Android App: Keyboard or Malware

Andrew Pannell, Pentest Limited

1 Introduction

ANDROID is the most popular mobile operating system. Over 1.5 million new Android devices are activated daily [1], with over 2 million apps available for download from the official Google Play Store [2]. This rapid growth has meant a rapid release of applications as developers try to cash in on the latest technology. Android has been a target for malware due to its wide adoption across a large user base (over 1.4 billion active devices worldwide).

This paper will look at an application with over 50 million downloads, and examines how the application is abusing Android permissions in order to harvest personal private user data and insert adverts maliciously.

2 Previous Work

Previous work relating to the Android operating system by the author of this paper includes;

Wireless backup applications, network and data security test, 2014

Android Operating System: A Forensic Examiners Perspective, 2012

Android Operating System: Vulnerabilities, exploits and malicious code, 2012

3 Android Security

Part of the Android security architecture is developed around an application "permission" model. An application starts with zero permissions, which means that by default it should not be able to do anything to impact any data on the device. An application will have an AndroidManifest.xml file which presents information to the Android operating system about the application. The Manifest file will include:

- ▷ Details of application components
- ▷ The minimum version of the Android API required
- ▷ Details of the system permissions required

The request for permissions are declared in the manifest file as: `<uses-permission />`, for example if an application requests access to the devices storage (either internal or external such as a micro SD card) the manifest declaration would be:

```
<uses-permission android:name=
"android.permission.WRITE_EXTERNAL_STORAGE" />
```

Prior to Android 6.0 (Marshmallow) when an application was installed, permissions were accepted or denied in full, i.e. if an application requests access to the camera and contact information upon the point of install, the user would either agree to both of these and therefore the application would be granted these permissions and the application would install. Or the user denied these permissions to the application and the application would fail and not be installed.

4 Android Applications

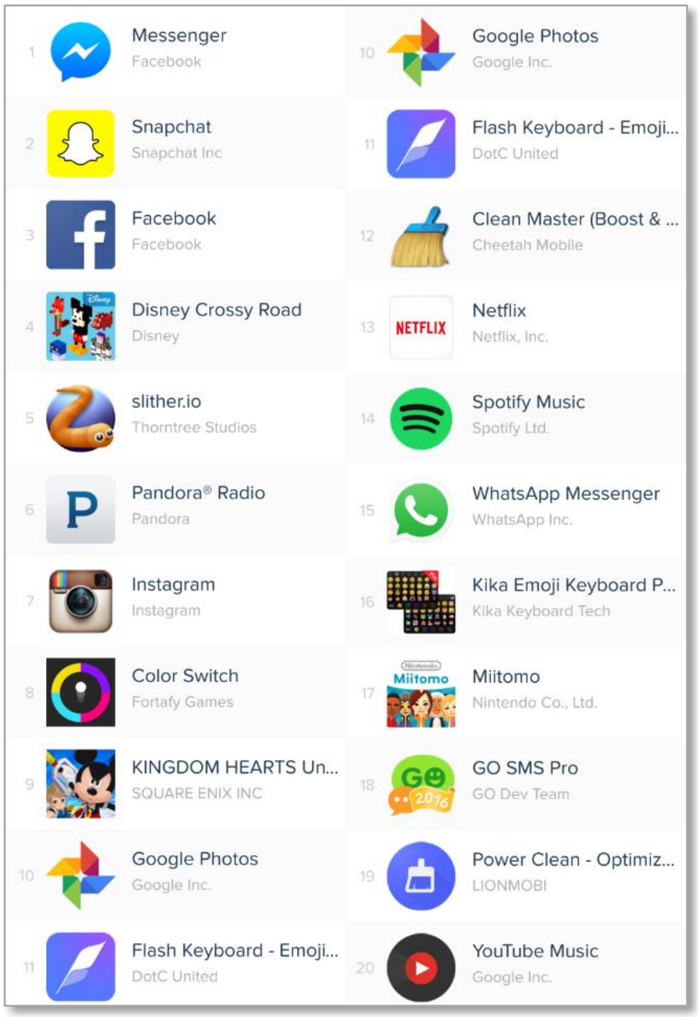
Android applications are written in Java, and because of the vast form factors of Android devices the applications are downloaded to the device in Java code format. When the application is installed Android uses its Android Runtime to fully compile the application from its code, to generate a compiled app executable specific for the device. Therefore it is trivial to take the application and reverse it into the Java classes it came from. Although decompilation of Android applications is a commonly discussed topic, it is worthwhile describing the process in this paper for the sake of clarity.

