A FRAMEWORK FOR THE DETECTION OF BANKING TROJANS IN ANDROID

Subarna Adhikari, Sushil Nepal and Rabindra Bista

Dept. of Computer Science & Engineering, Kathmandu University, Dhulikhel, Nepal

ABSTRACT

Android is the most widely used operating system today and occupies more than 70% share of the smartphone market. It is also a popular target for attackers looking to exploit mobile operating systems for personal gains. More and more malware are targeting android operating system like Android Banking Trojans (ABTs) which are widely being discovered. To detect such malware, we propose a prediction model for ABTs that is based on hybrid analysis. The feature sets used with the machine learning algorithms are permissions, API calls, hidden application icon and device administrator. Feature selection methods based on frequency and gain ratio are used to minimize the number of features as well as to eliminate the low-impact features. The proposed system is able to achieve significant performance with selected machine learning algorithms and achieves accuracy up to 98% using random forest classifier.

KEYWORDS

Malware Detection, Android Banking Trojans, Hybrid Analysis, Machine Learning.

1. INTRODUCTION

In the last decade (2011-2021), Android has overtaken both Symbian OS and iOS to become the most popular mobile operating system in use occupying more than 70% of the smartphone market share [1]. The widespread use of android also means that more than 90% of the malware targeting smartphones are specific to android operating system [2]. Malware or malicious software is a program that is designed to gain illegal access or to cause damage to systems and their users [3]. Android Banking Trojans (ABTs) occupy a significant portion of android malware ecosystem with other major malware classes like spyware, ransom-ware and adware [4]. ABTs are designed to steal banking credentials of the users. For that purpose, they employ a range of tactics that include intercepting text messages and phone calls, hiding their presence once installed and creating fake overlays for legitimate banking applications [5].

Traditionally, malware detection systems use signatures to identify malware. Signature of a malware is a piece of string that is extracted from the malware and is unique. One major drawback of signature-based detection system is that it is very easy to alter the signature of a malware with minor modifications to escape detection [6]. Hence, commercial anti-virus products are moving towards the adoption of machine learning techniques to detect malware [7]. The features used to build the machine learning models can be obtained through static analysis, dynamic analysis or a combination of both [8]. In static analysis, the contents of a malware are analysed without executing it whereas in case of dynamic analysis, a malware is executed to examine its behaviour and its features are extracted. In hybrid analysis, features from static and dynamic analysis are combined. The features are then used with detection methods to classify malware. Our proposed system uses hybrid analysis with machine learning algorithm to design a

detection system for ABTs. We extract four feature sets - permissions, API calls, icon hiding and device administrator and use them with multiple machine learning algorithms to see their behaviour towards the detection of ABTs. Two features - permissions and API calls are extracted through static analysis whereas icon hiding and device administrator features are extracted through dynamic analysis.

The rest of this paper is organized as follows: In Section 2, we present the related works which is followed by methodology in Section 3. In Section 3, we introduce the proposed system and elaborate on techniques used for data collection and selection of features. The results of the research are presented in Section 4 and the analysis of the results is presented in section 5. Finally, the conclusion and future works are presented in Section 6.

2. RELATED WORK

The In this section, previous research works related to the static, dynamic and hybrid analysis of malware are presented. DroidAnalytics [6] is a signature based detection system that defines a three level signature generation which is robust against obfuscation. Other prominent work in the field of android malware detection include DroidAPIMiner [9], which uses API calls to predict android malware. Peiravian et al. [10] on the other hand, uses both permission and API call features to build detection models for android malware and shows that a combination of permissions and API calls is more effective than using either of the features separately. SigPID [11] uses a reduced permission set to detect android malware with significant accuracy reducing the analysis time. Both Drebin [12] and [13] use wide range of feature sets such as system commands, permissions and API calls extracted by static analysis for building prediction models. For prediction model based on dynamic analysis, [14] and [15] use system call as feature. In the work of [16], system call along with other behavioural features such as cryptographic operation, network operation and information leaks extracted using DroidBox tool are used for detecting android malware. Aside from system calls, Arora et al. [17] and Chen et al. [18] studied network traffic-related parameters to build the prediction model.

