Detection of Malicious Applications on Android OS

Francesco Di Cerbo[1], Andrea Girardello[2],
Florian Michahelles[2], and Svetlana Voronkova[1]

[1] Center for Applied Software Engineering, Free University of Bolzano-Bozen,
Bolzano-Bozen, Italy
{fdicerbo,svoronkova}@unibz.it
[2] Information Management, ETH Zurich
Zurich, Switzerland
{agirardello,fmichahelles}@ethz.ch

Abstract. The paper presents a methodology for mobile forensics analysis, to detect "malicious" (or "malware") applications, i.e., those that deceive users hiding some of their functionalities. This methodology is specifically targeted for the Android mobile operating system, and relies on its security model features, namely the set of permissions exposed by each application. The methodology has been trained on more than 13,000 applications hosted on the Android Market, collected with AppAware. A case study is presented as a preliminary validation of the methodology.

Keywords: Mobile Forensics, Android OS, Security.

1 Introduction

The amount and the significance of personal data stored on cellular phones, PDA's and Smart Phones is equal to those carried by computers, due to the use of numerous cloud synchronization services like Funambol [1], Microsoft ActiveSync[2], Apple MobileMe[3]. This aspect is particularly relevant, as mobile phones are also used when committing crimes: in many cases, for instance, wiretapping gives valuable benefits to investigations. This results in a growing attention on mobile devices involved in crimes, as a valuable source of information. Thus, there is also a strong interest in the evolution of digital forensics techniques. Mobile phones contain sensible data, that in a trial could be precious to demonstrate innocence or guilt. With "sensible (or sensitive) data", we refer to a broad definition of information that are relating to race or ethnic origin, political opinions, religious beliefs, physical/mental health, trade union membership, sexual life or criminal activities. They include communications log, SMS, MMS, contacts list, appointments, tasks and so on.

Evidence discovered during a mobile device analysis might have a vital importance for the case investigation. Here evidence is interpreted as information of probative value that is stored or transmitted in binary form [4]. Digital forensics is generally defined as a branch of e-discovery that examines how the extraction of digital evidence can aid in crime investigations and identification of potential suspects [5]. In some cases, as it happens on regular desktop systems, criminals may use mobile applications to conduct frauds, i.e., stealing home banking and other sensible credentials from mobile phones: this kind of applications are generally called "malware". In those situations, some evidences and insights on the crime are contained in an application, therefore it is necessary to identify it precisely, in order to gather as much information as possible.

The methodology we propose is a mobile forensics technique, aimed at supporting a forensics analyst to detect applications that deal with sensible data in a way that could be defined as "suspicious", i.e., not aligned with respect to the trends of all other applications, that are part of a significant dataset of safe applications.

In the following Sections 2 and 3 we will provide a short overview on mobile forensics and on Android OS, especially on its key issues that are significant for the mobile forensics science. In Section 4 it will be presented AppAware, the application we used to collect data on Android applications, and finally Section 5 and 6 will cover the proposed methodology, and a use case based on real applications. Section 7 contains the final remarks and future perspectives.

2 Mobile Forensics

The mobile forensics science as a part of digital forensics focuses on recovering digital evidence from a mobile phone under forensically sound conditions using accepted methods [6]. The mobile forensics techniques and methodologies focus mainly of 3 different areas [7]:

- *SIM card forensics*: it aims to extract the data stored on this physical item and provide so called primary image of it.
- *Digital data acquisition*: extraction of data carried by flash memory of mobile device using through filesystem
- *Physical data acquisition*: extraction of full memory image bit-by-bit.

Although the goals of computer forensics and mobile forensics are basically the same, the examiners consider mobile forensics to be much more complicated. The key issues that cause a trouble for investigators is the uniqueness of mobile device and its pervasive nature. Each cell phone manufacturer sets up his own standards, uses particular operating system, hardware and software [8]. Therefore information acquisition from such devices in a forensically sound manner becomes a real challenge. Digital forensics techniques are usually divided into 2 main groups [8]: *post mortem analysis*: when the device is switched off; *live analysis*: techniques performed on the device that is turned on.

