MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
SUMY STATE UNIVERSITY
UKRAINIAN FEDERATION OF INFORMATICS

PROCEEDINGS

OF THE VI INTERNATIONAL SCIENTIFIC CONFERENCE

ADVANCED INFORMATION SYSTEMS AND TECHNOLOGIES

AIST-2018

(Sumy, May 16–18, 2018)

SUMY
SUMY STATE UNIVERSITY
2018

UDC 004(063)
A

A Advanced Information Systems and Technologies: proceedings of the VI international scientific conference, Sumy, May 16–18 2018 / Edited by S. I. Protsenko, V. V. Shendryk – Sumy: Sumy State University, 2018 – 145 p.

This book comprises the proceedings of the VI International Scientific Conference “Advanced Information Systems and Technologies, AIST-2018”. The proceeding papers cover issues related to system analysis and modeling, project management, information system engineering, intelligent data processing, computer networking and telecomunications, modern methods and information technologies of sustainable development. They will be useful for students, graduate students, researchers who interested in computer science.

UDC 004(063)

ISSN 2311-8504

International Scientific Committee:

A.N. Chornous, Sc.D (Ukraine)
A.S. Dovbysh, Sc.D (Ukraine)
O.A. Borisenko, Sc.D (Ukraine)
E.A. Lavrov, Sc.D (Ukraine)
V.O. Lyubchak, PhD (Ukraine)
S.I. Protsenko, Sc.D (Ukraine)
A.M. Kulish, Sc.D (Ukraine)
M.M. Glybovets, Sc.D (Ukraine)
Yu. I. Grytsyuk , Sc.D (Ukraine)
I.V. Grebennik, Sc.D (Ukraine)
O.O. Yemets, Sc.D (Ukraine)
D.D. Peleshko, Sc.D (Ukraine
V.I. Lytvynenko, Sc.D (Ukraine)
N. B. Shakhovska, Sc.D (Ukraine)
E.A. Druzynin, Sc.D (Ukraine)
S.I. Dotsenko, Sc.D (Ukrain)
T.V. Kovalyuk, PhD (Ukraine)
A. Pakštas, PhD (United Kingdom)
O.Romanko, PhD (Canada)
I. Polik, PhD (USA)
V.Kalashnikov, Sc.D (Mexico)
S. Berezyuk, PhD (Canada)
P. Davidsson, PhD (Sweden)
M. Biagi, PhD (Italy)

Organizing Committee:

Protsenko S. I., Sc.D, chairman (Ukraine);
Shendryk V. V., PhD, co-chairman(Ukraine);
Vaschenko S. M., PhD, co-chairman (Ukraine);
Parfenenko Y. V., PhD (Ukraine), Nahornyi V. V. , PhD (Ukraine),
Zakharchenko V. P. (Ukraine), Shendryk S.O. (Ukraine),
Boiko O. V., PhD, executive secretary (Ukraine).

Contacts:

Address: AIST conference, Sumy State University
2 Rimsky-Korsakov Str., Sumy, 40000, Ukraine
website: www.aist.sumdu.edu.ua
e-mail: aist@sumdu.edu.ua.

CONTENTS

Secure Mobile Application Development

Roman Yatsenko, Viktor Obodiak[1], Valerii Yatsenko
Sumy State University, Ukraine, [1]v.obodyak@cs.sumdu.edu.ua

***Abstract* – The article deals with a problem of mobile applications vulnerability. It explores the most popular attack types and ways to avoid them. There were identified the main security rules of code writing for mobile developers. Recommendations how to protect a mobile application from cracking have been given.**

Keywords – mobile application; vulnerabilities; attack types; security rules.

I. Introduction

When developing a mobile application, it should be taken into account that the data that this application operates on may be of some interest to third parties. The degree of value of these data varies widely, however, even the most simple private information, for example, the password for entering the application, requires the elaboration of its protection. This is especially important in the light of the spread of mobile applications to all areas of electronic services, including financial, banking, storage and transfer of personal data.

