Emerging security threats for mobile platforms

Emerging Security Threats for Mobile Platforms

G. Delac*, M. Silic* and J. Krolo**
* Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
**Google Inc., New York, USA
{goran.delac, marin.silic}@fer.hr, jakov@google.com

Abstract - The proliferation of smart-phone devices, with ever advancing technological features, has brought the issue of mobile device security back into focus. Mobile devices are rapidly becoming attractive targets for malicious attacks due to significant advances in both hardware and operating systems. The modern mobile platforms, like Android, iOS and Symbian, increasingly resemble traditional operating systems for PCs. Therefore, the challenges in enforcing smart-phone security are becoming similar to those present in PC platforms. By installing malicious content, smart-phones can be infected with worms, trojan horses or other virus families, which can compromise user's security and privacy or even gain complete control over the device. Such malicious content can easily spread due to advances in mobile network technologies which provide smart-phones with capability of constant Internet connection over 3G or Wi-Fi networks. Additionally, the improvements in smart-phone features introduce new types of security concerns. By compromising mobile OS, malicious applications can access voice-recording devices, cameras, intercept SMS messages or gain location information. Such security breaches severely compromise user's privacy. In this paper we present an analysis of contemporary mobile platform threats and give an in-depth overview of threat mitigation mechanisms built into state of the art mobile operating systems.

I. INTRODUCTION

In recent years the expanding mobile hand-held device market is becoming an increasingly attractive target for malicious attacks. According to recent security reports [1][2], the number of possible malicious exploits and executed attacks is going to surge in 2011. This trend can be attributed to two key factors: the ever increasing user base and the emergence of smart-phone technology. The size of mobile device market is clearly visible from the latest reports issued by the ITU [3] which indicate that by the end of the 2010 there will be an estimated 5.3 billion mobile phone users in the world. Although malicious exploits for mobile phones have been steadily developing over the last decade [4][5], the constraints in both hardware and operating systems have limited the attacks both in their scale and impact. Therefore, the endorsement of smart-phone technology, which provides more computing power and functionality, is proving to be a turning point in development of malicious exploits for mobile hand-held devices. The estimates indicate that the market share of smart-phones in the US could exceed 50% of the total mobile hand-held device market by the end of 2011 [6]. Since the rest of the developed world is showing similar trends, a significant increase in smart-phone user base is expected over the next few years, thus making this platform an attractive target for malicious attacks. In addition, the smart-phones are starting to resemble PCs both in their capabilities and the way people use them. The smart-phone mobile platforms, like Android, iOS, Blackberry OS, Symbian and Windows Mobile are increasingly resembling PC operating systems[7]. Thus, the standard malicious attacks for PCs, like worms and trojans, as well as attack vectors, like the Internet access, are becoming applicable to the mobile platforms. Since the modern mobile platforms can be installed on devices other than smart-phones, like tablets or other appliances, the same security issues exist. We will shortly refer to all devices with a mobile platform installed as mobile devices in the rest of this paper.

Although rich in basic features, mobile platforms can be extended by installing applications in much the same way it is possible for PC operating systems. Since applications can originate from third party providers, they present an opportunity for introduction of malicious exploits. Apart from utilizing computing power provided by mobile devices, the attackers are starting to target the data. This is due to the fact that the smart-phones are becoming storage units for personal information through use of various social networking applications, personal organizers and e-mail clients.

In this paper we present an attacker-centric threat model for mobile platforms. The threat model addresses three key issues of mobile device security: attacker's goals, attack vectors and mobile malware. First, it defines the motives for attacking mobile platforms in order to identify the attacker's interests and potential targets. The attack goals focus on the motives introduced by modern mobile platforms and devices. Second, the model incorporates the attack vectors in order to present possible entry points for malicious content on mobile devices. Finally, the model considers threat types applicable to mobile platforms if the presented attack vectors are successfully utilized.

