Journal of Informatics
Electrical and Electronics Engineering, 2022,
Vol. 03, Iss. 01, S. No. 004, pp. 1-11
ISSN (Online): 2582-7006

Vulnerabilities in Android OS and Security of Android Devices

Mayank Singh Saini [1], Sheenu Rizvi [2]

[1,2]Amity School of Engineering and Technology, Amity University Uttar Pradesh, Lucknow Campus, India
[1]mayanksinghsaini16@gmail.com, [2]srizvi@amity.edu

How to cite this paper: M. S. Saini and S. Rizvi, "Vulnerabilities in Android OS and Security of Android Devices," *Journal of Informatics Electrical and Electronics Engineering* (*JIEEE*), Vol. 03, Iss. 01, S No. 004, pp. 1–11, 2022.

http://doi.org/10.54060/JIEEE/003.01.004

Received: 05/04/2022
Accepted: 24/04/2022
Published: 25/04/2022

Abstract

The number of malicious mobile and android based application targeting Android users and their smartphones has increase at a fast rate. In addition, these malicious harms causing application are powerful enough to download modules from remote servers which are usually run by malicious infectors, due to which unexpected events can be triggered inside the smartphones. Then, the attacker gets control and hold on the personal and private information and data which is stored within the person's smartphone illegally.

Keywords

Vulnerabilities; Android Security; Smartphone; Security; Attacks.

1. Introduction

Smartphones today are not just palmtops or phones but can also be referred to as portable computers, providing a very diverse services needed by the user in life which include calling, texting, email services, GPS, camera-photo/video capturing, Wi-Fi service and Bluetooth related application. These applications hold and are responsible to manage a diversified intrinsic data of the user as well as quiet sensitive private info of the user such as an address book. Smartphones help us enable rapid and easy data-exchange with the help 3G, 4G, and upcoming 5G bands and Wi-Fi or Bluetooth. Therefore, the personal information which is stored in our Smartphones is vulnerable and prone to leakage or attacks by external malicious software.

The immense recent growth in the Global Smartphone Market has proved experts' estimates wrong and underestimated [1]. In accordance with the IDC, as depicted in Figure 1, in total the combined share of the Android and iOS Smartphones' market has had a rapid surge above 90% in 1st and 2nd quarter of year 2020. Though it was well predicted by IDC that 2020 would witness a huge surge in the smartphone shipments in comparison to the Feature Phone, the research predicted that about 1.2 billion newly manufactured smartphones were expected to be shipped for sale this year.

According to the market research performed by the IDC, about 1.2 billion brand new smart phones would be manufactured and shipped in 2021, out of which nearly 960 million smart phones would be running and powered by Android-OS, therefore accounting to about 79 percent of the overall new smart phone market share in 2014, thus becoming the invincible leader of the overall smartphone market. iOS, which firmly holds the second-spot, would then have about 14.9 percent share of the market with about 180 million shipments this year.

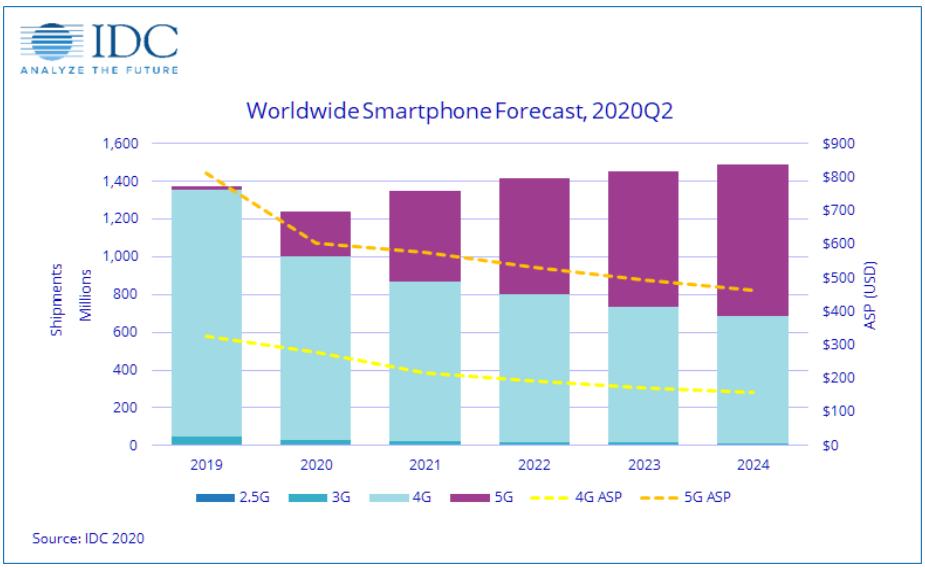

Figure 1. Global Smartphone Market Share forecast 2019-2024 by IDC.

