Keyboard or Keylogger?: a security analysis of third-party keyboards on Android

Junsung Cho, Geumhwan Cho and Hyoungshick Kim
Department of Computer Science and Engineering
Sungkyunkwan University, Republic of Korea
{js.cho, geumhwan, hyoung}@skku.edu

Abstract—**Use of third-party keyboards makes Android more flexible and customizable. However, we demonstrate their potential security risks by implementing a proof-of-concept keylogger that can effectively steal users' sensitive keystrokes with 81 popular websites (out of 100 tested websites). We also empirically analyzed the security behaviors of 139 keyboard applications that were available on Google Play. Our study results show that the majority of existing keyboard applications (84 out of 139) could be potentially misused as malicious keyloggers. To avoid such keylogging attacks, we discuss possible defense mechanisms.**

I. Introduction

Google allows third-party custom keyboard applications on Android. Therefore, users can freely change the default keyboard with third-party ones. However, installing third-party keyboard applications might yield a serious security problem. Recently, an Android developer showed a possible threat in using third-party keyboard applications; a legitimate third-party keyboard called "SwiftKey" can turn into a malicious keylogger by injecting code snippets.[1]

In this paper, we extend his work into a more controlled setting to explore in depth secure concerns raised by third-party keyboard applications. We implemented a proof-of-concept keylogger application to show the potential risk of third-party keyboard applications. Unfortunately, Google's existing application review process does not work effectively in reducing the risk of such keylogger applications. The key contributions of the paper can be summarized as follows:

- We implemented a proof-of-concept keylogger application that requires the `INTERNET` permission alone and tested its feasibility with 100 popular websites. Our experimental results show that user login data from 81 websites can be effectively captured by our keylogger application. Our keylogger application successfully passed Google's application review process within a few hours despite its dangerous functions for stealing users' keystrokes (see Section III).
- We investigated the security behaviors of the 139 third-party keyboard applications that were freely available on the Google Play store to analyze their potential risks. Fortunately, the existing applications now seem innocent but the majority of those applications with the `INTERNET` permission could be potentially misused as keyloggers (see Section IV).
- We suggest several practical recommendations to fix the security problem associated with third-party keyboard applications (see Section V).

[1]http://thehackernews.com/2013/03/android-swiftkey-keyboard-turned-into.html

II. Related Work

The user is the ultimate "client" of the system. Therefore, the user's input channel is frequently attacked by malware that tries to steal sensitive information such as user password. This type of malware is typically called keylogger.

Recently, keylogging issues on mobile devices have been studied [1]. Cai et al. [2] particularly introduced a new side channel attack which focuses on guessing a user's touched positions for typing a screen keyboard on the touch screen.

As for keylogging attacks using third-party keyboards, Mohsen et al. [3] introduced several possible attack scenarios. Their study results, however, may not be sufficient to show the real impacts of such attacks because their study mainly focused on analyzing existing keyboard applications' permissions information that could be misused for keylogging operations. In contrast, we demonstrated how such a keylogger application can be implemented through intensive tests with real websites.

III. How to implement a keylogger for Android

We demonstrate how to implement a proof-of-concept keylogger disguising itself as a legitimate keyboard application. Suppose that a user has installed our keylogger application on her Android device to use that application as a third-party custom keyboard. Figure 1 shows the overview of our system.

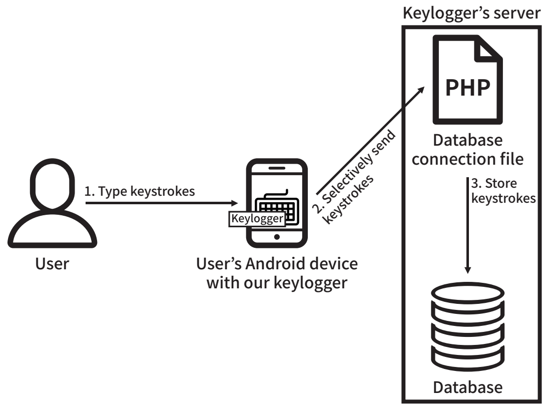


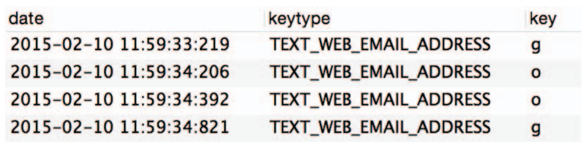
Fig. 1: Overview of our proof-of-concept keylogging system

