Evaluation of Ensemble Machine Learning Methods in Mobile Threat Detection

Sanjay Kumar*, Ari Viinikainen† and Timo Hamalainen‡
Faculty of Information Technology
University of Jyvaskyla
Jyvaskyla, Finland
Email: *sanjay.k.kumar@jyu.fi, †ari.viinikainen@jyu.fi, ‡timo.t.hamalainen@jyu.fi

***Abstract*—The rapid growing trend of mobile devices continues to soar causing massive increase in cyber security threats. Most pervasive threats include ransom-ware, banking malware, premium SMS fraud. The solitary hackers use tailored techniques to avoid detection by the traditional antivirus. The emerging need is to detect these threats by any flow-based network solution. Therefore, we propose and evaluate a network based model which uses ensemble Machine Learning (ML) methods in order to identify the mobile threats, by analyzing the network flows of the malware communication. The ensemble ML methods not only protect over-fitting of the model but also cope with the issues related to the changing behavior of the attackers. The focus of this study is on android based mobile malwares due to its popularity among users. We have used ensemble methods to combine output of 5 supervised ML algorithms such as RF, PART, JRIP, J.48 and Ridor. Based on the evaluation results, the proposed model was found efficient at detecting known and unknown threats with the accuracy of 98.2%.**

***Index Terms*—Intrusion Detection, Machine Learning, Ensemble Methods, Supervised Machine Learning, Mobile Threats, Anomaly Detection**

I. Introduction

According to research by Sophos [1], by 2020 more than 6 billion users will be using mobile devices. Mobile devices are rapidly overtaking personal computers from web surfing to mobile banking, due to their portability and smart features. Therefore, the potential usage has caught the attention of cybercriminals who maximize their efforts to obtain user information. Most of the users do not care about the security measure of their devices and thus become the victim of these threats. These applications could lead to several mobile threats, such as theft of financial information, ransom-ware, misuse of premium SMS and theft of personal information. A traditional anti-virus can detect only 50% of the threats and on the other hand around 71% of the smart-phone users do not use any kind of anti-virus [2]. Thus there is a need for an extra layer of security at the network side to protect the users from advanced threats, which a traditional anti-virus could not detect.

Most of the NIDS use signatures to detect attacks and therefore capable of detecting only known attacks [3]. A minor modification in attack can bypass a signature based NIDS and could generate up to 90% of false alarms [4]. Deep packet inspection is also difficult when the traffic is encrypted and computationally expensive [5]. Flow based techniques are useful in combating several issues caused by encrypted traffic [6]. Machine learning methods are getting popular in the detection of advanced threats. This transition is supported by Arp et al [7].

To build a ML classifier, a dataset is required. The well-known datasets available in the field of intrusion detection, which uses network traffic features are KDD99, DARPA 1998/1999 and ISCX 2012 IDS dataset. However, some short-comings were observed in these datasets by [8] and [9]. These datasets are quite old and not applicable to mobile attacks. Due to non-availability of public datasets in this domain, we have used a dataset which was created in our previous research [10] for evaluation purpose. This dataset was build using the traffic generated by several benign and malicious samples. The dataset is based on bidirectional flows extracted from real malware traffic, which makes it unique. The dataset contains several threats such as unauthorized premium SMS sender, Spam sender, bots, back-door, root exploit, fake anti-virus, ransom-ware and information theft. These datasets were used to train and build the ML classifiers using several ML algorithms. The classifier is used to make predictions on new data to detect normal or malicious patterns in the traffic. A signature-based NIDS could miss the threats in the traffic, for which the signature is not yet known. However, ML classifiers can detect known and unknown threats by analyzing traffic patterns.

The main focus of this paper is the performance evaluation of ensemble ML techniques that combines output of several ML algorithms. The benefit of using ensemble method is not only to increase the efficiency of the classifier but also to reduce the risk of over-fitting the model. This can also address the problem caused by minor changes in the attack pattern, in order to avoid concept drift situation. The concept drift situation occurs in machine learning methods when the relation between the features used to train the model and target to be predicted changes over an interval of time. The concept drift causes decrease in accuracy when prediction is made on unseen data.

