# Mobile OS Security and Threats: A Critical Review

Ahmad Talha Siddiqui
Research Scholar
IFTM University
Moradabad

Mudasir M Kirmani
Assistant Professor
SKUSAT-K

Abdul Wahid, Ph.D
Associate Professor
Maulana Azad National Urdu Univesity

## ABSTRACT
The adoption of Smartphone's in daily lives is transforming from simple communication to smart and the use of these multipurpose devices is rapidly increasing. The main reason for the increase in the Smartphone usage is their small size, their enhanced functionality and their ability to host many useful and attractive applications. However, this vast use of mobile platform makes these smart devices a soft target for security attacks and breach of privacy. The cases about the attacks on personal mobiles phones are on the rise which is a motivation for developing Smartphone application with better effective and efficient security measures to mitigate the impact of possible threats. This paper examines the feasibility of malware development in Smartphone platforms by average programmers that have access to the official tools and programming libraries provided by Smartphone platforms. In this paper comparison of Smartphones like Android, Blackberry, Apple iOS, Symbian, Window Mobile is given based on the specific evaluation criterions used for assessing the security level.

## Keywords
Smartphone, Security, Malware, Attack, Evaluation Criteria, Operating System

## 1. INTRODUCTION
Smartphone is the devices that enhance vision of universal computing: their small size, connectivity capabilities, storage capacity, mobility and their multi-purpose use are some of the reasons for their vast pervasiveness[10] the malware has also appeared in the Smartphone platform[12]. Apart from the increasing Smartphone sale, the annual downloads of Smartphone application are also on rise. Furthermore the use of Smartphone perimeter of an organization has increased besides increasing the sale of Smartphone and annual download of application from official as well as free source. Smartphone contains a vast amount of the user data, thus giving a serious privacy threat to the sector. In additional Smartphone consist of popular web applications like e-mail, YouTube, social media, social networks facebook and twitter etc[11][19] are being accessed through native applications instead of their useful web browser interface. In this context Smartphone often manage a vast amount of user data, causing a serious threat to privacy of data.

This data is extremely useful for attackers'. Hence attacker tries to destroy or damage Smartphone with malware applications, harvesting Smartphone data without the user's knowledge and consent. It is worth mentioning that the everyday use of Smartphone by non technical and non security savvy people has increased and the likelihood of using Smartphone as a security and privacy threat has increased as well. This paper examines the feasibility and easiness of malware development on Smartphone by average programmer that have access to the official tools and programming libraries provided by Smartphone[5][6][21].

It is important to mention that due to the lack of awareness about security concerns among user community of these multipurpose devices it has become a hot cake for cyber attacks[16][27] and people trying to explore weakness of Smartphone softwares.

## 2. SMARTPHONE SECURITY MODELS
In this section we discuss the security models and development environments of the Smartphone platform

a) Android
b) Blackberry
c) Symbian
d) iOS
e) Window Mobile.

### 2.1 Android
The Android is a Linux based open source OS developed and maintained by Google[33][34][35]. Android provides a free and publicly available software development kit that consists of tools, documentation and emulator necessary for the development of new application in Java. A core element of the android security model[23] is the manifest file. The manifest[33][34][35] provides the necessary information to android for the execution of an application, the manifest file is crucial for the system, since a developer defines within the application permission. Everyday android applications have to be digitally signed by its developer. Android's security model then maps the signature of the developer with a unique ID of the application package and enforces signature level permission authorization[23]. In the Android security model the applications are usually digitally signed with self signed certificates, providing only for source origin and integrity protection.

### 2.2 Blackberry
The Blackberry is an operating system maintained by Research In Motion Inc (RIM). The current version of the OS is 6. Documentation about the OS details is not provided by RIM. However[36][37], RIM provides, through the Blackberry software development kit, the related documentation, tools, APIs and emulator that are necessary for application development. The platform of security model[25] forced restrictions to 3rd party applications trying to access which is protected APIs of the OS by demanding the application which is signed with a cryptographic key provided by RIM[14]. In order to acquire a valid RIM signing key pair a developer needs to pay a small amount. However this process provides only poor source of origin and code integrity and does not offer any assurance about the validity and the security level of the 3rd party application.

