Proceedings of the Asia-Pacific Advanced Network **2013** v. 36, p. 23-31.
http://dx.doi.org/10.7125/APAN.36.4
ISSN 2227-3026

Detecting Android Malware by Analyzing Manifest Files

Ryo Sato[1,*], Daiki Chiba[2] and Shigeki Goto[1,*]

1 Waseda University / 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan

2 NTT Secure Platform Laboratories / 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

E-mails: r-sato@goto.info.waseda.ac.jp, chiba.daiki@lab.ntt.co.jp, goto@goto.info.waseda.ac.jp

* Tel.: +81-3-5286-3182; Fax: +81-3-5286-3182

Abstract: The threat of Android malware has increased owing to the increasing popularity of smartphones. Once an Android smartphone is infected with malware, the user suffers from various damages, such as the theft of personal information stored in the smartphones, the unintentional sending of short messages to premium-rate numbers without the user's knowledge, and the ability for the infected smartphones to be remotely operated and used for other malicious attacks. However, there are currently insufficient defense mechanisms against Android malware. This study proposes a new method to detect Android malware. The new method analyzes only manifest files that are required in Android applications. It realizes a lightweight approach for detection, and its effectiveness is experimentally confirmed by employing real samples of Android malware. The result shows that the new method can effectively detect Android malware, even when the sample is unknown.

Keywords: Android; Malware; Manifest files; Mobile Security; Static Analysis.

1. Introduction

With the rapid entry of smartphones into daily lives, Android *malware* has been rapidly spreading. The Android operating system (OS) is an easy target for attackers, because the market share of Android has increased, and many Android applications are written in the Java programming language.

According to a global survey of the smartphone OS market, Android possessed 68.8% of the market share in 2012 [1], implying that the popularity of Android has undergone significant growth. It is easy for malware to infect Android smartphones because of the large number of phones. Moreover, Android applications are easy targets for reverse engineering, which is a specific characteristic of Java applications in general, and which is often abused by malicious attackers, who attempt to embed malicious program into benign applications, hence creating subspecies of existing malware. Yajin et al. [2] illustrated that 86.0% of Android malware are created by conversion from benign applications. Hence, Android is considered to be an easy target for malicious attackers, and therefore, the privacy and integrity of the user's data are seriously threatened.

There have been numerous studies focusing on the detection of Android malware. One of the popular approaches includes signature-based methods, which extract signatures from malware samples. While it is effective for detecting known malware, it is inadequate for detecting unknown malware. Iland et al. [3] suggested a detection method at the network level. They observed network traffic originating from a sample application and tried to detect malware by comparing with DNS-based and IP-address-based *blacklists*. This method cannot detect unknown malware, because the blacklists are generated from known malicious activities. Isohara et al. [4] presented a method for detecting malware by analyzing attributes of files within sample applications. While this approach can detect only some unknown malware that are undetected by blacklist or signature-based methods, the analysis cost depends on the number of files within sample applications. Enck et al. [5] proposed a lightweight method to block the installation of applications that have dangerous *permissions* or *intent filter* (a mechanism for realizing cooperation between Android applications) combinations. However, the method may lead to incorrect detections, because the information used in the method is not sufficient to differentiate malware from benign applications. Wu et al. [6] developed a system to provide a static analysis paradigm for detecting malware, called DroidMat. They obtained some distinguishable characteristics such as permissions, *components* (essential functions such as Activity, Service and Receiver) and API calls by analyzing manifest files and *smali* files (disassembly codes). This system can discriminate between malware and benign applications. However, the cost of their analysis depends on the size and numbers of smali files. Our preliminary study measured the average file sizes and number of files that are the main resources in Android applications. Table 1 and 2 show the results. We investigated 30 benign samples and 30 malware samples.

