International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:9, No:9, 2015

# Analysis of Security Vulnerabilities for Mobile Health Applications

Y. Cifuentes, L. Beltrán, L. Ramírez

***Abstract*—**The availability to deploy mobile applications for health care is increasing daily thru different mobile app stores. But within these capabilities the number of hacking attacks has also increased, in particular into medical mobile applications. The security vulnerabilities in medical mobile apps can be triggered by errors in code, incorrect logic, poor design, among other parameters. This is usually used by malicious attackers to steal or modify the users' information. The aim of this research is to analyze the vulnerabilities detected in mobile medical apps according to risk factor standards defined by OWASP in 2014.

***Keywords*—**mHealth apps, OWASP, protocols, security vulnerabilities, risk factors.

## I. Introduction

THE recent evolution on the development of mobile apps for health services have changed the traditional relationship between patients and medical personnel. Mobile health –mHealth apps are a software component that runs in multiple devices to diagnose diseases or other conditions such as: mitigation, treatment, cure or prevention of diseases [1]. mHealth apps are targeted to control and improve the users' medical experience, specifically the healthcare and wellness management. Medical apps normally require users' basic information and physiological measurements according to application performance. The most common physiological measurements are: heart rate, energy expenditure, temperature, blood pressure, among others.

Users' health data is collected through sensors embedded into mobile devices or peripheral devices with bluetooth communication. Collected data is stored in different databases that require a high level of security in data storage to protect information.

mHealth apps have been categorized into two main groups: the first one is general health apps to track health parameters by users, and the second group is the fitness apps to provide physical activity guidelines [2]. According to the IMS, mHealth apps can be categorized according to their functions and user satisfaction as shown in Fig. 1 [2].

No matter the mHealth category, mHealth apps are vulnerable to different malware threats according to simultaneous increasing use of mobile apps. Consequently, the information provided by registered users is vulnerable for security flaws in mHealth apps specifically the data transmission between the service app and server.

Y. Cifuentes, L. Beltran and L. Ramirez are with the Telecommunications Engineering Department, University Military Nueva Granada, Bogotá, Colombia (e-mail: cifuentes.yuli@gmail.com, u1401042@unimilitar.edu.co, Leonardo.ramirez@unimilitar.edu.co).

In 2014, Identity Theft Resource Center - ITRC reported data breaches in different systems. The analysis on medical/healthcare category indicates that 333 numbers of breaches incidents were identified over 8.277.991 of firm's records [3]. Food and Drug Administration -FDA and the European Union –EU have been working to create policies and standards for controlling the use and develop of medical apps in mobile market since 2013.

In 2015, IBM reported that over 11.6 million mobile devices have been affected by malicious attacks [4]. The most frequently attacks in mobile apps are: Code Injection and Cross-Site Scripting (XSS). Both mobile apps attacks are inherited security vulnerabilities from web applications.

In order to reduce the threats impact, mobile security risks have been classified by OWASP Mobile Security Project development. Additionally the main purpose is to provide resources to maintain security on mobile apps. The OWASP mobile security project objective is not only based on the end user device security, but also on the server-side infrastructure [5].

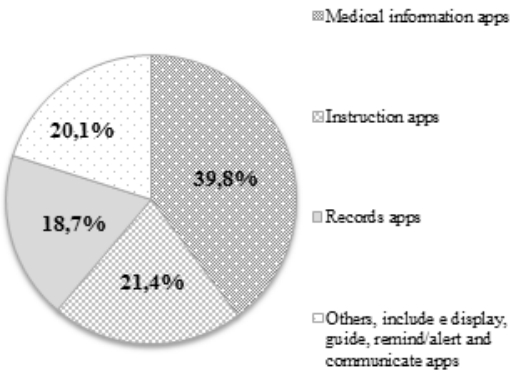


Fig. 1 Medical application categorization based on their functions

In this paper, an analysis of security vulnerabilities for mHealth apps is introduced, according to next categories: medical information, education and awareness, remote monitoring, diagnostic support, treatment support and communication and training for healthcare workers.

