Performance Analysis of Machine Learning Classifiers in Android Malware Classification

Zarni Aung[#1], Win Zaw[*2]

[#1]*Faculty of ICT, University of Technology (Yatanarpon Cyber City)*
Myanmar
[1]zarniaung.utycc@gmail.com

[*]*Department of Information Technology, Technological University (Than Lynn)*
Myanmar
[2]winzaw@gmail.com

***Abstract*— Mobile phones are used as the central computing and communication device today. They are started to replace traditional computers because of their superior mobility and nearly omnipresent Internet access. At the same time, Android is the latest technology for mobile OS that is getting a lot of attention, and its popularity is growing exponentially each year. There has been increased in android applications because android phones are widely used and they have become more popular than other platforms. Unfortunately, such phenomenon becomes an ideal target for malware developers so that android malware applications have rapidly increased. Therefore, many researchers developed malware detection framework to detect malicious applications. In this paper, an android malware classification system which classifies android applications (normal or malicious) is proposed. Our proposed framework intends to evaluate the performance of machine learning classifiers in android malware classification.**

***Keywords*— android, malware application, machine learning, classification**

I. INTRODUCTION

Since Google developed an android platform, android phones have increased all over the world. They are widespread and steadily gaining popularity so that android applications have also rapidly increased. Anyone can develop android applications, upload the developed applications on the websites of android applications market and download easily free applications. Such phenomenon attracts the malicious application developers so that malware applications have rapidly increased. But, the general android phones users can't distinguish which applications are malicious or not. If they downloaded and installed malicious applications in their devices, they could cause serious damages, such as making the phone unusable, stealing private information, and sending SMS or phone call for unwanted billing, to users. Therefore, it is crucial to classify the android applications whether they are malicious or normal.

To detect android malware applications, the researchers proposed many malware detection frameworks but they didn't consider the facts that are based on permission requests of android applications. Every android application produces permission requests at installation time to confirm the user's allowance. Some applications can have unnecessary and dangerous permission requests but the users who are unfamiliar with android applications can allow these permission requests. If these applications were malicious applications, they could damage the resources of mobile devices. This paper proposes a machine learning based malware classification framework to classify which applications are malicious or normal by using machine learning classifiers.

The remainder of this paper is organized as follows. Section II describes the related work of malware detection frameworks and the different types of malware detection techniques are presented in Section III. After that, the proposed framework is described in Section IV and Section V presents about the machine learning classifiers. The brief experimental results are given in Section VI. Finally, we conclude the paper and suggest future work in Section VII.

II. RELATED WORK

There has been a significant work on the problem of detecting malicious applications on mobile devices. Many researchers developed android malware detection frameworks by using machine learning techniques. [8] presented a machine learning-based system for the detection of malware on Android devices. This system extracts a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner. They used a classifier that can be trained only one class and they can classify only normal applications. Juxtapp, is a scalable infrastructure for code similarity analysis among android applications [1]. It provides a key solution to a number of problems in Android security, including determining if applications contain copies of buggy code, have significant code reuse that indicates piracy, or are instances of known malware. Several approaches [2], [4], [9] monitor the power usage of applications, and report abnormal consumption. Others [7], [10] monitor system calls and attempt to detect unusual system call patterns. The new method for categorising Android applications through machine-learning techniques is proposed in [3]. To represent each application, different feature set are extracted from the Android Market and they evaluated their approach of automatically categorization of Android applications.

III. Malware Detection Techniques

There are three malware detection techniques to detect android malware applications. These techniques available for detecting mobile malware and other security vulnerabilities have varying strengths and weaknesses.

A. *Static Analysis*

Static analysis is a quick, inexpensive approach to find malicious characteristics or bad code segments in an application without executing them. The techniques based on system calls, taint and source codes are widely used in a preliminary analysis, when suspicious applications are first evaluated to detect any obvious security threats [6].

B. *Dynamic Analysis*

Unlike static analysis, dynamic analysis involves executing the mobile application in an isolated environment, such as a virtual machine or emulator, so that researchers can monitor the application's dynamic behavior. Researchers primarily use dynamic analysis in taint tracking or system call tracing [6].

C. *Application Permission Analysis*

Permissions play a key role in mobile applications and they convey the application's intentions and back-end activities to the user. In smartphones, permissions are clearly defined, so application authors must acquire appropriate permissions. However, some authors purposely hide the permissions they use in the application leading to application vulnerability [6].