The Android application is downloaded onto the device in a file with an apk extension, for example FlashKeyboard.apk. This apk file is a zipped directory containing the resources required for the application. These resources include images, the previously mentioned AndroidManifest.xml file and a classes.dex file. The dex (Dalvik executable) file contains the compiled Java code that we're interested in reverse engineering. To convert the classes.dex file to a readable format is a two-step process. Firstly, it needs to be converted to a jar file using the tool dex2jar [3]. Secondly, the jar file can read using a Java Decompiler such as JD-GUI.

5 Flash Keyboard

At the time that this research began (25th February 2016) the application Flash Keyboard developed by DotC United was the 11th most popular downloaded Android application available on the Google Play Store. The application had at that time been installed between 50 million and 100 million times. The application is so popular it had been downloaded more times that other well recognized applications such as WhatsApp, this is shown in Figure 1, which shows the top 20 downloads for the American Play Store for April 2016. Flash keyboard is 11th.

Figure 1. Top 20 Downloaded applications



1	Messenger	Facebook
2	Snapchat	Snapchat Inc
3	Facebook	Facebook
4	Disney Crossy Road	Disney
5	slither.io	Thorntree Studios
6	Pandora® Radio	Pandora
7	Instagram	Instagram
8	Color Switch	Fortafy Games
9	KINGDOM HEARTS Un...	SQUARE ENIX INC
10	Google Photos	Google Inc.
11	Flash Keyboard - Emoji...	DotC United
10	Google Photos	Google Inc.
11	Flash Keyboard - Emoji...	DotC United
12	Clean Master (Boost & ...	Cheetah Mobile
13	Netflix	Netflix, Inc.
14	Spotify Music	Spotify Ltd.
15	WhatsApp Messenger	WhatsApp Inc.
16	Kika Emoji Keyboard P...	Kika Keyboard Tech
17	Miitomo	Nintendo Co., Ltd.
18	GO SMS Pro	GO Dev Team
19	Power Clean - Optimiz...	LIONMOBI
20	YouTube Music	Google Inc.

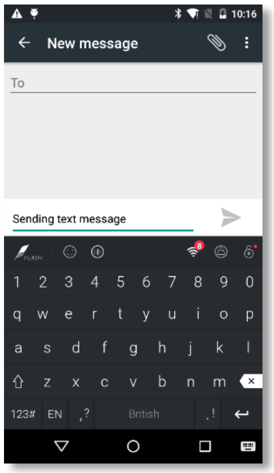
This paper was written against version 1.0.27 of the Android application. The application is intended to be used as a replacement to the stock keyboard and can be seen in Figure 2.

The idea of a 3rd party keyboard application is to improve the stock keyboard in some way, whether that's extra customisation, improved auto-correct or otherwise.

The developers make a number of statements regarding Privacy. First, on the application's Play Store description the developer make the following statement:

"Privacy. Flash keyboard values your privacy. We don't collect any personal data without your explicit permission. "

Figure 2. Flash Keyboard



Additionally, a post was also made on the Flash Keyboard Facebook page on 16th February 2016 which states:

"Flash keyboard values your privacy. The warning that our keyboard may collect ' all the text you type, including personal data like passwords and credit card numbers' is part of Android notification that appears when you install ANY third-party keyboard. We are just a keyboard not 007 and your trust means a lot to us. Don't give us 1-star just because of that warning, please~~~"