The structure of this paper is as follows. Section 2 focuses on previous research conducted in this area. Section 3 describes the machine learning algorithms used in this study. Section 4 focuses methodology of this research. Section 5 is based on the performance evaluation of ensemble machine learning classifiers using different datasets. Several experiments that are

978-1-908320/93/3/$31.00©2017 IEEE

performed for evaluation purposes are also explained. Finally, in Section 6, conclusion and future work of this research are outlined.

II. Related Work

Recently, a lot of research has been done in the area of machine learning to solve many cybersecurity issues. Most of the research done in the field of using ML techniques to detect Android-based malware is based on features such as system or API calls. There are only a few studies which have focused on network-based intrusion detection for android malwares [10] [11]. In some of the studies [12] [7] [13] [14], the detection engine or model need to be installed on the mobile phone to detect e.g. intrusions or malicious applications. However, most of the smartphone users do not install security solutions in their phones.

Arp et al. [7] developed Drebin which uses Support Vector Machine (SVM) to detect malicious android applications. Drebin is based on the features such as permissions, API calls and Network addresses. The detection of Drebin is limited when the malwares uses dynamic code or any obfuscation technique. Many researchers [12] [13] used malwares from MalGenome [15] dataset to generate traffic and build classifiers on various traffic based features. The concept drift has been seen in some studies [12] where the classifiers produced significant decrease in the TPR when evaluated on the unseen traffic. Input features used to train the model play important role in the field of machine learning as the attackers change their behavior with time to avoid detection. Features like IP address could lead to produce concept drift in the model. In some of the studies [13] [11], the classifiers were not tested on unseen data which is crucial part in the evaluation of ML classifiers.

Many researchers [16] [17] [6] used flow-based models for network traffic classification. Furthermore, the flow-based techniques to detect botnets were studied by several researchers [18] [19] [20]. Most of network traffic in the malware communication is encrypted [5] and therefore flow-based features seems to be efficient in detecting these threats [6].

III. Machine Learning Algorithms

In this research work, we have used the ensemble of several machine learning algorithms such as Random Forest, J48, RIDOR, JRIP and PART. The performance evaluation of these algorithms was already performed individually in our previous work [10]. In this study, we have used ensemble methods to combine the output of these ML algorithms to increase the effectiveness of the ML classifiers.The combination methods used in our study such as majority voting, maximum probability and product of probabilities were adopted from [21] [22]. ML classifiers used in the network traffic analysis becomes less efficient with time as the attackers change their behavior and this situation is known as concept drift. During the model building, each algorithm has different weight-age for each feature. Some of the algorithms have built-in feature selection algorithms and they make the decisions on limited features. If there is any change in traffic pattern, each algorithm behaves differently. Combining the output of these individual algorithms not only increases the efficiency but also the stability in case of minor changes in the traffic pattern.

J48 is the WEKA Implementation of C4.5 [23] algorithm which was developed by Ross Quinlan in 1993. C4.5 is a decision tree based algorithm, which works on the ”divide and conquer” rule. C4.5 first divides the training dataset with highest single class instances, then it checks the feature with the highest information gain in the subset and splits it into further subsets according to that feature. It repeats these steps for each subset [23].

The Random Forest(RF) [24] is one of the most popular ML algorithms used for classification, developed in 2011 by Leo Breimen. Random forest is the collection of decision trees built from random subsets of dataset (bootstraps) with random features selected in each subset. Each tree is trained by 2/3 of the dataset and remaining 1/3 is used to estimate error rate. This 1/3 of the dataset is same as the validation set in other ML algorithms, therefore there is no need for a separate validation dataset. The output of the random forest is based on the majority vote by each decision tree output.

Ridor is the WEKA implementation of Ripple-Down Rule Learner [25], which was developed by Gaines and Compton in 1995. Ridor uses Incremental reduced error pruning (IREP) algorithm [26] to build its rules. Pseudo code for IREP is mentioned in Figure 1 of [27]. Ridor generates its first rule as a default rule for one class and then builds the rules for other classes depending on the weighted error rate known as exception rules. Let’s suppose there are two classes ”Deny” and ”Allow”. First, it makes a default rule for ”Deny” and then it builds up the rules for ”Allow” [25].

