J Med Syst (2016) 40: 152
DOI 10.1007/s10916-016-0513-6

MOBILE SYSTEMS

# Security Recommendations for mHealth Apps: Elaboration of a Developer's Guide

**Enrique Pérez Morera**[1] · **Isabel de la Torre Díez**[1] · **Begoña Garcia-Zapirain**[2] · **Miguel López-Coronado**[1] · **Jon Arambarri**[3]

Received: 22 April 2016 / Accepted: 28 April 2016 / Published online: 4 May 2016


**Abstract** Being the third fastest-growing app category behind games and utilities, mHealth apps are changing the healthcare model, as medicine today involves the data they compile and analyse, information known as Big Data. However, the majority of apps are lacking in security when gathering and dealing with the information, which becomes a serious problem. This article presents a guide regarding security solution, intended to be of great use for developers of mHealth apps. In August 2015 current mobile health apps were sought out in virtual stores such as Android Google Play, Apple iTunes App Store etc., in order to classify them in terms of usefulness. After this search, the most widespread weaknesses in the field of security in the development of these mobile apps were examined, based on sources such as the "OWASP Mobile Security Project, the initiative recently launched by the Office of Civil Rights (OCR), and other articles of scientific interest. An informative, elemental guide has been created for the development of mHealth apps. It includes information about elements of security and its implementation on different levels for all types of mobile health apps based on the data that each app manipulates, the associated calculated risk as a result of the likelihood of occurrence and the threat level resulting from its vulnerabilities - high level (apps for monitoring, diagnosis, treatment and care) from $6 \leq 9$, medium level (calculator, localizer and alarm) from $3 \leq 6$ and low level (informative and educational apps) from $0 \leq 3$. The guide aims to guarantee and facilitate security measures in the development of mobile health applications by programmers unconnected to the ITC and professional health areas.

**Keywords** Apps · Developers' guide · mHealth · Security

This article is part of the Topical Collection on *Mobile Systems*

✉ Isabel de la Torre Díez
isator@tel.uva.es

[1] Department of Signal Theory and Communications, and Telematics Engineering, University of Valladolid, Paseo de Belén, 15, 47011 Valladolid, Spain

[2] University of Deusto, Avenida de las Universidades 24, 48007 Bilbao, Spain

[3] VirtualWare Labs Foundation, C/ Usausuaga, 7, 48970 Basauri, Vizcaya, Spain

## Introduction

Nowadays there are numerous *smartphone* manufacturers whose total sales between 2015 and early 2016 [1] reached 1.6 billion, with Google Android [2] and Apple iOS [3] controlling 97.8 % of the market [4]. The recent study by the GSMA (*Groupe Speciale Mobile Association*) [5] forecasts that by 2020 the number of connected *smartphones* could be six billion, thanks to Chinese manufacturers, whose terminals are of a reduced price, and the emerging markets of countries such as India, China, Indonesia and Brazil.

The reach of the mobile is greater than ever and the conclusions of the Cisco VNI Mobile report [6] prove this, claiming that in 2020 there will be 11.6 billion connected devices (*smartphones*, *tablets and wearables*), of which 72 % will be intelligent. In Spain, this figure will reach some 105.5 million – the equivalent of at least two devices per person -and the most prolific and most profitable market over the next 2 years will be that of *wearables*. According to McKinsey & Company [7], the boom in these connected devices means that the figure of 130 million fitness wearables connected today could rise to 1.3 billion in 2025. This is thanks to the evolution of mobile broadband, whose active subscribers rose by 47 % in 2015, offering 3G coverage to

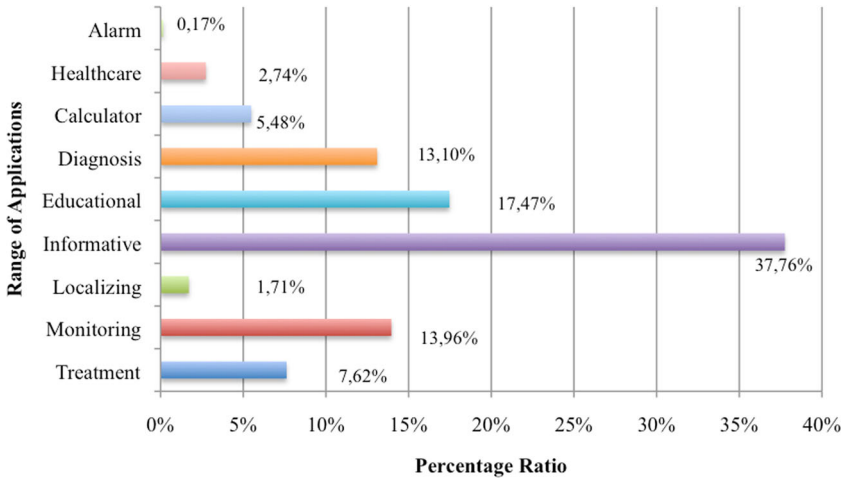


**Fig. 1** Percentage of applications per utility and type of illness

69 % of the world population [8]. It is also thanks to the mobile app market which offers solutions to the diverse needs of users. These solutions are intended for and adapted to all types of device, and therefore applicable to health through mobile health apps or mHealth.

The WHO defines this as: "*the practice of medicine and public health supported by mobile devices such as mobile phones*, *patient monitoring devices*, *personal digital assistants and other wireless devices*" [9]. Interest in mobile health applications grows year after year. In 2015 [10] it was estimated that three billion mobile health apps were downloaded from the principal app stores.

The success of these applications lies in the fact that what mobile devices can do today is to make gathering the information about patients easier and they can help in the early detection of outbreaks of diseases, thanks to their numerous sensors. They also ease communication with health professionals, leading to a new kind of doctor-patient relationship with numerous benefits to both, from optimising consultation times to improving therapeutic adherence and chronic patient monitoring. As affirmed by Doctor John W. Denniger in the IOT Solutions World Congress in 2015 [11], healthcare today involves analysis of information collected by *smartphones* and *wearables* through diverse apps, as their potential is enormous because, unlike the patient, they do not lie. However, 2015 was blemished by a high number of security breaches (some 250), with almost 220 million credentials leaked, which represents a threat both to users and other companies [12]. The *Bitglass 2016 Healthcare Breach Report* [13] claims that in the USA alone, 111 million individuals suffered hacking aimed at data related to health. An increase by 80 % of personal data breaches through hacking illustrates the enormous interest in medical data by cyber-criminals due to its high black-market value along with the possibility of being able to access medical attention by pretending to be the victim or by extorting a company. The average amount paid for each record lost or stolen containing sensitive or confidential health information went from $145 in 2014 to $154 in 2015, according to a study by the Ponemon Institute [14]. This overall average cost rises to $363 if the lost or stolen health data comes from a healthcare organisation.

