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

We have implemented Reinforcement learning for Android Malware Detection.It is one of the main issues in Cyber Security.As Android is open source, it is more prone to attacks by hackers.It is easier to attack a system if a developer changes internal settings or root permissions.

In this paper, we have used Q-Learning for classification of android malware on Drebin dataset and compared output with .We performed feature extraction using Random Forest Classifier and Extra Trees Classifier ans selected top 15 features according to their importance.The results of classification were compared with other machine learning algorithms.

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.[1]

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.

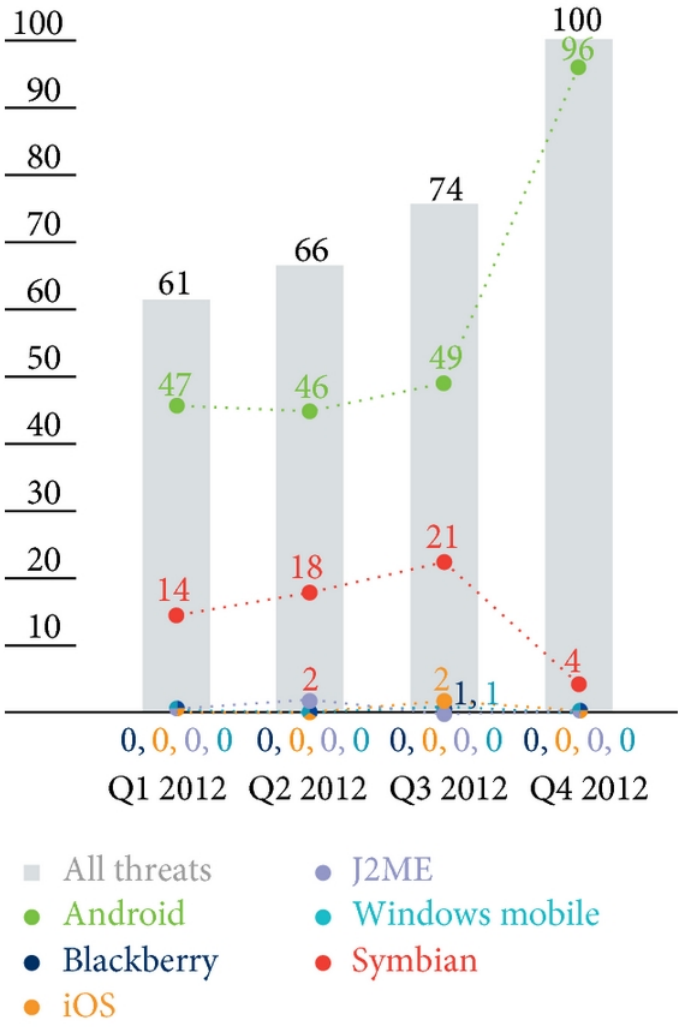

Figure 1: [1]is a report by Finnish security company F-secure which states that, 301 mobile malware samples from 2012, 238 samples targeted the Android platform. While the amount of malware that targeted other mobile platforms gradually decreased as time went on from the 1st to 4th quarter, Android showed 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 : Drebin Dataset

Output : Accuracy

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 Mobile Malware Detection Techniques[2]

As Android is open-source,it is more vulnerable to malware attacks.Most of the mobile phones are Android based.Hence it is important to curb these attacks.Mainly,5 types of malware detection techniques are seen-

1. Signature Based :-Malware detection is done during program compilation.Semantics are ignored ,thus,giving malware code a chance to go undetected.

2. Specification-based :- Depending on the behaviour of normal program,a rule set is defined.The programs not following the rule set are deemed as malware.

3. Behavioural-based :- Different malware families are studied on a target system.A classifier is trained to detect malware behaviour from normal app behaviour.

4. Data-Mining based :- Defined patterns are detected from large amount of data and classifiers are used for malware detection.

5. Cloud Based :- Google uses a service called Bouncer to check for malware in the apps uploaded on the play store.If a third party app is installed and android detects malware in it,Android has the ability to delete such harmful apps.

