https://dx.doi.org/10.4314/WOJAST.v15i1.78

DYNAMIC ANALYSIS OF MALWARAE INTRUSION IN MOBILE DEVICES USING ADABOOST ALGORITHM, KNN AND SVM BASE CLASSIFIERS

ISSN: 2141 – 3290
www.wojast.com

OYONG, S. B., EKONG, U. O., OBOT, O. U.
Department of Computer Science, University of Uyo, Uyo. Nigeria
samueloyong@gmail.com; uyiekong@yahoo.com;
okureobot@uniuyo.edu.ng

ABSTRACT

Cyber security is becoming more worrisome; malware is spreading by the day through proliferation and distribution of variants of known family signatures using obfuscation techniques. Mobile devices components such as central processing unit, memory, battery life, executable files and operating systems are constantly being attacked and rendered unusable. Attack agents are specifically evading detection, damaging mobile devices' executive files, stealing information, surcharging users for SMS sent and received without their knowledge or permission, and freezing applications for a ransom among others. This research work is keying into the fight against malware intrusion by designing and developing an intrusion detection system (IDS) using ensemble learning, boosting. Adaboost algorithm trains base classifiers (KNN and SVM) using network security laboratory-knowledge discovery in databases (NSL-KDD) dataset to build a more formidable classifier that will detect malware intrusion in mobile devices using cloud technology. The result obtained in this combination technique is 91.4% accurate with a bias (standard deviation) as low as 2.7%.

KEYWORDS: Mobile Devices, Malware Intrusion, IDS, Adaboost, KNN, SVM**.**

INTRODUCTION

Mobile devices have penetrated the world's economy in almost all fields, especially in e-commerce, social media, medicine, communication, education, among others. Their ubiquity has endeared them to many users including malware developers and hackers who exploit them for their nefarious activities such as blackmail and arm-twist users to grant dangerous permissions, distort operations and steal information and personal data to make money (Atkinson, 2015; Gamao, 2018). Other motives behind malware attacks include espionage, ideology (terrorism) and fun (Verizon, 2018; GSMA, 2019). For instance, nations are hacking nations, middlemen are hijacking businesses such as financial negotiations between clients, stealing credit card numbers and other sensitive data via the internet. Middlemen (sometimes called yahoo boys/girls) are gaining grounds by the day, despite nations' fight against their activities. For instance, online identity theft amounted to 16,128 cases and non-delivery fraud recorded 65,116 cases in 2018 (Stein, 2020). It was observed that Android devices are worst hit by malware and hackers because she sacrificed security for profit. Third party applications are accepted in Google play store, Google "bouncer" is not as effective as expected in reducing fake applications in Android market, and third-party organizations licensed to Android application variants are slow to effect patches made by Google. For instance, Android devices were infected by malware and bots to the tune of 47.15% in 2018 (GSMA, 2019). This research work tends to contribute positively to the fight against these threats, challenges and vulnerabilities to the security of mobile devices by developing an intrusion detection system (IDS), one of the security mechanisms. This is achieved by applying ensemble learning, using Adaboost algorithm to train base classifiers (KNN and SVM). The dataset used to train these classifiers is NSL-KDD dataset, obtained from Kaggle, a public data repository. It is made up of training and testing data sets, contained as observations (rows) and features (columns).

Recent trends in communication technology include proliferation of malware variants, use of encryption to hide embedded code in seemingly genuine applications, use of ransomware to harass innocent users, by freezing their phones for a ransom (Stein, 2020). These nefarious activities are achieved by malware developers using sophisticated tools, most of which are freely available on the net. Novel cyber-attacks are on the increase (encrypting embedded malware code in seemingly genuine applications) and recognizing controlled environment (sandboxes) and delay launching their payload to avoid detection. These tactics and many more have forced companies to incur serious financial losses, litigations for breach of contracts, and indeed, the reputation of affected companies (Sullivan, 2015). The attacks are categorized into denial of service (DOS), probe (surveillance), user to root (U2R), and remote to local (R2L). They are perpetrated by agent (programs written to carry out the wicked intents of their masters): viruses, worms, Trojans, rootkits, backdoor, ransomware, among others (Wang *et al.,* 2015; Bhuyan *et al.,* 2014)

