section I. This data includes app and user-sensitive data present on the device memory.

### A. Design Consideration

RBPI mitigates data security concerns through a fine-grained privilege separation model. The segregation is not done by an app (such as a webbrowser, email client, PDF reader) but by a notional 'role' or by the kind of activity an app is about to perform. Here, a 'role' can be random_browsing, secure_browsing, office_email, private_email, unknown_doc, sensitive_doc and the kind of activity can be accessing an unknown website, accessing a financial website, accessing corporate email, accessing official document etc. We have extended the traditional definition of a 'role' in role based access control model to include a notion of a functional purpose or app usage categories. When a user accesses a bank website, he is playing a different "role" than when he visits his employer's internet portal, and yet another role when he visits an untrusted website. Similarly, viewing a PDF pay-slip received from the employer and viewing a PDF received from an unknown person, are deemed to be different "roles". However, multiple apps used in the same "role" can also leak data. RBPI proposal is to isolate data in different "roles", even if they are using the same app. By using various roles, we can run different instances of an app with different, non-overlapping set of access rights. Thereby, the data pertaining to the each app instance can be secured using this solution.

For practical use, RBPI solution's preliminary model features a user-friendly interface that allows the user to segregate "important" work from "less important" and "casual" work by allowing the user to create as many "roles" (e.g. random_browsing, secure_browsing, office_email, private_email) as needed. This makes sure that compromise in one role does not affect any other role(s). For each app instance, a new role can be created on the fly or an existing role (if any) can be chosen. Here the distinction on role selection is to be done logically based on app usage context. Thus, compromise in one instance (accessing an unknown website) of an app does not affect other instances (accessing a financial website). This preliminary model provides end-user with a basic security environment that deals with role separation. With such an approach, RBPI works as an authorization model and minimizes the theft or loss of corporate or personal data on app compromise through cross-site request forgery, session or cookie hijacking that exploit app or platform vulnerabilities such as buffer overflows, improper input validation. Our development experience indicates that the proposed RBPI solution itself will be technology-neutral, and ought to be adaptable to most end-user computing environments (such as laptop and desktop environments) in common use today, even if the present paper focuses on Android.

### B. Architecture details

An architecture model of RBPI is shown in Fig. 1. Once the user chooses to secure an app by integrating with RBPI, whenever a user invokes the app, app wrapper will be invoked, instead of the actual app, and provides the unified RBPI User Interface (UI) to the user. Using the RBPI UI the user may either select a role from the list, or create a role on the fly based on the app usage category, and then select the newly created role. Once the user selects a role in the RBPI UI, the actual app gets invoked in that role. Thus, isolation is achieved among app instances.

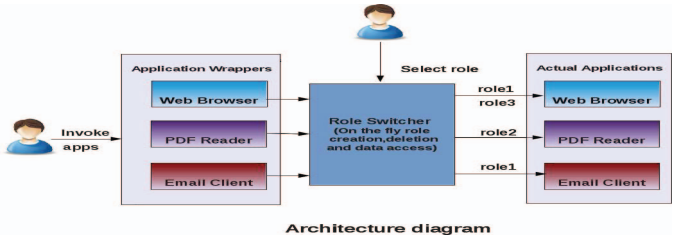


Figure 1. Architecture overview

To protect an app (thus all its instances) under RBPI, user has to integrate it with RBPI using the UI. It does not provide any default roles or pre-created roles to the user. Thus, on the initial use of an app user need to explicitly create a suitable role based on app usage category.

### C. RBPI components

RBPI consists of three main components, as shown in Fig. 2. The first component is 'Role management module', which is a privileged component to manage roles. The second component is 'Application integration module', which is used to integrate app(s) with role management module. The third component is 'Data security module', which handles data accumulation pertaining to various roles. RBPI provides a unified UI to interact with these modules.

