Reduced Transmission of Application Information Security on Android with Synchronization

[1]Y. Chandra Moleeswara Reddy, [2]S. Jagannathan

[1]M.Tech Student, [2]Professor CSE Dept
[1,2]A.P.S. College of Engineering, Bangalore
Email:[1]mouliyerram@gmail.com, [2]jagan4@gmail.com

Abstract—In present day's mobile phones which use android operating system is equipped with richer applications than mobile phones with traditional operating system. These makes user to store most of the information on mobile phones. The mobile phones with android operating system need to be upgrades with latest version of its operating system. Here comes the problem, whenever the mobile phone is upgraded the past important information of the user is lost or corrupted. So backup is needed to keep the information safe. It is well known the user can synchronize all the important information to the Internet and retrieves back their information. This paper talks first about the Android and threats to the user information. Then participants in synchronization and data compression techniques before the user information are synchronized with Internet. This improves security and speedup the retrieving of the information back to user and another approach is the management of exchange protocol used in transmission from mobile to server.

Keywords—Android Operating System, User information, Data compression techniques, Synchronization

I. INTRODUCTION

Android is a mobile operating system running on the Linux kernel. The Android platform is used by tablets, net books, e-book readers, mobile phones and a variety of other consumer electronics. In 2007, a group of handset manufactures, wireless carriers, and software developers (notably, Google) formed an Open Handset Alliance, and with goal of developing the next generation of wireless platform, Android is the first complete, open and free mobile platform. Android is Linux based open source operating system for mobile devices which provides operating system, an application middleware layer, a Java software development kit(SDK) and a collection of system applications[1]. It used by cell phones, tablets, net books and set-top boxes like Google TV.

Android has a large community of developers of applications ("apps") that extend functionality of the electronic devices. There are 200,000 apps are currently available for devices on Android Market. Android Market is online apps store run by Google, though apps can also be downloaded from third party[2]. Developers write apps in Java language and controlling the device via Google-developed libraries.

In this thesis, some research is done on low bit transmission of user information like personal information, business or professional information and other important information security on mobile platform and a solution improve security by using data compression techniques and speedup recovery of lost information when mobile is upgraded.

II. THREATS TO USER INFROMATION ON ANDROID

Apps Market provides many benefits to user like ability to rapidly and effectively acquire and enhance software. As it is designed for open and networked devices, it vulnerable to many attacks. These attacks makes phone partially or fully unusable due to these the important information of user is lost or corrupted. With emergence of new apps witnessed increased security threats that are exploiting this model of provisioning software. Attack vectors include Bluetooth, Internet (via Wi-Fi, 3G networks), USB and cellular networks. In the Android Market now days there are numerous cases apps infected by malwares and spywares. As user carry business or professional information in mobile phones these feature attract the attackers to attack the user information. This may lead to data leakage of the most sensitive data of the company if phones are attacked. So now it is challenge to improve the user information security on Android phones than on PC [3].

Google helps to run Android code on Apache Tomcat Server, which is open free software so most of the attacks can be done easily. The Android is more prone to attacks because of its open source software stack, whose source code and be easily manipulated, accessed and exploited by attackers. As everyone know that Android is built on Linux kernel, which is more secured and it has its own security mechanisms [4]. Many security mechanisms have been proposed these makes Android operating system more secured.

When new malware and spyware emerges then information will be lost before any preventive measure taken. So we compress the information and then synchronize the information to the Internet. These always helps to keep the information private and recovered back very fast as the data memory is reduced by compression, when information is lost in recent days.

III. PROPOSED SCHEME

Android phones applications access and store sensitive information including Short Message Service (SMS) billing, bank account login credentials. Premium phone calls, e-mails, contacts and business information. Access to this information by attacker in new ways to steal sensitive data. Emails and contacts can be sync to Internet but apps are partially sync to Internet with AndroidPhone. Android “partially” support synchronization of information to Internet. By the help of Content Providers as discussed below we can ensure security mechanism in Android and problems can be easily solved.

A. Content Provider

Content Providers manage access to a structured set of data, and provide mechanisms for defining data security. Content providers are the standard interfaces that connect data in one process with code running in another process. Android allows us to expose data sources through a REST (Representational State Transfer) like abstraction called content provider which stores and retrieves data and make it accessible to all applications. Android itself includes content providers that manage data such as audio, video, images and personal contacts information. You can see some of them listed in the reference documents for the android. Provider. Package [5].

