INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

7th SEMESTER MINI PROJECT

REINFORCEMENT LEARNING FOR ANDROID MALWARE DETECTION

Submitted By:

Surabhi Gogte (IIT2016053)

Simran Gill (IIT2016088)

Chahak Sharma (IIT2016103)

Submitted To:

Prof. Om Prakash Vyas

CERTIFICATE FROM SUPERVISOR

I propose the mini project report prepared under my supervision titled "Reinforcement Learning For Android Malware Detection" be accepted for the completion of mid-semester of seventh semester of Bachelor of Technology in Information Technology. Due acknowledgements have been made to all the resources and frameworks used.

Supervisor :

Date:

.....

Place : Allahabad

Prof. Om Prakash Vyas

Abstract

The number of interconnected systems has increased considerably, so these systems are being exposed to cyber attacks more than ever. The growing amount and diversity of Android malware has significantly weakened the effectiveness of the conventional defense mechanisms, and thus Android platform often remains unprotected from new and unknown malware. Reinforcement Learning methods are used to address these issues.

In this paper, we attempt to apply Reinforcement Learning on cyber security use case: Android malware detection . This paper presents a survey of Reinforcement Learning approach developed for cyber security and compare it with the previous works done using Deep Learning. Our conclusions are based on an extensive review of the literature as well as on experiments performed on real enterprise systems and network traffic.

Contents

1 Introduction

Cyber security is the protection of internet-connected systems, including hardware, software and data, from cyber attacks. The purpose of cyber security is to help prevent cyber attacks, data breaches and identity theft and can aid in risk management. Nowadays, mobile applications (apps) have become the predominant means of accessing personalized computing services such as email, banking, etc. However, this rapid deployment and extensive availability of mobile apps has made them attractive targets for various malware.

This report is focused on implementation of Reinforcement Learning for Android Malware Detection. The implementation is to be done using large dataset for Android malware detection. Many different published research papers are analyzed for this task and a model is selected looking at the accuracy and error analysis between different models. Our approach is to propose a RL model for addressing these issues. Over the next sections we formalize our problem definition and then introduce our proposed model.

2 Motivation

A large number of research works have thus been studied for analyzing and detecting Android malware . These approaches discover patterns to distinguish Android malware which are effective solutions for known Android malware detection. However, this kind of approaches proved out to be less effective when detecting unknown Android malware. Deep learning methods have also been studied for detecting malware detection. These methods extract features from known Android apps and malware, then use machine learning algorithms to learn these features in order to detect unknown Android malware.

The main motivation behind this research is to apply reinforcement learning for cyber security issues and compare it with the previous works done using Deep Learning.