This paper is structured in five sections: Security standards on mobile apps are presented in Section II. mHealth apps vulnerabilities assessment is presented in Section III. The result analyses are discussed in Section IV. Finally, conclusion is presented in Section V.

## II. Related work

The World Health Organization –WHO defines mHealth or mobile health as: *"Medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants –PDAs, and other wireless devices"* [6] and the Food and Drug Administration –FDA defines mHealth as: *"An available software for smart phones, tablet computers, among others that can be executed on a mobile platform (Android, iOS, Windows) with or without wireless connectivity"* [1].

mHealth apps are categorized according to their medical functionality in:

- **Medical information** shows information about any medical area by displaying them on text, photo, video, among others formats.
- **Education and awareness** present disease actual information to facilitate patient active education.
- **Remote monitoring** captures and stores users' physiological activity to be monitored remotely.
- **Diagnostic support** presents a disease diagnosis according to a users' medical alteration.
- **Treatment support** suggests guidelines for use a treatment and ensures the user is following it properly.
- **Communication and training for healthcare workers** manages information such as medical centers location, appointments management, among others.

### *A. Security in Mobile Devices*

Despite of several organizations are working in data security, there aren't a global security standard for mobile devices. However, many companies, universities and manufacturers have proposed some guidelines. The majority of recommendations are created to avoid or detect mobile devices actual threats in different OSI layers as physical layer, transport layer and application layer.

Some mobile devices risks were identified by the National Institute of Standards and Technology –NIST. The first one is device theft or loss. The second one is access of malware and virus to the device through peripheral interfaces such as: wireless port, bluetooth, microSD port and miniUSB port, among others. The last one is when an attacker can access to user's information through spam. [7] However, each threat can be avoided as next: in case of theft or loss, the user should immediately contact the service provider to report the event; the service provider must block the device and service. [8] In case of virus and malware, NIST leads three recommendations: the first one is to change passwords in a regular period of time; the second is to avoid the valuable information storage on devices with a safe copy. Finally, if the interfaces are not in use, they should be disabled. [8]

User must delete received messages from unknown sources to avoid spam risks. The last recommendation is based on software installation for data protection such as: firewall, antivirus, intrusion detectors, anti-spam, among others. Additionally, network of organizations secure connections and maintain security policies for information management [8].

### *B. Mobile Application Security Requirements*

There are many advantages of *Android* open-source operating systems to develop applications. Nevertheless the security on developed apps is vulnerable due to an intruder can get root access by privileges escalation [9].

CSM (Competitive Supervision Mechanism) is a technique takes into account three aspects: desire for application security, responsibility for the security app and the last one is the concern of the user. With the first topic can be installed in the two cell types of applications that provide a security one that lifts the OS to root mode and the other application that provides security in the background. The second field is to look for evidence that the installed application is not a single bad behavior. The last kind rests with the user who should be careful in the permissions that headquarters because some applications request bad root permissions [10].

### *C. Communication Protocol for Mobile Apps*

The Wireless Application Protocol –WAP forum was an enterprise group to provide specifications and services for wireless technologies. WAP are a global suite of protocols to enable the transmission of information and services in most of wireless networks. The WAP standards leverage mobile network technologies and internet technologies such as IP, HTTP and other protocols. In mobile apps three protocols are based on internet technologies: Wireless Profiled HTTP –WP-HTTP, Transport Layer Security -TLS and Wireless Profiled TCP -WP-TCP. On the other hand, in WAP protocol stack there are four legacy protocol layers: Wireless Session Protocol -WSP, Wireless Transaction Protocol -WTP, Wireless Transport Layer Security -WTLS, Wireless Datagram Protocol -WDP. Also WAP was presented in WAP forum as standard for providing the way to access into services as e-mail by mobile phone, news headlines, among other interactive data services [11].

