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

In today's world of technology,we are surrounded by electronics gadgets and connected through the internet constantly. The risk of privacy invasion, data breach and spam is at an all time high. Cyber security plays an important role in preventing and avoiding these threats. Here we have focused on Android Malware detection.It is a process by which malware is detected in android .apk files and applications using different techniques.
We have implemented Reinforcement learning,an area of Machine 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 accuracy of the classification was compared with other machine learning algorithms.

Contents

1 Introduction

Cyber security is the protection of all hardware devices,software related technologies and data associated with the applications and devices connected with the internet.Cyber security is very important in today's world,as there are hackers and scammers all over the internet ,who try to steal sensitive user data,scam people and cause disruptions in normal flow of internet traffic.Today,Mobile devices are present in every nook and corner of the world and privacy of users and their data is of utmost importance.Android is the most used and popular platform for mobile devices.

This report is focused on implementation of Reinforcement Learning for Android Malware Detection. The implementation hes been done using Drebin dataset for Android malware detection. Many different published research papers have been 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

Android has been detected with a number of malwares in the past few years. This number has increased gradually in the past years, unlike other OS platforms of malware that targeted other mobile platforms. The main cause behind this was the open source policy of Android . Android easily allows a malware to be inserted in an Android app. There has been a wide range of researches for detection of malware in Android. After various researches, it was concluded that there are some patterns on basis of which we can divide the malware detection techniques to be effective or not. However,these researches could not find an effective technique which would give satisfactory results for detection of unknown malware files. Deep learning methods have also been studied for detecting malware detection. These methods extract features from famous Android apps,some of which contain malware and some of them don't. After extracting features,machine learning algorithms were implemented on them to train the model on these features and to test these models for detect unknown malware files in Android apps.[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.

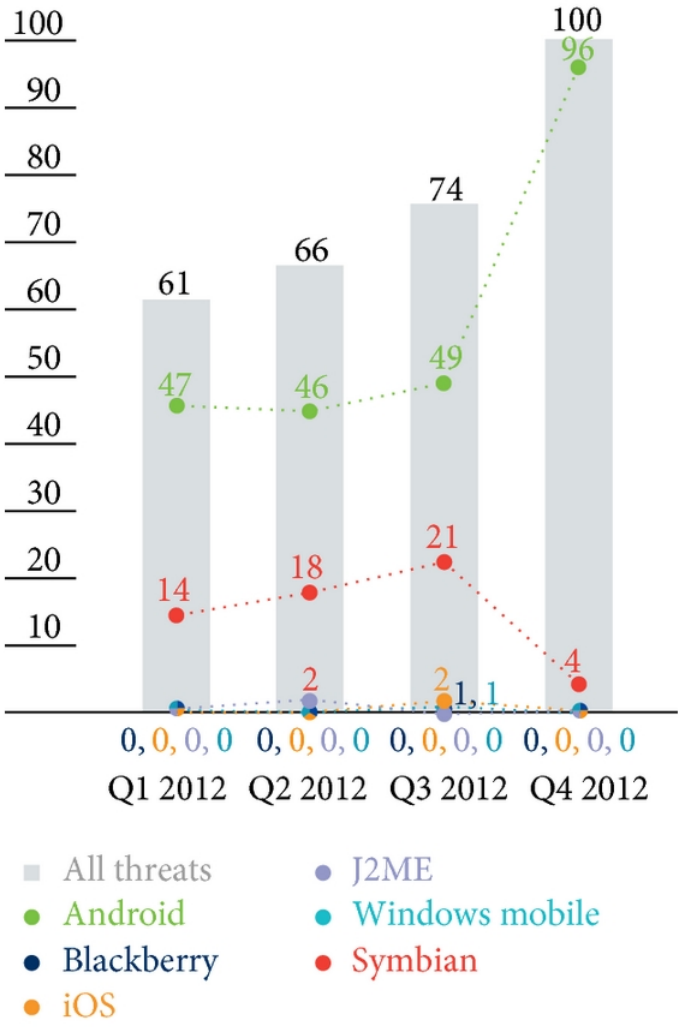

Figure 1: The figure shows comparison between malware samples collected from different devices with varied mobile OS platforms(218 samples).They were analysed in four quaters of year 2012.Other platforms showed a decrease in malware samples while Android showed an increase in amount of malware.[1]

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:** Trojan appears to be valid like a normal program or application but when opened or executed,it installs malware on the device.It may spread to other parts of the device.