*Role management module:* The role management module provides role management capabilities, which includes creation, deletion or selection of roles. Role is a device level human user on Android operating system. Thus, we have analyzed Android built-in multi-user and restricted profile features. These features provide a way to set multiple human user accounts and provide same level of data isolation. With multi-user feature, owner can create users with access to entire android stock apps whereas with restricted profiles feature, owner can create users with access to limited Android stock apps. Moreover, apps associated with user accounts cannot be made available in the restricted profile. In other words, a role created based on restricted profile feature may not have all apps. Hence, we have chosen multi-user feature over restricted profile for creation of role. The multi-user feature requires an extra app installation step for third-party apps in a role, which RBPI handles transparently. As we are leveraging multi-user feature, RBPI achieves data isolation across roles and avoids the need for app or platform changes. Role management module will receive user inputs to create roles. Upon receiving the user inputs, the role management module will create the roles. Such provision allows the end-user to self-define their own roles based on the current usage of the app. By doing so, an end-user can impose fine-grained access restrictions on the apps. This module provides a list of pre-defined roles from which a user may identify a role to be associated with the app instance. Role management module allows users to create two types of roles namely permanent role and temporary role. The permanent role and its corresponding data are intended to be persistent even after app usage instance got completed. This type of role is recommended to avoid re-creation of role(s) for tasks that are done on a regular basis. The temporary role and its corresponding data are intended to be temporary (deleted automatically) for the app usage instance. This type of role is recommended for temporary tasks and useful to overcome the maximum allowed users limit on a smart device.

*Application integration module:* The application integration module is used to integrate app(s) with role management module. This module receives user inputs that identify an app to be integrated with the role management module for securing its data. User interface (UI) to this module can be invoked in two ways. Upon selecting the application integration module in the RBPI unified UI and invoking a pre-integrated app. Once the UI is shown, user can select an app from the list of supported and/or installed apps. Once an app is integrated, role management module will be invoked first whenever the app instance is opened/invoked. This reminds the user to select an appropriate role in which the app instance needs to be executed.

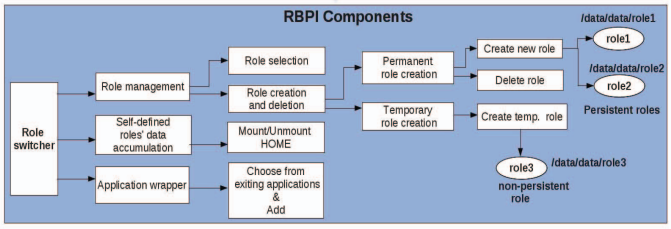


Figure 2. RBPI components

*Data security module:* The data security module handles data accumulation pertaining to various roles. By doing so, a device level human user can access data belonging to all self-defined roles and copy it to another safe location. If the user copies and removes roles' data regularly, thus making no sensitive data available in that role, data loss or ex-filtration can be prevented even if some malicious script takes control over an app instance running in some particular role.

### D. Implementation

RBPI application can be used on any Android devices (tablets and smart phones) running Android 4.2 Jelly Bean and above versions, if these devices have the support for creating multiple user accounts. The multi-user feature is available for tablets using Android 4.2, but the feature was not available for smart phones running the same version. However, with the arrival of Android 5.0 Lollipop version, the multi-user support is been available to Android smart phones. RBPI is implemented on Android OS version 5.0 Lollipop (API Level

21) for smart phones. RBPI app requires managing of user account, so we choose Android 5.0, as this version offers the ability to deploy device owner app. Device owner app provides a way to create a user, switch to a user session from its handle and it can also remove a specific user from its user handle. Device owner app is a special kind of admin app which has full control over the device. This device owner app can not be modified by the user and the only way of removing this app is to do a factory reset. Setting a device owner app is done by using a NFC message and this has to be done on an unprovisioned device. Once the NFC message is delivered, using network connection, RBPI APK will be downloaded and installed on the device. It is assumed that RBPI is already provisioned as device admin app and is installed in the device running Android 5.0. We have successfully tested RBPI application on an HTC Nexus 9 on Android 5.1 LMY47X Lollipop.

Following are the RBPI screenshots of activites and their respective description of each. The first screen or activity shown in the Fig. 3 represents the role management module activity and the second screen represents the application integration module activity. The role management activity will help to manage roles in the device. This activity is used to create, delete and switch roles. Here the user will be able to define suitable roles based on various apps usage, for example, a webbrowser app in random_browsing role (to access unknown or arbitrary websites, probably malicious) and secure_browsing role (to access banking website, corporate portal). The second activity i.e., application integration module, contains the list of installed apps on the device. This activity is mainly used to integrate installed applications with role management module activity. Each application will have an ON/OFF options, through which user have to identify the apps to be integrated with the role management module for securing its data. Once the app (ex webbrowser) is integrated, role management module will be invoked first whenever the webbrowser app instance is opened or invoked. This reminds the user to select an appropriate role or create new role on fly in which the app instance needs to be executed.