Such a rapid and remarkable surge in the smartphone market could be attributed to ubiquitous computing performed on the varied mobile phones and the ease of availability of various diverse application to accomplish the required need of the user. Various new features added periodically with updated and the data which is so easily available on the Smartphones has become so diversified and quiet complex with proportion to the ever diversifying of the application. Surge in universal awareness and realization of importance of the privacy protection of the smartphone user has highly driven the increase in demand for the reliable security of data that is stored and is easily assessable on Smartphones. Although all the smartphone developers and the skin or running OS developer in the industry have established and supported the specific security measures which they provide in-built, still many vulnerable points of the software are revealed over time as the users explore of the services and uses them. Many such malicious vulnerable codes have been detected and reported that tend to lead to some vulnerability thus resulting in the flaws in the security of the operating system which in turn leads to the financial loss or the private information may get leaked and the smartphone gets DDoS [2].

According to reports by AVTest, the 2013-2022 duration saw a huge surge in the malicious codes that led to vulnerability (Refer Figure 2) and this time was accompanied by very significant surge in the number of the Android users [3]. The malicious codes lead to various security vulnerabilities and flaws continue to increase and evolve thus vary in the terms of functionality, and the way it operates the malicious activity and the techniques evolve too with time. Notably, the Android provides the world with open sources to facilitate in the application development which in turn favors the free and lesser restricted environments, thus many vulnerabilities creep in and have been found either by the firms or by the community members that keep a close eye on the code evolution and are striving constantly looking for bugs and vulnerabilities in the system and code of the android operating system. [4]