WP-TCP is a connection-oriented protocol to optimize wireless communication environments and data transmission between two mobile devices on an IP network. WP-TCP is compatible with TCP and it is one of the most reliable transport protocols in wireless networks [12].

TLS protocol guarantees data confidentiality and integrity that is transmitted between two applications over the network.

Private and reliable connection between applications, identity authentication by at least one of both applications, safe negotiation of the secret, and negotiation between applications is reliable are the main TLS features. By other side, in order of their priority, TLS goals are: cryptographic security, interoperability and extensibility [13].

WP-HTTP is a modification of the HTTP protocol for wireless communications. WP-HTTP supports the payload compression of response messages for efficient times in the transmission. Additionally, WP-HTTP provides solution for wireless problems using a safety tunnel as end to end security [14].

WSP is a session layer protocol for remote operations between a client and the mobile apps server. WSP is structured by three stages: 1. Start of session, 2. session and transfer

information (back and forth), 3. End of session. WSP protocol does not use IP networks [15].

WTP is used for mobile phones and it does not work with IP networks. WTP works similar to TCP protocol with the exception that reduces the quantity of information necessary on each transaction. WTP protocol has four common features: 1. Three types of service transaction: unidirectional and unsafe request, unidirectional and safe request, and bi-directional and safe request. 2. Optional user-to-user security. 3. Concatenation of PDUs and delay of acknowledgements of receipt to reduce the number of sent messages 4. Asynchronous transfers [16].

Transport layer protocol WDP exchanges datagrams and provides a service to upper layers protocols [17].

*D.Security Vulnerabilities on Mobile Apps*

Security vulnerabilities are evident in most popular mobile applications. Unauthorized user access is one of the most serious security vulnerabilities; it leads to an expected privilege escalation and to data theft or loss.

Different researches have assessed some vulnerabilities that are present on mobile apps:

1) *Cross-site scripting –XSS* is a common vulnerability present in web applications. An android user interacts with a Web app through Webview components as graphical user interface. Webview is vulnerable to attacks with malicious code. [18] Cross-site scripting has affected websites mobile apps such as: Google, Yahoo and Facebook, among others. The attacker introduces a malicious app that was specifically designed for the web application; users run malicious scripts through malicious application, so that the attacker can gain full access to the sensitive data and information such as cookies, contacts, and location, among others [19].
2) *SQL Injection:* An intruder inserts new SQL keywords into a SQL instruction. Consequently, the instruction logic changes. Intruder can access to SQL servers and to include malicious SQL code under user privileges. Attacker can access to the database to create, alter, update, read or delete data [20].
3) *Hijacking and Spoofing Intents:* This vulnerability occurs when a malicious service intercepts intent meant for a legitimate service. Consequently, in order to steal data supplied by the user (i.e., phishing), it leaves a connection between the application and the malicious service. The Hijacking is not apparent to the user because no user interface is involved. The less known vulnerability is Intent Spoofing, it is characterized by launch an Intent spoofing attack by sending an Intent to an exported component that is not expecting Intents from that application [21].
4) *Sticky broadcast tampering:* this vulnerability is characterized by persistent intents of communication from legitimate mobile apps that can be accessed and removed by malicious mobile apps. Initially, sticky broadcast was a tool used by developers to communicate with others apps, due to that; an app cannot communicate directly with other app. Sticky broadcast vulnerability generally works through of the OS. First, it sends an intent, that is message used by an app, and then this message is received for other app, through OS. Normally the process would end, but Sticky broadcast keeps the broadcasts to notify it to other apps thinking that information may be needed for other apps [22]. It is leaving an opportunity for unauthorized access from malicious applications.
5) *Freak - SSL/TLS vulnerability*: is a weakness in some implementations of SSL/TLS. First, it was though that this vulnerability affected only Apple (iOS) and Android, but is seen that also affected other mobiles platforms. Freak - SSL/TLS vulnerability allows an intruder to intercept HTTPS connections between vulnerable clients and servers, forcing them to use weakened encryption. Consequently, attacker steals or manipulates sensitive data [23].
6) *Heartbleed Bug* is the most popular OpenSSL library vulnerability that enables to steal protected information that is encrypted under SSL/TLS protocols. The implementation of TLS heartbeat request/response module is an actual Open SSL fault. Consequently, an attacker can easily access to an OpenSSL system where he can read encrypted and stored information; safe keys, passwords and users' information can be stolen or changed [24].
7) *Insecure storage:* Stored data without digital encryption and that is accessible from a mobile application can be converted in an objective for the attackers. Insecure storage relates to the improper protection of sensitive data on devices, or in the cloud, when the data are not encrypted [25].
8) *SSL-stripping* is a type of vulnerability that enables to an intruder performs an attack Man in the Middle. An MITM attack is a form of active attack in which the attacker intercepts and selectively modifies intercepted data in order to impersonate a legitimate user involved in the client and server communication [26]. MITM uses a malicious app that is placed on the communication line between the user and the service consumed. It pretends to be a required service for the user, filtering the entire information passing, through of communication line, for manipulate or steal it.

