IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

Protection of Android Application against Malware Attack

Priyanka Rane[1], **Madhuri Dalal**[2]

Student, Department of Computer Engineering, Mumbai University, Mumbai, India[1]

Assistant Professor, Department of Computer Engineering, Mumbai University, Mumbai, India[2]

Abstract: Mobile malware is rapidly becoming a serious threat. In this paper, we survey the current state of mobile malware in the wild. We analyse the incentives behind 46 pieces of iOS, Android, and Symbian malware that spread in the wild from 2009 to 2011. We also use this data set to evaluate the effectiveness of techniques for preventing and identifying mobile malware. After observing that 4 pieces of malware use root exploits to mount sophisticated attacks on Android phones, we also examine the incentives that cause non-malicious smartphone tinkerers to publish root exploits and survey the availability of root exploits.

Keywords: Android malware, Android detector, Benign apps, Droid Box, Malicious apps.

I. INTRODUCTION

People use smartphones for many of the same purposes as desktop computers: web browsing, social networking, online banking, and more. Smartphones also provide features that are unique to mobile phones, like SMS messaging, constantly-updated location data, and ubiquitous access. As a result of their popularity and functionality, smartphones are a burgeoning target for malicious activities. In order to understand the motives of real mobile malware, we classify the malware in our data set by behaviour.

We find that the most common malicious activities are collecting user information (61%) and sending premium-rate SMS messages (52%), in addition to malware that was written for novelty or amusement, credential theft, SMS spam, search engine optimization fraud, and ransom.

We describe the incentives that promote each type of malicious behaviour and present defences that disincentive some of the behaviours. In particular, malware that abuses SMS messages could be prevented with small changes to the Android and Symbian platforms. We also discuss incentives that we believe will motivate future mobile malware.

II. LITERATURE SURVEY

We consider that there are three critical issues in machine-learning-based Android malware detection. [4] The first issue is the machine learning model used. In this study, we selected deep learning because it can learn high-level representations by associating features from static analysis with those from dynamic analysis, [1] which makes it possible to better characterize Android malware. Our experiments also demonstrated that the deep learning model significantly outperforms traditional machine learning models. Which are described as follows:

[1] Droid Detector: Android Malware Characterization and Detection Using Deep Learning. In this paper Deep learning is a new area of machine learning research that has gained increasing attention in artificial intelligence. In this study, we propose to associate the features from the static analysis with features from dynamic analysis of Android apps and characterize malware using deep learning techniques.

[2] A Machine Learning Approach: Android Malware Detection, Automated Mining and Characterization of Fine-grained Malicious Behaviours in Android Applications, focuses on The problem of using a machine learning-based classifier to detect malware presents two main challenges: first, given an application, we must extract some sort of feature representation of the application; second, we have a data set that is almost exclusively benign, so we must choose a classifier that can be trained on only one class. To address the first problem, we extract a heterogeneous feature set, and process each feature independently using multiple kernels To address the second problem, we use a One-Class Support Vector Machine, which we train using only benign applications.
We have presented a novel machine learning-based malware detection system for the Android operating system. Our system has shown promising results in that it has a very low false negative rate, but also much room for improvement in its high false positive rate. There are a number of possible improvements that could be investigated.

[3] Droid Miner: Droid Miner, which uses static analysis to automatically mine malicious program logic from known Android malware, abstracts this logic into a sequence of threat modalities, and then seeks out these threat modality patterns in other unknown Android apps. Droid Miner can scan a new Android app to (i) determine

whether it contains malicious modalities, (ii) diagnose the malware family to which it is most closely associated, (iii) and provide further evidence as to why the app is considered to be malicious by including a concise description of identified malicious behaviours.

[4] DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In this paper, we propose DREBIN, a lightweight method for detection of Android malware that enables identifying malicious applications directly on the smartphone. DREBIN performs a broad static analysis, gathering as many features of an application as possible.

III. MOTIVATION AND SYSTEM GOALS

Motivations
We motivate our system design by introducing the inner working of a real-world Android malware. It attempts to perform the following malicious behaviours in the background after the phone is booted: stealing users' personal sensitive information (e.g., IMEI and IMSI) and sending them to remote servers, sending and deleting SMS messages, downloading unwanted apps, and issuing HTTP search requests to increase websites' search rankings on the search engine. As illustrated in Figure 1, once the phone is booted, the receiver will send out an alarm every two minutes and trigger another receiver (named "MyAlarmReceiver") by using three API calls: AlarmManager(), getServiceSystem(), and getBroadcast(). Then, MyAlarmReceiver starts a background service (named "MyService") by calling startService() in its lifecycle call onReceive(). Once the service is triggered, it will read the device ID (getDeviceId()) and subscriber ID (getSubscriberId()) in the phone, and register an object handler to access the short message database (content://sms/). Meanwhile, the service monitors changes to the SMS Inbox database (content://sms/inbox/) by calling ContentObserver.onChange() and deleting particular messages using delete (), and also attempts to download unwanted APK files (e.g., "myupdate.apk"). More details can be found in our extended technical report.