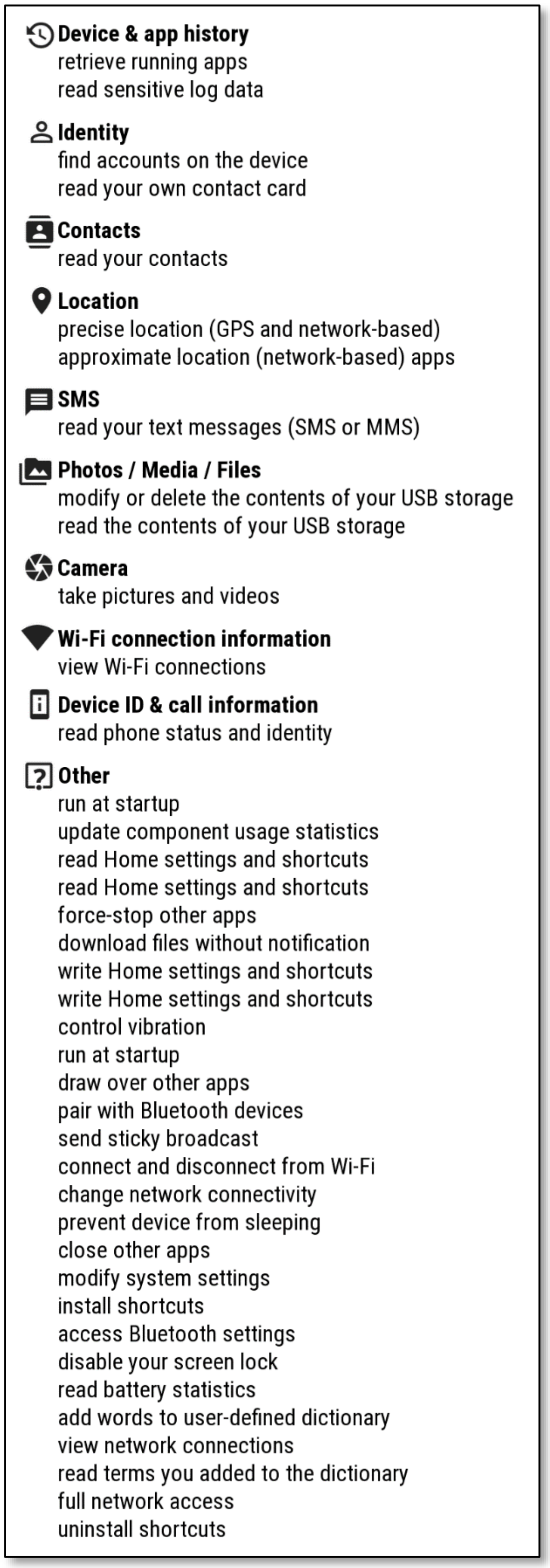
This shows that the developers of Flash Keyboard present themselves as taking the privacy of their users' data seriously.

6 Permissions

When an application is installed on Android, the user is presented with a list of permission the application requires, so they can make an informed choice before installing. Table 1 shows the permissions required by the Flash Keyboard application.

Google have developed Android's permission system to allow non-technical users to make informed decisions when installing an application. However, it is evident from the number of installations of Flash Keyboard that many users do not read the requested permission or do not understand the risk related to them. Maybe because the application is listed in the top 20 applications or the statements of privacy from the developers reassures the users to override their concerns.

Table 1. List of permissions shown on Play Store



Device & app history
retrieve running apps
read sensitive log data

Identity
find accounts on the device
read your own contact card

Contacts
read your contacts

Location
precise location (GPS and network-based)
approximate location (network-based) apps

SMS
read your text messages (SMS or MMS)

Photos / Media / Files
modify or delete the contents of your USB storage
read the contents of your USB storage