This situation stems from a lack of security. 65 % of organisations claim that security is often compromised by the demands of the client and 77 % mention "*a rush to launch*" as doctors and patients are adopting mobile technology faster than the suppliers can protect their security and privacy, the main reason why apps contain vulnerable codes. According to a recent survey by the HIMSS, *Healthcare Information and Management Systems Society* [15], the clinical use of mobile technology to gather data rose to 67 %, compared to the 45 % of the previous year, and 93 % of doctors use their personal *smartphone* [16], known as BYDO (*Bring Your Own Device*),

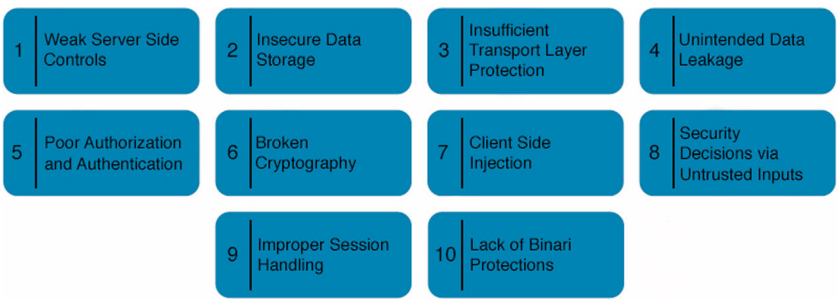


**Fig. 2** Ten vulnerabilities to consider in the development of health apps

to access the Electronic Health Record (EHR). But only 57 % did so under a formal mobile policy, causing a significant problem as a result.

Therefore the aim of this article is to prepare a guide with recommendations for security in the design and development of mobile health *apps*, based on existing health applications in relation to the most fatal illnesses of the present time adjudged by the WHO, and also on various studies on security regarding the most widespread vulnerabilities in the mobile application area.

The following are the points which will be discussed: Firstly, the methodology employed in the research work when searching for applications will be addressed, presenting the stores where to look and the selection criteria, along with ten

**Table 1** Ten vulnerabilities in mobile health apps

| Measure | Description |
|---|---|
| Weak Server Side Controls | Corresponds to malicious code injections in a server to exploit the app. Injection flaws such as SQL and LDAP occur when untrustworthy input is sent to an interpreter as part of a command or query. The hostile data can trick the interpreter into executing unintentional commands and access unauthorised data. |
| Insecure Data Storage | Mainly involves lost or stolen mobile devices, although there is the possibility to access these devices without physically having the device through exploits-in-the-wild and/or different malicious codes, or the use of any *jailbreak* or *rooting* which circumvent any encoded information. |
| Insufficient Transport Layer Protection | When designing an application, data is commonly exchanged between the client and the server. Mobile applications frequently do not protect network traffic. They may use SSL/TLS during authentication but if the coding is weak, various techniques exist to intercept sensitive data while it -is travelling between client and server. |
| Unintended Data Leakage | Mobile applications have to interact with operative systems, digital infrastructures, new hardware etc. which are not the property of the developers, so they cannot control changes or flaws which are outside their applications [33]. Therefore, it is possible for data to be lost if evaluation is not carried out to understand how applications interact with all the elements of the device. |
| Poor Authorization and Authentication | Authentication patterns exist which are considered unsafe and which must be avoided. Unlike traditional web applications, mobile applications do not expect the user to be online at all times to authenticate through a backend server, but do so off-line and are liable to suffer binary attacks bypassing the authentication of the application. The functions of the application relating to authentication and session management are frequently applied incorrectly, allowing cyber-attackers to compromise passwords, keys or session tokens or to exploit other implementation flaws to assume the identity of other users [33, 34]. |
| Broken Cryptography | On occasions data encryption methods become an almost obsolete practice. Creating and utilising its own encryption algorithms and using outdated algorithms are examples of malpractice. There are two fundamental ways encoding errors are manifested in mobile applications. Firstly, the mobile application may use an erroneous encryption/ decryption process behind the operative system which can be used by the cybercriminal to decipher data using tools such as ClutchMed or GBD. Secondly, the mobile application can apply or use a cipher/decipher algorithm which is weak by nature using tools such as IDA or Hopper. Many algorithms and cryptographic protocols should not be used because it has been demonstrated that they have significant weaknesses or they are insufficient for the necessities of modern security. These include RC2, MD4, MD5, and SHA1 [35, 36]. |
| Client Side Injection | As long as there is the possibility that external users, internal users or the application itself can send untrusted data to the system, an adversary can inject simple exploits (text-based) such as SQL Injection to mobile applications, causing the potential risk of information theft. |
| Security Decisions via Untrusted Inputs | To understand this case better, we have to understand the concept of IPC, Inter Process Communication better. Processes between applications and the operative system share memory space to permit mutual communication and synchronisation. To reduce attack risk, the mobile application should only allow communication with other trusted applications, sensitive operations should require interaction with the user, sensitive information should not be sent through the IPC etc. |
| Improper Session Handling | Mobile applications use session tokens to maintain state through protocols such as HTTP or SOAP. To maintain state the mobile application must first authenticate the user through the backend. In response to successful authentication, the server issues a session cookie to the mobile app. The mobile app adds this cookie to all future service transactions between the mobile app and the server. This allows the server to conveniently enforce authentication and authorisation for any service requests issued by the mobile application. Improper session handling occurs when the session token is involuntarily shared with the cybercriminal during a subsequent transaction between the mobile app and the backend servers. This is why it is so important to handle the session correctly once it is opened. If small but important cautions are not taken, non-authorised third parties with access to the HTTP/S traffic of the app or cookie data can intercept other user's information. |
| Lack of Binary Protections | Lack of binary protection allows an attack through reverse engineering. If a programmer is not the creator of the programme code at binary level and it is not protected, an adversary can easily find flaws in the code, copy it, make minor changes and re-sell a new mobile app as if it were his own. |

**Table 2** Impact levels according to probability of occurrence of each vulnerability

| Probability of occurrence and levels of impact. | | |
|---|---|---|
| 6≤9 | High | Vulnerability which if exploited would endanger all the data of the user. Must be remedied immediately. |
| 3≤6 | Medium | Vulnerability which if exploited would have a light impact on the user. Solution should be found within a prudent |
| 0≤3 | Low | Vulnerability which if exploited would not cause major inconvenience. A remedy need not be immediate. |

essential security blocks to bear in mind regarding the vulnerabilities discovered. Afterwards security results by level (high, medium or low) will be shown for the various types of mobile health *apps*, and lastly, the conclusions arrived at, based on the most relevant characteristics of the study, will be discussed.