Much work in android malware detection is based on hybrid analysis that combines both static analysis and dynamic analysis for better detection accuracy. Raghuraman et al. [19] extract and use permissions and network protocols whereas Kabakus et al. [20] use permissions and network log of applications using various machine learning algorithms to detect android malware. Similarly, Surendran et al. [21] extract API calls, permissions and system calls of android applications and use Tree Augmented Naive Bayes (TAN) to identify android malware.

Little research has focused on specific category of malware like banking trojans. One such work [5] focuses on analysis of banking trojan BadAccents which infected thousands of Android users in the year 2014-2015. DroydSeuss [22] is a mobile banking trojan tracker that analyses an application both statically and dynamically to extract and examine strings related to communication endpoints. DBank [23] defines a novel concept called Triadic Suspicion Graph (TSG) based on android package information which is used to extract TSG features. The authors used TSG features in conjugation with other static and dynamic features like manifest, API package and API class to detect ABTs with high accuracy of and low false positive rate.

Most of the studies in android malware detection do not separate different categories of malware. Malware analysis targeting a specific type of malware is relatively less explored. An advantage of analysing only a specific category malware is that it can reveal previously unknown traits which may be significant in their detection. Our work addresses this gap in the detection of android malware as it deals exclusively with android banking trojans. A significant work [23] in this field uses novel TSG features which is different from the features we use in our work.

Our proposed framework uses four feature sets obtained through hybrid analysis to distinguish between malicious and non-malicious applications, specially focusing on ABTs. The four features include permissions and API calls as static features and icon hiding and device administrator as dynamic features. The combination of feature sets used in this work is unique because the dynamic features are less explored as compared to features like system calls and network traffic.

The studies related to our work in the past 5 years and their details have been summarized in Table 1.

Table 1. Summary of related works.

Author	**Year**	**Title**	**Significant Contribution**
Li et al. [11]	2018	Significant Permission Identification for Machine-Learning-Based Android Malware Detection	The authors achieve significant detection using reduced feature set while reducing the analysis time by 4 to 32 times.
Kabakus et al. [20]	2018	An In-Depth Analysis of Android Malware Using Hybrid Techniques	Analyze the static and dynamic characteristics of android malware and benign applications to see the difference in the type of permissions and network connection they request.
Feng et al.[16]	2018	A Novel Dynamic Android Malware Detection System With Ensemble Learning	Take application behavior into account along with system calls for designing the detection model and use ensemble learning algorithms for prediction models.
Chen et al. [18]	2018	Machine Learning Based Mobile Malware Detection Using Highly Imbalanced Network Traffic	Propose a prototype system to address the imbalance in benign and malicious network traffic data when used with classification algorithms.
Bai et al. [23]	2019	Dbank: Predictive Behavioral Analysis of Recent Android Banking Trojans	Introduce novel Triadic Suspicion Graph (TSG) features and show that the detection accuracy obtained by TSG features is comparable with traditional features like manifest and API package. TSG features when combined with traditional features can achieve high detection of android banking trojans.
Raghuraman et al. [19]	2020	Static and Dynamic Malware Analysis Using Machine Learning	Use Principal Component Analysis (PCA) to reduce the number of features and increase the accuracy of classifier model significantly.
Surendran et al. [21]	2020	A Tan Based Hybrid Model for Android Malware Detection	Propose a novel TAN (Tree Augmented naive Bayes) based detection mechanism which addresses the conditional dependencies between static and dynamic features to improve the accuracy of the classifier.

3. METHODOLOGY

This section presents the methodology used in this study. It elaborates on the various stages of the proposed system like data collection, feature extraction, feature selection and learning algorithms. The performance metrics used to evaluate the algorithms are also presented in this section.

3.1. Proposed System

Selected android applications (banking trojans and good-ware) are statically and dynamically analyzed to extract respective features as shown in Figure 1. Each application is in an *apk* format which is an archive file that consists of all the essentials of an android application. An *apk* contains of a single manifest file and one or more DEX files. All the permissions needed by an android application are listed in the Manifest file whereas the DEX files contain the source code of the application [24]. An *apk* file needs to be decompiled to extract the features from Manifest and DEX files. Once the features are extracted, Feature selection algorithms are used to create a reduced set of features that is used with various machine learning algorithms to build prediction models. A reduced set of features is preferable because using few relevant features instead of all available features has positive impact on the analysis time and accuracy of the prediction model [11] [19] [25].