II. Formulation of the Problem

Mobile technologies are developing very quickly and at the same time the number of vulnerabilities and methods of hacking is growing. Every year, Apple announces a new iPhone, and Google is pleased users with a new version of Android, but with no less frequency, you can see articles about a new virus or a leak of private user information. Therefore, developers should constantly monitor both new versions of products and the appearance of new types of attacks.

In general, there are three main types of attacks on a mobile application:

1) Decompiling the application file (.ipa files for Apple iOS and .apk files for Google Android) and parsing the locally stored data. These files are archives in which executable files, configuration files, application resources, etc. are stored. If you unpack them and analyze the configuration files, you can often find lines of code that developers forgot to remove from the final version of the product. These lines of code are most often used for debugging during the development period of the application, and they can greatly facilitate the attacker's task of obtaining confidential data or implementing other unauthorized actions. At the testing stage, developers often incorrectly assign access rights and forget to edit them at the final release of the software product, which is why the attackers have even more opportunities for unauthorized access. Despite attempts by developers of mobile operating systems, unfortunately, getting an application file is quite easy for an attacker and by the fact that the protection of this, the most important at present, level entirely lies on the shoulders of the mobile developer.

2) Interception of data transmitted over the network (MITM-attacks). Most mobile applications are client-server, therefore, constantly transmit and receive large amounts of information. And although modern mobile and web development actively complete the transition to the HTTPS-protocol of communication, nevertheless, you should not rely on a single line of protection in the form of a secure communication channel. The way to store important information on the server is still the safest, but choosing the wrong method or a sequence of data transfer can be a fatal error.

3) Rooting of the device and attack on the application and the algorithms used in it through external debugging tools. An attacker will always be able to get to the very depths of the program code and learn the method used to store, transfer and encrypt data.

The security requirements for mobile applications are not the same as for web applications. It is assumed that the user can work offline so you can not do without local storage of information on the device. For example, for online authorization with the storage of session data and cookies. After the identification data (login and password) have been entered and the application has authorized the user, it stores a special identifier, which is then presented to the server with each request coming from the application. If an attacker received a user ID and the system did not perform the procedures for checking the session IP address or having more than one connection within the session, the attacker could gain access to the system with the rights of the user's account. If these are applications related to Internet banking or to the personal cabinet of the payment system, then it is not difficult to guess the consequences of unauthorized access. Another problem is the inadequate control of client applications in some markets. This is the process of verifying software uploaded to Application stores. Before you go to the App Store, iOS-applications are checked in detail for vulnerabilities and compliance with Apple development standards. Each application installed on iOS must be signed with a special certificate "iOS Developer Program", issued by Apple only after a number of necessary checks. Such security measures ensure the absence of malicious software in the App Store. While in the Android operating system, applications are not scanned for malicious code before they are uploaded to the Google Play platform. Instead of the preliminary verification procedure, Google has implemented a mechanism for automatically scanning the application store for potentially harmful software. As practice shows, this method of analyzing information security increases the percentage of penetration of malicious applications and their further distribution to end users' devices.

III. CONSIDERATION OF VULNERABILITIES AND METHODS OF THEIR PREVENTION

Any vulnerability of the mobile application is based on shortcomings in the technology of operating with critical user data. Critical user data includes any data that should not be available to a third party, it concerns the user's personal data (date of birth, address of residence, personal correspondence) and his private data (passwords, credit card details, bank account numbers, order numbers and so on). Each time a mobile application is hacked is a race between an attacker and a developer in using technology and responding in a timely manner to threats and bugs. It is the developer and the technologies that he uses that determine the security of the application and the safety of user data.