Table 1. Average : File size (KB).

	smali files	Resources	Manifest file
30 Benign samples	6305	4759	7
30 Malware samples	3036	1431	4

Table 2. Average : number of files.

	smali files	Resources	Manifest file
30 Benign samples	674	385	1
30 Malware samples	249	101	1

From Table 1 and 2, we can observe that the cost of analyzing smali files is higher than that of manifest file.

This study proposes a new method for detecting Android malware by analyzing only manifest files. Each Android application must have a manifest file, which presents essential information about the application. Our preliminary investigations revealed that there are certain differences between the manifest files of benign applications and malware. Our proposed method is based on the characteristic analysis of Android manifest files and is effective for detecting well-known existing malware and unknown malware. Moreover, the cost is low, because this method analyzes only a manifest file. Table 1 and 2 show a manifest file is usually a small file.

The remained of this study is organized as follows: Section 2 proposes our new method. Section 3 describes the experiment used to demonstrate the effectiveness of the new detection method. Section 4 concludes this paper and discusses the possible future extension of the proposed method.

2. New Method for Detecting Android Malware

This study proposes a new method for detecting Android malware by analyzing only manifest files. Android applications consist of the following resources: a manifest file, application programs for Dalvik virtual machine (VM), and application resources. Figure 1 shows an Android application package (.apk), which includes a manifest file.

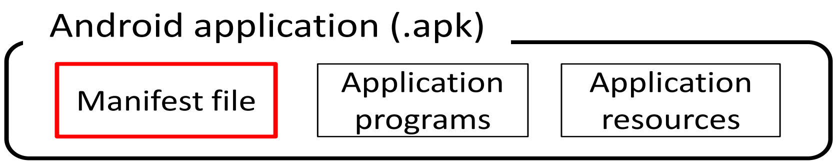

Figure 1. Android application package (.apk).

The manifest file takes the form of "AndroidManifest.xml," which must be present in all Android applications. Application programs are collected as "classes.dex." Application resources consist of pictures, music, and some xml files, which describe the layout information.

Android malware is detected by the following steps: [Step 1] Extract specific information described in the manifest file of a sample application. [Step 2] Compare the extracted information with the keyword lists that are provided in this new method. Then, calculate the malignancy score of the sample by comparing the information in Step 1 with the lists. [Step 3] Compare the malignancy score in Step 2 and the threshold values, which are set by this new method. If the malignancy score exceeds the threshold value, the sample is judged as malware. Figure 2 shows the flow of the new detection method.

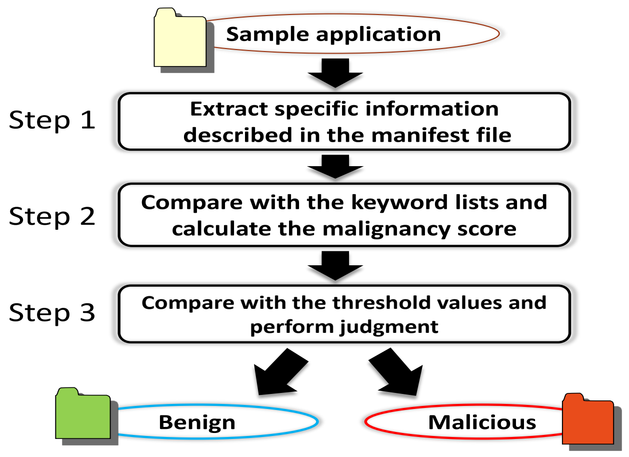
Sample application
Step 1
Extract specific information described in the manifest file
Step 2
Compare with the keyword lists and calculate the malignancy score
Step 3
Compare with the threshold values and perform judgment
Benign
Malicious

Figure 2. Flow of Detecting Android Malware.

2.1. Extract information items

Manifest files have essential information about Android applications, such as the version number of an application, the name of a package, required permission, and the API level. The format of the manifest file is identical in both benign applications and malware. However, there are certain differences in the characteristics of several information items. We investigated 30 benign samples and 30 malware samples, giving a total of 60 samples. We selected specific information items that show a wide variety of malware as compared to benign applications. Table 3 shows six information items that are extracted from manifest files and that are used to detect Android malware by the proposed method. The items are represented as text strings or numbers.