IV. Proposed Framework

In this paper, we propose a malware application classification framework to evaluate the performance of different machine learning classifiers. This framework is based on feature extraction as a first phase, dataset generation as a second phase, classification of this new dataset which is generated by second phase as third phase, and finally evaluating the performance of this proposed model in terms of accuracy, precision and recall. The proposed framework for android malware classification is shown in Fig. 2.

A. *Feature Extraction*

Firstly, android application files (including malicious applications) are collected by downloading from the websites of android application market such as Google Play. Then, these downloaded application files are decompressed into the contents of android application files: classes.dex, AndroidManifest.xml, resources.rsc. Android application files are mainly focused on only two components: classes.dex and Androidmanifest.xml. Android application produces permission requests at the installation time to confirm the allowance of the users. Application authors were developing their applications with appropriate permissions. However, some authors purposely hide the permissions they use in the application leading to application vulnerability. Most of the permission features are included in AndroidManifest.xml and classes.dex files. This proposed framework will extract these permission-based features from these important files by using python script file.

B. *Dataset Generation*

The permissions extracted from the first stage are used to create android malware dataset. This dataset is developed in ARFF (Attribute Relation File Format). This file consists of two parts. The first part contains the set of attributes and the values of these attributes are described in the second part. The design of dataset generation from android applications is described in Fig. 1.

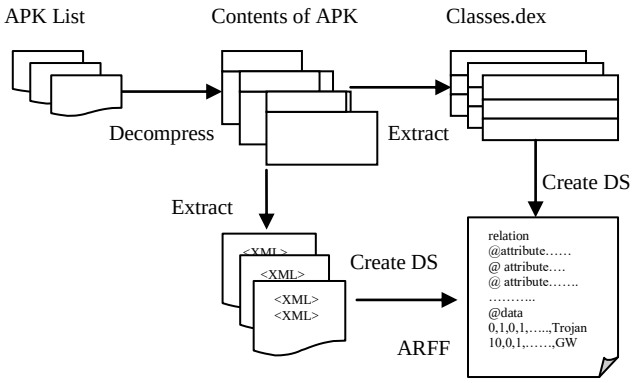


Fig.1. Feature Extraction and Dataset Generation from Android APK

C. *Malware Classification*

To predict and classify potentially android malware applications, various machine learning classifiers have been utilized. There are at least three techniques which are used to calculate a classifier's accuracy [11]. One technique is to split the training set by using two-thirds for training and the other third for estimating performance. In another technique, known as cross-validation, the training set is divided into mutually exclusive and equal-sized subsets and for each subset the classifier is trained on the union of all the other subsets. The average of the error rate of each subset is therefore an estimate of the error rate of the classifier. Leave-one-out validation is a special case of cross validation. All test subsets consist of a single instance. This type of validation is, of course, more expensive computationally, but useful when the most accurate estimate of a classifier's error rate is required.

V. Machine Learning Classifiers

Specifically, the analysis procedure takes into account and cross-evaluates three supervised machine learning algorithms, i.e. Bayesian Network, RBF Network and Random Forest. A Bayesian network, also called a belief network model, is an annotated directed graph that encodes the probabilistic relationships among variables of interest. A Bayesian network classifier is a statistical classification eager method [5] that may be used as a classifier that gives the posterior probability distribution of the class node given the values of other features.

On the other hand, RBF is a type of ANN that consists of an input layer, a hidden layer and an output layer. Specifically, RBF is a single hidden layer feed-forward network and has a

static Gaussian function as the nonlinearity for the hidden layer processing elements [11].

Finally, Random Forest is well-respected amongst the statistics and machine learning communities as a versatile eager method that produces accurate classifiers for many types of data [1]. Random forests (RF) are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlations between them. Using a random selection of features to split each node yields error rates that compare favourably to Adaboost, and are more robust with respect to noise. A common method for comparing supervised ML algorithms is to perform statistical comparisons of the accuracies of trained classifiers on specific datasets.

