International Journal of Engineering & Technology, 7 (4.6) (2018) 410-414

SPC

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Neural Network Based Hybrid Approach for Analysing and Detecting Malware Threat in Android Applications

Hetal Suresh[1], Joseph Raymond V [2]

[1]M.Tech in Information Security and Cyber Forensics, [1,2]SRM Institute of Science and Technology, Kattankulathur, Chennai, India

**Corresponding author E-mail: hetalsuresh01@gmail.com*

Abstract

Mobile phones has become very integral part in our day to day life. In the digitalized world most of our day to day activities rely on mobile phone like banking activities, wallet payments, credentials, social accounts etc. Our system works in such a way that if there is an advantage to a technology there also exists a disadvantage. Every users have all their private and sensitive data in their mobile phones and download random applications from different platforms like play store, App store etc. There is a huge possibility that the applications downloaded are malicious applications. The existing system provides a solution for detection of such applications with the help of antivirus which has pre-built signatures that can be used to obtain an already existing malware which can be modified and manipulated by the hacker if they tend to do so. In this project, our purpose is to identify the malicious applications using Machine learning. By combining both static analysis and dynamic analysis we can use a Hybrid approach for analysing and detecting malware threats in android applications using Recurrent Neural Network (RNN). The main aim of this project will be to ensure that the application installed is benign, if it is not, it should block such applications and notify the user.

Keywords: *Android, Malicious Application, Machine Learning, Hybrid approach, RNN*

1. Introduction

The Android operating system, is a trending OS now that is used in most of the mobile phones. There are many Advantages of android OS, these operating systems powers more than billions of smartphones, tablets and TV. Android phones are also available in good budget rates, so common man can also afford it. Since there are various end users they are mostly targeted for security breach as any other technology.The existing system approach concentrates on signature based approach which is antivirus, the antivirus based approach has pre-built signatures which can be used to detect malware threats whose signatures are given to those pre-built signatures. This is a vulnerable approach since the attacker tries to penetrate a malicious application bypassing the signatures that are pre-built. The attacker bypasses the signature by creating a new/encrypted signature thereby reaching the target device for performing malicious activity. In our proposed system, we try to overcome the major disadvantage of using antivirus based approach. We propose a machine learning approach that can find the malware threats by using a learning environment.

Why machine learning is the future? In this generation of technology data is everything and we can find data everywhere, everything is stored in form of data in different formats. From text messages to messenger, whatsapp, social networking apps, maps, shopping apps everything uses data as a common medium for storage. So, it is necessary to manage these data. Humans cannot manage this whole lot of data, they have limit for managing data. One of the major solution found in recent years is an approach called machine learning. A machine learning is to make a machine learn about particular pattern without being explicitly programmed. This approach is basically like you teach the machine twice on how the patterns are and analysis report, the third time the machine itself brings up with a solution and the fourth time it does it in a better way. For making our system more secure and less vulnerable we use this machine learning approach in our android applications. Our system with machine learning approach will be more efficient than the previous

existing approach in majority of the ways. That was what our aim and that is exactly what which leads to the development of an android application, using machine learning approach, capable of distinguishing a malicious application from a benign application. We use Recurrent Neural Network (RNN) model to detect and analyse the malware threats in android application. This is one of the deep neural network approach of machine learning. This algorithm has provided various significant result when provided with appropriate dataset. Let's continue with how we are going to build our system.

2. Analysis of Application

2.1. Application Compatibility

Application Compatibility is the term used for checking whether the android application is compatible i.e. whether it is capable of working on various devices and platforms. Since there exists a features based variations, based upon the configuration of the devices. There are various devices like mobile phone, tablets, smart

TV etc. Which has to be given its own input of imported dataset according to our project which will be given to the machine learning RNN algorithm. There are certain device configurations that are different for different versions of android OS. The scope of our project is to identify the malware threats in mobile phones and its applications.

2.2. Static Code Analysis

In malware analysis technique, to analyse without running it, we use static analysis. To find whether the input is malicious or not the static analysis uses different techniques.
In our system static analysis occurs before installing the android applications, which analyses the android Manifest.xml file which has pre-built permissions. For android application the extension used is .apk which is android package kit file that is package file format, like how windows uses .exe executable file format. The apk file consists of one of the most important file which is Manifest.xml file contains all the details needed by the android system about the android application like permissions and works as an intermediate between
developer and the platform. For static analysis we use this manifest.xml file. All the permissions in the android Manifest.xml file will be given as an input for static analysis that checks for malicious behaviour. The analysis produced will be given as an input to the machine learning environment to our recurrent neural network algorithm for producing significant output.

