Reinforcement Learning for Android Malware Detection

Project Supervisor :

Dr. Om Prakash Vyas

Group Members :

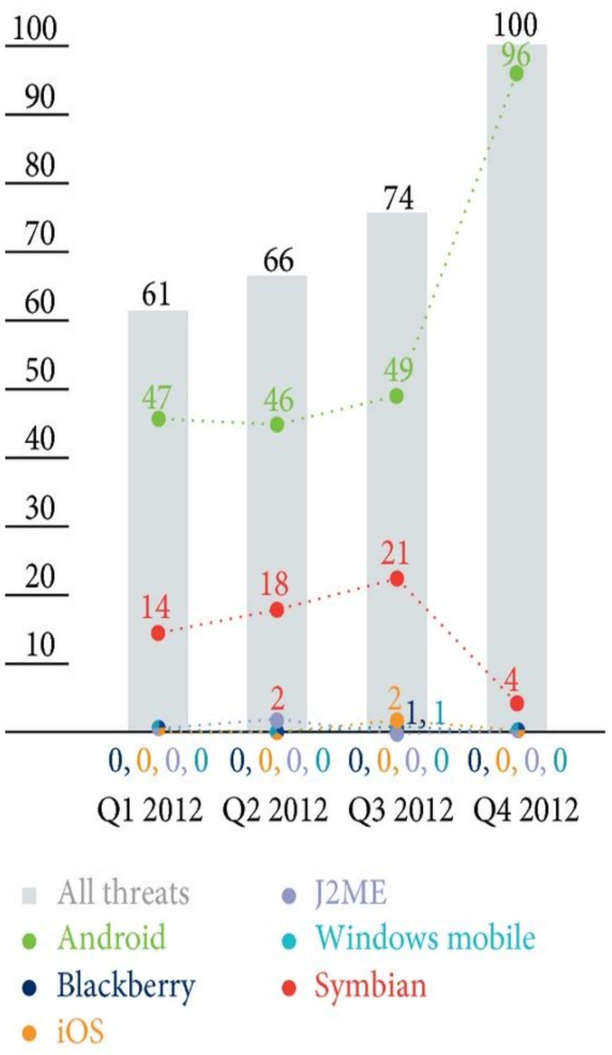
IIT2016053 - Surabhi Gogte
IIT2016088 - Simran Gill
IIT2016103 - Chahak Sharma

Problem Statement

Propose a reinforcement learning model for Android Malware detection and compare the accuracy with the previous Machine Learning and Deep Learning models.

Motivation

- Recent studies show that the amount of malware that targeted other mobile platforms gradually decreased , whereas 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 .
- The main motivation behind this research is to apply reinforcement learning for android malware detection and compare it with the previous works done using Deep Learning.



LITERATURE REVIEW

Android Malware Types :

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.