3.2. Data Collection

The dataset is prepared using 2182 android applications out of which 1087 are Android Banking Trojans and 1095 are goodware (non-malicious applications). The banking trojan samples are collected from various malware repositories that provide malware samples for research such as VirusShare [26], Contagio [27], Virusign [28] and Koodous [29]. All these repositories are web-based and accessible through web browser. The malware samples belong to banking trojan familes Cerberus, Anubis, Faketoken, Spitmo, Svpeng, Wroba, Zitmo, Fakebank, Marcher, Asacub, Xbot and ZertSecurity. The goodware samples on the other hand, are collected from APKPure [30]. The good-ware samples consist of android applications belonging to business, communication, education, events, lifestyle, medical and productivity categories. All the malware samples are identified as malicious and the good-ware samples were identified as non-malicious by VirusTotal [31]. Virustotal is a online service that provides analysis on files and URLs using more than 70 commercial and open source antivirus software.

3.3. Feature Extraction

Feature extraction for static analysis is performed using Androguard tool. The extracted features - permissions and API calls are then stored as log files for further analysis. We extract a total of 1761 API calls and 1280 permissions (including custom permissions) from the applications initially. Likewise for dynamic analysis, we execute each application in a Nexus 5X API 23 Android Virtual Device [32] and observe the behaviour of the application. Monkeyrunner [33] simulates the user interaction with the installed applications. The log from the monkeyrunner is analysed to observe whether the application requests activation of device administrator feature or not. Likewise the icon hiding behaviour is examined before and after interacting with the application.

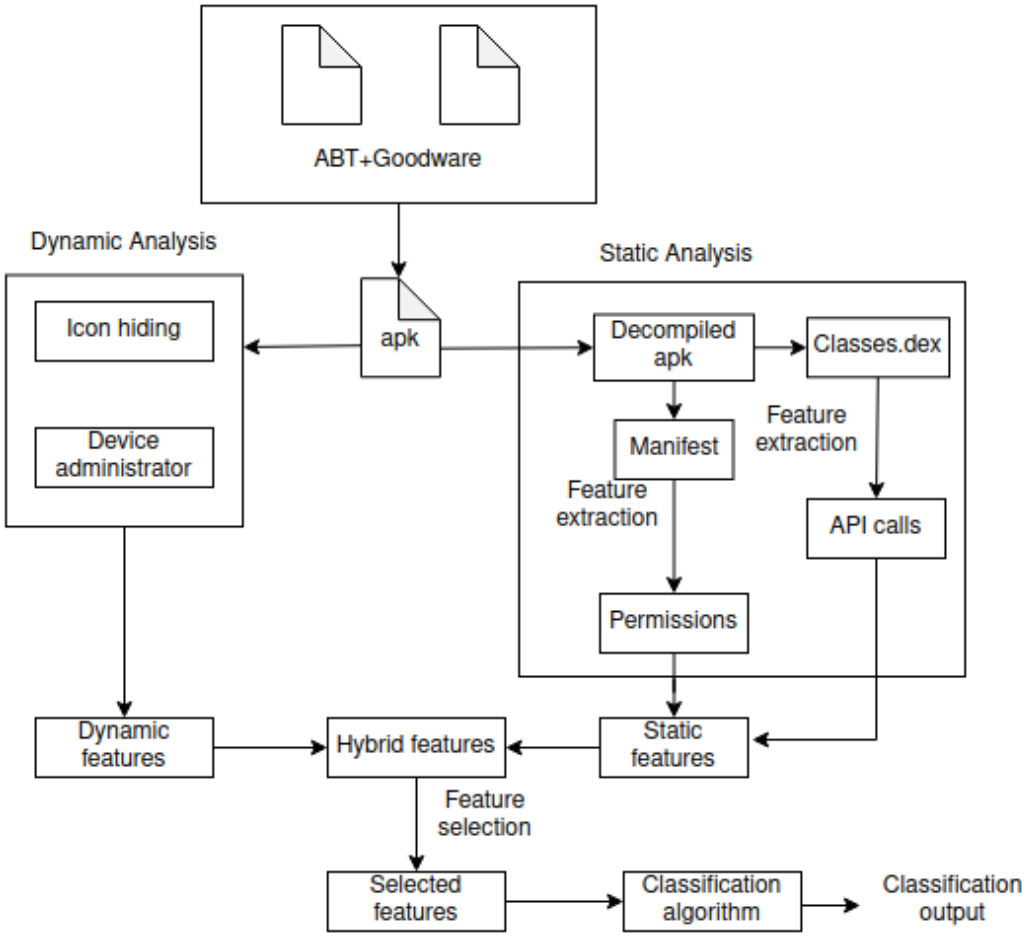

Figure 1. Proposed System

3.4. Feature Selection

Two feature selection algorithms are used to determine the distinguishing features for building the prediction model for banking trojans. The motivation behind using two feature selection algorithms is to see if different selection algorithms have significant differences on the performance of the prediction models. They feature selection algorithms used in this study are:

i. Frequency Based Selection
ii. Gain Ratio Based Selection

3.4.1. Frequency Based Selection

Frequency Based Selection is based on the previous studies by [9] and [10]. In frequency based selection, the features with highest frequency difference between good-ware and malware are considered and the features with low frequency difference are eliminated. Ten (10) permissions with highest frequency difference between banking trojans and good-ware are listed in Table 2.

Table 2. Top 10 permissions with highest frequency difference.

SN	Permission	Frequency Difference
1.	RECEIVE_SMS	1004
2.	READ_SMS	963
3.	SEND_SMS	945
4.	READ_CONTACTS	786
5.	READ_PHONE_STATE	768
6.	SYSTEM_ALERT_WINDOW	679
7.	WRITE_SMS	655
8.	CALL_PHONE	645
9.	RECEIVE_BOOT_COMPLETED	618
10.	BIND_GET_INSTALL_REFERRER_SERVICE	588

From Table 1, we can see the permissions with largest frequency difference between banking trojans and good-ware are related to SMS and phone calls. Android banking trojans generally request more permissions related to SMS and phone calls. Similarly, the ten API calls with highest frequency difference between banking trojans and good-ware are listed in Table 3.

Table 3. Top 10 API calls with highest frequency difference.

SN	API Calls	Frequency Difference
1.	android/content/Context->createPackageContext()	947
2.	android/os/Handler->removeMessages()	941
3.	java/net/HttpURLConnection->setInstanceFollow Redirects()	937
4.	android/content/pm/PackageManager->getPackage Installer()	932
5.	java/net/HttpURLConnection->getHeaderField()	931
6.	android/os/Handler->obtainMessage()	928
7.	android/content/Intent->getBooleanExtra()	925
8.	android/content/pm/PackageManager->hasSystem Feature()	924
9.	android/content/Intent->putExtra()	911
10.	android/os/Bundle->putDouble()	904

3.4.2. Gain Ratio Based Selection

Further In gain ratio based selection, the features are ranked based on their gain ratio and insignificant features with lower gain ratio are eliminated. We select thirty-eight (38) features with gain ratio greater than 0.5 to use because the threshold value of 0.5 provides the best detection rate. For our dataset, both increasing and decreasing the threshold value of gain ratio decreases the accuracy of the prediction models. The list of ten features with largest gain ratio is presented in Table 4.

Table 4. Features ranked by Gain Ratio Evaluator.

SN	Feature
1.	android.permission.RECEIVE_SMS
2.	android.permission.READ_SMS
3.	android.permission.SEND_SMS
4.	android/os/Handler->removeMessages()
5.	android/content/Context->createPackageContext()
6.	android/content/pm/PackageManager->getPackage Installer()

7.	android/os/Handler->obtainMessage()
8.	java/net/HttpURLConnection->setInstanceFollow Redirects()
9.	java/net/HttpURLConnection->getHeaderField()
10.	android/content/Intent->getBooleanExtra()

From Table 3, we can see that the features with largest gain ratio are permissions related to SMS and API calls related to 'android/os', 'android/content' and 'java/net' packages. Furthermore, no dynamic features are selected through this process because the gain ratios of both dynamic features are less than 0.5 as shown in Table 5.

Table 5. Gain Ratio of dynamic features.

SN	**Feature**	**Gain Ratio**
1.	Icon hiding	0.37857
2.	Device Administrator	0.16265

3.5. Dataset Preparation

Once the features are selected, we prepare the dataset for the prediction model. Each feature is a binary attribute and can have value of 0 or 1. The value 0 indicates the absence of a feature in an application and the value 1 indicates the presence of a feature. The class value to be predicted has two labels - trojan and good-ware. An instance in the dataset with 39 attributes is represented in the vector form as:

[1,1,1,0,0,1,0,1,0,1,0,0,0,1,1,0,0,0,0,0,0,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,0,trojan]

3.6. Classification

Once the The proposed system classifies android banking trojans and good-ware. The four classification algorithms used in this work are Support Vector Machine, Decision Tree, Random Forest and Neural Network. The classification algorithms have been selected based on their performance in previous related works [12] [23].