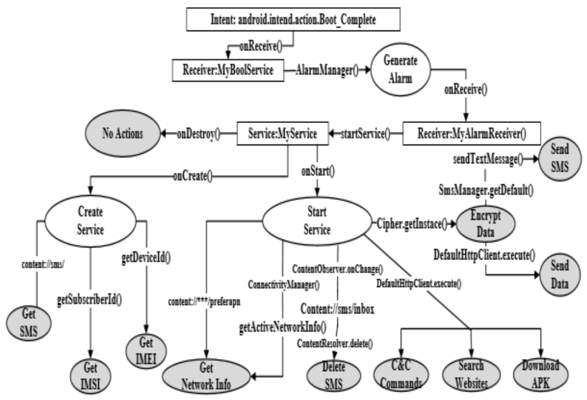

Figure 1. Capabilities embedded in malware from the ADRD family.

The sample achieves its malicious functionalities by mainly invoking a series of framework APIs in order.

IV. THREAT MODEL

We present three types of threats posed by third-party smartphone applications and discuss the security measures that are intended to detect and prevent them. Threat Types The mobile threat model includes three types of threats: malware, grayware, and personal spyware. We distinguish between the three based on their delivery method, legality, and notice to the user. This paper focuses specifically on malware; personal spyware and grayware use different attack vectors, have different motivations, and require different defense mechanisms. Malware. Malware gains access to a device for the purpose of stealing data, damaging the device, or annoying the user, etc.

The attacker defrauds the user into installing the malicious application or gains unauthorized remote access by taking advantage of a device vulnerability. Malware provides no legal notice to the affected user. This threat includes Trojans, worms, botnets, and viruses. Malware is illegal in many countries, including the United States, and the distribution of it may be punishable by jail time. Personal Spyware. Spyware collects personal information such as location or text message history over a period of time. With personal spyware, the attacker has physical access to the device and installs the software without the user's knowledge. Personal spyware sends the victim's information to the person who installed the application onto the victim's device, rather than to the author of the application.

For example, a person might install personal spyware onto a spouse's phone. It is legal to sell personal spyware in the U.S. because it does not defraud the purchaser (i.e., the attacker). Personal spyware is honest about its purpose to the person who purchases and installs the application. However, it may be illegal to install personal spyware on another person's smartphone without his or her authorization. Grayware. Some legitimate applications collect user data for the purpose of marketing or user profiling. Grayware spies on users, but the companies that distribute grayware do not aim to harm users. Pieces of grayware provide real functionality and value to the users.

The companies that distribute grayware may disclose their collection habits in their privacy policies, with varying degrees of clarity. Grayware sits at the edge of legality; its behavior may be legal or illegal depending on the jurisdiction of the complaint and the wording of its privacy policy.
Unlike malware or personal spyware, illegal grayware is punished with corporate fines rather than personal sentences. Even when the activity of grayware is legal, users may object to the data collection if they discover it. Application markets may choose to remove or allow grayware when detected on a case-by-case basis.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

Feature Extraction
To systematically characterize Android apps (i.e., both malware and benign apps), we conduct static and dynamic analyses to extract features from each app, as shown in Fig. 1. All the features fall under one of three types: required permissions, sensitive APIs, and dynamic behaviours.

Among them, required permissions and sensitive APIs are extracted through the static analysis, whereas dynamic behaviours are extracted through dynamic analysis. Specifically, all we need is the installation file (i.e., apk file) of each Android app.

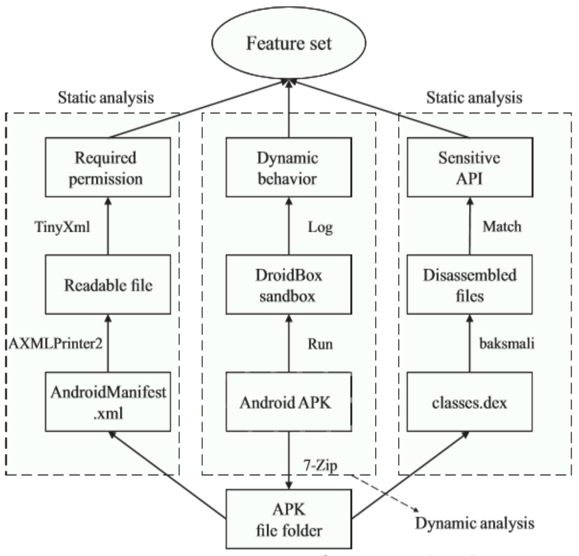

Fig. 2 Feature extraction for an Android app

In the static phase, we uncompress the .apk file with the 7-Zip tool and then focus on parsing the two files AndroidManifest.xml and classes.dex. By parsing the Android Manifest.xml file with the tool AXML-Printer2 and the parser TinyXml, we can obtain the permissions required by the app. For example, android.permission.call phone is the permission required for an app to make a phone call and android.permission.camera is the permission required for an app to access the camera. In this step, we looked for a total of 120 permissions.