Camera
take pictures and videos

Wi-Fi connection information
view Wi-Fi connections

Device ID & call information
read phone status and identity

Other
run at startup
update component usage statistics
read Home settings and shortcuts
read Home settings and shortcuts
force-stop other apps
download files without notification
write Home settings and shortcuts
write Home settings and shortcuts
control vibration
run at startup
draw over other apps
pair with Bluetooth devices
send sticky broadcast
connect and disconnect from Wi-Fi
change network connectivity
prevent device from sleeping
close other apps
modify system settings
install shortcuts
access Bluetooth settings
disable your screen lock
read battery statistics
add words to user-defined dictionary
view network connections
read terms you added to the dictionary
full network access
uninstall shortcuts

6.1 Excessive Permissions

Pentest consider a number of the requested permissions to be particularly “dangerous”. These are permissions that are in excess of what would be required for the normal operation of a keyboard application. Further explanation is provided below of the requested permissions of particular concern.

6.1.1 Bluetooth

`BLUETOOTH & BLUETOOTH_ADMIN` These permissions allow the Flash Keyboard application to “connect to paired Bluetooth devices as well as discover and pair Bluetooth devices”. It is not clear why Flash Keyboard requests these permissions.

6.1.2 Camera

`CAMERA` This is a strange permission for a keyboard application to request, there isn't a common use case where a keyboard application would regularly require access to the hardware camera. Further analysis of the applications code shows this permission is used for a custom “sticker” generator.

6.1.3 Contacts

`READ_CONTACTS` This permission is used to allow the application to read the user's contacts data. It is understood this is so that Flash Keyboard can add the user’s contacts to the dictionary for predictive text.

6.1.4 Device Admin

`BIND_DEVICE_ADMIN` This permission is used to grant administrator access to the application. Device admin is developed in mind for enterprise environments in order to place certain restrictions on the device. Figure 3a and Figure 3b show the details of which parts of the device admin API is used by Flash Keyboard. These are “allow shortcut functions”, which isn’t documented or currently used in the application and, “prevent unexpected uninstallations”, which makes it difficult for users to uninstall the application. Lastly, “lock the screen”, which controls how the lock screen works.

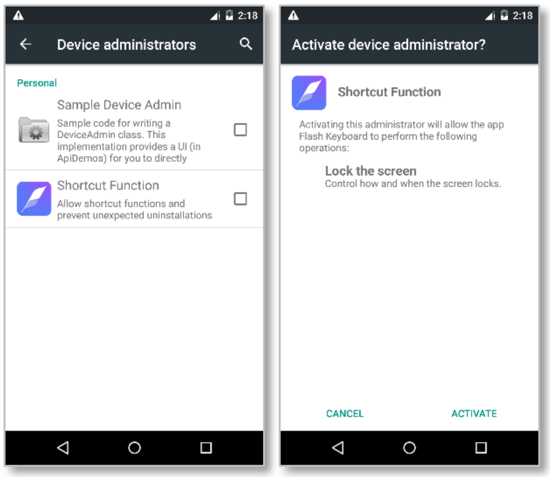


Figure 3. (a) and (b) Device Admin

2:18

Device administrators

Personal

Sample Device Admin

Sample code for writing a DeviceAdmin class. This implementation provides a UI (in ApiDemos) for you to directly

Shortcut Function

Allow shortcut functions and prevent unexpected uninstallations

2:18

Activate device administrator?

Shortcut Function

Activating this administrator will allow the app Flash Keyboard to perform the following operations:

Lock the screen

Control how and when the screen locks.

CANCEL ACTIVATE

Figure 4 shows the message received when attempting to uninstall the application. The uninstall process fails because the application has been granted device administrator privilege.

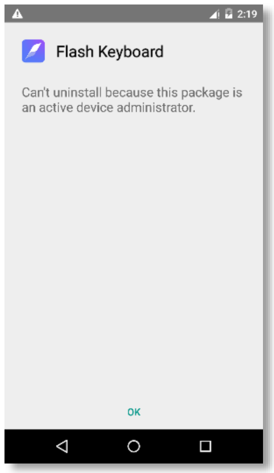


Figure 4. Uninstallation

2:19

Flash Keyboard

Can't uninstall because this package is an active device administrator.

