INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2015) Vol. 4 (2) 123−134

SECURITY RISKS AND THEIR PREVENTION CAPABILITIES IN MOBILE APPLICATION DEVELOPMENT

ANETA MICHALSKA, ANETA PONISZEWSKA-MARANDA

Institute of Information Technology, Lodz University of Technology, Poland

Mobile applications fulfill the more and more significant role in everyday life of the rapidly growing number of Smartphone users. IT companies try to establish new standards of data management as well as create mobile applications extending the functionalities of existing systems to enable the users to benefit from the newest technological advances. The paper presents a review of the known mobile application vulnerabilities for the two most popular mobile platforms, Apple's iOS and Google's Android, and proposes the secure development model to overcome the existing threats faced by mobile application developers.

Keywords: Mobile Application Development, Security of Mobile Applications, IOS Platform, Android Platform

1. Introduction

Together with the growth of the popularity and number of the smartphone users more and more real-life applications for mobile devices were invented and brought to life. Seeing this trend many companies incorporated mobile systems by establishing new standards of data management as well as by creating mobile applications extending the functionalities of their existing systems to enable the users to benefit from the newest technological advances. Nevertheless, after initial phase of going into raptures over the simplicity and multiple possibilities which mobile applications give to their users, the more serious issue of data and information security was raised. It is necessary to talk about the possibly mechanisms to secure

the mobile application, to talk about the development strategies which should be used to prevent capturing sensitive confidential data by third-party applications installed on our smartphones.

However, first of all the common and less common mobile application vulnerabilities should be established to find the incipient development solutions which aim to overcome the existing threats faced by mobile application developers.

The variety of applications embraces not only their functional differences, but also the different technologies they were created in. As the security of a single mobile application is greatly dependent on its target platform the review, presented in the first parts of this paper, will be divided into parts where threats specific for each considered platform will be discussed separately [1, 3, 15, 18].

The choice of the platforms seems to be easy looking at the current trends and usage statistics. Currently the market is led by two most popular mobile operating systems – Apple's iOS and Google's Android. Mobile applications written for these two platforms constitute the majority of the overall number of created applications with the stable position of iOS [2, 4] on the market recently interrupted by the increasing popularity of Android [5, 6].

iOS owes its popularity both to the luxurious brand status as well as to its reputation of being more secure and less prone to external attacks than any other system. However, the intuitiveness and ease of use of Android applications make it comparably attractive despite its known drawbacks in the area of security. Any developer of mobile application for these platforms should be aware of the threats and vulnerabilities that each of them carries and should adjust the development strategies in such a way so that the optimal level of safety is assured especially when interaction with confidential or sensitive data is required [13, 18].

The presented paper is composed as follows: section 2 presents the outline of iOS architecture from the security point of view and the possible security risks in development of iOS applications. Section 3 deals with the security issues of Android operating system architecture and shows the major vulnerabilities of its applications. Finally, section 4 describes the proposition of secure development model for mobile applications.

2. Security architecture and security risks in iOS application development

According to [2] security lies at the core of the architecture of iOS operating system. In fact a variety of mechanisms is implemented within the iOS framework which aims to protect the device and data without user knowledge or developer interference. Apple provides special precautions regarding system security itself as well as security of installed applications, the file system, network services and device control.

The general architecture of iOS system is based on layers (Fig. 1). This means that applications installed and running on the device do not have to communicate with the hardware directly, but instead they use a set of public APIs through which system requests are handled. The developer can freely use these APIs to achieve desired actions, but simultaneously he is limited to a number of operations that Apple made accessible through these APIs. In order to assure that developers do not try to perform forbidden operations the code-signing process takes place [2, 4].

