Malware Detection In Mobile Through Analysis of Application Network Behavior By Web Application

[1]Himgouri P. Barge,[2]Pankaj R. Chandre
[1] Student ME(Computer),[2]Prof. ME(Computer)
[1]Department of Computer Engineering
[1]Flora Institute of Technology, Khopi, Pune, India

***Abstract*– This system detects the mobile malware by analyzing suspicious network activities through the traffic analysis. In our system, the detection algorithms which we are using are works as modules inside the Open Flow controller, and the security rules can be imposed in real time. Here, we are using new behavior-based anomaly detection system which is used for identifying meaningful deviations in a mobile application's network behavior. Here, we are trying to detect a new type of mobile malware with self-updating capabilities. This kind of malware neither identified by using the standard signatures approach nor applying static or dynamic analysis methods. The detection is completely based on the application's network traffic patterns only. Here we are using Semi-supervised machine-learning techniques for learning the normal behavioral patterns.**

***IndexTerms*– Android Mobile Malware, Network Traffic, Machine learning, Smart-Phones Security**.

I. INTRODUCTION

Now a days there is momentous growth in the popularity of smart phones and the number of available mobile applications, the amount of malware that damage users or compromise their privacy has also increased[1][2]. Smart phones are most widely used in every aspect of our life including personal, official, political, social and educational. That's why, smart phones are used not only for making calls, but also making important business decisions in professional life, internet services such as online shopping, online ticket booking, online social network etc. Large numbers of apps are available for various mobile operating systems to these services. Net banking is one of the most increasing areas where mobile devices (i. e. banking app.) are widely used now days.

Keeping mobile secure is important task, so secure the mobile devices from malicious files and applications. There can be three properties of security based data security:

1. Confidentiality is nothing but preventing the data from unauthorized users. The data which is stored in mobile device should not be used by anyone else without mobile user.
2. Integrity means prevention of modification of information or data by unauthorized user.
3. Availability is nothing but preventing unauthorized access of information. Mobile's applications and its services must be available to the authorized mobile user at any time.

Use of smart phones for sensitive and important services like net banking is increasing state so mobile security is most important. Mobile security is very challenging task because the malicious applications are increasing every day. So basic understanding about various viruses and malwares for mobile user is very important.

Malware is a software program or mobile application which exhibits malicious behavior [3][4]. This is a general term used to refer to number of intrusive applications and are characterized into virus, bonnets, worm, Spyware, Adware, Root kit as well as Trojan horse based on their behavior of affecting the mobile device. Malwares are intentionally used by black hat hackers for accessing the personal data and sensitive information of a mobile device. They also uses the malwares to collect the sensitive or important data of a corporate or government websites.

A virus is a program that destroys the personal data as well as applications of mobile device. Viruses have the characteristics of self-modification, encryption which makes its detection very difficult to an antivirus application. A bonnet is very bad threat to the information society at present. It can control Internet Relay Chat (IRC) or can also send spam emails. Bonnets are used to steal the data from a computer such as different login IDs, application serial number, and financial information such as credit cards number etc. worms are the stand alone software applications that may runs without a host and can have the capacity to self –replicate and propagate around in the network. Worms can also used by the bonnets for controlling the computers connected to internet which are used by spam sender for sending junk emails. The worm is risk-free because it does not perform any malicious activity but because of the continuous searching for another Bluetooth device, the battery life will reduces. A spyware is a software program that collects person and organization information and can also send the information to another entity without the permission of user. Key logger is Spyware software which is used to remember the key struck on a keyboard. It remembers all key interactions without the permission of user. Hackers use this software to obtain passwords and usernames of a computer. Trojan horse is a category of Spyware that can arrive at a computer system via online games, internet driven applications.

It can provide the access of computer which is targeted to a hacker that can use the machine as a part of bonnet, can steal the sensitive data, it can also download any malicious files, and it can upload any file into target machine. It can also

remotely control the whole computer system. Generally, any malware can be downloaded and executed on a device using such a "remote payload" technique. In the case of Drop dialer Trojan, the downloaded malicious package sent SMS messages to premium-rate numbers. The download action can be scheduled for both specific and random time in the future, or may be initiated remotely by sending a command message to the devices, using, for instance, Google's push notification service (GCM or former C2DM). Several different techniques allowing Android application update from a remote location exist. The details of these techniques are given in Self-updating malware section [5].

