2015 12th International Conference on Information Technology - New Generations

Android vs. iOS Security: A Comparative Study

Ibtisam Mohamed, PhD Candidate
Department of Computer Science
University of Denver
Denver, CO 80208, USA
ibtisammarai@gmail.com
Dhiren Patel, Visiting Professor
dhiren29p@gmail.com

***Abstract*—The massive adoption of mobile devices by individuals as well as by organizations has brought forth many security concerns. Their significant abilities have resulted in their permeating use while correspondingly increasing their attractiveness as targets for cybercriminals. Consequently, mobile device vendors have increasingly focused on security in their design efforts. However, present security features might still be insufficient to protect users' assets. In this paper, factors that influence security within the two leading mobile platforms, Android and iOS, are presented and examined to promote discussion while studying them under one umbrella. We consider various factors that influence security on both platforms, such as application provenance, application permissions, application isolation, and encryption mechanisms.**

Keywords: Android; iOS; Application store; Mobile Platform; Security.

I. INTRODUCTION

Mobility and connectivity offered by smartphones and tablets make them an essential part of our modern life. Many types of attacks have compromised mobile devices such as worms and viruses, posting malicious applications to the application stores, obtaining sensitive information, modifying data on a device, sending unwanted SMS massages, gaining location information, etc.

According to Canalys [1, 2], the number of smart mobile devices (notebooks, tablets and smart phones) shipped worldwide for the first quarter of 2013 reached 308.7 million units, which represents a year-on-year growth of 37.4%, and their combined market share is expected to grow to 66% by 2016.

We restrict our study to the two most leading mobile platforms: Android and iOS. It should be noted that these platforms are updated quite frequently and that strides are made to improve security. Nevertheless, it has been our experience that general trends tend to persist in terms of statistics regarding attacks. It should be noted that this paper is not intended to compare and contrast the relative vulnerabilities of iOS and Android. It is intended to give a broad understanding of the present state of mobile security efforts and the varying concepts that come from the two leading platforms. People are continuously attracted to the number of applications (apps) available in the application stores of the two most popular platforms. Figures 1 and 2 illustrate some of the statistics regarding their popularity per Canalys [1,3].

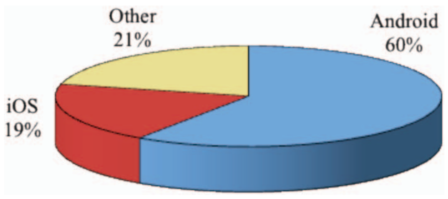

Figure 1: Mobile Devices Shipped

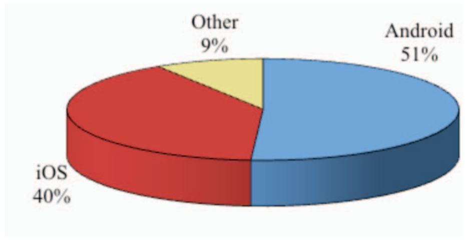

Figure 2: Apps Downloaded in First Quarter 2013

As of May 2013, there were over 800,000 third-party applications in the Apple App Store and about the same number on Google Play for the Android [4]. The large user base of mobile devices has made them attractive targets for attackers.

As mobile devices are holding very sensitive information (such as login credentials), the users' confidentiality is at risk of being breached. Cybercriminals have found many sophisticated ways to perform malicious activities on mobile devices. Cybercrimes such as identity theft, information theft, distribution of malwares, and financial fraud have become a real threat to individuals, organizations, and service providers. Nevertheless, there are defenses that can be put in place to minimize and mitigate these threats.

The continually changing contextual and situation usage of mobile devices poses a significant challenge on those developing mobile security. Devices at home, such as desktop computers and Blu-ray players are not as frequently subjected to public networks, theft risks, and as wide of a range of situational possibilities for skimming, scanning, and outright attack. As the growth of use, flexibility, and necessity of mobile devices continues, opportunities and reasons for opportunistic security compromise will inevitably follow. The risks may take forms as yet unpredicted.