## Methods

The search for applications carried out centred on those commercial applications available to the public in general in the various stores in the main platforms in August 2015 [17, 18], which were – in order of market share – Android's Google Play [2], Apple's iTunes App Store [3], Microsoft's Windows Phone Apps + games [19], BlackBerry's World of BlackBerry (former RIM or *Research In Motion*) [20] and others such as Nokia's Ovi Store [21].

Altogether, in all the mobile app stores, there are more than 165,000 *app*s [22] related to health and medicine, making it the third highest-growing category behind games and utilities, according to the "*Patient Adoption of mHealth*" report published by the *Institute for Healthcare Informatics* (*IMS*).

Based on this premise, the guidelines followed when examining the results of the search carried out were to discard those *apps* which were not available in English or Spanish and to only accept those *apps* grouped under the heading "medicine" or "health and wellbeing" (health and physical fitness) as valid, as 65 % of health apps [22] are aimed at the general public, being related to physical exercise and wellbeing, while 35 % are aimed at professional health workers and their patients. The most common feature which applications related to health offer is to provide information, which represents more than two thirds of all *mHealth*, applications, and their principal use is related to prevention and lifestyles.

The search carried out for commercial applications for a total of 26 illnesses, according to the WHO [23, 24], shows that in 2013 (the last year for which figures are available) [25] there were 2837 valid applications available on different mobile platforms and up until today there has been a growth of 70 % [22], which gives an estimated 4823 applications for these types of illnesses.

Google Play rendered the highest number of results, closely followed by the Apple App Store [26], which has grown by 106 % since 2013, from having 43,689 apps to boasting a catalogue of 90,088 health-related apps. As regards the operative system utilised, Android is notably predominant. In *smartphone*s 2015Q2 [27], Android represented 82.8 % of them, followed by the iOS operative system with a presence of 13.9 %. However, as regards *tablet*s [26], Android reaches 66 %, while iOS makes up 25.5 %.

Diabetes is the illness with the highest number of related applications, followed in second place by cardiovascular conditions. Other illnesses then come way below these two, divided into three blocks depending on the number of apps. The first block consists of Alzheimer's disease, HIV, asthma, throat, bronchial and lung cancer, and strokes. The second group is made up of breast cancer, osteoarthritis, gastrointestinal illness, loss of vision, lower respiratory infections, hearing loss and chronic obstructive pulmonary disease (COPD). The last group includes areas with few, if any, available apps, such as anaemia, malaria, colon and rectal cancer, tuberculosis, measles, premature birth, traffic accidents, intestinal infections, malnutrition, whooping cough and disorders caused by alcohol.

**Table 3** Causative Threat Agents

| Causative threat agents. | | | |
|---|---|---|---|
| Technical skills | Motivation | Opportunity | Size |
| No knowledge (1) | Nothing of interest (1) | No access (0) | Developers (2) |
| Some knowledge (3) | Some interest (4) | Limited access (4) | System admin (2) |
| Advanced user (4) | Appreciable interest. (9) | Special access (7) | Internal users (4) |
| Networks and programming (6) | | Total access (9) | Business partners (5) |
| Security intrusion (9) | | | Authenticated users (6) |
| | | | Anonymous users (9) |

**Table 4** Factors which cause vulnerability

| Factors which cause vulnerability | | | |
|---|---|---|---|
| Ease of discovery | Ease of exploitation | Knowledge | Detection of intrusion |
| Practically impossible (1) | Complex (1) | Unknown (0) | Detection active in the *app* (1) |
| Difficult (3) | Medium Difficulty (3) | Hidden (4) | Authenticated and monitored (3) |
| Easy (7) | Simple (5) | Obvious (7) | Authenticated without monitoring (8) |
| Automated tools available (9) | Automated tools available (9) | Public knowledge (9) | Not authenticated (9) |

After carrying out a revision of the different types of *apps*, the percentage is obtained of applications according to their use, as shown in Fig. 1. Not having to apply any special characteristic, the most abundant applications by far are informative, making up [25] almost four in every ten apps (37.4 %). Educational apps are in second place (17.3 %). In essence, these two sections are the same, but they change the way of focusing on and showing the information to the user. Putting them together as one, apps of this type reach a figure of 54.78 %. That is to say, one of every two applications offers information or more scientific content to the user and they are aimed principally at health professionals. Tools for monitoring physical parameters and diagnostic aid are second in importance, with both reaching around 14 %. They contain ample medical content, but are conceived to support patients and/or relatives or carers in the course of an illness or medical problem. The rest of the classes are not so visible, being those which have a more general content, helping in primary prevention and care of the health aimed at society in general: those which aid in following treatment (7.65 %), calculators (5.57 %) healthcare (2.78 %), localizing (1.74 %) and alarms (0.17 %). These last two are used for Alzheimer's and cerebrovascular disease respectively.

The security elements which application developers should apply will be explained shortly. These measures should be implemented to guarantee the security and privacy of the personal information of the user, complying with current security and privacy guidelines such as ISO/IEC 27001/2013 regarding security of information [28], drawn up by leading specialists in the field and providing a methodology to implement when managing company information. Supported by the "*OWASP Mobile Security Project*" [29] of 2014, the article "*Privacy and Security in Mobile Health Apps*: *A Review and Recommendations*" [30] and the initiative recently launched by the Office of Civil Rights (OCR) [31, 32], as can be seen in Fig. 2 and is explained in Table 1, we will identify the ten vulnerabilities which affect mobile application security, using many standards, books, tools and organisations as references, including MITRE [37], PCI DSS (*Payment Card Industry Data Security Standard*) [38], DISA (*Defence Information Systems Agency*) [39], FCT (*Federal Trade Commission*) [40], y and many more.

## Results

In this section we will carry out a security assessment considering the real risk of suffering an attack through vulnerability. *OWASP* [41] recommends the use of a general methodology to break down security discoveries and evaluate risks to prioritize and deal with them. The first step is to identify a security risk which needs to be addressed. To do so, information needs to be compiled about the agents that are causing the threat, the type of attack they are using, the vulnerability involved and the impact of a successful exploitation on the app.

When risks have been recognised, the probability has to be estimated as to whether a particular vulnerability will be discovered and exploited. Initially it is recommended that qualitative parameters of grading be defined to estimate likelihood. To calculate with greater certainty, quantitative calculation is recommendable.