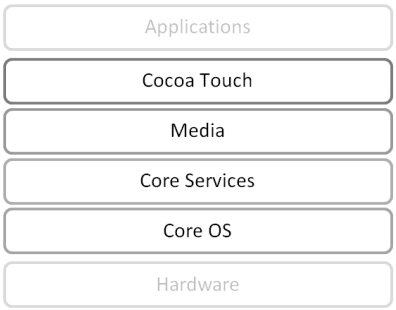

Figure 1. Overview of layered iOS architecture

The lower-level layers Core Services and Core OS contain fundamental technologies available on the device. The high-level layers provide the developers with more specific and sophisticated interfaces in order to accelerate the development process and make actions more understandable and accessible through abstraction and encapsulation. It is recommended that at all times applications should make use of these high-level interfaces instead of the low-level ones as there is better chance that they will conform to Apple implementation standards during code verification [16].

The core security mechanisms implemented as a part of iOS security architecture are as follows:

- layered architecture specifying public APIs for general use for application developers,
- *vetting* process,
- *System Software Authorization* process,
- encryption mechanisms,
- *Data Protection* feature,
- *Keychain* and code-signing.

In order to assess and find flaws in the security of any mobile application it is worth determining what are the possible points of interest for the potential attacker. The most obvious reasons for breaking into the application are stealing the confidential data like passwords, account or credit card numbers and other personal data. Sometimes also capturing the multimedia data like videos or photos, data contained

in the address book, mail or location info may be the target of the attack. Among other reasons one may also distinguish the will to omit some licensing issues or simply doing this "for fun".
The vulnerable points, significant risks of iOS operating system which may become potential security leaks can be named:

- stealing sensitive data stored in *key-chains*,
- installation of third-party applications invoking internal system calls,
- running unsigned code on the device by *jailbreaking* the device,
- *method swizzling* – changing implementation of methods during runtime,
- vast exchange of private personal information for the statistical and advertising purposes.

Basing on such potential aims that the attackers want to gain an access to one can deduce that the most vulnerable parts of the application will embrace:

- data storage points,
- permission management policy
- application file system and
- any kind of configuration files that may be stored within the device.

According to the Apple's security report the design ore of the iOS architecture is its security [2, 4]. And it is beyond doubt that a lot care has been taken to assure proper level of safety by incorporating such mechanisms as data encryption, code-signing and sandboxing (Fig. 2).

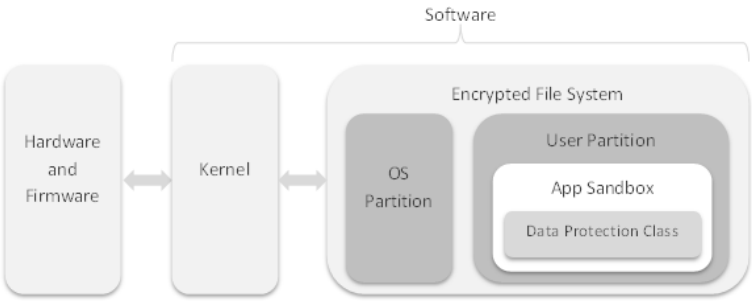

Figure 2. iOS security architecture [2]

Encryption is a standard mechanism and should prevent any unauthorized party from decoding information even if they are captured. *Code-signing* is connected with a strict procedure reserved for any application to be published in the AppStore which assures that only applications conforming to the Apple's standards, using allowed API's methods and submitted by registered distributors will let to be published. *Sandboxing* is a well-known mechanism for running programs separately from other system resources so that there is full control over the permissions and

allowed access that the application requires. Sandboxing is frequently used with untrusted programs which have to be run on the device but here it is used to prevent downloaded application to use the resources of other application or the system or accessing the kernel resources that they are not allowed to.

3. Security architecture and major vulnerabilities of Android applications

Android applications are reported to be the aim of malicious attacks significantly more often then iOS ones. Android developers recognized security as an essential feature of their system and consequently they decided to design the architecture in such a way, so that it provided basic but reliable security mechanisms for applications [5, 8, 10]. The main focus of Android security architecture is to assure proper protection of data and system resources as well as to achieve the effect of isolation of applications, so that they did not interfere with other resources if it is not necessary (Fig. 3).
The basic security techniques built into the internal architecture of Android operating system are the following:

- set of basic security functionalities assured by the used Linux kernel,
- separation of all processes and system resources,
- identification of applications by UID identifiers,
- complex encryption mechanisms,
- permission-based security policy,
- *inter-process communication* (IPC) and *code-signing*.