The first screen shown in the Fig. 4 represents the data management module activity and the second screen represents RBPI settings activity. The data management activity handles data accumulation pertaining to various roles. The list of created role will be shown to the user. Here, a device level human user can access data belonging to all self-defined roles and copy it to another safe location. By taking regular backups of sensitive data from all the roles and removing it from roles will help users to prevent data loss or ex-filtration, even if some malicious script takes control over an application instance running in some particular role. The RBPI settings activity will be used to start and stop RBPI service manually. RBPI service runs in the background continuously to check RBPI integrated applications activity stack, if it find a new activity then it will show role management activity so that user can select or create new roles as per their requirement. If user wants to remove all RBPI rules then user can click

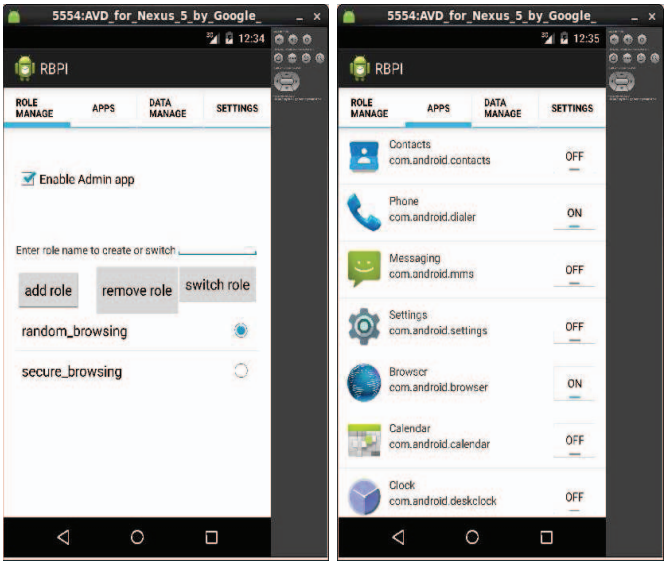


Figure 3. Role Management activity and Application integration activity

on "stop service" button. This module provides a very fast and easy way to enable or disable the RBPI service. In RBPI, each role is a device level human user with his/her own data storage at `/data/user/<user_id>/` and application specific data storage at `/data/user/<user_id>/<app_package_name>`. Here, 'user_id' represents the human user and app_package_name represents the application package name.

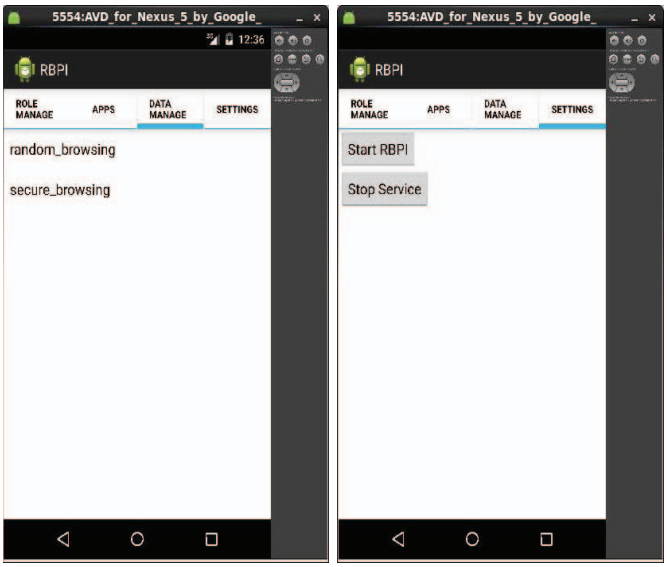


Figure 4. Data Management activity and Start/Stop RBPI service

The first screen shown in the Fig. 5 lists all the created and running roles in the device and the second screen demonstrates the process isolation for the webbrowser application started on different users (u10 and u11). Process isolation is achieved as the webbrowser is started under different UIDs u10_a15 and u11_a15. Here, "a15" represents package specific application user id for webbrowser and it is same across different users.