One way to fight the menace of malware and their masters is the use of machine learning (ML) tools. ML attempts to find a suitable solution in a large space of possible solutions. Intrusion detection system (IDS) is one possible method of diagnosing attacks and abnormal behavior, through continuous observation of specific locations or objects of a network (Wang *et al.,* 2015*).* To develop an effective IDS, the research community has proposed the combination of ML techniques, as no single algorithm or classifier can do it all (Bamhdi *et al,* 2021; Bui *et al,* 2017). To conform to the above proposal, this research work combines Adaboost, KNN and SVM using ensemble learning to produce a more efficient classifier that will conduct binary classification of benign (normal) from anomalous applications seeking for permission from the user to either install themselves, other malicious applications or distort operations of the mobile device.

https://dx.doi.org/10.4314/WOJAST.v15i1.78

MATERIALS AND METHODS

The dataset used to train and test the proposed IDS is NSL-KDD dataset. It is obtained from Kaggle, a public data repository. It is further subdivided into training dataset (80%) with 125,973 records and testing dataset (20%) with 22, 544 records. In all, there are forty-two (42) features in each record of the dataset (Pham *et al.,* 2018*).* Figure 1 depicts column chart of the percentage of training and testing datasets used to train and test the models respectively.

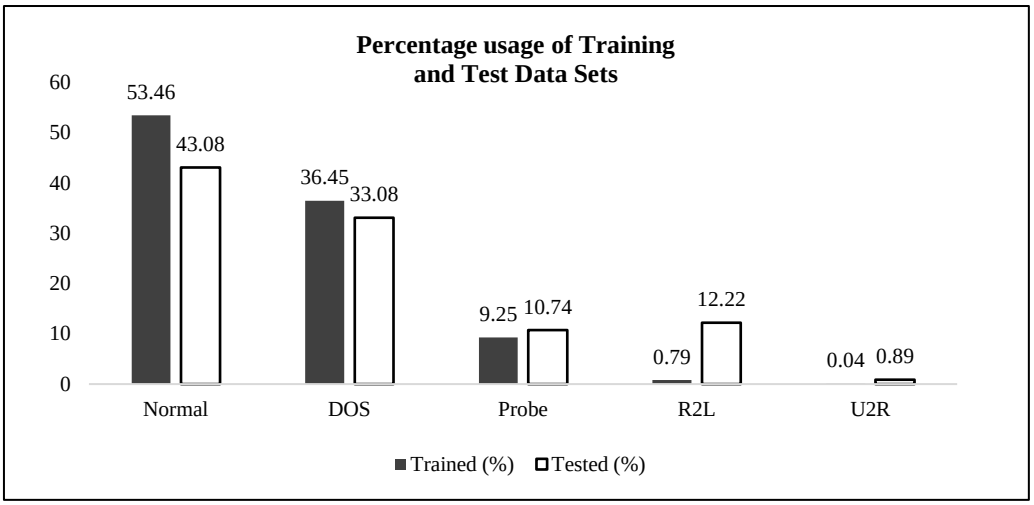

Figure 1: Column chart illustrating percentage of Training and Test datasets

System Development Tools

The laptop system used to develop the program is made up of the following configurations:

i. Operating System: Windows 10, with 64-bit word length
ii CPU: Inte® Celeron® 1000M, 1.80GHZ speed,
iii RAM: 4.00GB, HDD 500 GB, DVD Drive, Keyboard, and Mouse.
iv Printer: hp LaserJet p2035
v Applications: Microsoft Office Suite ver. 16: MS Word, Excel, and Power Point

Algorithms used in the research work

The training algorithm, the base models and voting classifier include.

i. Adaboost (Adaptive Boost) – Used to train and monitor base models and update the weights of misclassified labels.
ii. K-Nearest Neighbor (K-NN) – Base model, used to load training dataset into memory, finds the nearest neighbors to the target object using Euclidean distance measure; and assigns the target object to the majority vote.
iii. Support Vector Machine (SVM) – Base model 2, trains with the improved dataset and classify the test dataset into benign and anomalous classes.
iv. Voting Classification technique (majority vote or plurality vote or hard vote), used to aggregate the predictions of the base classifiers and determine a befitting classifier.
v. Python programming language, and its external libraries (scikit learn, pandas, numpy, matplotlib, etc). was used to code and plot the system (IDS).