4.2 Static and Dynamic Analysis of Android Malware[3]

1. To find static features,apk files are reverse engineered. AndroidManifest.xml contains several features useful for static analysis.

2. Android has total 135 permissions.A binary vector is be created from each apk file and stored in dataset.

3. Dynamic analysis uses system calls .Android emulators are used to find these system calls.

4.3 DREBIN : Effective and Explainable Detection of Android Malware in Your Pocket[4]

1. Features are extracted using broad static analysis.
2. The features are embedded into a joint vector space.
3. SVM is used for learning based detection.
4. respective patterns are detected and mapped to a meaningful description.
5. Out of 123453 applications, 5560 malware samples were detected.

4.4 Linear Support Vector Machine[5]

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.

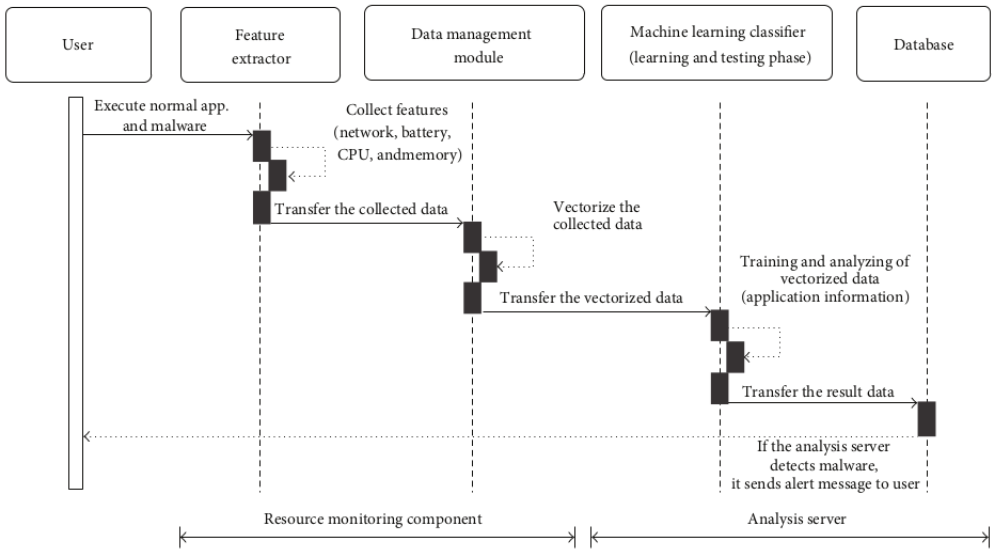

Figure 2: Sequence diagram for malware detection system.

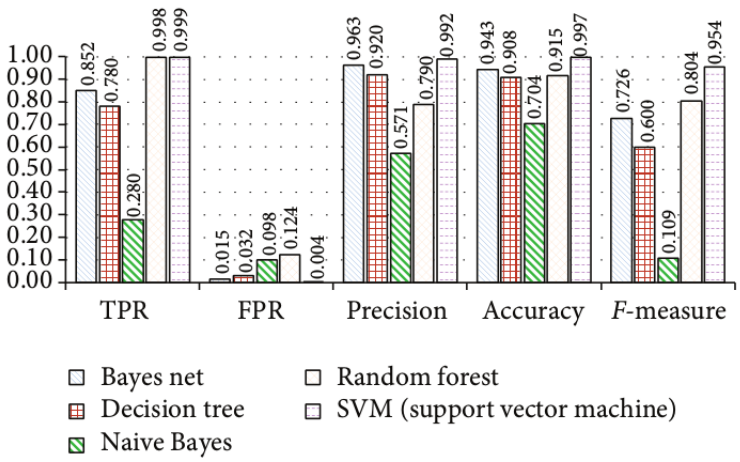

Figure 3: The figure shows comparison between outputs of different machine learning classifiers namely,Bayesian networks, Decision Tree ,Random Forest, Naive Bayes and SVM.It can be seen that SVM gives better results than others.