Lack of user awareness of risks involved creates a challenging environment for vendors to put all the defensive measures in place. The data security on a device depends on the mobile platform that is used. Each of the mobile operating systems has its own security features to mitigate security risks [5].

The rest of the paper is organized as follows: section 2 formulates the approval process for third-party apps (as followed by Apple and Google). Section 3 illustrates the application permissions used by Apple and Google. Section 4 shows a comparison between isolation systems in iOS and Android. Encryption techniques for improving security defenses used by the two platforms are discussed in section 5. The number of vulnerabilities and malware affecting these platforms are illustrated in section 6, and a conclusion is drawn in section 7.

II. APPLICATION PROVENANCE

Provenance is a method where each third-party application is scrutinized for its intended functionality and security. Before posting an app into the applications marketplace, mobile platform vendor validates the provenance and authenticity of this application. After the application is checked, it is stamped with its author's identity. It is digitally signed with a vendor-issued digital certificate to make it tamper resistant. Users usually download an app based on its author identity [6].

A. iOS Application Provenance:

App Store, the official Apple marketplace, is the only way for iOS developers to distribute an application. Before distributing an application via the app store, iOS developers first need to register with Apple [7]. Apple has a licensing agreement where each application submitted by a third-party developer is tested for its privacy and security violations. This step is known as the vetting process and it takes up to two weeks. If the application does not violate the licensing agreement, it is accepted, signed digitally, and published in the App Store [3,6,8].

Each app needs to be digitally signed with a digital certificate issued by Apple, and the identity of the developer is embedded into the app. The process of digitally signing an app aims to guaranty that both the identity of the app's developer and the app are not modified or tampered with. By making software developers go through the certification process, Apple ensures that those developers become more responsible in making high quality apps. Furthermore, before posting an app on the App Store, Apple checks each app for malicious behavior and violations of Apple's policies [6].

This vetting process deters hackers from writing malicious apps to be sold in the App Store or even try to attack a published app. They first need to give their identities in order to register with Apple and acquire a signing certificate, which adds another defensive measure against any malicious activity. Also, once the app is submitted to the App Store and certified by Apple, any attempt to modify the app would result in breaking the seal on the digital signature of the app [7].

Nonetheless, according to Symantec, these security defenses are not absolutely effective. In fact, hackers can use stolen identities to register with Apple to post harmful applications [6]. A skillful attacker may be able to insert a malicious code without being caught during the certification process. Additionally, Apple does not reveal its vetting process to the public, so it is hard to evaluate its efficacy. There are examples where a malicious application passed the approval process. When Apple knows about its offending behavior, it removes it from the App Store. For instance, in 2009, Apple removed all applications developed by Storm8 from the App Store when it discovered that these applications were collecting users' personal information [9].

Even with the efforts made to improve the security of the App Store, there are still many tradeoffs. Aside from the fact that the vetting process could deter some would-be developers from proceeding with new or improved developments, thus slowing maturation of applications and encouraging security setbacks. As with most things that Apple does in a proprietary fashion, lack of details and potential changes regarding vetting can prove to be a double-edged sword. On the one hand, it ensures that efforts are more difficult to circumvent the process. On the other, more resource drain and consequently financial burdens may be placed on developers who may misunderstand or be unaware of what is appropriate behavior for an app in certain circumstances. In addition, as the popularity and market-share of Apple grows, so recedes the age-old aegis that "no one bothers to attack an Apple." As iOS is placed in the hands of more people and the scope of its usage is widened, users of Apple mobile devices may need to become more security and risk savvy than in the past. This is a consequence of the tradeoff between resources, control, and generality of use.

B. Android Application Provenance:

Like Apple, Android requires apps to be digitally signed. However, the process of digitally signing an app for Android differs from Apple. Android does not require the applications' developers to register with Google Play and have Google-issued signing certificates. In fact, app developers who want to post their apps on Google are able to create as many signing certificates as they want without being monitored by Google [7].

