A SmartWatch-based Password Input Extension for Android

Bohao Gao, Qing Mu, Quanxin Zhang *, Yuanzhang Li, Yu‟an Tan
Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application,
School of Computer Science and Technology,
Beijing Institute of Technology,
Beijing, 100081, China,
E-mail: zhangqx@bit.edu.cn
* Corresponding author

Abstract—In recent years, there are many companies competing for innovation in the field of wearable smart devices. In this paper, we present a secure password input extension of smart watch that has not been implemented on any other smart watches yet. A SmartWatch extension module is designed and runs on both the SmartWatch2 and the Smartphone by researching and using SONY Smart Extension APIs. The password is entered on SONY SmartWatch2, so the malware running on the Smartphone cannot intercept the password using the hooks, because these hooks only works with the normal password input on the touch screen, and they cannot detect the password input action on the SmartWatch2. The remote-input password application runs well in the SmartWatch2 and also can be ported to other wearable smart devices if they provide the similar development kit like SONY SmartWatch2. This extension shows a new scenario for wearable smart devices besides message notification and health management.

Keywords- SmartWatch2; Smart Extension APIs; Android; password protectio; wearable

I. INTRODUCTION

Driven by industry giants such as Google and Samsung, wearable smart devices are transforming from conceptualization to commercial at an alarming rate. The current research leaders in the field of smart watches are Samsung company and Sony company. Samsung's Gear series smart watches carry Android system [1-3]. The Gear smart watch is a smart device that can be used completely independent. While Sony‟s SmartWatch2, as an embedded system wearable smart device, is centering on the Smartphones. Although Sony‟s engineers such as Marlin Liew developed a great many extension functions based on Sony Smart Extension APIs, the function of SmartWatch2 is not quite enough. The research for development framework and development method of extension function for SmartWatch2 is too lack overall. Meanwhile, Smartphones‟ security issues are increasingly receiving much more attention.

Therefore, in this paper we finally realize a password input application based on SmartWatch2 after researching and analyzing the development framework of SmartWatch2 and the code source of Sony Smart Extension APIs. Other than directly inputting password on the Smartphone, we could use merely SmartWatch2 to input password by means of this extension application. We dispose the process of inputting password on the SmartWatch2. This paper provides convenience for developing the function of SmartWatch2 further, and in the hope of forwarding the development and popularization for smart watches. Meanwhile, we are in the hope of providing a novel way for making advantage of wearable smart devices to safeguard smartphones‟ security.

II. SONY SMARTWATCH2 ARCHITECTURE

A. Core Component

The following section describes the applications that make up the Smart Accessories architecture – the host application, Smart Connect, and the Smart Extension application.

The SmartWatch2 communicate with the Android device using Bluetooth™. The Smart Connect and the host applications (described in the next paragraph) on the Android device handle all communication with the Smart Accessories. The relationship is shown in Fig .1.

The host application handles all interaction with the accessory. The host application for SmartWatch2 is an Android application names SmartWatch2. The host application uses the content providers in Smart Connect to find information about which Smart Extension apps (applications) should be available on the accessory. Smart extension apps using the Smart Extension APIs interact with the SmartWatch2 through the host application and Smart Connect.

Smart Connect is a framework for Android devices that manages applications and related settings for SmartWatch2. It is responsible for the connection between smartphone and SmartWatch2. The content providers for two of the Smart Extension APIs – the Notification API and the Registration and Capabilities API – are the most important part of Smart Connect. It works like a database [4-6]. Host application can get data (for example, notification needed to be sent) from it. Then it sends the data to Smart Extension app. Then Smart Extension app shows it on the SmartWatch2 in established forms or method.

As previously mentioned, the Smart Extension app is the app that you “extend” to work with Smart Accessory products. The difference to general Android app is Smart Extension app is based on the Smart Extension APIs. The Smart Extension APIs make it possible to communicate with the Smart Accessories.