In accordance with the classification of the Open Web Application Security Project (OWASP) [1], the main vulnerabilities of mobile devices are: systemic vulnerabilities (architectural solutions of mobile platforms); unsafe storage of data; insufficient security of data transfer protocols; the vulnerability of the authorization and authentication system; weak cryptostability; the vulnerability of the application code; hidden application functionality; improper control over client applications.

The most common mistake is the use of unprotected local storages. This vulnerability occurs very often, it is expressed in the storage of critical user data in unprotected or weakly protected local storages for a specific platform. Even if the application developer sets the level of access to the storage as private, prohibiting its use by other applications, the local storage can still be read from outside – by rooting the device or connecting it to a PC. In this case, the presence of special skills for the attacker is almost not required.

The second, by prevalence, vulnerability can be considered the storage of critical user data in the application code: in static constant strings, in application resources, etc. As examples: storing password salt in a constant or macro, which is used throughout the code to encrypt passwords; storage of a private key for asymmetric algorithms; storage of passwords and logins for server nodes or databases. Particularly it is relevant for applications written for the Android platform, but iOS applications are also subject to this problem. Such a vulnerability is easily revealed by a third party with basic code decompilation skills. It is important to take into account that the popular method of protecting the source code from decompilation – obfuscation, in no way protects applications from vulnerabilities, or can only partially slow down the search process [4].

In addition to storage, it is also dangerous to transfer critical data to the external environment in an open manner. This vulnerability is expressed in the transmission of data without the use of encryption on any available communication channel with the external environment, whether it is data transfer to a third-party application or transmission to the network. It can be opened indirectly by opening not the application, but its storage, or another application receiving this data. In this case, hacking success is demanding of the skills of the attacker, but only if the storage is protected. Any critical data before going outside the application must be encrypted. Local platform storages are not an application area, they must also receive only encrypted data, otherwise the security of the data can be compromised.

Using encryption of information is still not secure, there can be an application of algorithms with the storage of a private key. Vulnerability is relevant in the case when private information of the algorithm – private key, is stored in the code or resources of the mobile application, which often happens. Even despite the use of encryption in this case for an attacker, there is still a way to find a private key by decompiling, as in the previous vulnerability, and deciphering the algorithm. In mobile development, it is desirable to use only modern symmetric algorithms with a generated random one-time key that have high resistance to brute-force cracking, or to output an asymmetric private key outside the application, or to personalize this key, for example – a private key can be a user input code stored and encrypted in a protected operating system store. To implement this protection, there are many official libraries from Google that easily integrate into applications for both Android and iOS.

A similar vulnerability is the use of an asymmetric algorithm with a private key, known to the server. In this case, the danger may exist on the server side. This vulnerability is of a dual nature. Storage of a private key allows for decryption of user data on the server side. First, it is incorrect from the security point of view, if the server is hacked – the attacker will also have access to private user data, and secondly, it violates the privacy of personal data. The user should always be sure that his personal information is not known to anyone but himself, unless he explicitly gave permission to publish it. Often applications position themselves as protected, but in fact they are not, as they contain the means for deciphering personal information inside themselves. Without the explicit need and explicit permission of the user, most often through a license agreement, neither the application nor the server should have any opportunity to decrypt the user's private data. The simplest example is that the user's password must go to the server already in the form of a hash, and the hash should be checked, and not the original password, the server absolutely does not need to know the user's password; if the user forgot it, then for such a situation there is a long-established mechanism for recovering the password, including with two-factor authorization of the client for increased security of the recovery procedure [4]. Technically, as in the previous example, you can use the official libraries from Google that are designed to send and receive information from the server securely. Moreover, if you do not need to additionally process data before storing it, you can use ready-made server from Google such as Firebase Realtime Database or Firebase Cloud Firestore [3].

