International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-6, March 2020

Static Pentesting of Android Application

Pooja P, Puneeth K M

Abstract: ***Mobile application use has become increasingly common, particularly android is the most well known working framework being utilized. The portable application is getting developed in all areas like social insurance, individual consideration, Gaming, Business, and so forth, and these applications need the client to give their own information, wherein the client's information are stored on the Mobile gadget, or it is transmitted through the Network as API's. If there are any security, misconfigurations present may lead the hacker to attack the mobile device or the API's which helps to gain access on the user's data. Hence, the mobile devices are becoming more prone to security threats. Consequently, the portable applications must ensure to be secure before they are conveyed in the market, and it is the duty of the application proprietor to perform Pentesting on the application to provide security to the client's information. Pentesting of any android or iOS applications can be performed in two ways, static and dynamic Pentesting. Static Pentesting helps in understating the security threats at the local storage and on the manifest file by reverse engineering the application's APK or IPA file, whereas dynamic Pentesting helps in finding security threats in the real-time exchange of data through the network. In this paper, the Static Pentesting methodology that can be used for android application Pentesting is described with the use of open-source Pentesting tools and a sample android application. The static Pentesting of any application encourages the engineer to comprehend the security misconfigurations, which ought to be dodged at the development stage of the application itself. This guarantees the security of the client's information at the device level.***

Keywords: API's, Dynamic Pentesting, Pentesting, Reverse Engineering, Static Pentesting.

I. INTRODUCTION

Mobile applications make the world little, individuals' lives better and simpler. The Mobile applications are utilized in an assortment of fields like shopping, banking, Education, Entertainment, Food conveyance, voyaging, correspondence, and so forth. In present, android is the most utilized working framework in mobile platforms. The android has approximately 82% of the market share [6].

Manuscript received on February 10, 2020.
Revised Manuscript received on February 20, 2020.
Manuscript published on March 30, 2020.
* Correspondence Author
Pooja P, Mtech, Networking and Internt Engineering, Jss Science and technological University, Mysuru,,Karnataka, PIN-570008, India. E-mail: poojaprasad575@gmail.com
Puneeth K M, Mtech,, Electroncs and Communication Engineering, Jss Science and technological University, Mysuru, Karnataka, PIN-570008, India. E-mail: kmpuneeth@sjce.ac.in

The Mobile application security must have the extreme need as these applications transmit and store delicate information of the client [1]. However, security is an overall issue, on which every person and the associations are worried about. Consequently, the association must perform security tests before the applications are sent to the market.

The security tests of mobile applications quantified by performing static and dynamic Pentesting [4]. Pentesting is a procedure that moral programmers or pentester performs to discover security dangers or vulnerabilities in the applications by utilizing open source tools. For the most part, the Pentesting of the application can be characterized as grey box hacking. Through Pentesting, all the security misconfigurations present in the application can be identified. Any security misconfigurations present in the application must be addressed in the development stage itself, because any vulnerabilities or security misconfigurations identified later after the deployment of the application to the market might prompt incredible misfortune to the association in both the financial riches and furthermore in the trust of their clients. So the found vulnerabilities at the Pentesting stage should be patched up before the deployment of the application to the market to guarantee the security of the client's information [1] [5].

The Security Pentesting is categorized into two kinds static and dynamic Pentesting. The static Pentesting helps in the investigation of local storage created by the application after the installation of the application on the device, and it helps in the examination of the files that is extracted by Reverse Engineering of the APK. The dynamic Pentesting helps in the investigation of constant information move out of the application through the network as APIs.

The security vulnerabilities of android applications or some other applications like iOS, web application, Thick Client, Network applications can tested for security vulnerabilities with the reference of OWASP (Open Web Application Security Project) [15]. OWASP is an online community that provides articles, tools, documentation needed for Pentesting [15]. The OWASP Mobile Security Project is an asset for the security groups and application engineers, to pentest the versatile application or develop applications in a secure manner. The OWASP arranges security hazards as Top 10 vulnerabilities and gives mitigation procedure to pentesters or developers in order to reduce the effect of the security vulnerabilities on the application [9]. There are several open-source tools available to perform the Pentesting, Example Burp Suite, Nmap, SSL scan, dex2jar, JD GUI, Winscp, etc. Tools like Burp Suite, Nmap, SSL scan, and all these tools help in the dynamic Pentesting.