Since we consider the mobile applications to be the most convenient method for performing malicious attacks, we analyze the security model implemented by two widely spread platforms: the Android by Google and the iOS by Apple. The analyzed mobile platforms differ in their approaches to enforcing application security. The Android isolates applications in order to prevent them from interfering with other applications or the operating system. On the other hand, the iOS applications are screened for malicious intentions by code reviewers, thus allowing implementation of simpler security mechanisms. In order to demonstrate how Android's permission based security model could be breached, we present a fictive mobile application.

The rest of the paper is structured as follows: In Section 2 we present an attacker-centric threat model for mobile platforms. Overview of the Android security model is presented in Section 3 followed by description of iOS security model in Section 4. An example of a malicious application for Android is presented in Section 5. Section 6 concludes the paper.

II. Threat model for mobile platforms

In order to present a broad overview of challenges facing mobile devices' security, we present an attacker-centric threat model for mobile platforms. We analyze attacker's goals and motives as well as delivery methods and attack strategies. Therefore, the threat model is divided into three sections: attack goals, attack vectors and mobile malware.

A. Attack Goals

In this subsection we present three basic motives for breaching mobile device's security. The first two goals described are covert, while the latter is harmful. Covert approach to executing an attack is to perform malicious operations while avoiding user's detection. The goal of such attacks is to disrupt the operation of the device as little as possible while performing activities useful to the attacker. On the other hand, harmful attacks are aimed at disrupting the normal operation of a mobile device.

1) Collect Private Data

Since the mobile devices are in effect becoming storage units for personal data, they are an attractive target for breaching user's privacy. The attackers target both the confidentiality and integrity of stored information. A successfully executed attack can empower the attacker with ability to read SMS and MMS messages, e-mail messages, call logs and contact details. Furthermore, the attacker can intercept or send fake SMS, forward e-mails to alternative inboxes, and access the information from personal organizers and calendars. Additional information can be gathered by reading Instant Messaging client logs, data stored by applications used to access social networks or data stored by browsers. Any other data located in device's memory or on SD card, like documents, photos or videos, could also be compromised [8].

In addition, tapping into phone's basic hardware features provides an opportunity to collect additional data from user's surroundings. By utilizing the voice recording hardware, the attacker can turn the infected mobile phone into a listening device. Accessing the camera provides an opportunity to take photos or record video of the user's surroundings. Additional momentum to compromising user's privacy can be achieved by exploiting the location information. Mobile devices can provide location information by utilizing the GPS module, or by triangulating the position using the service provider's network infrastructure.

2) Utilize Computing Resources

The increase in computing resources is setting the contemporary mobile devices into focus for malicious attacks with aim to covertly exploit the raw computing power in combination with broadband network access. For example, high end mobile devices have CPU operating frequencies in excess of 1GHz, and physical memory well over 512MB. In addition, multicore processors for mobile devices are under development. Combined with high speed Internet links, mobile devices are becoming attractive for malicious exploits, such as deployment of botnets[9].

3) Harmful Malicious Actions

Harmful malicious actions are aimed at generating device user's discomfort rather on performing useful operations for the attacker. Although such attacks are usually easily discoverable, they are aimed at inflicting as much damage as possible. The attacks can range from data loss to draining the devices battery[10] and other resources, like generating huge network traffic. Ultimately, by gaining access to critical systems the attacker could disable the device permanently, i.e. brick the device.

B. Attack vectors

Mobile platforms provide multiple attack vectors for delivery of malicious content. We classify the attack vectors into four categories: mobile network services, Internet access, Bluetooth, and access to USB and other peripheral devices.

1) Mobile network services

Cellular services like SMS, MMS and voice calls can be used as attack vectors for mobile devices. For example, SMS and MMS messages can be used to deliver malicious content and to maintain the communication with the attacker. This is especially applicable to MMS messages as they support rich content which allows the attacker to embed hidden XML messages [11]. Furthermore, the cellular services provide opportunities for phishing attacks. Phishing is an attack strategy in which the attacker gains sensitive information from the user by presenting itself as a trustworthy entity. Two basic phishing attacks over mobile networks exist: smishing and vishing. Smishing[12] is a phising attack executed using SMS messages. The attacker uses SMS to send URL links that when clicked automatically open a browser window rendering the device vulnerable to attack. On the other hand, vishing[12] attack is carried out using voice calls. By masking the true voice call id, the attacker can trick the user into calling a certain number. The attacker can then gain sensitive information from the user by pretending to be a trustworthy entity, like a bank or insurance company.