JRIP is a WEKA implementation of RIPPER, which was proposed by William Cohen in 1995, as an optimized version of IREP [26], [27]. JRIP divides the training set into two subsets in the ratio of 2:1 in the form of grow:prune.

PART is a partial decision tree algorithm developed by Frank [28] in 1998. This algorithm works on the separate-and-conquer rule and is a combination of C4.5 rules and RIPPER algorithm, excluding the global optimization feature. This algorithm produces rules in ordered sets, which makes a decision list. The rules are based on ”Best” leaf of the partial C4.5 decision tree [28].

J48 and Random Forest are the tree based algorithms while RIDOR, JRIP and PART are rule based algorithms. All of these ML algorithms have some internal validation function for tuning to avoid over-fitting e.g Random forest [24] uses 1/3 of the dataset for estimating the error rate. JRIP [27] and RIDOR [25] both use IREP which selects 2/3 of the dataset for training and 1/3 of the dataset for the pruning of the model. J48 [23] has an internal mechanism of pre-pruning and post-pruning to avoid over-fitting and PART [28] is the combination of J48 and JRIP.

Random Forest has several advantages, such as high accuracy and effectiveness on large datasets [29]. Random Forest

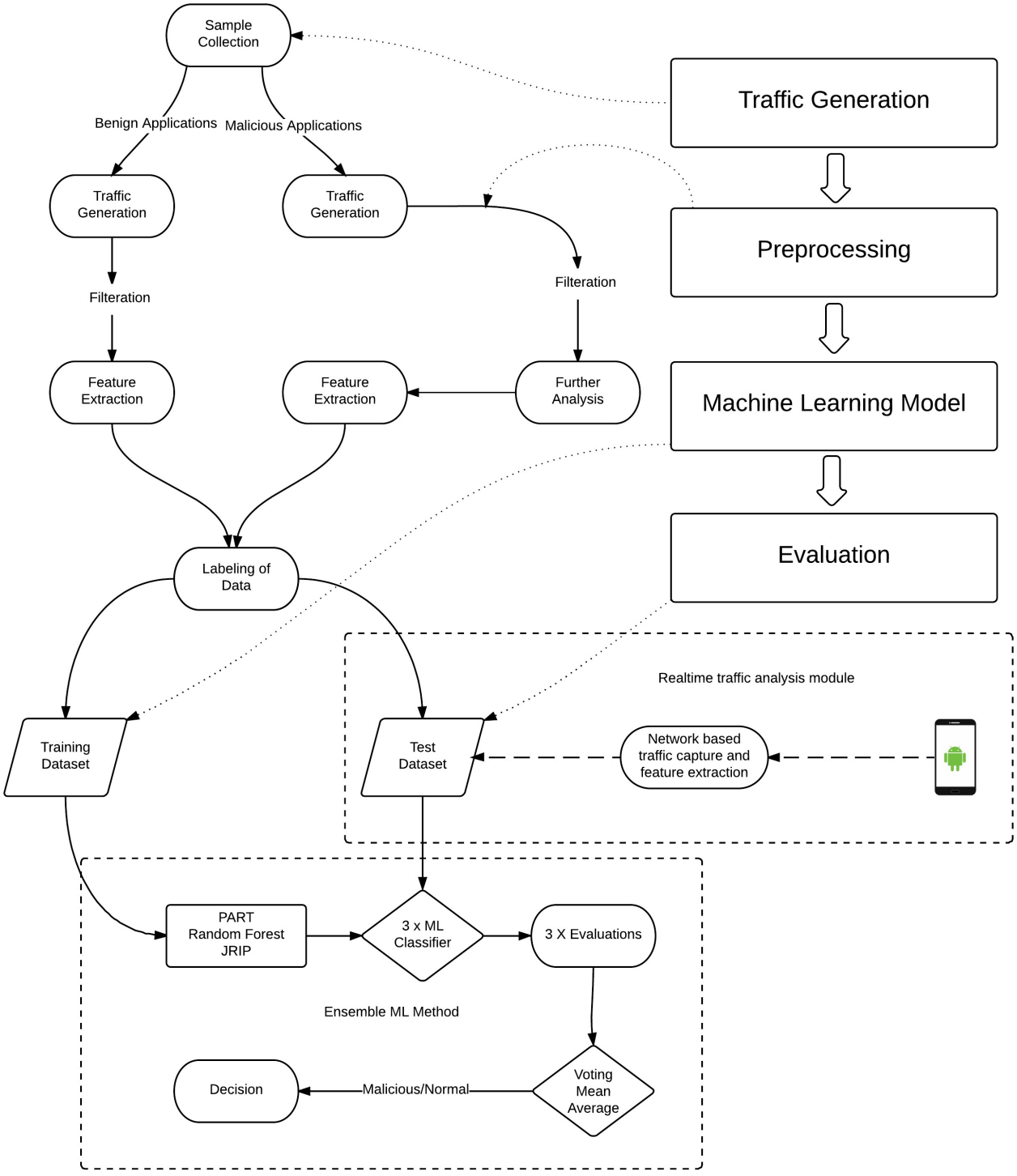

Fig. 1: Ensemble Machine Learning model for Intrusion Detection

is an ensemble of multiple decision trees. Although the output of Random Forest is hard to understand, the performance of this classifier makes it outstanding. J48 provides speed over Random Forest, but the accuracy is not as high as that of Random Forest. J48 is a decision tree so it is easy to understand. RIDOR, PART and JRIP are rule-based algorithms, so the rules generated by these algorithms can be used in any knowledge-based expert system.

IV. Methodology and Implementation

In this research, an ensemble ML based Network intrusion detection system is proposed and evaluated, shown in Fig. 1. The first step was traffic generation, followed by filtration, feature extraction and labeling of the dataset. The dataset was used to build the ML Classifiers using WEKA [30]. In this study, the focus is only on the evaluation of ensemble ML classification model, as the evaluation of individual classifiers is already done in our previous work [10].