Burp suite acts as a proxy between the mobile application and the server that the application connects to; by this, all the requests, which need to be sent to the server, will be captured by the Burp suite. By this, Pen tester analyses, the requests captured in Burp Suite modifies it, forwards the request to the server, and

checks the response from the server for the security vulnerability present in the application server side [10]. Tools like Nmap and SSL scans help in understanding the encryption techniques used and the TLS (Transport layer security version) used in the application. Apktool, dex2jar, JDGUI, Winscp, all these tools are used in static Pentesting. These tools help to do reverse engineer the apk file, with the reverse engineered apk file one can find the vulnerabilities in the coding part and the local storage of the android application.

Hence, the Pentesting of the android applications helps the organization to maintain its reputation in the society. Furthermore, the Pentesting helps the developer or the association to maintain a strategic distance from the security misconfigurations of the application, if any serious security defenselessness found while playing out the Dynamic Pentesting or any security misconfiguration found in the static Pentesting can be avoided based on the severity of the security issue found. This guarantees the client's information to be secure and helps to keep away from the information ruptures or data breaches.

II. LITERATURE SURVEY

Mobile devices act as storage for mobile applications, including sensitive data of the user like photos, videos, authentication credentials, personal information, professional information and many more [3]. In mobile devices, android plays a major role, as it is a versatile working framework [2]. Hence android application security turns into a critical issue. Android applications has incredible capability of posing privacy and security risks of the android device users, because the applications installed on the android device transmit client information to the outside servers, along with that android application developers may likewise utilize third party libraries or open-source libraries that has a some security issues, which may prompt significant security threats [4]. Therefore, application developers and the organization must take the responsibility of mitigating the security vulnerabilities present in their application before the application goes to the market for the use. If the security vulnerabilities are not addressed before, the deployment of the application in the market, and later the vulnerabilities found or any data breaches occurs leads to a great impact on the reputation and the economy of the organization. Hence ensuring the security of the application at both the device level (Static) and the communication level (Dynamic) must be addressed by the organization [1]. The security testing of the application is a procedure, which is known as Pentesting, the Pentesting assembles all the information identified with the application and spotlights on the few entry points where the information rupture can happen [5].

The Pentesting methodology can be defined as shown in Fig 1. The First step is Information gathering, Before Pentesting the pentester or the security engineer must comprehend the work process and the framework of the application, for the most part, this stage can be characterized as black box testing, which helps in totally understanding the application [5]. The second step is static Pentesting or device level analysis, where the pentester finds the security misconfiguration done by the developers at the source code level. And checks for the permissions provided to the application on the device, to be specific a detailed analysis of the storage created by the application on the device will be done by the pentesters to check the security misconfigurations. The third step is dynamic Pentesting or communication level security testing. The dynamic Pentesting analyzes the security vulnerability at the network level by setting the proxy between the application and the server. The proxy set up captures all the requests sent by the application, with the help of this the pentester does an analysis on the requests, modifies the request captured in the proxy tool, and forwards it to the server. In response to the request sent by the proxy tool, the server responds to the modified request. So if any security misconfiguration did in either server end or client end can be found using this dynamic Pentesting. The fourth step is to analyze the risk level based on the misconfiguration and Reporting the security misconfiguration or vulnerabilities found after pentesting to the application development team to patch the found misconfiguration [5]. This ensures the security of the application and the user's data.