**Table 5** Technical Impact Factors

| Technical impact factors | | | |
|---|---|---|---|
| Confidentiality | Integrity | Loss of availability | Responsibility |
| Minimum disclosure of non-sensitive data (2) | Slightly corrupt data (1) | Minimum (non-critical services) (1) | Some auditable (1) |
| Minimum disclosure of sensitive data (6) | Minimum seriously damaged data (3) | Minimum (critical services) (5) | Auditable (7) |
| Ample disclosure of non-sensitive data (6) | Large amount of slightly damaged data (5) | Considerable (non-critical services) (5) | Anonymous (9) |
| Ample disclosure of sensitive data (9) | Large amount of quite damaged data (9) | Considerable (critical services) (7) | |
| | | All services lost (9) | |

**Table 6** Security levels according to app type, user data and site of information

| App type | User data | Site data | Security level | Likelihood |
|---|---|---|---|---|
| Informative | No | App/sometime on the server | Low | 0≤3 |
| Educational | No | App/sometime on the server | Low | 0≤3 |
| Monitoring | Yes | App and Server | High | 6≤9 |
| Diagnosis | Yes | App and Server | High | 6≤9 |
| Healthcare | Yes | App and Server | High | 6≤9 |
| Calculator | Yes | Mainly in the App | Medium | 3≤6 |
| Treatment | Yes | App and Server | High | 6≤9 |
| Localizing | Yes | App and GPS | Medium | 3≤6 |
| Alarm | Yes | Mainly in the App | Medium | 3≤6 |

Once a potential risk has been identified, the first step is to estimate the overall and technical "occurrence likelihood" to figure out how serious it is. Generally, it is sufficient to identify whether the likelihood is low, medium or high. To determine the severity of risk, it is necessary to work with the likelihood of the threat and the general impact on the application, as shown in Table 2.

There are various factors to help to carry out this estimation. The first group of factors, shown in Table 3, are related to agents that cause the threat involved. The objective is to estimate the probability of a successful attack by a group of possible cyber-attackers. There could be multiple agents that exploit one vulnerability in particular, shown in Table 4, so generally it is best to consider a worst case scenario. Each factor has a combination of options and each option has an associated valuation from 0 to 9 on their likelihood of occurrence. These figures are used at a later point to estimate overall and technical likelihood which advises us on which security level, high, medium or low, to employ.

When a threat is materialised, technical impact must be considered, as can be seen in Table 5. The factors must be considered and those which are key must be identified, as they are influencing the result. It is possible that the first impression was incorrect after considering risk aspects which were not obvious. There is a fair amount of uncertainty in these estimations and the purpose of these factors is to help in arriving at a reasonable result. The first step is to choose one of the options associated with each factor. Afterwards we simply calculate the average score for each one of the factors obtaining, as a result, the technical and the overall likelihood of occurrence. Automated tools which give the calculations may be used for this process. Once again, less than 3 is considered low, 3 to 6 is medium and 6 to 9, high.

Finally three levels of security are given as the result (high, medium, low) for each type of app looked at in the previous section.

As can be seen in Table 6, the app type is shown according to its function (informative, educational, monitoring, diagnosis, healthcare, calculator, treatment, localising and alarm), the details of the user manipulating each app, the place where the information is treated and the associated security risk, the likelihood and level of impact if a possible threat emerging from a vulnerability were to occur. Within each security level (low, medium, high) the desire is to bring a series of clear and specific security guidelines to the reader, a visual list of basic recommendations for medical personnel and amateur application developers, who previously paid little attention to this issue through lack of awareness, to bear in mind.

A look at Fig. 3 shows the three essential aspects to consider in the communication between a *smartphone* or *tablet*, where the health app is run, and the server (cloud/backend), where normally user data compiled from the *apps* is stored. Between them we can see that the communication channel, in the case of mobile technology, is not a cable but the air which transmits the information via electromagnetic waves, making


**Fig. 3** Diagram of the essential elements in communication between devices

use of mobile telephone networks or *Wi-Fi* to connect to the Internet. Where the app is informative or educational, the user information which the app handles is practically nil (with some exceptions), being a mere portal in which to see informative or educational content which is contained on the *app* or, failing this, downloading from a server through an internet connection. Afterwards, Table 7 shows the security measures to implement.

The next level, shown in Table 8, is composed of calculator, localizer and alarm -type apps, which have a medium level of security requisites as they deal with a low amount of user information of even less relevance, with the exception of the Global Positioning System in the case of localizers. The great part of this information is dealt with by the app itself from the mobile device.

Finally, in Table 9, the types of applications shown are for monitoring, diagnosis, treatment and healthcare, which are in the highest level of security because of the full nature of the information the mobile app handles. User information or data is found both in the mobile device, through the application, and the server. Data is transferred *Full-duplex* via the Internet thanks to an infrastructure of 3G/4G mobile telephone antennas or Wi-Fi networks, whose main channel or means of transmission is the air. In this case, extreme caution must be practiced, focussing attention on the app, the channel and, finally, the server.

The security implementation is a guideline. Application features can vary in their different interactions, with functions added or removed which require the security measures of one level when at first security was considered to be of a lower or higher level. In any case, besides requirements, it is recommended periodic checks, preferably performed by external companies, of the security and privacy policies carried out by the designers. In this way, the designers can certify that their policies meet with legal privacy and security requirements.

**Table 7** Security measures for a low security level

| Low security level: informative and educational apps |
|---|
| App (*Frontend*) |
| Control access to information (contacts, camera, localization) through permissions |
| Recommend use of firewall, for both iOS and Android in the *app* |
| The app should not store *Cookies* |
| Hide the *buffer* where the content copied and pasted by the *App* is stored |
| Allow deactivation of running of the application in the background |
| Avoid reverse engineering of the app, obfuscating the app code |
| React in case of a code integrity violation preventing it being modified |
| Programme the app with trusted third party libraries |
| Distribute the apps in secure repositories (App Store, Google Play…) |
| Validate all information input and output to the app, avoiding cases such as *QR Code Leaks* |
| Use secure functions to prevent buffer overflow |
| Test applications running them on the minimum privilege level |
| Make security updates regularly through patches |
| Provide channels (email) to report problems with the App |
| Use static code analysers and *fuzzers* when developing the *App* |
| Communication Channel |
| Establish a secure communication channel such as SSL/TLS with 128 bit encryption |
| Server (*Backend*) |
| Analyse the server periodically and keep it updated |

## Discussion

If we stop to ask how many inexperienced developers or people foreign to the area of ITC are aware and take adequate measures to meet a series of standards related to security in their apps, many would immediately say none. This research work explores this field which is so important in the development of mobile health applications, summing up measures seen in 10 guidelines for the development of secure health apps, as seen in Fig. 4 and explained in Table 10.