TABLE I: Input types for text fields used in Android

No.	Input Type	No.	Input Type
1	TYPE_DATETIME_VARIATION_DATE	11	TYPE_TEXT_VARIATION_PASSWORD
2	TYPE_DATETIME_VARIATION_NORMAL	12	TYPE_TEXT_VARIATION_PERSON_NAME
3	TYPE_DATETIME_VARIATION_TIME	13	TYPE_TEXT_VARIATION_PHONETIC
4	TYPE_NUMBER_VARIATION_NORMAL	14	TYPE_TEXT_VARIATION_POSTAL_ADDRESS
5	TYPE_NUMBER_VARIATION_PASSWORD	15	TYPE_TEXT_VARIATION_SHORT_MESSAGE
6	TYPE_TEXT_VARIATION_EMAIL_ADDRESS	16	TYPE_TEXT_VARIATION_URI
7	TYPE_TEXT_VARIATION_EMAIL_SUBJECT	17	TYPE_TEXT_VARIATION_VISIBLE_PASSWORD
8	TYPE_TEXT_VARIATION_FILTER	18	TYPE_TEXT_VARIATION_WEB_EDIT_TEXT
9	TYPE_TEXT_VARIATION_LONG_MESSAGE	19	TYPE_TEXT_VARIATION_WEB_EMAIL_ADDRESS
10	TYPE_TEXT_VARIATION_NORMAL	20	TYPE_TEXT_VARIATION_WEB_PASSWORD

In the Android platform, the keylogging process can be particularly efficiently implemented; our keylogger application can selectively collect users' sensitive input keystrokes alone with the default feature provided by Android. When a user types a keystroke, our keylogger application can obtain the information about the input type for each text field in an application or a web page. That is, that application can check whether the typed keystrokes are likely to be valuable (or sensitive) with the text field information. For example, if a keylogger developer is only interested in harvesting passwords, the developer can collect the typed keystrokes in the only text fields associated with the password (e.g., TYPE_TEXT_VARIATION_PASSWORD). The full list of input types used in Android is shown in Table I.

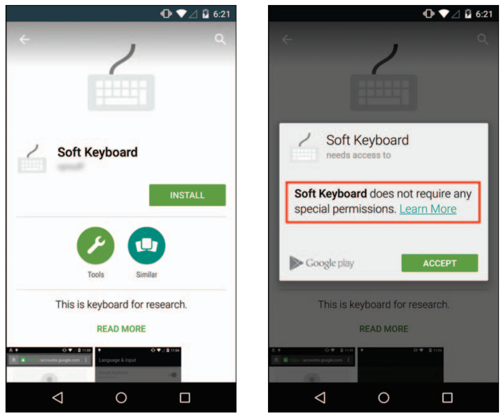
After capturing keystrokes, our keylogger application sends them to a (keylogger's) remote server using a PHP script to connect the keylogger application with a MySQL database in that server; INTERNET is the only permission needed to complete this process. Since a background thread can be run stealthily to perform the network delivery process, the victim user may not be aware of that activity. Figure 2 shows an example of keystroke values stored in our database. For simplification, our keylogger application delivers each character as typed. We tested this overall procedure with Google Nexus 5 which is running Android 5.0 and confirmed that our proof-of-concept keylogger application works well.



date	keytype	key
2015-02-10 11:59:33:219	TEXT_WEB_EMAIL_ADDRESS	g
2015-02-10 11:59:34:206	TEXT_WEB_EMAIL_ADDRESS	o
2015-02-10 11:59:34:392	TEXT_WEB_EMAIL_ADDRESS	o
2015-02-10 11:59:34:821	TEXT_WEB_EMAIL_ADDRESS	g