Given the limitations of mobile devices, which are limited processing power, limited memory, battery longevity, and constrained operating system, the data and code are uploaded to the cloud, via the internet, for analysis and detection of malicious applications. This process is achieved using cloud computing technology.

System Requirements Specification (SRS)

This subsection explains the key operations to the programmer that lead to the computation of results. The program is written in python programming language, using both in-built and external libraries.

i. Import the relevant external libraries
ii. Load the dataset to train and access the models, using pandas (pd.). For instance,

Data = pd.read_csv("H:/user/oyong/desktop/nsl_kdd.csv")

iii Divide the dataset into input vector, $x \in R^d$ and label, $y \in \{-1, +1\}$.
iv Convert the Categorical features into numerical values One_hot_encoder() and Label_encoder() of sklearn library before processing
v Normalization the dataset using minMaxScaler() function, and split it into training dataset (80%) with 125,973 records and test dataset (20%) with 22,544 records using train_test_split() of sklearn library
vi Reduce the dimension of the dataset using principal component analysis (PCA)
vii Train the base models (KNN and SVM) using Adaboost algorithm, and aggregate their predictions into a formidable classifier (hard vote) using votingclassifier() function of sklearn library. For instance:

Ab_clf = AdaboostClassifier (n_estimators = 2, base_estimator = "KNN").

viii Classify (predict) the test dataset using hard vote classifier
ix Compare the predicted results with the expected values using confusion matrix, being a supervised learning problem and ascertain TP, TN, FP and FN values.

https://dx.doi.org/10.4314/WOJAST.v15i1.78

x Compute standard metrics such as Accuracy, Precision, Recall, F-measure, false positive rates (FPR) with respect to anomalous data types such as DOS, Probe, R2L and U2R.
xi Compare the results of this research work with that of other works in literature
xii Draw conclusion based on the results of the work, and suggest further works based on the limitations experienced with this research work.

RESULTS

In this Section, the predictions of boosted KNN and boosted SVM are depicted with respect to voting classification (Hard-vote). Hard vote counts the votes of each classifier in the ensemble and picks the class that gets the highest votes; box and whiskers visual representation, accuracy, precision, recall, f1-score and FPR.

Voting classification

In voting classification, the screenshot of mean and standard deviations of boosted KNN and boosted SVM are depicted in Figure 2. Observe that the Hard_vote value, which is an aggregation of the five sets of KNN with different k values, has 90.2% mean and 3.4% standard deviation (std.).

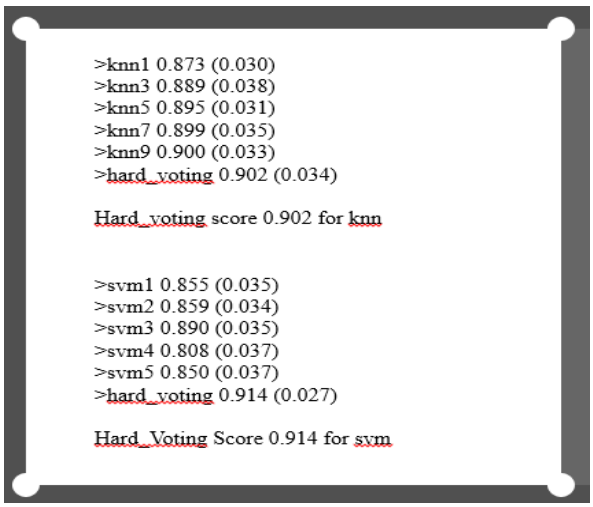

Figure 2: Screenshot of Mean and standard Deviation of KNN and SVM

Similarly, in SVM, the linear values are varied and the mean of the aggregated (hard_vote) model is 91.4%. It is observed that as the models are trained, and the weights of misclassified labels updated, the standard deviation or bias also falls, as depicted in the std. (2.7%) of boosted SVM as against that of boosted KNN, which is 3.4%.

Box and whiskers plot

Box-and-whisker plot depicts the spread of the data values in a dataset. Figure 3 depicts the graphical representation of boosted KNN using the box and whisker plot.