Such self-updating malware cannot be found by standard static techniques as the original version of the application is wholly benign and it does not contain any malicious code. Transforming a benign application into a malicious application using dynamic code it makes the static analysis of the host application virtually irrelevant. Detection by using dynamic analysis can be avoided by using a time delayed or filtered deployment of the malicious payload. For example, on first few days, an attacker may offer benign update, and later on switches to the malicious one or offers the updation to a certain geo-location, filter by carrier, device type, OS version, etc. Moreover, a sophisticated attacker can use server-side polymorphism utilizing obfuscation and encryption to completely discourage and complicate the dynamic/static analysis efforts. It is also hard to identify this type of emerging malware since the self-updating technique is frequently used by legitimate applications for benign purposes as well (e.g., to upgrade already installed games with new levels, bug fixes, update of the detection engine of anti-virus applications, etc).

Our main method is based on watching application which is running on a device, learning the models which represent their "normal" network behavior, and detecting any differences from the learned patterns [6]. Such continuous watching of an application's network behavior could offer suggestions about some unexplained changes in the application's network behavior any time they occur.

The new system supports two main use cases. The first case is to applications already installed on a device and the second, to newly downloaded and installed applications. In the first case, the network traffic pattern of an application can be changed due to: (1) a difference in the user's behavior; (2) an application update to a new benign version; or (3) a malicious attack. In this case the system's idea is to detect the variation in the application's traffic pattern and correctly categorize it to one of the three above mentioned reasons. In the second case, the system's aim is to discover whether the innovative application is really a modification of another application with some new (possibly malicious) behavior. This is done by comparing with the network behavior patterns and behavior of the same application as observed on other devices that are already running the application or are characterized with the behavior of a similar application.

II. Related Work

A. A Survey of Mobile Malware In The Wild

A.P. Felt and his colleague [3] published this paper, this data set to evaluate the effectiveness of techniques for preventing and identifying mobile malware. After observing that 4 pieces of malware it use root exploits to put 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

B. Practical Analysis of Smart phone Security

Woongryul Jeon and his partner [4] used general developments in mobile technologies have formed a new kind of device, a programmable mobile phone, the smartphone. Generally, smartphone users can program any application which is customized for needs. Moreover, they can distribute these applications in online market. Due to this security vulnerabilities may causes. Therefore, in this work, we analyze security of smartphone based on its environments and describe countermeasures.

.

III. Implementation Details

In this paper, we have developed an Antivirus application to detect the infected files. The application is in two modes. First is a standalone application. User can download this application from the internet and can install it in mobile device. This application has two versions, one is developed in J2SE and another is in J2ME. Based on the application requirements, mobile devices can use the version and compatibility with mobile device. Both versions provide the facility to scan the device. J2SE version has more facilities than J2ME. The second mode of application is a web based application. The application can be used when the device is connected to internet. User can register his/her device, can scan file online, can report a file as thread, and can also download updates for Antivirus application. Securing mobile devices is a difficult task. People use mobiles for net banking, online shopping and many other works. Security of mobile devices is very important because, if a device has infected file(virus) , its data will no longer be safe , virus can control the mobile data , also can send the critical data while connected to internet.

A. Web Client:

To study various antivirus solutions which are available in the literature. To design and develop an application for detecting the infected files (virus) in a mobile device using J2SE and J2ME. Extend the application as a web application that will scan the mobile files online, other services to report a file as thread. The client side software monitors the applications that are already installed and running on the device, learn their user specific local models and detects any deviations from the observed "normal" behavior. The Collaborative Learning Server is liable for collecting data reported by various mobile devices and deriving collaborative models, which represents the common traffic patterns of numerous users for each application.