### 2.3 Symbian
Symbian is an operating system whose current version is Symbian-*3* which is maintained by Nokia[38]. In the Smartphone Symbian is executed and provides multiple free and publicly available SDKs. The tools documentation and emulators that are necessary for the development of new

applications written in C++ which are included by SDK. The cornerstone in Symbian's security model[38] is the use of capabilities for defining restrictions to sensitive platform APIs. These capabilities are grouped in the following categories: a) basic, b) extended, c) manufacturer. The first category includes basic functionality during installation where the users are prompted for its authorization.

The use of sensitive API that is only granted through the Symbian signed process[38] which is controlled by the second capability category. The last capabilities category controls application access to the most sensitive API in the platform and there all capabilities are only granted by a device manufacture. Nokia which is indicated the basic capability category contained sufficient functionality for application development for the installation of the each application, signing the applications package file with .sys extension is required by the Symbian security model.

Signing confirms that the application is not using API apart from the one corresponding to the applications signing level. The developer can self sign it if the application uses only basic capabilities self signing has the advantages to be performed in the developer's computer and it is not necessary to map the application installation package file with a device IMEI in the result there is no restriction during the installation of the multiple device, but the Smartphone users will be prompted with security warnings at installation time since the signing key is not trusted. The developer submits the application to Symbian signed along with the list of device IMEIs to get rid of the warnings and access sensitive capabilities.

## 2.4 Apple iOS

iOS is a proprietary operating system is only executed in Apple Smartphone and tablets. After registration on Apple[39][40] provides to the company center, documentation tools and the necessary API for application development[13]. It should be noted that the tool set provided by Apple is only compatible with MAC operating system. The security model[39][40] of the iOS only permits the installation of applications that have been signed by Apple. An application is tested for its functionality consistency and for malicious behaviour before it's signed. However the testing process and criteria are not openly available which is applied by Apple. Applications are installed on the device with: (a) the use of an application installed on the device that connects to Apple's App Store, or (b) the use of cross-platform synchronization desktop software provided by Apple. Once the application is installed to the device the user neither controls nor is prompted when an application accesses some OS sensitive resources.

## 2.5 Window Mobile

Window mobile is an OS Smartphone which is developed and maintained by Microsoft. The security model of windows mobile[31] depends on the operational policy of the device for controlling that which application one allowed to be executed on the device[41] what functionality of the OS is reading availability to the application, how the user or application use specific device selfing and how desktop applications interact with the Smartphone for their operational responsible. The operational policy on a window mobile Smartphone is either one-tier access or two-tier access [31].

A device with one tier access policy operational only controls, if one application runs on the device or not without examining officially if the application is using sensitive API. This decision depends on whether the applications installation package file is correctly signed with a certificate that exist in the device's certificate store if the application is signed with a known certificate then the application runs in advantage mode with the ability to call any API, access and modify anything in the device file system and registry but if the application is unsigned or signed with a unknown certificate later on policy checks take place for the decision of application execution. In this case, security policies setting define whether the user is ready to give her permission for the application to run. It must be clarified that if the user permits the execution, then the application run in advantage. This indicate that an unknown and unsigned application maintains full access to the device.

According to the lack of security configuration of windows mobile, provides[26][31] weak security protection as: (a) it allows the execution of unsigned applications or suffer harm with an unknown certificate, and (b) in this case the user is permits to authorize the execution of the application. Hence the lack of the security configurations in both access tiers. Unsigned and unknown code is executed with the user's approval either in normal mode or advantageous mode. Moreover although one-tier access does not provide strong safety. It is the default access tier[20] in some devices it should be solved that the safety model permits mobile operators to make post.

# 3. COMPARATIVE EVALUATION OF SMARTPHONE

A comparative evaluation of the Smartphone platforms in terms of malware distribution and development. Our analysis examines the feasibility of attack implemented by average application developers. More peculiarly the presented evaluation is based on: (a) the definition of qualitative evaluation criteria, and (b) a proof of concept malware implementation study in which the development of location tracking application is examined.

## 3.1 Evaluation Criteria

A comparative evaluation of Smartphone platform is performed by using a set of evaluation criteria that are elaborated in the table-1. The proposed criteria concern the development platform and the developer. The platform based criteria are objective, relying solely on the platform characteristics. Contrarily the developer related criteria are subjective, giving details about the development effort and as a result depends on the developer's skills and background.

