Scholars Journal of Engineering and Technology (SJET)
Sch. J. Eng. Tech., 2017; 5(9):508-512
©Scholars Academic and Scientific Publisher
(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

ISSN 2321-435X (Online)
ISSN 2347-9523 (Print)

Time Based Random Encryption for Inter apps Communication in Enterprise Mobile Application

M.M.F. Naja[1], M.I.I. Mohamed[2]
[1]Department of Information and Communication Technology, South Eastern University of Sri Lanka Oluvil, Sri Lanka
[2]IFS R&D Private Limited, Colombo 06, Sri Lamka

***Corresponding author**
M.I.I. Mohamed

Article History
Received: 03.09.2017
Accepted: 09.09.2017
Published: 30.09.2017

DOI:
10.21276/sjet.2017.5.9.10

Abstract: The penetration of smartphones within enterprises is rapidly increasing, so as the usage of it. Nowadays, Android applications are used widely in enterprise software to support their system. Software systems like SAP and IFS has their own mobile apps as a part of their software solutions, especially in field service domain. These enterprise mobile applications are not usually one single application. Rather it might be several apps that are inter related and communicate with each other. A number of ways are available which aids inter application communication. One way of such inter app communication is using "intents". Intents are simple message objects that are used to communicate from one activity to another. Intents could be also defined as the intention of an application. Although the intents are used in inter apps communication, the problem with intents is that they are not secure. Even Android Operating System doesn't claim that these intents are secure. Therefore our research aims at finding how intents messages could be encrypted for enterprise systems considering high security, light weight ness and fast processing such that it would not be easy to crack. Throughout this research we are proposing a new algorithm called Time Based Random Encryption Key algorithm which would be a most efficient encryption algorithm for the inter apps communication in Enterprise Mobile Apps.

Keywords: android security, intent message passing, enterprise application inter communication

INTRODUCTION

Enterprise mobile security is one of the topics with prime importance in today's industry. Due to the technical advancements in mobile industry, it has resulted in the increasing of security threats and attacks.

Usually Enterprise Applications use several apps on smartphone devices that work together to achieve a single call. For example, think about an enterprise application for stock management, which was implemented to scan barcodes and send the data to the online service, but on the other hand the same data can be processed by another application in the device which was developed by the same vendor [1].

Above mentioned implementation is a best practice when it comes to 'Component Based' development that will give least coupling and strong cohesion. So, both apps are not depending on each other, both of them were implemented to do their own functionality but still both are inter-operable [2].

Android Operation system provides such possibility through a concept called Broadcast receivers, which allows an application to respond to messages which is an Android Intent that is broadcasted by either Android OS of any other app. So, as we mentioned above, an application can listen to a message intent) which is totally not related to it. This is a well-known approach in Android development.

Problem in this approach is, these messages are listenable by any other application, and thus there is a risk of security breach when it comes to Enterprise level applications. Also these messages can be manipulated by other processes in a man in the middle manner [3].

For example, think of app A and app B from an enterprise software vendor where app A broadcasts an intent which is received and processed by app B, also it does mean any other apps, know the intent action from app A can receive the message and alter it and broadcast it back to the system with the same intent action identification, later app B will receive the manipulated message and start processing it. Knowing intent action is a very simple procedure, it's all about get the .apk file and reverse engineer it.

In this research paper, we are going to discuss our finding based on our research work on intent message encryption. Usually encryption and decryption is an overhead but when it comes to traditional enterprise applications we can eliminate the overhead time. Since IoT is emerging with Big data and real-time processing, huge amount of data encryption/decryption will also make the entire system meaningless, since we need data to be processed on time. So we will be discussing on how this encryption and decryption can be done very strongly and also with less time overhead.

INTER APPLICATION COMMUNICATION IN ANDROID

A. **Intents**

Android provides a sophisticated message passing system, in which Intents are used to link applications [2]. Intent is a message that declares a recipient and optionally includes data; An Intent can be thought of as a standalone object that specifies a remote procedure to invoke and includes the associated arguments. The applications use Intentions for both communication between applications and for intra-application communication. In addition, the operating system sends Intents to applications as event notifications. Some of these event notifications are system-wide events that can only be sent by the operating system. We call these messages the attempts to spread the system. Attempts can be used for explicit or implicit communication. An explicit intent specifies that it must be delivered to a specific application specified by the attempt, while an implicit attempt requests delivery to any application that supports a desired operation. In other words, an explicit attempt identifies the recipient by its name, while an implicit attempt leaves it to the Android platform to determine which application (s) should receive the attempt. For example, consider an application that stores contact information. When the user clicks on the address of a contact, the contacts application must ask another application to display a map of that location. To do this, the contacts application could send an explicit attempt directly to Google Maps or send an implicit attempt to be delivered to any application that indicates that it provides mapping functionality [4] (for example, Yahoo! Maps or Bing Maps). The use of an explicit intent ensures that the intent is delivered to the intended recipient, while the implicit attempts allow late-execution binding between different applications.

