2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) | 978-1-6654-1658-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/TRUSTCOM53373.2021.00178

Secure Development Strategy Model Framework for Security of Mobile Applications

1st Aneta Poniszewska-Marańda
Institute of Information Technology
Lodz University of Technology
Lodz, Poland
aneta.poniszewska-maranda@p.lodz.pl,
ORCID: 0000-0001-7596-0813

2nd Łukasz Chomatek
Institute of Information Technology
Lodz University of Technology
Lodz, Poland
lukasz.chomatek@p.lodz.pl,
ORCID: 0000-0002-7651-6877

3rd Joanna Ochelska-Mierzejewska
Institute of Information Technology
Lodz University of Technology
Lodz, Poland
joanna.ochelska-mierzejewska@p.lodz.pl,
ORCID: 0000-0002-9295-3962

***Abstract*—The growing popularity of mobile technologies and mobile systems, especially Android system, brings many possibilities. Using the mobile application, it is possible to make bank transfers, communicate with patients for applications used in medicine and many others. Popularity, unfortunately, also brings many risks. With the development of the Internet and information technology, the range of opportunities for cybercriminals has developed. With the increasing popularity of systems and mobile devices, we can also observe an increase in attacks and the number of malicious software targeted for these technologies. The development of the mobile application fulfills more and more important role in the everyday lives of the visibly growing number of smartphone and tablet users, especially from the point of view of security aspects. The paper presents an outline of the proposed secure development model to overcome the existing threats faced by the mobile application developers and its implementation in the form of the mobile security framework.**

***Index Terms*—mobile application development, security of mobile applications, secure development strategy model, access control model, security framework**

I. Introduction

Data security is to ensure data protection against damage or actions made by unauthorized persons. Data in this context may be a database, the data stored on our personal computers or any other form of information that is possible to understand the man. Noteworthy is that it can also be data that does not have to leave the source or is not understood by someone else. We could say that data security or private data is quite a fluid concept because each person differently approaches this issue – for one person, it is essential that the photos were not publicly available; another person will be happy to share with people his photos.

Everyone in some way trying to protect their own data, and in the era of universal access to the Internet and various social networking sites, it is not so easy to take care of it. Therefore it is essential to create a safe working environment for each device on which we can find our data were best protected against any leakage. Potentially mobile devices are the place where user data can most easily be stolen. Many people store a large amount of their private data on them, have the accounts linked with social networking sites, email accounts, or make payments using their smartphones. Moreover, smartphones are small devices that are easy to lose or lose from theft.

Unfortunately, it often happens so that people affect a speedy completion of some activities (including on the phone), therefore excluding any collateral, which could slow down the operation of their phone, including security, such as a screen lock. Another issue is that many people use illegal software on the phone and never be sure whether installing a program from an untrusted source will not install some malicious software.

Among mobile devices, there are many different systems, and each offers something different. The most popular systems and at the same time the direct competition with each other are Android and iOS. The first of them is available on more than 60% smartphones available on the market. The second one is located in less than 30% devices. Android is an open system that gives the users the ability to devise configurations to their liking. Because of the publicly available source code, a community of people involved in creating their own version based on these sources was created, introducing their amendments or accessories. iOS, in turn, is a closed system with little opportunity configuration. The main objective of the Apple company – iOS developers – is to create a product that is easy to use, intuitive and much more secured [2], [3], [12], [14], [16]

The operation of the above systems differs significantly, and hence the process of creating applications on each requires different tools and skills. It is preventing the threats associated with the creation process of mobile applications, their deployment and use. Still, the weaknesses that characterize these platforms are not a task that can be realized and performed using one targeted solution. Of course, it is also necessary to think about the suitable policy for securing mobile applications, covering different aspects of security breaches.

The development process of mobile applications should be aware of the threats and vulnerabilities of mobile platforms. It should adjust the development strategies to obtain the optimal level of safety, especially during the interaction with confidential or sensitive data is required [11], [12].

Privacy and security of sensitive data storage and access seem to be the most significant area requiring interest. It is possible to grant or deny access to specific sensitive informa-

2324-9013/21/$31.00 ©2021 IEEE
DOI 10.1109/TrustCom53373.2021.00178

tion by introducing an access detection system based on the determined rules, models or recommendation engine. There are the works concerning the security solution for mobile application and their development – both for iOS [4], [6], [7], [13] and Android [5], [8], [9], [15] platforms.