However, there some restrictions that content providers do not store special data like application settings in Android. With some restrictions content providers are accessible to any Android applications.

B. Understanding Security on Android[6]

Android consists of an application framework, application libraries and Dalvik virtual machine-based runtime all running on top of a Linux kernel. By taking advantage of Linux kernel, Android gets a number of operating system services, including the management of processes and memory, a network stack, divers, a hardware abstraction layer and security issues. Android uses Linux facilities such as process-level security, user and group IDs that are associated with the application, and permissions to enforce what operations an application allowed to perform. On Android user ID identifies an application. User IDs are assigned when the application is installed, and they remain for the life span of the application on the device.

Android offers several security mechanisms, which provide service to make it as a secure system. In the following section we discuss deeper in the security mechanisms which are incorporated into Android framework.

1) **Application Signing**

All Android applications (.apk files) must be signed with a certificate whose private key is help by their developer[7]. This certificate identifies the author of the application. The purpose of certificates in Android is to distinguish application authors. This allows system to grant or deny applications access to signature-level permission.

2) **User IDs and File Access**

Android applications run on their own Linux process and are assigned with unique user ID. By default application run inside a basic sandbox process with no permissions assigned, thus preventing such applications from accessing the system resources. Android application can request the permissions, however through the application manifest file. Android application can allow access resources to other application by declaring the appropriate manifest permissions and running in the same process with other trusted application, thus sharing access to their data and code. Different application can run in the same process by you must first sign and using same private key, you must also assign same Linux user ID using manifest file.

Any data stored by an application will be assigned that application’s user ID, and normally accessible to other packages. Files in Android are subject to the Linux permissions.

3) **Using Permissions**

By default a basic android application has no permissions, so it cannot do anything that would adversely impact the user experience or any data on the device. You must include AndroidManifest.xml or more than one <user-permission>tags declaring the permissions that your application needs. At application install time, permissions requested by the application are granted to it by the installer, based on checks against the signatures of the applications declaring those permissions and/or interaction with the user.

No checks with the user are done while an application is running. The permissions provided by the Android system can be found at Mnifest. permission. Any application may also define and enforce its own permission, so this is not a comprehensive list of all possible permissions.

C. DATA COMPRESSION TECHNIQUES

In order to improve security in Android synchronization, first we compress the data using different compression techniques and then we sync the data from phone to Internet. This compression techniques encodes the data and compress the data, so necessary decode technique is used. The compression techniques reduce memory space and improve the transmission speed and also protect the data from attacker. This paper provides a way for enhancing security on application data before it is synchronized to Internet. There many compression techniques out of these the most efficient are Lempel-Ziv[8] and Huffman encoding[9], that are discussed below.

1) **Lempel-Ziv universal data compression**

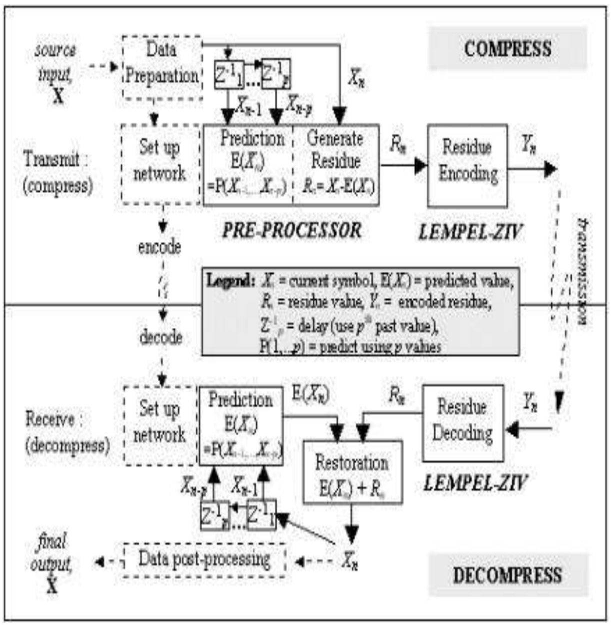
The Lempel-Ziv data compression algorithm uses variable-to-variable length code, in which both the number of source symbols encoded and the number of encoded bits per code word are variable. They do not require prior knowledge of the source statistics, yet over tine they adapt so that the average code word length L per source letter is minimized. Such algorithm is called universal. They have been widely used in practice and provide simple approach to understand universal data compression.