By parsing the classes.dex file with the disassembler baksmali, we can know which API functions are called. For example, chmod is a sensitive API that might be used for changing users' permissions on files and ContentResolver;->delete is a sensitive API that might be used for deleting users' messages or contacts. In this step, we looked for a total of 59 sensitive API functions. In the dynamic phase, we install and run each app in DroidBox.

DroidBox is an Android application sandbox that extends TaintDroid, which can execute a dynamic taint analysis with system hooking at the application framework level and monitor a variety of app actions such as information leaks, network and file input/output, cryptography operations, Short Message Services (SMS), and mobile phone calls. In this study, we ran the apps inside DroidBox for a period of time to obtain the executed app actions (i.e., dynamic behaviours) of each app. In this phase, we monitored a total of 13 app actions. For instance, action sendnet is the action that sends data over the network, action phonecalls is the action that makes a phone call, and action sendsms is the action that sends SMS messages.

In this way, we obtained a total of 192 features for each app through static and dynamic analyses. Note that each feature is binary, indicating that when a feature occurs in an app, its feature value is 1; otherwise, its feature value is 0. In addition, all the tools (i.e., 7-Zip, AXMLPrinter2, TinyXml, baksmali, and DroidBox) referred to in this section are open source for use by the public.

V. MALWARE DETECTION WITH PERMISSIONS

Contrasting permission pattern
Considering a training dataset, it is split into two subsets: One containing malicious android applications and other containing clean applications. By applying Associative Rule mining on these two subsets, three types of permission patterns were created:

(1) Malicious Permission Patterns: These are unique frequently required permission patterns found only in malware dataset. Hence, the support degree of their item sets in clean dataset is 0.

(2) Clean Permission Patterns: These are unique frequently required permission patterns found only in clean dataset. Hence, the support degree of their item sets in malware dataset is 0.

(3) Commonly required Permission Patterns: These are the frequently required permission patterns that are found in both the dataset. In such case these patterns have different support degree in two datasets These frequent item-sets are called as Contrasting Permission Patterns.

If an unknown application has more malicious permission patterns than the clean patterns, then it can be said that the application is a malware and vice versa. But if the unknown application contains commonly required permission patterns the difference in support degrees in respective datasets is considered for distinguishing the application. Suppose the common permission pattern has a higher support degree in malware dataset then the app is more likely to be a malware.

VI. CONCLUSION

In this paper, a malware detection in android platform using Contrasting permission patterns is presented. These contrasting permission patterns and their support degrees are the characteristics that help us in selective and classifying malicious applications from clean applications.

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 10, October 2016

REFERENCES

1. Zhenlong Yuan, Yongqiang Lu, and Yibo Xue "DroidDetector: Android Malware Characterization and Detection Using Deep Learning " Volume 21, Number 1, February 2016
2. Chao Yang1, 'et al' "DroidMiner: Automated Mining and Characterization of Fine-grained Malicious Behaviors in Android Applications" .
3. Daniel Arp1 , Michael Spreitzenbarth2 , Malte Hubner ¨ 1 , Hugo Gascon1 , Konrad Rieck1 "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket".
4. Justin Sahs and Latifur Khan "A Machine Learning Approach to Android Malware Detection" Volume 21, Number 1, February 2016
5. Chinmay Sanjay Kapare 'et al' "DroidDetector: An Android application based on Contrasting Permission Patterns" Volume 5 Issue 3, March 2016.
6. A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild, in Proceedings of the 1st ACM Workshop on Security and Privacy in SmartphonesandMobileDevices"(SPSM),2011,pp.3–14.
7. Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, Hey, You, "Get off of my market: Detecting malicious apps in official and alternative Android markets, in Proceedings of the 19th Annual Symposium on Network and Distributed System Security (NDSS)", 2012.
8. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, Riskranker: "Scalable and accurate zero-day Android malware detection, in Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services" (MobiSys), 2012, pp. 281–294.
9. V. Rastogi, Y. Chen, and X. Jiang, Droidchameleon: "Evaluating Android anti-malware against transformation attacks, in Proceedings of the 8th ACM Symposium on Information, Computer and Communications Security "(ASIA CCS), 2013, pp. 329–334. .
10. S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna, Execute "Analyzing unsafe and malicious dynamic code loading in Android applications, in Proceedings of the 21th Annual Symposiumon Network and Distributed System Security" (NDSS), 2014.