Security Mechanisms for Android Devices to Protect Application Data and Provide Information about Intrusion Attempts in Real Time

Rohit Joshi
Department of Information Technology,
V.E.S. Institute of Technology,
Chembur, Mumbai. India.

Arjun Bhadra
Department of Information Technology,
V.E.S. Institute of Technology,
Chembur, Mumbai. India.

Adwait Vyas
Department of Information Technology,
V.E.S. Institute of Technology,
Chembur, Mumbai. India.

ABSTRACT
The amount of data being stored on Mobile Devices, particularly Android Devices, is increasing rapidly. Using passcodes to lock applications solves this problem to some extent but, this only protects the data and does not provide any information about the intruder's identity. This paper explores existing technologies that can be used to store the identity of the intruder and inform the users about intrusion in real time.

General Terms
Computer Security, Mobile Security, Data Security

Keywords
Android Application, Intruder Identification, Intrusion Notification

1. INTRODUCTION
Passcodes such as passwords, patterns etc. are the most popular way of implementing Authorization in Computer Systems. Authorization is used for Access Control [1] in Computer Systems. Traditional passcode locks have been used on smartphone devices since their inception. Since then, many loopholes and exploits in these have been discovered [2]. Although, most of the cracking attempts go unsuccessful, as the users are unaware of these attempts and identity of intruders, there is no decrease in number of attempts made on cracking passcodes. In turn, the number of passcodes cracked has not decreased. Thus, with maturity of the password model [3] on smartphones, there is a need to strengthen this model, in order to increase the level of security provided by the model. A key concept which can be used to make this model more secure are real time notifications, which can inform user about a hacking attempt in real time, allowing them to take appropriate actions, storing the identity of intruders, which will allow users to prevent further hacking attempt additional safety and convenience features, which will result in increased use of passcodes, thus increasing the security as a whole on smartphone devices. This paper coalesces some of the most efficient and widely used technologies with the password model which produces a convenient and highly secured password model than the traditional one.

2. ADD-ON FEATURES
The password model is well secured and resource efficient. Some more secured models like biometric identity are not yet feasible to be implemented on smartphone devices due to resource constrained environment, as these models consume a large amount of resources and take a lot of time for processing. Thus, instead of replacing the password model, some features can be added to the existing password model to provide the additional security features and at the same time keep the model time and resource efficient. These features are independent and efficient and can be implemented in any possible combination. Some of them may not support all devices, due to hardware and data rate restrictions, but most of them are feasible even on low-end smartphones.

3. CONVENIENCE & SECURITY FEATURES
A good security model should ultimately provide high level of security against a wide range of attacks. Although, security is important, the quality of model also depends on user convenience. This section discusses some of the features that provide additional security and ease of operation.

3.1 Relock Policy
When locking applications, traditional application lockers prompt the user for passcode, every time the application is opened or resumed. Many applications require users to open and close them many times in a short period and single use. In such cases, entering a passcode again and again to get back to existing applications becomes inconvenient. Relock Policy is a convenience feature that allows user to unlock an application for a fixed time frame. The application remains unlocked for a fixed period of time, irrespective of whether the application is opened, closed or minimized. Following algorithm may be used for implementing Relock Policy.
Step 1: When Boolean RLock (Relock) is disabled, prompt user for a passcode every time application instance is launched in canvas (on opening, resuming and reopening application)
Step 2: When Boolean RLock (Relock) is enabled, prompt the user to select time period t (Relock Time).
Step 3: Once an application is unlocked with RLock enabled, start the clock and do not prompt the user for passcode on that application for period of t seconds.
Step 4: If the application is still open and time t has elapsed, prompt the user for passcode.
Step 5: If application is not opened and time t has elapsed, prompt the user for passcode when application is re-launched or resumed.
Step 6: End.