2.3. Dynamic Behavior Analysis

Dynamic analysis is an analysis based on the behaviour of the file while running the application, after the application is installed and the evaluation is done by executing the data in run-time. For the dynamic analysis to be effective, while installing the application the list of activities it undergoes after the installation is tracked during the runtime. These activities basically are like monitoring the network traffic, CPU utilization, amount of battery it will be using during its runtime and other background activities.
Our system with dynamic analysis can acquire continuous activities from the API (application program interface) call event log which may consists of registry change, file download, other unauthorized activity to be malicious that cannot be showcased immediately after the installation. The dynamic behaviour analysis can take the high user privilege as admin level, and provide more accurate results as root access ensures complete access. Thereby providing accurate measures for analysing the behaviour.

2.4. Hybrid Analysis

Static code analysis can only perform the analysis without running the application. Which can be a disadvantage for the malware triggering after the application apk is installed. Similarly when considering the dynamic behaviour analysis can perform analysis only after the apk application is installed and executed. Which can be a disadvantage for the malware that are pushed before installing the apk application file. To ensure our system have a reliable approach we take collaborative steps by combining both static code analysis and dynamic behaviour analysis.
Combining the results of both static and dynamic analysis known as Hybrid analysis. It is a common fact that static analysis alone or dynamic analysis alone cannot make the desired results. Our system uses both static analysis and dynamic analysis for better analysing purposes, since combined result can have accurate results. In static code analysis the uncompressed apk is disassembled for searching patterns which are in byte and hash signatures. It can be used to obtain a threshold value from the results it obtained. The dynamic behaviour analysis obtained after installing the apk and executing the apk file, which generate input on patterns based on the registry change, file download etc. This threshold value along with the output of dynamic analysis will be considered for the final output. The overall system idea is to ensure that our system is efficient by using both static and dynamic analysis which we call it as hybrid approach. Using this hybrid analysis, we can differentiate the respective performance for both the static analysis and dynamic analysis which enables a model that gives better analysis when combined together. The significant output of both the analysis can be given as a dataset into the machine learning environment using RNN (recurrent neural network) algorithm which will give respective output for our proposed system.

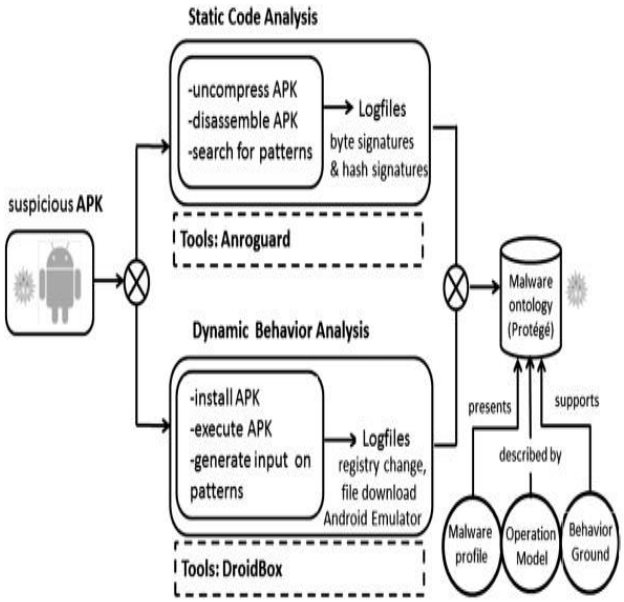

Fig 2.4.1: Hybrid analysis for a suspicious apk file.

Combining both the analysis we call it as hybrid analysis which supports both the code based and behaviour based dataset for training the machine learning.

2.5. Root Privilegde

In static code analysis, while using Manifest.xml the high user privilege comes into the picture we call it as root privilege. The purpose of providing root privilege is to ensure that we get the access to all the commands, registry activities, system files, folder locations and access privilege that are provided only to the root user that are not in reach of a common user to perform.
Root privilege is given as removing user privilege and giving administrator privilege that gives complete access to the user in terms of the mobile device. In our case, root privilege lets the Manifest .xml file in apk to have the complete access without which it is not possible to get easy access to the files inside apk file. Therefore, Root privilege is required for our system while performing static analysis. Using which we give dataset to the machine learning so for appropriate results of analysis we give root privilege.