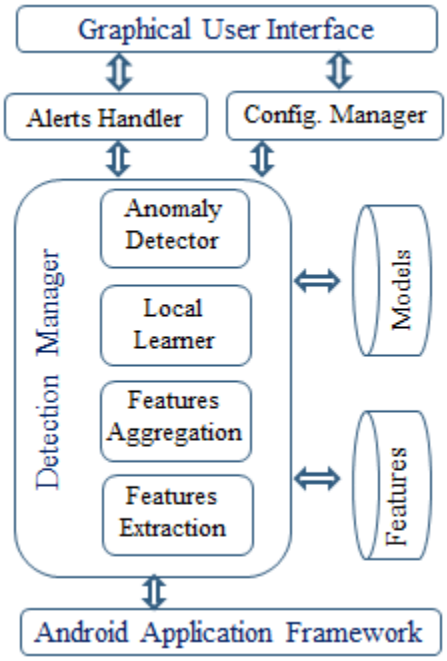

Figure 1. Architecture of malware detection system.

B. System Architecture

The system architecture consists of a client component installed on the Android mobile device and a server component. The task of the client-side software is to watch the applications that are already installed and running on the device, study user specific local models and to detect the deviations from the observed "normal" behavior. The Collaborative Learning Server is answerable for collecting and aggregating data reported by various mobile devices and deriving collaborative models, which represents the common traffic patterns of many users for each application. Any application or file which copies the files into root directory of mobile device or it makes new directory while user may not be aware about it is Malware application. There are many heuristic rules to report a suspicious file as Malware. Many Malwares are formed, revised and edited on daily basis so their detection difficult. That's why; it is not easy task to sense all the malicious applications of a device.

The analysis and discovery of Android malware has been a vivid area of research in the recent years. Several techniques have been proposed to counter the growing amount and sophistication of this malware. An summery of the current malware is provided in the studies of Felt and Zhou.

1) Feature Extraction

As mentioned above, this module is responsible for extraction (i.e., measuring and capturing) of the defined list of features for each running application at each time period. For this purpose it uses the APIs provided by the Android Software Development Kit (SDK). Below there is a list of the currently extracted features:

- Sent\received data in bytes and percent;
- Network state (Cellular, Wi-Fi or "No network");
- Time (in seconds) since application's last send\receive data;
- Send\receive mode (eventual\continuous) – derived from "since-last-send\receive-seconds", i.e., if the last send or receive data event was detected less than a specified number of second ago, the corresponding (send or receive) mode is continuous, otherwise it is eventual;

The time period for this process is a configurable parameter. For the initial experiments with the system we set it to 5 seconds, however it is subject to change according to the results of future evaluation experiments.

2) Features Aggregation

The purpose of this module is to present a brief representation of the extracted application's traffic data. For this purpose, many aggregation functions were defined. The instances of the aggregated data are used to induce machine-learning models representing an application's behavior and for further anomalies detection. To get an idea of the usefulness of the various features for our problem, in the current work an extended list of possible aggregated features was defined and evaluated. According to the evaluation results, a preliminary list of the few most useful features is determined in the Evaluation Section.

As similar to the extraction, the aggregation time period is a configurable parameter. For the current experiment it was set to 1 minute.

3) Local Models Learning

Our main aim is to learn user specific network traffic patterns for each application and find out if meaningful amend occurs in the application's network behavior [6]. This assignment relates to the family of semi-supervised anomaly detection problems, which considers that the training data has samples for "normal" data examples only. These types of problems can be

solved, by using one-class support vector machines (SVMs), the local outlier factor (LOF) method, or clustering based techniques [7, 8].

In this paper we decided to convert the semi-supervised learning problem into a set of supervised problems for which many well established and quick algorithms exist. For this purpose we follow the "cross-feature analysis" approach presented in Huang [9] and then further analyzed by Noto in [8]. Both of these works have been found that this approach successful and useful for anomalies detection. However, [10], only considers features with discrete values and [11] mainly focuses on methods for combining the results of multiple feature predictors to make the final decision about instance normality. In our problem most of the features are numerical and an efficient implementation is desired as it is supposed to run on mobile phones.

C. Anomaly Detection

This module is answerable for the online analysis of an application's network behavior and the discovery of deviations from normal patterns. The procedure used for testing each individual instance is further described.