Table 3. List of information items.

(1) Permission
(2) Intent filter (action)
(3) Intent filter (category)
(4) Process name
(5) Intent filter (priority)
(6) Number of redefined permission

2.2. Keyword lists and malignancy score

With this new method, several keyword lists are compiled for an application. Benign or malicious strings in a manifest file are recorded in the keyword list. We make four types of keyword lists: (1) Permission, (2) Intent filter (action), (3) Intent filter (category), and (4) Process name. Because (5) Intent filter (priority) and (6) Number of redefined permission are represented by an integer, and not a text string, we have no keyword lists for them. Figure 3 counts the number of keywords, which are classified as “Permission” items.

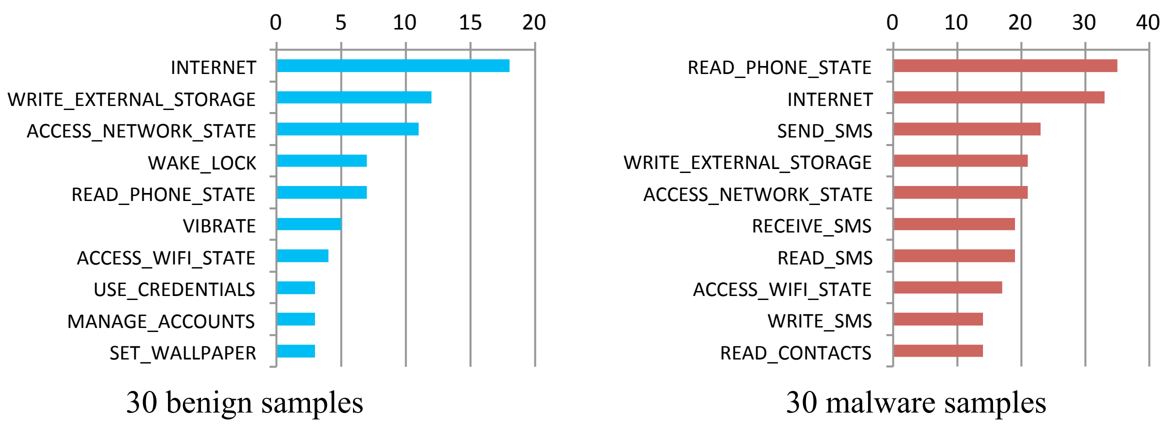

Figure 3. Permission keywords in each set of 30 samples.

Figure 3 shows the occurrences of popular permission keywords. This figure shows that the permissions which are related to short message service (SMS), such as SEND_SMS, RECEIVE_SMS, and READ_SMS, are frequently used by malware samples. These permissions are registered with the keyword list as malicious strings. A similar process is also performed for (2) Intent filter (action), (3) Intent filter (category), and (4) Process name. We have four keyword lists, which are shown in Table 4. Most keywords are considered to be malicious, while some are classified as benign.

Table 4. Keyword lists.

(List 1) Permission

1. READ_SMS
2. SEND_SMS
3. RECEIVE_SMS
4. WRITE_SMS
5. PROCESS_OUTGOING_CALLS
6. MOUNT_UNMOUNT_FILESYSTEMS
7. READ_HISTORY_BOOKMARKS
8. WRITE_HISTORY_BOOKMARKS
9. READ_LOGS
10. INSTALL_PACKAGES
11. MODIFY_PHONE_STATE

(List 2) Intent-filter (action)

1. BOOT_COMPLETED
2. SMS_RECEIVED
3. CONNECTIVITY_CHANGE
4. USER_PRESENT
5. PHONE_STATE
6. NEW_OUTGOING_CALL
7. UNINSTALL_SHORTCUT
8. INSTALL_SHORTCUT
9. left_up
10. right_up
11. left_down
12. right_down
13. SIG_STR
14. VIEW (*benign keyword*)