However, both techniques differ in many aspects from the traditional methods when it comes to the cell phone investigation. Post mortem analysis (also called off-line analysis) of small scale device becomes more complex than computer examination, due to the fact that mobile devices contain an internal clock, which continuously changes data stored in the cell phone's flash memory. Therefore, it is impossible to reproduce a consistent bit-to-bit image of the entire memory. Considering the live forensic, device connectivity plays a vital role [9]. It is necessary to keep the device isolated from any networks during the analysis time, otherwise it may lead to the loss of some information that could be beneficial for the investigation. However, in the case of mobile devices, the preservation of this requirement is more difficult, because of the expanded connectivity options (i.e., the possibility to deal with cellular network, and through it with Internet services).

3 Android OS

Android OS (referred as "Android") delivers a complete set of software for mobile devices: an operating system, middleware and key mobile applications [10]. It enables the developers to take advantage of all functionalities and features included in the handset to create innovative and sophisticated mobile applications. Each Android application runs in its own process on Dalvik, a custom virtual machine designed for embedded use. Android relies on modified Linux Kernel version 2.6 for core system services such as security, memory management, process management, network stack, and driver model [11]. It also includes a set of Java libraries that provide the functionalities available in standard Java Programming Language, and C/C++ libraries such as SQLite relational database management system, 3D libraries, Media Libraries etc.

3.1 Android Security Model - A Permission-Based Approach

Android's security model combines the standard Linux OS features that control the security at the process level and the permission based mechanism. The permission [11] is a right that a developer has to declare in its application to be able to interact with the system or access the components of other applications. As each program is executed as a distinguished process, typically applications neither read nor write each other's data or code. Sharing data between applications must be done explicitly. However, after a permissions request, an application has access to the protected features of Android to which each permission refers. A permission is generally simple text string assigned to a predefined list of functionalities of the system, i.e., "INTERNET" to connect to the Internet, "READ_SMS" to read SMS messaging, and so on. Permissions have to be statically defined in the application package, so that during the deployment, a user could grant them to the application, or abort the process. In Listing 1.1, it is shown an example of a permission to write data to the SD card:

```
<permission xmlns:android=
    "http://schemas.android.com/apk/res/android"
  android:name="com.isecpartners.android.
      WRITE_EXTERNAL_STORAGE"
  android:description="@string/access_sd_card"
  android:protectionLevel="normal"
  android:label="@string/access_perm_label">
</permission>
```

Listing 1.1. An example of an Android OS security permission

Each permission contains the following attributes: name, description, label and the protection level.

3.2 Android Threads

As an open source handset that relies on Linux Kernel, Android is considered to be a secure platform. Despite the Android malware market is still in an infancy stage, detection of some malfunction on Android Market had proven that it can be easily exploited by attackers.

Recently, a report of SMobile [13] considering 48,694 applications, found 29 of them to be possibly spyware available on the Android Market, while for other 383 it is possible to access authentication credentials stored on the mobile phone. The cited analysis has some commonalities with the case study presented in Section 6, as both of them use applications' permissions as a privileged source of information on mobile applications. The report confirms the urgency of developing anti-malware systems and checks, as well as forensics methodologies to be used against those applications in a trial. In particular, in [13] it is proposed that a number of specific permissions combinations can lead to declare applications as notable, suspicious or spyware. SMobile methodology is not fully disclosed, therefore a precise comparation with our approach cannot be developed. However, from what is disclosed, SMobile methodology seems different from the one proposed here, even if the identification of "notable permissions" and their identification into Android applications (i.e. permissions that allow to access sensible data stored on the phone) seems to be similar to the concept of sensible data access profile, which is defined later.

An example of Android malware is an application called "DROID09", that was discovered on Android Market in January of 2010 [12]. The application "pretended" to be a useful online banking utility, which was supposed to connect the user to its bank web page and process the transactions. However, it turned out that it was only facilitating a web browser connection and actually stealing online banking credentials of the users. Indeed it is not known how exactly the application was performing the fraud, how long it has been in Android Market and how many users installed it, until this application was removed.

However, in order to develop efficient forensics tools there should be a clear definition of what kind of applications should be considered as suspicious in Android. Applications can be considered spyware if they have the ability to spy the users sensitive data in a specific way by capturing them and transmitting it outside the local system.

4 AppAware

Today most mobile operating systems provide users with an application portal where they can search for applications published by third-party developers. However, finding new apps is not an easy task. Application portals offer thousands of applications and users often have to go through countless titles to find what they are looking for. AppAware [14] is a mobile application that tries to solve this issue by allowing users to discover mobile applications in a serendipitous manner. AppAware captures and shares installations, updates, and removals of Android programs in real-time. AppAware also offers statistics to discover the top installed, updated or removed applications.