The presented paper is composed as follows: section 2 presents the secure development model with its strategy and its three pillars: data storage, data access and data transfer, while section 3 describes the technical implementation details of the security mobile framework incorporating safety procedures at different levels of mobile application development and use according to the rules of secure development model described in the previous section.

II. Secure Development Model

The field of mobile application security requires adequate solutions to assure the security of data. It is important to specify how to reduce the risks connected with mobile security in the context of the development process itself. One of the most crucial aspects is creating a model that assures less probability of capturing the sensitive data by the malicious software or hackers [10], [14], [16].

The model called *Secure Development Strategy (SDS)* was developed for building the mobile applications to be less vulnerable to external attacks and leaks of sensitive data (Fig. 1) [17]. SDS approach equally focuses on all aspects of sensitive data management, not only on the transmission of sensitive data to external services. Safe data transfer between mobile and external devices is for sure a crucial point in the process of securing the applications, however not the only one.

The idea of *Secure Development Strategy* is based on the concept that mobile application should conform to predefined security standards containing three main areas: *storage* (of sensitive data on the mobile device), *access* and *transfer* of sensitive data. Conformation to the standards is achieved by implementing a threefold security pattern for each of the mentioned areas. The model defines how to achieve a proper level of security in each field and provides necessary details of mechanisms to achieve the desired safety effects [17].

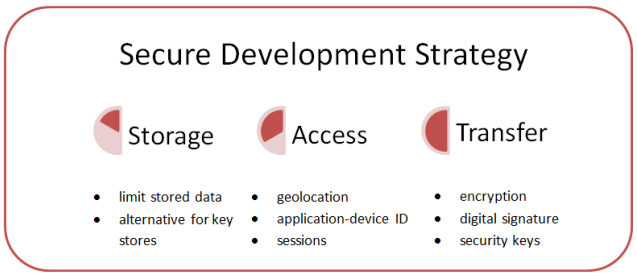

Fig. 1. Pillars of Secure Development Strategy for mobile applications.

The foundation of data storage patterns should be based on the limitation of the data stored on the device and (in case the storage is inevitable) the proper level of protection achieved by encryption. As far as typical storage mechanisms are concerned, they assure the most reliable degree of security. However, in the case of capturing the device, extraction of data from built-in key stores does not constitute a significant difficulty. The most desired practice would be the limitation of the amount of the data stored, especially this critical importance. However, as it is not possible to avoid it entirely, additional precautions should be considered, encryption seeming to be of the greatest significance. The main focus for accessing data is put on the identification and verification of the device and the user who attempts to gain access to the application resources. Also, an additional access privilege verification process can be implemented for specific methods which require such access [17].

The first pillar of the SDS storage concerns the client-side of the system, i.e., the mobile application. The major assumptions of data storage patterns embrace sensitive data encryption, limitation, and restricted access.There are two places where the application data can be stored while designing mobile applications. The external server can be chosen where data will be stored in databases, and special firewall mechanisms will block access to it. However, there is information that the application uses, and it is necessary to be stored on the device. It is because an application works in an offline mode when there is no communication with the server and external database.

The second pillar of the SDS model concerns access to data. It is very important because mobile applications need to communicate with external services and other applications. The major assumptions of this security area contain three mechanisms that enable identification of the user requesting access to application resources.The access to application resources can be controlled at different levels. The first point can be an access of application functions to the resources, which is vital if anyone tries to modify the behavior of an application, e.g., by the swizzling method. The second point is access to data by unprivileged services or remote calls to the server from parties impersonating valid devices. The SDS strategies for such threats include the use of *geolocation*, *device unique identifier* and *sessions* as the means of distinguishing suspicious behaviours [17].

The data transfer pillar refers to the mechanisms for exchanging data between the mobile application and the external services. These mechanisms should incorporate additional security procedures in their action flow, such as data encryption, digital signatures, and security keys.The data transfer is, in fact, the weakest link in the entire process of mobile application development. Because of the requests that are traveling over the Internet in an unprotected space, they are prone to different kinds of attacks, e.g. ”man in the middle” attack.