However, the developers who are willing to distribute their apps through Google are asked to pay $25 as a fee using credit cards and wait for their apps to be certified. Here, Google uses this process to link a certain app with its digital certificate generated by the developer. Nonetheless, this linkage cannot be guaranteed since the attackers can use a stolen credit card to pay the fee [6,7].

One of the biggest security concerns with Android is that Google Play is not the only place for distributing apps to Android users. In fact, Android app developers can post their apps on any website on the Internet, where the apps are not evaluated by Google. Attackers take advantage of this open ecosystem and distribute harmful software [10].

Given this freedom, it is hard to ensure that the application's developer will not create an anonymous digital certificate and avoid being caught. Furthermore, hackers can use famous company names and place them into the certificate to trick users. More importantly, Google cannot totally prevent attackers from tampering with a legitimate app. An attacker can insert a malicious code into a legitimate app, generate a new anonymous certificate, and then distribute the malicious app into the Internet or into Android's marketplace, thus creating a Trojan horse [6,8].

All of the factors and concerns as to the dangers of the Android platform and its corresponding Google Play store have led to a balance of the security resource usage and concerns being placed into the hands of consumers. Being exposed to third-party threats and a relatively unrestricted marketplace requires consumers to be more savvy and careful when it comes to their app downloading and usage behaviors. Where to place trust is a tradeoff of the wide-spread use and form of the Android platform. Too much control leads to reduced usefulness and discourages development, while too much freedom leads to potentially-malicious opportunism. It is widely known that many Android apps, such as popular games, often request permissions that widely-overreach reasonable security parameters. Android users are left with the decision of where to place their trust. This may have leave developers with certain legal pitfalls as the lack of mechanisms preventing certain application behavior may allow unintentional violations of privacy laws.

III. APPLICATION PERMISSIONS

When an application is downloaded on a device, it can access the entire device resources based on the given permissions. Typically the user is informed about the permissions to which the application requires access. But, because most users are unaware of all the implications, they end up choosing the default choices, which may not be the best from the security point of view.

A. iOS Application Permissions:

In iOS, users are not often shown the permissions to which the application requires access. In fact, Apple gives third party applications a limited set of permissions that are required by the application sandbox, where each app is isolated from every other app on the system. Furthermore, iOS blocks access to many of a device's sensitive subsystems. Basically iOS does not make users responsible for making security decisions. The iOS isolation policy handles some of the permissions that the app needs without explicitly asking the user. However, in some cases permissions to access some resources have to be given by the user of the device [11]. The permissions that iOS's users are asked to give are as follows: accessing location data from the global positioning system (GPS) of the device, receiving remote push notifications from the Internet, starting an outgoing phone call, and sending an outgoing SMS or email message. When a user gives permission to an app to access some resources, the app can permanently access these resources. Nonetheless, the user is prompted any time an app needs to access these features [12].

B. Android Application Permissions:

Each Android application must explicitly inform the user about what privileges it requires. During the installation of an application on a device, a list of all permissions that the application needs is shown to the user. Then, the user needs to decide if he should continue installing the app. Here, the user has to decide whether or not the app actually needs these permissions [13]. For example, if a calendar application requires the Internet access permission, the user should understand the real intent behind this app. In fact, once an app is given permission, it is hard to guarantee that the application will not abuse the permission provided. Sometimes, if the user decides not to give permission to an app, it will stop installing. Once the application is installed with a given set of permissions, the app is allowed those permissions until it is deleted. Given that the significant onus of making security decisions is placed on the user, unless the user base is security-aware, this may not be a wise choice [14].

Attackers can misuse the permission system in Android. In fact, they attempt to use their social engineering skills to convince users to install their applications. Essentially, users are willing to get these apps at any cost, however without full permissions some apps refuse to install. So, users end up giving what the apps are asking for and then their devices will be under the attacker's control [15].