2. **Spyware:** Spyware is encompassed in the original software and deployed to users.The main aim of spyware is to collect data about people,groups and organizations without their known consent or by fooling them. It can also gain control of the device secretly without users' knowledge.

3. **Root permission acquisition (exploit):** Some applications trick user into giving root permission .This gives hackers or exploiters almost full access to the data and control of the device.

4. **Adware:** It keeps opening different pop-ups and tries to redirect users to misleading sites and malware.

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

As Android is open-source,it is more vulnerable to malware attacks.Most of the mobile phones are Android based.In 2013, Vinit B. Mohata1 and Dhananjay M. Dakhane advised 5 types malware detection techniques : Signature Based, Behavioural-based, Specification-based, Data-Mining based, Cloud Based[2]. In signature Based, malware detection is done during program compilation.Semantics are ignored ,thus,giving malware code a chance to go undetected.In 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. In behavioural-based, different malware families are studied on a target system. A classifier is trained to detect malware behaviour from normal app behaviour.In data-Mining based, defined patterns are detected from large amount of data and classifiers are used for malware detection. In 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. Another research by Ankita Kapratwar, Fabio Di Troia and Mark Stamp showed that features could be extracted by analysing statically or dynamically[3].To find static features,apk files are reverse engineered. AndroidManifest.xml contains several features useful for static analysis.Android has total 135 permissions.A binary vector is be created from each apk file and stored in dataset. Dynamic analysis uses system calls .Android emulators are used to find these system calls.

The drebin dataset has data from August 2010 and October 2012. There were 15,036 applications out of which 5,560 were malware and 9,476 were benign[4]. In 2014, Hyo-Sik Ham, Hwan-Hee Kim, Myung-Sup Kim, and Mi-Jung Choi used Linear SVM-Based Android Malware Detection for Reliable IoT Services[5]. In this model, a classifier is generated to detect Android APK files for malware.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.For SVM True Positive Results came to be 0.999 with 99.7% accuracy and precision of 0.992. Further ,some researchers build a model for the same using deep learning[6].A feed forward network is trained using back-propogation mechanism. A dataset containing 37,107 samples collected from Opera Mobile Store over a period of 9 months

was used.For classification purpose, final fully connected layer used sigmoid function and the model gave accuracy of 94 % with 0.834 precision.In 2018, Giovanni Apruzzese, Michele Colajanni build a model for 3 cases of malware detection: intrusion, malware, and spam[7].This paper explains different types of Machine Learning techniques namely Deep and Shallow Learning. It discusses which ML algorithms work best for different areas of Cyber Security. Recently, a model was build using Random Forest for malware detection.The model achieved an F1 Score of 98.24%[8]Another research done by Suleiman Y. Yerima, Sakir Sezer, Igor Muttik used a combination of linear classifiers. It yielded them a maximum accuracy of 98.8%[9].Researchers aser Peiravian, Xingquan Zhu observed that if app requests a dangerous or higher level permission instead of requiring it while installation, it may be malware[10].

