Android Security

[1]Parth Chhabhaiya, [2]Prof D.A.Parikh
Computer Department, LD College Of Engineering , Gujarat Technological University
[1]parth12692@gmail.com, [2]d a parikh@yahoo.com

***Abstract*—Android is a popular mobile device platform developed by Google. Android Application is now part of our day to day life, and there are several confidential data store in mobile devices. Most of the Android users are unaware about security of data and therefore many users' data is unsecure. Recently, the threat of Android malware is spreading rapidly. Therefore several Android Applications are defected by malware. Defected apps manipulate personal information such as SMS messages and Contacts, and leakage of such information may cause great loss to the android users.**

***Keywords*— Android, Malware, Android Apps, Security**

I. INTRODUCTION

Android operating devices becomes more and more popular in the past three year. Unfortunately, the increasing adoption of smart phone comes with the growing of mobile malware. A recent report from IDC[1] shows that there are 722 millions of Smartphone sold in 2012, Among them 497 millions of Smartphone have android operating system. Another report from Appbrain[2] mention that the android market surpassed 850000 app mark in December 2013. Unfortunately, such popularity also attracts the malware developers. Recent survey report from Juniper[3] alert that there is "614 percent increase in Android-based malware from march 2012 to march 2013". In android devices sensitive information are sent to unknown destination without user awareness. Thus privacy and security of data is become more important.

II. TIME LINE OF ANDROID MALWARE

Android is becoming an increasingly appealing target for cyber criminals[4]: in the second half of the year 2012, a new malware strain for the smart operating system was discovered every two minutes. During that period, G Data Se-curity Labs discovered almost 140,000 new malicious files – five times as many as in the preceding six months. With regard to Android malware, the perpetrators have been mainly relying on Trojan horses and tried and tested eCrime strategies. This involves using manipulated copies of known apps and applications with apparently legitimate functions, such as apps claiming to be weather apps.

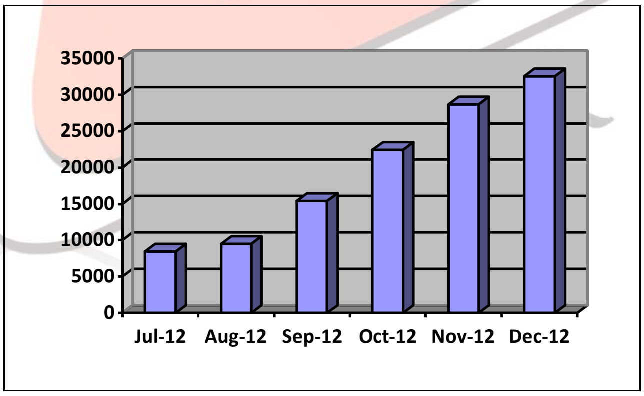

Fig 1 Malware growth in the second half of 2012

III. MALWARE CHARACTERIZATION

We present a systematic characterization of existing Android malware, ranging from their installation, activation, to the carried malicious payloads [5].

A. Malware installation

By manually analysing malware samples, we categorize existing ways Android malware use to install onto phones and generalize them into three main social engineering-based techniques, i.e., repackaging, update attack, and drive-by download. These techniques are not manually exclusive as different variants of the same type may use different techniques to entice users for downloading [5].

1. Repackaging

Repackaging is one of the most common techniques malware authors use to piggyback malicious payloads into popular applications. In essence, malware authors may locate and download popular apps, disassemble them, enclose malicious payloads, and then re-assemble and submit the new apps to official and/or alternative Android Markets. Users could be vulnerable by being enticed to download and install these infected apps. To quantify the use of repackaging technique, we take the following approach: if a sample shares the same package name with an app in the official Android market, we then download the official app and manually compare the difference, which typically contains the malicious payload added by malware authors. If the original app is not available, we choose to disassemble the malware sample and manually determine whether the malicious payload is a natural part of the main functionality of the host app. If not, it is considered as repackaged app.