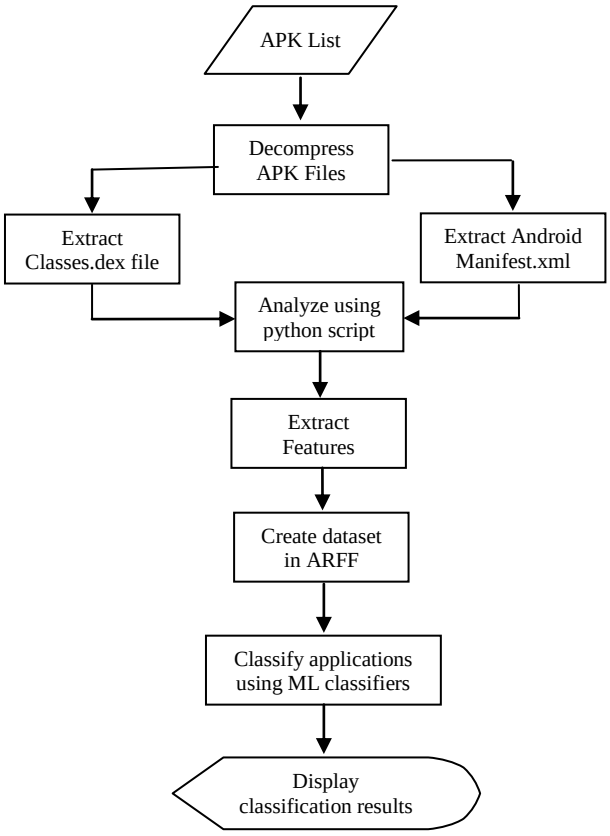


Fig.2: Proposed Framework for Android Malware Classification

VI. EXPERIMENTS

Supervised machine learning techniques are applicable in numerous domains. We tested our system against a collection of 500 sample Android applications. We created two datasets with 200 and 500 android applications by extracting features and data analysis was carried out using the well known machine learning software package namely Waikato Environment for Knowledge Analysis (Weka) [14] to evaluate the performance of machine learning classifiers.

A. *Performance Evaluation Criteria*

We used the machine learning techniques to classify the android malware applications. Firstly, we built ARFF file from the extracted features and used decision tree machine learning algorithms to classify the malware applications. From the response of classifiers, relevant confusion matrices were created. The following four metrics define the members of the matrix.

True Positive (TP) : number of correctly classified goodware applications.

False Positive (FP) : number of incorrectly classified malware applications.

True Negative (TN) : number of correctly classified malware applications.

False Negative (FN) : number of incorrectly classified goodware applications.

True Positive Rate (TPR) : percentage of correctly classified goodware applications

$$(TP / TP+FN)$$

False Positive Rate (FPR): Percentage of incorrectly classified malware applications

$$(FP / TN+FP)$$

Overall Accuracy (ACC): Percentage of correctly classified applications

$$(TP+TN / TP+TN+FP+FN)$$

The performances of machine learning techniques were evaluated using the true positive rate, false positive rate, precision, recall and overall accuracy which are defined above.

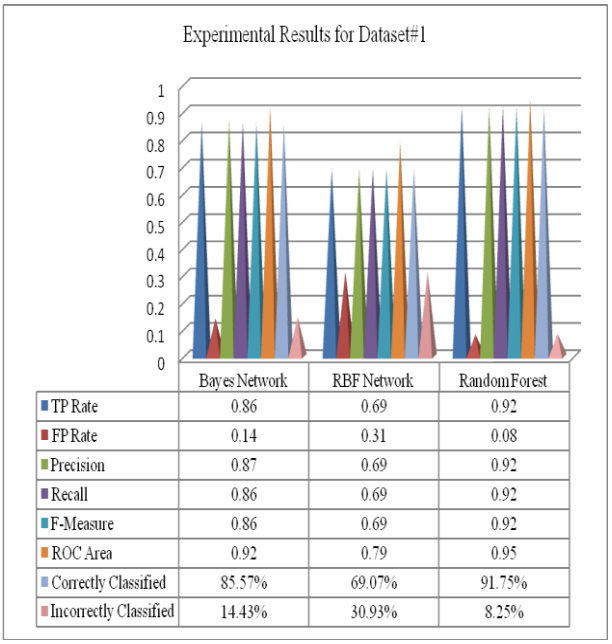
B. *Experimental Results*

Firstly, we extracted the necessary features to analyze from sample applications (goodware and malware). Then, we built android malware dataset in (.arff) file format from the extracted features. We used these two datasets to distinguish malware and goodware applications by using machine learning approaches. The performance of different machine learning classifiers from two datasets and the different types of error rate of these machine learning classifiers are shown in the following figures 3, 4, 5 and 6.