*E. Risks Factors Detected by OWASP Mobile Security Project*

OWASP- *Open Web Application Security Project* defined some possible risk factors that affect mobile apps security. The project has worked on risks classification to provide controls that minimize their impact or likelihood of exploitation. [5] List of top 10 risks is presented in Fig. 2.

The risks of injection are in top of OWASP list because this type of risk can be exploited by errors in the programming of consultations. Secondly in the list are the vulnerabilities of session management, these vulnerabilities are design errors. The rest of risks factor of the ranking have flaws in the design of applications also as mistakes of those who administer them.

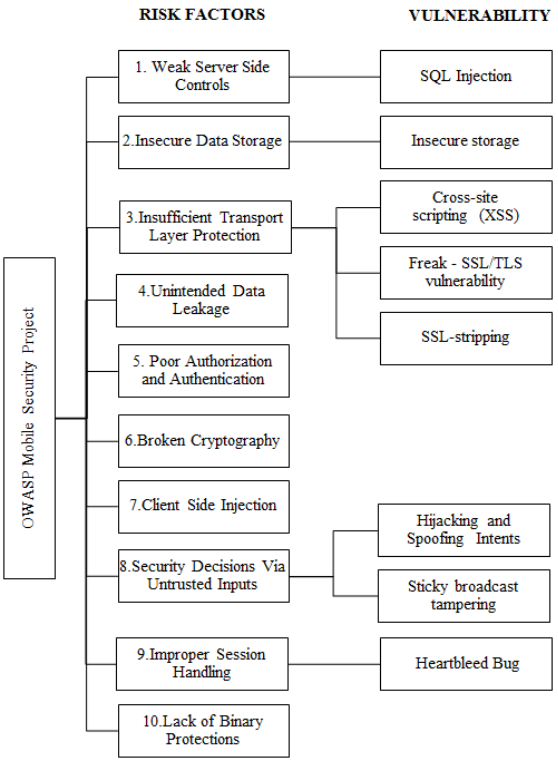


Fig. 2 Risk factors established by OWASP and vulnerabilities detected in mobiles apps

## III. Evaluation of Vulnerabilities in mHealth Apps

In this section the mHealth apps vulnerabilities criteria assessment are submitted.

### *A. Selection of mHealth Applications*

mHealth apps were categorized in six groups according to their functionality. Ten applications of each group were selected and they were downloaded from Google play store. The research assesses variables such as: level of risk, type of vulnerability, and type of risk factor according to OWASP 2014

### *B. Description of the Procedure*

The analysis was based on mobile apps vulnerabilities that were documented in previous works, databases and technical reports. Analysis includes the most common vulnerabilities for Android platform. We employed an IBM company software for the vulnerabilities analysis. Software decompresses and analyzes the APK code on each one of the mobile apps.

Applications risk level was classified in high, medium and low risks based on OWASP top 10 list.