4.5 Deep-Net: Deep Neural Network for Cyber Security Use Cases[6]

In this paper,Android Malware detection using Deep Neural networks is studied.

1. A feed forward network is trained using back-propogation mechanism.
2. A dataset containing 37,107 samples collected from Opera Mobile Store over a period of 9 months was used.
3. To avoid overfitting, batch normalization and dropout were used.A non-linear activation function ,ReLU is used here.
4. 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

4.6 On the Effectiveness of Machine and Deep Learning for Cyber Security[7]

Three main issues in Cyber security that are tackled using ML and Deep Learning are Android Malware Detection ,Intrusion Detection and Spam and phishing detection.

1. This paper explains different types of Machine Learning techniques namely Deep and Shallow Learning.
2. Both types are further divided into Supervised and Unsupervised Learning.
3. It discusses which ML algorithms work best for different areas of Cyber Security.

4.7 Maldroid: Dynamic Malware Detection using Random Forest Algorithm[8]

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% .

4.8 Android Malware Detection Using Parallel Machine Learning Classifiers[9]

1. Using different machine learning classifiers, classification accuracy is obtained.
2. In Parallel classifiers,combination of classifiers is used as average of probabilities, product,maximum of probabilities.
3. Some improvement is seen using the product and maximum pf probabilities.

4.9 Machine Learning for Android Malware Detection Using Permission and API Calls[10]

1. Repackaging,Updating and Downloading are the three types of penetrating techniques used by malware applications.
2. There are four threat levels of permission- Normal,Dangerous, Signature ,System.
3. If app requests a dangerous or higher level permission instead of requiring it while installation, it may be malware.
4. Benign applications use more API calls than malware ones.

4.10 Toward optimal feature selection using ranking methods and classification algorithms[11]

Feature Selection Algorithms are divided into filter,wrapper and embedded type.

1. Filter methods evaluate the quality of selected features,wrapper methods require the classifier to evaluate the quality,Embedded methods use learning of optimal parameters.
2. Feature selection consists for 4 steps-subset generation,subset evaluation,stopping criteria and result validation.

3. Different techniques like Information Gain,Gain Ratio etc are used for feature ranking.

4. Algorithms like IB1,Naive Bayes,C4.5 Decision Tree,RBF Network are used for elimination of redundant features.

4.11 Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic[12]

1. The dataset is divided into training and testing set using n-cross validation.

2. Bayesian Probability of each feature is calculated.The ranks of features is calculated based on ranking value.

3. A backward elimination approach is used to eliminate features in which importance of a feature is determined by variance of classification accuracy with or without the feature.

4.12 Reinforcement Learning for Solving Classification Problems[13]

1. In this paper, RL is combined with multilayer perceptrons to find Value function of each state .

2. A input vector of length m,and target classes yi are there making a Dataset D=(x1,y1),(x2,y2)...(xn,yn)

3. A single reward function is there.There are same number of agents as classifiers.

4. While training, agent with same class as training input selects actions to maximize reward for that instance and other agents will minimize their rewards.

5. For testing,each state gets a value function which is trained on a neural network using back-propogation.

6. For a given input, class is predicted using maximum of value function of the state.

4.13 Relating Reinforcement Learning Performance to Classification Performance[14]

1. Using reinforcement learning,a policy for set of (observations,reward,actions) and a new observation mapping to set of actions is created.

2. The problem is defined into a conditional probability using past observations.

3. T classifiers are used for T steps.An action is selected by which expected sum of rewards is maximized.

4. Previous actions are taken into account while taking the next action.

5. The goal is to reduce reinforcement learner into binary classifier.

4.14 Q-Learning[15]

1. Q-Learning can be described as model-free reinforcement learning.
2. Bellman's equation is used to calculate Q-value of every state.
3. At every step ,agent observes the current state,selects and performs the best action ,decides the next state,calculates the reward for that state and adjusts the sum of rewards according to the Bellman equation.