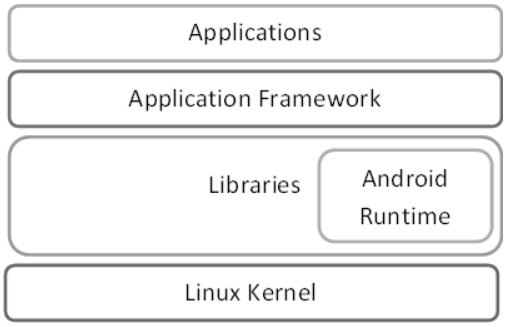

Figure 3. Layered Android architecture

Similarly as in case of iOS architecture the Android operating system can also be viewed as a layered structure. At the core of this structure lies the Linux kernel which assures the stable and reliable environment as it has been thoroughly studied and improved basing on experiences from the original applications on desktop computers.

The Linux kernel provides basic security functionalities such as:

- permission-based access model,
- user-based permissions,
- isolation of processes,
- secure inter-process communication (IPC),
- possibility to remove suspicious parts of the kernel.

Android provides a variety of mechanisms to protect sensitive content. The major focus is put on the separation of resources belonging to different applications which reflects in well-developed sandboxing techniques and widely used permission-based model of accessing resources and performing operations.

Compared with iOS applications, Android ones more frequently become the target of attacks of malicious software [11, 14, 17]. Even though they have been reported as more vulnerable are terms of security. This may derive from the fact that unlike Apple, Google is not performing a strict *vetting* process i.e. it does not check the compliance of the applications published on Google Play with the company's standards. Therefore more malicious software may be slipped through on to the market.

From among major risks connected with development of Android applications the following threats seem to be of utmost importance:

- running third-party applications with *root* privileges,
- reverse-engineering Android applications resulting in access to the application resources like *AndroidManifest.xml* file,
- changing byte code of the application,
- firing *activities* without user interaction,
- vast exchange of data for commercial purposes.

Android applications are commonly written in Java which makes them more easily reversible than iOS ones. As it was mentioned reversing any application allows gaining the information on its structure, data flow and controlling flow. Thus, the security risks for Android applications are in this case corresponding to the ones encountered when dealing with iOS ones. Although the changes that can be done to the application are a lot further-gone as after reversing an Android application it is possible to change the obtained byte-code and repackage the application. The target of such repackaging may be for instance an XML file containing permission configuration – *AndroidManifest.xml*. Among others the information which permissions does the application have is stored: Internet access permission, sharing location, accessing contact list, etc. By altering this file, doing which in fact does not require the changing of byte-code, one may easily increase the range of permissions available for an application. Reversing Android applications is the first step in finding all the information the potential attacker would have in his list. As prov-

en by [17] anything starting from configuration files, database files, certificates, key-stores with the use of proper and available tools can be recovered from the byte-code of the application and altered.

4. Secure Development Model

As was presented in the previous sections the field of security of mobile applications requires the elevated attention because of the privacy issues of millions of users of smartphones and the lack of adequate solutions to assure the security of data. Having this in mind it is worth to think about more ways of how to reduce the risks connected with mobile security in the context of development process itself. It is possible to provide a model that would assure less probability of capturing sensitive data by malicious software or hackers [9, 12].

The idea of *Secure Development Strategy (SDS)* was introduced for building mobile applications so that they would be less vulnerable to external attacks and leaks of sensitive data (Fig. 4).

The existing approaches to mobile application security focus mainly on the transmission of sensitive data to external services that is not always necessary and desired. Safe data transfer between mobile and external devices is undoubtedly a crucial link in the process of securing the applications, however not the only one. The idea of *Secure Development Strategy* assumes that application should conform to pre-defined security standards embracing storage, access and transfer of sensitive data. Conformation to the standards should be achieved by implementing threefold security pattern for each of the mentioned areas. The model specifies the assumptions on how to achieve a proper level of security in each field and provides necessary details on the implementation of mechanisms which will allow achieving desired safety effects.