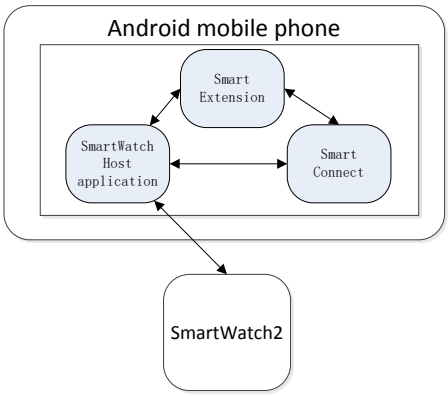


Figure 1. Architecture overview of the Smart Extension apps with Smart Accessories examples

B. *Smart Extesnsion API Architeture*

There is one host application for each Smart Accessory. The hardware unit communicates through Bluetooth™ with a host application that runs on the phone. The host application together with the accessory software controls what is shown on the hardware.

The Smart Extension application communicates with a host application through the Smart Extension APIs that are described in this section. A Smart Extension app is not tied to a certain accessory. Specifically, one extension that enables some data can provide this data to several different accessories. It is a host application"s task to render the data so that it matches the accessory"s capabilities.

C. *Smart Extension APIS Function*

Smart Extension APIs include five APIs, Some of the APIs are intent-based (Control, Widget, Sensor) and some are based on Content Providers (Notification and Registration & Capabilities). In this paper, the password input Extension app only needs the Control API and Registration & Capabilities API. They are discussed below.

The Control API enables the extension to take total control of the accessory. It takes control over the display, LEDs, vibrator, input events. Because of this, only one extension can run in this mode at a time. It is use to render data on device. It supports Grid view, List view, menu, image rendering, touch event and key press.

The Registration and Capabilities API defines and implements an Android ContentProvider that the app extensions can access via the Android ContentResolver API. Before an app extension can interact with an accessory it must provide (register) some information needed by the host applications [7-9]. It is also used by the Smart Extension app to inform the host application about which of the other APIs that are used. Typically host applications insert and maintain information about the accessories capabilities. Extensions use the capability information in order to interact with the accessories in a correct way. The ContentProvider implementation is backed by a database implementation.

III. DESIGN AND IMPLEMENTATION FOR THE EXTENSION

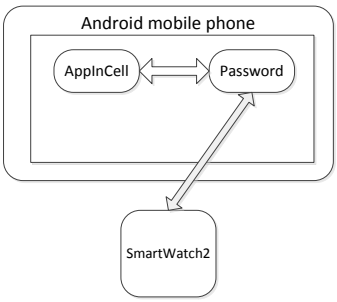
A. *Design Thoughts*

Generally, whether using the default keyboard or the custom keyboard in Android, Smartphones" input is inevitably monitored by hacker so that users" information could be stolen [10-12]. For this reason, the overall design idea for the password input extension is providing an approach that only uses SmartWatch2 to input password. We completely dispose the password input process on the SmartWatch2. It can prevent password being hacked to a certain extent.

The password input extension provides the password input keyboard for general Android apps which need special protection. In the meanwhile, it needs a general Android app to coordinate with the extension to achieve the effect. The structure relationship among the general Android app, the password input extension, and smartphone is shown in Fig .2.

Figure 2. Architecture overview of the extension app with general app

B. *UI(User Interface) Structure*



Too many characters on the screen of SmartWatch2 will render poorly display effect and touch effect at small sizes. Therefore, due to SmartWatch2"s screen size (1.6 inch) and security of the password, numeric characters are enough for password of extension [13-14]. Based on the above, the extension"s UI is designed to include ten numeric buttons, one delete button, one backspacing button and a password display box. It"s shown in Fig .3.

Figure 3. User interface

C. *Working Mechanism*

The password input extension"s overall implementation is disposing the password input process on the SmartWatch2. It"s safer and more convenience than inputting on Smartphones. Firstly, users input the preset four digit numeric password on the SmartWatch2. Then the extension obtains the password, and encapsulates it to an Intent. Then, the extension sends it to the apps in Smartphone by Broadcast. Particular apps can receive the Broadcast information and get the password from the

associated Intent. Then the app in Smartphone checks the password from Intent. The app"s activity will activate if the password is correct. If not, it will not activate and pop a toast that indicates the input password is incorrect. In the meanwhile, SmartWatch2 will vibrate if the Smartphone within striking distance. Note that if the input password is less than four digits, nothing will be done.

It"s essential to use Control API to display the password input extension app"s UI based on researching and study Smart Extensions" content. While, it"s with no need for Notification API, Sensor API and Widget API. The extension calls the function in ControlExtension class to display the ImageView widget declared in the XML file. The actuall role of these pictures widget is the password buttons. After user touches a button on the SmartWatch2, the extension will call function sendImage to change the source of these buttons and display box to realize like press a real button. Then the Handler class, playing an auxiliary role, refresh the button to restore the default state of the button by sendImage after 500ms.