2) Internet access

Mobile devices can access the Internet using Wi-Fi networks or 3G/4G services provided by mobile network operators. Although such high speed Internet connections ensure comfortable browsing, they also expose the mobile devices to the same threats as PCs. Since mobile devices are usually constantly switched on, they can maintain a continuous connection to the Internet. However, prolonged connection to the Internet also increases the chances of a successful malicious attack. The attack probably increases if the device is connected to a public network over a Wi-Fi hotspot. In addition, the attackers can use multiple emerging Internet services to spread malicious content. For example, social networks, like Tweeter or Facebook, are commonly used to share URL links in order to indicate items of interest on the Web. Since links sometimes become long and unpractical to share, the URL shortening services are becoming a common way for reducing the link size. For example, the URL: http://www.mipro.hr/MIPRO2011.ISS/ELink.aspx could be shortened to: http://goo.gl/tL20e .

Since the shortened link completely replaces the original, it is not possible to find out the destination without clicking on the link. By spreading the links over social networks, using previously compromised devices, the attackers can easily fool the users into clicking the harmful links. This way the attackers can trick users into downloading malicious content or navigating them to phishing sites.

3) Bluetooth

Bluetooth attacks are a method used for device-to-device malware spreading. Once the two devices are in range, the compromised device pairs with its target by using default Bluetooth passwords. When the connection is established, the compromised device sends malicious content. However, the Bluetooth is a limited attack vector for injecting malicious content due to several security factors. First, the mobile devices usually are not set as discoverable by default and the period in which they can be discovered is limited. Second, the user has to confirm the file transfer and then manually install the file.

4) USB and Other Periphetals

Apart from the mentioned attack vectors, mobile devices could be compromised by using other connections, like widely spread USB. The USB connection in commonly used to synchronize the mobile device with a personal computer. If the software used to synchronize the mobile device was compromised, the attacker could access private information and install malicious applications on the device. In addition, since some mobile platforms allow the device to connect as USB storage device, traditional USB malware could be applied.

C. Mobile Malware

Since the mobile platforms increasingly resemble traditional operating systems, the security threats characteristic for PCs are migrating to mobile devices. In this subsection we give a brief overview of the most common mobile malware. However, the actual attacks usually combine multiple variants of the presented mobile malware.

1) Trojan horse

By deploying malicious mobile applications the attacker could gain control over the device. Such applications usually perform some useful functionality while running malicious activities in the background. This way the Trojan can be used to gather private information or to install other malicious applications like worms or botnets. In addition, Trojans can be used to commit phishing activities. For example, a false banking application could collect sensitive data from the user. Such applications can easily spread through unsupervised application stores or through social networks.

2) Botnet

Botnet is a set of compromised devices which can be controlled and coordinated remotely. This attack strategy is used to utilize the computing power of compromised devices in order to commit various activities ranging from sending spam mail to committing DOS attacks. An example of a botnet designed specifically for mobile devices is Waledac[11]. Waledac uses SMS and MMS messages to exchange the data between nodes therefore enabling the botnet to remain active even if the nodes are not connected to the Internet.

3) Worm

Worm is a self-replicating malicious application designed to spread autonomously to uninfected systems. This type of malware has been ported to mobile platforms since the introduction of Cabir[12]. Cabir is a worm designed to attack Symbian S60 devices by spreading through Bluetooth links. A more recent example of a worm type malware for mobile devices is Ikee.B[13] which is used to steal financially sensitive data from jailbroken iPhones.

4) Rootkit

Rootkit is a malicious application which gained rights to run in a privileged mode. Such malicious applications usually mask their presence from the user by modifying standard operating system functionalities. Although no current rootkit type threats for mobile devices exist, recent research efforts [14] indicate the potential of this attack strategy and classify it as an emerging threat to mobile security.