This service provides also a new way to let user implicitly rate applications and thus define their acceptance. This acceptance is represented by a meter colored from red to green. When the gauge points toward the green range the acceptance is excellent, yellow range for good acceptance and red range if almost no AppAware user is keeping the application installed. The assumption behind this approach is that excellent/good applications are not removed once installed, whereas applications that are removed from the device are not liked/ considered useful according to users.

The AppAware system has a client-server architecture. The client component is the Android mobile application, which represents AppAware's graphical user interface (GUI) and allows following installations, updates and removals of applications shared by other users. The client application is also monitoring the Android OS, thus being able to detect installations, updates and removals of applications, even those not installed from the official Android Market. The applications monitored are more than 42,000 at the time of writing.

5 The Methodology

The methodology we propose aims at the detection of suspicious applications, using Android security permissions, especially those connected with personal information: credentials, contacts, calendar events, email, SMS/MMS and so on.

As remarked in Section 3.2, each application needs to declare explicitly the permissions it needs, with respect to the operating system or to other applications. This security constraint allows to discriminate exactly what data will be accessed, as the Android security model will prevent any other access from being actually performed. We propose the definition of profiles related to sensible data access, each of them distinguished by a specific set of permissions. Comparing the permissions requested by an application, with the reference model of sensible data access profiles, it is possible to detect if an application has different security requirements with respect to other applications in the same profile. In this way, an analyst could detect unexpected anomalies. The method is not going to clearly determine the maliciousness of a certain application, it simply signals to the forensics analyst a situation that could require additional specific investigations. In this way, it is possible to identify applications that hide their real features, pretending to be a game, for instance. The methodology could be used also by standard users, to detect if an application permission request is coherent with respect to those of direct competitors. However, this perspective is still being considered and it is not part of this work.

A limitation of our approach is the impossibility to identify suspicious software that exploits Android vulnerabilities, e.g., using Android native code. This possibility has been recentely demonstrated by Oberheide [16], and as no permissions are needed to access internal Android API, no methodology based on permission analysis can effectively identify the mentioned applications.

The method is composed by the following steps:

1. definition of a number of applications' classification profiles, associated to the manipulation of sensitive data types, managed on an Android mobile phone;
2. assessment of the permissions declared by a significant set of applications belonging to different classification profiles;
3. mining of association rules on the basis of the different classification profiles;
4. definition of a reference set of permissions for each classification profile.

To apply our method, we start with the definition of a set of classification profiles, to describe applications that manage or have access to sensible data. This classification (shown in Table 1) is based on the analysis of the default set of Android permissions, and takes into account features that are connected with each sensible data category profile. To analyze an application, multiple profiles could be considered, according to the specific functionalities offered by the application.

As second step, we conducted an assessment of the permissions requested by the most common applications. In order to perform this step, we used AppAware. In this way, at present we collected information on 13,098 selected applications (on over 42,000). We selected the dataset among application not reported to be malicious in users'comments, and from which we had permissions data. Both features are provided by the AppAware features. Moreover, the considered applications are distributed worldwide, and of almost any category available on the Android Market. This allows us to state that the sample considered is heterogeneous enough for our purposes.

For each application, we recorded the permissions that were declared at the installation on the device of AppAware users.

The third step was to analyze the dataset with the Apriori algorithm [17] (using the tool Weka[15]), with different parameters, in order to group the applications according to their similarity of requested permissions. Apriori is a technique used in association rules mining, and it is a process of discovering frequent patterns, associations and correlations between sets of items in database. The rule mining process has two main steps:

a. find all frequent itemsets. A frequent itemset is an itemset whose support is greater than the minimum support. The support of an itemset is a measure of how often the itemset occurs in a given set of transactions;

b. generate the association rules that have high confidence, from the frequent itemsets identified in the first step. Confidence is a measure of how often items Y appear in transaction that contain items X.

The Apriori algorithm performs these 2 steps using a "bottom up" approach, where frequent subsets are extended one item at a time, and groups of candidates are tested against the data. The result of the Apriori analysis is a list of itemsets, grouped by the number of simultaneous features. See the following Section 6 for an exemplification.

The fourth step is the identification of the most relevant clusters of applications, on the basis of the support. From this, it is also possible to consider the set of rules that depends directly on the selected clusters. It is possible to state that the clusters represent a typical configuration for applications that deal with the particular sensitive data profile in consideration.