OK

Although only a limited subset of device admin API feature are implemented. Additional device admin API functions could be added in future updates without notification to the user, as the user has granted the application the overall device admin permission. For example the developers could update the application to remotely lock the device immediately, set a screen unlock password essentially holding the user to ransom, disable the device's camera, and remotely wipe the device.

6.1.5 Keyguard

`DISABLE_KEYGUARD` Allows the application to disable the keyguard, this in Android means the lock screen.

This permission in conjunction with the `DEVICE_ADMIN` permission, allows Flash Keyboard to replace the standard Android lock screen with its own custom lock screen. This custom lock screen allows the developers to monitise the application by displaying paid for advertisements on the lockscreen. Figure 5a shows the Android lock screen on the device prior to installation of Flash Keyboard. Figure 5**Error! Reference source not found.**b shows the lock screen after the application has been installed.

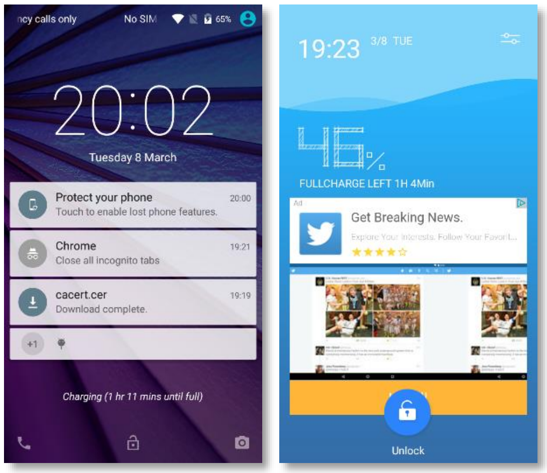


Figure 5. (a) and (b) Lockscreen

6.1.6 Location

`ACCESS_COARSE_LOCATION` & `ACCESS_FINE_LOCATION` These permissions allow approximate location and precise location respectively. These leverage Wi-Fi triangulation, Cell towers, and GPS in order to provide accuracy. The fine location permission has been known to provide between 1 and 3 meters of accuracy depending on the hardware implemented in the device. Location is considered sensitive information.

6.1.7 Logs

`READ_LOGS` Allows an application to read the low-level system log files. The Android developer's site makes the recommendation that this permissions is not for use by third-party applications, because Log entries can contain the user's private information. For example, research has shown that it is not unheard of to find developers writing user credentials to the log file.

6.1.8 Network/Wi-Fi State

`ACCESS_NETWORK_STATE` & `ACCESS_WIFI_STATE` & `CHANGE_NETWORK_STATE` & `CHANGE_WIFI_STATE` These permissions allow the application to toggle Wi-Fi on and off, as well as toggling mobile data. It is not clear why Flash Keyboard requests these permissions.

6.1.9 Processes

`KILL_BACKGROUND_PROCESSES` allows an application to call `killBackgroundProcesses(String)` This means that the application could make calls to kill other processes. For example, this could be abused for disabling other applications such as anti-virus.

6.1.10 SMS

`READ_SMS` Allows the application to read SMS messages stored on the device and SIM card. This permission is used by Flash Keyboard to add terms to the dictionary for predictive text. Figure 6 shows an example of the code used. The application reads the contents of text messages, and adds each word into the user's dictionary.

Figure 6. uv.class

```
public class uv
  implements SharedPreferences.OnSharedPreference
ChangeListener
{
  static final String PREF = "sms";
  static final String PREF_KEY_MAX_ID = "max_id";
  static final Logger jdField_a_of_type_OrgSlf4jL
ogger = LoggerFactory.getLogger("SmsDictionaryLoa
der");
  private final Context jdField_a_of_type_Android
ContentContext = MainApp.a();
  private final SharedPreferences jdField_a_of_ty
pe_AndroidContentSharedPreferences;
  private final ContentObserver jdField_a_of_type
_AndroidDatabaseContentObserver = new ContentObse
rver(null)
  {

jdField_a_of_type_OrgSlf4jLogger.debug("id: " + p
aramLong + " addWordsToDictionary success: " + a(
localuq.a) + str);
```

6.1.11 System Overlay

`SYSTEM_ALERT_WINDOW` This permission allows an application to create windows using the `TYPE_SYSTEM_ALERT`, shown on top of all other apps. The Android developer's site directly states "Very few apps should use this permission; these windows are intended for system-level interaction with the user."