The overall implementation and evaluation of this model performed in four main phases, as shown in Fig. 1, comprised of Traffic generation, preprocessing, model building and evaluation of the ML model.

TABLE I: Feature List

Feature No.	Feature	Description	Value
1	Duration	Connection Duration	Real
2	DP	Destination Port	Real
3	PktSent	Packet Sent	Real
4	PktRcv	Packets Received	Real
5	PLBytesSent	Payload bytes sent	Real
6	PLBytesRcv	Payload bytes received	Real
7	IFlagF	Initial Flags in Forward Direction	Nominal
8	IFlagR	Initial Flags in Reverse Direction	Nominal
9	UFlagsF	Union of Flags in Forward Direction	Nominal
10	UFlagR	Union of Flags in Reverse Direction	Nominal

Traffic was generated for both benign and malicious applications using the method mentioned in our previous research [10]. A number of benign applications were used to generate real traffic which were installed from Google playstore. These applications were executed at a different interval of time. Wireshark was used to capture the packets on the interface of the virtual machines. The samples of malware families (FakeAV, DroidKungFu, OPFake, GinMaster, FakeInst and Anserver) were downloaded from Virustotal using several conditions. The number of samples completed for the study was around 600. Traffic was generated through a public sandbox "Anubis (Andrubis)" [31] and "Cuckoo" [32].

During Processing feature extraction and labeling of traffic flows was done. The features were extracted using RFC-5103 BiFlow export method [33]. The following features (see Table I) were extracted from the flows of the traffic of benign and malicious applications. Instances were then labeled as normal or malicious respectively.

A. Machine learning classifiers

In this study, we have evaluated the combination of 5 decision tree and rule based algorithms. Output of these algorithms can be easily interpreted by security experts and can be integrated with the traditional NIDS. The classifiers build from these ML algorithms produce rules and trees which can be used to make predictions on new traffic to identify threats. By using ensemble methods, the combination of these ML classifiers was used to increase the efficiency of the classification model as shown in Fig. 1.

V. Performance Evaluation and Results

Several well-known parameters were used to evaluate ensemble ML classifiers.