6 Case Study

In order to show a use case of the described forensics analysis methodology, we consider two applications, MobileSpy and MobileStealth, known to be spyware by design [12]. Both applications access sensible information on an Android device, and are able to send to a criminal the data collected. Table 2 shows the capabilities possessed by MobileSpy and MobileStealth, while their permissions are shown in Table 3.

The features of MobileSpy and MobileStealth are absolutely peculiar, and cases of users that intentionally wish to expose all their private data to others are considered very rare by the authors.

Our case study assumes that both applications have been found on a mobile device: data gathered through them might have been used to commit a crime, such as a non-legitimate bank transfer, and a forensics analyst has to identify the source of the information leak.

The analyst, having created a forensics copy of the device, will start her activities retrieving the list of all application deployed on the mobile phone, together with all permissions requested for each application. To this purpose, we developed an application, called AForensics: it is an Android application that looks up for third-party applications installed on a device, and retrieves their permission information. AForensics gathers permissions that third-party applications request from Android, and permissions declared by applications. Moreover, it also collects complementary information, like services, broadcast receivers and content providers that each application defined.

Once executing AForensics, the analyst shall receive an XML file, that contains, among data about other applications, the node shown in Listing 1.2.

```
<application name="MobileSpy" package="x.y.z"
        version="j.k.l">
  <requested-permission permission="android.permission.
      ACCESS_FINE_LOCATION" />
```

```
    <requested-permission permission="android.permission.
        READ_CONTACTS" />
    <requested-permission permission="android.permission.
        READ_SMS" />
    <requested-permission permission="android.permission.
        WRITE_SECURE_SETTINGS" />
    <requested-permission permission="android.permission.
        READ_PHONE_STATE" />
  ....
</application>
```

Listing 1.2. The XML snippet representing an excerpt of permission requested by the MobileSpy application

Table 1. Secure information profiles and relative permissions

Category	Android Permissions
Contacts	WRITE_SYNC_SETTINGS, WRITE_CONTACTS, READ_CONTACTS, MANAGE_ACCOUNTS, GET_ACCOUNTS, ACCOUNT_MANAGER
SMS-MMS messages	WRITE_SYNC_SETTINGS, WRITE_SMS, VIBRATE, SET_ORIENTATION, SEND_SMS, RECEIVE_SMS, RECEIVE_MMS, READ_SMS, FLASHLIGHT, BROADCAST_SMS
Call Log	VIBRATE, PROCESS_OUTGOING_CALLS, FLASHLIGHT, CALL_PHONE, CALL_PRIVILEGED
Audio & Video	WRITE_EXTERNAL_STORAGE, SET_ORIENTATION, RECORD_AUDIO, MODIFY_AUDIO_SETTINGS, GLOBAL_SEARCH, FLASHLIGHT, BLUETOOTH, BLUETOOTH_ADMIN, CAMERA
Tasks & Calendar	WRITE_CALENDAR, REORDER_TASKS, READ_CALENDAR, GLOBAL_SEARCH, GET_TASKS, BLUETOOTH_ADMIN
Browser History	WRITE_SYNC_SETTINGS, WRITE_HISTORY_BOOKMARKS, READ_HISTORY_BOOKMARKS
Images	WRITE_EXTERNAL_STORAGE, SET_WALLPAPER_HINTS, SET_WALLPAPER, SET_ORIENTATION, READ_FRAME_BUFFER, GLOBAL_SEARCH, FLASHLIGHT, BLUETOOTH, BLUETOOTH_ADMIN, CAMERA
Phone Settings	WRITE_SYNC_SETTINGS, WRITE_SECURE_SETTINGS, WRITE_GSERVICES, WRITE_APN_SETTINGS, SET_WALLPAPER, SET_WALLPAPER_HINTS, SET_ORIENTATION, READ_SYNC_STATS, READ_SYNC_SETTINGS, READ_PHONE_STATE, MODIFY_AUDIO_SETTINGS, FLASHLIGHT, ACCESS_CHECKIN_PROPERTIES, CHANGE_CONFIGURATION, DEVICE_POWER
Email & services related to web	WRITE_SYNC_SETTINGS, WRITE_GSERVICES, USE_CREDENTIALS, SET_ORIENTATION, MANAGE_ACCOUNTS, GLOBAL_SEARCH, GET_ACCOUNTS, FLASHLIGHT, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE, ACCOUNT_MANAGER, CHANGE_NETWORK_STATE, INTERNET

Table 2. Functionalities of MobileSpy and MobileStealth applications

MobileSpy	MobileStealth
monitor SMS messages	recording of surrounding
view inbound and outbound call details	SMS logging
access GPS location	contact details
view all websites visited from the device	picture logging
expose a web interface to view and manage the captured logs	GPS Tracking

Table 3. Permissions for applications MobileSpy and MobileStealth. All permissions belong to the *android.permission* package, except for *_HISTORY_BOOKMARKS, part of *com.android.browser.permission*.