Android has about 100 built-in permissions that handle resource access, for instance, CAMERA for taking pictures, and INTERNET for accessing the Internet, etc. According to Symantec, applications can request different levels of permissions to access some subsystems (e.g. Network Subsystems: access networks, Messaging Systems: access emails in the inbox; outbox and SMS systems of the device; Global Positioning System: acquire the location of the device) [8,10].

Using such permissions, attackers can launch data loss attacks, distributed denial of service attacks, and data integrity attacks. For example, using device identifiers, the application can acquire some sensitive information such as the phone number and the device ID number. Cybercriminals can use this information to make cellular phone fraud [6].

The Android permission system does not seem to be effective because the security decisions are made by device users. The user is in-charge of judging whether an app is safe. Indeed, most users do not typically have the requisite knowledge to make these security decisions correctly [8].

IV. ISOLATION

During execution, each application on a device is isolated in a sandbox environment, i.e. each application is executed in its own environment and cannot modify any other applications.

A. iOS Isolation Mechanisms:

Each app is isolated from other apps on the iOS operating system. Apps on a device are prevented from escalating privileges or accessing the kernel of the operating system. Moreover, not every application can interact with every other application on the system or even know about their existence. In general, apps cannot modify or abuse other apps on the system [16]. The iOS operating system controls third-party applications and limits their influence over the device. In addition to isolating the apps from each other and from the kernel, the iOS operating system also isolates some apps from the in-out email boxes and the SMS of a device. For instance, apps are not allowed to send SMS without the users' involvement. However, iOS apps can also access some resources without asking the user for any permission. For instance, the iOS's applications can access the device calendar, video camera and files, the Wi-Fi connection, and the device's ID [12].

Separating each application from other applications and from the kernel does enhance the security. Some major attacks, such as web-based attacks and network-based attacks, are prevented by the application isolation. As a result of sandboxing, malware attacks can be mitigated and data loss can be prevented. But, as mentioned before, by default, applications on iOS can still access some resources on the system. Thus, an attacker can use a malicious application to steal private information on the device such as the device's unique ID, send email spam, or perform a of service (DoS) attack [6].

B. Android Isolation Mechanisms:

Android isolation system is similar to that of iOS, where each application is separated from other applications on the system. Here, each application is given permissions to access certain resources, and the isolation system prevents accessing resources beyond the approved permissions. Furthermore, each app is isolated from the system's kernel to prevent any compromised application from gaining administrator-level control. Given that protection, an attacker cannot use a malicious application to compromise other applications on Android [14].

However, using certain permissions, an application can get a list of applications that are on the device. Furthermore, an application can launch other applications on the system, such as the map application. Not only that, but also an app can check the programming logic of every application [14]. For example, the Secure Digital (SD) card, that has the user's downloaded programs, music, and video files, can be read and modified by an application without any restriction. Therefore, depending on the set of permissions given, an app can access some of the device's subsystems and launch various types of attacks [15].

V. ENCRYPTION

Mobile devices are at risk of being lost or stolen, which makes them vulnerable to data breaches. Mobile platform vendors have addressed this issue by encrypting data on mobile devices and the data sent and received from these devices. Encryption prohibits someone from accessing private information on a lost or a stolen device. To decrypt the data, a user needs to have a password or a secret key [17]. However, there is always a concern that such password or secret key may be stored insecurely on the device itself. Furthermore, the fact that remembered keys and passwords are more likely to be forgotten leads to users choosing overly simple ones that may make it into the dictionaries of certain applications (texting and searching for example). Of particular importance to this context is the upcoming prevalence of on-device fingerprint scanning capabilities.

As on-device fingerprint technologies are frequently small, relatively cheap, and not observed for proper usage, they can at times provide a false sense of security. On the one hand, fingerprints are relatively complex, not easily lost, and do not need to be stored on the device for usage. On the other hand, they may be stored right on the screen of the device in the form of residue and can be falsified easily in certain contexts. The public as a whole may not be largely aware of the ramifications of using fingerprints for device protection and encryption and might underestimate their limitations and pitfalls, leading to an overly-large and overly-broad trust being placed in their usage. User identification and authentication to map device to person is ever a concern and is one of the primary aspects of access limitation and authentication. While fingerprint technologies and other biometric solutions along with encryption are making strides on mobile platforms, there are still fundamental outlying architectural and protocol-related aspects yet to be resolved. Perhaps this is just a part of the maturation process. As technology developers and user understanding grows, capabilities and behavioral improvements will hopefully, ultimately follow.