Feature selection is divided into four steps namely subset generation,subset evaluation,stopping criteria and result validation. Redundant features were eliminated using IB1,Naive Bayes,C4.5 Decision Tree,RBF Networks.Features were ranked using Information Gain, Gain Ratio etc[11].In a research,random forest algorithm was used for feature selection of a breast cancer dataset ,diving the dataset using n-cross validation and using backward elimination approach for feature ranking[12].For using RL as classifier,agent with same class as training input selects actions to maximize reward for that instance and other agents will minimize their rewards.Class was predicted using maximum of value function of the state [13].In another study,the problem was defined into a conditional probability problem using past observations.T classifiers were used for T steps.An action was selected by which expected sum of rewards is maximized [14].A study of Q-learning described it as model-free reinforcement learning.Bellman's equation is used to calculate Q-value of every state and best action was chosen by selecting state with maximum q-value [15].

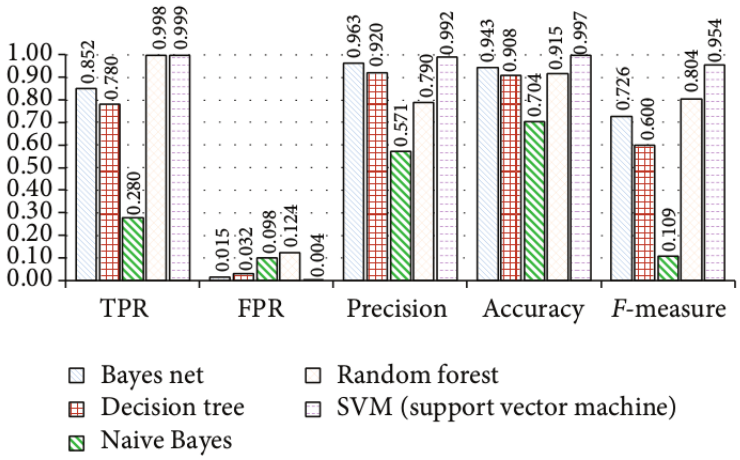

Figure 2: The figure shows comparison between outputs of different machine learning classifiers namely,Bayesian networks, Decision Tree ,Random Forest, Naive Bayes and SVM.It can be seen that SVM gives better results than others.[5]

5 Proposed Methodology

In this paper we are major focus was to implement the Reinforcement Model for Cyber Security - Android Malware Detection. There has not been much research and development in the field of

cyber security using reinforcement learning. We divided our implementation into 3 categories :-

1. **Dataset selection** - Drebin dataset.
2. **Feature Selection** - using Random Forest Classifier and Extra Trees Classifier.
3. **Implementing Reinforcement learning** - using Q- learning

5.1 Dataset - The Drebin Dataset[16]

We would be using The Drebin Dataset to train our Reinforcement Learning Model. The dataset consists of 215 attributes and their feature vectors. It is extracted from 15,036 applications consisting of 5560 malware files collected from August 2010 to October 2012 and 9,476 benign files. 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 decision tree is characterised by randomly selecting vairiables and data. For prediciting a particular tree in a forest we use the remaining dataset. 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 Implementing Reinforcement Learning

Reinforcement Learning iteratively trains the agent and increases the learning experience of the agent which distinguishes it from other supervised learning algorithms.[17] RL is defined by a tuple of state, reward and action. Agent interacts with the environment for a particular set of state and

reward. It performs an optimal action and environment gives a reward or a penalty. Environment iteratively returns a new set of state and reward to the agent. Our aim is to maximize the reward and gradually filter the bad actions and provide a set of suitable action for a particular state. [18]

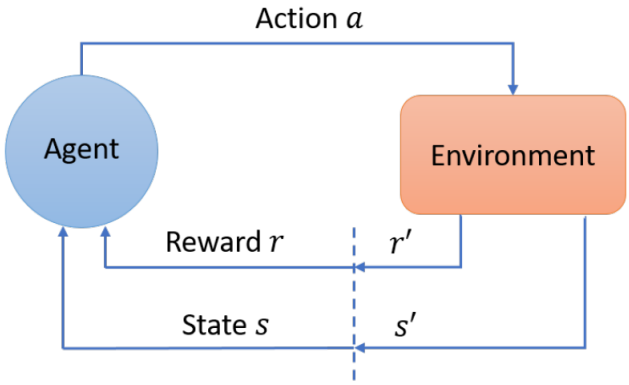

Figure 3: RL has an environment, where an agent interacts for a particular set of state and action. The agent takes an optimal action on the state and gives a reward. The environment then returns the agent a new set of state and reward. Agent again takes a new optimal action and gives a reward,this process takes place iteratively until a terminal state is reached.[17]