Fig. 2: Keystroke values stored in our database

We performed additional experiments to see how our keylogger application can also work well in real environments. We used the top 100 websites selected from the Alexa Top-500 Global Sites[2]. To test whether keylogging attacks can be successfully achieved, we used the Chrome web browser. For each website, we tested whether user typed keystrokes for login can be successfully stored in our database. From our test results, user login data can be successfully captured in 81 out of the tested websites; eight websites did not have a web page with login form; four websites were not connected; two websites only supported the *single-sign-on* authentication such as OAuth [4]. Interestingly, even for financial institutions including Bank of America and Chase Bank which might be high-risk targets, user login data can be successfully captured in the same manner.


6:21
Soft Keyboard
INSTALL
Tools
Similar
This is keyboard for research.
READ MORE
6:21
Soft Keyboard
needs access to
Soft Keyboard does not require any special permissions. Learn More
Google play
ACCEPT
This is keyboard for research.
READ MORE

(a) App information (b) Installation warning

Fig. 3: Ineffective warning messages on the Google Play store

When we submitted our proof-of-concept keylogger application to the Google Play store, that application successfully passed Google's application review process within several hours despite the functions that can steal sensitive inputs from users. Figure 3(a) shows that our proof-of-concept application was successfully registered in the Google Play store. To make matters worse, when a user tries to install that application, the Google Play store's warning messages about permissions requested by that application do not seem to be effective at all. Google has recently made changes to how permissions are displayed[3]; no warning messages were displayed about the INTERNET permission during the installation process when Android applications require that permission (see Figure 3(b)). If a user wants to check whether the INTERNET permission is requested by an application, the user must manually scroll down on the Google Play store application. Therefore, casual users may fail to recognize the potential risk of such applications and just continue the installation process.

Finally, we analyzed the effectiveness of antivirus scanners

[2]http://www.alexa.com/topsites

[3]https://support.google.com/googleplay/answer/6014972

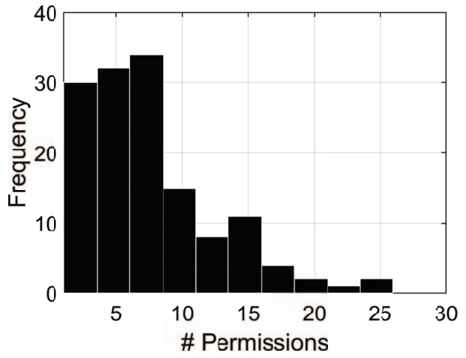


Fig. 4: Frequency of permissions from each thrid-party keyboard application

in detecting such keylogger applications. We used AndroTotal [5] which is a service to scan suspicious Android applications with multiple Android antivirus scanners. Unfortunately, all the eight scanners used in AndroTotal were not effective to detect our keylogger application; five scanners reported "no threat"; three scanners took "time out" in analyzing our application (see the test results in http://andrototal.org/scan/result/UNXhD78MSdiD6HG7K-txIg).

IV. Analysis of third-party keyboards from the Google Play Store

We explore potential risks associated with third-party keyboards in the Google Play store. We empirically analyzed whether those existing keyboard applications behave suspiciously or not by observing their characteristics. We downloaded all 139 third-party keyboard applications that were freely available on the Google Play store[4] and analyzed them individually. Our analysis particularly focused on two points: (1) numbers and types of requested permissions and (2) purposes of the `INTERNET` permission.

A. Numbers and Types of permissions

We used a tool named "aapt" (Android Asset Packaging Tool[5]) that is popularly used for extracting the Android manifest file from an application package (i.e., `APK` file) where the permissions required by the application are specified. Figure 4 shows the distribution of the number of permissions for the downloaded keyboard applications. From the histogram in Figure 4, we can see that those third-party keyboard applications require much more permissions than they might need.