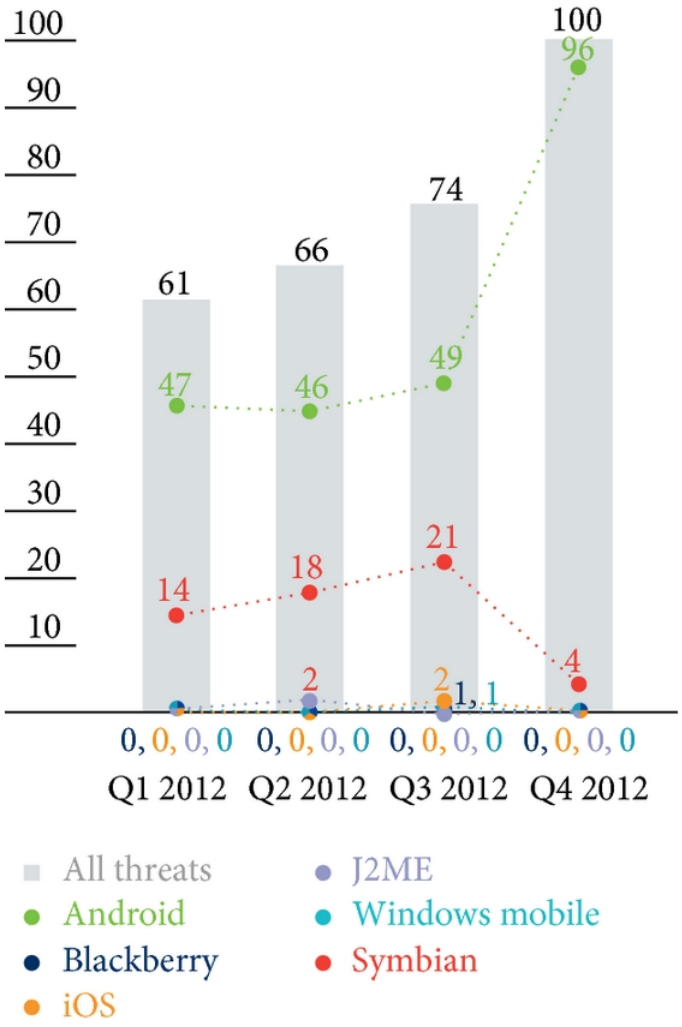

Figure 1: [1]is a report by Finnish security company F-secure which states that, 301 mobile malware samples from 2012, 238 samples targeted the Android platform. While the amount of malware that targeted other mobile platforms gradually decreased as time went on from the 1st to 4th quarter, Android showed a contrasting result. The reason for the increase in Android malware was its open source policy and its leniency to market application verification. In addition, it easily allowed the distribution of malware in the market through the repackaging method of inserting it in a normal application.

3 Problem Definition

To train a Reinforcement Model that could detect an Android Malware.

Input : APK file

Output : Benign / Malicious.

4 Literature Review

Most of the Android-targeted malware is divided into Trojan, Spyware, Root Permission Acquisition (exploit) and Installer (dropper).

1. **Trojan:** it is a program containing a risk factor in effect . It executes malware when running the application.
2. **Spyware:** it is secretly installed on device to collect information. It repeatedly opens pop-ups and causes inconvenience by changing a device's settings or being difficult to delete.
3. **Root permission acquisition (exploit):** it acquires root permissions to clear security settings and make additional attacks.
4. **Installer (dropper):** it conceals malware in a program and guides users to run malware and spyware.

Following models were analysed under the literature survey of the proposed problem statement to choose the best preferable model to be implemented.

4.1 Feed-Forward Network Model[2]

In this model, Deep Neural Network was applied for Android Malware detection.

1. The model identifies Android Malware based on API information used.
2. The units in input to hidden layer and hidden to output layer are fully connected.
3. The feed forward network is trained using back-propogation mechanism.
4. To avoid overfitting, batch normalization and dropout were used.ReLU was used a non-linear activation function.
5. For classification purpose, final fully connected layer used sigmoid function and the model gave accuracy of 94 % with 0.834 precision.

The test results of DNN are as follows

Accuracy	**Precision**	**Recall**	**F-score**
0.94	0.834	0.868	0.851

4.2 Random Forest[3]

In this model, a Random Forest Machine Learning Algorithm was used for malware detection.

1. This algorithm develops lots of decision trees.
2. These trees are based on random selection of data and random selection of variables.
3. The remaining data set, apart from the training set is used for predicting the tree in forest which makes best classification of data points and the tree having most predictive power is shown as output.
4. The decision of majority of trees is chosen by the algorithm as the final decision.
5. The model achieved an F1 Score of 98.24% .

4.3 Linear Support Vector Machine[4]

In this model, a classifier is generated to detect Android APK files for malware.

1. Traditionally, signature-based, behavior based and dynamic analysis techniques have been used for malware detection.

2. Behavior-based detection involves the inconvenience of having to determine malware infection status by examining numerous features.

3. Using Machine Learning, classification is automated thereby providing more accuracy and precision.Of the input features, unnecessary ones are removed by the SVM machine learning classifier itself and the modeling is carried out.

4. For SVM True Positive Results came to be 0.999 with 99.7% accuracy and precision of 0.992.

5. SVM has FPR = 0.004, which could be determined as the best classifier because its ratio of incorrectly classifying normal applications as malicious is small, and it shows far better performance than other classifiers also in terms of accuracy and precision.