(List 3) Intent-filter (category)

1. HOME
2. BROWSABLE (*benign keyword*)

(List 4) Process name

1. remote2
2. main
3. two
4. three

After we obtained the keyword lists, the malignancy score for the above four information items are calculated. This process is performed by classifying the keywords as being benign or malicious. The malignancy score is calculated by formula (1).

$$\mathrm{P} = \frac{\mathrm{M} - \mathrm{B}}{\mathrm{E}} \tag{1}$$

where P: malignancy score, M: number of malicious strings, B: number of benign strings, E: number of total information items.

Table 5 shows an example. This sample uses five permissions items.

Table 5. Permissions keywords in a sample.

```
<uses-permission android:name=”android.permission.INTERNET” />
<uses-permission android:name=”android.permission.READ PHONE STATE” />
<uses-permission android:name=”android.permission.READ SMS” />
<uses-permission android:name=”android.permission.RECEIVE SMS” />
<uses-permission android:name=”android.permission.SEND SMS” />
```

Among these five permissions, READ_SMS, RECEIVE SMS, and SEND SMS are recorded in the keyword list and are classified as malicious strings in Table 4. Then, the malignancy score of this sample is calculated by formula (2).

$$\mathrm{P} = \frac{3 - 0}{5} = 0.6 \tag{2}$$

Similar calculations are also performed for (2) Intent filter (action), (3) Intent filter (category), and (4) Process name. With regards to (5) Intent filter (priority), the set-up value is counted and used for the judgment in Step 3. (6) Number of redefined permission is also counted and considered.

2.3. Thresholds and judgment

The proposed method provides threshold values for the malignancy score. We use a data mining tool, Weka [7], to determine the threshold values. As with the four categories of information items (1), (2), (3), and (4), the threshold values are set using the Weka J48 algorithm, which is based on a decision tree. We use both benign samples and malicious samples for machine learning. Specific samples are explained in Section 3. With regards to the threshold value for items (5) and (6), we set the threshold value at 1000 for (5) and 3 for (6) based on the result of our preliminary analysis, which was described in Section 2.1.

Judgment for an application sample is performed on the basis of conditions 1, 2, and formula (3), which are given below. Condition 1 describes the characteristics of malware. Condition 2 is made to avoid incorrect judgments. In formula (3), the SCORE refers to the final malignancy score of the sample. C1 and C2 count the number of items satisfied by a sample in condition 1, and condition 2, respectively.

Condition 1:

- Malignancy score is greater than the threshold value determined by Weka.
- Count of Intent filter (priority) is greater than the threshold value.
- Count of redefined permissions is greater than the threshold value

Condition 2:

- Malignancy score of (2) Intent filter (action) is negative (< 0)
- Malignancy score of (3) Intent filter (category) is negative (< 0)

Criteria formula:

$$\mathrm{SCORE} = \mathrm{C1} - \mathrm{C2} \tag{3}$$

If the final SCORE is greater than or equal to 1, the sample application is considered to be malware.

3. Experiment

To evaluate the performance of the proposed method, we conducted the following experiment with Android application samples.

3.1. Overview of the experiment

We collected 235 benign application samples and 130 malware samples. Benign samples were collected from Google Play [8] and some related markets. Malware samples were obtained from a web site that provides samples for research purposes [9]. All samples have a unique MD5 hash value and are classified into two groups: “Learning data” and “Test data.” Learning data is used to determine the suitable threshold values used by Weka, and the keyword lists are the same as in Table 4. Test data is used to evaluate the proposed new method. In this experiment, the samples are first analyzed by VirusTotal [10], which is an on-line scanning tool for malware. We classified a malware sample into the Learning data if the first registered data is before September 2011. The remaining malware samples are used for Test data. This date is selected to enable the acquisition of a sufficient number of malware samples for learning and testing. We can treat malicious Learning data as known samples and malicious Testing data as unknown samples. Note that malicious Testing data include samples that are not detected by signature-based methods. Incidentally, benign Learning data and Test data were randomly selected from the collected benign application samples. Table 6 shows the number of samples that were used in this experiment.