5 Proposed Methodology

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

5.1 Dataset - The Drebin Dataset[16]

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 Feature Selection

A feature is a coloumn in our dataset. Feature selection is the method by which we can select the top features that are most useful. Training for more features not only takes more time but may also end up in decreasing the accuracy of our model because its not necessary that every feature has an impact on the output. Thus, adding these irrelavant features will result in decreased training speed and decreased accuracy.
There are several methods to perform feature selection. We applied Random Forest Classifier and Extremely Randomised Tree Classifier(Extra Trees Classifiers) to achieve this.In both the methods, we get a score of importance of the attributes for each feature . Larger the score, more the importance of the feature.[12]

5.2.1 Random Forest Classifier

A hundred thousand number of decision trees are constructed in Random Forest. Each of the decision tree is based on the random selection of data and variables. The remaining dataset is used for predicting the tree in the forest which makes the best classification. Each tree yields to a final answer , yes or no. Decision of the majority of trees is chosen as the final decision.

5.2.2 Extra Trees Classifier

This method is similar to Random Forest Classifying method. In this method, in each tree, random set of k features from the set of features is given at each of the node.The only difference between random forest classifier and extra tree classifier is that all the features we select for the split may or may not be the best features of the split since the point to split is randomly chosen. This leads

to more diversified trees and less evaluation of splitters at each node. Thus it is preferred over Random Forest Classifier because it takes less time.

5.3 Reinforcement Learning Preliminary[17][18]

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.

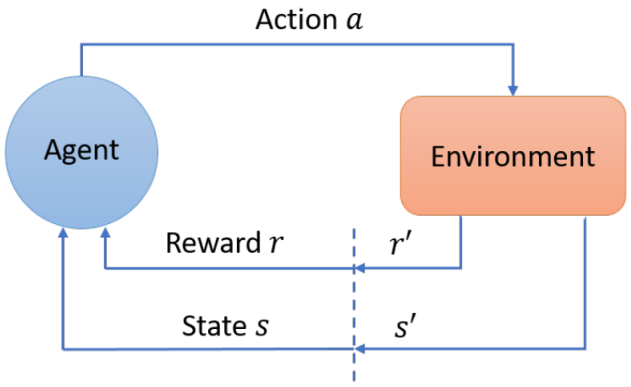

Figure 4: Interactions between the agent and environment in RL, characterized by state, action and reward. Based on the current state s and reward r,the agent will take an optimal action, leading to changes of states and rewards. The agent then receives the next state s' and reward r' from the environment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

5.4 Markov Decision Processes MDP's [19]

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

$$(\text{S, A, T, R, } \gamma)$$

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.

5.5 Classification using Q-Learning[15]

Q learning improves the behaviour of a learning agent iteratively by using Q-values Q(s,a). Where Q(s,a) is the estimation of how good it is to take an action a at a state s. Every time an action is performed, a reward or a penalty is awarded until it reaches a terminating state where it is said to complete one episode. Actions are chosen on the basis of an ϵ-greedy policy : either take an action with max q value or perform a random action .

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha(R(s,a) + \gamma max_{a'} Q(s',a') - Q_{t-1}(s,a))$$

Bellman's Equation for calculation Q value at a state 's' on taking an action 'a', where Q(s,a) is the old q value, α is the learning rate, R(s,a) is reward at state s and action a, γ is discount factor and max Q(s',a') is estimate of optimal future value. We maintain a Q table to store the q value of each state-action pair.

5.5.1 MDP Formulation

(S, A, T, R, γ)

1. S : each state is a tuple of possible combination of feature values.
2. A : actions defined are either benign or malicious.
3. T : next state is defined as the next tuple in the dataset.
4. R : if predicted true reward of +1 else a penalty of -1.
5. γ : discount factor is chosen as 0.95.