LZ77 is a lossless compression algorithm, which means that if you compress a document using the algorithm, and then decompress the compressed version, the result will be an exact copy of the original document. If variable –width codes are used, and then encoder and decoder must be careful to change the width at same point the encoded data. The encoder increase width from p to p+1 when sequence w+s are encountered, where w is window size, s is sequence and p is bits. Figure1 shows both encoding and decoding the data with Lempel-Ziv.

2) **Huffman coding**

The basic idea in Huffman coding is to assign short code words to those input blocks with high probabilities and long code words to those with low probabilities. A Huffman code is designed by merging together the two least probable characters, repeating this process until there is only one character remaining. A code tree is thus generated and Huffman code is obtained from labeling of the code tree. It is a lossless data compression technique widely used.

Huffman codes are not unique and are optimal in the sense that no other lossless fixed-to-variable length code has a lower average rate.


COMPRESS
source input, X
Data Preparation
Z^{-1}_1 ... Z^{-1}_p
X_n
X_{n-1}
X_{n-p}
Transmit : (compress)
Set up network
Prediction $E(X_n)$ $=P(X_{n-1}...X_{n-p})$
Generate Residue $R_n = X_n - E(X_n)$
R_n
Residue Encoding
Y_n
PRE-PROCESSOR
LEMPEL-ZIV
encode
transmission
Legend: X_n = current symbol, $E(X_n)$ = predicted value, R_n = residue value, Y_n = encoded residue, Z^{-1}_p = delay (use p^{th} past value), $P(1,..p)$ = predict using p values
decode
Receive : (decompress)
Set up network
Prediction $E(X_n)$ $=P(X_{n-1}...X_{n-p})$
$E(X_n)$
R_n
Residue Decoding
Y_n
Restoration $E(X_n) + R_n$
LEMPEL-ZIV
X_{n-p}
X_{n-1}
Z^{-1}_p ... Z^{-1}_1
final output, X
Data post-processing
X_n
DECOMPRESS

. figure1:Compress and decompress of data with Lempel-Ziv

A Huffman encoder takes block of input characters with fixed length and produces a block of output bits of variable length. It is fixed-to-variable length code. The figure2 shows final static code tree.

The table 1 is comparison between Huffman and Lempel-Ziv for respective files.

	Adaptive Huffman	Lempel-Ziv
Latex file	66%	44%
Speech file	65%	64%
Image file	94%	88%
Size of compressed file as percentage of the original file		

Table 1: difference between Huffman and Lempel-Ziv

Huffman coding does not matter how the characters are arranged and does not matter how the final code tree are labeled. This reduces the memory spaces and improves the security. After compression we perform synchronization of data to Internet.

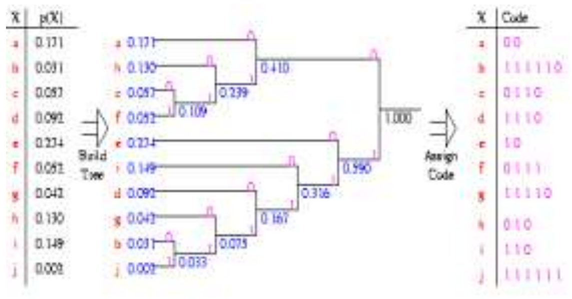


figure2: final static code tree by Huffman code.

D. SYNCHRONIZATION OPERATION

After compressing the application data we synchronize the data to the Internet. As android operating system consists of several layers the Linux kernel, Android runtime, c/c++ libraries, application framework and the applications. We could add an application which stores other application settings and synchronize to the Internet.

it uses SQL lite database to develop an application which stores the other application setting. As we discussed above that for every application has a User ID and application name. So we create a table which has columns as App_ID, APP_Name, Created Date and App_Settings [10]. It must be limited to a proper size to prevent some applications storing too large data into SQL lite database, which will slow down the whole application.

