ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 7, July 2017

Server Based Secure Mechanism To Prevent Reverse Engineering And Privacy Theft In Android Application

[1] Raja P, [2] Geogen George
[1] PG Scholar, Information Security And Cyber Forensics SRM University Chennai, India
[2] Assistant Professor Information Security And Cyber Forensics SRM University Chennai, India

Abstract: ***Android an open source operating system has become the most preferred one in recent years. We have many security issues in android application which is to be addressed. Any anonymous user can extract the source code of android application by reverse engineering technique [1] and can re-use one's work they can also find the web services from the URL present inside the application and that they can get all the real time values from the server [2]. In our proposed concept we are going to secure the android application in maximum ways from the anonymous users. We develop our own appstore and launch all our application in it, where the user can install our application from our app store. Once the appstore is executed in users mobile, initially it will check for the root access privilege [11].***

Once root privilege the users are verified the users will be prompted to register themselves, pay and make use of child applications. We retrieve few unique ids from the user and store it in database. Each of our child application, the main logic are executed from the server as web application logic, by which we will be able to secure our logic of the application also the child application will execute only if our app store is present. We also prevent privacy theft such as user authentication, black listing the users who misuse our application. To secure the web service URL'S, we encrypt the URL and decrypt it using the dynamic keys at the server point.

Keywords- ***Reverse engineering, root access, appstore, code obfuscation.***

I. INTRODUCTION

In recent years usage of android operating system has become wider for mobile devices and the market for Android applications has increasingly grown in variety and financial volume. This platform is designed in a way that developers can upload and publish an app on Android market [1] without a review from Google and users can easily download and install new applications.

Most of the security solutions present in the market now address only on client side [2]. Firewalls, antivirus, digitally signed software etc provide software protection at user end perspective [6]. These security measures do not protect against exploitation of software vulnerabilities and reverse engineer [5].

The Android platform requires various steps and tools till APK is built and made ready deployment. Initially, the build process begins with compilation of java file to a ".class" file. Next is the transformation of Java byte code to Dalvik byte code [1]. The result file .dex will be saved as "class. dex". Then the .dex file will be added to the APK. Finally the build process is of signing and packing the android application.

1.2 Reverse Engineering:

Reverse engineering is the method of involving how things work and extracting the source code and reusing it for further purpose. This concept can be implemented in android platform where the source code of an android applications are extracted using the technique called reverse engineering and those source code are made use for future purpose. Reverse engineering of android application can be performed for many purpose, such as we can read other's code find vulnerabilities present in the code, extract and search for any sensitive data hardcoded inside the code, Modifying the existing functions of the application and making it to behave anonymously can also be performed.

Decompilation is a technique of transforming software binaries into a readable text format by which the source code is written in a high level language, in order where the developers will be able to read it. Performing the operation of decompiling the byte code, we will able to retrieve the source code of the program.

Reverse engineering of the software application, one can know the working method and the technology behind the software used. Having this in hand re-designing of that software can be done. This involves, stealing of some ideas from the existing

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 7, July 2017

software. The idyllic outcome of a reverse engineering process for an android application is to rebuilt, java source code out of the distributed binary form. This techniques is widely used where Code obfuscation will not be able to prevent reverse engineering of the application, but it can make little bit hard to crack and get the source code. and moreover it is time consuming[4].

II. RELATED WORK :

Currently most of the developer does not worry about the security of the application. At the most is they make use of byte code conversion, proguard mechanism and encrypt the application, writing dummy codes.

Obfuscation:

[4] Obfuscator completely changes the JavaScript code to indecipherable form, preventing it from examining and theft. It means class files, to read the source from class file but even reading the code they won't be able to make use them directly but unfortunately they will be able to understand the logic behind the application which makes easy to create the duplication of the application.

Smog Obfuscation System:

The concept of SMOG [4] is interpretation obfuscation. Instead of transferring and performing a normal application, the smog system allows the software vendor to pre-process their dex file to produce an obfuscated apk in order to tackle malicious attack such as reverse engineering attempts. [4]Normal Dalvik VM cannot comprehend this obfuscated apk file. Thus the execution token together with the obfuscated app decide the unique execution environment for an end user.

Extend Loop Condition:

Obfuscating the conditions inside the loop will confuse the reverse engineering process and makes it more difficult. It could be implemented by prolonging the loop condition to second or third condition that does not perform any operations. For example, in the following example we have a simple if condition[1].

Before:

```
int x = 1;
if (x > 200)
{
  …
  x ++;
  // call function
  abc(x)
      }
```

After:

```
int x = 1;
while (x> 200 ||
   x%200==0)
{
   …
   x ++;
   // call function
   abc(x)
}
```

Clone Methods:

It is significant on behalf of a reverse engineer to recognize the purpose of a function written inside the application and it is correspondingly important to understand the difference between the conditions under which the function is called nomenclature of obfuscation [1]. One could create clones of a function and call these functions under matching conditions. We can call the function reliant on any outward factor, which appears to be a determining factor but is actually not. Thus the obfuscated code will not be able to convert back to original form.