**Table 1. Proposed Evaluation Criteria**

| Evaluation Criteria | Type |
|---|---|
| Development Tools availability | String (Yes, Partial, No |
| Development Friendliness | Boolean |
| Installation Vector | String (Multiple, Restricted) |
| Application Portability | Boolean |
| Application Testing | Boolean |
| Aplication Removal | Boolean |
| Unofficial Repositories | Boolean |
| Distribution Cost | Boolean |

| | |
|---|---|
| API Restrictions | Boolean |
| Application Signing | Boolean |

# 4. IMPLEMENTATION OF MALWARE ATTACK

Our study examines the implementation feasibility of a simple attack scenario. As a proof of concept to study the implementation of a malware attack is presented. To evaluate the robustness and the security properties of the smartphones platform under examination of criteria defined in the previous section are applied.

The application collects the Smartphone's user GPS coordinator and sends them to the attackers. It is assumed that the malicious functionality is included in a free GPS navigation application. The malware described in most cases, that it is executed without creating any suspicion to a naïve Smartphone user. The reason for this is that the applications regrets are compatible with the application's expected functionality.

The analysis of the results regarding the development and use of this malicious application to the Smartphone platform. The platform that we have examined are: Android OS, Blackberry OS, Symbian OS, Apple iOS and Window Mobile OS. The results of the case study are presented.

## 4.1 Case Study: Android

On the Android platform our attack implementation was successfully developed in one day for the implementation purposes. The reasons that why our attack implementation was efficient are: (a) the platform adopts the widely used programming language, (b) the effective documentation of its API. In addition the same source code successfully accumulates and executed in version 2.1 and 2.2 of the Android platform; hence within the Android platform the application is considered portable. There are many options for the attackers regarding application distribution the reason for this is that the Android platform does not force any restriction neither on the source of application nor on the installation vector. For the placement of applications in the official repository a small registration fee is required but it is considered inadequate to impede an attacker moreover, even application testing for malicious behavior is not taking place, if the official repository is selected for the distribution. Hence, it is likely that malware such as one described in this case study is currently presented and downloaded by navie users from the repository.

As we have already described the security model of the Android platform imposes some application restriction concerning the application signing and API control. We argue that there restrictions provide only partial security protection. API control restrictions are authorized by the navie user for the former only during the application installation. After the installation no further checks about application permissions task place. Hence, especially in our case, it is likely that the malicious application would be granted the requested permission where it is fully match the expected applications functionality.

**Table 2. Android Analysis and Result**

| Evaluation Criteria | Android |
|---|---|
| Development Tools availability | Yes |
| Development Friendliness | Yes |
| Installation Vector | Multiple |
| Application Portability | Yes |
| Application Testing | No |
| Aplication Removal | Yes |
| Unofficial Repositories | Yes |
| Distribution Cost | No |
| API Restrictions | Yes |
| Application Signing | Yes |

**(Source: http://developer.android.com/guide/topics/security/ security.html)**

## 4.2 Case Study: Blackberry

The result was again successful regarding our attack analysis on Blackberry platform. The attack was conducted by the RIM's official development toolkit was used for attack implementation. The duration of the attack implementation was not demanding. The adoption by the platform of a widely used programming language and the effective documentation its API are the reason for the effectiveness of the attack implementation. Moreover, the same source code successfully compiled and executed in version 5/6 of the Blackberry platform, therefore the application is considered portable. Regarding the origin of the application the security model of the Blackberry does not impose any restrictions. Nonetheless to access restricted and sensitive platform APIs the application package file must be signed. Before accepting the submission of an application the security model of the RIM's platform does not empty any application moreover, there is no application removal system automatically removing applications with malicious behaviour. Hence, if the malware application is submitted in the official repository then it is very likely to be downloaded and installed in Blackberry devices.

**Table 3. Blackberry Analysis and Result**

| Evaluation Criteria | Blackberry |
|---|---|
| Development Tools availability | Yes |
| Development Friendliness | Yes |
| Installation Vector | Multiple |
| Application Portability | Yes |
| Application Testing | No |
| Aplication Removal | No |
| Unofficial Repositories | Yes |
| Distribution Cost | Yes |
| API Restrictions | Yes |
| Application Signing | yes |

**(Source:://docs.blackberry.com/en/developers/deliverables/21091/Security_ overview_1304155_11.jsp)**

## 4.3 Case Study: Symbian

Symbian OS provides basic functionality sufficient for application development providing the developer the option to

self sign her application nonetheless, since the location capability, some compatibility issues exit, which controls access to API determining the location of the device does not reside in the basic capability category in some Symbian OS version.