To fulfill the requirements and conditions of mobile applications, the access control model was created for them. It encompasses the traditional access control ideas and solutions and the definition of rules for mobile applications, containing both static and dynamic access control aspects. It was made especially to ensure the functionality of the second pillar of SDS strategy, i.e. data access security model. The

proposed mobile access control approach was named *mobile Application-based Access Control (mABAC) model*. The core part of mABAC approach essentially represents the basic elements of access control, such as *subjects*, *objects* and *methods*.We distinguished two main types of subjects in mABAC: user (*User* and mobile device (*Device*). These elements are represented by element **Subject** that is the superclass of User and Device. *Subjects* hold and execute indirectly certain rights on the objects. Mobile applications are working on behalf of users who execute them on devices directly or indirectly. Detailed view of presented mobile Application-based Access Control (mABAC) model with the set of all elements and relationships is given in figure 2.

III. iSec framework as the implementation of the SDS model

The SDS model for building secure mobile applications specifies the guidelines that should be considered while creating such applications by the developers. It is, however, common knowledge that the conditions under which most of the developers work – tight schedules, close deadlines, performance efficiency – do not serve well, assuring impeccable security.

Even being aware of the basic assumptions of the SDS, it may not always be possible to assure all of the mentioned precautions as they require additional time to implement. This is the reason why a need for a unified framework incorporating all the enlisted mechanisms comes to mind.

The implementation of ready-to-use classes and methods to ensure mobile applications' security seems to suit the needs for time efficiency and safety. It was developed in a prototype framework named *iSec*. Its main components correspond to three pillars of the security model presented in section 2: storage, access and transfer. All the major components of the framework use *encryption module* which provides basic and complex encryption methods. Another component common for others is *XMLParser* which allows to read and write the security configuration stored in the external XML file. The component model of the created framework is presented in figure 3.

A. Storage Module

The *storage module* is responsible for managing data operations connected with their storage on the device. This includes two main functionalities – managing data stored in the key-chain and managing data stored in the external database files. The first functionality is realized by a wrapper class providing all the basic key-chain operations.

Unlike most of the existing key-chain wrapper classes, the one implemented within the *iSec* framework provides the automatic value encryption, so there is no chance to save unencrypted variables. Analogically while attempting to obtain the value from a key-chain the class performs automatic decryption. What is crucial here is the choice of the encryption algorithm to be used for the key store values. This algorithm is to be selected at the initialization of the key-chain variable. If no algorithm is selected, then the values will be encrypted with a default value.

The second functionality provides similar encryption mechanisms for data stored in the internal device database. The management and creation of the database objects should be performed by the developers according to the well-known techniques using the core data library, but the methods for extracting and saving data are provided within this class. They are generic methods thanks to which it is possible to manage different types of objects.

The storage module cooperates with the encryption component.

B. Access Module

The *access module* provides basic mechanisms which could be implemented to control access to data not only within the application itself but also during the communication and data exchange with the external web service – in this way; it supports and cooperates with the transfer module described in the following subsection.

As far as internal access is concerned, the component in the subject provides end interface to verify methods that require specific internal resources or the data stored on the device. The access privileges for different functions of the application are stored within the XML configuration file, and the developer specifies them during their implementation. Every request for the data stored in the key-chain or the device database files undergoes a verification process against permissions stored in the configuration file. If the verification is successful, the data is obtained from the source and decrypted using methods from the encryption module. Other functionalities of the access module are specific for the external communication with the server.

The complementary mechanism to the request is the custom device identifier (ADID). The generation method for ADID as well as obtaining its value are also provided by this component. ADID is device-application-specific. It uses a part of the device's unique identification number to generate the resultant value combined with a couple of other variables. The encryption of the ADID value is realized with the methods provided by the encryption module, as noted for all previously described mechanisms (Fig. 4).

C. Transfer Module

The *transfer module* is responsible for generating and signing the requests issued to the server. It uses the encryption component to cipher the payload of the sent message if the developer decides to use the methods generating an encrypted content. It is also possible to send an unencrypted payload. The transfer module enables to send data in two formats – XML and JSON.