3. Problem Statement

The recent solutions for malware threats in android application only have antivirus, a signature based solution as of now. These antivirus based solution for detecting malware threats are vulnerable to zero-day exploits as the malware coders are capable of replicating a new or encrypted signature for inserting malicious content into the

application on their own which bypasses the antivirus. Since the malware coders have a major advantage of zero day vulnerability they have higher scope of proving vulnerable and weak in protecting our mobile phones. As they try to manipulate the malicious code as such that the signature based approach will not be able to detect them. There also exists a basic check for malicious apps in play store. There do exists a lot of malware infected application in the play store which do affect thousands of its users who download and use it.
Our proposed system is created to face such issues that is created by the malware coders who can bypass the signature based approach used in antivirus. Which has some pre-built signatures in the manifest.xml file for static analysis and collects the activity changes encountered after the application is installed from the play store for dynamic analysis. Both the analysis combined as

hybrid analysis is given as a dataset for the machine learning to learn the malicious code and identify the pattern which ensures and differentiates the malicious application from the benign application.

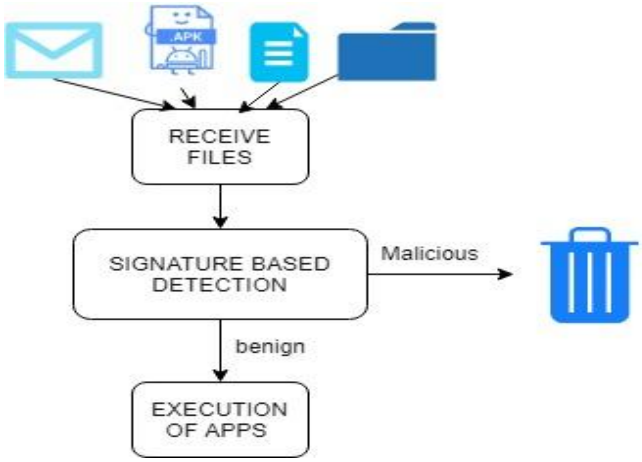

Fig: 3.1: Signature based detection using antivirus

Flaw in the existing system is that signatures can be bypassed by the malware coders.

4. Proposed Methodology

Our proposed methodology deals with the implementation of machine learning based detection and analysis of malware threats to find whether an application is a malicious or benign application. The existing system deals with a signature based approach that stores pre-defined malicious encounters as a signature in an anti-virus application. Such an anti-virus application can be manipulated by malware coders and change the signatures in run-time as well.
Our approach of combining both the static and dynamic analysis as a hybrid approach makes our system more effective by filtering the harmful applications with accuracy and efficiency. The issue in the existing methodology is addressed in our system which has permissions inside the Manifest.xml file in apk and not signatures that can be bypassed by a good malware coder. These permissions along with the denial of activities that are found to be malicious at the run time.
In this proposed methodology, we have created a predefined rule set which is created for high success rate on our own to ensure high efficiency. We propose a hybrid approach that also combines both the static and dynamic analysis results to acquire an efficient model. The data set we use for machine learning purpose is prepared based on these static and dynamic analysis based result. These results are given as a training set for our machine learning that uses RNN algorithm for training our machine with detection of malicious applications. In this paper, the RNN model we use makes use of the sequential information. Generally the input given and the output obtained are independent of each other. To predict the malicious code or activity of an application we analyse a code with the set of various malicious code and train our machine. RNN is also known as Recurrent because of their sequential nature they have the ability to perform the same task again and again, with the output obtained is based upon the previous computations. The alternative way of describing RNN would be the memory it has. The memory can compute the information captured with the previous obtained calculations.
We use this model of machine learning for analyzing and detecting malware threats in our project.This dataset acquired by our hybrid analysis are given as input to the machine learning based RNN model. By using our RNN model we can determine a desirable result which can be done more number of times, which ensures that the result acquired is more efficient and by training the machine again and again it also gives more accurate results. This accurate result of determining the malicious code is the main advantage of our proposed methodology. This model ensures that the malicious codes format is obtained for the varying inputs of dataset and obtains a reliable approach for trusting an
application compared to the previous approaches used.
For better understanding we use sequential diagram that demonstrates the sequence of execution it performs to secure the user by identifying whether the application is benign or not. We use a feature extractor that extracts the features like Network details, CPU utilization and battery and transfers the collected data from both the analysis to the dataset module. This dataset module further vectorizes the collected data from the feature extractor and transfers the vectored data to machine learning classifier. We use RNN based machine learning classifier that analyses the dataset by training and analysing the vectored data. Then the Result is obtained respectively and if analysed result detects malware, it blocks the application else it allows the application to be used by the user.