Only the basic capability category was used for the development of the malware attack scenario. The user has to accept the self signed applications create a security warning at installation time. Even so the user would likely accept the installation of the application by passing and ignoring the security warnings. Apart for signing the application, the security model of the platform doesn't restrict the application's distribution and as a result the attacker has many distribution options in the Symbian platform the attacker implementation was not successfully compiled even though with the officially recommended development toolkits implementation was performed, the case study developers were unable to compile their code. Even the sample applications provided by Symbian could not be compiled the installation of the development toolkit was fully automated and the developers did not participate in its configuration.

**Table 4. Symbian Analysis and Result**

| Evaluation Criteria | Symbian |
|---|---|
| Development Tools availability | Yes |
| Development Friendliness | No |
| Installation Vector | Multiple |
| Application Portability | No |
| Application Testing | No |
| Aplication Removal | Yes |
| Unofficial Repositories | Yes |
| Distribution Cost | No |
| API Restrictions | Yes |
| Application Signing | Yes |

**(Source: https://www.symbiansigned.com/app/page)**

## 4.4 Case Study: Apple iOS

The implementation of our simple location tracking attack was successfully completed on Apple's iOS. The implementation was tested on emulators iOS. The installation of applications to devices running the iOS system is only possible via Apple's App store. Hence, for running the devices of the official version of iOS unofficial application repositories are not available.

In our case study, the user would only be prompted to permit access to location data. But, the user is expected to confirm access to location data as the application is providing location based services. Afterward, the data would be transfermed to the attacker's remote server without the user noticing it from Apple devices and their removal from the application repository it has developed security mechanism allowing the remote detection of malcious application. This is a consequential post installation security mechanism in case Apple the user become suspicious of the malware software.

**Table 5. Apple iOS Analysis and Result**

| Evaluation Criteria | Apple iOS |
|---|---|
| Development Tools availability | Partial |
| Development Friendliness | No |
| Installation Vector | Restricted |
| Application Portability | Yes |
| Application Testing | Yes |
| Aplication Removal | Yes |
| Unofficial Repositories | No |
| Distribution Cost | Yes |
| API Restrictions | No |
| Application Signing | Yes |

**(Source: http://developer.apple.com/devcenter/ios/)**

## 4.5 Case Study: Windows Mobile

On the installation vector of applications the security model of windows mobile does not impose restrictions furthermore, even if they are downloaded from a source outside microsoft's official reposotory application are able to be installed on the devices. Hence, the attackers does not have application distribution costs. Moreover, for malicious behaviour the application is not being tested, since it is not distributed by Microsoft distribution services. Nevertheless, the application removal mechanism applied by Microsoft may be used for the automated removal of the implemented malware application. To sum up, in windows mobile the feasibility of our malware attack depends on the device configurations regarding the security model, the user authorization at installation time and the automated application removal security mechanism.

**Table 6 Windows Mobile Analysis and Result**

| Evaluation Criteria | Window Mobile |
|---|---|
| Development Tools availability | Yes |
| Development Friendliness | Yes |
| Installation Vector | Multiple |
| Application Portability | Yes |
| Application Testing | No |
| Aplication Removal | Yes |
| Unofficial Repositories | Yes |
| Distribution Cost | No |
| API Restrictions | No |
| Application Signing | No |

**(Source: http://msdn.microsoft.com/en-us/windowsmobile/dd569132.aspx)**

## 4.6 Overview of the Case Study

The security model of a Smartphone operating system has two contradicting goals. On the one hand concerning the execution of 3rd party applications on their devices it must provide users with security assurance. On the other hand, it must provide the developers with a security system where on the other hand,

consumers are willing to install new application and on the other, to implement new application it is easy and efficient under certain circumstances the proof of concept exercise demonstrated that, the security model of all available Smartphone platform would not counter a location tracking attack. Moreover, the attack on all Smartphone platforms it showed that it is possible to easily implement, except from Symbian and iOS.

The reasons of the implementation failure on Symbian were not security related. They were related with the developer's programming skills and Symbia's structured API documentation. In all other platform the implementation was efficiently and effectively completed by average programmers and tested on the officially provided emulators.

For malicious behaviour application testing cannot be avoided only on Apple's iOS. Moreover, having strict installation requirements iOS was the only platform. Only Windows Mobile allowed, among the examined platform under some security model configurations, the execution of unsigned applications. Yet to the Smartphone holder on the examined platforms the digital signature process gives different security assurance. On the other hand, the user was not found having any control on the API running in the device, on some platforms.