When a feature's vector representing a normal event is tested against C_i, there is a higher probability for the predicted value to match (for discrete features) or very similar to the observed value. However, in the case of a vector representing abnormal behavior, the probability of this kind of match or similarity is much lower. Thus, by applying all the features models to a tested vector and by combining their result, a decision about vector normality can be derived. The more unusual the predictions are of the true values of the corresponding features, the more likely that the observed vector comes from a different distribution than the training set (i.e., represents an anomaly event).

1) Detection using Static Analysis

The first approach is for sensing Android malware have been inspired by concepts from static program analysis [12]. Several methods have been proposed that statically inspect applications and disassemble their code. For example, the Kirin method checks the permission of the applications for indications of malicious activity. Similarly, Stowaway analyzes API calls for detecting over privileged applications and Risk Ranker statically identifies applications with different security risks. General open-source tools for static analysis are Small and Android, which enable dissecting the content of applications with little effort.

2) Detection using Dynamic Analysis

A second branch of research has studied the discovery of Android malware at run-time [13]. Most notably, are the analysis system Taint Droid [14] and Droid Scope that enable dynamically monitoring applications in a protected environment, where the first focuses on taint analysis and the later enables detection at different layers of the platform. While both systems provide detailed information about the behavior of applications, they are technically too involved to be deployed on smart phones and detect malicious software directly. As a result, dynamic analysis is applied for offline detection of malware, such as scanning and analyzing large collections of Android applications. For example, the methods Droid Ranger, Apps Playground, and Copper Droid have been successfully applied to study applications with malicious behavior in different Android markets. A similar detection system called Bouncer is currently operated by Google. Such dynamic analysis systems are suitable for filtering malicious applications from Android markets. Due to the openness of the Android platform, however, applications may also be installed from other sources, such as web pages and memory sticks, which require detection mechanisms operating on the smart phone. Paranoid Android is one of the few detection systems that employs dynamic analysis and can spot malicious activity on the smart phone. To this end, a virtual clone of the smart phone is run in parallel on a dedicated server and synchronized with the activities of the device. This setting allows for monitoring the behavior of applications on the clone without disrupting the functionality of the real device. The duplication of functionality, however, is involved and with number of smart phones in practice operating Paranoid Android at large scale is technically not feasible.

IV. Evaluation Result

To test the proposed system, we gather 12 datasets, 6 with train and test records from the same application version and 6 from different application versions, was used. Additionally, the system was tested with one self-written and five real malware applications. The detection rate of the Decision Table and REP Tree algorithms in conjunction with the features subset #1 and #2 correspondingly, on the evaluated datasets are presented in following Table.

It can be seen that for all the malware applications, the high level deviations (60-100%) were detected. Furthermore, variations at various levels were detected in most cases when they learned models were tested with instances from a different application version. The unnoticed versions of Facebook and WhatsApp applications can be explained by very few or no network-related changes in the considered application versions. Furthermore, the detection rate for all the cases when they learned models were tested with instances from the same application version are below the defined "anomaly acceptance" rate of 20 percent for the Decision Table algorithm and of 25 percent for the REP Tree algorithm. Thus, using the Decision Table algorithm's "anomaly acceptance" rate of 20 percent, the estimated method's accuracy on the test data is the following: TPR=0.82, FPR=0.0 and Total Accuracy=0.875. For the REPTree algorithm with the determined "anomaly acceptance" rate of 25 percent, the estimated accuracy on the test data is even higher: TPR=0.91, FPR=0.0, and Total Accuracy=0.94.

Consider the surprisingly high detection rate in the several real malware applications. Note that in the self-written malware the 100% detection rate is not surprising, as the benign and malicious versions are significantly different in their network usage patterns. However, in the case of the real malware applications, the 100% detection rate is not obvious. In the applications infected with the Trojans, the main application's functionality is preserved and some new functionality is added. Thus, some part of the data

related to the old functionality might be expected to remain unchanged. This is actually the case with the Fling application where online mobile advertisements are displayed while the application is in the phone's frontend in both versions. Thus, the records corresponding to the time when the game was actually played were less affected by the Trojan functionality and thus the observed detection rate is "only" 60%. Analysis of the data aggregated from the benign and malicious versions of the evaluated applications shows that the significant differences are caused by a background process that is running even when an application is not active and performs multiple connections (or connection attempts) with the server at constant time intervals. This behavior has a significant effect on such features, such as avg. sent\received bytes, number of sent\receive events, global outer\inner sent\receive intervals, and others. Most of the mentioned and significantly influenced features are contained in the utilized features subsets and this explains the high detection rate.