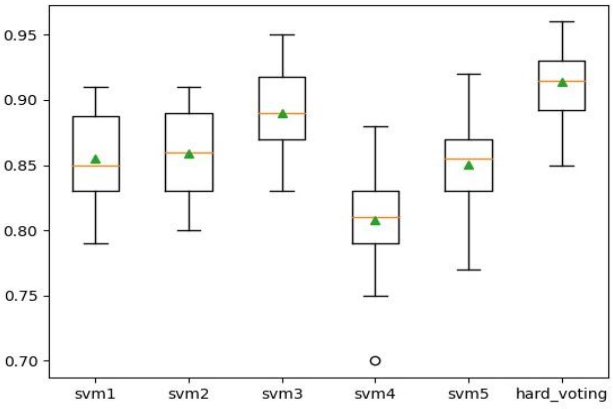

Figure 3: Graphical representation of voting classification of KNN with hard voting

It is observed that the spread seems steadily increasing as the values of the data elements or weights increase. Unlike knn3, the mean value of each box is lower than that of the median. However, the box and whiskers plot of SVM exhibits an interesting pattern. Almost all the box plots are skewed to the bottom, with the lower whiskers longer than the top whiskers as depicted in Figure 4. Another observation is that the variability is, indeed, scattered.

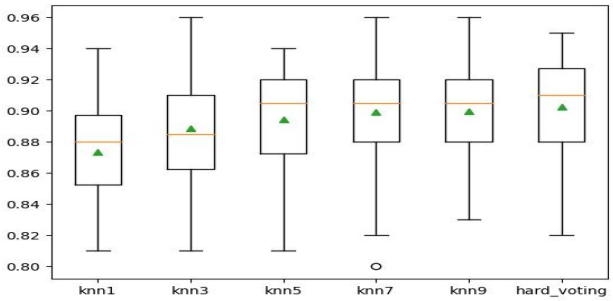

Figure 4: Box and whiskers plot of SVM voting classification.

Confusion matrix

The results of the trained model are evaluated to ascertain its generalization and performance with unknown dataset (test dataset). One way to achieve this is the use of confusion matrix. Confusion matrix is a cross table that records the number of occurrences between the predicted and actual classifications. While the columns represent model predictions, rows represent actual values (Kulkarni, 2022; Grandini *et al.*, 2020; Bhandari, 2020). Table 1depicts confusion matrix of multiclass classification problem for boosted KNN with the following classes (labels): DOS, Probe, R2L and U2R. With this type of matrix, unlike confusion matrix of binary classification problem, parameters such as True positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP) do not apply directly (Markoulidakis *et al*., 2021; Grandini *et al*., 2020; Bhandari, 2020). Because of that, the classes are analyzed one by one, with confusion matrix parameters TP, TN, FP

https://dx.doi.org/10.4314/WOJAST.v15i1.78

and FN determined in each case. Then performance parameters such as Accuracy, Precision, Recall, F1-score, etc. are computed using appropriate formulae.

Table 1: Confusion Matrix for Boosted KNN

		PREDICTED VALUE			
	Classes	DOS	Probe	R2L	U2R
ACTUAL VALUES	DOS	Cell 1 13979	Cell 3 36	Cell 4 0	Cell 5 0
	Probe	Cell 11 114	Cell 13 3213	Cell 14 1	Cell 15 0
	R2L	Cell 16 28	Cell 18 10	Cell 19 0	Cell 20 1
	U2R	Cell 21 0	Cell 23 0	Cell 24 0	Cell 25 0

Source: Researcher

Algorithm to compute Multiclass Confusion matrix parameters.

To calculate TP, TN, FP, and FN for each class, the following observations on Figure 7 are taken into consideration (Grandini *et al.*, 2020; Bhandari, 2020). For ease of explanation, the cells are numbered, while the numerical value there-in are generated by the developed python program using scikit learn and other external libraries:

TP: This is the cell value where the predicted and actual value are the same, and for each class, only one TP value is considered.

FN: For a class analysis, FN is the sum of values in cells of the corresponding row, except the TP cell value.

FP: The FP value for a class analysis is the sum of cells' values in the corresponding column, except the TP cell value.

TN: In a class analysis, the TN value is the sum of all the values in columns and rows, aside from those in the class being considered.

The same technique is applied in analyzing all the classes, then the computations are aggregated using python scikit learn metrics.confusion_matrix (y_test, y_pred) function.