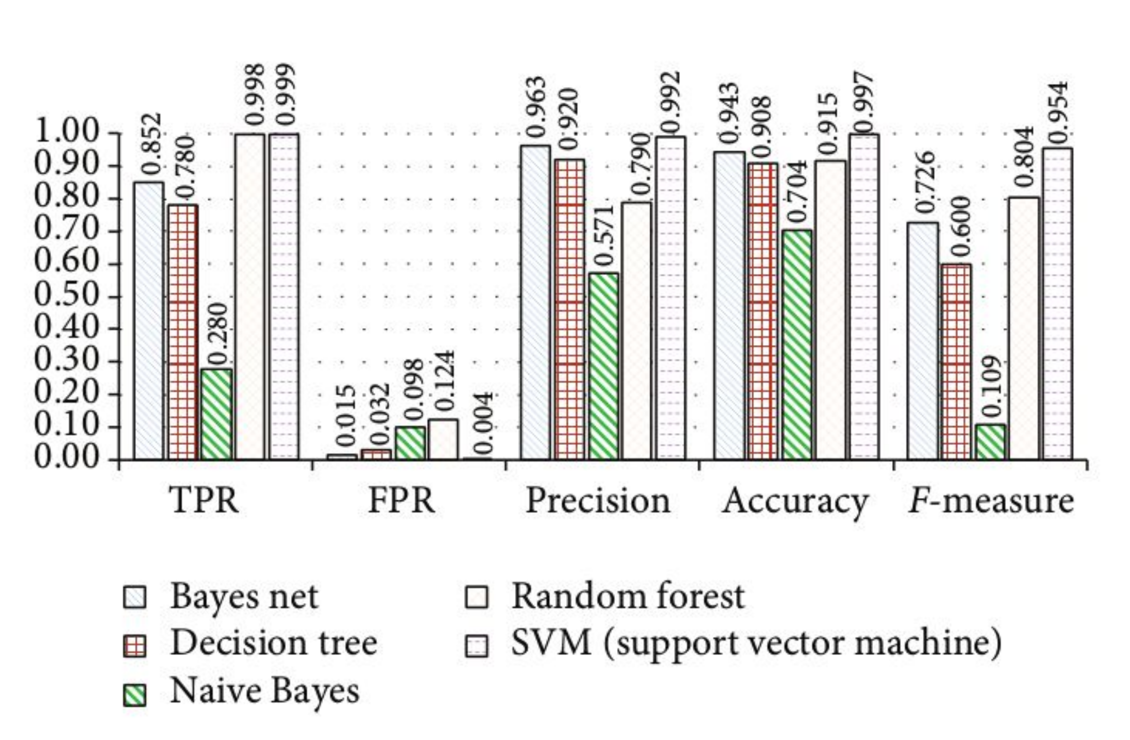
3. **Root permission acquisition (exploit):** it acquires root permissions to clear security settings and make additional attacks.

4. **Installer (dropper):** it conceals malware in a program and guides users to run malware and spyware.

Linear SVM

- Traditionally behavior based analysis technique have been used for malware detection.
- Behavior-based detection involves the inconvenience of having to determine malware infection status by examining numerous features.
- SVM has high performance and can classify non linear data.
- 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 .
- 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.

Comparison with other ML Algorithms



Feature Selection using Random Forest Classifier

- The dataset is divided into training and testing set using n-cross validation.
- Bayesian Probability of each feature is calculated.
- The ranks of features is calculated based on ranking value.
- 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

Reinforcement Learning for Solving Classification Problems

- In this paper, RL is combined with multilayer perceptrons to find Value function of each state .
- x^i is taken as a input vector of length m and y^i is target class belonging to the input.We have a dataset D = $\{(x_1 , y_1), . . . , (x_n , y_n)\}$ of labeled examples.
- There is a single reward function that is independent of the target class.
- The agent with the same class as a training instance will select actions to maximize its obtained rewards, whereas an agent of another class will select actions that minimize its obtained rewards.
- For testing purposes ,all values $V_i(s_0)$ for all classes i and agents AC_i belonging to these classes.
- The input vector is classified with the predicted class y_p belonging to the agent with the largest state value: $y_p = \arg\max V_i(s_0)$.

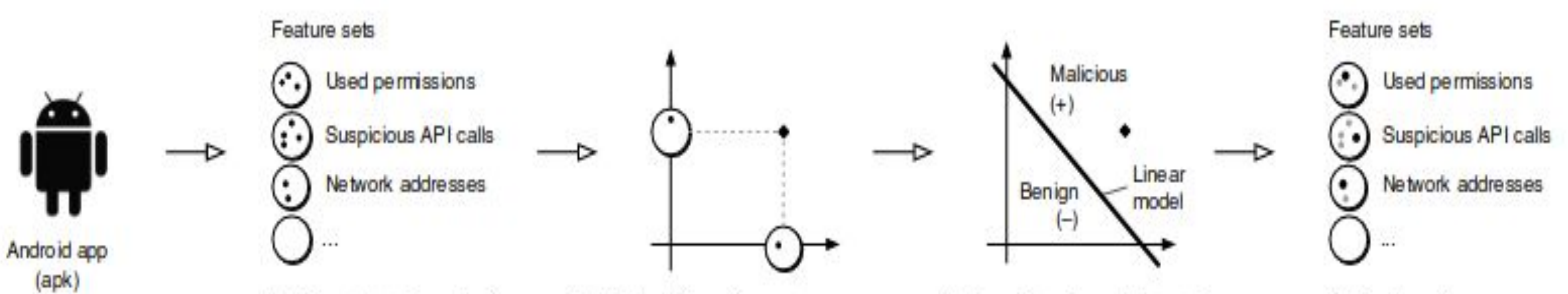
PROPOSED METHODOLOGY : Reinforcement Learning

Dataset

- 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.
- Drebin is one of the most popular benchmark datasets for Android malware detection.