D. Main Class

The extension names Password and the general Android app names AppInCell. AppInCell is the app that associates with the extension. It includes a subclass from BoradcastReceiver to receive the Intent by Broadcast from extension and a subclass from Activity to display application"s interface. Password includes four main classes. The class diagram is shown in Fig .4.

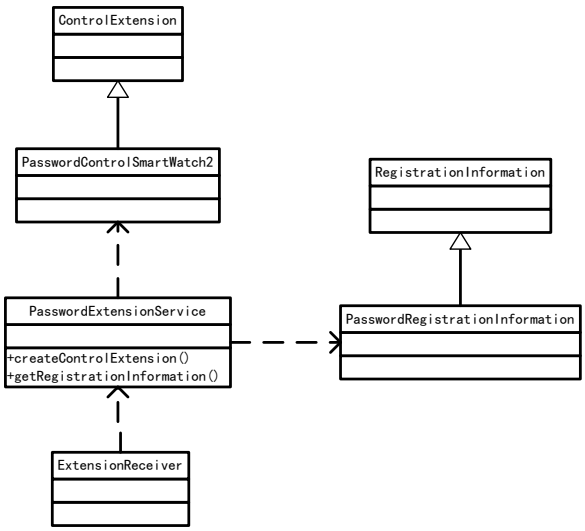


Figure 4. Class diagram

1) *PasswordControlSmartWatch2.java:* The core function portion. It is responsible for displaying the interface on the SmartWatch2 to input password and send the password to AppInCell.

2) *ExtensionReceiver.java:* It receives broadcast to activate ExtensionService.

3) *PasswordExtensionService.java:* The core logic portion. It is responsible for registering the Extension and creating a PasswordControlSmartWatch2 object.

4) *PasswordRegistrationInformation.java:* It provides the required registration information when registering the extension especially the information about which API is required.

The Sequence Diagram of the extension Password and general app AppInCell is shown in Fig .5. Three classes in the left of the diagram is belong to Password and others belong to AppInCell.

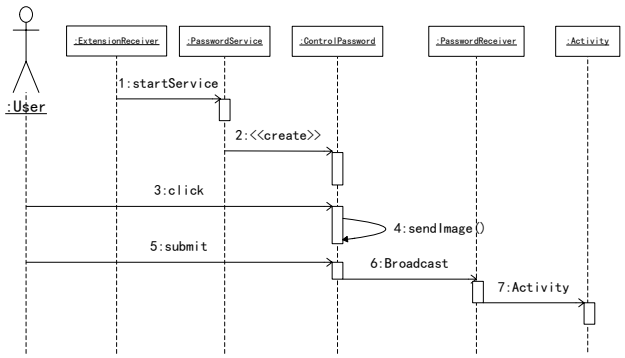


Figure 5. Sequence Diagram

IV. CONCRETE IMPLEMENTATION PROCESS

A. Development Environment

It"s important to note that there is not hard disk or flash in SmartWatch2 but RAM. To be clear, Smart Extension apps for SmartWatch2 are actually operating in smartphones as Android apps but not SmartWatch2. While, they are based on the Smart Extension APIs. SmartWatch2 displays, or render message and control smartphone by Smart Extension APIs. Every smart extension app render data by Bluetooth. Development for SmartWatch2 consists of three necessary part, Java environment, Android development environment and Sony Add-on SDK which is an add-on to the Android SDK that includes the Smart Extension APIs..

After building the general Android development environment, Sony Add-on SDK is extra to develop smart extensions. Firstly, two sample codes that should be imported are SmartExtensionAPI and SmartExtensionUtils. All other Smart Extension sample codes will need these library projects. These two sample codes should be set as lib package.

Moreover, it is obliged to add two right declarations for file AndroidManifest.xml as follow.

<uses-permission adroid:name="com.sonyericsson.extras.liveware.aef.EXTENSION_PERMISSIN"> </uses-permission>

<uses-permission adroid:name="com.sonyericsson.extras.liveware.aef.registra-tion.Registration.HOSTAPP_PERMISSION"> </uses-permission>

B. Concrete Implementation

- Start Eclipse and to create a new project names „Password".
- Add a Receiver widget and a Service widget in the file AndroidManifets.xml in the project. Then it needs an important self-defined right declaration „com.sonyericsson.extras.liveware.aef.EXTENSION_PERMISSION".