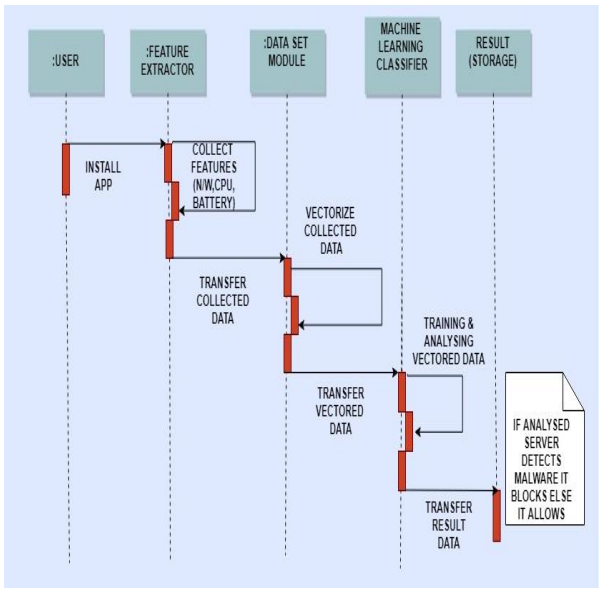

Fig: 4.1: Sequential diagram of our proposed system

Detailed sequence of our malware detection system is explained.
In our working system when we install the app, first it performs static analysis with the pre-built permissions in the form of dataset and gives the output to the dynamic analysis which itself has an input of the activities that the app goes through after installation and gives through result accordingly. If the result is that the app is malicious
it blocks the application, else it allows the application.

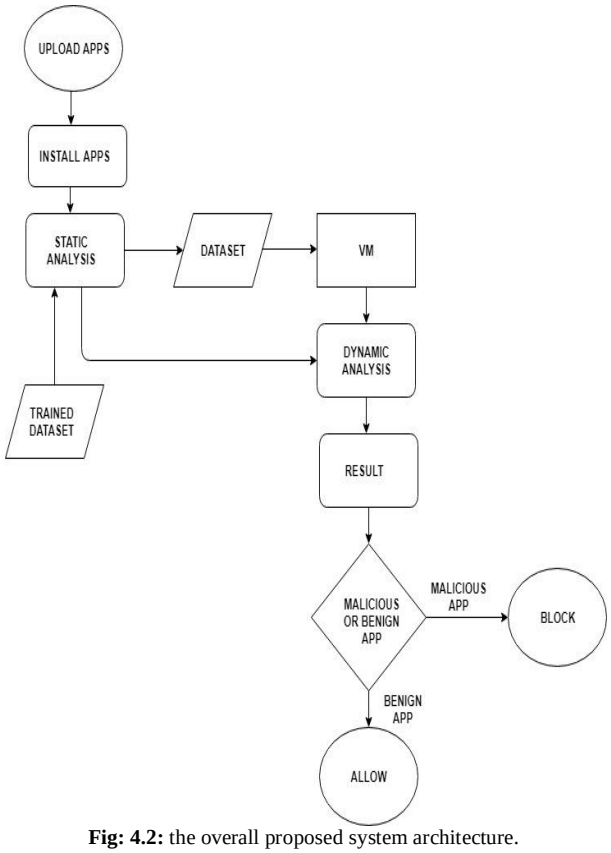

Fig: 4.2: the overall proposed system architecture.

Analyse the downloaded app using static analysis using android manifest.xml file and perform dynamic analysis using datasets of application behaviour.

4.1 Mathematical Calculations

The mathematical modelling for our system relies on the machine learning approach we use. Since we use Recurrent Neural Network approach our model proposes the following equations:

$$s_t = \tanh(Ux_t + Ws_{t-1})$$
$$o_t = \text{softmax}(Vs_t)$$

I) x_t is the input given at the time step t. For example, x_1 can be taken as a one-hot vector which corresponds to the second word.
II) s_t is taken as the hidden state at time step t. S_t is the memory that is given to the network. s_t is computed with the help of the previous hidden state and the input given at the current step:

$$s_t = f(Ux_t + Ws_{t-1}).$$

The function f is generally a nonlinearity as tanh or ReLU. s_{-1}, that is needed to compute the initial 1st hidden state, is generally initialized to all zeroes.
III) o_t is taken out as a desired output at step t. For example, if the next word in a sentence is required to be predicted it would be vector of our activities change in our environment.

$$o_t = \text{softmax}(Vs_t).$$

4.2. Algorithm

```
rnn = RECURRENT()
b = rnn.stepper(a) # a = input vector, b= RNN's output vector

class RECURRENT:

def stepper(self, a):
   # hidden state is updated
self.c=np.tanc(np.dot(self.W_cc,self.c)+np.dot(self.W_ac, a))
# output vector computation
b = np.dot(self.W_cb, self.c)
return b

b1 = rnn1.stepper(a)
b = rnn2.stepper(b1)
```

5. Expected Output Result

The expected output result after executing the proposed methodology by analyzing the activities
and factors using static and dynamic analysis in a hybrid approach and acquiring the dataset to be given into the machine learning approach for obtaining desired result after the complete project is done and implemented into a developed application with the capability of finding whether the application in the play store is malicious or benign by using RNN based Machine Learning Technique as the primary method. The expected minimum success rate for the system is above 96%.