A. iOS Encryption Mechanisms:

The encryption capabilities on iOS use layers of combined hardware and software technologies. First, iOS uses hardware Advanced Encryption Standard (AES) cryptographic accelerator to encrypt all data kept in the device's flash memory. The AES accelerator includes both keys, the device unique ID key (UID) and the device group ID key (GID). There are AES 256-bit keys built into the application processor during manufacturing. Each device has its unique UID, which lets the data to be cryptographically linked to a specific device. Furthermore, the UID derives some of the device-specific AES keys. These keys encrypt keychain items, file system metadata, and files. For instance, the key that protects the file system has a UID, so the files cannot be accessed if someone moves the memory chips from one device to another. All processors in each class of devices have a common GID. During restoration and installation of the system software, the GID is used to add an extra level of protection [18,19].

Besides the hardware encryption technologies built into iOS, Apple uses a system called Data Protection. It is used by iOS4 operating systems and beyond to further protect sensitive data on the device's flash memory. All files are encrypted with a unique File Key. Data Protection technology protects data while connecting a device to the

Internet or receiving phone calls, text, or email. It prevents decrypting sensitive data and downloading new information while the device is locked. Moreover, remote wiping is another level of security that is used by iOS to protect data on a device. For a rapid remote wipe operation, the embedded encryption accelerators are used for block-level encryption on the system and data partitions. By wiping a single encryption key, all of the device's data are rendered inaccessible. Therefore, in the case of a device being lost or stolen a user can send a remote wipe command in time to the device through Mobile Device Management (MDM) and protect all data on the device. In addition to these security features, iOS supports four-digit alphanumeric passcodes that unlock the device and protect access to certain data on a device. A user can configure a device to automatically discard the hardware encryption key and render the whole data unreadable if an incorrect passcode is entered more than 10 times [6,18,19].

Even with the reasonable level of protection provided by iOS, there are some defects. Background applications run and access the device storage, even when it is locked. iOS keeps a copy of the decryption key always to decrypt data needed by the background applications. Therefore, if a device has a malicious app, or if a legitimate app is compromised, or if the attacker has physical access to the device, an attacker can access and steal a user's data. Furthermore, using available tools, such as the jail-breaking tool, an attacker with physical access to a device could guess the iOS four-digit passcodes in less than 20 minutes [6,18].

B. Android Encryption Mechanisms:

Android has two encryption systems, which are the file-system based encryption and the Android KeyChain. The use of the user password is required by Android 3.0 and beyond, to offer full file-system encryption where all user data is encrypted in the kernel. The use of the user password is required by the file-system encryption. AES128 protects the encryption key by using a key derived from the user password; here, without the user password, access to the stored data is prevented. Android file-system encryption has to be manually activated by the user or enforced by the MDM rule. However, it is hard to guarantee that the file-system encryption configuration is enabled, since the system's security relies on the settings chosen by the user [20].

In addition to the file-system encryption, Android uses KeyChain to securely store credentials used by an app on the file-system. A developer is able to use KeyChain even without activating the file-system encryption system. Since Version 1.6, Android has applied a low-level credential store, which is a system daemon that uses Unix socket interface. Applications use credential store to keep Wi-Fi passwords, private key material and certificate chains. In Android 4.1, a hardware support was added to the KeyChain Application Programming Interface (API) to better protect the stored keys. An app's developer decides whether to use the KeyChain to store the credential or not. A user should choose a passcode to lock a device to enforce the security features of the KeyChain system, if applications use them [20].