## IV. Results and Discussion

The aim of the "Results and discussion" section is to present the key results and the lessons learned during this mHealth apps vulnerabilities research.

### *A. Overall Result Analysis*

60 mHealth apps were analyzed and 157 vulnerabilities were found. According to the results presented in Fig. 3, remote monitoring apps are the most vulnerable category with 32.12% of found vulnerabilities. Also, communication and training for healthcare workers apps is the less vulnerable category with 8,92% of vulnerabilities. Vulnerability percentage in other categories is as next: diagnostic support apps have 18.47% of vulnerabilities, medical information apps have 17.20%, treatment support apps have 12.10% and finally, education and awareness apps have 10.19%.

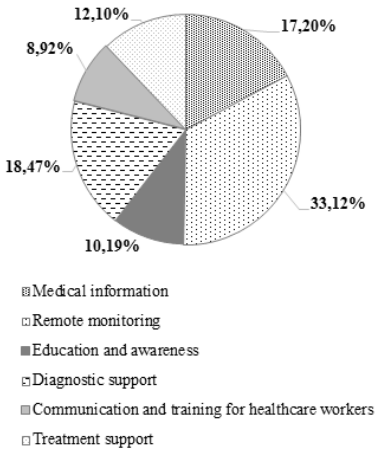


Fig. 3 Results of vulnerabilities detected in general classification mHealth apps

According to the risk level, all apps categories present low-risk and medium-risk vulnerabilities, as depicted in Fig. 4. High-risk vulnerabilities are only presented in four categories. Medical information apps and treatment support apps have not submitted high risk vulnerabilities. Remote monitoring apps is the category with most high-risk vulnerabilities; it is percentage is 31%. Subsequently, Education and Awareness apps have 19% of high-risk vulnerabilities, Communication and training apps have 7% and Diagnostic support apps have 3%. The high-risk vulnerabilities that were detected more frequently in this research are: XSS -Cross-site scripting through MiTM -Man-in-the-Middle and scripts with malicious content.

Medium-risk vulnerabilities are present in all categories. Diagnostic support apps is the category with most medium-level vulnerabilities. The research detected 62% of these vulnerabilities on diagnostic support apps. Pseudo-random number generators are the main vulnerability on diagnostic support apps. Consequently, they contribute to a security bug in the OpenSSL cryptography.

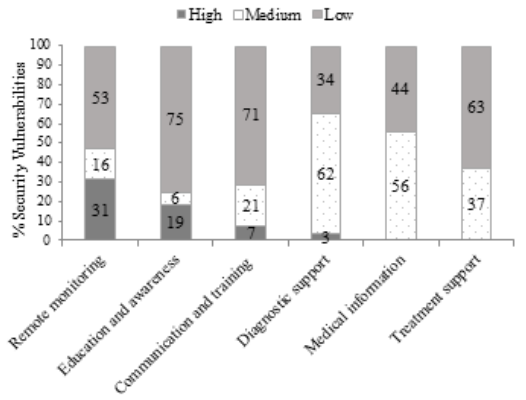


Fig. 4 Percentage of security vulnerabilities according to mHealth apps categories

Backup enabled is the low-risk vulnerability that arises most frequently. An attacker can undermine the app integrity and confidentiality. Nonetheless, results indicate that all mHealth apps have vulnerabilities that compromise user information confidentiality, integrity and availability.

### *B. Applications Analysis Using OWASP Model*

Assessment using the OWASP model determined that 64% of found vulnerabilities correspond to "security decisions via untrusted inputs". Due to security in mobile applications protocols are weak, thus leading the application to enable traffic input from any source. By this, an attack can easier be present on the application.

14% of vulnerabilities correspond to Broken Cryptography, 13% corresponds to Client-Side Injection, 6% corresponds to "Insufficient Transport Layer Protection", and finally 3% corresponds to "Insecure Data Storage".

One of the most severe consequences of Client Side Injection for mHealth apps is that an intruder can loads simple text-based attacks that are used to spoof identity and to tamper existing data in the apps. Additionally, it enables an attacker to include a malicious application to control the browser URL parameter included, for example Web View.