It is not clear why Flash Keyboard requests this permission. This permission can be abused in a similar way to Clickjacking in Web Applications.

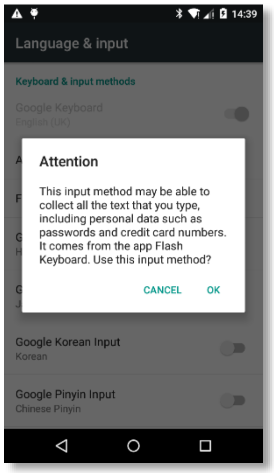
6.1.12 Download Notification

`DOWNLOAD_WITHOUT_NOTIFICATION` This permission allows an application to remove the notification to the user that the application is downloading external files. Facebook and WhatsApp have previously used this permission to silently update applications without updating the application via the Play Store. This is a violation of the terms and conditions provided by the Play Store and these apps have since removed this permission.

6.1.13 Keypresses

`BIND_INPUT_METHOD` This permission allows the application to act as a keyboard, connecting to the input method editor. This allows keypresses to be sent to the Flash Keyboard application be to be processed. By its very definition and use case, a keyboard requires access to keypresses, therefore having interaction with everything the user types. Android brings this to user's attention by presenting a warning when enabling the keyboard as shown in Figure 7.

Figure 7. Warning



Attention

This input method may be able to collect all the text that you type, including personal data such as passwords and credit card numbers. It comes from the app Flash Keyboard. Use this input method?

At this point there is no evidence to suggest Flash keyboard is logging keys entered by the user.

6.2 Internet Access

From June 2015 Google enabled Internet access by default for applications from Android 6.0 [4]. Prior to this point application has to request Internet access. The application still needs to declare the `android.permission.INTERNET` in the manifest, but this is option is no longer presented to the user. To confirm; the `INTERNET` permission is declared in Flash Keyboard's manifest file, and is therefore the granted access to the Internet.

7 TRANSMITTED DATA

Analysis of the application made it clear it was communicating out to servers in various countries. This included the United States, the Netherlands and China. The data going to China is possibly for the use of analytics. Figure 8**Error! Not a valid bookmark self-reference.** shows an example a HTTP POST request made by the application which send encoded data to the address tdcv3.talkingdata.net. TalkingData is China's largest independent Big Data service platform with focus on the mobile Internet.

Figure 8. Example of transmitted data

```
POST /g/d HTTP/1.1
Content-Length: 2305
Host: tdcv3.talkingdata.net
Connection: close

U&?¯3õp¢cFø=k×·X3hÊiÍ•ÇzÕ±Á×†acÚ0•møÀð•'XìäÉ•Ï•bŸ-
ÄÀtpÂA:KI"£B
---SNIP---
```

By decompiling application and analysing the code for the relevant class, it was identified that the data was being encoded using GZIP. Figure 9 shows the decompiled class which makes reference to GZIPOutputStream and the Talking Data URL.

Figure 9. com.tendcloud.tenddata.s

```
package com.tendcloud.tenddata;

import android.os.SystemClock;
import android.util.Log;
import java.io.ByteArrayOutputStream;
import java.util.Map;
import java.util.TreeMap;
import java.util.zip.GZIPOutputStream;

public final class s
{
  private static final di a = new di("", "tdcv3.t
alkingdata.net", b, 443);
  public static final String a = "tdcv3.talkingda
ta.net";
  private static String b = "211.151.164.164";
  private static final String c = "http://tdcv3.t
alkingdata.net/g/d";

  public static boolean a(du paramdu)
  {
    TreeMap localTreeMap = new TreeMap();
    try
    {
      ByteArrayOutputStream localByteArrayOutputS
tream = new ByteArrayOutputStream(1024);
      GZIPOutputStream localGZIPOutputStream = ne
w GZIPOutputStream(localByteArrayOutputStream);
      new co(localGZIPOutputStream).a(paramdu);
      localGZIPOutputStream.finish();
      localGZIPOutputStream.close();
```