The *low level* of security is aimed at *informative* or *educational* apps. These hardly use user data because their main function is to provide complete and detailed information about a particular pathology or area of medical specialisation, be it in form of a text, image or video. They also provide updated information about a certain illness, found in the app itself or downloaded from a server on the Internet, providing active education to the patient or public to which it is aimed. These applications do not need constant use of the Internet through mobile networks or WIFI to run or make use of its various functions. Aspects to bear in mind are shown in Table 11.

The *medium level* of security, shown in Table 12, involves apps for *calculation*, *localization* and *alarm*. This type of mobile health app enables the calculation of body mass index, basal metabolic rate, maximum heart rate, the amount of alcohol in the blood, the recommended antipyretic dosage according to child weight etc. They can also ask for help or orientation rapidly and effectively by means of GPS, or alert with an alarm about any important occurrence regarding health. Only on occasions do they make use of the Internet except in the case of localizers, which need continuous connection to utilise GPS services. A characteristic of the medium level of security is that more use is made of user information, either introduced manually by the user or obtained by a wearable capable of quantifying health parameters.

The *high level* of security corresponds to applications of *monitoring*, *diagnosis*, *treatment* and *healthcare*. They help us

**Table 8** Security measures for a medium security level

| Medium security level: calculators, localizers and alarm apps |
|---|
| App (*Frontend*) |
| Control access to information (contacts, camera, localisation) through permission |
| Recommend use of *Firewall* for both *iOS* and *Android* in the *app* |
| The app should not store *Cookies* |
| Hide the *buffer* where content copied and pasted by the *App* is stored |
| Allow deactivation of running of the application in the background |
| Use of access passwords, *Pin Scramble* or fingerprint to verify user identity |
| Do not send data to third parties, or, failing that, inform user |
| Session timeout after 30 min maximum in the case of connection to a server |
| Avoid reverse engineering of the application, "*obfuscating*" the app code |
| React in case of a code integrity violation preventing it being modified |
| Encrypt the code. *Apps* must be encoded and signed by trusted sources |
| Make security updates regularly through patches |
| Avoid that credentials (personal data) be visible in the cache or in the code |
| Avoid simple *exploits* such as SQL injection |
| Avoid *Cross-Site Script Attacks* (*HTML* modifications through *malware*) |
| Avoid *Cross-Application Scripting Attacks* |
| Use static code analysers and *fuzzers* when developing the *App* |
| Include privacy policies which are easy for the reader to read and understand |
| Include a section for minors with the requirement of approval from a legal tutor |
| Leave the policy accessible in an application file of easy access |
| Compiled data should be temporary and be eliminated after the app objective is completed. |
| Programme the application with trusted third-party libraries |
| Distribute the apps in secure repositories (*App Store*, *Google Play*…) |
| Make security updates regularly through patches |
| Validate all information input and output to the *App*, avoiding cases such as *QR Code Leaks* |
| Use secure functions to prevent buffer overflow |
| Test applications running them on the minimum privilege level |
| Deleting application data after 10 attempts invalid password |
| For applications that contain sensitive data, apply anti debugging techniques |
| Do not store sensitive data on the key iOS devices |
| Avoid iOS apps screenshots because they are stored in cache |
| Provide channels (email) to report problems with the App |
| Localization icon in the *App* which advises the user the use of its whereabouts |
| Communication Channel |
| Establish a secure communication channel such as SSL/TLS with 128 bit encryption |
| Use cryptographic methods to obtain *BSNs* in the case of pulse monitors (wearables) |
| Use robust, certified, signed and trusted encoded algorithms |
| Server (*Backend*) |
| Analyse the server periodically and keep it updated |
| Store information preferably in the server (*backend*) and not in the *App* |
| Destroy the session initiated by the user on the server side and do not accept previous cookies |
| Avoid injections and binary attacks (*DoS*) in the server |

to take control of our wellbeing through the quantifying of all our physical parameters, making the process of identifying a particular illness or medical disorder easier by providing valuable data to health professionals. They also help in the control and treatment of illnesses and the taking of medication, and enable more continuous and immediate attention than a traditional consultation. They also improve healthcare management, providing immediacy in referral of treatment and clinical observation of the patient, observing their condition at all times.

In short, this study shows that this type of application needs the Internet to function. They send information obtained about the user, and doctors and other users receive this data. It is for this reason there must be a high security level both on the server side and in the application, not forgetting the means of communication, as shown in Table 13.

## Conclusion

When mobile applications access personal data, documents or information which is often stored in the device, the risk of data theft grows when the device is lost or when the data is shared with other applications. It is fundamental that developers invest and implant their *apps* with a "remote wipe" capacity to deal with lost or stolen devices, and to eliminate any *PHI* on the server once the objective of the app has been fulfilled and notify it has been done.

Furthermore, data which mobile applications have access to must be encoded and controlled (*whitebox*), to safeguard the data against malware and other means of access. Another fundamental aspect is that of control of access, giving the user the possibility of giving permissions to elements of both software and hardware to those who can access an app at any moment.

As these types of mobile applications allow the user to transfer information (personal data) with hospital services, the risk level is high. Developers must adapt their applications according to how the information is used, being conscious of the risks and restricting their functionality through policies which take into consideration risk factors such as the security