3.2 Auto Start on Reboot
This feature also increases the ease of use of the password model. The security mechanism can either be enabled or disabled. When a device reboots, this feature automatically

enables the security mechanism if it was enabled when the device was shut down. It avoids the need for user to enable security again and again after rebooting the devices. This becomes useful as many smartphone devices are unintentionally rebooted due to battery constraints, crashes etc. after which, sometimes user may forget to enable security again, which may result in loss of confidentiality of the data. Following algorithm may be used to implement this feature.

Step 1: Take permission to auto start the security application from Android system.
Step 2: Auto start the system after reboot and check if Boolean AutoStart.
Step 3: If Boolean AutoStart is disabled, don't apply the passcode security to applications. Go to Step 7.
Step 4: If Boolean AutoStart is enabled, check the previous state of the security.
Step 5: If previous state of the security was disabled, keep the security disabled. Go to step 7.
Step 6: If previous state of the security was enabled, enable the security.
Step 7: End.

3.3 Randomized Keypad

Many of the passcode systems use either numeric or pattern method as a passcode. In pattern method, it is difficult to determine a passcode by observing finger movement of user. But, patterns are set of non-repeatable dots, which allow only 389112 combinations, very less compared to other passcode methods, thus can be cracked significantly faster than numeric passcodes. Thus, although patterns are easier to remember, numeric passcodes provide better security. But one major flaw in numeric codes is that a person in close vicinity to the user can guess numeric passcode using finger movement of the user while entering the passcode, since same number exists at a certain place on the screen every time, and there are only 10 numbers. This is a high risk vulnerability when a smartphone is used in crowded places. This can be mitigated using Randomized Keypad which randomly shuffles the 10 numbers on the keypad thus, making passcode guessing using finger movement impossible. A good way to implement this feature would be to use the Collections.Shuffle() function in Random Class of java.util [4] package. An example of this is as follows

```
List<Integer> numbers = new
ArrayList<Integer>();
for(int i=0;i<10;i++) numbers.add(i);
Collections.shuffle(numbers);
for(int i=0;i<10;i++) System.out.println(numbers.get(i));
```

3.4 Location Lock, Remote Lock & Remote Device Reset

Location Lock will allow users to set a particular center point or "home" point. Users can select a radius around this home point. If a device moves out of this certain radius, some action can be taken such as sending an SMS or E-Mail and user can be alerted. This can be useful for increasing security as well as convenience. Many users do not want to use a passcode when they are at home but want to use it, when say they are at work. Location Lock can be used to set one's home inside the radius where the security application will not ask the user for passcode, but if the device is outside the radius, the user will be asked to enter the passcode. This can also be useful in case a device is stolen and taken out of the set radius. Location Lock can work in conjunction with Remote Lock and Remote Device Reset. A stolen device and be locked remotely using user's credentials and can even be reset to factory defaults using Remote Lock and Remote Device Reset. Many of the security software already provide these features, but they do not include Location Lock, which can increase the effectiveness of these features. This set of features may not work on some devices since they will require constant data connectivity to update GPS location of the device after certain intervals. But, when used, these features can act as a great anti-theft security systems. The Location API of Android OS [5] can be used to implement these features.

4. INTRUSION IDENTIFICATION FEATURES

Many of the intrusion attempts on passcode systems go undetected, as these systems do not identify the attacker nor provide any information about them to user. This makes it difficult for users to identify that an intrusion was attempted, thus making it difficult to predict or prevent further attempts.

4.1 Intruder Identification Manager

Intruder Identification Manager is a feature which mitigates repeated attacks by the same intruder by identify and storing information about intruder and previous attempts and making it available to the user. One of the main feature of Intrusion Identification Manager is to take a picture of intruder upon wrong passcode entry using the front facing camera of smartphone. This feature has become more feasible since most of the smartphones including many low end ones now feature a front facing camera. In its natural position, front camera captures an image which shows intruder's face in almost all the cases. These images can be stored in a private gallery of the passcode application and can be viewed by the user to collect information about previous attacks including intruder's identification, intrusion attempt time, date, location etc. Android OS Camera API [5] which allows an application to capture images can be used to implement this feature.