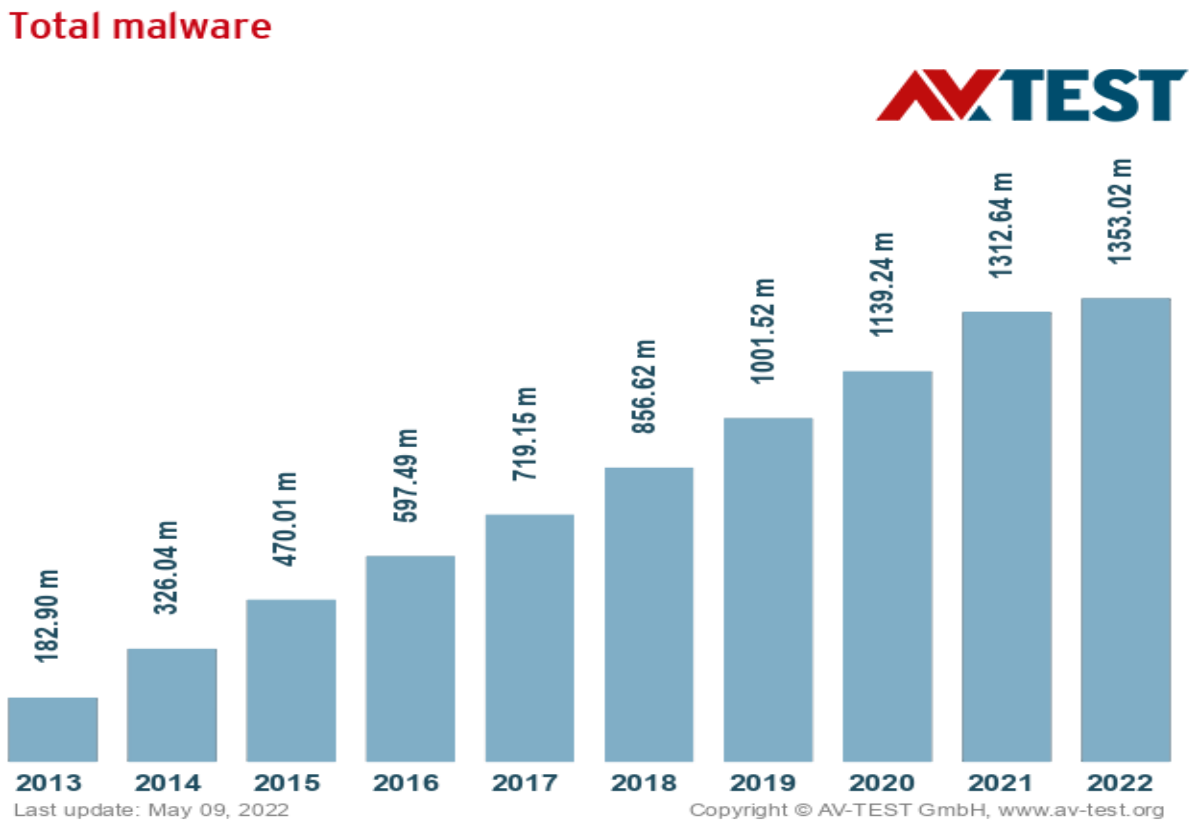

Figure 2. The rise in the malicious codes and its course over the years

Android operating system is evolved to manage each and every application and the data of the application separately in order to protect the data better. For application which require communication between other components and interact, such as the update application services provides by Google Play, data needs be exchanged and interacted with. To fulfill this motive of sharing of data among various application, Android has a provision of Content Providers for its user. The security frameworks of android and its architecture has its basis on Sandbox, which works to isolate an application's code and its data from that of the other application, and the required permissions, which controls the application's accessibility to the internal system functions. Out of the various ways one way to identify Trojan-horse type of the malicious application is to thoroughly examine if an application is requesting for excessive and unnecessary permissions which are irrelevant to its functioning for the system; according to a range of various studies conducted regarding the excessive permissions that applications request. Content Providers do make use of the permissions as well. However, the need to examine the permissions being requested by the attacker application is quiet far from the determining and prevention of the illegitimate accesses. This study paper strives to analyze various vulnerabilities of the Content Providers and strives to implicate relevant risks which may occur by implementation of the malicious application that seeks to leak or contaminate data of other system application by exploiting the loopholes in the security and vulnerabilities left open by coders.

2. Related Studies

This section elaborates the basic structure of the Android application, and basics of Android security architecture & the Content Provider.

2.1 Android Application Structure

Applications that are developed for the Android based Smartphones are usually the compressed files that comprise of the compiled source-code programs, its resources, necessary information files which are needed to run the application codes on the Android operating System and the developer code's signing file. As depicted in Figure 3, classes. dex is compiled JAVA source code program. Other resources such as the strings, the images and the UI interfaces are stored in the RES folder. Resources.arsc stored the information related to resource. The META-INF stores the electronic-signature files. The various electronic signature make use of SHA1 as its very own hash function and the RSA as default signature algo. Once the developers have signed the programs coded by them with the help of the varied private keys they hold, then the developed programs are ready to be installed on the users' smartphones [4].

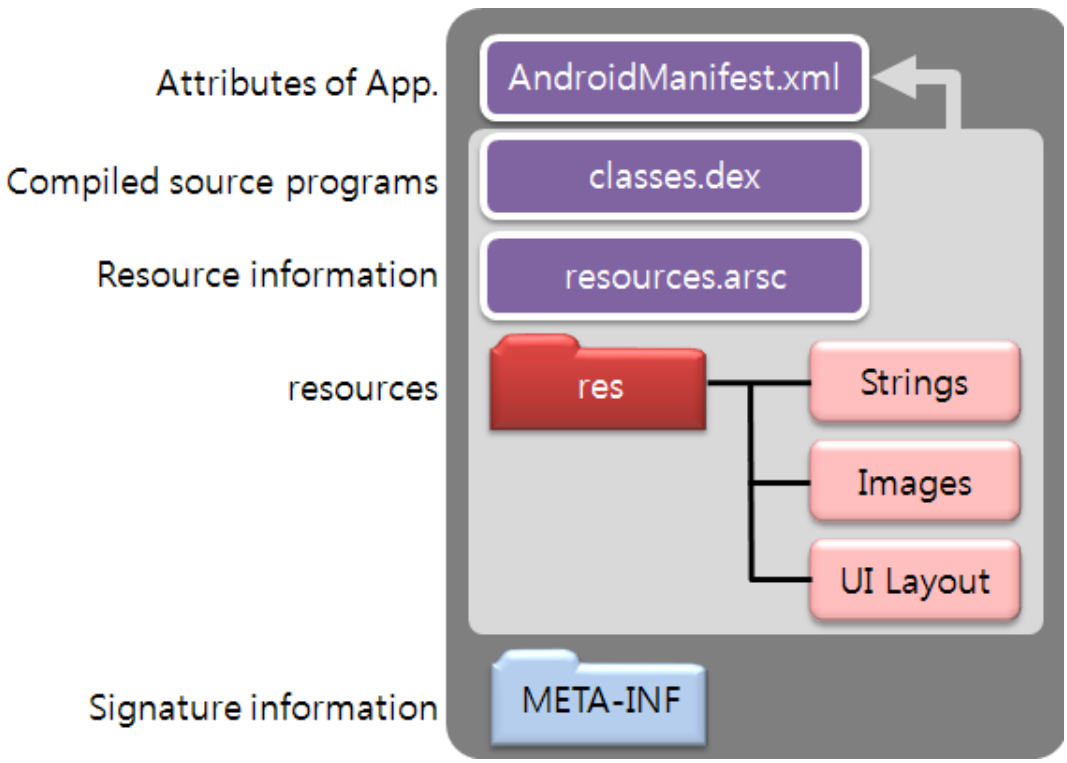

Figure 3. Android application file structure

AndroidManifest.xml is the compiled binary file which has the essential information which is required to be notified to the Android System before any application begins execution, e.g. the name of the application, the activities that make-up the application, the service it provides, information about the Content Provider, the information about the permissions to access its protected android API and then the version of the used Android API [5].