**Table 9** Security measures for a high security level

| High security level: monitoring, diagnostic, treatment and healthcare apps |
|---|
| App (*Frontend*) |
| Control of access to information (contacts, camera, localization…) through permission |
| Recommend use of *Firewall* in both *iOS* and *Android* in the *app* |
| The app should not store *Cookies* |
| Hide the *buffer* where content copied and pasted by the *App* is stored |
| Use access passwords (7 digits: alpha-numeric with some characters) or *Pin Scramble* |
| Allow deactivation of running the application in the background |
| Do not store passwords in flat files and confirm their entropy |
| Do not send data to third parties, or, failing that, inform user |
| Verify the identity of the user via an authentication system or biometric parameter. |
| Authentication with a unique *ID* linked to a *PKI*, *RSA* or symmetric key and password |
| It is recommended to use authentication in two stages using *TOTP* |
| Give authentication through a start of session *API* through *HTTPS* |
| Create, maintain and eliminate session *Tokens* (long, complex and quasi-random) through *OAuth* |
| The session should terminate in 15 min at a maximum |
| Avoid reverse engineering of the application,"*obfuscating*" the app code |
| React in case of a code integrity violation preventing it being modified |
| Enable mobile device encoding by means of *whitebox* |
| Encrypt the code. The *Apps* have to be encoded and signed by trusted sources |
| Do not store hard-coded password in the code |
| Avoid that credentials (personal data) be visible in the cache or in the code |
| Use AES of at least 192 bits to encode the *PHI* |
| Avoid simple *exploits* such as SQL injection |
| Avoid *Cross-Site Script Attacks* (*HTML* modifications by means of *malware*) |
| Avoid *Cross-Application Scripting Attacks* |
| Use static code analysers and *fuzzers* when developing the *App* |
| Use Public Digital Signature based on passwords |
| Include privacy policies which are easy for the user to read and understand |
| Include a section for minors with the requirement of approval from a legal |
| Leave the policy accessible in an application file of easy access |
| Allow remote cleaning or deactivation of the application and its contents |
| Have a clear data retention policy in an easily accessible application file |
| Gathered data should be of temporary character. Eliminate after the app objective has been completed |
| Programme the application with trusted third party libraries |
| Use the *javax.crypto* library to encode flat file data in mobiles with SD card |
| Distribute the applications in secure repositories (*App Store*, *Google Play*…) |
| Make security updates regularly through patches |
| Report security violations (of information) to the user and the media |
| Validate all information inputs and outputs to the app avoiding cases such as *QR Code Leaks* |
| Use secure functions to avoid *buffer* overflow |
| Test applications running them on the minimum privilege level |
| Deleting application data after 10 attempts invalid password |
| Integrate a CAPTCHA solution to improve functionality / security app |
| For applications that contain sensitive data, apply anti debugging techniques |
| Do not store sensitive data on the key iOS devices |
| Avoid iOS apps screenshots because they are stored in cache |
| Do not publish your digital certificates in the app |
| Provide channels (email) to report problems with the App |
| Localization icon in the *App* which advise the user of its whereabouts |
| Icon in the *App* which informs of the transfer of data |
| Communication Channel |
| Establish a secure communication channel such as SSL/TLS with 128 bits encryption |
| Use cryptographic methods to obtain *BSNs* in the case of *wearables* |
| Use *VPN* (Virtual Private Networks) or HTTPS logins |
| Audit communication mechanisms in search of possible information leaks |
| Invalidation of session must be performed in the server |
| Use robust, certified, signed and trusted encoded algorithms |
| Server (*Backend*) |
| Analyse the server periodically and keep it updated |
| Eliminate the *PHI* of the server once the objectives of the application is complete and report it |
| Store information preferably in the *back end* server |
| Destroy the session initiated by the user on the server side |
| *Cookies* from previous sessions will not be accepted |
| Avoid injections in the server |
| Verify the identity of the user before executing the service offered by the application |
| Ensure that the server rejects all requests unencrypted |
| Measure risk by correlating factors such as IP address |
| Take measures to avoid denial of service attacks (*Binary attacks - DoS*) |

of the device, the location of the user and the security of the network connection, among others.

Risk factors can be measured by correlating information. For example, it would seem very suspicious if an IP address was accessed twice in a short period of time from two different locations which are a great distance apart. Other factors to consider would be user access patrons and constant access to user profile data. This approach should be extensible so that the app is capable of detecting and responding to complex cyber-attacks

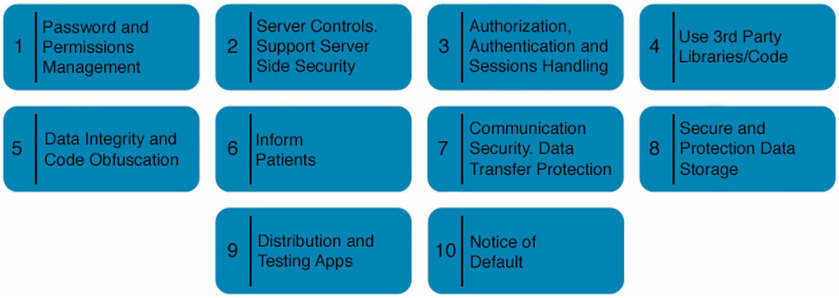


**Fig. 4** Ten points to consider so that an app will be secure

which can encompass apparently unrelated multiple interaction channels and security events.

Above all it is important to bear in mind that passwords should gradually be eliminated as they are a shared secret and they do not really identify the user but are substitutes for means of user authentication through physical or external parameters. A robust identification system must be based on two of three factors: something you know (a password), something you have (a mobile phone by means of a SIM), and something you are (biometrics using fingerprints, iris verification, or capillary exploration of blood vessels under the skin). The online bank of today already uses multiple-factor authentication thanks to a password and a validation by mobile phone. The process involves the input of a second unique code in the app sent to the mobile through an SMS which gives the user consent of access to the services of the app. The motive for this

**Table 10** Ten guidelines to consider so that an app will be secure

| Measure | Description |
|---|---|
| Password and Permissions Management | Mobile devices can be configured to require passwords, personal identification numbers (PIN), access codes or patrons to gain access to the mobile terminal. The use of passwords, access codes, biometric parameters etc. is recommended. It is also necessary to understand the app interacts with all the elements of the device by means of permissions. |
| Server Controls. Support Server Side Security | The back end programmer finds the server side responsible for interacting with databases and verifying user session handling. That is to say, the manipulation of the data which the mobile health app generates. Therefore it must protect itself from all types of injections which can be made on the server so that it isn't vulnerable. |
| Authorization, Authentication and Session Handling | A combination of controls used to verify the identity of a user or other entity, interaction with software, and also that the applications handle passwords management securely and that sessions initiated by the app last as little time as possible. |
| Use 3rd Party Libraries/Code | A group of practices to ensure that the application integrates securely in the code produced by secure source third party libraries without backdoor sources, and also to validate data received from and sent to third party applications before implementing it in the app. |
| Data Integrity and Code Obfuscation | A combination of controls used to prevent reverse engineering of the code, which increases the ability and speed necessary to attack the application. Avoid changes being made or sensitive information being erased, identity being supplanted and existing data being altered when simple exploits are injected (based on text), such as SQL injection to the mobile applications |
| Inform Patients | According to the *Global Privacy Enforcement Network* [42] links must be included to privacy policies before, during and after downloading the app. |
| Communication Security. Data Transfer Protection | The transfer of files is designed to allow users to connect to the internet and share files. File sharing could give access permission to a mobile device to unauthorised users without the knowledge of the user. It is fundamental to ensure that sensitive data is protected when transferring it. |
| Secure and Protection Data Storage | Pay specific attention when giving consent for the gathering and utilisation of user information. It is also recommended to activate remote wipe, as *smartphones* are prone to theft or loss, and to store user data the necessary time for its initial purpose. Encrypt the data making it unusable, illegible and indecipherable to unauthorised persons. |
| Distribution and Testing Apps | A combination of controls to ensure that the app is tested and distributed free of vulnerabilities, providing mechanisms to inform of new security problems if detected, and also that the app has been designed to accept patches to deal with security. |
| Notice of Default | Applications are designed to offer services, but many do not invest in the privacy and security of the user. Any breach of any policy of the app must be reported. |

**Table 11** Low Secure Mobile Development Guidelines

| Level | Low |
|---|---|
| Risk | $0 \leq 3$ |
| Apps | Informative and Educational |
| User Data | No |
| Secure Mobile Development Guidelines | Password and Permissions Management |
| | Server Controls. Support Server Side Security |
| | Communication Security. Data Transfer Protection |
| | Distribution and Testing Apps |

second authentication factor is, for example, the problem that in the case of the fingerprint getting copied, the print on your finger cannot be changed. For this reason two of the three factors must be used.