As explained above, a device security depends on the user, the MDM, and the app's developer. Android's encryption system needs to be manually activated by the user or determined by the requirement set of the corresponding MDM-system. In addition, the level of complexity and the length of the passcode are influenced by the user choices and the appropriate MDM rules. Failing to activate the encryption system or choosing a strong passcode makes an attack on a stolen device very easy. Also, any bad choices made by the developer, such as storing the credentials unencrypted on the file system, can adversely affect the device's security [20].

VI. VULNERABILITIES & MALWARE

Mobile devices are vulnerable to attacks that can compromise the confidentiality, integrity, and availability of data saved on devices. Two indicators of risk are the number of vulnerabilities and the number of reported malware. The former indicates the number of weaknesses discovered in a platform that could potentially be used to compromise the platform, and the latter is the number of actual threats that were detected.

A. iOS Reported Vulnerabilities

Statistics by Common Vulnerabilities and Exposures (CVE) claimed that 408 of vulnerabilities were discovered in the iOS operating system during 2007-2014 in its various versions [21]. Furthermore, according to Symantec, the time that Apple took on average to patch a vulnerability was 12 days from the time it was reported. Most of these vulnerabilities were of lower severity, such as the ones that an attacker exploits to gain control of a single process. A few of the vulnerabilities were more severe and allow an attacker to take administrator-level control of a device [6].

Although iOS has a high number of vulnerabilities, a report by the U.S. Department of Homeland Security in June 2013 said that just 0.7% of malware attacks were targeting iOS operating systems [22].

B. Android Reported Vulnerabilities

Security vulnerabilities in Android range from low severity to higher severity. Statistics from Common Vulnerabilities and Exposures (CVE) claimed that 30 vulnerabilities were disclosed in the Android operating system during 2009-2013 in its various versions [21]. According to Symantec, the time it took Google to patch a vulnerability is eight days from the time it was reported [6].

Mobile malware has significantly increased, and writers of mobile malware are targeting mostly the Android platform. The most frequently targeted mobile platform in 2013 was Android with 79%, compared to 0% threats with iOS [23]. According to McAfee, a 30% increase in the number of attacks targeting the Android operating system was detected. Moreover, out of a total of 8,000 mobile malware, Android threats make up 7,000 [24]. Also, the U.S. Department of Homeland Security released a report in

June 2013 showing that Google's Android platform accounts for 79% of all mobile malware [22].

VII. CONCLUSION

The growing popularity and sophistication of mobile platforms have increased privacy and security concerns. To address these concerns, mobile platform vendors employ techniques such as application provenance, permission-based access control, encryption technique and isolation. The Android security approach has two major weaknesses. First, attackers can use anonymous digital certificates to sign their compromised apps and post them on the Internet. Moreover, attackers can redistribute a legitimate app with a new certificate after injecting it with malicious code. Second, Android's permission system relies on users to make security decisions, which makes them open to social engineering attacks. The devices' users grant whether or not to give permissions to an app. Unfortunately, most users are not capable of making sound security decisions.

On the other hand, iOS's security approach seems to be more resistant to attacks. First, the only place for iOS users to download apps is the App Store, where each app is tested for security violations. Second, Apple follows a strong signing model that prevents tampering with published apps. Also, the iOS permission model does not completely rely on the users' decisions. However, the Apple security approach with iOS with vetting is more restrictive and less conducive to developers. Where trust is placed and user knowledge is required seems to be one of the primary differences between the Android and iOS approach. Regardless of the platform, mobile security does not yet seem to be a concern of the past.

REFERENCES

[1] Canalys, (2013, May 9), "Smart mobile device shipments exceed 300 million in Q1," [Online], Available : http://www.canalys.com/newsroom/smart-mobile-device-shipments-exceed-300-million-q1-2013.

[2] Canalys, (2013, February 22), "Mobile device market to reach 2.6 billion units by 2016," [Online], Available: http://www.canalys.com/newsroom/mobile-device-market-reach-26-billion-units-2016.

