MACHINE LEARNING FOR AUTOMATIC MALWARE SIGNATURE GENERATION AND CLASSIFICATION

[1]Dr Bhoopathy V, [2]Veena Rani, [3]J.Sushma
[1]Professor, [2,3]Assistant Professor Department of Computer Science and Engineering,
Malla Reddy College of Engineering, Hyderabad

ABSTRACT:

Malware has always been a problem in regards to any technological advances in the software world. Thus, it is to be expected that smart phones and other mobile devices are facing the same issues. In this paper, a practical and effective anomaly based malware detection framework is proposed with an emphasis on Android mobile computing platform. A dataset consisting of both benign and malicious applications (apps) were installed on an Android device to analyze the behavioural patterns. We first generate the system metrics (feature vector) from each app by executing it in a controlled environment. Then, a variety of machine learning algorithms: Ada Boost, Gradient Boost, Gaussian Naive Bayes, Random Forest, and Decision Tree are used to classify the app as benign or malware. Each algorithm is assessed using various performance criteria to identify which ones are more suitable to detect malicious software. The results suggest that Random Forest and Decision Tree provide the best outcomes thus making them the most effective techniques for malware detection.

Keywords: Benign, Malicious, Ada Boost, Gradient Boost, Gaussian Naive Bayes, Random Forest, Decision Tree

1. INTRODUCTION

Malware is short form of malicious software. The term "Malware" is commonly used now-a-days to refer a variety of forms of intrusive software such as viruses, worms, spyware, trojan horses etc., The common function of malware is that they are specifically designed to damage, disrupt, steal, and some other illegitimate actions. Malware can infect any computing machines and the prevention of the malware have been well studied for personal computers. As the current generation uses smartphone more, the chances of malware intrusion is more compared to previous generation. Users download files like apk's from untrusted third parties which may include many malware executable files. They get downloaded along with the users downloaded file and get executed in the background of our mobile device. So, our solutions for finding malware in the mobile platform is less compared to finding malwares in Personal Computers.

2. LITERATURE SURVEY

2.1 INPUT VALIDATION TECHNIQUES

Input validation based attacks occur due to lack of inspection or insufficient inspection on the input provided by the clients. The two most common input validation based attacks are SQL injection attacks and Cross site scripting attacks.

2.1.1 SQL Injection Attacks

SQL Injection Attacks SQL language being a very rich language, paves the way for a number of attacks. The first real existence of SQL injection was explored in 1998 in a magazine Phrack . In this magazine an article on implementing attacks using SQL injection was explained by Rain Forest Puppy (RFP). An extensive study on the SQL injection attack and its classification was provided by Halfondetal.,. This work classified SQL injection attack based on the type of user input, namely Injection through cookies, injection through server

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-11, 2017
DOI: 10.21276/ijcesr.2017.4.11.44

variables and second order injections. The attack was also categorized based on the goal of the attacker, namely identifying injectable parameters, performing database finger-printing, determining database schema, extracting data, adding or modifying data, performing denial of service, evading detection, bypassing authentication, executing remote commands, performing privilege escalation.

Dynamic Analysis Dynamic analysis is done during runtime of a web application. The input given by the user is extracted and is combined with the code of the application to check if there is an intended attack. Dynamic analysis unearths most of the real time attacks at the cost of response time, and thereby affecting the performance of the server. Gregory et al.,andZhendongsu et al., implemented systems that checked queries at runtime to see if they conform to a model of expected queries. Taint based approaches like the Security gateway, implemented by David Scott et al., is a proxy filtering system that enforces input validation rules on the data flowing to a web application to detect and prevent SQLIA and XSS Attacks. KonstantinosKemalis et al., developed a prototype SQL injection detection system (SQL-IDS). This system monitored Java-based applications and detected SQL injection attacks in real time. The proposed detection technique was based on the assumption that injected SQL commands had differences in their structure with regard to the expected SQL commands that were built by the scripts of the web application.

2.1.2Cross-Site Scripting Attacks

Certain works provide client side solutions for CrossSite Scripting attacks where a change in the browser code or a fire wall set up is recommended. The disadvantages of these works are that client/user involvement is needed and the clients need to have technical knowledge for implementing these solutions. Peter et al., implemented a system named Secure Web Application protocol (SWAP), that uses proxy layer and provides a server-side solution to detect and prevent XSS attack. It also uses a modified browser that detects script content that is included by the user. Noxes is a client side solution for XSS attacks. It creates a personal firewall on the client side and checks every outgoing request and incoming response. It uses both manual and automatically generated rules to prevent XSS attacks on the client side.