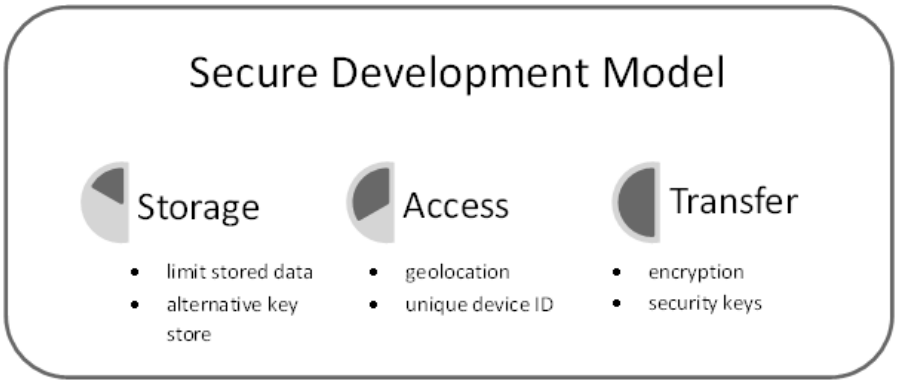

Figure 4. Pillars of Secure Development Strategy for mobile applications

The foundation of *data storage* pattern should be based on limitation of data stored on the device and application of an alternative storage space. The extraction of data from built-in key stores does not constitute a major difficulty. The best solution to this problem is to limit the amount of stored data especially the critical ones. However, as it is not possible to avoid it entirely, so additional precautions can be considered. The main focus for *accessing data* will be put on identification and verification of the device and user who wants to gain an access to the external resources. The last component which is *data transfer* is dependent on the operation of the system as a whole, therefore the most concerning aspects that will be taken into account are the format, encoding and permission checks of data which are going to be transferred. All of the mentioned mechanisms should be supported by appropriate encryption techniques.

4.1. Data storage security model

The first pillar of SDS – *storage* – concerns solely the application-side of the system. The major assumptions of data storage pattern embrace sensitive data encryption, limitation and restricted access (Fig. 5).

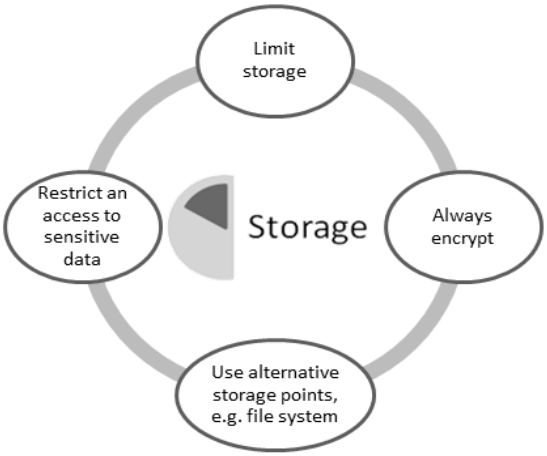

Figure 5. Assumptions of data storage model

While designing mobile applications the developer has two possibilities on where to store application data. He can choose external server where data will be stored in databases and special firewall mechanisms will block access to it. On the other hand if the amount of data is not too large he can choose to store some information on the device itself in the local database or file system.

The first option seems to be a better solution as it eliminates the risk of losing data when the device is damaged. Nonetheless, it requires a large amount of data

traffic between the application and the server. In that light the second option comes in handy – it reduces the amount of data transfer. However it seems to be less practical, as the data to be valid need to be updated. Moreover, the storage space of the device is also limited. Thus, the combination of both solutions comes from the need of keeping the data up-to-date and accessible by many devices at any time simultaneously giving the possibility to store a little number of crucial information on the device.