The alternative way to gain access to the services of the app is OAuth, a different protocol to access your accounts which consists of giving permission to each application. In this way, there is no need to worry about passwords or authentication codes protecting the credentials of your account. The initiated sessions with the server must be discontinued or be destroyed after 15 min, re-requesting use of one of the authenticating methods commented upon beforehand, and the use of secure user communication channels by means of SSL/TLS with 128 bits encoding or VPN.

"Obfuscating" the app code, thereby avoiding reverse engineering, or uploading it to a secure repository would be the final guidelines to follow when developing any application which handles sensitive user information. This data is very tempting for cybercriminals to hack and a future seeing the use of tools such as *Mobile Connect* [43], presented at the Mobile World Congress 2016, putting Europe one step ahead in terms of security, would make it very difficult for them. This service is completely secure and simple, whereby the user validates his identity through the mobile, typing in his PIN after having entered the required personal data into a digital portal, without ever having to share information without authorisation and allowing access to digital services. This default meets the requirements of regulation *eIDAS* [44] of the European Union regarding identification and electronic transactions.

**Table 12** Medium Secure Mobile Development Guidelines

| Level | Medium |
|---|---|
| Risk | $3 \leq 6$ |
| Apps | Calculator, Alarm and Localizing |
| User Data | Yes (Mainly in the App) |
| Secure Mobile Development Guidelines | Password and Permissions Management |
| | Server Controls. Support Server Side Security |
| | Authorization, Authentication and Session Handling |
| | Data Integrity and Code Obfuscation |
| | Communication Security. Data Transfer Protection |
| | Secure and Protection Data Storage |
| | Distribution and Testing Apps |

**Table 13** High Secure Mobile Development Guidelines

| Level | High |
|---|---|
| Risk | $6 \leq 9$ |
| Apps | Healthcare, Treatment, Diagnosis and Monitoring |
| User Data | Yes (App and Server) |
| Secure Mobile Development Guidelines | Password and Permissions Management |
| | Server Controls. Support Server Side Security |
| | Authorization, Authentication and Session Handling |
| | Use 3rd Party Libraries/Code |
| | Data Integrity and Code Obfuscation |
| | Inform Patients |
| | Communication Security. Data Transfer Protection |
| | Secure and Protection Data Storage |
| | Distribution and Testing Apps |
| | Notice of Default |

An innovative security mechanism currently patented by Amazon [45] will be able to tackle possible password theft when paying with a mobile. The user is identified in the app with a *selfie* and to accept the transmission of information they must make a pre-arranged facial gesture into the camera of the smartphone or tablet.

Furthermore, the huge amount of information compiled by mobile health apps which gives rise to Big Data will allow cyber-security companies to predict cyber-attacks. Through the analysis of an infinite number of malicious traces it is possible to pre-empt the cybercriminals and forecast how, when and where the next attack will occur. This is known as the "data lake" and it's possible to explore the lake to check which indicators have been compromised with the object of anticipating future attacks. This is going to contribute to the development of autonomic learning on the part of mobile devices and operative systems so that it is the system itself that learns and raises the alarm when a suspicious element is detected.

**Acknowledgments** This research has been partially supported by the European Commission and the Ministry of Industry, Energy and Tourism under the project AAL-20125036 named "WetakeCare: ICT- based Solution for (Self-) Management of Daily Living".