TABLE II: Confusion Matrix

		Predicted	
		Malicious	Normal
Actual	Malicious	$TruePositive$	$FalseNegative$
	Normal	$FalsePositive$	$TrueNegative$

True Positive (TP): Malicious instance classified as Malicious.
False Positive (FP): Benign instance classified as Malicious
False Negative (FN): Malicious instance classified as Normal.
True Negative (TN): Benign Instance classified as Normal.

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

$$TNR = \frac{TN}{TN+FP}$$

$$FNR = \frac{FN}{TP+FN}$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

ROC (Receiver Operating Characteristic) curve is a plot between TPR and FPR at various threshold settings [34]. Area under ROC Curve (AUC) is also an important parameter in evaluating the ML classifier, this value is derived from the ROC curve and it can tell which model makes best predictions. A higher AUC value shows a better ML Classifier. Accuracy is also an important parameter to consider as it is based on both TPR and FPR.

A. Evaluation of Classification Model

We have performed two experiments in order to evaluate the performance of the ML classifiers using ensemble methods. In the first experiment, we have tested the classifiers using cross validation and percentage split of the same data. In the second experiment the evaluation was performed using the new unseen dataset.

1) Experiment 1 - Ensemble Methods : Ensemble methods combine output of several ML classifiers by different techniques such as weighted voting or measuring probability as shown in Fig 1. In this experiment, we combined the output of several classifiers by 3 combination rules as shown in the Table III - IV.

These tables show the detailed performance evaluation of ensemble methods using different combination. In Experiment 1a, we have used 10 fold cross validation method which is most widely used validation method. This method splits

TABLE III: Experiment 1a (Ensemble Methods) using Cross Validation

Performance Evaluation by combining J48, RF, JRIP, RIDOR and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.995	0.030	0.970	0.005	0.991	0.984
Maximum Probability	0.983	0.005	0.995	0.017	0.985	0.999
Product of Probabilities	0.999	0.005	0.995	0.001	0.998	0.999
Performance Evaluation by combining J48, Random Forest, JRIP and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.996	0.041	0.959	0.004	0.990	0.978
Maximum Probability	0.993	0.030	0.970	0.007	0.989	0.999
Product of Probabilities	0.994	0.025	0.975	0.006	0.991	0.987
Performance Evaluation by combining J48, RF and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.996	0.023	0.977	0.004	0.993	0.986
Maximum Probability	0.993	0.043	0.957	0.007	0.987	0.999
Product of Probabilities	0.994	0.039	0.961	0.006	0.989	0.993

TABLE IV: Experiment 1b (Ensemble Methods) using Percentage Split Validation

Performance Evaluation by combining J48, RF, JRIP, RIDOR and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.995	0.030	0.970	0.005	0.991	0.982
Maximum Probability	0.983	0.005	0.995	0.017	0.985	0.999
Product of Probabilities	0.999	0.005	0.995	0.001	0.998	0.981
Performance Evaluation by combining J48, Random Forest, JRIP and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.996	0.041	0.959	0.004	0.990	0.982
Maximum Probability	0.993	0.030	0.970	0.007	0.989	0.999
Product of Probabilities	0.994	0.025	0.975	0.006	0.991	0.978
Performance Evaluation by combining J48, Random Forest and PART						
Combination Rule	**TPR**	**FPR**	**TNR**	**FNR**	**Accuracy**	**AUC**
Majority Voting	0.996	0.023	0.977	0.004	0.993	0.982
Maximum Probability	0.993	0.043	0.957	0.007	0.987	0.999
Product of Probabilities	0.994	0.039	0.961	0.006	0.989	0.981

the original dataset into ten pieces and repeats the training and testing for ten times using holdout technique. Cross validation method reduce the chances of over-fitting the model. In experiment 2a, percentage split was used to divide the dataset into two pieces and 70% of the data was used for training while the remaining 30% was used for testing. This method is very useful when the model needs to be used for predictions. The best results were obtained by the majority voting of output by Random Forest, PART and J48 classifiers. The TPR of 99.5% was observed using ensemble methods