B. **Android Components**

Intents are delivered to the building blocks of logical application which are said to be the components of Android. There are four types of components which are defined by Android.

Intent
Activity
Service
Broadcast Receiver
Content Provider

Fig-1: Android Components

Activity provides an interface for users to interact with the application and perform an action (for example: Login to a website). The different screens / windows of an application are the different activities. Usually, an application has multiple activities. Activities are like pages on a website. For example, in a Facebook application, the login screen is an activity and the news of your friends after logging in will be different.

A service is a component that runs in the background to perform long-term operations or to perform jobs in remote processes. A service does not provide a user interface. For example, a service can play music in the background while the user is in a different application or can get data through the network without blocking user interaction with an activity. Another component, as an activity, can start the service and let it run or link with it to interact with it.

A service is implemented as a subclass of Service and you can obtain more information about it in the Services Developers Guide [5].

A content provider manages a shared set of application data. You can store the data in the file system, in a SQLite database, on the Web, or in any other permanent storage location that your application can access. Through the content provider, other applications can query or even modify the data (if the content provider allows it). For example, the Android system provides a content provider that manages the contact information of the user. Therefore, any application that has the appropriate permissions can query a part of the content provider (such as ContactsContract.Data) to read and write information about a particular person.

Content providers are also useful for reading and writing private data for your application and not shared. For example, the sample application Note Pad uses a content provider to save notes [6].

A content provider is implemented as a ContentProvider subclass and must implement a standard set of APIs that allow other applications to perform transactions

A broadcast receiver is a component that responds to system-wide broadcast prompts. Many programs come from the system - for example, a program that announces that the screen is off, the battery is empty, or a picture has been taken. Applications can also trigger broadcasts - for example, to allow other applications to know that some data has been downloaded to and available for the device. Although broadcast receivers do not display a user interface, they can create a status bar notification to notify the user when a broadcast event occurs. More frequently, however, a radio receiver is only a "gate" to other components and is said to perform a very small amount of work [4]. For example, it could initiate a service to performsome work based on the event.

METHODOLOGY

Usually intents are broadcasted via the following call.

```
// Create the text message with a string

Intent sendIntent = new Intent();

sendIntent.setAction(“com.ApplicationPackage.TEST”)
;

sendIntent.putExtra(“companyId”, “199898”);

sendIntent.setType("text/plain");
```

Above code was implemented to broadcast a text with value 199898 to the system and could be received by a broadcast received with intent filter “com.ApplicationPackage.TEST”.

Now our approach is to encrypt the message using a key that is generated in parallel to the time.

A. Encryption

Intents are not encrypted from Android system; they are readable by the Android OS itself or other applications. So our first goal is to encrypt the message that is being passed via the intent. Our encryption algorithm is based on four steps:

1) Get a token
    a. If login supported system – Get a token from login mechanism
    b. If non-login system – Get any data from the device and tokenize it.
2) Encrypt the token obtained in Step (1) against current timestamp and get the Encrypted token.
3) Encrypt the intent communication message against the encrypted token obtained in Step (2).
4) Broadcast the Intent with the encrypted message obtained in Step (3).

Thus, the encryption token, which will be calculated in Step (2) of the algorithm dependents on two factors.

1. Current time stamp.
2. A unique token specific to user.

When it comes to timestamp, we can use Epoch timestamp of Unix systems, which is a string with 10 chars. Using a timestamp that supports milliseconds can be potentially make this system fail due to delays in the intent communication. If the message failed to be received in a given time, it will become useless.

The token can be calculated form the login mechanism. Login mechanisms like OAuth and Active Directory provide a user specific token which can be stored and usually being stored in AppSettings. However, if system does not provide login functionality, even though they are less secure, this methodology could be applied via any static string that is shared among the packages from the application vendor. In this case, the token needs to be a package specific one and we should not use device specific tokens such as IMEI, MAC address, etc. Because, it could be determined by attackers very easily.

Encryption can be done with the following formula where the user token is the text to be encrypted against the timestamp key. Any existing encryption methodology can be used for the process.

EnceryptedTokenPerTime = Encrypt (user token, timestamp)

Now the message should be encrypted that is being broadcasted with the generated encrypted key.

EncryptedMessage=Encrypt(Message,EnceryptedTokenPerTime)

So the encrypted message will give you digests per time. Now the message could be broadcasted.

B. Decryption

Form the listener app B, again EncryptedToken needs to be generated. This will be same as the EncryptedToken by app A since user token is same and can be generated in same time. Now the message could be decrypted using,

Message = Decrypt (EncryptedMessage, EnceryptedTokenPerTime)

It is a must to use same encryption methodology in the entire process. Also it has to be a symmetric one.

C. Gaining performance in the Encryption and Decryption process

Since this the research work is targeting towards enterprise applications with real-time message passing it is really important to consider the encryption time overhead as well. Also we have notice that the length of encryption keys contributes to the time.