3. Android Security Architecture

As the Android evolved the Security Features now are embedded in the Android operating system to lessen the various vulnerabilities and various security issues which are associated with the application. To be specific, the file system of the android system is encrypted to protect it against any of the theft or the loss of devices due to malicious software; due to which the Sandbox

concept was adopted to help isolate applications' data and the codes from all other application; and the developers can then define permissions which are specific to their application [6].The Sandbox also isolates each applications' codes' and the stored data from the data of the other applications', which helps in preventing them from being influenced or running any other application, which acts as a perpetuator which assists to protect the application's source code and its stored data.

Nevertheless, the address books or the photo albums are provided by android system are now required to be available for use and access by other chatting or media sharing applications, this aspect is being exploited and misused by many other malicious applications. In support of this approach and functionality, an API known as Content Provider is in charge of the authorization and managing the accesses to the databases of the other applications on the system which are provided. Content Provider and the permissions are elaborated in detail in the following sections.

4. Content Provider & Permission

An android-based application is unable to directly access the databases of the other applications even though it has the exact knowledge of their locations in the database and their structures. Thus, to form a way to provide permit for the access of the databases of the other applications; the Android offers the Content Providers [7] based on the client server model. The server applications which own databases should make use of the Content Providers which helps them with the process of sharing the stored data and the database with the other applications. Server applications are assigned URIs which help them to identify their own databases externally.

Thus, it works as a database identifier, then the server id responsible of assigning the URI in the AndroidManifest.xml file or to the Content Provider, and the client application that initially intend to use the other applications' database now makes requests by sending queries to the server applications via the given URI.

Thus, client applications are assumed to know the application names, the application table structures, and the URI of the application's database intended to use. Client applications make use of the Content Resolvers which help them to communicate with the server-side of the Content Provider.

The Client application then sends the queries and URIs to the Content Provider with the assistance of the Content Resolvers. Then, the Content Provider receives the query answer from the database via the DB Helper and then sends them back to the Content Resolvers. Differently stated, the Content Provider and the Content Resolvers do serve as the window which help in sharing the databases between server and the client applications. The Relevant structures and the processing flows are illustrated in Figure4.

To use some of the major system resources or the necessary functions, Android applications should request for the permissions from Android system via the AndroidManifest.xml (Refer Figure 5). For example, to capture the images from camera of a Smartphone, any application should get to the android.permission-group.CAMERA permissions from the underlying Android system. In same manner, to make use of the internet services, any application should get the nod from the android.permission.INTERNET permissions from the underlying operating system- Android. Android also defines and has over 200 permissions to offer [8] and it enables the developers to use and define their very own permissions. The Server application can then make use of the permissions which will in turn help them to restrict accesses to their source code and databases. In common, the server applications are defined in the AndroidManifest.xml; which also includes their specific permissions to allow them to perform the read and write actions to the database while they share their databases. The Client application described in the AndroidManifest.xml the various permissions which are assigned to the databases which they intend to make use of and assist them in making requests to the android system.

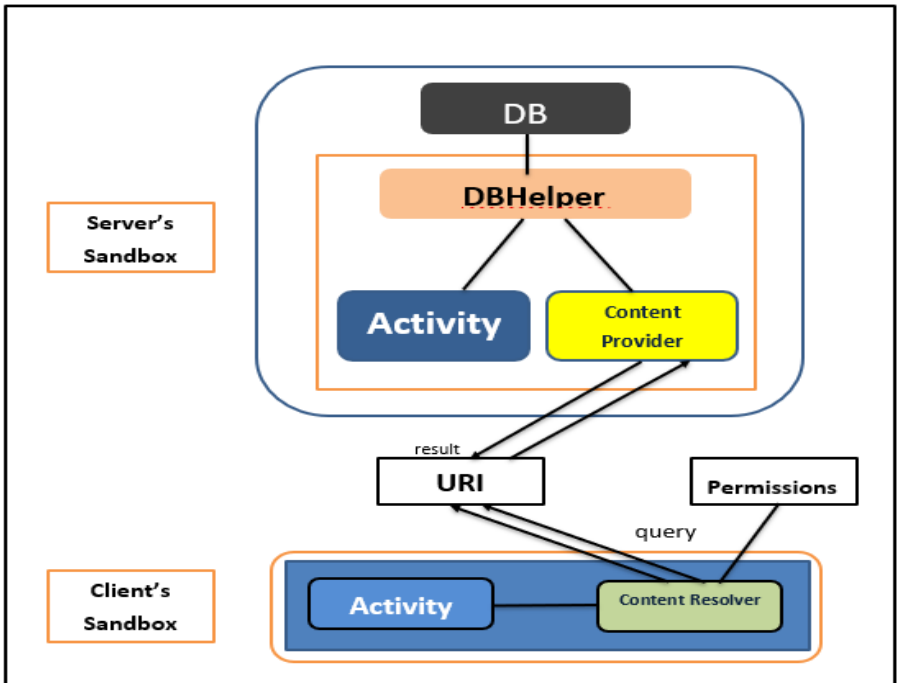

Figure 4. Database sharing between Application

Once the client application is running and the Content Resolvers sends queries to the Content Provider, then Android System checks if the client application is capable of owning permissions which are required to access the private and the requested database (Refer Figure 4). Unless the client application owns the permissions required for the access of those databases, the service gets denied.