It should also avoid using self-written encryption and protection algorithms. Since this is a direct violation of the Kerkhoffs principle [5]. Expressed in the developer's attempt to invent his personal, not known to anyone, and therefore a super-secure encryption algorithm. Any deviation from existing, repeatedly verified and studied, mathematically proven encryption algorithms in 99% of cases turns into a rapid hacking of such protection. Of course, hacking such an algorithm requires the medium-high skills of the attacker, but does not guarantee complete security. For such solutions, it is necessary to select a suitable algorithm only from well-established and current well-known cryptographic algorithms. In addition, it is also necessary to pay attention to the relevance and timeliness of the chosen encryption algorithm. As the computing power grows exponentially, so the well-functioning algorithms of the last decade can already be susceptible to breaking through brute force on modern high-speed video cards following the progress.

A common mistake of developers is simply ignoring the fact of the presence of rooted or infected devices. Rooted devices are devices where modifications have been made to obtain superuser rights for any operations originally prohibited by the operating system manufacturer. It is performed by the user on his device independently, and not necessarily voluntarily, when the client may not be aware that the device was hacked. Installing the application on a handheld device removes all standard protection for the operating system. Especially this problem is relevant for users of the Android operating system. There is a huge number of special programs called "patchers" that in runtime can substitute data in the application or unlock the limited functions. Such operations are possible only on rooted devices and do not require special skills and technical knowledge from the attacker. In such cases, if it is technically possible for the platform, it is desirable to limit the application if it was recognised that the launch is made on rooted device, or at least warn the user of possible danger and leave the decision of the application start at the discretion of the user.

It is also not acceptable to store critical data in protected storage, but in an open form. Developers are often inclined to store data in protected system stores without additional protection, because system mechanisms are well resilient to hacking. However, their level of durability drops to a minimum in case the device is rooted. Such data should not be used in the application without additional encryption. As soon as the need for open critical data has disappeared – they must immediately be either encrypted or destroyed. Using such a vulnerability also requires a cracker to have high skills, but still retains risks.

There is also not a deliberate transfer of part of the functionality to the built-in web engines. Most often it looks like transferring critical data to the built-in browser, where an external web page is loaded, performing its part of the functionality. The level of protection in this case is sharply reduced, especially for rooting devices, even if a browser is used for a well-known authoritative company, because in this case it does not matter. So, you should not use the built-in browser and built-in web engine in operations with important data. In extreme cases, encrypt them before transmission.

In conclusion, it is worth mentioning the vulnerability associated with reverse engineering algorithms that have an intellectual value. If the development of the application within the company uses some of its own algorithms that can be of high value to potential competitors or hackers, then these algorithms must be protected from unauthorized access. In this case, you can protect yourself from this theft with the help of a banal automatic, or even, depending on the scale of the company, manual obfuscation of the code. In the obfuscated form, the algorithm may no longer represent the primary value for the attacker, or simply dissolve into the sea of obfuscated code.

IV. BASIC RECOMMENDATIONS FOR THE PROTECTION OF MOBILE APPLICATION

There are several common points for all mobile platforms that you should follow when developing. Often, a personal password is used to protect personal data: PIN code, fingerprint scan, graphic password, etc., in this case it is important when the application goes to the background or when folding it is necessary to immediately display the input window for this security code, overlapping the entire screen of the application. This eliminates the possibility for an attacker to obtain private information in the event of device theft, while the application is still running and is in sleep mode. Any user code should have a limited number of attempts to enter, for example 5 times, then, in case of failure, the application should automatically be logged off, or completely blocked, depending on the particular application [4]. At present, when using digital codes, it is strongly recommended to use a code length restriction of at least 6 digits [2].

In the case of client-server applications, it is very useful to use a session mechanism with a limited session lifetime. This will prevent the application from idle in an unprotected mode, if the user simply forgot to close it and left the device in the public domain. It should be borne in mind that the duration of the session and its identifier are relevant to the user's critical data, with all the ensuing consequences. One of the successful examples of implementing such a mechanism is getting the absolute value of time from the server after passing the user authorization procedure, in which case the date and time should indicate when the session will become inactive. It is important to remember that the session end date and time should not be generated on the device, this reduces the security and flexibility of the application. In addition, the client-server application should not change the critical data in the local mode. Any action that requires changing important data must be synchronized with the server. The exception to this rule is only the user-defined login code that is set by the user personally and stored in a secure local store.