2. Update Attack

The first technique typically piggybacks the entire malicious payloads into host apps, which could potentially expose their presence. The second technique makes it difficult for detection. Specially, it may still repackage popular apps. But instead of enclosing the payload as a whole, it only includes an update component that will fetch or download the malicious payloads at runtime. As a result, a static scanning of the host apps may fail to capture the malicious payloads. Apparently, the stealthy natures of these update attacks poses significant challenges for their detection.

3. Drive-by Download

The third technique applies the traditional drive-by download attacks to mobile space. Though they are not directly exploiting mobile browser vulnerabilities, they are essentially enticing to download “interesting” or “feature-rich” apps.

B. Activation

Activation of android malware relies on particular event. Among all available system events, BOOT_COMPLETED is the most interested one to existing android malware.

C. Malicious Payloads

As existing Android malware can be largely characterized by their carried payloads, Partition the payload functionalities into four different categories: privilege escalation, remote control, financial charges, and personal information stealing.

1. Privilege Escalation

The Android platform is a complicated system that consists of not only the Linux kernel, but also the entire Android framework with more than 90 open-source libraries included, such as WebKit, SQLite, and OpenSSL. The complexity naturally introduces software vulnerabilities that can be potentially exploited for privilege escalation.

2. Remote Control

We also observe that some malware families attempt to be stealthy by encrypting the URLs of remote C&C (Command and Control *Server*) servers as well as their communication with C&C servers. We also find that most C&C servers are registered in domains controlled by attackers themselves. However, we also identify cases where the C&C servers are hosted in public clouds.

3. Financial Charge

One profitable way for attackers is to surreptitiously subscribe to (attacker-controlled) premium-rate services, such as by sending SMS messages. On Android, there is a permission-guarded function sendTextMessage that allows for sending an SMS message in the background without user’s awareness. Moreover, some malware choose *not* to hard-code premium-rate numbers. Instead, they leverage the flexible remote control to push down the numbers at runtime.

4. Information Collection

In addition to the above payloads, we also find that malware are actively harvesting various information on the infected phones, including SMS messages, phone numbers as well as user accounts. We consider the collection of users’ SMS messages is a highly suspicious behavior. The user credential may be included in SMS messages.

IV. EXISTING DETECTION MECHANISM

Android Application model has several interesting features. First, application must follow a specific structure, i.e., they must be composed of some basic kinds of components understood by android. This design encourages sharing of code and data across application. Next, interaction between components can be tightly controlled. By default, components within an application are sandboxed by android, and other applications may access such components only if they have the require permission to do so. This design promises some measure of protection from malicious application. However, enforcing permission is not sufficient to prevent security violation, since permission may be misused, intentionally or unintentionally, to introduce insecure data flow [6].

There are two methods about automatic detection of android malware.

A. Misuse Detection

Misuse Detection is signature-based approach by matching the rules and policies. The advantage of this approach is that it has high accuracy. But it is invalid to the new android malware [7]. Fuchs *et al* [8]. Propose *ScanDroid* that extracts security specifications of the application, and apply data flow analysis to check if any of data flows violate them.

B. Anomaly Detection

Anomaly Detection is different from misuse detection. It usually applies machine learning algorithms for obtaining the known malware behavior and predicting the unknown or new malware. But it sometimes cause high false positive [9]. Iker Burguera *et al* [10]. Propose Crowdroid that analyzing data collected of traces from an unlimited number of real user based on crowd sourcing. Crwdroid framework has been demonstrates by analyzing the data collected in the central server.

There are two common analysing methods to investigate the structure and behaviour of android malware.

A. *Static Analysis*

The static analysis method only considers the contents of every application after decompiling the code [3]. This method can reduce the cost and improve the performance, but it will face the great difficulty when meeting the code obfuscation technique [9]. Wei Tang *et al.* [11] propose ASESD that measures the dangerous level of certain permission combination presentation. Fuch *et al.*

[8] propose *ScanDroid* for automated security certification of Android applications. It analyses data policies in application manifest and data flows across content providers.