* Support Vector Machine (SVM):

In SVM, the different classification groups are divided by a hyper-plane such that the data points on each side are at a maximum distance from the hyper-plane [34].

* Decision Tree (DT):

Decision tree represents the decision process in the form of a tree. The root node is the feature which can optimally divide the data set and the leaf nodes are classification labels. The non-leaf nodes are composed of other features and the braches represent the value associated the features [34].

* Random Forest (RF):

Random Forest is a classification algorithm that combines multiple decision trees. For this purpose, the algorithm splits the training data into multiple data sets which are then used to train multiple decision trees [35].

* Neural Network (NN):

Neural Network is a classification algorithm that is based on human brain. Multiple layers of nodes are interconnected to mimic the behavior of neurons in human brain. The initial input is provided in the input layer and output is obtained from the output layer. Between these two layers, there can be multiple hidden layers where the actual processing of the input takes place [34].

The classification is performed using Waikato Environment for Knowledge Analysis (WEKA) tool [36]. The model is trained using 10 fold cross validation and evaluated using following metrics:

* Accuracy: Accuracy is the ratio of true instances to the total number of instances in a data set.

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

* Precision: Precision of a classifier is the ratio of true positive instances to the total number of positive instances.

$$Precision = (TP)/(TP+FP)$$

* Recall: Recall is the ratio of the true positive instances to the total number of actual positive instances.

$$Recall = (TP)/(TP+FN)$$

* F1 score: F1 score is the harmonic mean between precision and recall.

$$F1\ score = 2*(Precison*Recall)/(Precison+Recall)$$

* False Positive Rate (FPR): False Positive Ratio is the ratio of false positive instances to the total number of actual negative instances.

$$FPR = (FP)/(TN+FP)$$

The values of the performance metrics, precision, recall, f1-score and false positive rate used in this work are weighted average of the positive class (android banking trojan) and negative class (good-ware). Weka calculates the weighted metrics by calculating individual precision, recall, f1-score and false positive rate of each class and averaging the values with weight equal to the proportion of the data present in each class [37]. Weighted metrics are considered when both class values are relevant for classification. Hence, this research work uses weighted precision, recall, f1 score and false positive rate to compare the performance of the selected classifiers.

4. RESULTS

Prior to feature selection, we analyse the performance of various classification algorithms for all 3043 features (1761 API calls, 1280 permissions and 2 dynamic features). The accuracy, precision, recall, F1 score and false positive rate of the algorithms is presented in Table 6.

Table 6. Performance metrics of classifiers without feature selection.

Algorithm	Accuracy	Precision	Recall	F1 score	FPR
SVM	95.3%	95.3%	95.3%	95.3%	4.7%
Decision Tree	97.3%	97.3%	97.3%	97.3%	2.7%
Random Forest	97.4%	97.4%	97.4%	97.4%	2.6%

4.1. Frequency Based Selection

In Frequency based selection, we use the features with largest frequency difference as attributes for the classifier. The variation of accuracy in different classifiers with the frequency of permissions is presented in Figure 2. Similarly, the variation of accuracy in different classifiers with the frequency of API calls is presented in Figure 3.