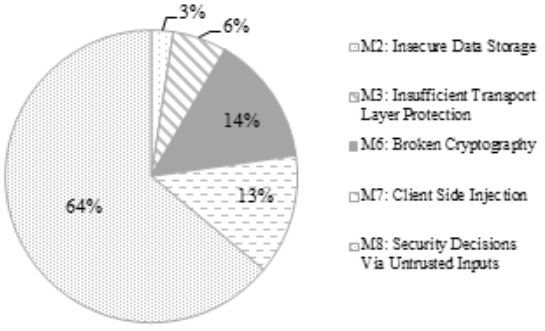


Fig. 5 Percent of risk factors in mhealth Apps using OWASP model

The applications frequently are not protect against network traffic; they may use SSL/TLS during authentication, but not elsewhere, exposing the data and session IDs to interception. This risk allows a MiTM attacker can perform an XSS attack to undermine the integrity and confidentiality of the application destination.

Finally, the insecure data storage can result in mobile apps data loss, but this vulnerability prevalence is relatively low in mHealth apps, as it only represents 3 % of risk of attack. The vulnerability is caused by creation of files with permissive rights. Results from risk factors in mhealth Apps using OWASP model are presented in Fig. 5.

### *C. Discussion*

This research proposes an analysis of security vulnerabilities using OWASP criteria for determining the risk of attack in mHealth apps for Android-based mobile devices. The security assessment only considers the APK code analysis. An APK file is a ZIP file with the executable. If this code has a security flaw may negatively affect the confidentiality, integrity and availability of the recorded information in the mHealth apps. Also the market has available different types of scanners that perform tests of vulnerability with attacks known, and has a high effectiveness in the attack detection; this can be used for this type of analysis.

## V. Conclusion

With the analysis, it was possible to detect safety flaws in the Apk code, which can trigger threats over user's data and over the information recorded into the server. This kind of mistakes may limit the use of mobile apps in telemedicine; because it opens a gap for the theft or data modification.

It iss important that developers and users of mobile mHealth apps try to raise awareness about safety standards that must be deployed at the time of develop or update this kind of apps

With the risk assessment analyzed with OWASP, it is possible to conclude that the vulnerabilities are mainly triggered by accepting data from different sources. It is important to establish mechanisms with limitation on information sources that are employed by applications, which would improve confidence at the end user side.

For future work, it is necessary to analyze vulnerabilities over communication channels, between mobile applications and database servers, in order to establish threats even with encrypted techniques.

## Acknowledgment

This work was possible in part with the financial support of the Military University Nueva Granada in the project ING-1772 with GISSIC and TIGUM research groups.

## References

[1] Food and Drug Administration, et al. Mobile Medical Applications: Guidance for Industry and Food and Drug Administration Staff. USA: Food and Drug Administration, Tech. Rep, 2013.

[2] M, Aitken; C, Gauntlett "Patient Apps for Improved Healthcare from Novelty to Mainstream". IMS Institute for Healthcare Informatics Tech. Rep, 2013, pp. 1-65.