In the rest of this paper we will focus on mobile apps as delivery methods for malicious activates. This threat type is characteristic for state-of-the-art mobile platforms, since the apps are used to augment basic functionalities provided by the operating system. Furthermore, rapid spread of mobile applications, both in their number and overall number of downloads through application stores and other sources, provides appealing opportunities for injecting and spreading malicious activities.

III. ANDROID SECURITY MODEL

Android is an application execution platform for mobile devices comprised out of an operating system, core libraries, development framework and basic applications [15]. Android operating system is built on top of a Linux kernel. The Linux kernel is responsible for executing core system services such as: memory access, process management, access to physical devices through drivers, network management and security. Atop the Linux kernel is the Dalvik virtual machine along with basic system libraries. The Dalvik VM is a register based execution engine used to run Android applications. In order to access lower level system services, the Android provides an API through afore mentioned C/C++ system libraries. In addition to the basic system libraries, the development framework provides access the top level services, like content providers, location manager or telephony manager. This means that it is possible to develop applications which use the same system resources as the basic set of applications, like built-in web browser or mail client. However, such a rich development framework presents security issues since it is necessary to prevent applications from stealing private data, maliciously disrupting other applications or the operating system itself. In order to address the security issues, the Android platform implements a permission based security model, as demonstrated in Figure 1.

The model is based on application isolation in a sandbox environment [16]. This means that each application executes in its own environment and is unable to influence or modify execution of any other application. Application sandboxing is performed at the Linux kernel level. In order to achieve isolation, Android utilizes standard Linux access control mechanisms. Each Android application package (.apk) is on installation assigned a unique Linux user ID. This approach allows the Android to enforce standard Linux file access rights. Since each file is associated with its owner's user ID, applications cannot access files that belong to other applications without being granted appropriate permissions. Each file can be assigned read, write and execute access permission. Since the root user owns system files, applications are not able to act maliciously by accessing or modifying critical system components. On the other hand, to achieve memory isolation, each application is running in its own process, i.e. each application has its own memory space assigned. Additional security is achieved by utilizing memory management unit (MMU), a hardware component used to translate between virtual and physical address spaces. This way an application cannot compromise system security by running native code in privileged mode, i.e. the application is unable to modify the memory segment assigned to the operating system.

The presented isolation model provides a secure environment for application execution. However, restrictions enforced by the model also reduce the overall application functionality. For example, useful functionalities could be achieved by accessing critical systems like: access to network services, camera or location services. Furthermore, exchange of data and functionalities between applications enhances the capabilities of the development framework. The shared user ID and permissions are two mechanisms, introduced by the Android, which can be used to lift the restrictions enforced by the isolation model. The mechanisms must provide sufficient flexibility to the application developers but also preserve the overall system security. As presented in Figure 1, two applications can share data and application components, i.e. activities, content providers, services and broadcast receivers. For example, an application could run an activity belonging to other application or access its files.

The *shared user ID* allows applications to share data and application components. In order to be assigned a *shared user ID* the two applications must be signed with the same digital certificate. In effect, the developers can bypass the isolation model restrictions by signing applications with the same private key. However, since there is not a central certification authority, the developers are responsible to keep their private keys secure. By sharing the user ID, applications gain the ability to run in the same process.

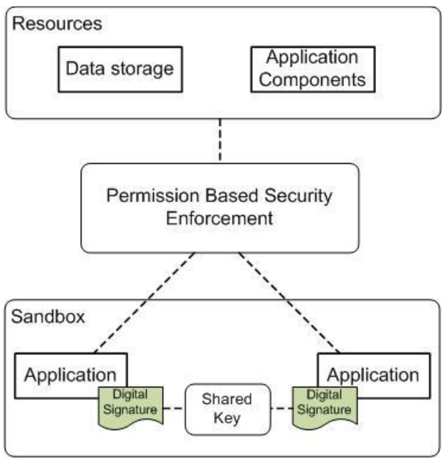

Figure 1. Android Security Model

The alternative to the *shared user ID* approach is to use the Android permissions. In addition to sharing data and components, the permissions are used to gain access to critical system modules. Each android application can request and define a set of permissions. This means that each application can expose a subset of its functionalities to other applications if they have been granted the corresponding permissions. In addition, each application can request a set of permissions to access other applications or system libraries. Permissions are granted by the operating system at installation and cannot be changed afterwards. There are four types of permissions: normal, dangerous, signature and signature-or-system.