2.2 VULNERABILITY DETECTION IN SMARTPHONE APPLICATIONS

Recent researches carried out in the field of Smartphone security have concentrated mostly on Android operating system (OS). The security issues in Android are a concern owing to the fact that it is open source and developing of android application is very easy. Adrienne Porter felt et al., surveyed and classified mobile malwares based on their behavior. Some of the commonly noted behaviors were, collecting user information, sending premium-rate Short messaging Service (SMS), malware written for amusement, for credential theft, search engine fraud and ransom. Different defensive mechanisms that were available against these malwares were discussed and evaluated. Threats have been classified as physical threats, application-based threats, network-based threats, web-based threats, and mobile vulnerabilities by Jalaluddin Khan et al.,. They presented a survey on already existing privacy threats and defensive mechanisms. Biometric authentication was suggested as a defensive mechanism for these attacks. AlexiosMylonasetal.,classified users based on their trust and awareness on web applications. They provided a database guideline for trusted repository, license and agreement, poor security checks, pirated applications and the users' perception. Another type of attack called the android scraping was reported by Ken Munro et al.,. Android scraping is the unauthorized extraction of information from Smartphones. Various methods for scraping data were discussed, namely

2.2.1 Behavioral Analysis

Jacob et al., conducted a survey of different reasoning techniques used by behavioral detectors. They classified these detectors into main families, namely stimulation based detectors and formal detectors depending on their data collection and data interpretation mechanisms. Timothy K. Buennemeyer et al., in their work introduced capabilities developed for

Battery-Sensing Intrusion protection system (BSIPS) that raised an alert when an abnormal current change occurred. They developed a Correlation Intrusion Detection Engine (CIDE) that provides power profiling for mobile devices and a correlated view of B-SIPS and Snort alerts. The B-SIPS client was designed with customizable features to accommodate varying user skill levels. Users with advanced computer skills could configure the application to provide more refined detection and alert information. The basic users could effectively operate the system with default settings.

2.2.2 Signature Based Analysis

Griffin et al., used signature-based method that depended on the identifying unique signatures that defined the malware. This work used string signatures, each of which was a contiguous byte sequence that potentially could match many variants of a malware family. A similar work had been proposed by Yu Feng et al.,. A system named Apposcopy, which used a semantics-based approach to identify a prevalent class of Android malware that stole private information, was developed. It used a high level language for specifying signatures of malware families and performed static analysis to decide if an application was benign or malicious. G.Suarez-Tangil et al., introduced a system called dendroid based on text mining and information retrieval. This system automatically classified malicious applications as families based on common code structure. For 31 family classification this system used vector space model. This also provided relationship among families and their evolution from common ancestors. If an unknown application was installed, the system would automatically identify its family and assign this application to that family. The limitation of these systems was that it can be applied only to known malicious code.

III.SYSTEM ANALYSIS

A. Existing System

In this existing system, But the main point is now days malware writers have techniques such as polymorphism which can fool these pattern-based methods very easily. But there is a different approach that can try that is malware detection behaviour-based techniques. This approach is very different than pattern based as here check behaviour and collect its information by executing it and that information we will use inorder to detect malware or not. In general, most of the time we can observe that new malware's that arise on a regular basis are just a little changed version of older malware using some clever techniques. So, this should be clear by now that techniques based on behaviour of malware are a great choice in order classify or detect malware.

Limitations

- Efficiency levels are very low

B. Proposed System

For each newly collected unknown Android app, it will be first parsed through the unzipper and decompiles to get the smali codes, then its API calls will be extracted from the smali codes, and the relationships among these API calls will be further analysed. Based on the extracted features and using the constructed classification model, this app will be labelled as either benign or malicious. Classification is the only technique used to categorize the given app as malicious or the genuine one.

Advantages

- Efficiency levels are high

C. Hardware Requirements

- System :Windows
- HardDisk : 120 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram : 4GB

D. Software Requirements

- Python
- Included development tools: Jupyter Notebook, Spyder etc.,
- Compatible tools: PyCharm

E. Architecture

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-11, 2017
DOI: 10.21276/ijcesr.2017.4.11.44

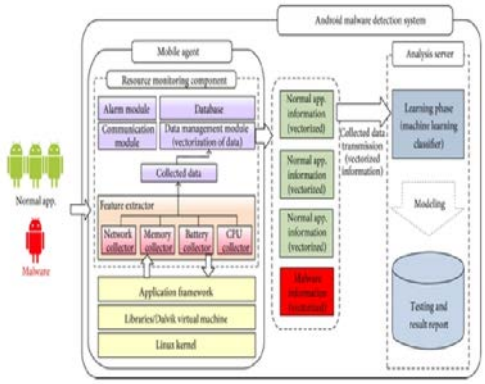

Fig. 1: Architecture

IV. MODULES AND DESCRIPTION

A. Modules

1. Mobile Agent

• Resource Monitoring Component

2. Analysis Server

B. Modules Description

1. Mobile Agent

Mobile Agent is the Module which is used to send the data about the app when the features of it are extracted, to the Analysis Server. In this Module, the normal app and the malicious app are given as input. After the input is given to it, the feature extractor does the process of extraction of feature by sending the app for execution where it is checked for the libraries and other framework in the Linux kernel. The data is sent to the "Resource Monitoring System".