5. REAL TIME NOTIFICATIONS

Many attempts to intrude a secure system can be prevented if the users have real time knowledge of the intrusion. Many attempts go successful as users do not know about ongoing intrusion. If users can be alerted about intrusion in real time, many attempts will fail due to faster response, which will result in less time for cracking a passcode. Given enough time and resources, any secure system can be hacked. Thus by increasing response time, the probability that an intrusion attempt is successful will decrease.

5.1 Alert Tone

Playing an alert tone upon a number of wrong entries of passcode is a good way to alert user about ongoing intrusion when the device is away but in close vicinity of the user. The device simply plays a specific tone set by a user to indicate that there have been n number of wrong passcode entries, where n is set by the user. But, this may be problematic since some users may want their phone to be silent all the time. This can be handled by allowing users to easily disable such features.

5.2 SMS Alert

Another way of alerting user in real time about intrusion attempt is by using SMS feature. Since, most of the mobile devices use SIM Cards with an exception for a few tablet devices, SMS can be sent using most of these devices. User can simply feed the security application with a mobile number, which should be alerted when an intrusion attempt is made. But, sending SMS costs money. Thus, user can customize the number of wrong attempts after which SMS alert is sent or even turn them off if they wish, to save money.

Android OS provides Telephony API [5] which allows applications to send SMS. This API can be used for implementing SMS Alert feature.

5.3 E-Mail Notification

The best way to notify users about intrusion is using E-Mail. All the users may not check their E-Mails frequently thus, E-Mail notification will not exactly be real time, but the amount of information which can be provided by an E-Mail notification is exceeds that of SMS Alert. An E-Mail notification containing intruder's identity (image captured using Intruder Identification Manager), location of the device, time of intrusion etc. can be sent to the user which can even avoid intrusion on stolen devices. But, E-Mail notifications cannot work without data connectivity which is a major drawback of this feature. E-Mail Notification feature can be implemented using JavaMail API provided by Oracle [6] and Location API provided by Android OS.

6. CONCLUSION

This paper discusses various techniques which can be used to strengthen a passcode security application on mobile devices. A passcode security model, although very efficient, falls behind compared to level of security provided by other security applications such as biometrics. Use of some or all of these techniques with a passcode security application can significantly enhance the level of security provided by these applications while still being one of the most efficient ways to provide security on mobile devices at physical level. Although, use of all of these features may not be possible on some of the mobile devices due to OS restrictions and resource restrictions, many Operating Systems such as Android OS, Fire OS etc. provide all the permissions and APIs required to program these features with ease. Furthermore, the level of benefits provided by implementing these techniques outweighs some of the disadvantages, these techniques may have thus, making them extremely productive specially when used conjunction with each other.

7. REFERENCES

[1] Authorization, Access Control from Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Authorization http://en.wikipedia.org/wiki/Access_control

[2] "Username and Password – A dying security model" by HID Global http://www.2fa.com/downloads/collateral/hotd_username_password_wp_en.pdf

[3] "Password Security & Markov Models" by Markus Dürmuth, Horst Görtz Institute for IT-Security, Ruhr-University Bochum http://passwords12.at.ifi.uio.no/Markus_Due

[4] Java.util Package, Java.util.Random Class http://docs.oracle.com/javase/6/docs/api/java/util/Random.html http://docs.oracle.com/javase/7/docs/api/java/util/package-summary.html

[5] Android OS Location API, Android OS Camera API, Android Telephony API https://developer.android.com/google/play-services/location.html http://developer.android.com/reference/android/hardware/Camera.html http://developer.android.com/reference/android/telephony/package-summary.html

[6] JavaMail API http://www.oracle.com/technetwork/java/javamail/index.html