The storage mechanisms depend on the place of where the data is saved on the device. Two places for data storage which are also a potential risk points can be discussed for SDS: *key store* or *keychain* as it is called in iOS and *file system*.
The three rules regarding data storage can be formulated as follows:

- sensitive data should never be stored as plain text, but they should always be encrypted and stored as such in key-chains and any other storage places,
- sensitive data could be stored within the application database files and encrypted using encryption keys stored in the external server databases to limit the risk of reading the data,
- access to the internal database objects should be restricted only to the privileged functions (function calls).

4.2. Data access security model

The second pillar of SDS strategy concerns the *access* to data. This comes from the fact that mobile applications need to communicate with external services and other applications.

The major assumptions of this area of security embrace the three mechanisms which aim to enable identification of the user requesting access to application resources (Fig.6).

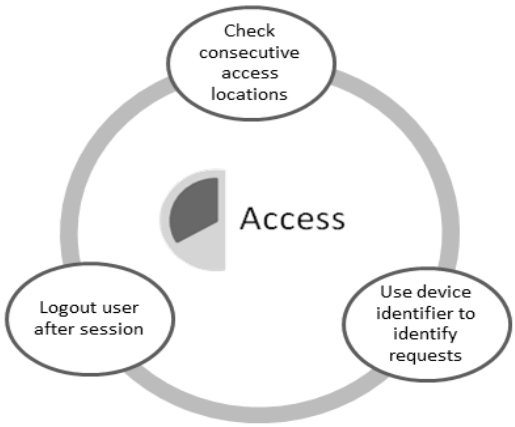

Figure 6. Assumptions of data access model

The fundamental SDS rules for data access are:

- mobile application should inform about the current location of the device every time it requires an access to sensitive data,
- mobile application should always present itself with a digital signature composed of unique device identifier,
- server should always check whether the device session is open before it realizes any requests.

4.3. Data transfer security model

Data *transfer* pattern refers to all mechanisms which involve exchange of data between the mobile application and external services. These mechanisms should incorporate in their action flow additional security procedures – *data encryption*, the use of *security keys* and check of requests integrity (Fig. 7).

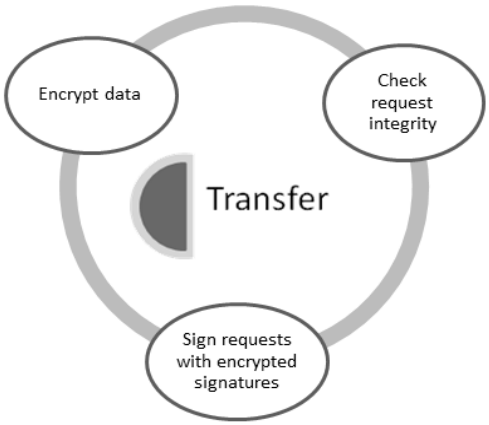

Figure 7. Assumptions of data transfer model

Although this is the last one of the described pillars of SDS, it is absolutely not the least important one. On the contrary data transfer seems to be the weakest link in the entire process of mobile application development. This comes from the fact that requests are travelling over the Internet in an unprotected space and they are prone to a special kind of attacks called *the man in the middle*.

This attack means that between the mobile and server application a third-party may be listening and waiting for exchange of information. There exist three types of attacks for the "man in the middle" scenario:

- attack on the privacy of data – stealing confidential information,
- attack on the integrity of data – changing the content of the message,
- impersonation – impersonating other device/user.

Firstly, the data sent to the server should always be *encrypted*. Special care should be of course taken of data which are considered sensitive like passwords, credit card numbers and others alike. *Encryption* is crucial in case of the first attack scenario regarding privacy – even in case of capturing the data, the content of the message will be difficult to read.

As far integrity of data is concerned it is a good and common practice to use the *digital signatures*. They enable to check whether the message received is exactly the same as message sent and if it was not modified on the way to the receiver. The digital signature mechanism relies on encryption which should assure the authentication.

The last suggested mechanism represents *security keys* which seem to be valuable in case of using checksum algorithms for digital signatures. Security keys introduced by SDS are for example 128-bit strings which should be sent to the server before the request for sensitive data. The security key can be unified for the entire application and independent from the device. When transferred it should also be encoded with the cryptographic algorithm.