Analysis of the decoded GZIP data reveals that the following information was being sent:

- ▷ Device manufacturer
- ▷ Device model number
- ▷ Device IMEI
- ▷ Android version
- ▷ Owners email address
- ▷ Wi-Fi SSID
- ▷ Wi-Fi MAC
- ▷ Mobile Network (e.g. Vodafone)
- ▷ GPS co-ordinates accurate to 1-3 meters
- ▷ Information about nearby Bluetooth devices
- ▷ Details of any proxies used by the device

It is worth noting that the Wi-Fi SSID and MAC included all nearby Wi-Fi access point not just the access point that device was connected to.

Evidently the application sends personal information such as email address and location to this Chinese analytical server without the knowledge of the user.

8 DECEPTIVE BEHAVIOUR

Google forbids deceptive behaviour and the Android Developer Policy Center states:

"We don't allow apps that attempt to deceive users. Apps must provide accurate disclosure of their functionality and should perform as reasonably expected by the user. Apps must not attempt to mimic functionality or warnings from the operating system or other apps. Any changes to device settings must be made with the user's knowledge and consent and be easily reversible by the user."

Pentest believe that the Flash keyboard is in breaches this policy for the following reasons:

- ▷ Mimics operating system functionality by replacing the built-in lock screen with its own.
- ▷ Does not disclose that it replaces the lock screen to display advertisements.
- ▷ Allows application updates and intentionally hides operating system notifications that would alert the user to the update.
- ▷ Makes it difficult for the average user to uninstall.
- ▷ Sends personal information to a 3rd party site without the user's knowledge.

9 Conclusion

It is Pentest's opinion that this application was not written by the developers to be intentionally malicious. However through disregard for Android's development policy and a desire to monitise a free application, have created an application that deceives users, gathers personal information and obstructs uninstallation.

In more sinister hands, this application could covertly download updates that weaponises the application; to exploit the granted privileges for mass or even targeted surveillance.

Pentest Limited attempted to make contact with the developers of Flash Keyboard in order to make them aware of this research, the findings, and to offer an explanation but received no response.

Pentest also notified Google on the 22nd April 2016. Although no official response has been received, as of 6th June 2016 Flash Keyboard appears to be removed from the Google Play Store.

Following its removal, another application called "Flash Keyboard Lite" by a developer called "Flash Keyboard team" has since appeared on the Google Play store. An initial inspection this "new" application appears to be built from the same original codebase as Flash Keyboard.

10 Technical Specifics

This section provides the details about the equipment used during testing.

Hardware device: Motorola G (3rd Generation), Android version 5.1.1, Build Number LPI23.72-47.

Application: Flash Keyboard, Version: 1.0.27, MD5 Hash: e5e2323a48959b12113206f83a094220

About Pentest

Established in 2001, Pentest Limited is a leading international provider of IT security, specialising in Web Application Security and Penetration Testing services. Pentest consultants offer expertise, flexibility, clear communication and extensive support before, during and after any assessment. Pentest is an ISO 27001 & 9001 accredited organisation, committed to providing an unparalleled service in the Information Security industry. For more information, or for further details about Pentest's services, please visit www.pentest.co.uk or call +44 (0) 161 233 0100.

References

[1] "1.5m Android devices activated daily," [Online]. Available: http://www.pocket-lint.com/news/122459-1-5m-android-devices-activated-daily-1-billion-total-devices-on-horizon.

[2] "Number of available applications in the Google Play Store," [Online]. Available: http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[3] "GitHub - Dex2Jar," [Online]. Available: https://github.com/pxb1988/dex2jar.

[4] "Normal Permissions," [Online]. Available: https://developer.android.com/guide/topics/security/normal-permissions.html.