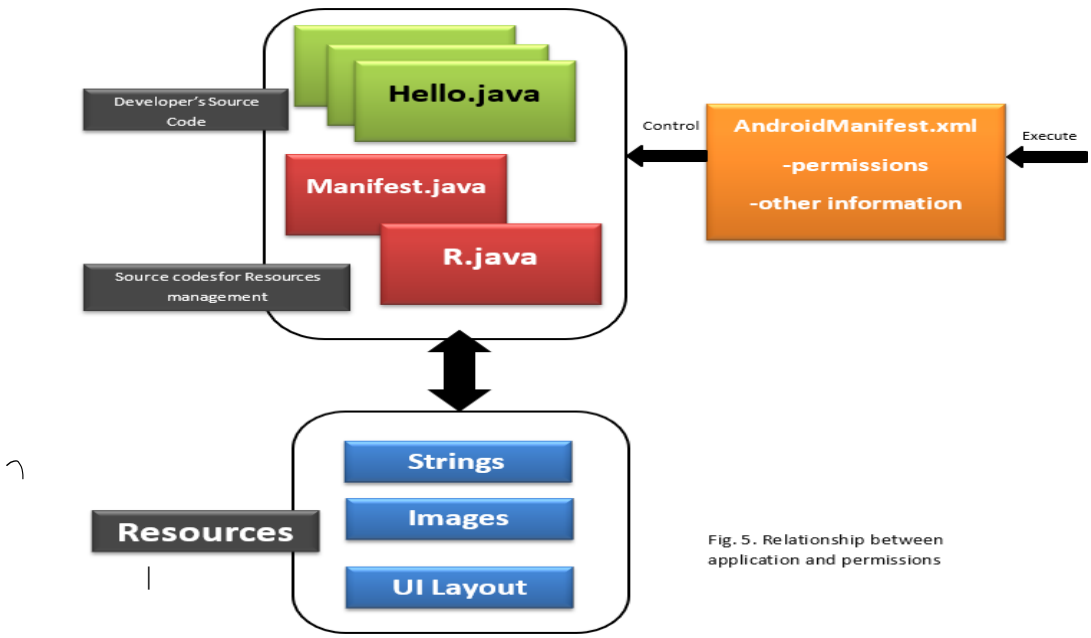

Figure 5. Relationship between Application and permission

5. Reverse Engineering

In order to access the databases of the other applications, URIs are assigned to the given database, the data tables are required for the requested queries, the information about the fields attributes and the various permissions requested by the server application are required. Taking in consideration the databases including the address books, Android system provides it as a default means to be shared with any other application, not just the URIs but also the access methods are usually offered with the assistance of APIs. In addition to this, when any server application developers do develop the client application too, they now own all the necessary information to be used at their disposal. This study, thigh, focuses on the various possibilities and the risks of the malicious application leaking the private information by the use of any illegitimate applications' roaches. Similar to the compiled byte code file, the classes.dex stores the information which attackers require in order to file and write queries through the client application, the attackers need to restore these source programs with the assistance of the reverse engineering.

5.1. Dedexer

Figure 6 depicts the process of the file conversion which is in creation of the final .apk file for the application being developed in android development processes. Android developers code and compile Java Source programs in .java to create class files, which are basically byte-codes which are based on the JVM (Java Virtual Machine). These .class extension files are then converted to a dex type file using the dx tool. Then the Virtual Machine (VM) running applications on the Android devices is basically called the dalvik [9], which is capable of reading the byte codes of the converted dex files. The newly generated dex files, xml file and the resources are then compressed into java'sarchive .jar file format which helps to generate an apk file for the application. Dedexer is a widely used tool used for reversing and compiling the dex files into the Java source code files. Dedexer is used to reconfigure not only the source code written by developer in java but also the R.Java, it is a program which is related to many resources which are generated in the due course of development. Although the Dedexer tool are not capable of always restoring the complete sources but that mostly do contain errors, it also gives us an overview of the program flows.