Table 2 depicts confusion matrix of multiclass classification of boosted SVM. As explained in Table 1, the classes are also analyzed one by one, parameters computed using the algorithm and aggregated.

Table 2: Confusion matrix of boosted SVM

		PREDICTED VALUE			
	Classes	DOS	Probe	R2L	U2R
ACTUAL VALUES	DOS	Cell 1 13979	Cell 3 36	Cell 4 0	Cell 5 0
	Probe Normal	Cell 11 114	Cell 13 3213	Cell 14 1	Cell 15 0
	R2L	Cell 16 28	Cell 18 10	Cell 19 0	Cell 20 1
	U2R	Cell 21 0	Cell 23 0	Cell 24 0	Cell 25 0

Source: Researcher

Performance evaluation

Standard metrics were computed using confusion matrix multiclass parameters such as TP, FP, TN and FN. They include accuracy, precision, recall, f1-measure and false positive rate (FPR) with respect to actual and predicted values. The trained (boosted) KNN results are presented in Table 3.

Table 3: Performance Metrics with Respect to Anomalous Types using linear space for boosted KNN.

Attack Types	Accuracy %	Precision %	Recall %	F-Measure	FPR
DOS	99.0	99.6	99.7	99.7	0.0021
Probe	99.0	99.1	99.0	99.0	0.009
R2L	99.5	89.2	89.2	89.20	0.003
U2R	99.5	76.0	53.0	62.0	0.0001

(Source: system code)

Attack Type	Accuracy %	Precision %	Recall %	F-Measure %	FPR
DOS	94	97	86.5	91.3	1.8
Probe	88.7	84.3	96.0	90.0	19
R2L	97.5	95.0	97.5	88.6	0
U2R	99.92	94.6	98.0	97.0	0

The trained (boosted) SVM classification reports are presented in Table 4.

Table 4: Performance Metrics with Respect to Anomalous Types using linear kernel for boosted SVM.

Source: system code

From the computations in Table 3 and Table 4, it is observed that the boosted KNN performed almost as good as boosted SVM in linear space. While the highest accuracy in boosted KNN is (99.5%) that of SVM is (99.92%). This points to the fact that SVM is not efficient in linear space, but higher spaces (Wang and Wang, 2015). The hard vote value (91.4%) of SVM is indeed better than that of KNN (90.2%). This proves that Adaboost did a good job in training the models, starting with KNN and after evaluating its mistakes, updated the weights of misclassified labels, it came up with a better score (91.4%) for SVM with standard deviation as low as (2.7%).

From Figure 5, all the performance measures recorded almost the same values for DOS and Probe; R2L also recorded almost equal values in Precision, Recall and F-measure. However, U2R has a progressive decline in terms of Accuracy, Precision, Recall and F-measure. False positive rate was insignificant, in all the anomalous types.

Figure 6 depicts boosted SVM in terms of anomalous types. Please note that the Normal class was allowed to terminate, as our interest is in the anomalous classes and how to control their effect on mobile devices. The second part of this research work handles the control part (IRS and how it selects optimum counter measure against each attack type)

.

Figure 5 illustrates the standard metrics of KNN using a bar char.