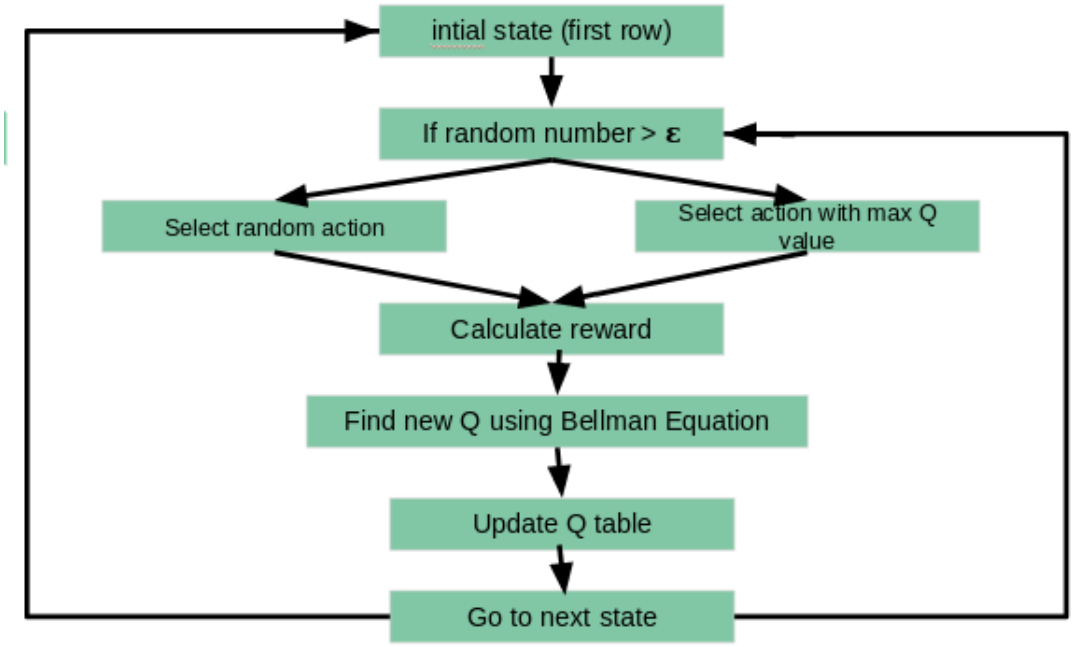

Figure 5: Flowchart of Q Learning algo implementation.

6 Results

The accuracy of the model varied with different values of learning rate. It was observed that accuracy decreases after a certain value of learning rate.The following graph depicts the variation of accuracy with learning rate.

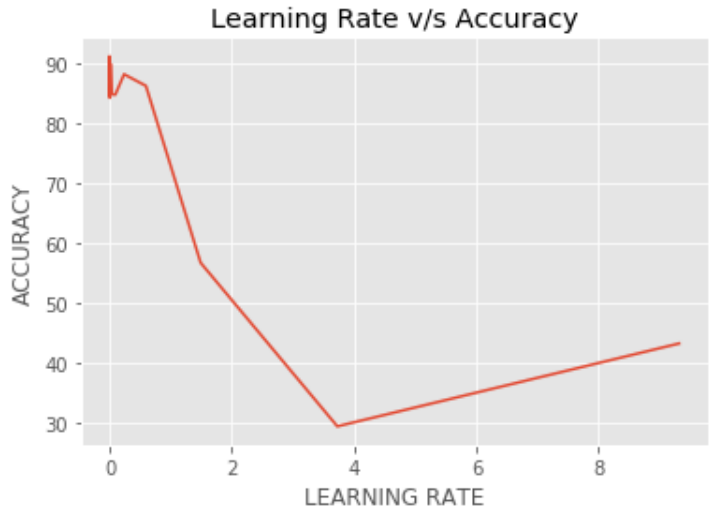

Figure 6: Plot of Learning rate v/s Accuracy.

We obtained the maximum accuracy of 91.287% at the learning rate of 0.0003 with an F1 score of 0.873.

We applied feature selection using both random forest and extra trees classifiers at different learning rates.We obtained the following accuracies and F1 score for our model. The results differed if we used different feature selection methods. Extra tree classifier performed better at desired learning rates of 0.00015 and 0.00039 as accuracy of our Q-Learning model is maximum at Learning Rate of 0.00015 .