5.3.1 Markov Decision Processes MDP's [19]

Mathematical formulation of Reinforcement Learning can be described using 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 [19].

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

5.3.2 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.

5.3.3 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.

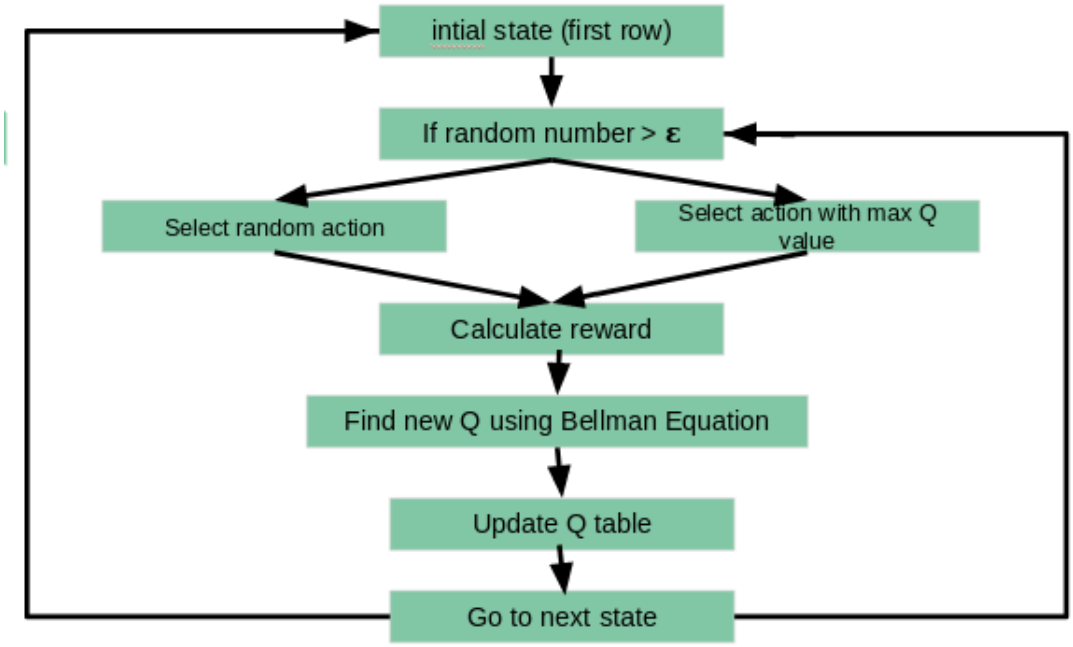

Figure 4: Flowchart of Q Learning algo implementation.

6 Results

Our major focus in this paper was to apply RL algorithm(Q-learing) in the field of cyber security as major work/research has not been done in this field. The accuracy of the model varied with different values of learning rate.
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.

6.1 For Random Forest Classifier

In Random Forest Classifier the accuracy decreased with an increasing learning rate. It performed the best at a learning rate of 0.00039 with an accuracy of 87.652% .

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

6.2 For Extra trees Classifier

In Extra trees Classifier the accuracy decreased with an increasing learning rate. It performed the best at a learning rate of 0.00039 with an accuracy of 91.287% .

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

The following graph depicts the variation of accuracy with learning rate using Extra Trees Classfier.
.