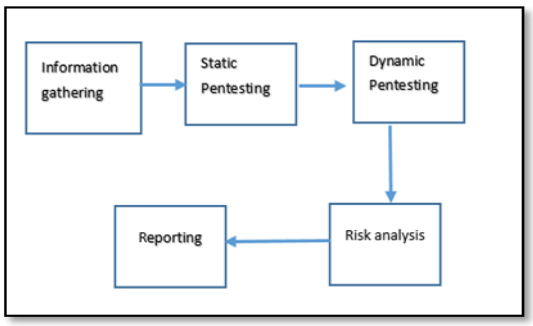

Fig. 1. Pentesting Methodology

According to the OWASP(Open web application security Project) Mobile security, which has classified the mobile application Vulnerabilities to TOP 10 vulnerability according to their risk severity. Wherein insecure data storage (Static Pentesting) is the leading security risk in Top 10 security vulnerabilities because sensitive data can be revealed if the security is not properly provided [3]. Hence, static Pentesting is a significant job in the security of the android mobile application.

III. ANDROID APPLICATION PENTESTING METHODOLOGY

New technology always introduces new security risks, and android applications are of no exceptions. The modern mobile Operating systems are more secure when compared to the desktop operating system, but problems can still occur when the security is not implemented on the application at the development stage itself.

The Pentesting or security testing of any of the applications like web, android, iOS, thick customer, arrange, and so forth, is performed with the reference of OWASP. OWASP arranges versatile application security risks to Top 10 vulnerabilities as shown in Fig 2.

According to the Top 10 vulnerabilities, the application security must be tested for both the client end and the server end. The Top 10 vulnerabilities includes Weak server-side controls and Client-side injection which checks for the injection attacks in the server and the client ends, Insufficient TLS (Transport Layer Security) which checks for the TLS version and encryption methods used in the transmission of the data from client to server. Poor authorization checks whether the server is validating the authorization token of the client, Improper session handling checks for the session token expiration and validation of the session token at the server-side [11]. Apart from the server and client-side testing, the OWASP categorizes some other vulnerability related to the static part of the application (Static Pentesting) like Insecure data storage, which checks local storage created by the application in the device. Broken cryptography checks for the weak or unencrypted user's data stored in the device by the application and Binary protection which checks the security misconfiguration at the source code level [11].

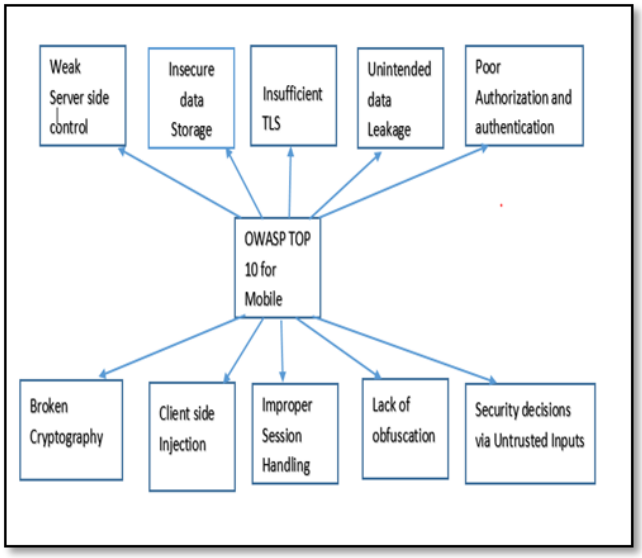

Fig. 2. OWASP TOP 10 Mobile security Vulnerabilities

A. Static Pentesting Methodology Abbreviations

The static Pentesting helps in finding the security misconfiguration in the source code and on the storage created by the application in the device. The static Pentesting methodology is as shown in Fig 3. The initial phase in static Pentesting is to get the apk file of the application; once the apk file is ready, the further pentesting procedure can be executed. The subsequent step is to Reverse Engineer the apk file, to Reverse Engineer the apk file there a few open-source tools available, one can utilize the open source tools to Reverse Engineer the apk file to break down and discover the misconfigurations in the build or development of an apk. The build of an apk refers to different files that the apk file has like a manifest file, META-INF file, assets, classes.dex file, lib, and res directory. These files are extracted by Reverse engineering the apk file. The third step in the static analysis is to get the local storage created by the application in the device. To get the local storage of an application a rooted android phone is required because the application creates the local storage under the root directory and the normal user cannot access the root directory of the android phone unless the phone is rooted. So by using the rooted device the local storage of the application can be tested for the security misconfigurations. The application creates several folders in the device like databases, cache, shared preferences, etc., the next step is to report all the security misconfigurations found to the developers with the severity of the security misconfigurations, the final step is to mitigate all the found misconfigurations.