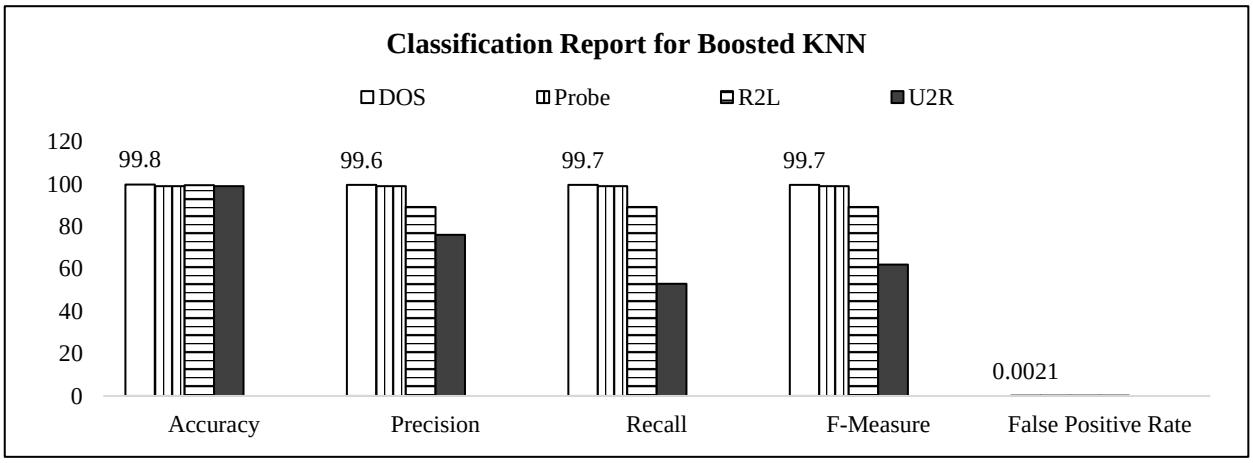

Figure 5: Classification Report on boosted KNN in terms of anomalous types.

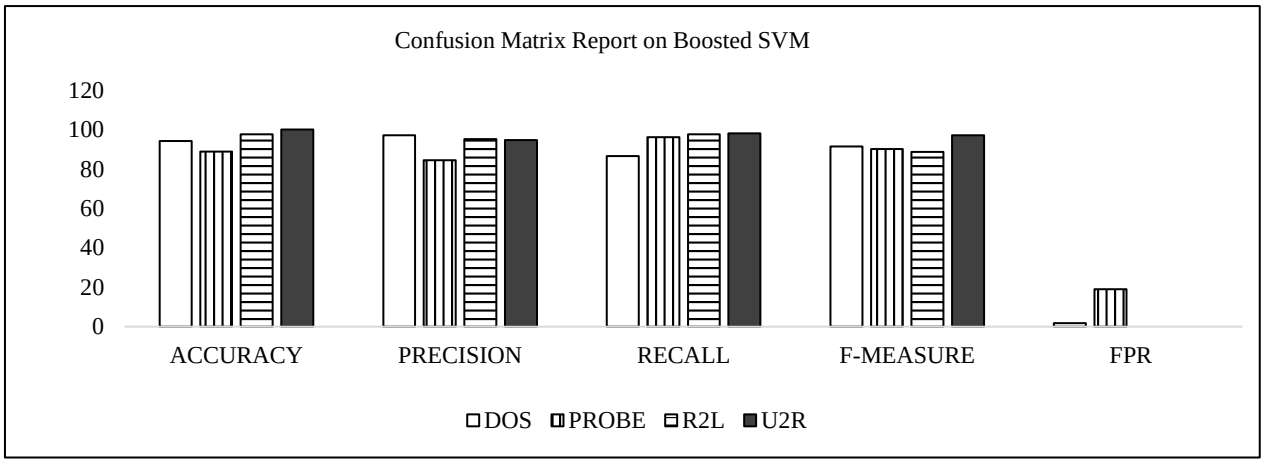

Figure 6: Classification Report on boosted SVM in terms of anomalous types

From figure 6, U2R is virtually uniform in terms of all the measurement values.

U2R has the highest accuracy (99.92%), while Probe provides the least accuracy (88.7%). In precision, DOS has the highest value (97%), while Probe has the least value (84.3%). Similarly, in Recall, U2R has the highest value of (98.0%) while DOS has the least value (86.5%). In F-measure, U2R provides the highest value of (97.0%), while R2L has as low as (88.6%). FPR is high in some anomalous types like Probe (19.0%) while in others, zero percent was recorded, especially R2L and U2R. These values, especially in FPR, are below the saturation point of 0.5 (50% random guess error for Adaboost), which is the accepted tolerance. The saturation point of 50% random guess error for Adaboost is the turning point where adding more weak classifiers would produce no further increase in efficiency (Bhandari, 2022)

DISCUSSION

In this section, the results of this research work are compared with that of other works in literature.

Basis for Comparison of Results

The works in literature compared with this research work operate under the same Android operating system, used either ensemble learning approach or hybridization technique. The dataset used to train and analyze the models is KDDcup'99 or its variant NSL-KDD dataset. The base models trained include KNN and SVM. They have a common intent of detecting malware using permissions, although some in addition used APIs. The analysis was done using cloud computing technology, although some papers used static analysis. However, while this work uses Adaboost (an ensemble boosting technique) to train the base models (KNN and SVM), the other works used Adaboost or particle swarm optimization (PSO) algorithm.