5. Conclusion

The analysis of the current state of the art in the field of mobile application security enables to state that threats encountered in mobile application development do not vary significantly between the chosen platforms. Both iOS and Android operating systems are prone to similar threats and the existing solutions aiming to prevent them oscillate around the same issue of privacy leaks in applications.

While designing and building the mobile applications there will always be a tradeoff between functionality, optimization and security. In the long run however none of these areas should be omitted and treated with less importance than the others as only balanced combination of those three will enable to create reliable and useful solutions.

Currently there exists no standardized solution or methodology for developing the applications which would threat-resistant. Together with the invention of newer and newer strategies to prevent the security breaches the more sophisticated threats emerge and new vulnerabilities are detected by the platform developers.

The presented in the paper Secure Development Strategy for mobile applications introduces three pillars which should be taken into consideration while designing and implementing the mobile applications. All these pillars: data storage, data access and data transfer should be treated as equally significant in developing applications. The SDS provides details on its assumptions and mechanisms which should be implemented within the application framework in order to provide the security.

REFERENCES

[1] Porter Felt A., Finifter M., Chin E., Hanna S., Wagner D. (2011) *A survey of mobile malware in the wild*, 1st ACM workshop on Security and privacy in smartphones and mobile devices, 3–14.

[2] Apple (2014) *iOS Security*.

[3] Souppaya M. P., Scarfone K. A. (2013) *Guidelines for Managing the Security of Mobile Devices in the Enterprise*, NIST.

[4] Agarwal Y., Hall M. (2013) *ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing*, 1th Annual International Conference on Mobile systems, applications, and services, 97–110.

[5] Zhou Y., Jiang X. (2012) *Dissecting Android Malware: Characterization and Evolution*, 33rd IEEE Symposium on Security and Privacy.

[6] Vidas T., Votipka D., Christin N. (2011) *All Your Droid Are Belong to Us: A Survey of Current Android Attacks*, 5th USENIX Workshop on Offensive Technologies.

[7] Seriot N. (2010) *iPhone Privacy*, In Black Hat DC, USA.

[8] Enck W., Ongtang M., McDaniel P. (2009) *Understanding Android Security*, Security & Privacy, IEEE, Vol. 7, Issue 1, 50–57.

[9] Porter Felt A., Wang H. J., Moshchuk A., Hanna S., ChinE (2011) *Permission Re-Delegation: Attacks and Defenses*, 20th USENIX Security Symposium.

[10] Ongtang M., McLaughlin S., Enck W., McDaniel P. (2009) *Semantically Rich Application-Centric Security in Android*, Computer Security Applications Conference.

[11] Enck W., Octeau D., McDaniel P., Chaudhuri S. (2011) *A Study of Android Application Security*, 20th USENIX Security Symposium.

[12] Park M. (2012) *Mobile Application Security: Who, How and Why*, Trustwave SpiderLabs.

[13] Fitzgerald W. M., Neville U., Foley S. N. (2013) *MASON: Mobile autonomic security for network access controls*, Journal of Information Security and Applications, Vol. 18, Issue 1, 14–29.

[14] Zdziarski J. (2012) *Hacking and Securing iOS Applications. Stealing Data, Hijacking Software, and How to Prevent It*, O'Reilly Media.

[15] Alhamed M., Amir K., Omari M., Le W (2013) *Comparing Privacy Control Methods for Smartphone Platforms*, Engineering of Mobile-Enabled Systems, MOBS.

[16] Gianchandani P. (2013) *IOS Application Security Part 12 – Dumping Keychain Data. Keychanin basics*, Infosec Institute.

[17] Benedict C. (2012) *Under the Hood: Reversing Android Applications*, Infosec.

[18] Khan S., Nauman M., Othman A. T., Musa S. (2012) *How secure is your smartphone: an analysis of smartphone security mechanisms*, International conference on cyber security, cyber warfare and digital forensic, 76–81.