B. Pentesting by Reverse Engineering of APK

Reverse Engineering is a process of extracting all the files which are combined to form an apk file. When the apk is file is Reverse Engineered, several files like manifest files, assets, lib, res file, etc., can be found. In this section, the process and tools used for Reverse Engineering an apk file are explained along with process of how to analyze and find the security misconfiguration in the apk file.

The first step of reverse engineering an apk file is to use the apktool [12]. To understand the static Pentesting, a demo application named as "demo.apk" is used in this paper. The demo.apk file is reverse engineered as shown in Fig 4 and Fig 5. The files from apk file can be extracted with the help of apktool, to reverse engineer any apk file, two files must be downloaded one is apktool.jar file and the other is apktool.bat file [12]. These two files need to be in the same folder and in the path where these two files are present, the apk file (demo.apk) should be placed and the command in the Fig 4 must be executed. By doing this all the files like manifest, lib, assets, res folders used to build the APK file can be extracted as shown in Fig 5.

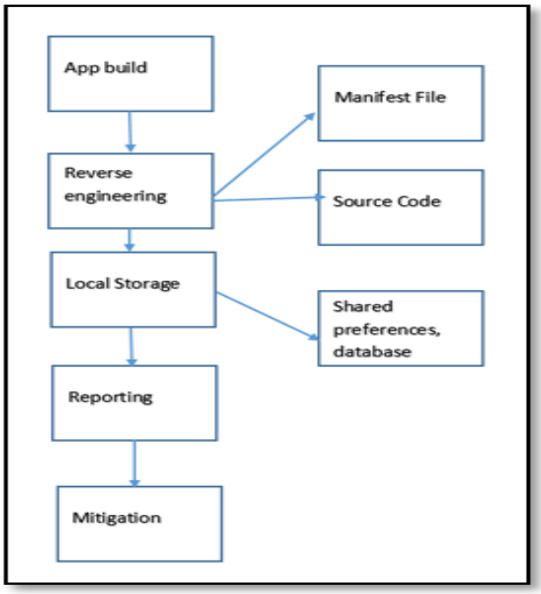

Fig. 3. Static Pentesting Methodology

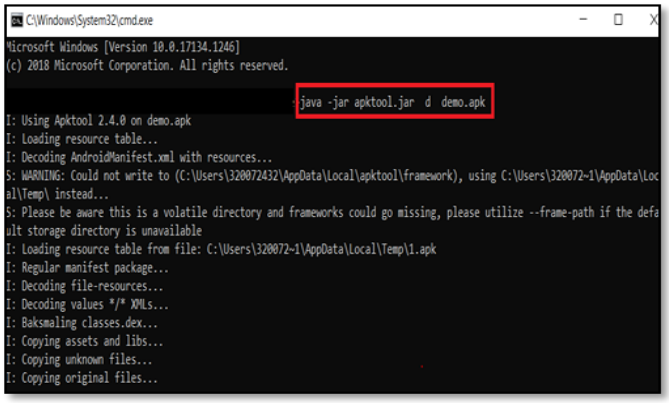

Fig. 4. Apk tool for Reverse Engineering of apk file

To find the source code of the application classes.dex file is necessary, from Fig 5 it is clear that the classes.dex file can be retrieved by Reverse engineering. The next step is to convert the classes.dex file to classes.jar file so that a tool called JDGUI can open the jar file. So to convert the dex file to a jar file a tool called dex2jar can be used, and by executing the command "d2j-dex2jar.bat classes.dex" in d2j folder, the dex file can be converted to jar file, once this command is executed the classes.jar file can be found in the same folder as shown in Fig 6 [13]. The JDGUI tool can be used to view this classes.jar file as shown in Fig 7 [14]. From this, all the coding part of the application, which is used to build the application, will be found, and each file of the source code can be checked with JDGUI tool, and also any flaws in coding like unencrypted or unreferenced password in the coding can be identified by parsing through the entire source code of the application.