Encryption of Application:

Few developers make use of cryptography concept like encryption of APK file until it gets installed but after installation [8], the source code will be extracted and store in a location of system memory, so the hackers extract the decrypted APK file from the device, some developers develop the application as system application so after install it will store in system memory so usually can't extract that app but if we root the device we will be able to extract the apk file easily.

Proguard:

Maximum developers are using the proguard based security; this is the maximum level security of current security system. Proguard is to create the alias name for all the classes, methods and variables and it will make the class file as 1 into many so hackers can't even use the code as well as very much difficult to understand the logic of the application, but still hackers can hack the web service URLs from this code.

Packer and Unpacker :

[5] The packer has in hand of executable file of binary file, encrypts those file so that no one will be able to understand, even by the attacker unless or until it is decrypted. The encrypted dex file is generated using AES [5]. GroupID that is generated while registering, the passphrase is used for generating the key and it is kept as Password Based Encryption (PBE) [5]. This key can be used to retrieve the .dex file with an encrypted Dalvik bytecode. Finally the server is stored with the application that comprises encrypted .dex file [5]. The key manager stores the Password Based Encryption Groupkey by making use of use public keys of users registered [5]. At last every user's will be able to extract their GroupKey by their own private key which in turn be used for decryption for later purpose [5].
The important component of our design is Decryptor stub where it should be executed while starting the application itself [5]. The Decryptor stub performs few operations such as fetching .dex file, lading and decrypting the dex file into the memory and then executing it there. Loading the dex file are generally attained by reflecting which piles dex file into the memory from a location.

III. OUR CONTRIBUTION:

Our proposed system uses web server based methodology where most of the application logics are in web server and executed at run time. By this way the attackers won't be able to access our application logic, even if they do reverse engineering. Moreover the attacker's will not be able to access the URL provided in the client application, since we encrypt the URL within the java file. We also use reverse authentication process to authenticate the users each time the application executes. Each child application will check for the presence of app sore in the user's device. Only if appstore exist in the device the application execute and send the encrypted base URL as a request to the server. This provides an additional security to application.
Once our app store is installed, it verifies whether the device is rooted or not. If incase the device is a rooted one we do not allow for the next step to happen.

Dynamic keys are used for encryption and decryption of the other web service urls, adding to the previous measures, a proguard mechanism is also used to secure the application. Making use of dummy classes inside the application will give an additional security, where even if someone reverse engineer and get the source code, the logic behind it will not be our application logic. This actually confuses the attacker and tries to provide as much security as possible.

System Design:

The fig 3.1 shows the overall design of the project which implements security mechanism for android application. The architecture diagram has components such as database, webserver, appstore and the child application. This provides a detailed work done in the android application for securing the source code from attacker. The diagram describes the entire work in a sequential order. The initial step is downloading and installing the appstore application. The appstore checks for the root privilege (ie) whether the device is rooted or un-rooted by revoking "build.tags" and the presence of "sudo" file returns a value, depending upon it the next step is carried out. Next is device registration which creates a unique id for each user's in-order to have a record of legitimate users and the one who do privacy theft. Once the device is registered the child application is installed, which has the encrypted base URL, where its hash value is checked and communication between client and server establish by private IP. Then the source code is executed by UI based which leaves fewer artifacts if it crashes.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 2, Issue 7, July 2017