- Create file ExtensionReceiver.java, ExtensionReceiver class extends with BroadcastReceiver class. It works as bridge between SmartWatch2 and Host application. It will receive event generated and all different Broadcast information [9] in SmartWatch2 and forward it to PasswordService.java.
- Create file PasswordRegistration.java, PasswordRegistration class extends with RegistrationInformation class. As the name suggests, it is for registering relevant registration information for the extension. An object of PasswordRegistration class is created in PasswordService to register relevant in-formation to ContentProvider in Host application. Five main methods are overwritten in PasswordRegistration.java, among these four functions are used to declare required APIs and one methods used for extension registration. The function public ContentValues getExtensionRegistrationConfiguration(), is the most important function, is to provider registration information required to register the extension. The relevant information is shown in Fig .6, and the information bolded is indispensable.

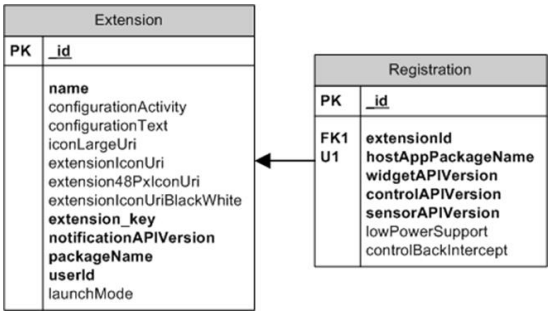


Figure 6. Registration information

- Create file PasswordService.java, PasswordService class extends with ExtensionService class and ExtensionService class extends with Service class. It contains main logic of the extension as the most important class. Three main abstract functions are implemented. Firstly, the function names getRegistrationInformation() which creates and sends object of PasswordRegistration class to finish registering the extension. Secondly, the function names keepRunningWhenConnected() return true to keep ExtensionService running as long as SmartWatch2 is connected with Smart Phone. Last, the function named public ControlExtension createControlExtension() which creates the widget to display interface.
- Create file ControlPassword.java, ControlPassword class extends with ControlExtension class. It is the most important class for the extension as the function implementation class. It implements to display the extension‟s interface on the SmartWatch2. In fact, it uses XML layout file to display the interface by function showlayout. We use function sendImage to change the source of picture to implement the effect of status switching for password buttons for the widgets (mainly ImageView) declared in the XML layout file. An important abstract function is implemented and a new function is defined. In the meanwhile, a thread class for auxiliary is defined. Function public void onObjectClick() is called when an object is clicked. Function private void setupClickables() creates an ordinary View object through XML layout file. Then the View object is transformed to a ControlViewGroup object. Function findViewById of ControlViewGroup returns ControlView object which is required to be used afterwards. Then it is set Click monitor. The password display box will be refreshed after click numeric button or rollback button. While, the Broadcast information includes password will be sent after send button is clicked. The thread class is help Handler class to implement the effect of delay display to implement the function of status switching for button.

V. RESULT AND TESTING EFFECT

A. *Key Problems*

There are some problems are noteworthy as follow in the process of research for the Smart Extension APIs and development and test for the extension.

- Two packages, SmartExtensionAPI and SmartExtensionUtils, are not added to Library property for the extension. It has to be set SmartExtensionAPI as a Lib for SmartExtensionUtils. These two package must be added in the property Library for the extension project „Properties>Android“ at first.
- The transmit rate of Bluetooth is needed to be concerned because the extension renders data from Smartphone to SmartWatch2 by Bluetooth. To guarantee the effect, the update frequency shouldn‟t be too large which means pictures transmitted in one minute should be limited to 20.
- In consideration of SmartWatch2‟s resolution (220 * 176) is small, XML layout file should adopt relative layout and absolute dimension „px“ when we designed the UI. Therefore, it has to add „‡tools:ignore="ContentDescription,PxUsage“ to layout attributes.

B. *Test Effect*

The test environment includes Samsung Smartphone Galax S3 within Android 4.1.2 and SmartWatch2.