	Application Name	Detected anomalous records(%)	
		Decision Table	REP Tree
Regular applications	Different version		
	twitter	57.8	62.2
	twitter	78.2	34.8
	facebook	0.5	3.3
	google	80.9	87.2
	whatsapp	16.7	28.9
	Same version		
	google	0.0	0.0
	google	0.0	15.0
	gmail	14.8	22.2
	facebook	16.0	15.7
	firebox	20.0	22.8
Malware apps	Self-Written malware		
	Snake	100.0	100.0
	Real malware		
	Fling	63.6	66.8
	Open Sudoku	100.0	100.0
	ShotGun	97.0	89.5
	Squibble	90.0	95.0
	Crazy Fish	100	100.0

V. CONCLUSION

In this paper, we presented a new system for detecting meaningful difference in a mobile application's network traffic patterns that can be used for detecting an emerging type of malware with self-updating capabilities that allow for stealing user data or spying on users. The presented system, although initially planned for host-based of a client server system, is a fully-functioning, stand-alone monitoring application for mobile devices, which can be used alone or in conjunction with other methods. One of the main capabilities of the proposed system is guarding of mobile device users from malicious attacks on their phones. The detection is performed based only on the application's network traffic patterns.

We will further study the properties of mobile malware, investigate more malware detection techniques and explore the possibilities of employing them in the context of SDN as future work. In addition, we plan to take better advantage of GENI infrastructure and test our system at an even larger scale in order to optimize our system design.

VI. ACKNOWLEDGEMENT

I want to thank all the people who helped me in different way. Especially I am thankful to my guide Prof. Pankaj R. Chandre for his continuous support and guidance in my work. Also, I would like thank our HOD Prof. S. A. Joshi , and Principal Dr. A. S. Padalkar.

REFERENCES

[1] Mylonas A, et. al, "Delegate the smartphone user? Security awareness in smartphone platforms", Comput Secur; 2012.
[2] Mylonas A, et. al, "Assessing privacy risks in Android: a user-centric approach", 2013.
[3] A.P. Felt, et al., "A Survey of Mobile Malware In The Wild," 1st Workshop on Sec. & Privacy in Smart phones and Mobile Devices, 2011
[4] L. Chekina, et al., "Detection of Deviations in Mobile Applications Network Behavior," available on http://arxiv.org/corr/home
[5] Woongryul Jeon, et. al, "A Practical Analysis of Smart phone Security", Springer, pp 311-320.

[6] Amos B, et. al, "Applying machine learning classifiers to dynamic android malware detection at scale.", In: International Conference on Wireless Communications and Mobile Computing (IWCMC).

[7] Luoxu Min et. al, "Runtime-based Behavior Dynamic Analysis System for Android Malware Detection", Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012), pp 233-236.

[8] Chandola V, et. al, "Anomaly detection: a survey. ACM Comput Surv 2009";41(3):1e58.

[9] Noto K, et. al, "Anomaly detection using an ensemble of feature models. In: Proceedings of the 10th IEEEinternational conference on data mining"; 2010.

[10] Huang YA, et. al, "Cross-feature analysis for detecting ad-hoc routing anomalies. 23rd international conference on distributed computing systems. Washington, DC, USA", IEEE Computer.

[11] J. Jamaluddin, et. al, "Mobile phone vulnerabilities: a new generation of malware," in IEEE International Symposium on Consumer Electronics, Sept 2004, pp. 199–202.

[12] D. Damopoulos et. al, "iSAM: An iPhone stealth airborne malware," Future Challenges in Security and Privacy for Academia and Industry, pp. 17–28, 2011.

[13] Daniel Arp, et. al, "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket", 14, 23-26 February 2014, San Diego, CA, USA.

[14] Enck W et. al, "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones.", In: Proc. of USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp393–400.

[15] Himgouri P. Barge, et. al, "Mobile malware detection through analysis of web application network behavior", International Journal of Computer Science and Mobile Computing Vol.3, December- 2014