Comparison of Results with that of other works in literature

Table 5 depicts a collection of papers and their result that will be compared with values from this research work. From Table 3, serial number 4 and 7 depict values from this research work.

https://dx.doi.org/10.4314/WOJAST.v15i1.78

Table 5: Comparing Results of This Research Work with Other works in literature

S/N	AUTHORS/CLASIFIERS	ACCURACY	FAR
1	Kakavand *et al*. (2018) Static analysis		
	KN	80.50%	N/A
	SVM	79.08%	N/A
2	Pham *et al*. (2018) Ensemble learning:		
	Adaboost with J48 DT)	80.59%	3.16%
	(Adaboost with RF)	80.07%	3.04%
3	Li *et al*. (2015) Single Classifier: SVM	86.00%	N/A
4	[This work]: Single classifier:		
	KNN	90.20%	0
	Adaboost +SVM	91.40%	2.7%
	Adaboost + DT	85.18%	N/A

	Authors/Classifiers	ACCURACIES					FAR %
		Normal %	DOS %	Probe %	R2L %	U2R %	
5	Aburomman et al. (2016): Ensemble learning: PSO algorithm (with KNN, and SVM as base Classifiers)	83.46	98.85	96.14	84.73	99.81	N/A
6	This research work:	N/A	94.00	88.70	97.50	99.92	19

N/A – Not Applicable
KNN – K-Nearest Neighbor
SVM – Support Vector Machine
J48 – Decision Tree Classifier
RF – Random Forest Classifier
PSO – Particle Swarm Optimization
FAR – False Alarm Rate
DOS – Denial of Service attack type
Probe – Surveillance attack type
R2L – Remote to Local attack type
U2R – User to Root attack type

From Table 5, the following observations are made with respect to each paper.

i. Kakavand *et al*. (2018) carried out static analysis of applications, and got the following accuracies KNN (80.50%) and SVM (79.08%). This research work used ensemble learning to dynamically train and analyze classifiers, and outperforms the paper with KNN (90.20%) and SVM (91.40%).

ii. Pham *et al*. (2018) carried out an ensemble learning analysis of applications using both bagging and boosting. The results of adaboost with tree-based classifiers was considered and compared with this work. Adaboost (J48) had an accuracy of 80.59% and Adaboost (RF) had an accuracy of 80.07%. However, in this research work, Adaboost (KNN) outperformed Adboost (J48) with an accuracy of 90.20% and Adaboost (SVM) having an accuracy of 91.40%. which is better than Adaboost + RF in Pham *et al.* Observe that Pham *et al.* (2018) computed FAR of 3.16% DT and a reduced value of 3.04% RF . However, this research work has FAR as high as 19% (Probe), with a bias as low as 2.7%.

iii. Li *et al*. (2015) applied dynamic analysis on SVM using risky permissions and vulnerable APIs. The result had an accuracy of 86.00%. This research work had a better SVM result of 91.40%. The drawback Li *et al.,* (2015) is that it used only one classify, SVM, to carry out its analysis, given the sophistication of malware these days, this is not desirable.

iv. Aburomman *et al.* (2016) combined classifiers using ensemble learning, with particle swarm optimization (PSO) algorithm, and KNN and SVM as base classifiers. Metrics results were also spread through the classes. Comparing their results with Adaboost (Hard_vote), it is observed that Adaboost (Hard_vote) was outperformed by PSO in DOS (98.85%) and Probe (96.14%); while Adaboost in this research work outperformed PSO in Aburomman *et al.,* (2016) with results of R2L (97.50%) and U2R (99.92%)

CONCLUSION

The efficiency rate of 91.4% and a false positive rate of 0.024% will give the user trust and confidence in transacting business over the internet – electronic commerce (also called e-commerce). It will also instill confidence in the use of mobile devices to transact business and reduce queues in banking halls, fear of hijacking business transactions by middle men (popularly called 'yahoo boys/girls' or 419 operators). This system will in no small way increase sales and usage of mobile devices, reduce risk of theft, reduce cost and accident rates since most transactions can be carried out in the comfort of one's room.