Next to the request generation, the component uses the access module to add necessary device-specific information mentioned in the previous section. Moreover, it provides a method for generating the message's digital signature and adds it to the request. The transfer module also cooperates with

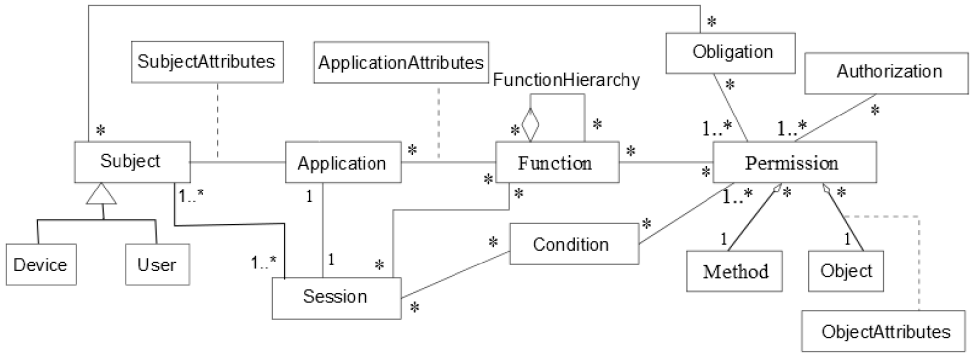

Fig. 2. Meta-model of mABAC model.

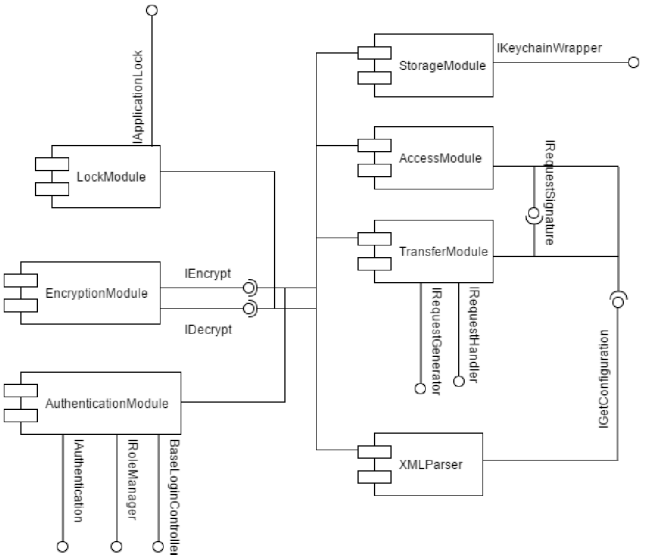

Fig. 3. Components of the *iSec* framework.

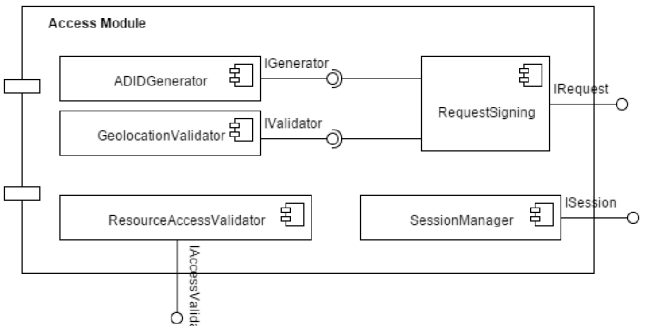

Fig. 4. Subcomponents of the *Access Module*.

the XML parsing component as it requires some information stored there. This information includes the address of the server to which all the requests are issued and the security key specific to the application. The developer specifies this data during application development, and it cannot be changed or easily accessed. Also, one avoids storing the server address directly in the code of the application as plain text.

D. Authentication Module

One of the most common functionalities that most applications incorporate is registering and logging in to the users who use the application.

The visual design of registration and login views can be extremely differentiated among different applications. Nevertheless, the basic flow of operations is practically the same or similar in most cases. That is why the very useful components of the *iSec* framework are the authentication and authorization modules, which provide the universal API for common registration and login procedures. The developers can use the existing controllers for the two views without the necessity of specifying multiple times the same operations for different applications they build. The provided for reuse controller classes are:

- *BaseRegistrationViewController* and
- *BaseLoginViewController*.

The developers are able to use these controllers by directly linking them to the views they created or extend their functionality and adjust them to their needs.

Furthermore, the modules specify the model classes for the user and role model elements. They can also be reused directly or extended and customized. The components constituting the authentication model are given in figure 5.