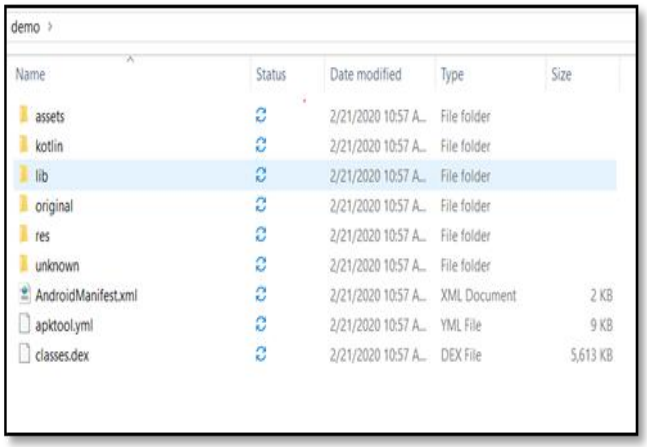

Fig. 5. Result of Reverse Engineered apk file

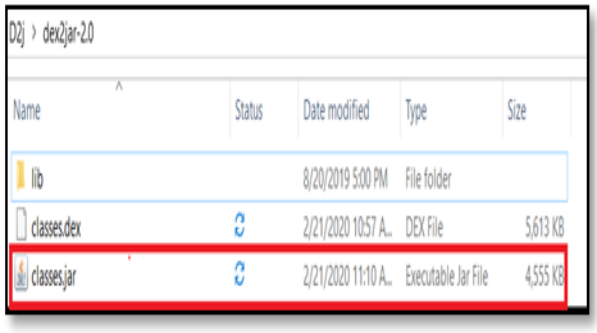

Fig. 6. Converting to dex to jar by using dex2jar tool

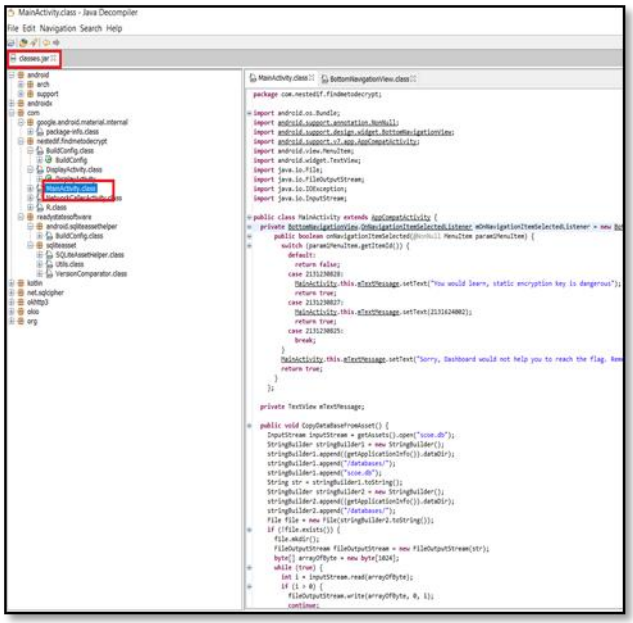

Fig. 7. Source code of apk

Android manifest file helps to check the permissions provided for the application in the device as shown in Fig 8, an application provided with read and write permission to an external storage (SD card). Any of the application should not provide permission to external storage, which is a major problem with this permission, because any data written to the SD card will not be deleted even though the application is uninstalled from the device. Hence the application provided with this kind of permissions must be avoided.