Resource Monitoring System:

The extracted features are sent to this Module. This module consists of Network Collector, Memory Collector, Battery Collector, CPU Collector. The malicious file may effect one of the above resources. Therefore, the features which effect those, are extracted and sent to the Data Management Module. Further, the data is vectorized and the vectorized data is sent to the Analysis Server.

2. Analysis Server

In this Module, the vectorized data, collected from the Mobile Agent, is received. The Vectorized data may be of either Malicious App data or Genuine App Data. The machine gets trained with the dataset given in prior by the admin. After the training phase of machine, the data from the above module is entered into the testing phase where the Classification Algorithms are applied. Those Algorithms categorize the benign and malicious data. Finally the Prediction is shown, which algorithm shows more accuracy in-terms of Classification.

V. IMPLEMENTATION

1. Data Collection

In this we have used a dataset which is obtained from the *******. It contains all the features for an app being malicious to predict whether the given app is benign or not. All the features represent the app as malicious or benign based on many values present in the dataset. There are many datasets present in Kaggle, Github and other repositories which are used not only Malware classification but also in Genre Classification etc., As we wanted to use for Malware and Benign Classification, this dataset is very helpful and shows high accuracy in finding or in prediction of Malware.

2. Data Description

Our dataset contains many features which are used to categorize the malware and benign apps. There are a total of 54 features which help in classification of them. But the main features which our project extracted for the Classification is 13. Some of the features are:

Subsystem, Dll Characteristics, Size of Optional Header, Resources Max Entropy, Sections Max Entropy, Major Subsystem version, Size of Stack Reserve, Major Operating SystemVersion etc.,

Outputs are represented in the form of percentages of accuracy values of each Classification Algorithm. The final accuracy shows how best the Algorithm is interms of prediction.

VI. FUTURE ENHANCEMENT

As we know the same malware doesn't continue to be long time. New forms malware and its intrusion is possible in the days ahead. So, we would like to modify the following in our project.

• Increase the size of the training set.

• To use Algorithms which gives more accuracy if any new kind of malware evolves.

• Implement unique techniques to control the malware.

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-4, ISSUE-11, 2017
DOI: 10.21276/ijcesr.2017.4.11.44

VII.CONCLUSION

We can predict that it is possible to reduce the number of malware by being analysed for mobile malware detection, while maintaining high effectiveness and accuracy. This technique has been designed to extract not only significant permission detection analysis, but also many other modes for creation of malware through a systematic approach.

REFERNCES

[1] IDC, "Smartphone os market share, 2017 q1." [Online]. Available: https://www.idc.com/promo/smartphone-market-share/os

[2] Statista, "Cumulative number of apps downloaded from the google play as of may 2016." [Online]. Available: https://www.statista.com/ statistics/281106/number-of-android-app-downloads-from-google-play/

[3] G. Kelly, "Report: 97% of mobile malware is on android. this is the easy way you stay safe," in Forbes Tech, 2014

[4] G. DATA, "8,400 new android malware samples every day." [Online]. Available:https://www.gdatasoftware.com/blog/ 2017/ 04/29712-8-400-new-android-malware-samples-every-day

[5] Symantec, "Latest intelligence for march 2016," in Symantec Official Blog, 2016.

[6] 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. ACM, 2012, pp. 281–294.

[7] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security. ACM, 2011, pp. 627–638.

[8] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "Taintdroid: an informationflow tracking system for realtime privacy monitoring on smartphones," ACM Transactions on Computer Systems (TOCS), vol. 32, no. 2, p. 5, 2014.

[9] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and ¨ C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS), 2014.

[10] C. Yang, Z. Xu, G. Gu, V. Yegneswaran, and P. Porras, "Droidminer: Automated mining and characterization of fine-grained malicious behaviors in android applications," in European Symposium on Research in Computer Security. Springer, 2014, pp. 163–182.

[11] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis–1,000,000 apps later: A view on current android malware behaviors," in Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on. IEEE, 2014, pp. 3–17.