E. Encryption Module

The *encryption module* is a common component for all other modules of the *iSec* framework as it provides general decryption and encryption methods. A vital feature of this module is the possibility to select from among a variety of available encryption algorithms. The developer can decide which of the encryption methods to use or rely on the default

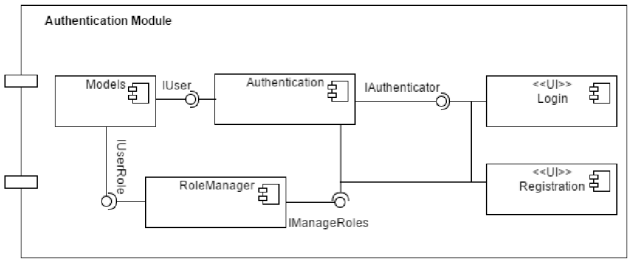

Fig. 5. Subcomponents of the *Authentication Module*.

settings. The available encryption algorithms and hashing functions are: MD5, SHA-1, DES, 3DES and AES.

The most significant advantage for developers is the elimination of the necessity to search for external libraries providing given encryption functionalities – all of the most reliable and frequently used solutions are gathered in one place.

F. XML Parser Module

The *XML parser module* is responsible for managing the data stored in encrypted XML configuration file. This file stores basic configuration variables used mostly by the access and transfer modules. These variables are external server addresses, application key, function privilege settings.

As mentioned, the file itself is encrypted, which indicates the use of the interface methods provided by the encryption module. The file is stored on the device, and its backup copy should also be stored on the server to prevent sabotaging the operation of an application. However, it should be implemented manually by the developer to be performed during the first application launch.

The file contains the information in XML format. The required methods provided by this component enable a proper parsing of the document together with the functions for extracting valid data against which permission checks and data extraction may be performed. The key for deciphering the XML configuration file should be stored on the external server.

G. Request Handling Module

One of the significant elements of the framework is a component handling the requests coming to and from the mobile application called the *RequestHandler*. This component cooperates with the common modules of the framework – *XMLParser*, *Encryption* and *PermissionHandler* using their functionalities to secure the data.

The *RequestHandler* manages the requests both within the mobile application and those enabling the communication with the server. Internal requests are those where certain functions (methods) invoked by the mobile application require access to the sensitive data stored on the device. The *RequestHandler* enables issuing those requests, passing them through a verification process to check the privileges for data access and then allowing to obtain a success or failure response. External message handling enables creating the correct syntax of the server request, issuing it to the server and interpreting the received response. While making the request, the *RequestHandler* itself attaches the default request components such as the checksum or security key.

The communication schema between *RequestHandler* component and other components is presented in figure 6.

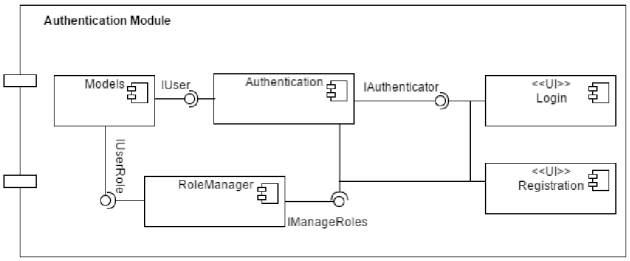

Fig. 6. RequestHandler and common components of the *iSec* framework.

H. Lock Module

The *lock module* is the additional component to be incorporated in the *iSec* framework providing the functionality to secure the mobile application or the entire device in case the device using it is stolen or lost.

In case the application deals with confidential data, or it may be presumed that the users might not want their data to fall into the wrong hands, the application may offer the possibility to lock the device if it receives the appropriate signal. The role of the developers in such a case is that they can place a separate setting in the configuration view of the application, which will allow the users to select the desired behavior if an unauthorized third party uses the telephone.

The *iSec* framework lock module offers two types of behavior in such case: locking access to the application and removing all application data from the internal phone storage. These operations can be performed if any of the following actions occur:

- attempt to input invalid credentials to the application more than three times,
- sending a message to the device with valid lock text.

The unlocking of the device can only be performed with SMS. If the user decides to use the SMS lock option, he would have to specify the phone number from which such SMS could be sent. Locking the phone, especially with the opportunity to remove internal application data, seems to be, however, the last resort action to undertake. That is why it should be used wisely.

Firstly, the application designers should thoroughly think through whether such a possibility is necessary to implement. If so, the users should be clearly warned about the consequences of selecting particular options. Nevertheless, this option allows the users to have more control over their device when it is stolen or lost. They do not have to install any additional applications responsible only for device locking.