1. For Random Forest Classifier

Learning Rate	Accuracy	F1 score
2.5e-05	87.186 %	0.791
0.00015	86.56 %	0.777
0.00039	87.652 %	0.799
0.095	86.144 %	0.769
1.490	57.171 %	0.349
3.725	34.426 %	0.027

2. For Extra trees Classifier

Learning Rate	Accuracy	F1 score
2.5e-05	88.516 %	0.822
0.00015	88.561 %	0.823
0.00039	91.287 %	0.873
0.095	84.859 %	0.752
1.490	56.705 %	0.361
3.725	29.306 %	0.033

In the past years, various researchers have build models to classify benign and malware files using Drebin dataset with different approaches like machine learning, deep neural networks, etc. Comparing these with our model we have the following results.Reinforcement learning with 91.287 % accuracy performs better than some models like XGBoost(74.1%) and Random Forest(81.4%). SVM(99.5%) and DNN(94%) performed better than our Q-Learning model.

Reinforcement Learning vs other algorithms

Algorithm Used	Accuracy	F1 score
SVM[5]	99.5 %	0.954
XGBoost[6]	74.1 %	0.134
5-layered DNN[6]	94.0 %	0.851
Random Forest[8]	81.4 %	0.79
Reinforcement Learning (Q-Learning)	91.287 %	0.873

7 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

8 References

1. Mobile Threat Report Q4 2012

2. Mohata, Vinit B., Dhananjay M. Dakhane, and Ravindra L. Pardhi. "Mobile malware detection techniques." Int J Comput Sci Eng Technol (IJCSET) 4.04 (2013): 2229-3345.

3. Kapratwar, Ankita, Fabio Di Troia, and Mark Stamp. "Static and dynamic analysis of android malware." ICISSP. 2017.

4. Arp, Daniel, et al. "Drebin: Effective and explainable detection of android malware in your pocket." Ndss. Vol. 14. 2014.

5. Hyo-Sik Ham, Hwan-Hee Kim, Myung-Sup Kim and Mi-Jung Choi. Linear SVM-Based Android Malware Detection for Reliable IoT Services

6. Vinayakumar R, Barathi Ganesh HB1, Prabaharan Poornachandran, Anand Kumar M and Soman KP. Deep-Net: Deep Neural Network for Cyber Security Use Cases

7. Apruzzese, Giovanni, et al. "On the effectiveness of machine and deep learning for cyber security." 2018 10th International Conference on Cyber Conflict (CyCon). IEEE, 2018.

8. Gowtham Sethupathi, Swapnil Siddharth, Vikash Kumar, Pratyush Kumar, Ashwani Yadav. Maldroid: Dynamic Malware Detection using Random Forest Algorithm.

9. Yerima, Suleiman Y., Sakir Sezer, and Igor Muttik. "Android malware detection using parallel machine learning classifiers." 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies. IEEE, 2014.

10. Peiravian, Naser, and Xingquan Zhu. "Machine learning for android malware detection using permission and api calls." 2013 IEEE 25th international conference on tools with artificial intelligence. IEEE, 2013..

11. Novaković, Jasmina. "Toward optimal feature selection using ranking methods and classification algorithms." Yugoslav Journal of Operations Research 21.1 (2016).

12. Nguyen, Cuong, Yong Wang, and Ha Nam Nguyen. "Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic." Journal of Biomedical Science and Engineering 6.05 (2013): 551.

13. Wiering, Marco A., et al. "Reinforcement learning algorithms for solving classification problems." 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 2011.

14. Langford, John Zadrozny, Bianca. (2005). Relating reinforcement learning performance to classification performance. 473-480.

15. Watkins, C.J.C.H. Dayan, P. Mach Learn (1992) 8: 279.

16. The Drebin Dataset

17. Thanh Thi Nguyen and Vijay Janapa Reddi. Deep Reinforcement Learning for Cyber Security

18. A. Gosavi Reinforcement Learning: A Tutorial Survey and Recent Advances

19. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep Reinforcement Learning : A brief survey