MobileSpy	MobileStealth
RECEIVE_SMS	RECEIVE_BOOT_COMPLETED
SEND_SMS	READ_CONTACTS
READ_CONTACTS	RECEIVE_SMS
INTERNET	INJECT_EVENTS
READ_PHONE_STATE	PROCESS_OUTGOING_CALLS
ACCESS_FINE_LOCATION	READ_SMS
READ_CALENDAR	CHANGE_WIFI_STATE
RECEIVE_BOOT_COMPLETED	WRITE_SETTINGS
READ_SMS	READ_PHONE_STATE
WRITE_SMS	READ_LOGS
WRITE_CONTACTS	ACCESS_FINE_LOCATION
ACCESS_NETWORK_STATE	DISABLE_KEYGUARD
MODIFY_PHONE_STATE	INTERNET
ACCESS_COARSE_LOCATION	ACCESS_NETWORK_STATE
WRITE_CALENDAR	ACCESS_WIFI_STATE
READ_HISTORY_BOOKMARKS	WRITE_SECURE_SETTINGS
WRITE_HISTORY_BOOKMARKS	

The analyst should now compare the application permissions listed in Table 3 obtained with AForensics, with the reference models of application permissions discussed in Section 5.

To perform this comparison, the analyst could use the full AppAware dataset, or extract a subset of applications that belongs to the same or similar sensible data profiles. In this case, a subset could be generated considering applications that request READ_SMS, READ_CONTACTS and READ_PHONE_STATE permissions. If the second option is chosen, this leads to the creation of a subset, composed by 140 applications. The subset analysis gives a set of tables, containing the typical profiles of applications that manage user contacts. The full result presents 11 tables, considering itemsets that request from 1 to 11 permissions at the same

Table 4. Most relevant itemsets at the L7 of Apriori algorithm

Itemsets
RECEIVE_SMS, READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, SEND_SMS, WRITE_SMS
RECEIVE_SMS, READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, CALL_PHONE, WRITE_SMS
READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, CALL_PHONE, WRITE_CONTACTS, WRITE_SMS
RECEIVE_SMS, READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, WRITE_CONTACTS, WRITE_SMS
READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, CALL_PHONE, SEND_SMS, WRITE_SMS
READ_CONTACTS, READ_SMS, INTERNET, VIBRATE, READ_PHONE_STATE, SEND_SMS, WRITE_SMS
RECEIVE_BOOT_COMPLETED, RECEIVE_SMS, READ_CONTACTS, READ_SMS, INTERNET, READ_PHONE_STATE, WRITE_SMS

time (identified as "L1" to "L11"). The number of elements in each itemset varies from 4 (L11) to 3702 (L5). An excerpt of one of these tables resulting from the Apriori analysis is shown in Table 4, representing L7 itemset. Each row in Table 4 represents a group of 7 permissions.

It is clear that the permission sets of MobileSpy and MobileStealth do not appear in Table 3, and the authors have checked that such a permission set does not appear in any of the sets generated in the full analysis output.

In the cases of MobileSpy and MobileStealth, however, to declare the applications as suspicious it is sufficient to notice that, for instance, the pair READ_CONTACS and INJECT_EVENTS is present only once on the whole dataset, by a software development library. The difference between MobileStealth/MobileSpy and a software library corroborates a legitimate suspicion on the application; moreover, considering any other requested permission, the number of similar applications decreases to 0.

7 Conclusions

We presented a methodology for the detection of malicious applications in a forensics analysis. Malicious applications in this context are those that have access capabilities to sensible data, and transmission capabilities as well, at the same time deceiving the users, by pretending to offer services that typically do not require such capabilities, or to make a legitimate use of them. The methodology relies on the comparison of the Android security permission of each application, with a set of reference models, for applications that manage sensitive data. An extensive validation of the methodology is in progress, but we are facing difficulties due to limited information and data publicly available so far. In this research we only considered a rather simple analysis technique, namely association rules. Association rules are easy to apply and provide a first understanding of the problem at hand.