[3] Identity Theft Resource Center®, “ITRC Data Breaches Reports 2014”, Tech. Rep, 2014. Retrieved from website: http://www.idtheftcenter.org/ITRC-SurveysStudies/2014databreaches.html.
[4] Arxan IBM, “Arxan Application Protection with IBM Security Trusteer” Tech. Rep, 2015. Retrieved from website: https://www.arxan.com/wp-content/uploads/assets1/pdf/Arxan_Application_Protection_with_IBM_Trusteer_-_Solution_Brief.pdf.
[5] OWASP, Mobile Security Project Top 10 Mobile Risks. (Online), 2015 Retrieved from website: https://www.owasp.org/index.php/OWASP_Mobile_Security_Project.
[6] KAY, Misha; SANTOS, Jonathan; TAKANE, Marina. "mHealth: New Horizons for Health through Mobile Technologies." World Health Organization, 2011, pp. 66-71.
[7] B, Hasan, B.; Dmitriyev, V.; Gomez, J.M.; Kurzhofer, J., "A Framework Along with Guidelines for Designing Secure Mobile Enterprise Applications," *Security Technology (ICCST), 2014 International Carnahan Conference on* , vol., no., pp.1,6, 13-16 Oct. 2014.
[8] Copeland, W.; Chia-Chu Chiang, "Securing Enterprise Mobile Information," *Computer, Consumer and Control (IS3C), 2012 International Symposium on*, vol., no., pp.80,83, 4-6 June 2012.
[9] Nicholas Penning, Michael Hoffman, Jason Nikolai, Yong Wang. “Mobile Malware Security Challeges and Cloud-Based Detection”, 2014.
[10] Yonglin Sun, Yongjun Wang, Xiaobin Wang. “Mobile Security Apps: Loyal Gaurds or Hypercritical Thieves?” 2014.
[11] Open Mobile Alliance, “‘Wireless Application Protocol WAP 2.0” Tech. Rep., 2002.
[12] H. Rutagemwa, “Performance Modeling, Design and Analysis of Transport Mechanisms in Integrated heterogeneous Wireless Networks”, Diss. University of Waterloo, 2007.
[13] DIERKS, T.; ALLEN, C. The TLS Protocol (rfc 2246). *Internet Engineering Task Force (IETF)*, 1999.
[14] R. J, Boncella. "Wireless Security: An Overview." Communications of the Association for Information Systems, 2003, vol. 9, no 1, pp. 15.
[15] W, WSP, “Wireless Application Protocol”, *Wireless Session Protocol Specification*, 1999, vol. 30, pp 84.
[16] A. S, Godbole; A.S.G.A. Kahate, Web Technologies: Tcp/ip to Internet Application Architectures. Tata McGraw-Hill Education, 2002.
[17] V, Kumar; S, Parimi ; D.P, Agrawal,., "WAP: Present and Future," Pervasive Computing, IEEE , vol.2, no.1, pp.79,83, Jan-Mar 2003 doi: 10.1109/MPRV.2003.1186729.
[18] A. B. Bhavani. Cross-Site Scripting Attacks on Android Webview. arXiv Preprint arXiv:1304.7451, 2013.
[19] T, Luo; H, Hao; W, Du; Y, Wang; Yin, H. “Attacks on WebView in the Android System”. *In Proceedings of the 27th Annual Computer Security Applications Conference*. ACM, 2011. pp. 343-352.
[20] K.Wei; M, Muthuprasanna; S, Kothari. “Preventing SQL Injection Attacks in Stored Procedures”. *In Software Engineering Conference, 2006.* Australian. IEEE, 2006. pp. 8.
[21] E. Chin; A.P. Felt; K, Greenwood; D. Wagner. “Analyzing Inter-Application Communication in Android”. *In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services.* ACM, 2011. pp. 239-252.
[22] H. Dwivedi. *Mobile Application Security*. Tata McGraw-Hill Education, *2010*.
[23] United States Computer Emergency Readiness Team “FREAK SSL/TLS Vulnerability”, (online). March 2015. Avalaible: https://www.us-cert.gov/ncas/current-activity/2015/03/06/FREAK-SSLTLS-Vulnerability.
[24] S. Gujrathi. “Heartbleed Bug: Anopenssl Heartbeat Vulnerability”. *International Journal of Computer Science and Engine ter Science and Engineering*, 2014, vol. 2, no 5, pp. 61-64.
[25] A. K. Jain; D. Shanbhag. “Addressing Security and Privacy Risks in Mobile Applications”. *IT Professional*, 2012. no 5. pp. 28-33
[26] M. L. Das; N. Samdaria. “On the Security of SSL/TLS-Enabled Applications”. *Applied Computing and Informatics*, 2014, vol. 10, no 1, pp. 68-81.