Drebin - Dataset

- Gathers features from an APK files or application’s code
- Embedded these into a joint vector space .
- Further applied SVM for learning based detection .
- For each detected application the respective patterns can be extracted, mapped to meaningful descriptions and then provided to the user as explanation for the detection.



(a) Broad static analysis (b) Embedding in vector space (c) Learning-based detection (d) Explanation

Feature Selection

- More the number of features, more is the chance of decreased accuracy and increased training time.
- Redundant features need to be removed and we need to select top features for classification.
- We used 2 methods for feature selection-
  - Random Forest Classifier (Accuracy - 87.5%)
  - Extremely Randomised Tree Classifier(Extra Trees Classifiers). (Accuracy - 91.25%)
- In both,importance of each feature is calculated and the top ranked features are selected.

Reinforcement Learning Preliminary

- Concept of state,action, and reward.
- It is a trial and error approach .
- Agent takes action at each time step that causes two changes :
  - current state of the environment is changed to a new state,
  - 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.

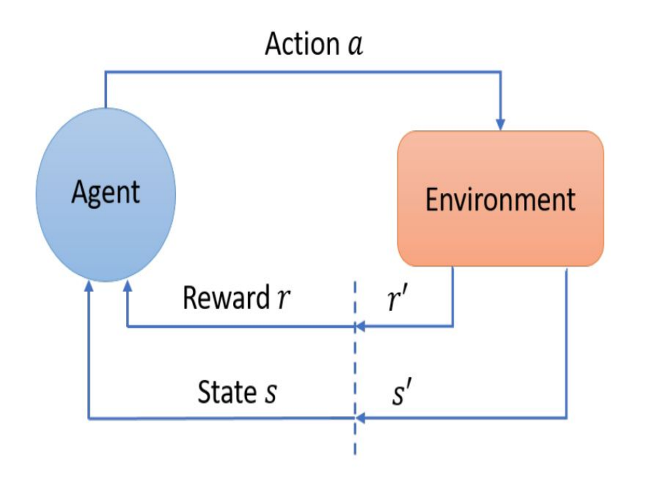


Fig. Iterative process of agent-environment interactions.

Markov Decision Processes (MDP's)

Formally RL can be described as Markov Decision Processes (MDP's) which consists of: (S, A, T, R, Ɣ)

- set of **states** S .
- set of **actions** A,
- **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.
- an instantaneous **reward function** $R(s_t, a_t, s_{t+1})$.
- a **discount factor** Ɣ between 0 and 1 : this quantifies the difference in importance between immediate rewards and future rewards.
- **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.

MDP Formulation

(S, A, T, R, γ)

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

Algorithm used: Q-Learning

Q-learning uses Q(s,a) to iteratively improve the behaviour of agent.

1. Q(s,a) : is the estimation of how good it is to take action a on state s.
2. Reward and Episode : At every step of state transition , agent receives a reward . When agent is at one of its terminating state , an episode is said to end.
3. Bellman Equation
4. Choose action based on ϵ - greedy policy : either take an action with max q value or perform a random action .