**Compliance with ethical standards**

**Conflicts of interest** The authors declare that they have no competing interests.

## References


1. KPBC. Internet Trends 2015 – Code Conference. Available from: http://www.kpcb.com/internet-trends (last accessed 22 Feb 2016), 2015.
2. Google. Google Play. Available from: http://play.google.com/store (last accessed 22 Feb 2016), 2016.
3. Apple. iTunes. Available from: http://www.apple.com/itunes (last accessed 22 Feb 2016), 2016.
4. Gartner. Gartner Says Emerging Markets Drove Worldwide Smartphone Sales to 15.5 Percent Growth in Third Quarter of 2015. Available from: http://www.gartner.com/newsroom/id/3169417 (last accessed 23 Feb 2016), 2015.
5. GSMA, GSM Association. The Mobile Economy 2015. 2015. Available from: http://www.gsmamobileeconomy.com/GSMA_Global_Mobile_Economy_Report_2015.pdf (last accessed 24 Feb 2016).
6. Cisco. VNI Mobile Forecast Hightlights, 2015–2020. Available from: http://www.cisco.com/assets/sol/sp/vni/forecast_highlights_mobile/index.html (last accessed 24 Feb 2016), 2015.
7. MGI, McKinsey Global Institute. The Internet of Things: Mapping the Value Beyond the Hype. Available from: http://www.mckinsey.com/insights/business_technology/the_internet_of_things_the_value_of_digitizing_the_physical_world (last accessed 25 Feb 2016), 2015.
8. ITU, International Telecommunication Union. ICT Facts and Figures – The World in 2015. Available from: http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf (last accessed 25 Feb 2016), 2015.
9. World Health Organization. mHealth: New horizons for health through mobile technologies: Based on the Findings of the Second Global Survey on eHealth (Global Observatory for eHealth Series, Volume 3). Available from: http://www.who.int/goe/publications/goe_mhealth_web.pdf?ua=1 (last accessed 25 Feb 2016), 2011.
10. Research2guidance. mHealth App Developer Economics 2015: The Current Status and Trends of the mHealth App Market (5th Annual Study on mHealth App Publishing based on 5000 plus respondents). Available from: http://research2guidance.com/r2g/r2g-mHealth-App-Developer-Economics-2015.pdf (last accessed 25 Feb 2016), 2015.
11. IOT Solutions World Congress. The Future of Healthcare Wearables - Innovation - IOTSWC15. Available from: https://www.youtube.com/watch?v=VR7LPXYYaC0 (last accessed 25 Feb 2016), 2015.
12. Telefónica. Trend Report 2015: The Year of Information leaks. Available from: https://www.elevenpaths.com/wp-content/uploads/2015/12/2015_the_year_of_information_leaks_EN.pdf (last accessed 25 Feb 2016), 2015.
13. Bitglass. Healthcare Breach Report 2016: What a Difference a Year makes. Available from: http://pages.bitglass.com/rs/418-ZAL-815/images/BR_Healthcare_Breach_Report_2016.pdf (last accessed 25 Mar 2016), 2016.
14. Ponemon Institute. 2015 Cost of Data Breach Study: Global Analysis (Benchmark Research Sponsored by IBM). Available from: http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=WH&infotype=SA&htmlfid=SEW03053WWEN&attachment=SEW03053WWEN.PDF (last accessed 27 Feb 2016), 2015.
15. HIMMS Analytics. 2015 Mobile Technology Survey | Executive summary. Available from: http://www.himss.org/ResourceLibrary/genResourceDetailPDF.aspx?ItemNumber=41510 (last accessed 27 Feb 2016), 2015.
16. HIMMS. Marrying the BYOD phenomenon to HIPAA compliance. Available from: http://www.himss.org/ResourceLibrary/GenResourceDetail.aspx?ItemNumber=18909 (last accessed 27 Feb 2016), 2013.
17. Gartner. Gartner Says Worldwide Smartphone Sales Recorded Slowest Growth Rates Since 2013. Available from: http://www.gartner.com/newsroom/id/3115517. (last accessed 29 Feb 2016), 2015.
18. IDC. Smartphone OS Market Share, 2015 Q2. Available from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp (last accessed 29 Feb 2016), 2015.
19. Microsoft. Windows Phone Apps+Games. Available from: http://www.windowsphone.com/es-es/store (last accessed 1 Mar 2016), 2016.
20. BlackBerry. BlackBerry World. Available from: http://www.appworld.blackberry.com/webstore/product/1/ (last accessed 1 Mar 2016), 2016.
21. Nokia. Ovi Store. Available from: http://store.ovi.com/ (last accessed 1 Mar 2016), 2016.
22. IMS Institute for Healthcare Informatics. Patient Adoption of mHealth. Available from: http://www.imshealth.com (last accessed 3 Mar 2016), 2015.
23. World Health Organization. Disease and injury regional estimates, cause-specific mortality: regional estimates for 2012. Available from: http://www.who.int/mediacentre/factsheets/fs310/es/ (last accessed 3 Mar 2016), 2012.
24. World Health Organization. Global Burden of Disease: 2004 Update 2008. Available from: http://www.who.int/healthinfo/global_burden_disease/2004_report_update/en/ (last accessed 3 Mar 2016), 2008.
25. Calvo-González, D., De la Torre-Díez, I., and López-Coronado, M., Análisis y evolución de aplicaciones móviles en el campo de la salud. *I+S Informatica Salud: Sociedad Española Informática Salud* 108:63–70, 2014.
26. IDC. Worldwide Tablet Shipments Expected to Decline −8.0% in 2015 While 2 – in – 1f Devices Pick Up Momentum, Growing 86.5%, According to IDC. Available from: http://www.idc.com/getdoc.jsp?containerId=prUS25867215 (last accessed 3 Mar 2016), 2015.
27. IDC. Smartphone OS Market Share, 2015 Q2. Available from: http://www.idc.com/prodserv/smartphone-os-market-share.jsp (last accessed 8 Mar 2016), 2015.
28. ISO. ISO/IEC 27001:2013 Information technology - Security techniques - Information security management Systems – Requirements. Available from: http://www.iso27001security.com/html/27001.html (last accessed 8 Mar 2016), 2013.
29. OWASP. OWASP Mobile Security Project. Available from: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Home (last accessed 8 Mar 2016), 2015.
30. Martinez-Pérez, B., de la Torre-Díez, I., and Lopez-Coronado, M., Privacy and security in mobile health apps: a review and recommendations. *J. Med. Syst.* 39:181, 2015.
31. HealthIt. Your Mobile Device and Health Information Privacy and Security. Available from: http://www.healthit.gov/providers-professionals/your-mobile-device-and-health-information-privacy-and-security (last accessed 8 Mar 2016), 2014.
32. Senft, D. J., Mobile devices: technology aid - security risk. *Geriatr. Nurs.* 34:149–150, 2013.
33. Chiou, S. Y., Ying, Z., and Liu, J., Improvement of a privacy authentication scheme based on cloud for medical environment. *J. Med. Syst.* 40(4):101, 2016.
34. Chen, Y. L., Liau, R. H., and Chang, L. Y., Applications of multichannel safety authentication protocols in wireless networks. *J. Med. Syst.* 40(1):26, 2016.
35. Guo, P., Wang, J., Ji, S., Geng, S. H., and Xiong, N. N., A lightweight encryption scheme combined with trust management for

privacy-preserving in body sensor networks. *J. Med. Syst.* 39(12): 190, 2015.
36. Cho, H., Lim, J., Kim, H., and Yi, J. H., Anti-debugging scheme for protecting mobile apps on android platform. *J. Med. Syst.* 72(1): 232–246, 2016.
37. MITRE Corporation. The MITRE Corporation. Available from: http://www.mitre.org (last accessed 10 Mar 2016), 2016.
38. PCI Security Standards Council. Official PCI Security Standards Council Site. Available from: https://es.pcisecuritystandards.org/pci_security (last accessed 10 Mar 2016), 2016.
39. DISA. Defense Information Systems Agency. Available from: http://www.disa.mil (last accessed 14 Mar 2016), 2016.
40. FTC. Federal Trade Commision | Protecting America's Consumers. Available from: https://www.ftc.gov (last accessed 14 Mar 2016), 2016.
41. OWASP. OWASP Risk Rating Methodology. Available from: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology (last accessed 14 Mar 2016), 2016.
42. Office of the Privacy Commissioner of Canada. Results of the 2014 Global Privacy Enforcement Network Sweep. Available from: https://www.priv.gc.ca/media/nr-c/2014/bg_140910_e.asp (last accessed 14 Mar 2016), 2014.
43. Mobile Connect. Mobile Connect. Available from: https://mobileconnect.io (last accessed 18 Mar 2016), 2016.
44. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG (last accessed 18 Mar 2016).
45. Amazon. Amazon.com. Available from: http://www.amazon.com (last accessed 18 Mar 2016), 2016.