An important aspect is the competent operation of dates. When you run an application with dates, such as the time to destroy a session, you should not rely on relative time. So, the data transmitted from the server should not contain the date in the form of N seconds / hours / days from the current moment. Due to potentially high delays in transferring data over the network from the mobile application to the server and back, such a synchronization method will have too much error. In addition, an attacker, or simply an unscrupulous user, can simply change the time zone on the device, thus violating the logic of the application's limiting mechanisms. It is always needed to transmit only the absolute value of time. Absolute values should be transmitted using universal ways of exchanging such information, without reference to the time zone of a particular user device. Most often, the best option is the behavior of the application, in which the data is displayed to the user in its local time zone, but their storage and transmission is done in a format not tied to the time zone. Suitable formats for dates and times are either a universal UNIX timestamp stored in a variable of a 64-bit integer sign type, or, in extreme cases, a string in the full ISO-8601 format with a zero time zone. The UNIX timestamp is preferred, it avoids potential errors and problems with converting strings to date and back on different mobile platforms [3].

If we talk about general security principles, then the application should not display private user information in large, bright, well-readable fonts, without explicit need and without a separate user request, to exclude the possibility of reading this data from the device's screen.

Do not blindly trust open source libraries that offer some kind of protection for private user data. The only exception is libraries that are time-tested and used in large corporate projects, for example, built-in encryption in the open engine of the Realm database. The standard mechanisms of protection of the operating system and publicly available cryptographic algorithms in the overwhelming majority of cases will be more than enough.

It is also better not to use cryptographic libraries with closed source code, even if they are paid. In such solutions, you can not in any way check whether the library is effective, or how honest it is protecting, whether there is a backdoor mechanism, or whether your data is being sent to some third party. For the release builds of applications, data logging to the system console and unprotected files should be disabled. Logs are log files containing records of all events occurring in the mobile operating system, with a high level of detail. On the device, any application during installation can request access rights to read the logs. Many users do not pay attention to this request, but the danger is that any installed application that requested access to read logs, and at the same time received approval from the user, will get the right to read all the information that the application puts into the logs, if logging not turned off by the user.

CONCLUSIONS

For each separately created mobile application, the number of applied protection levels will vary greatly. For example, if the application is not at all client-server, does not contain any critical data, and does not operate with valuable internal algorithms, then it makes no sense to attach any protection to it. If the application is oriented, for example, to performing banking operations, or storing user passwords, then the level of its security should be the highest. However, the previously mentioned common vulnerabilities of the mobile sector can easily be excluded from the application, most often it does not add any additional costs if the applying of the required level of protection was started at the early stages of application development. But the implementation of post-factum protection in an already running application may well be associated with significant expenses of the developers' efforts and time. Therefore, the choice and coordination of the level of security, as well as a list of critical data in the application being developed, should be performed at the earliest stages of design.

REFERENCES:

[1] K. Anton, J. Bird and J. Manico, *Top-10 Proactive Controls 2016*, The OWASP Foundation, 2016. [E-book] Available: OWASP e-book.

[2] ENISA, *Smartphone Development Guidelines*, European Union Agency for Network and Information Security 2017. [E-book] Available: ENISA e-book.

[3] Firebase, “Build apps fast, without managing infrastructure”, *Firebase*. [Online]. Available: https://firebase.google.com [Accessed: Apr. 10, 2018].

[4] NowSecure, *Secure Mobile Development Best Practices*, NowSecure Inc., 2016. [E-book] Available: NowSecure e-book.

[5] Victor de Casto, *Just cryptography*. St. Petersburg: Strata, 2014.