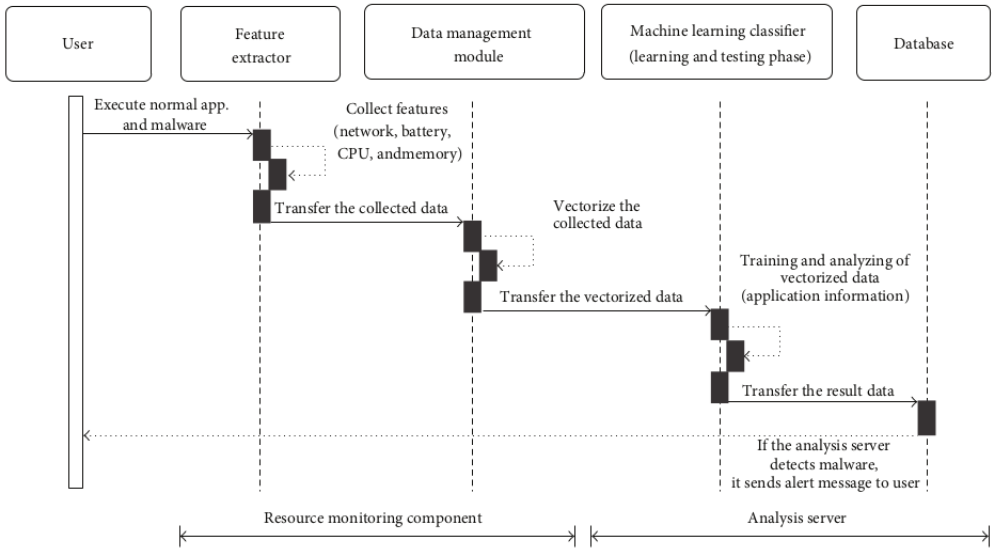

Figure 2: Sequence diagram for malware detection system.

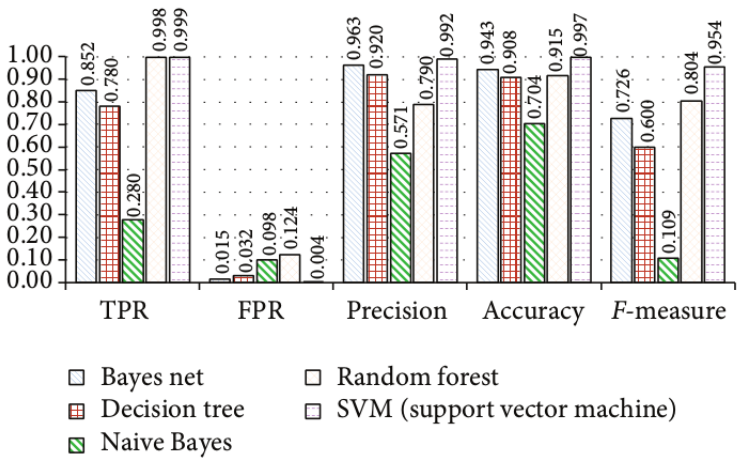

Figure 3: The figure shows comparison between outputs of different machine learning classifiers namely,Bayesian networks, Decision Tree ,Random Forest, Naive Bayes and SVM.It can be seen that SVM gives better results than others.

5 Proposed Methodology

In this paper we are proposing the Reinforcement Model for Cyber Security - Android Malware Detection.

5.1 Dataset - The Drebin Dataset[5]

We would be using The Drebin Dataset to train our Reinforcement Learning Model. Dataset consisting of feature vectors of 215 attributes extracted from 15,036 applications (5,560 malware apps from Drebin project and 9,476 benign apps). Contains 5560 malware files collected from August 2010 to October 2012. All malware samples are labeled as 1 of 179 malware families.Drebin is one of the most popular benchmark datasets for Android malware detection.

5.2 Reinforcement Learning Preliminary[6][7]

Different from the other popular branch of ML, i.e., supervised methods learning by examples, RL characterizes an agent by creating its own learning experiences through interacting directly with the environment. RL is described by concepts of state,action, and reward. It is a trial and error approach in which the agent takes action at each time step that causes two changes: current state of the environment is changed to a new state, and the agent receives a reward or penalty from the environment. Given a state, the reward is a function that can tell the agent how good or bad an action is. Based on received rewards, the agent learns to take more good actions and gradually filter out bad actions.