Based on the comparison of different mobile operating systems Android, Blackberry & Windows outperformed. Development friendliness Apple iOS has application capability and don't unofficial repositories to make it a potential option for secure operating system. The symbian's restricted vector makes a good contender. However android's free distributed makes a good option for selection.

**Table 7 Overview of Malware Implementation**

| **Evaluation Criteria** | **Android** | **Blackberry** | **Symbian** | **Apple iOS** | **Window Mobile** |
|---|---|---|---|---|---|
| Development Tools availability | Yes | Yes | Yes | Partial | Yes |
| Development Friendliness | Yes | Yes | No | No | Yes |
| Installation Vector | Multiple | Multiple | Multiple | Restricted | Multiple |
| Application Portability | Yes | Yes | No | Yes | Yes |
| Application Testing | No | No | No | Yes | No |
| Aplication Removal | Yes | No | Yes | Yes | Yes |
| Unofficial Repositories | Yes | Yes | Yes | No | Yes |
| Distribution Cost | No | Yes | No | Yes | No |
| API Restrictions | Yes | Yes | Yes | No | No |
| Application Signing | Yes | yes | Yes | Yes | No |

# 5. CONCLUSION

Smartphone devices are multi-purpose portable devices enclosing a vast amount of third party applications that augment the device's functionality. The existing Smartphone security models facilitate mechanisms and processes controlling the installation and execution of third party applications. Even so, the sufficiency of the adopted security mechanisms seems to be controversial. Their ability to protect the devices from being a privacy attack vector from developers, such as undergraduate and postgraduate computer science students, is proven to be unclear. Our paper (a) proposes evaluation criteria assessing the development of Smartphone malware, and (b) provides a comparative case study analysis where the implementation and distribution of proof of concept location tracking malware is attempted in the current Smartphone platforms. This critical review study has given an insight that all Smartphone platforms would not stop average developers from using them as privacy attack vectors, harvesting data from the device without the user's knowledge and consent. It also showed the easiness of malware application development by average programmers that have access to the official tools and programming libraries provided by Smartphone platforms. A silver bullet solution against similar attack scenarios is not available. Some of the possible steps that can reduce the possibility of being attacked is "prevention is better than cure". However, the following steps can be a possible way-out to reduce/avoid a potential malware outbreak in smartphones:

a) User awareness, i.e. informing user about security and privacy risks in Smartphone platforms; and
b) Providing secure application distribution in Smartphone platforms.

The scope for future research in this domain is in its infancy stage. This critical review is just a 'tip of an iceberg'. However, the case study can be repeated with more developer attributes taken into consideration and for analyzing more security attributes as well.

# 6. REFERENCES

[1] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici,S. Dolev, and C. Glezer. Google Android: A Comprehensive Security Assessment. IEEE Security & Privacy, 8(2):35 – 44, Mar. 2010.

[2] G. Lawton, "Is It Finally Time to Worry about Mobile Malware?" IEEE Computer, vol. 41, no. 5, 2008.

[3] J. Anderson, J. Bonneau, and F. Stajano. Inglorious Installers: Security in the Application Marketplace. In Proceedings of the 9th Workshop on the Economics of Information Security, 2010.

[4] J. Bickford et al., "Rootkits on Smart Phones: Attacks, Implications and Opportunities," in Workshop on Mobile Computing Sys. and Appl. (HotMobile'10). ACM, 2010.

[5] Egele, M., Kruegel, C., Kirda, E., Vigna, G.: Pios: Detecting privacy leaks in iOS application. In: Network and distributed System Security Symposium (2011).

[6] Enck, W., Gilbert, P., Chun, G., Cox, P., Jung, J., McDaniel, p., Sheth, N.:Taintdroid: an information-flow tracking system for real time privacy monitoring on smartphone. In: 9th USENIX Symposium on Operating System Design and Implementation (OSDI), pp. 1-6. USENIX Association (2010).

[7] J. Oberheide and F. Jahanian, "When Mobile is Harder Than Fixed (and Vice Versa): Demystifying Security

Challenges in Mobile Environments," in Workshop on Mobile Computing Systems and Applications (HotMobile), February 2010.

[8] M. Egele et al., "PiOS: Detecting Privacy Leaks in iOS Applications," in Network and Distributed System Security Symposium (NDSS), Feb. 2011.