To analyze the characteristics of those permissions, we also looked at the most commonly requested permissions. Figure 5 shows the list of the top 20 most commonly requested permissions and their distribution. We can see that some potentially dangerous permissions (`INTERNET`, `READ_CONTACTS`, `WRITE_EXTERNAL_STORAGE`, `RECORD_AUDIO` and `READ_PHONE_STATE`)[6] were popularly requested although they do not seem to be necessary to function properly.

[4]There were 250 third-party keyboard applications in total and 111 of them were paid applications.

[5]http://www.kandroid.org/guide/developing/tools/aapt.html

[6]Those permissions are also popularly requested by Android malware [6]

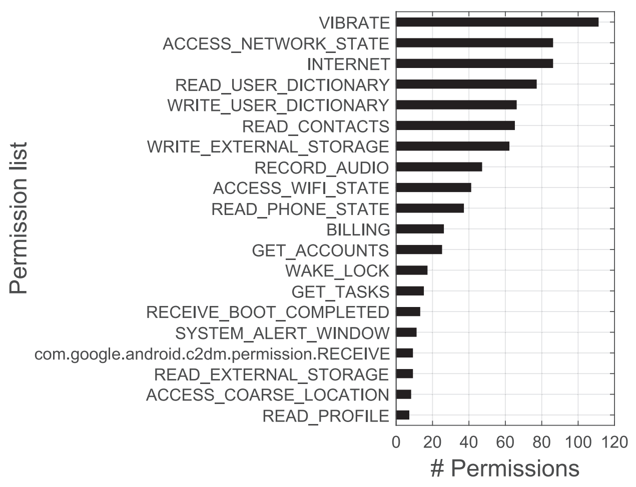


Fig. 5: Top 20 most used permissions in the 139 third-party keyboard applications downloaded from the Google Play store

B. Purposes of the `INTERNET` permission

As discussed in Section III, the `INTERNET` permission – requested by 61.8% (86 of 139) of keyboard applications – might particularly be dangerous since that permission can be misused for keylogging. We use Wireshark as a network sniffing tool to analyze network traffic for keyboard applications. We tested 84 applications in total since two applications were not working.

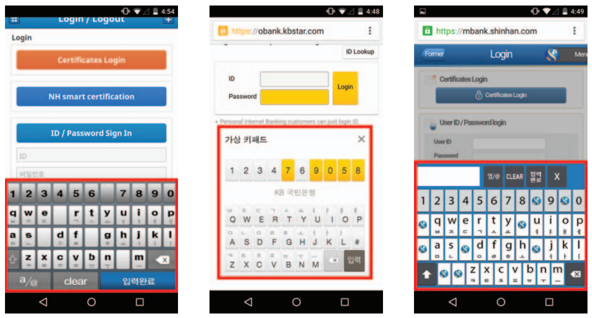
For each keyboard application, we recorded the network traffic generated by that application while trying to sign in at Gmail (http://www.gmail.com). Fortunately, in our experiments, all the tested applications seem innocent; only 7.9% (11 out of 139) of those applications generated network traffic but did not deliver user login data to a suspicious host. We think that the `INTERNET` permission is generally used to check the application version. However, some applications could be a time or logic bomb [7] that we did not detect and in theory, those 84 applications (out of 139) with the `INTERNET` permission can always be transformed to keyloggers.

V. Countermeasures

We discuss several mechanisms possible to mitigate keyloggers as described in Section III.

A. Preventing data leakage via information flow tracking

To prevent third-party keyboard applications' keylogging behaviors, we can use fine-grained information flow tracking by monitoring how the typed information from a keyboard application propagates within a system. That is, an information flow tracking system can warn the user when a keyboard application delivers sensitive information (e.g., user password) to unknown external servers. Previous studies [8], [9] proposed a possible solution based on dynamic analysis for realtime privacy monitoring on smartphones. Such a technique could be easily applied to mitigate the keyloggers presented in Section III. However, this approach typically imposes a significant runtime overhead because all information flow operations are monitored. In order to avoid the runtime overhead, static analysis tools (e.g., [10]) were recently introduced.