Dong-Jie Wu *et al* [7] propose DroidMat a static feature-based mechanism to provide a static analyst paradigm for android malware detection with the consideration of detection effectiveness and cross-version capable analysis. The mechanism considers the static information including permissions, deployment of components, Intent messages passing and API calls for characterizing the Android applications behaviour. In order to recognize different intentions of Android malware, different kinds of clustering algorithms can be applied to enhance the malware modelling capability.

B. *Dynamic Analysis*

The dynamic analysis method continuously monitors the various running situation of the malware (such as reading and writing operations, API calls, power consumption, incoming and outgoing network information, and so on) and then constructs some models for detecting malware. However, it is subjected to high cost in deployment and manual efforts [9].

1. Logged behavior sequence: Zhao *et al.* [12] propose *AntiMalDroid* to detect Android malware that use logged behavior sequence as the feature, and construct the models for further detecting malware and its variants effectively in runtime. They also extend malware features database dynamically.

2. System calls: Burguera *et al*[10]. Propose *Crowdroid* that traces the system calls behavior, converts them into feature vectors, and applies k-means algorithm for detecting malware.

3. Dynamic Tainting Data flow and Control flow: Enck *et al.* [13] propose *TaintDroid*, an extension to the Android mobile-phone platform that tracks the flow of privacy sensitive data through third-party applications.

CONCLUSION

There are emerging lots of android malware on android platform. They are very harmful to user. So how to detect android malware quickly and effectively is urgently required and facing big challenge. Existing malware detection mechanisms are capable to detect Repackaging Application. But they don't detect malware apps which download malicious code at update time. So some modification is required in existing malware detection mechanism or may develop malware detection mechanism to detect repackaging as well as detect malware apps which download malware code at update time.

REFERENCES

[1] IDC Survey Report http://www.idc.com/getdoc.jsp? containerId=pr US23946013.

[2] Appbrain Survey Report http://www.appbrain.com /stats/number-of-android-apps

[3] Malicious Mobile threats Report 2012/2013. www.juniper.net/us /.../ 3rd-jnpr-**mobile-threats**-report-exec-summary.pdf

[4] GDATA survey report https://www.gdatasoftware.co.uk/press-center/news/ article/article/3106-android-malware-becoming-an-ep.html

[5] Zhou, Yajin, and Xuxian Jiang. "Dissecting android malware: Characterization and evolution." *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012.

[6] Fuchs, Adam P., Avik Chaudhuri, and Jeffrey S. Foster. "SCanDroid: Automated security certification of Android applications." *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/~ avik/projects/scandroidascaa* (2009).

[7] Wu, Dong-Jie, et al. "DroidMat: Android Malware Detection through Manifest and API Calls Tracing." *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*. IEEE, 2012.

[8] Fuchs, Adam P., Avik Chaudhuri, and Jeffrey S. Foster. "SCanDroid: Automated security certification of Android applications." *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/~ avik/projects/scandroidascaa* (2009).

[9] Luoshi, Zhang, et al. "A3: Automatic Analysis of Android Malware." *1st International Workshop on Cloud Computing and Information Security*. Atlantis Press, 2013.

[10] Burguera, Iker, Urko Zurutuza, and Simin Nadjm-Tehrani. "Crowdroid: behavior-based malware detection system for android." *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011.

[11] Tang, Wei, et al. "Extending Android security enforcement with a security distance model." *Internet Technology and Applications (iTAP), 2011 International Conference on*. IEEE, 2011.

[12] Zhao, Min, et al. "AntiMalDroid: An Efficient SVM-Based Malware Detection Framework for Android." *Information Computing and Applications*. Springer Berlin Heidelberg, 2011. 158-166.

[13] Enck, William, et al. "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones." *OSDI*. Vol. 10. 2010.