[9] N. Leavitt, "Mobile Phones: The Next Frontier for Hackers?" IEEE Computer, vol. 38, no. 4, 2005.

[10] Gartner: Market Share: Mobile Communication Devices by Region and Country, 3Q11. Technical Report(2011).

[11] Hogben, G., Dekkar, M .: Smartphone : Information Security Risks, Opportunities and Recommendation for users. Technical Report. ENISA (December 2010).

[12] Hypponen, M.: Malware goes Mobile. Scientific American 295(5),70-77(2006).

[13] iOS dev center, http://developer.apple.com/devcenter/ios/

[14] java code signing keys, http://us.blackberry.com/developers/javaappdev/codekeys.jsp

[15] P. Zheng and L. M. Ni, "The Rise of the Smart Phone," IEEE Distributed Systems Online, vol. 7, no. 3, 2006.

[16] Lineberry, A., Richardson, D., Wyatt, T.: These aren't the Permissions you are looking for. Technical Report, DEFCON(2010).

[17] W. Enck, M. Ongtang, and P. McDaniel. Understanding Android Security. IEEE Security & Privacy,7(1):50–57, 2009.

[18] Stephen Smalley, Robert Craig, "Security Enhanced (SE) Android: Bringing Flexible MAC to Android, april 23, 2013.

[19] Mobile Privacy, http://www.gsmworld.com/our-work/public-policy/mobile_privacy.htm.

[20] Mylonas, A., Dritsas, S., Gritzalis, D.: A Secure Smartphone Security Evaluation: The Malware Attack Case. In: Samarati, P ., lopez, j (eds) International Conference of Security and Cryptography (SECRYPT'11), pp.25-36. Scitechpress(2011).

[21] Mylonas, A., Tsoumas, B., Dritsas, S., Gritzalis, D.: A Secure Smartphone Application Roll-out Scheme. In: furnell,s., lambrinoudakis,C., pernul, g (eds) Trust, Privacy and Security in Digital Business (TrustBus). LNCS vol. 6863. Pp. 49-61. Springer berlin/Heidelberg(2011).

[22] Fu Cai, Huang Qingfeng, Han LanSheng, Shen Li and Liu Xiao-Yang, "Virus propagation power of the dynamic network, springer EURASIP, 2013.

[23] Security And Permission, http://developer.android.com/guide/topics/security/security.htm.

[24] Andre Egners, Bjorn Marscholleck and Ulrike Meyer, "Messing with Android's Permission Model, IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communication page no : 505-514, april 2012.

[25] security overview, http://docs.blackberry.com/en/developers/deliverables/21091/security_overview_1304155_11.jsp.

[26] Security Policy Settings, http://msdn.microsoft.com/en-us/library/bb416355.aspx.

[27] Seriot, N .: iphone Privacy. Technical Report, Black Hat DC(2010).

[28] Xuetao Wei, Lorenzo Gomez, Ililian Neamtiu, Michalis Faloutsos, "Permission Evolution in the Android Ecosystem, ACM ACSAC, Dec: 3-7, 2012.

[29] David Bareera, Jeremy Clark, Daniel McCarney, "Understanding and Improving App Installation Security Mechanisms through Empirical Analysis of Android, ACM SPSM oct 19, 2012.

[30] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, David Wagner, "AdDroid: Privilege Separation for Applications and Advertisers in Android, ACM CCS, 2012.

[31] Windows Mobile Device Security Model, http://msdn.microsoft.com/en-us/library/bb416353.aspx

[32] Pern Hui Chia, Yusuke Yamamoto, N. Asokan, "Is this App Safe? A Large Scale Study on Application Permissions and Risk Signals", ACM, IW3C2 april 16-20, 2012.

[33] http://android.developers.blogspot.com/2010/06/exercising-our-remote-application.html

[34] http://developer.android.com/resources/dashboard/platform-versions.html

[35] http://developer.android.com/guide/topic/security/security.html

[36] http://us.blackberry.com/developers/javaappdev/ codekeys.jsp.

[37] http://docs.blackberry.com/en/developers/deliverables/21091/ Security_overview_1304155_11.jsp

[38] https://www.symbiansigned.com/app/page

[39] http://developer.apple.com/devcenter/ios/

[40] http://developer.apple.com/programs/ios/

[41] http://msdn.microsoft.com/en-us/windowsmobile/dd569132.aspx