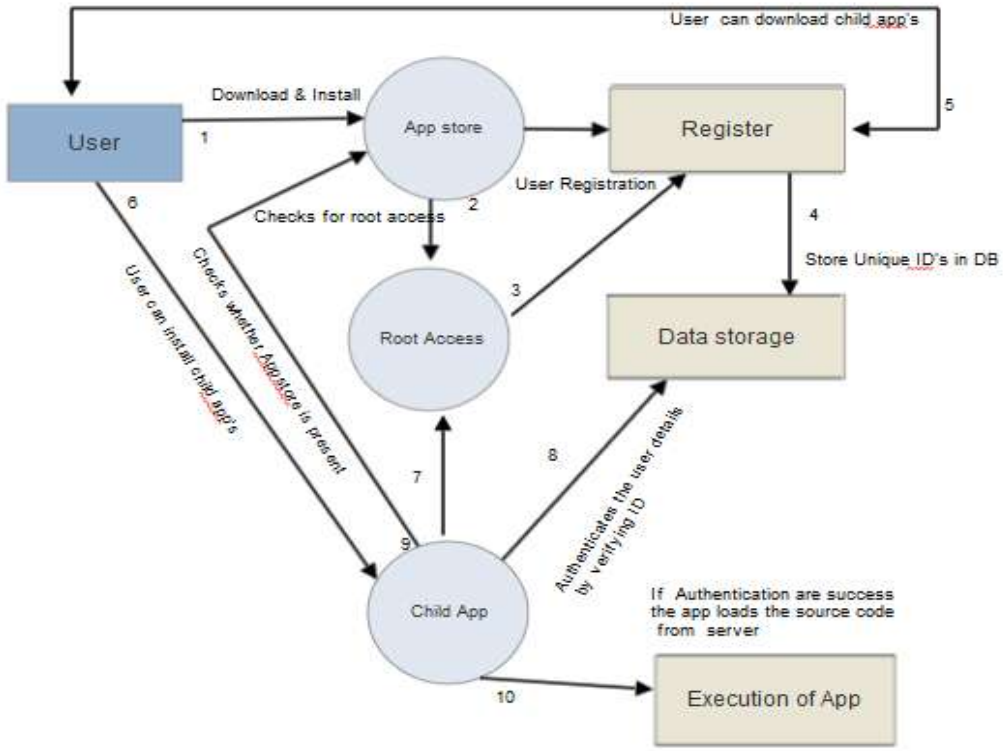

Figure 3.1

IV. BACKGROUND RESEARCH

This project all about android apk security, it provides the security to our application logic from the attacker.

Proguard:

Proguard, it is an open source tool (ie) available to anyone in the market for Android SDK [10]. This proguard is used for the development process while building the application. Basically the proguard is a Java obfuscator used widely for Android applications, since the android applications are used are written in java. The proguard set includes identifier of obfuscation for android application package, classes, functions and methods. In spite of this, protection mechanism it can add dead code (ie) dummy codes for securing the source code from anonymous user doing reverse engineering.
It Highlight's the dummy codes and where the user can split up the source code to hide. Classes and functions that are used inside the source code of the application is splatted into junk names and segregated apart and while it

Dalvik-obfuscator:

An open-source bytecode obfuscation tool [3]. Given a standard APK file as input, it outputs its corresponding obfuscated APK version. The underlying algorithm is the well- known under the x86 architecture junk byte injection.

Java Guard:

JavaGuard is a bytecode obfuscator, designed to fit effortlessly into your consistent build and testing procedure [13].. It provides a valuable Java code that is more secure while performing decompilation and while performing the other forms of reverse engineering[13]..

DashO:

It applies a layered approach to binary code protection using obfuscation, encryption and tamper proofing making your apps more difficult for people and machines to reverse engineer and alter [13].

Some Of The Ways DashO Protects Your Apps:

Performing the methods such as renaming the methods and making the source to be secure and difficult for the hacker to understand the concept by reverse engineering. Flow control familiarizes conditional statements as false and the

other deceptive constructs to confuse the hacker and make difficult to break the source code. Encryption of string permits to encrypt the strings and place those as a sensitive part inside the application and make use of it when required. Examines the source code to find the dummy methods, functions, types and fields, and removes them which in turn make the application to execute faster and reduce the compile time.

V. CONCLUSION:

This paper proposed a concept of protection system by SSM to prevent reverse engineering and privacy theft of an android application, which makes it difficult for an attacker to make use of one's source code.

We have described the architecture of the SSM protection system to enhance the protection of the android app and also prevents theft of privacy. Also we have implemented other security methods such as code obfuscation using proguard that makes the attacker, more unusable code that makes his work more tedious.

V. REFERENCES :

1. S. R. Tandan, Kamlesh Lahre, *June 2013* " *Shielding Android Application Against Reverse Engineering"* International Journal of Engineering Research & Technology.
2. G. Naumovich and N. Memon, "*Preventing piracy, reverse engineering, and tampering*," *Computer*, vol. 36, no. 7, pp. 64-71, July 2003.
3. Jim Huang July 2011, "*Practice of Android Reverse Engineering*".
4. S. Schrittwieser and S. Katzenbeisser, "*Code obfuscation against static and dynamic reverse engineering*," *Information Hiding*, pp. 270-284,2011.
5. Muhammad Shoaib, Noor Yasin *April 2016*" *Smart Card Based Protection for Dalvik Byte code Dynamically Loadable"* International Journal of Computer Theory and Engineering.
6. Pau Oliva Fora *"Feb 24 2016"* Beginners Guide to Reverse Engineering Android Apps
7. Nak Young Kim1 Jaewoo Shim1 Seong-je Cho " *Android Application Protection against Static Reverse Engineering based on Multidexing*"
8. Exestealth protector. [Online]. Available: http://www.webtoolmaster.com/exestealth.html/
9. http://geeknizer.com/decompile-reverse-engineer-android-apk/
10. http://www.singhajit.com/convert-apk-file-to-java-code/
11. https://github.com/honeynet/apkinspector/
12. https://www.honeynet.org/node/783
13. https://www.digitalmunition.me/2016/03/lobotomy-android-reverse-engineering-framework toolkit/
14. http://stackoverflow.com/questions/3424195/determining-if-an-android-device-is-rooted-programatically.