TABLE V: Performance Evaluation of Experiment 2a - Detecting Unknowns

Evaluation on ML classifiers on unknown dataset						
ML Algorithm	TPR	FPR	TNR	FNR	Accuracy	AUC
Random Forest	1.000	0.029	0.971	0.000	0.975	0.998
PART	1.000	0.117	0.883	0.000	0.900	0.917
JRIP	1.000	0.043	0.957	0.000	0.964	0.979
Ensemble Methods	1.000	0.021	0.979	0.000	0.982	0.989

TABLE VI: Performance Evaluation of Experiment 2b - Detecting Unknowns

Evaluation on ML classifiers on unknown dataset after interval of time						
ML Algorithm	TPR	FPR	TNR	FNR	Accuracy	AUC
Random Forest	0.932	0.038	0.962	0.068	0.955	0.947
PART	0.926	0.112	0.888	0.074	0.897	0.893
JRIP	0.919	0.047	0.953	0.081	0.945	0.937
Ensemble Method	0.939	0.025	0.975	0.061	0.966	0.957

which was better than that of Random Forest. However, the FPR of 4.1% observed which is higher than Random Forest. It can also be seen that the best accuracy is seen by combining output of the five classifiers using the product of probabilities.

2) Experiment 2 - Detecting Unknown: This experiment was performed to evaluate the performance of ensemble ML classifiers on a new dataset that contains unknown instances from malicious samples and contains new traffic from different benign applications. We have limit this experiment to 3 ML Algorithms (Random Forest, PART and JRIP) as these ML algorithms produced the best results in individual evaluation. We have combined the output of these ML algorithms by majority vote method as shown in the Table V.

In Table V, performance evaluation of different classifiers can be seen. The RF performed best in individual classifiers with the highest TPR and lower FPR. The FPR produced by PART was significantly high. However, the ensemble classifier outperformed all the individual classifiers. In Table V, it can be seen that ensemble method performed better than RF in detecting unknown threats with the accuracy of 98.2% and the FPR was reduced to 2.1%.

This experiment showed that the ensemble methods are not only able to detect unknown threats but they are also good at identifying benign traffic. True Negative Rate (TNR) produced by these ensemble classifiers is also high which shows the efficiency of the classifier in distinguishing between normal and malicious instances.

Another experiment 2b (see Table V) was performed to check the performance of the classifiers after an interval of time. For that purpose, unseen traffic from some new malicious and benign applications was added to the test dataset. The performance of the individual classifiers decreased a bit with time due to the changes in the traffic patterns generated by the new malware samples. The ensemble methods increased the performance by combining the output of these individual classifiers. It can be clearly seen that the ensemble methods produced the highest accuracy and AUC value by combining the output of these classifiers.

VI. Conclusion

The ensemble methods used in this study were able to detect known and unknown threats. This study is the first step towards a more advanced ML based intrusion detection system. Ensemble methods not only produce better results but also reduce the chance of concept drift. Intrusion detection systems which rely only on ML techniques need frequent retraining. Otherwise, the decrease in TPR could be seen. In our previous studies, several experiments were performed to compare the ML model with antivirus vendors and we observed that ML classifiers were more efficient than some of the traditional antivirus. Furthermore, the efficiency of the ML classifiers was enhanced by using ensemble methods and these methods also helped with concept drift. Moreover, we have observed that the feature extraction and selection play an important role in the output of the classifier. Wrong features such as "IP Address" could over-fit the model and produce concept drift condition in the system.

The ensemble ML classifiers built were able to detect malicious traffic with a TPR of 99.9%, while the output from individual classifiers was observed between 94%-99.6%. As the ML classifiers are built for predictions, it is also important to evaluate the performance of the ML classifiers on new data. In this research, we have evaluated ML classifiers on unseen data and the accuracy of 98.2% was observed by ensemble methods while the accuracy observed by individual classifiers was between 90% - 97.5%. These results showed that the ensemble methods are more efficient than individual classifiers. Future work in progress aims to integrate the ML classifiers with traditional NIDS and to introduce some innovative methods in order to reduce the chance of concept drift.