The App which consists of all the information of other application settings in compressed by using the above data compression techniques and then it is synchronized to the Internet. Each application in Android are compressed based on the above compression techniques and then an application is created which stores the compressed data in the SQL lite database which prevents from storing large data and slowing down the App.

We can use Google accounts to authenticate users. When App needs to be synchronized to the Internet it will send a request to server first. Then server returns with a result by verification of user identity. If it matches then user can synchronize the required App to the Internet. If it does not match then synchronization fails.

The figure3 shows how the applications on Android phone are compressed and then they are stored in SQL database as App based on categories like Business information, personal contacts, e-mail and other important information. This Apps are synchronized to the internet with the help of storage sever and Google server

IV. PERFORMANCE ANALYSIS

The analytical framework presented in previous section can be used to find the performance of the android mobile. As size of data is reduced by data compression algorithms these helps in fast transmission of data to the goggle server. And the transmission time is reduced when compared without applying data compression techniques to the application data. And also as size of data is reduced hence it is named as reduced transmission. The security is increased because the data is compressed using compression techniques and authenticity is also maintained with the Google server in transmission of application data.

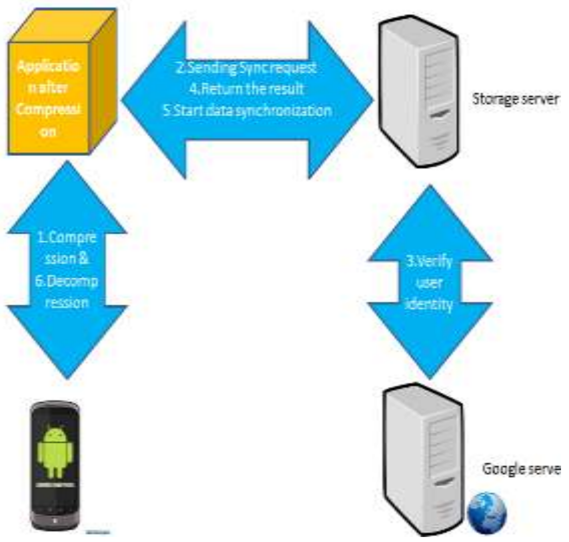


Figure 3: compression and synchronization operation

The second approach management of the exchange protocol during the transmission of application data with server also helps in reduction of cost and times required to connect with server whenever we upgrade the mobile. So the exchange protocol uses FTP4 which mainly helps in faster connection and speed transmission of application data with server.

V. CONCLUSION

This paper analyses on improving the user information security and synchronizing to the Internet. By use of compression techniques we reduce the storage space in database and compression techniques also improves security. The enhancement can be done in synchronizing data to internet by having 802.11ac Wi-Fi which provide high throughputs and this will sync in few seconds. The speed of 802.11ac will take hassle out of backing up mobile phones. Users can have peace of mind knowing that they will always have access to their phone data, even if phone itself is no longer available.

REFERENCES

[1] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security", IEEE Security & Privacy, vol. 7, no. 1, pp. 50–57, 2009.

[2] Andrew Kameka , "Android has 150k apps, 350k daily activations, and more notes from

Eric Schmidt’s MWC keynote", Androinica.com, March 2011.

[3] Asaf Shabtai, Yuval Fledel, Uri anonov, Yuval Elovici and Shlomi Dolev, "Google Android: A State-of-the-Art Review of security Mechanisms", arXiv: 0912.5101v1, 2009, pp. 1-2.

[4] SHABTAI, A., FLEDEL, Y., ANONOV, U., ELOVICI, Y.,DOLEV, S., AND GLEZER, C., "Google Android: AComprehensive Security Assessment", IEEE Security & Privacy, vol.9, no. 2, 2010, pp. 35 - 44.

[5] Sayed Hashimi, Satya Komatineni, and Dave MacLean, Pro Android2, Apress, USA, 2010, pp. 76-77.

[6] Google Inc., "Content Providers",android.com, Apr 2011.

[7] Google Inc., "Security and Permissions",android.com, Apr 2011

[8] http://www.datacompression.com/lossless.html, 2011

[9] http://www.datacompression.com/lossless.html, 2011

[10] Changhao Ai and Je Liu, Chaunxio Fan, Xiaoying Zhang and Junwei Zou “Enhancing Personal Information Security on Android with a new Synchronization scheme”, IEEE 2011