(a) Nonghyup Bank (b) KBstar Bank (c) Shinhan Bank

Fig. 6: Trustworthy keyboards in bank websites

B. Use of trustworthy keyboards

For critical applications or situations (e.g., user login), the use of a trustworthy keyboard can only be enforced to reduce the risk of using a suspicious keyboard application. For example, when a user types her password to login to a bank website, a predefined trustworthy keyboard could be provided by the website instead of the installed third-party keyboard application. Figure 6 shows such examples of keyboards provided by websites[7]. When a user tries to login to their websites, a trustworthy keyboard interface is automatically popped up.

Since the current Android platform can be used without any change to support this approach, this technique seems to be highly recommendable.

C. Redesigning permission warning messages

As mentioned in Section III, `INTERNET` is the only permission needed to complete keylogging process. However, the Google Play store does not inform the `INTERNET` permission to users (see Figure 3(b)). According to the Google's explanation[8], this is because most applications typically require the `INTERNET` permission.

We surmise that the current warning system is designed for average applications and thus might underestimate risks for dangerous applications. We suggest that permission warning messages for third-party keyboard applications should be reconsidered particularly due to their potential risks.

D. Improvement of the application review process

Even though Google runs an application review process before publishing an application, the successful registration of our proof-of-concept keylogger application in the store shows the limitations of this process. Thus, Google should improve its automatic application review process to avoid potentially harmful applications being added to the store.

VI. Ethical Considerations

We just wanted to show that such a malicious keylogger can successfully pass the Google's application review process. It is not our intention to distribute a potentially dangerous keylogger application to users. Therefore, we added warning messages into the application description to prevent actual users from installing this keylogger application, and no one has downloaded our application during the study. After the study has been completed, we removed the application from the Google Play store.

[7]They are popular bank websites in South Korea
[8]https://support.google.com/googleplay/answer/6014972

VII. Conclusion

In this paper, we discussed a potential security issue about third-party custom keyboards by implementing a proof-of-concept keylogger and successfully registering that application in the Google Play store. Interestingly, the `INTERNET` permission alone is enough to deploy a malicious keylogger that can be used to steal a user's sensitive input keystrokes.

We also examined the security behaviors of 139 keyboard applications that were freely available on the Google Play store. Even though we failed to discover real keyloggers that are actively running on Android devices, we note that keyboard applications with the `INTERNET` permission can be easily transformed to malicious keyloggers anytime. Therefore, it will be important to develop defense mechanisms so as to make such keyloggers ineffective.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2014R1A1A1003707), the ITRC (Information Technology Research Center) support program (IITP-2015-H8501-15-1008), and the ICT R&D program (2014-044-072-003, 'Development of Cyber Quarantine System using SDN Techniques') of MSIP/IITP.

References

[1] D. Damopoulos, G. Kambourakis, and S. Gritzalis, "From keyloggers to touchloggers: Take the rough with the smooth," *Computers & Security*, vol. 32, pp. 102–114, 2013.

[2] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion." in *Proceeding of the 6th USENIX Workshop on Hot Topics in Security*, 2011.

[3] F. Mohsen and M. Shehab, "Android keylogging threat," in *Proceeding of the 9th IEEE Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2013, pp. 545–552.

[4] B. Leiba, "Oauth web authorization protocol," *IEEE Internet Computing*, vol. 16, no. 1, pp. 74–77, 2012.

[5] F. Maggi, A. Valdi, and S. Zanero, "Andrototal: A flexible, scalable toolbox and service for testing mobile malware detectors," in *Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices*, 2013, pp. 49–54.

[6] AV-Comparatives, "Mobile security review," Tech. Rep., 2012.

[7] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws," *ACM Computing Surveys*, vol. 26, no. 3, pp. 211–254, 1994.

[8] L.-K. Yan and H. Yin, "Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis." in *USENIX Security Symposium*, 2012, pp. 569–584.

[9] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems*, vol. 32, no. 2, p. 5, 2014.

[10] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, p. 29.