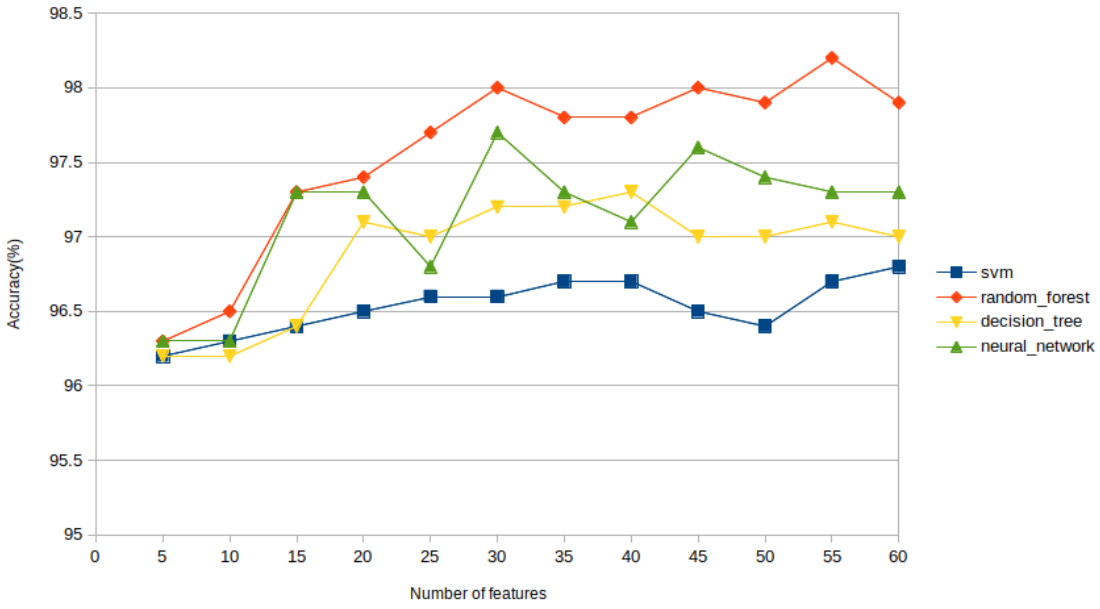

Figure 2. Accuracy of classifiers for permission related features

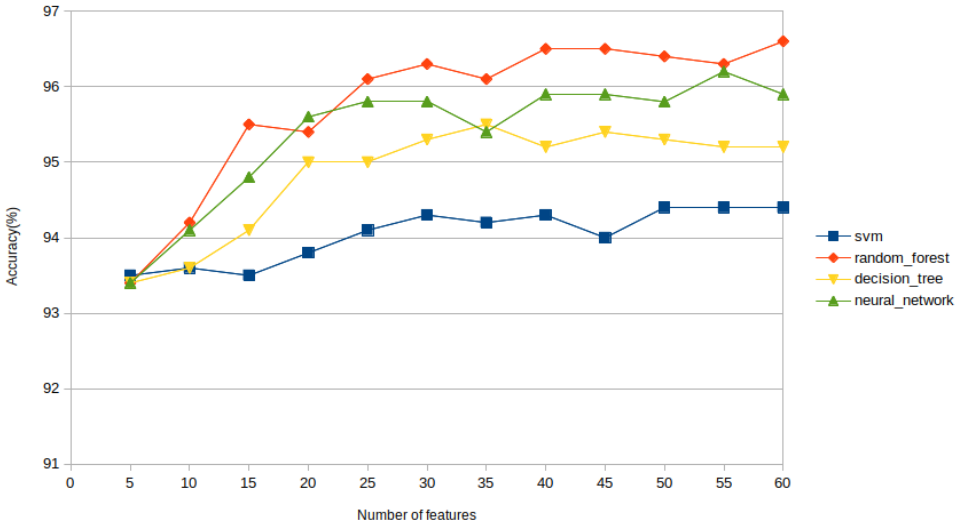

Figure 3. Accuracy of classifiers for API calls related features

From Figures 2 and 3, we can see that the accuracy of all the classifiers increase rapidly initially with increase in the number of features but the change in accuracy slows down as the number of features keep on increasing. Having a large number of features can lead to over-fitting which negatively impacts the performance of the prediction model.

The accuracy of the classifiers with respect to the combined features (permissions, API calls and dynamic features) is presented in Table 7. In the table, the notation `P' represents number of permissions, `A' represents number of API calls and `D' represents number of dynamic features.

Table 7. Accuracy of classifiers for combined features.

Number of features	**SVM**	**RF**	**DT**	**NN**
12(5P+5A+2D)	96.6%	96.9%	96.7%	97.1%
22(10P+10A+2D)	96.9%	97.3%	96.3%	97.4%
32(15P+15A+2D)	96.8%	97.6%	96.3%	97.6%
42(20P+20A+2D)	96.8%	97.7%	96.9%	97.7%
52(25P+25A+2D)	97.1%	97.8%	96.8%	97.3%
62(30P+30A+2D)	97.2%	**98.1%**	96.7%	97.5%
72(35P+35A+2D)	97.1%	97.9%	96.7%	97.5%
82(40P+40A+2D)	97.1%	97.9%	96.7%	97.3%

For combined features, the best accuracy is obtained with Random Forest with 62 features - 30 permissions, 30 API calls and 2 dynamic features.