Normal permissions give access to isolated application-level functionalities. These functionalities have little impact on system or user security and are therefore granted without an explicit user's approval. However, the user can review which permissions are requested prior to application installation. An example of a normal level permission is access to the phone's vibration hardware. Since it is an isolated functionality, i.e. user's privacy or other applications cannot be compromised, it is not considered a major security risk.

On the other hand, dangerous permissions provide access to private data and critical systems. For example, by obtaining a dangerous permission, an application can use telephony services, network access, location information or gain other private user data. Since dangerous permissions present a high security risk, the user is promoted to confirm them before the installation.

Signature permission can be granted to the application signed with the same certificate as application declaring the permission. The signature permission is in effect a refinement of the *shared user ID* approach and provides more control in sharing application data and components. On the other hand, signature-or-system permission extends the signature permission by granting permission to the applications installed in the Android system image. However, caution is required since both the signature and signature-or-system permissions will grant access rights without asking for the user's explicit approval.

IV. IOS SECURITY MODEL

Unlike the Android security architecture, iOS security model provides different philosophy for achieving mobile device security and user’s protection. The iOS application platform empowers developers to create new applications and to contribute to the application store. However, each application submitted by a third party developer is sent to the revision process. During the revision process the application code is analyzed by professional developers who make sure that the application is safe before it is released the application store. However, such an application, when installed, gets all the permissions on a mobile device. Application might access local camera, 3G/4G, Wi-Fi or GPS module without user's knowledge. While Android lets each user handle its own security on their own responsibility, the iOS platform makes developers to write safe code using iOS secure APIs and prevents malicious applications from getting into the app store.

The iOS security APIs are located in the Core Services layer of the operating system and are based on services in the Core OS (kernel) layer of the operating system [17]. Application that needs to execute a network task, may use secure networking functions through the CFNetwork API, which is also located in the Core Services layer. The iOS security implementation includes a daemon called the **Security Server** that implements several security protocols, such as access to keychain items and root certificate trust management. The Security Server has no public API. Instead, applications use the Keychain Services API and the Certificate, Key, and Trust services API, which in turn communicate with the Security Server.

Keychain Services API is used to store passwords, keys, certificates, and other secret data. Its implementation therefore requires both cryptographic functions (to encrypt and decrypt secrets) and data storage functions (to store the secrets and related data in files). To achieve these aims, Keychain Services uses the Common Crypto dynamic library.

CFNetwork is a high-level API that can be used by applications to create and maintain secure data streams and to add authentication information to a message. CFNetwork calls underlying security services to set up a secure connection.

The Certificate, Key, and Trust Services API include functions to create, manage, and read certificates, add certificates to a keychain, create encryption keys, encrypt and decrypt data, sign data and verify signatures and manage trust policies. To carry out all these services, the API calls the Common Crypto dynamic library and other Core OS–level services.

Randomization Services provides cryptographically secure pseudorandom numbers. Pseudorandom numbers are generated by a computer algorithm (and are therefore not truly random), but the algorithm is not discernible from the sequence. To generate these numbers, Randomization Services calls a random-number generator in the Core OS layer.

In case that the developers use the presented API properly and do not integrate malicious activities into the application, the application will be accepted into the App store.

V. EXAMPLE OF A MALICIOUS APPLICATION

In order to demonstrate how malicious content could be spread and used to extract sensitive information, we present a simple malicious application for the Android platform. We focus on the permission based security model implemented by the Android since one of the key security factors is the user himself. Since the Android is an open platform, which enforces security by sandboxing applications, it provides the users with the opportunity to install applications from various untrusted sources. Therefore, fooling a user into installing malicious content is an important attack strategy to consider.