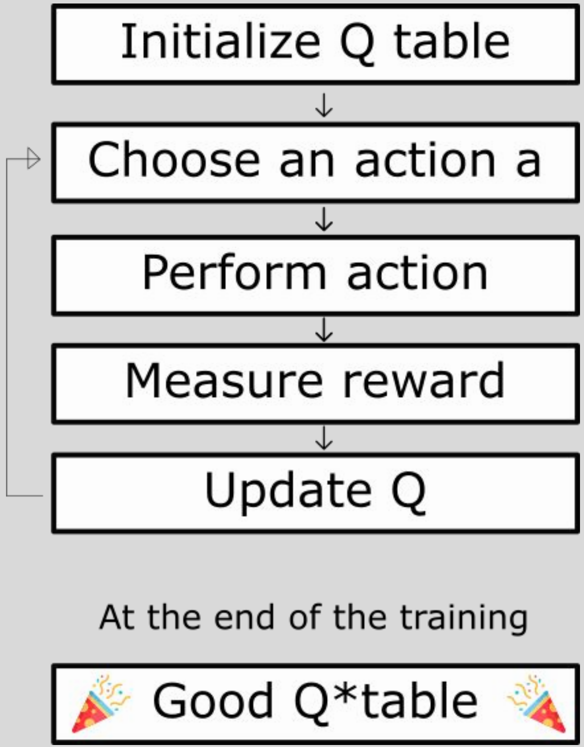
The Bellman Equation

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

- **Q(s,a)** : old q value
- **α** : learning rate
- **R(s,a)** : reward at state s and action a
- **ɣ** : discount factor
- **Max Q(s',a')** : estimate of optimal future value

We calculate the new Q value for state s , when a action a is performed.
We maintain a **Q table** to store the q value of each state-action pair.