I. Server-side (additional) Module

Besides assuring the basic security on the mobile device, it is worth noting that the server-side should also incorporate the mechanisms that would maintain the cooperation with those implemented on a mobile device. Such cooperation refers mainly to the transfer mechanisms and request verification. Thus, a separate library for the facilitation of this process and to be included in the main web service could be developed.

The schema of information flow using the mentioned library and additional system components is presented in figure 7.

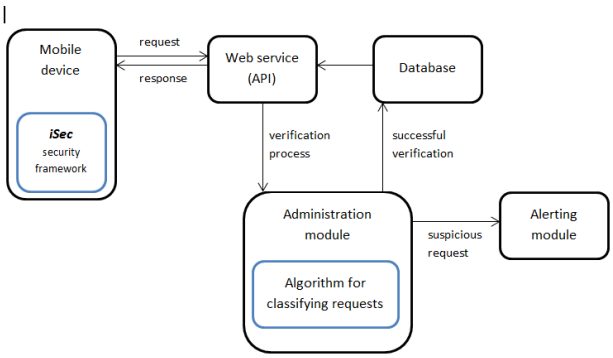

Fig. 7. *iSec* external components architecture.

Before the request reaches the database, it is processed by the web service and verified against its validity and integrity of data carried. After successful verification, the request is fulfilled, and the data from the database can be extracted or saved. Otherwise, appropriate logs are being created and, if configured so, an email can be sent to the system administrator. Next to the verification, a corresponding request for the signing module could secure the outgoing responses.

This module would work similarly to the transfer module in the *iSec* framework for iOS. Additional components which could be included here are administration and alerting modules. Thanks to the administration module, the privileged user would be able to manage the devices of an application and view the request history. Alerting module would allow him to automatically receive the information about suspicious attempts to read the data and view the details of such requests.

IV. Conclusions

Mobile devices are potential places where user data may be stolen much easier. Many people store any private data, connect to social media, email accounts or pay with a mobile device. Smartphones are small devices, easy to get lost or stolen. Unfortunately, people want to quickly perform an operation on a mobile phone, which is the reason for disabling anything that could slow down this process (even security settings). The fact that people are installing illegal software on their devices is another case. That software may be potentially malicious. The Secure Development Strategy created for mobile applications introduces three pillars that should be considered while designing and implementing the mobile applications. All these pillars: data storage, data access and data transfer, should be treated as equally significant throughout the entire development process.

The practical implementation of the components of SDS strategy is realized with the use of created security framework – *iSec*. The first stages of development have shown the weaker points of the assumptions made initially. It also enabled to look deeper into the structures of the applications and find other possible areas where security can be breached, and the breaches could be avoided. The possibilities include, especially the data access control mechanisms. The *iSec* framework constitutes a tool which enables the usage of a variety of security mechanism without the necessity to use additional external sources. It provides the implementation of the functionalities defined by the SDS. Its features aim to simplify the process of implementing security into mobile applications.

References

[1] A. Poniszewska-Maranda, "Modeling and design of role engineering in development of access control for dynamic information systems", *Bulletin of the Polish Academy of Sciences, Technical Science*, 61(3), 569–580 (2013).

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

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

[4] Apple, "iOS Security", *mages.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf* (2014).

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

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

[7] T. Werthmann, R. Hund, L. Davi, A. Sadeghi, and T. Holz, "PSiOS: Bring Your Own Privacy & Security to iOS Devices", (2013).

[8] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting Privacy Leaks in iOS Applications", (2011).

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

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

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

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

[13] J. Zdziarski, *Hacking and Securing iOS Applications*, O'Reilly Media, 2012.

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

[15] A. Zaheer, F. Lishoy, A. Tansir, C. Lobodzinski, D. Audsin, and J. Peng, "Enhancing the security of mobile Applications by using TEE and (U)SIM", *Proc. 10th International Conference on Ubiquitous Intelligence and Computing* (2013).

[16] A. Michalska, and A. Poniszewska-Marańda, "Security risks and their prevention capabilities in mobile application development", *Information Systems in Management, WULS Press* 4(2), 123–134 (2015).

[17] A. Majchrzycka, A. Poniszewska-Marańda, "Secure Development Model for mobile applications", *Bulletin of the Polish Academy of Sciences, Technical Science*, 64(3), 495–503 (2016).