4.2. Gain Ratio-Based Selection

Gain Ratio Based Selection uses gain ratio to determine the features with are most relevant to the prediction model. Gain Ratio of an attribute (feature) is defined as the ratio of Information gain of the attribute to its entropy. Thirty-eight (38) features with gain ratio greater than 0.5 are used with various classifiers and their performance is analysed. The performance of the classifiers in terms of accuracy, precision, recall, F1 score and False Positive Rate (FPR) is presented in Table 8.

Table 8. Performance metrics of classifiers with gain ratio based selection.

Algorithm	Accuracy	Precision	Recall	F1 score	FPR
SVM	96.3%	96.5%	96.3%	96.3%	3.7%
Decision Tree	96.3%	96.4%	96.3%	96.3%	3.7%
Random Forest	**97.9%**	97.9%	97.9%	97.9%	2.1%
Neural Network	97.1%	97.1%	97.1%	97.1%	2.9%

For gain ratio based selection, random forest classifier has the best detection rate with an accuracy of 97.9%.

5. ANALYSIS OF RESULT

Among all the learning algorithms, Random Forest shows the best performance for both feature selection methods with highest accuracy around 98%. The slightly better performance of Random Forest can be attributed to the fact that it is an ensemble learning based classifier and combines output from multiple decision trees. Apart from Random Forest, other three learning algorithms also achieve detection accuracy above 95%.

Comparing the performance of the classifiers before and after the feature selection (Table 5 and Table 7), it can be seen that the accuracy of Random Forest and Support Vector Machine classifiers increase from 97.4% to 97.9% and from 95.3% to 96.3% respectively. By using feature selection, we were able to eliminate less relevant features and decrease the number of features from 3043 to 38 (gain ratio) while maintaining some improvement in the accuracy of the prediction models. Reduction in the number of features means a better runtime performance for the models which can be significant when huge amount of data is used [11].

6. CONCLUSION AND FUTURE WORK

With the rising number of mobile banking application users, banking trojans targeting such mobile platforms are also increasing. The proposed system combines hybrid analysis and machine learning techniques to build a prediction model for android banking trojans. A combination of permissions, API calls, icon hiding and device administrator features is used with four different machine learning algorithms- Random Forest, Decision Tree, SVM and Neural Network to build the classifiers. Feature selection techniques are used to rank and select the most significant features. All the classifiers achieve good accuracy with Random Forest performing the best with an accuracy up to 98%.

As for future work, we will use ensemble machine learning techniques with the proposed system and study their performance in detecting ABTs.

REFERENCES

[1] Wallen, J. (2021). *Why is android more popular globally, while ios rules the us?*. https://www.techrepublic.com/article/why-is-android-more-popular-globally-while-ios-rules-the-us/

[2] Proske, S. (2017). *Another reason 99% of mobile malware targets androids*. https://blog.f-secure.com/another-reason-99-percent-of-mobile-malware-targets-androids/

[3] Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—a state of the art survey. *ACM Computing Surveys (CSUR)*, 52 (5), 1–48.

[4] Chebyshev, V. (2020). *Mobile malware evolution 2019*. https://securelist.com/mobile-malware-evolution-2019/96280/