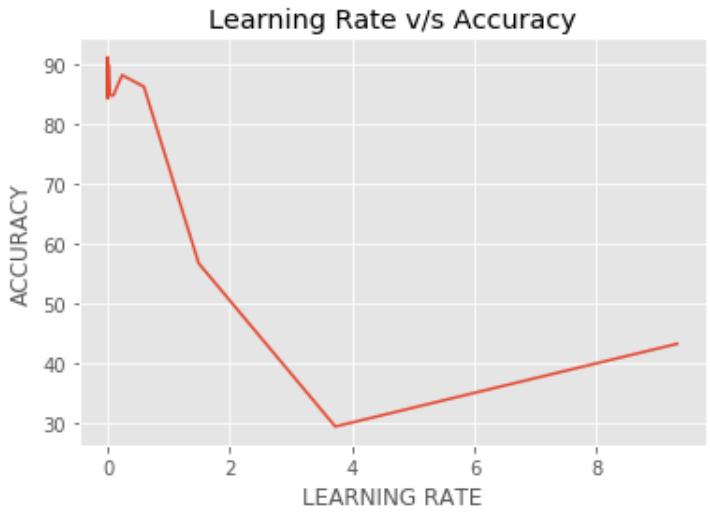

Figure 5: Plot of Learning rate v/s Accuracy using Extra Trees Classfier.

The results differed if we used different feature selection methods. Extra tree classifier performed better at learning rate of 0.00039 with an accuracy of 91.287%, whereas 87.652% using Random Forest Classifier. Therefore, we used Extra Trees Classifier for feature selection and our model showed an accuracy of 91.287% at learning rate of 0.00039.

6.3 Reinforcement Learning vs other algorithms on Drebin dataset

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. Through the works of various researchers, SVM performed with an accuracy of 97.2%, Random Forest had an accuracy of 94.17%, Decision Tree had an accuracy of 92.87%, Gradient Boosting

had an accuracy of 89.96% and LSTM had an accuracy of 70%. Our model had an accuracy of 91.287 % and performed better than Gradient Boosting and LSTM on Drebin dataset.

Algorithm Used	**Accuracy**	**F1 score**
SVM[20]	97.2 %	0.92
Random Forest[21]	94.17 %	0.94
Decision Tree[21]	92.87 %	0.93
Reinforcement Learning (Q-Learning)	91.287 %	0.873
Gradient Boosting[21]	89.96 %	0.89
LSTM[22]	70 %	0.104

6.4 Reinforcement Learning vs other algorithms on similar domain dataset

In the past years, various researchers have build models to classify benign and malware files using similar domain dataset with different approaches like machine learning, deep neural networks, etc. Through the works of various researchers, SVM performed with an accuracy of 99.5%, 5-layered DNN had an accuracy of 94%, Random Forest had an accuracy of 91.4%, XGBoost had an accuracy of 74.1% and Naive Bayes had an accuracy of 70.4%. Our model had an accuracy of 91.287 % and performed better than XGBoost and Naive Bayes on similar domain(Android Malware Dataset) datasets .

Algorithm Used	**Accuracy**	**F1 score**
SVM[5]	99.5 %	0.954
5-layered DNN[6]	94.0 %	0.851
Random Forest[8]	91.4 %	0.79
Reinforcement Learning (Q-Learning)	91.287 %	0.873
XGBoost[6]	74.1 %	0.134
Naive Bayes[6]	70.4 %	0.109

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 Pg. 8 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

20. Ayoubianzadeh, Zahra, and Vali Derhami. "Android malware detection using combination of Support vector machines and fuzzy logic."

21. Rana, Md Rahman, Sheikh Shah Mohammad Sung, Andrew. (2018). Evaluation of Tree Based Machine Learning Classifiers for Android Malware Detection: 10th International Conference, ICCCI 2018, Bristol, UK, September 5-7, 2018, Proceedings, Part II.

22. Hota, Abhilash, and Paul Irolla. "Deep Neural Networks for Android Malware Detection." (2019).