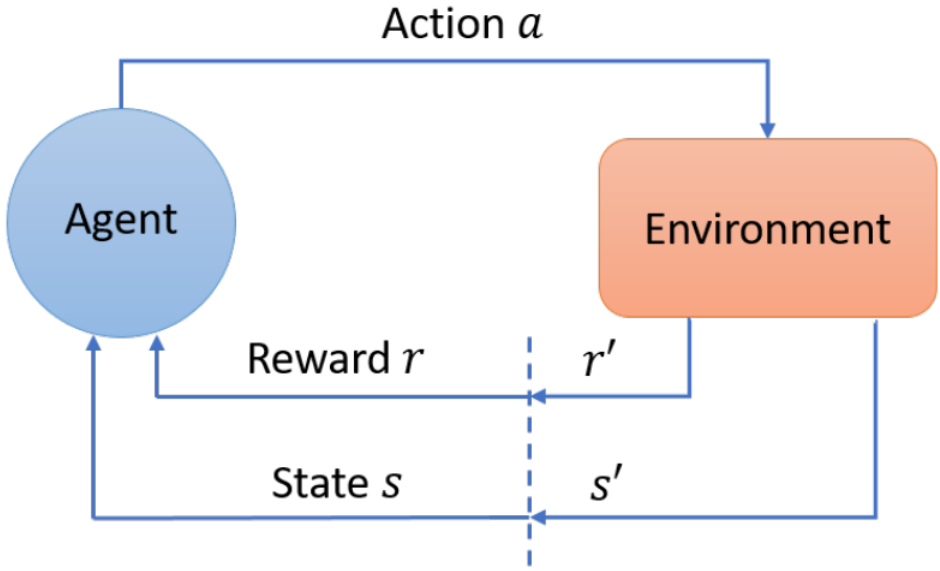

Figure 4: Interactions between the agent and environment in RL, characterized by state, action and reward. Based on the current state s and reward r,the agent will take an optimal action, leading to changes of states and rewards. The agent then receives the next state s' and reward r' from the environment to determine a next action, making an iterative process of agent-environment interactions.

5.3 Markov Decision Processes MDP's [8]

Formally RL can be described as Markov Decision Processes (MDP's) which consists of:

1. a set of **states** S.
2. a set of **actions** A.
3. **transition dynamics T(s_{t+1} | s_t, a_t):** that map a state-action pair at time t onto a distribution of states at time t+1.
4. **reward function R(s_t, a_t, s_{t+1})** .
5. **discount factor** γ between 0 and 1: this quantifies the difference in importance between immediate rewards and future rewards.
6. **memorylessness:** Once the current state is known, the history of the prev states can be erased because the current Markov state contains all useful information from the history.

$$\sum_{t=0}^{\infty} \gamma^t r(x(t), a(t))$$

Summing across all time steps t. For $\gamma = 1$, r(x,a) is a reward function. For state x and action a it gives the reward associated with taking that action a at state x. We're trying to maximize the sum of future rewards by taking the best action in each state.

Using Markov Decision Processes we set up our reinforcement learning problem and formalize the goal.

6 Implementation Plan and Time Line

We have implemented the Reinforcement Learning model. The model will be trained and tested with the dataset and the performance will be analysed.

Given below is the proposed Time Line of this project.

Aug-Sep 2019	Literature Survey
Sep-Oct 2019	Theoretical Basis of Chosen Model
Oct-Nov 2019	Code Implementation
Nov 2019	Training and testing of the model
Final submission	Analysis of model performance

7 References

1. Mobile Threat Report Q4 2012
2. Vinayakumar R, Barathi Ganesh HB1, Prabaharan Poornachandran, Anand Kumar M and Soman KP. Deep-Net: Deep Neural Network for Cyber Security Use Cases
3. Gowtham Sethupathi, Swapnil Siddharth, Vikash Kumar, Pratyush Kumar, Ashwani Yadav. Maldroid: Dynamic Malware Detection using Random Forest Algorithm.
4. Hyo-Sik Ham, Hwan-Hee Kim, Myung-Sup Kim and Mi-Jung Choi. Linear SVM-Based Android Malware Detection for Reliable IoT Services
5. The Drebin Dataset
6. Thanh Thi Nguyen and Vijay Janapa Reddi. Deep Reinforcement Learning for Cyber Security
7. A. Gosavi Reinforcement Learning: A Tutorial Survey and Recent Advances
8. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep Reinforcement Learning : A brief survey