[3] H. McCracken, (2013, April 16), "Who's winning, iOS or Android? all the numbers, all in one place," [Online], Available: http://techland.time.com/2013/04/16/ios-vs android/#ixzz2tpCMYgwa.

[4] Canalys, (2013, May 23), "Top iOS and Android apps largely absent on Windows Phone and BlackBerry," [Online], Available: http://www.canalys.com/newsroom/top-ios-and-android-apps-largely-absent-windows-phone-and-blackberry-10.

[5] Z. Benenson, O. Kroll-Peters, and M. Krupp, "Attitudes to IT security when using a smartphone," in Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on, IEEE, 2012, pp. 1179-1183.

[6] C. Nachenberg, "A window into mobile device security," Symantec Security Response, Symantec, 2011, pp.4-9.

[7] Z. Kazmi, F. Toni, J. A. Vila, and M. M. Marcos, "TASAM-Towards the Smart Devices App-Stores Applications Security Management Related Best Practices," in New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, IEEE, 2012, pp. 1-5.

[8] A. Shabtai, et al. "Google android: A comprehensive security assessment," IEEE security and Privacy, vol. 8, no.2, IEEE, 2010, pp.35-44.

[9] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting Privacy Leaks in iOS Applications," in NDSS, 2011, pp.1-15.

[10] S. Liebergeld and M. Lange, "Android security, pitfalls and lessons learned," in Information Sciences and Systems, Springer International Publishing, 2013, pp.409-417.

[11] C. Miller, "Mobile attacks and defense," IEEE Security and Privacy, vol.9, no. 4, IEEE, 2011, pp. 68-70.

[12] J. Han, et al. "Launching generic attacks on ios with approved third-party applications," in Applied Cryptography and Network Security, Springer, 2013, pp.272- 289.

[13] Y. J. Ham, H.W. Lee, J.D. Lim, and J.N. Kim, "DroidVulMon-Android based mobile device vulnerability analysis and monitoring system," in Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 7th International Conference on, IEEE, 2013, pp.26-31.

[14] L. Davi, A. Dmitrienko, A. R. Sadeghi, and M. Winandy, "Privilege escalation attacks on android," in Information Security, Springer, 2011, pp.346-360.

[15] A. Shabtai, Y. Fledel, and Y. Elovici. "Securing Android-powered mobile devices using SELinux," IEEE Security and Privacy, vol.8, no.3, IEEE, 2010, pp.36-44.

[16] T. Werthmann, R. Hund, L. Davi, A.R. Sadeghi, and T. Holz, "PSiOS: bring your own privacy & security to iOS devices," in Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, 2013, pp. 13-24.

[17] M.S. Ahmad, N. E. Musa, R. Nadarajah, R. Hassan, and N. E. Othman. "Comparison between android and iOS Operating System in terms of security," in Information Technology in Asia (CITA), 2013 8th International Conference on, IEEE, 2013, pp.1-4.

[18] D. A. Dai Zovi, "Apple iOS 4 security evaluation," Black Hat USA, 2011, pp.1-29.

[19] Apple, (2014, Decemper 15)," iOS security," [Online], Available: http://www.cse.wustl.edu/~jain/cse571-14/ftp/ios_security.pdf.

[20] P.Teufl, et al."Android encryption systems," in Privacy and Security in Mobile Systems (PRISMS), 2014 International Conference on, IEEE, 2014, pp. 1-8.

[21] Common Vulnerabilities and Exposures (CVE), (2014), [Online], Available: http://www.cvedetails.com/.

[22] U.S. Department of Homeland Security, (2013, July 23), "Threats to mobile devices using the Android operating system," [Online], Available:

http://info.publicintelligence.net/DHS-FBI-AndroidThreats.pdf.

[23] Symantec, (2014, April), "Internet security threat report 2014," vol.19,[Online],Available: http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf.

[24] McAfee, (2013, November). "McAfee 3rd Quarter 2013 Threat Report,"[Online],Available: http://malwarelist.net/2013/11/20/mcafee-3rd-quarter-threat-report-released/.