5.2. Apk Manager

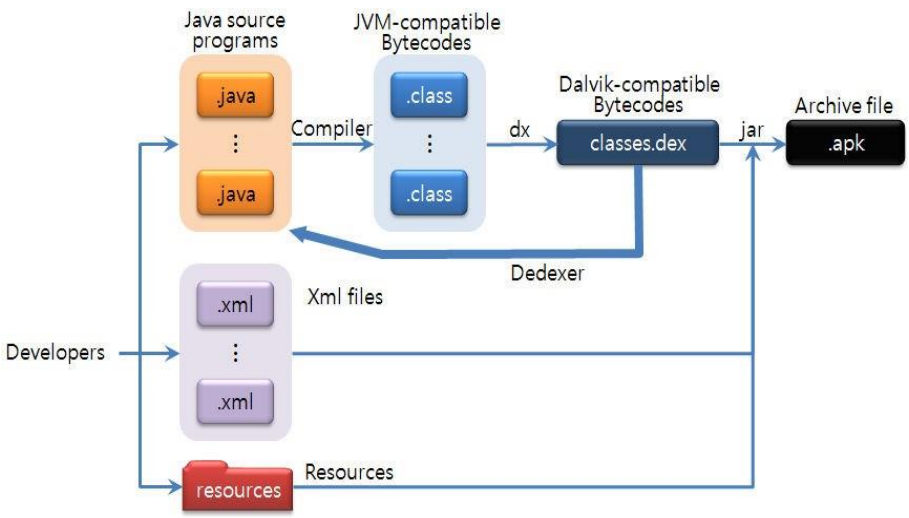

Figure 6. Conversion of the android application files

Dedexer can also be considered to be an ideal tool to be used by the attackers by using the method of the reverse engineering mechanism as it is capable of restoring the Java source code from the .dex files. But, as it has been mentioned earlier, it is not currently supporting the complete total restoration. Also, it restores the Java codes, but without supporting the resources and the xml files. Thus, it is quiet difficult for it to complete the mentioned operable application with just analyzing the existing android application and then turning them into the malicious codes. The ApkManager is used to reverse compile the original Java programs into the Smali-based ones, which are capable of reading not only the source code of the application in use but also the AndroidManifest.xml. modifying and compiling the restored codes generates the application which does operations on the various Android devices (Refer Figure 7) [10].

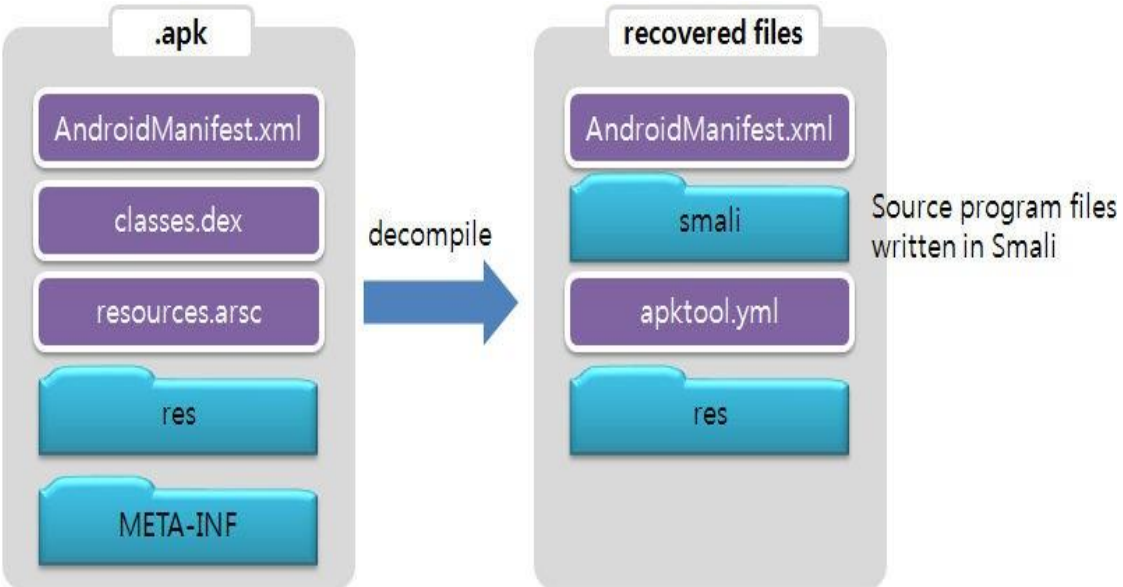

Figure 7. Reverse engineering with the ApkManager

5.3. Proguard

Android operating system provides the users with a feature known as *Proguard* which is instrumental in prevention of the reverse engineered programs from being utilized for the use in the malicious programs or from infringing the copyright [11]. *Proguard* can be used optionally by including the use of proguard.cfg file in any of the Android project. *Proguard usually* deletes the unnecessary codes to reduce and optimize the code volumes, and facilitates in changing the classes, the fields and the class names into a meaningless one for the obfuscation. Although it is a time-consuming process to determine the meanings of the changed names when the obfuscated sources are being restored using the process of reverse engineering, but analysis is not an impossible task as the logics are not altered, which is the limitation of obfuscation.

6. Safety Status of Mobile Devices

Smart-mobile devices today include smartphones, PDAs, and Tablet PC and so on. People now a days use them as a means to store the private information, send & receive e-mails, messages, media of various formats, browse the internet, process various documents, and to entertain themselves. In the recent years, as the smart phones and smart devices are evolving and are becoming more and more powerful and are having higher market shares, smart phone devices are playing an immensely important role in many people's lives. The useful application of the smart phone devices are covering almost all the aspects of day-

to-day life, e.g., chatting, navigating, gaming, listening music, processing document, remote payment and SNS. The more and more powerful and convenient functions smart phone devices have, more prominent the security issues become.