Storage isolation for the different instances of an webbrowser application is shown in Fig. 6. Here each webbrowser instance has its own storage space under `/data/user/10/`,

```
root@generic:/ # pm list users
Users:
        UserInfo{0:Owner:13} running
        UserInfo{10:rbpi-secure_1:10} running
        UserInfo{11:rbpi-secure_2:10} running

root@generic:/ # ps |grep com.android.browser
u11_a15   4831  2724  752680 78164 ffffffff b6deac9c S com.android.browser
u10_a15   5050  2724  778844 106180 ffffffff b6deac9c S com.android.browser
```

Figure 5. List of sample roles and Process isolation of webbrowser instances in roles

`/data/user/11/`, and file permissions on these directories ensure that a webbrowser application in one role cannot access another role's webbrowser data, thus achieving storage isolation. This improves security as even if an application is compromised, damage is limited to the data belonging to that role in which the application (here webbrowser) is started under and cannot affect the rest of the data owned by other roles. Hence RBPI provides a way to run different applications instances with different privileges using roles. This segregation will minimize the chances that a vulnerability in a webbrowser application or a website, can be used to compromise sensitive information.

```
root@generic:/ #  ls -l /data/user/10 | grep com.android.browser
drwxr-x--x u10_a15  u10_a15          2015-10-09 12:45 com.android.browser
root@generic:/ #
root@generic:/ #  ls -l /data/user/11 | grep com.android.browser
drwxr-x--x u11_a15  u11_a15          2015-10-09 12:47 com.android.browser
```

Figure 6. Storage isolation of multiple webbrowser instances in different roles

### *E. Adversarial model*

In this section, we present two important attack scenarios on Android devices that can be foiled with the implementation of RBPI design on these smart devices. The adversary's goal in these scenarios is to ex-filtrate data and remove data on the device. In the first attack, the adversary exploits legitimate or benign app vulnerabilities, like buffer overflows and improper input validation. An adversary can exploit these vulnerabilities by tricking a user to open a malicious website, document or a media file. By exploiting the app, adversary acts as or gets the privileges of the compromised app. Therefore, the adversary gets control over device sensitive data, such as contacts, SMS. In another attack, adversary develops malicious apps by incorporating data collection or removal capabilities. Adversary tricks the user to install such malicious apps by designing them to look benign such as Trojans. These apps asks for unnecessary permissions and misuse them to collect sensitive data.

### *F. Limitations*

Our proposed solution focuses to provide security for data present on the smart device. We have considered Android kernel as our trusted computing base and implemented RBPI solution based on in-built multi-user feature. Thus, data security issues raised through Android operating system kernel exploits or root exploits are not protected in RBPI model. The number of roles that are usable (unless the device is rooted) in a device depends the limit of the maximum users supported in that device, as role is implemented using multi-user feature. For example, we can create up to 4 users (and thus 4 roles) on a Nexus phone and 8 users (and thus 8 roles) on a Nexus tablet.

### *G. Security with and without RBPI solution*

With increasing threats from software attacks, the web-browser applications are easily exploited through the use of compromised or malicious or potentially dangerous websites. Malicious or untrusted websites can passively compromise the system, as soon as the user visits the site. Similarly, the email attached malicious document can compromise the system soon after opening the attachment. In all these cases, with exploiting vulnerabilities in applications (like a webbrowser, PDF reader), the attacker can steal user sensitive information like credentials, remove or alter user data files present on the system. Based on the application usage categories, RBPI would enable users to run multiple application instances of an application (here webbrowser) in different roles, thereby isolating malicious websites from trusted sites. The advantage of having RBPI is, a single application instance of an application (like a webbrowser) can be used only for browsing sensitive activities (such as net banking), and another instance of the same webbrowser application can be used for general purpose web browsing (such as browsing HTTP websites). This segregation will minimize the chances that a vulnerability in a webbrowser application or a website, can be used to compromise sensitive information. Other software applications on the end-user computing systems, such as email clients or document viewers (Adobe reader etc.,), or media players (audio, video, and image-viewers), may also use different roles for running different instances of an application(s) based on their usage purpose.

## IV. Conclusion

In this work, we presented Android permission model and sandbox mechanism and their shortcomings. In short, existing Android security model preclude apps from accessing each others' data stored on internal storage and doesn't sufficiently address threats originating from social engineering attacks. In particular, there exists no efficient solution to protect app and user personal data stored on external storage. As a solution we presented RBPI, an additional data security layer for Android. It provides a way to run different apps instances with different privileges using roles. Each role has its own isolated internal and external storage spaces. This improves security as even if an app is compromised, damage is limited to the data belonging to that role in which the app is started under and cannot affect the rest of the data owned by other roles.