[5] Rasthofer, S., Asrar, I., Huber, S., & Bodden, E. (2015). An investigation of the android/badaccents malware which exploits a new android tapjacking attack. *Technical report, Technische Universitt Darmstadt*.
[6] Zheng, M., Sun, M., & Lui, J. C. (2013). Droid analytics: A signature based analytic system to collect, extract, analyze and associate android malware. *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, 163–171.
[7] Marques, P., Rhode, M., & Gashi, I. (2021). Waste not: Using diverse neural networks from hyperparameter search for improved malware detection. *Computers & Security*, 108, 102339.
[8] Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The evolution of android malware and android analysis techniques. *ACM ComputingSurveys (CSUR)*, 49 (4), 1–41.
[9] Aafer, Y., Du, W., & Yin, H. (2013). Droidapiminer: Mining api-level features for robust malware detection in android. *International conference on security and privacy in communication systems*, 86–103.
[10] Peiravian, N., & Zhu, X. (2013). Machine learning for android malware detection using permission and api calls. *2013 IEEE 25th international conference on tools with artificial intelligence*, 300–305.
[11] Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, 14 (7), 3216–3225.
[12] Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. *Ndss*, 14, 23–26.
[13] Fereidooni, H., Conti, M., Yao, D., & Sperduti, A. (2016). Anastasia: Android malware detection using static analysis of applications. *2016 8th IFIP international conference on new technologies, mobility and security (NTMS)*, 1–5.
[14] Canfora, G., Medvet, E., Mercaldo, F., & Visaggio, C. A. (2015). Detecting android malware using sequences of system calls. *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, 13–20.
[15] Wang, C., Li, Z., Mo, X., Yang, H., & Zhao, Y. (2017). An android malware dynamic detection method based on service call co-occurrence matrices. *Annals of Telecommunications*, 72 (9-10), 607–615.
[16] Feng, P., Ma, J., Sun, C., Xu, X., & Ma, Y. (2018). A novel dynamic android malware detection system with ensemble learning. *IEEE Access*, 6, 30996–31011.
[17] Arora, A., Garg, S., & Peddoju, S. K. (2014). Malware detection using network traffic analysis in android based mobile devices. *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, 66–71.
[18] Chen, Z., Yan, Q., Han, H., Wang, S., Peng, L., Wang, L., & Yang, B. (2018). Machine learning based mobile malware detection using highly imbalanced network traffic. *Information Sciences*, 433, 346–364
[19] Raghuraman, C., Suresh, S., Shivshankar, S., & Chapaneri, R. (2020). Static and dynamic malware analysis using machine learning. *First International Conference on Sustainable Technologies for Computational Intelligence*, 793–806.
[20] Kabakus, A. T., & Dogru, I. A. (2018). An in-depth analysis of android malware using hybrid techniques. *Digital Investigation*, 24, 25–33.
[21] Surendran, R., Thomas, T., & Emmanuel, S. (2020). A tan based hybrid model for android malware detection. *Journal of Information Security and Applications*, 54, 102483.
[22] Coletta, A., Van Der Veen, V., & Maggi, F. (2016). Droydseuss: A mobile banking trojan tracker (short paper). *International Conference on Financial Cryptography and Data Security*, 250–259.
[23] Bai, C., Han, Q., Mezzour, G., Pierazzi, F., & Subrahmanian, V. (2019). Dbank: Predictive behavioral analysis of recent android banking trojans. *IEEE Transactions on Dependable and Secure Computing*.
[24] Google Developers. (n.d.). *Application fundamentals*. https://developer.android.com/guide/components/fundamentals
[25] Arshad, S., Shah, M. A., Wahid, A., Mehmood, A., Song, H., & Yu, H. (2018). Samadroid: A novel 3-level hybrid malware detection model for android operating system. *IEEE Access*, 6, 4321–4339.
[26] Virusshare. (n.d.). *Virusshare*. https://virusshare.com/
[27] Parkour, M. (n.d.). *Contagio mobile*. http://contagiominidump.blogspot.com/
[28] Virusign. (n.d.). *Virusign*. https://www.virusign.com/
[29] Koodous. (n.d.). *Koodous*. https://koodous.com/

[30] Apkpure. (n.d.). *Apkpure*. https://apkpure.com/
[31] Talukder, S., & Talukder, Z. (2020). A survey on malware detection and analysis tools. *International Journal of Network Security & Its Applications (IJNSA)*, 12.
[32] Grgurina, R., Brestovac, G., & Grbac, T. G. (2011). Development environment for android application development: An experience report. *2011 Proceedings of the 34th International Convention MIPRO*, 1693–1698.
[33] Google Developers. (2015). *Monkeyrunner*. https://developer.android.com/studio/test /monkeyrunner.
[34] Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review*, 26 (3), 159–190.
[35] Liaw, A., & Wiener, M. (2002). Classification and regression by random forest. *R news*, 2 (3), 18–22.
[36] Garner, S. R. (1996). Weka: The waikato environment for knowledge analysis.
[37] Eibe, F. (2019). *Micro averages in multi-class classification*. https://waikato.github.io/ weka-blog/posts/2019-02-16-micro average.

AUTHORS

Subarna Adhikari, Recent Graduate of Master's in Computer Engineering, Department of Computer Science and Engineering Kathmandu University, Dhulikhel, Nepal

Sushil Nepal, Assistant Professor
Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Nepal

Rabindra Bista*, Associate Professor
Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Nepal

* - Corresponding Author