VII. CONCLUSION AND FUTURE WORK

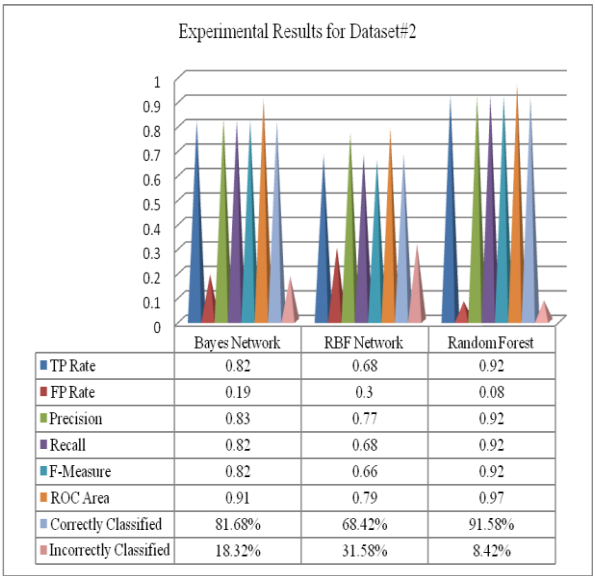
Mobile phones are becoming an essential part in communication, and they are starting to replace traditional computers because of their superior mobility and nearly Internet access. At the same time, Android is getting a lot of attention, and its popularity is growing exponentially each year. Because of the popularity and ubiquity of mobile phones, malware authors are starting to develop new threats for this platform that are being actively distributed via what is supposed to be a trusted source: the official Android Market. Therefore, many researchers became to emphasize these malware applications and developed different malware

detection frameworks. We propose a framework for classifying android applications using machine-learning techniques. To generate the models, we have extracted several features from several downloaded applications from android markets. Some of the malware applications are used from malware sample database and both malware and normal applications are classified by using machine learning techniques. Regarding future work, we will train models with larger dataset as soon as we obtain enough samples of malicious applications and we will extract more features from sample applications. We will even classify the types of malware applications (Trojan, Infosteal,.etc).



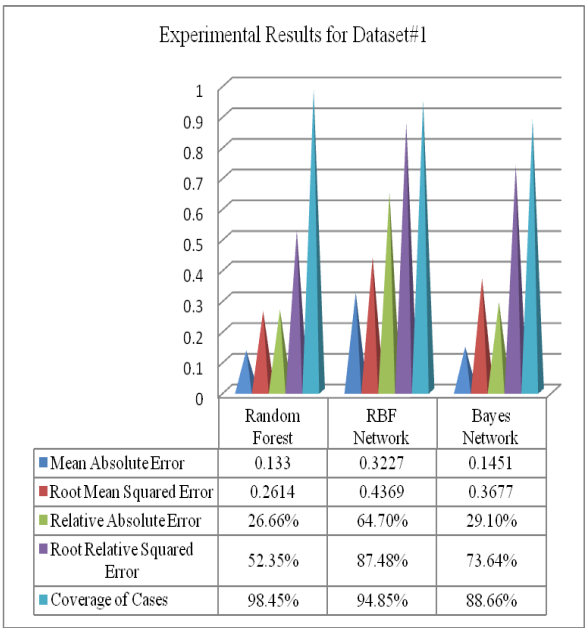
	Bayes Network	RBF Network	Random Forest
TP Rate	0.86	0.69	0.92
FP Rate	0.14	0.31	0.08
Precision	0.87	0.69	0.92
Recall	0.86	0.69	0.92
F-Measure	0.86	0.69	0.92
ROC Area	0.92	0.79	0.95
Correctly Classified	85.57%	69.07%	91.75%
Incorrectly Classified	14.43%	30.93%	8.25%