6. Conclusion

The existing system approach concentrates on signature based approach which is antivirus, the antivirus based approach has pre-built signatures which can be used to detect malware threats whose signatures are given to those pre-built signatures. This is a vulnerable approach since the attacker tries to penetrate a malicious application bypassing the signatures that are pre-built. The attacker bypasses the signature by creating a new/encrypted signature thereby reaching the target device for performing malicious activity. The techniques used in this paper is RNN model, a machine learning method to handle the malicious threats and thereby find out whether the application is benign or not. We use machine learning to ensure that in a learning environment where the datasets will be different each time an application responds in particular way. Machine Learning ensures that the inputs of various datasets are analysed and corresponding outputs are trained to provide an in-depth analysis of a malware threats in android applications. Using which we can come to a conclusion whether the application can be trusted or not.

References

[1] Nayeem Islam ; Saumitra Das ; Yin Chen(2017) On- Device Mobile Phone Security Exploits Machine Learnin IEEE Pervasive Computing
[2] (Volume: 16, Issue: 2, April-June 2017)
[3] Rhode, M., Burnap, P., Jones, K., Aug. 2017. Early Stage Malware Prediction Using Recurrent Neural Networks. arXiv:1708.03513 [cs]ArXiv:1708.03513.
[4] URL http://arxiv.org/abs/1708.03513
[5] D. Arp et al., "DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket," Proc. Network and Distributed System Security Symp. (NDSS), 2014; www.sec.cs.tubs.de/pubs/2014-ndss.pdf.
[6] Shuang Liang and Xiaojiang Du "Permission-Combination-based Scheme for Android Mobile Malware Detection" Dept. of Computer and Information Science Temple University, Philadelphia, PA 19121,USA {shuang.liang2012, dux}@temple.edu, IEEE
[7] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca,Member, IEEE, Daniel Arp, Konrad Rieck, Igino Corona,Giorgio Giacinto, Fabio Roli "Yes, Machine Learning Can Be More Secure! ACase Study on Android Malware Detection".

IEEE Transactions on Dependable and Secure Computing (Volume: PP, Issue: 99)
[8] Xiang Li , Jianyi Liu , Yanyu Huo , Ru Zhang , Yuangang Yao "An android malware detection method based On androidmanifest file", Proceedings of CCIS2016, 2016 IEEE
[9] T.Wang et al., "Jekyll on iOS: When Benign Apps Become Evil," Proc. 22nd Usenix Security Symp. (SEC),
[10] 2013;www.usenix.org/conference/usenixsecurity13/technicalsessions/presentation/wang_tielei.
[11] J. Oberheide and C. Miller, "Dissecting the Android Bouncer,"SummerCon, 2012.
[12] N.J. Percoco and S. Schulte, Adventures in Bouncerland: Failures of Automated Malware
[13] Detection within Mobile Application Markets, Black Hat, 2012.
[14] N. Idika and A.P. Mathur, A Survey of Malware Detection Techniques, tech. report, Purdue Univ., 2007.
[15] A.P. Felt et al., "A Survey of Mobile Malware in the Wild," Proc. First ACM Workshop Security and Privacy in Smartphones and Mobile Devices (SPSM), 2011, pp. 3–14.
[16] J. Bickford et al., "Security versus Energy Tradeoffs in Host-Based Mobile Malware Detection," Proc. 9th Int'l Conf. Mobile Systems, Applications, and Services (MobiSys), 2011, pp. 225–238.
[17] S. Poeplau et al., "Execute This! Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications," Proc. 20th Annual Network & Distributed System Security Symp. (NDSS), 2014;https://cs.ucsb.edu/~vigna/publications/2014_NDSS_ExecuteThis.pdf.
[18] Cyrille Artho,Armin Biere,"Combined Static and DynamicAnalysis",2005https://doi.org/10.1016/j.entcs.2005.01.018
[19] Willems, C., Freiling, F.C.: Reverse code engineering—state of the art and countermeasures. it-Information Technology, pp. 53–63 (2011)