Table 6. Number of samples used in the experiment.

	Learning data	Test data	Total
Benign samples	60	175	235
Malware samples	34	96	130

3.2. Result of the evaluation

Table 7 shows the result of the experiment. It shows that the correct ratio of detecting benign samples is 91.4%, detecting malware samples is 87.5%, and it is 90.0% in total. This result indicated that the proposed method can accurately classify Android applications. The samples that are used as Learning data consist of only those whose first detected time is earlier than that of any Test data samples. Therefore, the proposed method is shown to successfully extract essential information from manifest files, although it only learns from old samples for which the first detected times were before September 2011. Therefore, it can detect unknown malware samples successfully.

Table 7. Result of the experiment.

	Correct detection (%)	Incorrect detection (%)
Benign samples	91.4	8.6
Malware samples	87.5	12.5
Total	90.0	10.0

3.3. Discussion

Some malware samples were not detected by the proposed method. We found that the proposed method was inadequate for detecting adware samples. In addition to actions that display some advertisements superfluously, there is often a marginal difference between a benign application and adware. This means that both manifest files appear to be similar, and it is difficult for the proposed method to effectively detect adware based on the manifest analysis.

4. Conclusion and Future works

This paper proposed a new detection method for Android malware. The advantage of this new method is that it uses only manifest files to detect malware. Manifest files are required in all Android applications, and thus, the proposed method is applicable to all Android applications. Our results show that the proposed method can detect unknown malware samples that are undetectable by a simple signature-based approach. Moreover, the cost of analyzing only the manifest file is extremely low. The new method can also be combined with other methods to realize an even more precise detection method.

Our evaluation uses only a small number of samples; only 365 samples in total. In future, we plan to collect additional samples to obtain more precise results for the evaluation experiments.

The proposed method extracts six types of information from manifest files and uses them to detect Android malware. The essential information items can be easily changed, and we should closely observe trends in Android malware to determine whether to keep or revise the effective information items in the manifest file.

References

1. IDC, “Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year,” Feb 2013. http://www.idc.com/getdoc.jsp?containerId=prUS23946013 (accessed on 21 Feb. 2013).
2. Yajin Z.; Xuxian J. Dissecting Android Malware: Characterization and Evolution. Security and Privacy (SP), 2012 IEEE Symposium on, San Francisco, USA, 2012, 5, 95–109.
3. Iland D.; Pucher A.; Schauble T. Detecting Android Malware on Network Level. University of California, Santa Barbara, 2011, 12.
4. Isohara T.; Kawabata H.; Yakemori K.; Kubota A.; Kani J.; Agematsu H.; Nishigaki A. Detection Technique of Android Malware with Second Application. Proceedings of Computer Security Symposium 2011, Niigata, Japan, 2011, 10, 19-21.
5. Enck W.; Ongtang M.; McDaniel P. On Lightweight Mobile Phone ApplicationCertification. ACM CCS 2009, 2009, New York, USA, 11, 9-13.
6. Wu D.; Mao C.; Wei T.; Lee H.; Wu K. DroidMat: Android Malware Detection through Manifest and API Calls Tracing. Seventh Asia Joint Conference on Information Security, 2012, 8, 62-69.
7. Weka. http://www.cs.waikato.ac.nz/ml/weka/
8. GooglePlay. https://play.google.com/store
9. contagio mobile. http://contagiominidump.blogspot.jp/
10. VirusTotal. https://www.virustotal.com/ja/
11. Sato R.; Chiba D.; Goto S. Analysis of Manifest File for Detecting Android Malware. The 75th National Convention of IPSJ, 2013, 3.