Fig.3.Performance of Machine Learning Classifiers for Dataset#1



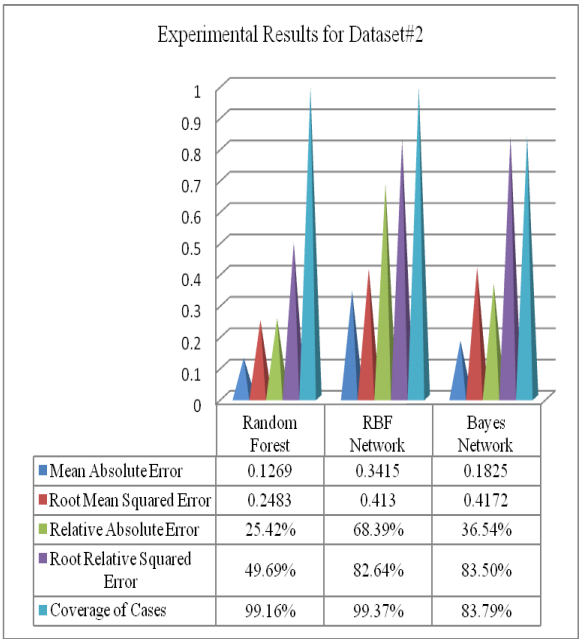
	Bayes Network	RBF Network	Random Forest
TP Rate	0.82	0.68	0.92
FP Rate	0.19	0.3	0.08
Precision	0.83	0.77	0.92
Recall	0.82	0.68	0.92
F-Measure	0.82	0.66	0.92
ROC Area	0.91	0.79	0.97
Correctly Classified	81.68%	68.42%	91.58%
Incorrectly Classified	18.32%	31.58%	8.42%

Fig.4: Performance of Machine Learning Classifers for Dataset#2



	Random Forest	RBF Network	Bayes Network
Mean Absolute Error	0.133	0.3227	0.1451
Root Mean Squared Error	0.2614	0.4369	0.3677
Relative Absolute Error	26.66%	64.70%	29.10%
Root Relative Squared Error	52.35%	87.48%	73.64%
Coverage of Cases	98.45%	94.85%	88.66%

Fig.5: Misclassification error of Machine Learning Classifiers for Dataset#1



	Random Forest	RBF Network	Bayes Network
Mean Absolute Error	0.1269	0.3415	0.1825
Root Mean Squared Error	0.2483	0.413	0.4172
Relative Absolute Error	25.42%	68.39%	36.54%
Root Relative Squared Error	49.69%	82.64%	83.50%
Coverage of Cases	99.16%	99.37%	83.79%

Fig.6: Misclassification error of Machine Learning Classifiers for Dataset#2

ACKNOWLEDGEMENT

I would like to thank my supervisor, Dr. Win Zaw, Associate Professor and Head of Department of Information Technology, for his excellent guidance, where the initial scope of the thesis was defined, and throughout the process of writing this paper. Then, I am especially thankful to Mr. Alejandro Mosquera, researcher at University of Alicante, Spain for his unvaluable advices and guidelines throughout my research.

REFERENCES

[1] Breiman L. (2001), Random Forests, Machine Learning 2001; 45(1): 5-32.

[2] Bryan Dixon, Yifei Jiang, Abhishek Jaiantilal, and Shivakant Mishra. Location based power analysis to detect malicious code in smartphones.In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 27–32, 2011.

[3] D. Kim, J. Kim, and S. Kim "A Malicious Application Detection Framework using Automatic Feature Extraction Tool on Android Market" on 3rd International Conference on Computer Science and Information Technology(ICCSIT'2013) January4-5, 2013 Bali, Indonesia

[4] Hahnsang Kim, Joshua Smith, and Kang G. Shin. "Detecting energy greedy anomalies and mobile malware variants." In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, pages 239–252, 2008.

[5] Heckerman D. A Tutorial on Learning with Bayesian Networks, Technical report 95-06, Microsoft Research Advanced Technology Division Microsoft Corporation, Redmond, USA, November. 1995.

[6] http://www.infoq.com/articles/detection-of-mobile-malware

[7] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, SPSM '11, pages 15–26, 2011.

[8] Justin Sahs and Latifur Khan "A Machine Learning Approach to Android Malware Detection" in 2012 European Intelligence and Security Informatics Conference

[9] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter: Preventing your cellphone from spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pages 244–264, 2009.

[10] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, and Sencun Zhu. pbmds: a behavior-based malware detection system for cellphone devices. In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pages 37–48, 2010.

[11] Neuro Dimension. Radial Basis Function. Available at: http://www.nd.com/models/rbf.htm

[12] S. B. Kotsiantis "Supervised Machine Learning: A Review of Classification Techniques" Informatica 31(2007) 249-268

[13] S. Hanna, L. Huang, E. Wu, S. Li, C. Chen, and D.Song "Juxtapp: A Scalable System for Detecting Code Reuse Among Android Applications" UC Berkeley ,Intel Labs

[14] The University of Waikato. Weka: Weka Machine Learning Project. Available at: http://www.cs.waikato.ac.nz/ml/weka