This research work has contributed to knowledge in the following ways:

i. Designing and implementing an IDS application that will detect anomalous applications in mobile devices using cloud computing technology.
ii. The application is tested for efficiency and scored 91.4% accuracy with FPR as low as 0.024%
iii. The work further demonstrated the use of ensemble learning to curb intrusion, and reduce the sophistication of malware over single classifiers.

The introduction of an intelligent system to detect malware in mobile devices for e-commerce is timely; especially these days that COVID'19 is ravaging the world with many variants, and now omicron variant. COVID'19 restricts movement, reduces social gathering, economic activities, especially in markets, and spiritual activities in churches and mosques.

REFERENCES

Atkinson, M. (2015). An Analysis of Android Application Permissions. Pew Research Center, *Internet and Technology.*

Bamhdi, A. M., Abram, I. and Masoodi, F. (2021). An Ensemble Based Approach for Effective Intrusion Detection Using Majority Voting. *TELKOMNIKA Telecommunication, Computing, Electronic and Control*. 19(2): 664-671.

Bhuyan, M. H., Bhattacharyya, D. K. and Kalita, J. K. (2014). Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys and Tutorials,* 16(1): 303-336.

Bui, L.T., Vu, V.T. and Dinhm T.T.H. (2017). A Novel Evolutionary Multi-Objective Ensemble Learning Approach for Forecasting Currency Exchange Rates. *Data knowledge Engineering* 114: 40-66.

Domingos, P. (2012). A Few Useful Things to Know About Machine Learning. *Communication of the ACM,* 55(10): 78–87.

Gamao, A. O. (2018). Malware Analysis on Android Applications: A Permission Based Approach. *Social Science and Humanity Journal of Humanity,* 02(10): 624-633

GitHub Inc. (2020). NSL-KDD Dataset. https://www.github.com.

GSMA (2019). Mobild Telecommunications Security Threat Landscape. Floor 2, the Walbrook Building, 25 Walbrook, London EC4N 8AF United Kingdom.

Iankoulova, I. and Daneva, M. (2012). Cloud computing security requirements: A systematic review. In: Research challenges in information science (RCIS) 2012, 6th International Conference on, 1.

Koshal J. and Bag, M. (2012). Cascading of C4.5 Decision Tree and Support Vector Machines for Rule Based Intrusion Detection System. *International Journal of Computer Network and Information Security,* 8: 8-20.

Lisehroodi, M. M., Muda, Z., and Yasin, W. (2013). An hybrid Framework Based on Neural MLP and k-Means Clustering for Intrusion Detection Systems. *Proceedings of the 4th International Conference on Computing and Information. (ICONCI), No. 020*

Popovic, K. and Hocenski, A. (2010). Cloud computing security issues and challenges. In:MIPRO. 2010 *proceedings of the 33rd International Convention. Pp.* 344-349.

Pham, N.T., Foo, E., Suriadi, S., Jeffrey, H and Lahza H.F. (2018). Improving Performance of Intrusion Detection Systems Using Ensemble Methods and Feature Selection. *In (ACSW) Australian Computer Science Week, Brisbane, OID Australia.* https://dio.org/10.1145/3167918.3167951

Stein, J. (2020). Data Breach Report, *Nortyh Carolina Department of Justice. www.ncdoj.gov/complaint*

Sullivan, D.T. (2015). Survey of Malware Threats and Recommendations to Improve Cyber Security for Industrial Control Systems version 1.0. US Army Research Laboratory (ARL)

Wang, P. and Wang, Y. (2015). Malware behavioral detection and vaccine development by using a support vector machine model classifier. *Journal of Computer and System Sciences*. 81: 1012 – 1026.

Wang, G., Hao, J; Ma, J. and Huang, L. (2010). A new approach to Intrusion Detection using Artificial Neural Networks and Fuzzy clustering. ELSEVIER: *Expert Systems with Applications*, 37:6225 – 6232

Wang, H., Guo, Y., Tang, Z., Bai, G. and Chen, X. (2015). Reevaluating Android Permission Gaps with Static and Dynamic Analyses. *School of Electronic and Computer Science, Peking University, Beijing, China.*

Verizon (2018). 2018 Data Breach Investigations Report. www.verizonenterprise.com/Federal