The security and guarding the privacy of the smartphone devices have become a major issue that can not be ignored easily. In the smart phone device market, IOS, Android and Windows phone are the three major relatively mature operating systems. According to the Gartner research results [1], Android system usually accounts for 50 percent plus of the total global smart phone devices' operating system. The open source of the android and the large market share it holds makes the android operating system the most vulnerable to malicious activities. Therefore, this paper strives to implement a secure system which enhances the security of the smart phone devices in the Android operating system which is based on the cloud. Using a client in the Android device to help interact with cloud.

7. Current Safety Status of Mobile Devices

As a response to the malicious threats to the smart phone devices, many security vendors are now actively offering various security products. One of the most common methods of detecting the malicious application which is used by the security products is by extracting the application's signature and then comparing it with the signature in malicious applications' signature database. If they are same, then the application is malicious. This method of detecting the malicious apps can detect the common and frequently used malicious applications, while it remains relatively inefficient for the newer, and unpopular or the third- party applications. This being the reason why the detection rate for the malicious applications is not as high as it should have been. One of the other drawbacks of this method of signatures comparison is, whenever any malicious application attempts to replace their signatures, the malicious application's signature database would also have been updated accordingly. And the growth in the malicious signature databases makes the comparison of signature become a tedious work consuming lots of time and electricity.

Therefore, the Security providers proposed a new and better efficient security model, popularly known as cloud antivirus for the mobile devices. The basic idea of cloud antivirus is to extract the signature of the doubtful application on the smart phone devices and send it to the remote cloud for the purpose of detection and returning the results to the smart devices from the cloud. In previously designed detection scenario, mobile devices had to bear a lot of processing work, which consumes the valuable power and the system resources of the smart devices. While when the signature is sent to the cloud antivirus, it bears a huge number of processes and tasks, thus the smart phone devices are liberated from the tedious work and the rate of detection gets improved. Still, the detection rate of the latest malicious applications is still not as high as it should be, using the existing cloud antivirus system provided by the security vendors.

8. Detecting Zero Day Attack

In the Android Operating Systems, the ways of detecting the zero-day attacks are broadly divided into two main categories, namely dynamic analysis and static analysis. The static analysis works by analyzing the .apk files or the .dex files of the android operating system. William Enck proposed the static analysis method which works by analyzing the permissions of an Android application [2]. In this method it first analyzes the permissions which are granted to the application while it is running and performing tasks. Then this method compares them with the permissions that are already declared in AndroidManifest.xml file by the developer. If the permissions are different, then the application maybe hiding the dangerous permissions which it may use to exploit the system and use it for malicious activities. So this installation is prohibited. But this method is only efficient in analyzing the hidden permissions in the application. If this is not the case and there are no such hidden permissions, then the user is own his own to determines if he wants to continue with the installation of the application or abort.

Even though any malicious application declares quiet a number of very dangerous exploitable permissions (e.g., CALL_PHONE, SEND_SMS), but if the application is not hiding these permissions, then the detection method will not prohibit the installation of this application as it will consider it to be legitimate as the permissions are not hidden. While the majority of the

Android users are unaware that these permissions may be a cause of serious consequences and they continue installing this malicious application. Thomas Blasing proposed a methodology to assist in detection of the malicious behavior by the assistance of an emulator. They basically make use of a monkey, which is capable of generating the pseudo-random streams of the user events such as the clicks, touches, or the gestures, even the number of system level events occurring. Then in this method we place a sandbox in the kernel space to help in logging the behavior of the application at the android operating system level. Then in this method the analysis of the resulting Log file id done which helps in determining whether the application in use is malicious or not. The method is quite efficient for the security vendors which help them detect the malicious behavior of any new application launched in the market. But this method is still not adapted to be used in the ordinary user's smartphone devices.

9. The Security Problems Faced by The Smartphone Devices

Before introducing the android operating system, we first need to eliminate the security issues and flaws being faced by the smartphone devices, which should include the following aspects:

- **Malicious application-** Recently, there have been a huge surge in the number of malicious applications over the web and the app stores. These malicious application now-a-days are not only published in some third-party application market, but also found hosted on the Google Play Store. Often these malicious applications cause serious problems and leaks which result it monetary losses. Some application are capable of revealing the user's last or current location, or personal contacts or other private information; some are able to send short .text messages or even make the phone calls in background without the knowledge of the user with the malicious aim of generating profit for the attacker; whereas some applications will download the other malicious applications from the server which would in return harm the device and open a backdoor for the attacker to seamlessly access the personal information of the user.
- **Unsafe websites-** Some websites do host a large number of the malicious applications. In addition, [6] this would lead a user of the Android phone who is using the web browsers to surf the Internet may get exploited if they visits these malicious pages and click on the links; then the attacker gets access to run any malicious code with the privileges that the web browser application provides or has been given by the user to the browser application.
- **Data security of mobile devices-** Smart phone devices are quite different from personal computers (PC) because of their ease of portability, and thus they are at a high risk of being lost. When an user loses his or her mobile device, then the local data on the device is quite difficult to retrieve. Some of the other reasons such as the misuse by the user, or damage caused to the device may also be the cause of data loss.
- **Network Data Security of the smartphone devices-** When any user connects to an access point (APN) set up by the attacker, then the traffic of his/her smartphone device can be sniffed. [7] elaborates that, the attacker if sets up a rogue Wireless Hotspot (Wi-Fi) can easily get a peek at the traffic of the users' device and thus can use it at his disposal to exploit the information of the android user. When a smartphone user uses this rouge hotspot to surf the web, then the user's personal data and the credentials (including the user's name, his password, maybe the credit card numbers etc.) would get compromised to the attacker and can then be exploited to the disposal of the attacker. Even if the user id using the SSL connection, even then the attackers no do have a lot of methods to get access to the user's personal data and info.

10. Conclusion

On the Android Smartphones, lots of personal information of the users is stored in the local databases. Android currently relies on the Sandbox to help protect the codes and the data of an application from the other malicious application, whilst it does offer the *Content Provider*s to allow the necessary and in sight sharing if databases when requested or required for the proper functioning of the other applications. Client applications which intend to share the database of any other application do need to be present within the same URIs in which the database identifiers and the database information is present as well as the permissions required by the server application should also be within the same URIs. Attackers who attempt to gain access by means of any unauthorized or illegitimate method can make use of reverse engineering which help them to extract all of the required

information to exploit the system. *Proguard still* does not guarantee us the prevention of the information extraction. In addition to assigning the permissions which controls the accesses is also not mandatory, which in turn leads to the security loopholes and make the system vulnerable and prone to attacks. In other words, the attackers can get easy access to the private information on Android Smartphone devices with almost no particularly complex analysis or the alteration or complicated forgery of the server application.

References

[1]. "Mobile Phone Tracker," IDC: The premier global market intelligence company. [Online]. Available: https://www.idc.com/get-doc.jsp?containerId=TEA004804. [Accessed: 28-Mar-2022].

[2]. S. Seo, D. Lee and K. Yim, "Analysis on Maliciousness for Mobile Applications," 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012, pp. 126-129, doi: 10.1109/IMIS.2012.190.

[3]. M. Diaz et al., "Technology news, analysis, comments and product reviews for IT professionals," ZDNet. [Online]. Available: https://www.zdnet.com/. [Accessed: 28-Mar-2022].

[4]. A. Singh and P. Singh and A. K. Tiwari, "A Comprehensive Survey on Machine Learning," Journal of Management and Ser-vice Science, vol. 1, no. 1, pp. 1–17, 2021.

[5]. N. Srivastava, U. Kumar, and P. Singh, "Software and Performance Testing Tools," Journal of Informatics Electrical and Electronics Engineering (JIEEE), vol. 2, no. 1, pp. 1–12, 2021.

[6]. "Gartner says worldwide smartphone sales declined 5% in fourth quarter of 2020," Gartner. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-02-22-4q20-smartphone-market-share-release. [Accessed: 28-Mar-2022].

[7]. W. Enck, M. Ongtang, and P. McDaniel. Mitigating Android Software Misuse Before It Happens. Technical Report NAS-TR-00942008, Network and Security Research Center, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, USA, November 2008.

[8]. Thomas Blasing, Aubrey-Derrick Schmidt, Leonid Batyuk, Seyit A. Camtepe, and Sahin Albayrak. An android application sandbox system for suspicious software detection. 5th International Conference on Malicious and Unwanted Software (Malware 2010), Nancy, France, 2010.

[9]. A. Singh, P. Singh, "Image Classification: A Survey. Journal of Informatics Electrical and Electronics Engineering", vol. 01, Iss. 02, S. no. 2, pp. 1-9, 2020.

[10]. S. Kumar, P. K. Srivastava, G. K. Srivastava, P. Singhal, D. Singh, and D. Goyal, "Chaos based image encryption security in cloud computing," J. Discrete Math. Sci. Cryptogr., vol. 25, no. 4, pp. 1041–1051, 2022.

[11]. N. Srivastava, U. Kumar and P. Singh (2021) Software and Performance Testing Tools. Journal of Informatics Electrical and Electron-ics Engineering, Vol. 02, Iss. 01, S. No. 001, pp. 1-12, 2021.

[12]. A. Houmansadr, S. A. Zonouz and R. Berthier, "A cloud-based intrusion detection and response system for mobile phones," 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W), 2011, pp. 31-32, doi: 10.1109/DSNW.2011.5958860.

[13]. G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: Versatile protection for smartphones," in *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC '10*, 2010.