Fig. 8. Insecure permission provided in manifest file

Lack of Binary Code Obfuscation is other major security misconfigurations that occurs in most of the android application. Obfuscation is nothing but keeping the data in a non-understandable format. Generally any of the apk files can be reverse-engineered and the code can be modified and the modified code can be recompiled to have the apk file again, by doing this the application losses its original functionality. So to avoid this the developers should write code in the obfuscated format, for example using alphabets or numbers as the name of the file and class instead of naming it in an understandable format.

In Fig 7, it is clear that there is no implementation of obfuscation so the application has lack of binary code obfuscation vulnerability. In this way, the analysis of the source code and the manifest file can be performed to find the security misconfigurations.

C. Pentesting of Local Storage

All the applications installed in the android device will create local storage of it in the device storage with the unique id, in the path /root/data/data/<package name>. The local storage of the demo.apk file is as shown in Fig 9. In local storage the application creates several folders like cache, databases, backup files, shared preferences etc. Generally, the apps retrieve the recent searches from the cache directory of the local storage without the internet connection, and also it stores the other information related to the applications and user details in database folder created in the local storage.

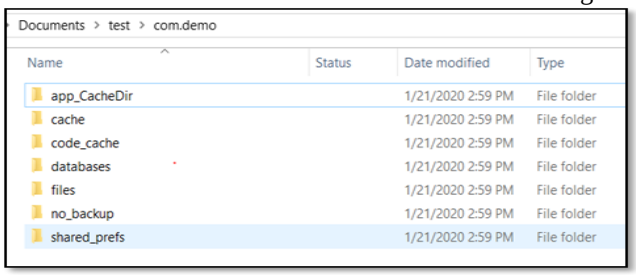

Fig. 9. Local Storage created by demo application (demo.apk)

Generally, all applications installed on the android device will have three files in common, they are Shared Preferences, database, and cache. The database is file-based databases with .db extension. The data stored in these database files must be ensured with a proper encryption technique. The shared preferences folder stores the key-value pairs. That is the preferences to share data with other applications stored in the device in a secured format by using the key value pairs. The files present in the shared preferences should not have any of the encryption keys or secret data present in it in an unencrypted format. If any of the encryption key found or any unencrypted data found the attacker may use that and gain access on other applications on the device or to the entire mobile device itself. Example from the "demo.apk", where the misconfiguration is present in the shared preferences folder, which consists of encryption key value as shown in Fig 10. This misconfiguration must be mitigated to ensure the application's security and the android device security.

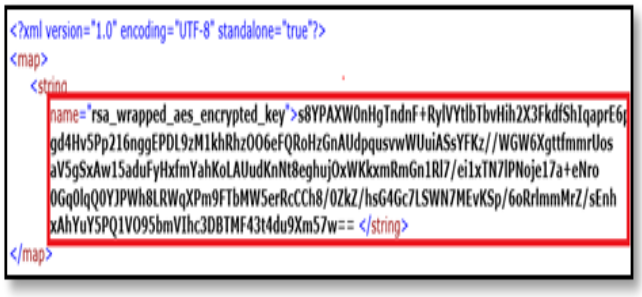

Fig. 10. Encryption key found under shared preferences Folder of local storage

IV. RESULT AND DISCUSSION

The Static Pentesting of Android application focuses on the client's data security and also the file system security created by the android application on the device. There are different types of vulnerabilities that are a combination of coding and Implementation. By analyzing the reverse engineered apk files the coding misconfigurations done in the application can be configured in a secure way. By the Pentesting of manifest file avoids several insecure access or permissions provided for the application. The local storage Pentesting avoids the insecure data misconfigurations. The OWASP mobile Top 10 vulnerabilities, ranges all the vulnerabilities of Static Pentesting from insecure storage, binary patching, insecure storage, cryptographic flaws, binary obfuscations, etc along with the prevention methodology [9]. The Pentesting process improves the security of the android applications through the process of information gathering, analysis of the gathered information and exploiting the security loopholes. Reporting the security vulnerabilities found in the application to the developers priorly helps in avoiding the security breaches later and also helps in maintaining the reputation of the organization and customer's data.