After compiling and installing the extension, we use above devices to test. Firstly, AppInCell doesn‟t activate after inputting wrong password (password length is 4), and a tip popup in the Smartphone and SmartWatch2 vibrates. Secondly, AppInCell doesn‟t activate but the tip doesn‟t popup in the smartphone after inputting password which is not long enough. It indicates the password is not sent. Then we click rollback button, it implement the rollback effect. At last, AppInCell activates after inputting the correct password. Through experiments and research, we implement the function we need. The test effect is shown in Fig .7.

Figure 7. Test effect

VI. CONCLUSION

In this paper, we put emphasis on the Smart Extension APIs. We research and study principle and development framework for SmartWatch2. We present a novel method to input password by SmartWatch2 and develop an smart extension app for interaction between Smartphone and smart watch. Users can only use SmartWatch2 to activate a general Android app conveniently and safely. The password is more difficult to be stolen than directly inputting password on the Smartphone. The man who picks the lost or stolen Smartphone can"t input password without SmartWatch2. Even though the person have a SmartWatch2, the information encrypted by the extension wouldn"t be seen for the existence of the extension. Otherwise, the extension in this paper is relatively easy. We are in the hope for guide or promote the development for smart watches even wearable smart devices by this paper.

In the meanwhile, Sony Smart Extension APIs is well expansible as open source codes. It provides well development framework for developers. With development and maturing, the framework must will play a driving role for wearable smart devices to make wearable smart devices really in people"s daily life.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China under Grant No. 61300047 and 863 Program (No. 2013AA01A212), Students' Innovative Plan of BIT(201310007043), Shanghai Aerospace Science and Technology Fund(SAST201341), China Aerospace Science and Technology Fund(CAST).

REFERENCES

[1] X.A. Chen, T. Grossman, D.J. Wigdor, and G. Fitzmaurice, "Duet: exploring joint interactions on a smart phone and a smart watch," In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems , ACM, pp. 159-168, 2014.

[2] E. Chin, A.P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in Android," Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, ACM, pp. 239-252. 2011.

[3] A.P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," Proceedings of the Eighth Symposium on Usable Privacy and Security, ACM, p. 3, 2012.

[4] D.J Geng, Y. Suo, Y. Chen, J. Wen, and Y.J. Lu, "Design and implementation of Android phone based access and control in smart space," Journal of Computer Applications, Vol. 2, pp. 560–562, 2011.

[5] Y. Li., G. Feng, L. Li, and Y.H. Luo, "Development and Research on Multimedia Application Based on Android," Computer and Modernization, Vol. 4, pp. 149-150, 2011.

[6] C.C Liu, "Measuring and prioritizing value of mobile phone usage," International Journal of Mobile Communications, Vol. 8, No. 1, pp. 41–52, 2010

[7] R. Meier, "Professional Android 4 application development," John Wiley & Sons, 2012.

[8] R. Rogers, J. Lombardo, Z. Mednieks, and B. Meike, "Android application development: Programming with the Google SDK," O'Reilly Media, Inc., 2009.

[9] M.Q Song, W.K. Xiong, and X.L. Fu, "Research on Architecture of Multimedia and Its Design Based on Android," 2010 Int. Conf. Internet Technology and Applications, IEEE press, Aug. 2010, pp.1-4.

[10] X.H. Sun, and Z.X. Feng, "Interaction Design for Wearable Devices," Art & Design, Vol.2, pp. 28-33, 2014.

[11] M.K. Swamy, P.K. Reddy, R.U. Kiran, and M.V. Reddy, "Temporality-based user interface design approaches for desktop and small screen environment," International Journal of Computational Science and Engineering, Vol. 7, No. 1, pp. 52-64, 2012.

[12] C. Wang, W. Duan, J. Z. Ma, and C. H. Wang, "The research of Android system architecture and application programming," Computer Science and Network Technology (ICCSNT 11), IEEE Press, Dec. 2011, pp. 785-790

[13] Y.J. Zhou, X.W. Zhang, X.X. Jiang, and V.W. Freeh. "Taming information-stealing smartphone applications (on android)," Trust and Trustworthy Computing. Springer Berlin Heidelberg, June. 2011. pp. 93-107.

[14] Y. Gao, Q.X. Zhang, Y.L. Chu, X. He, J. Wan, Z.Q. Zhou, and J. Lin, "The research and implementation of customised launcher in Android," International Journal of Wireless and Mobile Computing, Vol.6, No.5, pp. 441-447, 2013