REFERENCES

[1] SOPHOS, "When malware goes mobile," Tech. Rep., Access Date 26 Sep, 2017. [Online]. Available: https://www.sophos.com/en-us/security-news-trends/security-trends/malware-goes-mobile.aspx

[2] "Malware detection and subscriber protection infographic," Alcatel-Lucent, Tech. Rep., Access Date 10 Sep, 2017. [Online]. Available: https://www.alcatel-lucent.com/solutions/security-guardian-infographic

[3] K. Timm, "Strategies to reduce false positives and false negatives in nids," Tech. Rep., Access Date 10 Sep, 2017. [Online]. Available: http://www.symantec.com/connect/articles/strategies-reduce-false-positives-and-false-negatives-nids

[4] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 02*. Association for Computing Machinery (ACM), 2002.

[5] R. Koch, "Towards next-generation intrusion detection," in *Cyber Conflict (ICCC), 2011 3rd International Conference on*. IEEE, 2011, pp. 1–18.

[6] J. A. Copeland III, "Flow-based detection of network intrusions," Feb. 27 2007, uS Patent 7,185,368.

[7] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket," in *Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS)*, 2014.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.

[9] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM transactions on Information and system Security*, vol. 3, no. 4, pp. 262–294, 2000.

[10] S. Kumar, A. Viinikainen, and T. Hamalainen, "Machine learning classification model for network based intrusion detection system," in *Proc. 11th Int. Conf. for Internet Technology and Secured Transactions (ICITST)*, Dec. 2016, pp. 242–249.

[11] S. Wang, Z. Chen, L. Zhang, Q. Yan, B. Yang, L. Peng, and Z. Jia, "Trafficav: An effective and explainable detection of mobile malware behavior using network traffic," in *Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on*. IEEE, 2016, pp. 1–6.

[12] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Comput*, nov 2014.

[13] A. Feizollah, N. B. Anuar, R. Salleh, F. Amalina, R. R. Maarof, and S. Shamshirband, "A study of machine learning classifiers for anomaly-based mobile botnet detection," *Malaysian Journal of Computer Science*, vol. 26, no. 4, 2014.

[14] X. Su, M. C. Chuah, and G. Tan, "Smartphone dual defense protection framework: Detecting malicious applications in android markets," in *Mobile Ad-hoc and Sensor Networks (MSN), 2012 Eighth International Conference on*. IEEE, 2012, pp. 153–160.

[15] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 95–109.

[16] M. Soysal and E. G. Schmidt, "Machine learning algorithms for accurate flow-based network traffic classification: Evaluation and comparison," *Performance Evaluation*, vol. 67, no. 6, pp. 451–467, 2010.

[17] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for tcp traffic classification," *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009.

[18] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[19] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*. IEEE, 2014, pp. 797–801.

[20] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*. IEEE, 2006, pp. 967–974.

[21] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[22] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.

[23] R. Quinlan, "4.5: Programs for machine learning morgan kaufmann publishers inc," *San Francisco, USA*, 1993.

[24] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[25] B. R. Gaines and P. Compton, "Induction of ripple-down rules applied to modeling large databases," *Journal of Intelligent Information Systems*, vol. 5, no. 3, pp. 211–228, 1995.

[26] J. Fuernkranz and G. Widmer, "Incremental reduced error pruning," in *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, 1994, pp. 70–77.

[27] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

[28] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," 1998.

[29] M. Walker, "Random forests algorithm," Tech.

Rep., Accessed on 01.10.2014. [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[31] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. v. d. Veen, and C. Platzer, "Andrubis – 1,000,000 Apps later: A view on current android malware behaviors," in *Proc. Third Int. Workshop Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, Sep. 2014, pp. 3–17.

[32] I. M. Digit Oktavianto, *Cuckoo Malware Analysis*. Packt Publishing, 2013.

[33] B. H. Trammell and E. Boschi, "Bidirectional flow export using ip flow information export (ipfix) : Rfc-5103," IETF, Tech. Rep., Access Date 01 Jan, 2015. [Online]. Available: https://tools.ietf.org/html/rfc5103.html

[34] F. J. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms." in *ICML*, vol. 98, 1998, pp. 445–453.