V. CONCLUSION

Mobile phones, particularly Android gadgets are being used everywhere. In addition, day by day the count of mobile device users is increasing. The advancement of Mobile applications is enormously expanding to make individuals' lives simpler. The Mobile application introduced on the android devices must guarantee the security of the client, wherein security is the responsibility of the organization, which develops the application. To guarantee the security of the users or clients data the association must perform Pentesting, leading pentest makes the association to maintain a strategic distance from security bottlenecks.

REFERENCES

1. Sriramulu Bojjagani, V.N. Sastry, " VAPTAi: A Threat Model for Vulnerability Assessment and Penetration Testing of Android and iOS Mobile Banking Apps," IEEE, San Jose, CA, USA,pp. 77-86, October 2017 [IEEE 3rd International Conference on Collaboration and Internet Computing].
2. Suranya Jayan, Jiangfeng Sun, Dongwan Shin, "An Efficient Approach to Securing User Data in Android," IEEE, eju, South Korea,pp 400-405, December 2017 [2017 International Conference on Information and Communication Technology Convergence (ICTC)
3. Haya Altuwaijri, Sanaa Ghouzali, "Android data storage security," ScienceDirect July 2018 [Journal of King Saud University - Computer and Information Sciences]
4. Akond Rahman, Priysha Pradhan, Asif Partho, and Laurie Williams, "Predicting Android Application Security and Privacy Risk With Static Code Metrics," IEEE, Buenos Aires, Argentina, pp 149-153, July 2017 [2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)]
5. Kamran Shaukat, Amber Faisal, Rabia Masood, Ayesha Usman, Usman Shaukat, "Security Quality Assurance through Penetration Testing," IEEE, Islamabad, Pakistan, Febraury 2017 [2016 19th International Multi-Topic Conference (INMIC)]
6. IDC. Smartphone os market share. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Online; accessed Nov, 2015].

7. E. McCallister. Guide to protecting the confidentiality of personally identifiable information. Diane Publishing, 2010.
8. Y. Shao, X. Luo, and C. Qian. Rootguard: Protecting rooted android phones. Computer, 47(6):32–40, 2014.
9. https://owasp.org/www-project-mobile-security/
10. https://portswigger.net/burp
11. https://subscription.packtpub.com/book/application_development/9781785883378/1/ch01lvl1sec14/owasp-mobile-top-10-risks
12. https://ibotpeaches.github.io/Apktool/install/
13. https://sourceforge.net/projects/dex2jar/
14. http://domoticx.com/java-decompiler-jd-gui-jar-class-bestanden/
15. https://owasp.org/

AUTHORS PROFILE

Pooja P, Mtech, Currently pusuing Internship under the Cybersecurity domain. She has completed her Bachelors of Engineering degree in PES Engineering College, Mandya, and Masters in Networking and Internet Engineering stream in Sri Jayachamarajendra College of Engineering, Mysuru. She has worked on several projects including Steganography, Electro adhesive wall Climbing Robot. And she has knowledge in Web application and Mobile application Pentesting.

Puneeth K M, Mtech, is an Assistant Professor,in Department of Electronics and Communication Engineering, Sri Jayachamarajendra College of Engineering, Mysuru, KARNATAKA STATE. He has completed his Bachelors of Engineering under vishweshvaraya technological University and Masters in Industrial Electronics stream in Sri Jayachamarajendra College of Engineering, Mysuru. He has expertise in the field of Operating systems, Image Processing techniques, Mobile computing and Automotive electronics. He has worked on several projects including "Smart wireless Sensor Network for Water Quality Monitoring in Real Time". He has done publication of project “Modeling and Analysis of Three Level DC-DC Boost Converter for High Gain Applications”, in International Journal of Engineering Science and Innovative Technology (IJESIT), Volume 3, Issue 3, May 2014, PP 339-351.