As stated in the threat model presented in Section 2, multiple attack vectors for mobile devices exist. In context of modern smart-phone devices we focus on the Internet connection as the delivery path for malware. An example scenario for delivery of malicious content to Android devices via Internet is presented in Figure 2.

The scenario consists out of four entities: the attacker, social network sites, application hosting sites and the user community. First, the attacker deploys the malicious application at a hosting site (1). Since the Android applications do not undergo a code review, the attacker can place them on the Android Market. The attacker then places a link pointing to his application on popular social networks, like Tweeter or Facebook (2). In case that the attacker decided to host the malicious application at a location other than the Market, the true address can be masked by using shortened URLs. When the user clicks on the link (3) he gets redirected to the site from which the application can be downloaded (4). Finally, the user downloads the application and accepts the requested permissions (5).

In Figure 3 a fictive example malicious application *World Weather* is presented. The *World Weather* is a Trojan horse designed to provide weather forecasts depending on the user’s location, while in the background periodically sending location information to a remote server. Upon installation the application requests the permissions to access the Internet, location information and phone state and identity. Since the application did not go through a code review and is potentially harmful, the user has to decide upon his best judgment on whether application’s permission requests are well funded. In this case, it is logical to expect the application to request network access since it is necessary to fetch the forecast.

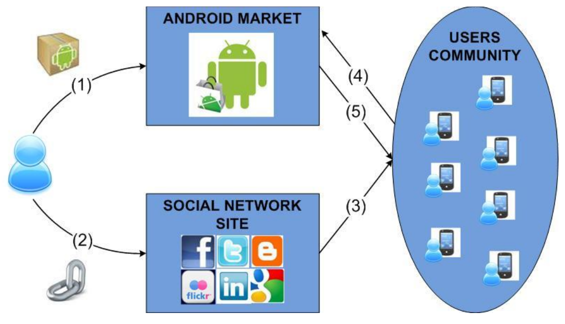

Figure 2. Malicious content delivery scenario for Android

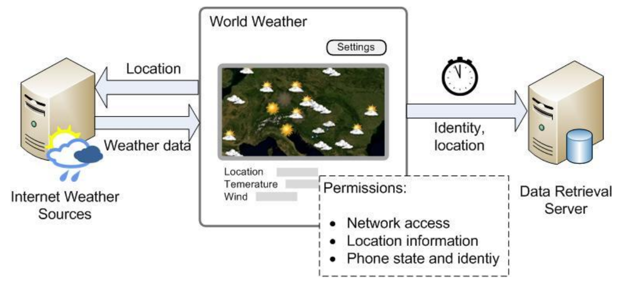

Figure 3. Example malicious alpplication for Android

In addition, the location information request is also expected since the location is used to retrieve the forecast for the user's current location. The phone state and identity permission is suspicious but is used by a lot of applications to check on phone status, like weather a voice call is in progress. For example, a music player app would use the phone status to mute the sound during the voice call. On the other hand the identity information is sometimes used to register the users that purchased an application in order to reduce the piracy. Unfortunately, this is an example of poor permission grouping since the applications that need to check on status often do not need to access the sensitive identity information, like IMEI or IMSI. Furthermore, the Android 1.6 applications are automatically assigned the phone state and identity permission. Since this permission is widely used, the user will be less suspicious if an application requests it. Acquiring the phone status and identity permission allows the *World Weather* application to pair the identity information with location information, i.e. link the user and location. By acquiring this information, the attacker can easily monitor the mobile device user's movement.

To conclude, the presented Trojan commits malicious actions by using permissions that are reasonable for its supposed functionality, i.e. weather forecast retrieval. Therefore, the user cannot detect the malware by observing the requested permissions but rather by monitoring the application's activity or by consulting other sources, like security reports and alerts.

VI. CONCLUSION

Recent advancements in mobile technology have brought the mobile devices into focus of malicious attacks. The trends show a severe increase in mobile malware as many threats, designed for PC operating systems, migrate to mobile platforms.