Since the key to obtain EncryptedTokenPerTime is fixed to 10 char size, we have experimented using AES encryption and found the below time taken to compute the algorithm against varying length of user tokens. The device used in this experiment featured is OnePlus 3 which has Snapdragon 820.

Table 1: Time Takento Compute Encrypted User Token Changes Against User Token String Length

User Token Size	Time taken in milliseconds
8	0.0678
9	0.0686
10	0.0694
11	0.0702
12	0.0711
13	0.0718
14	0.0726
15	0.0735
16	0.0742
17	0.0751
18	0.0757

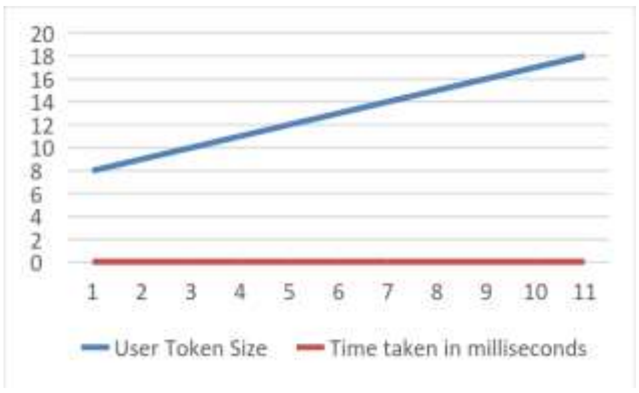

Fig-2: A graph showing how Time taken to compute Encrypted User Token changes against User Token string length

This chart shows us clearly that the time is growing uniformly against the length of token size. Also since we need to consider the time to encrypt the intent message which will be larger, we need to choose an optimal token size.

We need to maintain the balance between secure system and a system that can perform both encryption and decryption with in one second. Thus, we can choose 10 as the optimal char size for the user token. So we will get an encrypted token with char size of 24.

D. Avoid mal-formed messages

Our algorithm provides an encrypted digest that is valid only for one second. So the communication needed to be initiated, processed and completed before a new second is initiated from the system.

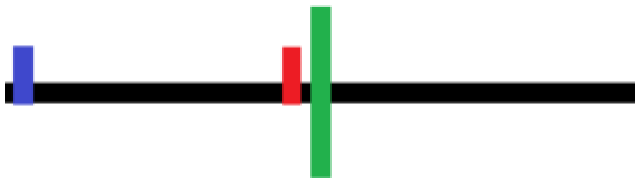

Fig-3: Timeline of the process, showing when process is being initiated and completed with the indication of new second

In Fig2, the process initiation is denoted with blue bar and the completion of entire process (encryption, decryption and successful intent receive event) is denoted by red bar, while the green bar indicated the new second. The decryption has to be done before the new second is initiated, else, the intent communication will give a mal-formed message, which is not meaningful.

To avoid this, the intent communication needs to be initiated at the beginning of a given second [5]. So the thread needs to wait until it hits the beginning of a new second. Following code snippet can be used for this purpose.

```
longwaitTime = 1000-System.currentTimeMillis() % 1000;
newjava.util.Timer().schedule(
newjava.util.TimerTask() {
@Override
public void run() {
// Call the Intent.
}
},
waitTime
);
```

CONCLUSIONS

The emergence of mobile applications in enterprise has increased a lot during the past years. Normally, these types of enterprise mobile applications are a collective number of applications that communicate with each other and sometimes depend on each other. A very common issue in the inter application communication is the security problems that arise during intents message passing, which is one mode of inter application communication. Due to this reason, the message passing mode needs to be much secure since sensitive data are transmitted during the inter application communication in mobile enterprise applications. This could be achieved by implementing encryption techniques during message passing. Therefore, for this purpose, we are proposing a time based encryption algorithm which could ensure secure inter application communication. Since this the research work is targeting towards enterprise applications with real-time message passing it is really important to consider the encryption time overhead as well. Also we have notice that the length of encryption keys contributes to the time.

REFERENCES

1. Chen H, Pan L. Android Application Visual Safety Analysis Based On Component Relations, International Journal of Computer Science and Artificial Intelligence. 2015; 5(1): 26-32.
2. Zhang M. Identifying and Analyzing Security Risks in Android Application Components, International Journal of Security and Its Applications 2016; 10(9): 165-174.
3. Komninos N, Vergados D, Douligeris C. Authentication in a layered security approach for mobile ad hoc networks, Computers & Security. 2007; 26(5): 373-380.
4. Liu Z, Han D. Dynamic Encryption Algorithm Based on Rijndael, Advanced Materials Research. 2012; 490-495; 339-342.
5. Jo M, Shin J. Study on Security Vulnerabilities of Implicit Intents in Android, Journal of the Korea Institute of Information Security and Cryptology. 2014; 24(6): 1175-1184.
6. Zhang M. Identifying and Analyzing Security Risks in Android Application Components, International Journal of Security and Its Applications. 2016; 10(9): 165-174.