## V. Future work

As our future work, we will extend the RBPI model presented in this paper to a more comprehensive model,

which works as a defense based on current role and context. This context might consider, for example, user environment (location, time, user identity etc.,) and application environment (type of network access through which the app is getting accessed) parameters. For example, defining default application specific roles such as random browsing, secure browsing for a webbrowser application and restricting malicious website browsing and giving an option to switch to random browsing while in a sensitive role such as secure browsing. More importantly, the context based awareness could (for user who opt for it) be extended to take a more aggressive stance on defense, by actively blocking user behavior (such as clicking on a potentially malicious URL while logged in to their bank account), that might compromise the security of the current "role". Such a model could void or reduce the burden on end-user in role definition and switching.

## REFERENCES


[1] IDC, *Android Smartphone Market Share, Q3 2014*, http://www.idc.com/prodserv/smartphone-os-market-share.jsp, (Access Date: 19 Feburary, 2015).

[2] Gartner Report,*Android Worldwide Devices Market Share, Q2 2014*, http://www.gartner.com/newsroom/id/2791017, (Access Date: 26 November, 2015).

[3] Android Source,*The Application Sandbox*, http://source.android.com/devices/tech/security/overview/kernel-security.html#the-application-sandbox, (Access Date: 26 November, 2015).

[4] Thomas Cannon, *Android Data Stealing Vulnerability*, http://thomascannon.net/blog/2010/11/android-data-stealing-vulnerability, 2010, (Access Date: 26 November, 2015).

[5] Sebastin, *How I Met Firefox: A Tale about Chained Vulnerabilities*, https://www.nowsecure.com/blog/2013/10/02/how-i-met-firefox-a-tale-about-chained-vulnerabilities/, (Access Date: 26 November, 2015).

[6] Wang, Rui and Xing, Luyi and Wang, XiaoFeng and Chen, Shuo, *Unauthorized Origin Crossing on Mobile Platforms: Threats and Mitigation*, Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, ACM, New York, NY, USA.

[7] Xuxian Jiang, *Android 2.3 (Gingerbread) Data Stealing Vulnerability*, http://www.csc.ncsu.edu/faculty/jiang/nexuss.html, 2011, (Access Date: 26 November, 2015).

[8] Bugiel, Sven and Davi, Lucas and Dmitrienko, Alexandra and Heuser, Stephan and Sadeghi, Ahmad-Reza and Shastry, Bhargava, *Practical and Lightweight Domain Isolation on Android*, Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM '11.

[9] Felix Rohrer and Yuting Zhang and Lou Chitkushev and Tanya Zlateva, *POSTER: Role Based Access Control For Android (RBACA*, Annual Computer Security Applications Conference (ACSAC), 2012.

[10] Xudong Ni and Zhimin Yang and Xiaole Bai and Adam C. Champion and Dong Xuan, *DiffUser: Differentiated User Access Control on Smartphone*,IEEE, 2009. Annual Computer Security Applications Conference (ACSAC), 2012.

[11] Tod Beardsley, *Major Android Bug is a Privacy Disaster (CVE-2014-6041)*, https://community.rapid7.com/community/metasploit/blog/2014/09/15/major-android-bug-is-a-privacy-disaster-cve-2014-6041, 2014, (Access Date: 26 November, 2015).

[12] Yorick Koster, *Adobe Reader for Android Exposes Insecure Javascript Interfaces*, https://www.securify.nl/advisory/SFY20140401/adobe_reader_for_android_exposes_insecure_javascript_interfaces.html, 2014, (Access Date: 26 November, 2015).

[13] Smalley, Stephen and Craig, Robert *Security Enhanced (SE) Android: Bringing Flexible MAC to Android*, http://internetsociety.org/doc/security-enhanced-se-android-bringing-flexible-mac-android, 2013, (Access Date: 26 November, 2015).

[14] Thiri The Wut Yee and Nilar Thein, *CtRBAC: Leveraging access control mechanism of android smartphone using context-related role-based access control model*, NCM, 2011.

[15] Su Mon Kywe,Christopher Landis,Yutong Pei,Justin Satterfield,Yuan Tian,Patrick Tague *PrivateDroid: Private Browsing Mode for Android*, Proceedings of the TRUSTCOM '14 Proceedings of the 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Pages 27-36.