Algorithm of Q-learning

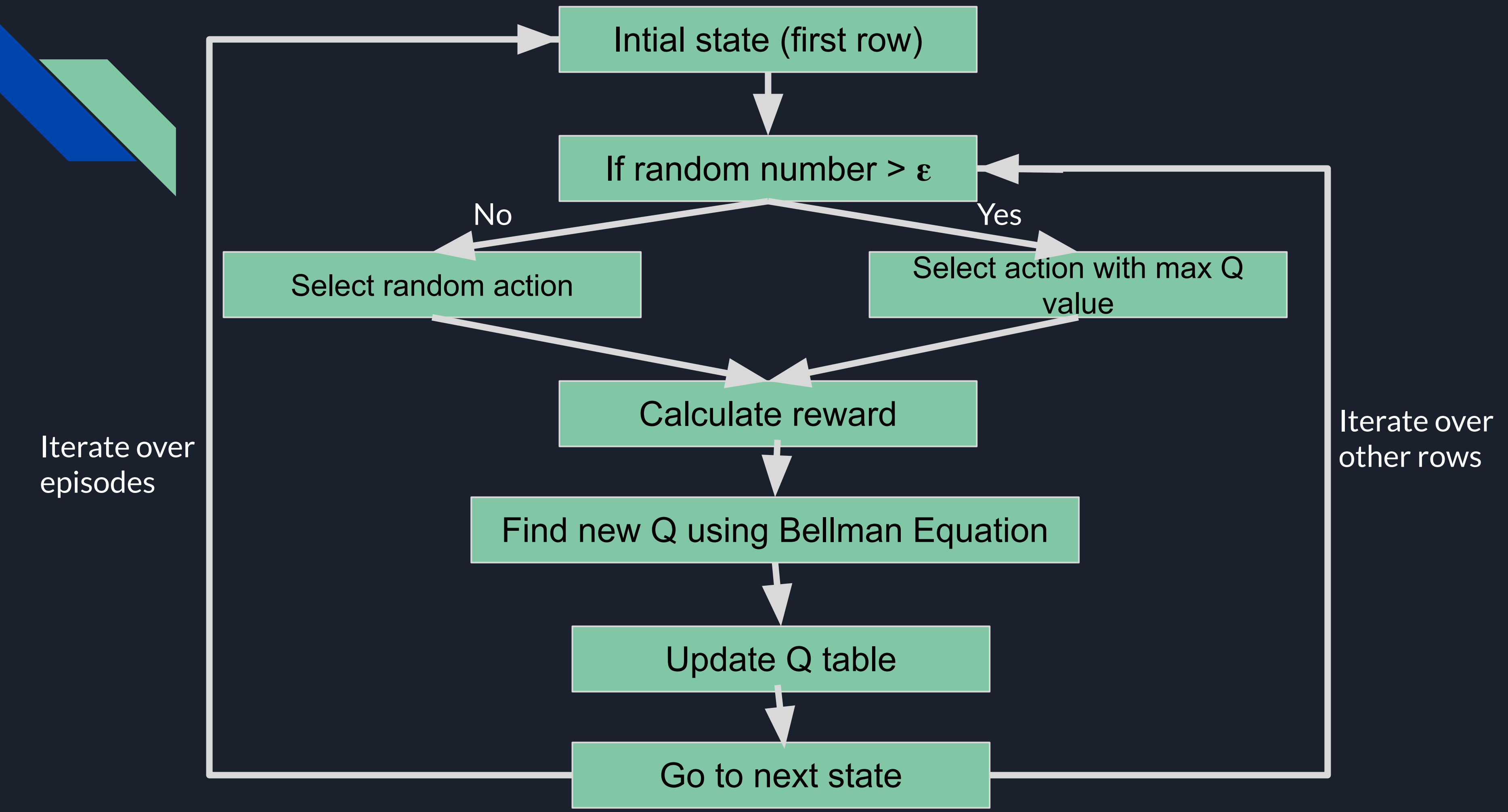


Implementation

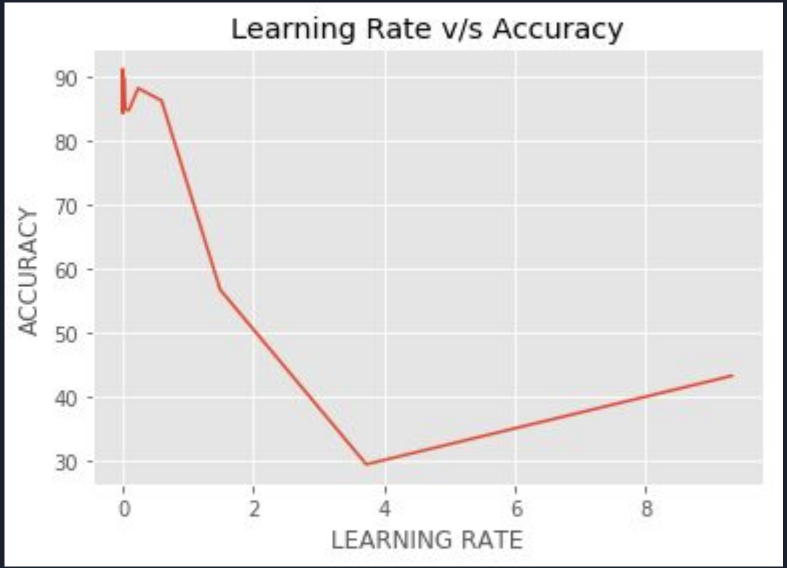
- Select features from dataset .
- Cleaning the dataset.
    a. Check for NULL values.
    b. Replace S and B with 0 and 1 for easy processing.
    c. Shuffle the entries of dataset.
- Divide dataset into 70% training and 30% testing.
- Create a table that maps each entry in x to a unique number for reference later .

Implementation

- Make a **Q - table** with dimensions = No_of_combinations* no_of_actions , initialised with value 0 .
  - No_of_combinations = 2^no_of_features.
  - No_of_action = 2 (either malware or not).
- **Action function :** if selected action type = 0 then no malware , else malware.
- **Reward Function :** if the action matches the actual entry in Y_train then Reward of 1 , else Penalty of -1.


Intial state (first row)
If random number > ε
No
Yes
Select random action
Select action with max Q value
Calculate reward
Find new Q using Bellman Equation
Update Q table
Go to next state
Iterate over episodes
Iterate over other rows

Results



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

Results-

Using 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

Using 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

Results

Reinforcement Learning vs Other Algorithms

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

Project Timeline

- **Aug** — Literature Survey
- **Sep** — Selection of dataset
- **Oct** — Application of Reinforcement Q - learning
- **Nov** — Results and Comparison

References

1. Mobile Threat Report Q4 , 2012
2. Hyo-Sik Ham, Hwan-Hee Kim, Myung-Sup Kim and Mi-Jung Choi. Linear SVM-Based Android Malware Detection for Reliable IoT Services 2014
3. Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic." Journal of Biomedical Science and Engineering 6.05 (2013): 551.
4. Reinforcement learning algorithms for solving classification problems." 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL). IEEE, 2011.
5. Arp, Daniel, et al. "Drebin: Effective and explainable detection of android malware in your pocket." Ndss. Vol. 14. 2014.
6. Thanh Thi Nguyen and Vijay Janapa Reddi. Deep Reinforcement Learning for Cyber Security, 2019
7. A. Gosavi Reinforcement Learning: A Tutorial Survey and Recent Advances, 2009
8. Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep Reinforcement Learning : A brief survey, 2017.

THANKYOU !