In this paper we presented an attacker-centric threat model for mobile platforms. We analyzed attacker's goals, attack vectors and attack strategies. Furthermore, we presented the security models implemented by two widely spread mobile platforms: the Google Android and Apple iOS. The two platforms have distinctly different approaches in dealing with security issues.

The Android security model relies on user's judgment to install applications from reliable sources or to evaluate whether the application requests reasonable permissions for its intended operation. By presenting an example malicious application, we demonstrated how careful choice of permissions could mask malicious activities. Therefore, we argue that the Android's permission based security model should be improved with the goal to separate some critical permission like phone status and identity information. Furthermore, since permissions often do not indicate application's malicious intentions, an official certification authority would contribute to overall security.

On the other hand, lack of isolation in iOS platform could severely compromise the mobile device since the malicious application would easily gain access to critical systems and private data. The risks are substantially higher in case of jailbroken devices. Such devices could easily be used to install and spread highly malicious content like rootkits and worms.

REFERENCES

[1] Cisco Systems inc., "The Tipping Point: Cybercriminals Targeting Mobile Platforms", Cisco Annual Security Report 2010, http://www.cisco.com/en/US/prod/collateral/vpndevc/security_annual_report_2010.pdf.

[2] M. Ahmad, et al., "Emerging Cyber Threats Report for 2011", Georgia Tech Information Security Center, October 2010, http://www.gtisc.gatech.edu/pdf/cyberThreatReport2011.pdf.

[3] International Telecommunication Union, "The World in 2010: ICT Facts and Figures", 2010, http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf.

[4] M.Jakobsson and S. Wetzel, "Security Weakness in Bluetooth", Lecture Notes in Computer Science, Springer, 2001, pp. 176-191.

[5] N. Leavitt, "Mobile phones: the next frontier for hackers?", IEEE Computer, vol. 2 no. 4, April 2005, pp. 20-23.

[6] R. Entner, "Smartphones to Overtake Feature Phones in U.S. by 2011", Nielsen Wire, March 2010, http://blog.nielsen.com/nielsenwire/consumer/smartphones-to-overtake-feature-phones-in-u-s-by-2011/.

[7] J. Chen, "An Introduction to Android", Google I/O 2008, May 2008, http://androidgroup.googlecode.com/files/Introduction%20to%20Android.pdf.

[8] T. Cannon, "Android Data Stealing Vulnerability", 2010, http://thomascannon.net/blog/2010/11/android-data-stealing-vulnerability/.

[9] P. Traynor, et al., „On cellular botnets: measuring the impact of malicious devices on a cellular network core", CCS '09 Proceedings of the 16th ACM conference on Computer and communications security, November 2009, pp. 223-234.

[10] R. Racic, "Exploiting mms vulnerabilities to stealthily exhaust mobile phone's battery", In SecureComm 06, 2006, pp. 1-10.

[11] A. R. Flo and Audun Josang, "Consequences of Botnets Spreading to Mobile Devices", Short-Paper Proceedings of the 14th Nordic Conference on Secure IT Systems (NordSec 2009), October 2009, pp. 37-43.

[12] K. Dunham, "Mobile Malware Attacks and Defense", Syngress Publishing, 2008.

[13] F-Secure, Worm:iPhoneOS/Ikee.B, http://www.f-secure.com/v-descs/worm_iphoneos_ikee_b.shtml.

[14] J. Bickford, et al., "Rootkits on Smart Phones: Attacks, Implications and Opportunities", HotMobile '10 Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, February 2010, pp. 49-54.

[15] Google Android, "The Developer's Guide", http://developer.android.com/guide/index.html.

[16] A. Shabtai, et al., Google Android: "A Comprehensive Security Assessment", IEEE Security & Privacy, vol. 8, no. 2, March-April 2010, pp. 35-44.

[17] Apple inc., iOS Reference Library, Security Overview, http://developer.apple.com/library/ios/#documentation/Security/Conceptual/Security_Overview/Introduction/Introduction.html#//apple_ref/doc/uid/TP30000976-CH201-TPXREF101