Some authors however report problems with their use, among them the potentially large number of groups and rules obtained, which may confuse the interpretation.

While there are solutions to some of these issues [18] in our future work we plan to investigate further data mining methods, for example classification and clustering algorithms. Classification is useful for predicting to which class an application with a given feature vector belongs, given that we can define our classes beforehand. Clustering of similar feature vectors could be used for defining classes as a first step. In the future, we plan to apply standard data mining and pattern recognition techniques (for instance [19]) to simplify the analysis process for the analyst. Particular attention will be devoted to the identification of the "best" learners for predicting the classification membership of new applications (see for example [20]).

Our ultimate objective is to implement a general model for an automated or semi-automated forensics system, which could detect application with abnormal permission requests, such as the previous case study of MobileSpy/MobileStealth. Such a system should be periodically re-trained (by performing the above steps) and therefore it shall be able to evolve over time. We are also considering to include the outcome of the research directly into AppAware, in order to warn all Android end users about suspicious software before installing it.

References

1. Funambol inc.: Funambol Open Source Mobile Cloud Sync and Push Email, `http://www.funambol.com`
2. Microsoft: ActiveSync, Windows Phone Synchronization, `http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-synch.mspx`
3. Apple inc.: Apple - MobileMe, `http://www.apple.com/mobileme/`
4. Scientific Working Groups on Digital Evidence and Imaging Technology: Combined Master Glossary of Terms (retrieved on July 2, 2010)
5. Robbins, J.: An Explanation of Computer Forensics, PC Software Forensics, `http://www.computerforensics.net/forensics.htm` (retrieved on July 2, 2010)
6. Jansen, W., Ayers, R.: Guidelines on Cell Phone Forensics. NIST Special Publication 800-101 (2007), `http://csrc.nist.gov/publications/nistpubs/800-101/SP800-101.pdf` (retrieved on July 2, 2010)
7. Kim, K., Hong, D., Chung, K., Ryou, J.: Data Acquisition from Cell Phone using Logical Approach. Proceedings of the World Academy of Science, Engineering and Technology 26 (2007)
8. Rizwan, A., Dharaskar, R.V.: Mobile Forensics: an Overview, Tools, Future trendsand Challenges from Law Enforcement perspective, `http://www.iceg.net/2008/books/2/34_312-323.pdf` (retrieved on July 2, 2010)
9. Carrier, B.D.: Risks of live digital forensic analysis. Commun. ACM. 49, 56–61 (2006)
10. Open Handset Alliance: Android, `http://www.openhandsetalliance.com/android_overview.html` (retrieved on July 2, 2010)

11. Android Community: What is Android?, `http://developer.android.com/guide/basics/what-is-android.html` (retrieved on July 2, 2010)
12. Vennon, T.: Android Malware, A Study of Known and Potential Malware Threats, `http://threatcenter.smobilesystems.com/wp-content/plugins/download-monitor/download.php?id=6` (published February 24, 2010)
13. Vennon, T., Stroop, D.: Android Malware, A Study of Known and Potential Malware Threats, June 21 (2010), `http://threatcenter.smobilesystems.com/wp-content/uploads/2010/06/Android-Market-Threat-Analysis-6-22-10-v1.pdf` (published June 21, 2010)
14. Girardello, A., Michahelles, F.: Explicit and Implicit Ratings for Mobile Applications. In: 3. Workshop Digitale Soziale Netze and der 40. Jahrestagung der Gesellshaft fr Informatik, Leipzig (September 2010)
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explor. Newsl. 11, 110–118 (2009)
16. Oberheide, J.: Remote Kill and Install on Google Android, `http://jon.oberheide.org/blog/2010/06/25/remote-kill-and-install-on-google-android/` (retrieved on July 2, 2010)
17. Orlando, S., Palmerini, P., Perego, R.: Enhancing the *apriori* algorithm for frequent set counting. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2001. LNCS, vol. 2114, pp. 71–82. Springer, Heidelberg (2001)
18. Garca, E., Romero, C.: Drawbacks and solutions of applying association rule mining in learning management systems. In: Proceedings of the International Workshop on Applying Data Mining in e-Learning (ADML 2007), pp. 14–22 (2007)
19. Duda, R.O., Hart, P.E., et al.: Pattern classification. Wiley-Interscience, Hoboken (2001)
20. Moser, R., Pedrycz, W., et al.: A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: Proceedings of the 30th International Conference on Software Engineering, pp. 181–190 (2008)