Detection of Android Adwares by using Machine Learning Algorithms

Dinesh C Dobhal, Purushottam Das, Kiran Aswal

Abstract*: With the rapid growth in technology, which is improving every day and becoming more pervasive, smartphone users also are increasing. These intelligent devices and gadgets are now being used in automated vehicles, IoT enabled industries, surveillance, education, entertainment, etc. Android, Linux kernel based mobile operating system, with its largest market share is now being used in almost every device that is capable to do some computation. These devices may include smartwatches, digital cameras, smart glasses, smart mirrors, Home Automation System (HAS), Internet of Things (IoT), Internet of Vehicles (IoV), and many more. Parallel to these developments, the android based systems also are being targeted by the cyber attacker by developing more advanced malwares. Such attacks may harm to the system as well as human life. The android malwares are evolving day by day and are capable to escape the traditional security solutions. Therefore, security is the primary issue of android based system, which requires to be re-investigated. In this paper, we analyze the pertinence of machine learning based solutions to detect android malware, particularly Adware. Logistic Regression (LR), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Classification And Regression Trees (CART), and Naive Bayes (NB) machine-learning algorithms are trained and tested for two scenarios. Scenario A for binary classification and Scenario B is for multi-class classification. The 60% of the dataset is used to train the ML algorithms and the remaining 40% is reserved for the testing. The algorithms are evaluated by using 10-fold cross-validation method.

Index Terms*: Network Security, Adware, Malware, Anomaly detection, Machine Learning (ML).

I. INTRODUCTION

Due to the availability of advanced technologies at an affordable cost, the smartphone has rapidly become a more prevalent communication and computing platform. Android, Linux kernel based mobile operating system, in particular, has seen even more impressive growth, with the highest worldwide market share of 75.16% in December 2018 [1]. It was designed primarily for touchscreen enabled mobile devices and has been the best-selling operating system across the world on smartphones since 2011 and on tablets since 2013. Today, the android OS can be found in the root of smartwatches, smart glasses, gaming console, smart TV, smart mirror, refrigerators, washing machines, Internet of Things (IoT), Internet of Vehicles (IoV), Home automation systems (HAS) and many more devices. According to [2], the number of android apps on Google Play Store is increased to 2.02 million apps from 1 million apps in July 2013. Parallel to these developments, the number of attacks made on the Internet is increasing day by day and the popularity of android based systems has not gone unnoticed by cyber attackers. There are already detected and documented incidents of malwares, such as Adwares, SMSware, Spyware, Trojan, Ransomware, etc., attacks on android based systems [3]. The malwares were studied by different authors in the past, and numerous solutions were recommended by them in[4]–[8]. Against these attacks, there are two basic approaches used to detect them; (1) Signature based detection approach, and (2) Anomaly based detection approach.

Signature-based methods use the database they created to detect attacks. This approach is quite successful, but the databases need to be constantly updated for the signature of new attacks. Moreover, even if the databases are up-to-date, they generate overhead and are vulnerable to the zero-day (previously unseen) attacks. As modern malwares do use encryption and obfuscation techniques, the signature based solutions are not effective against them.

The anomaly-based approaches focus on detecting unusual network behaviors by examining the network flow. This approach managed to get the attention of many researchers because of its ability to detect previously unseen attacks (aka zero day attacks) without the need to be configured with a specific signature of every possible attack [9]–[11].

Several of Machine Learning (ML) based algorithms have been used by academicians and researchers to study their applicability in anomaly-based malware detection solutions. Most of the research works, however, do not differentiate among different types of malwares [12]. The author in [5], survey and analyze the research since the early 2000's and found that most of the research has been carried out on simulated datasets. The authors highlighted that more research is to be carried out in light of real-world datasets. As the behavior of an attack varies from each other, it is required to study the behavior of each malware class in isolation. Therefore, the performance of machine learning algorithms is required to be re-investigated for binary and multi-class classification of malware families.

The goal of this research paper is to:

- Highlight the behavior and possible security threats of Adwares and its well-known family members.

Revised Manuscript Received on April 25, 2019.

Dinesh C Dobhal, Department of CS&E , Graphic Era University, Dehradun, India.

Purushottam Das, Department of CS&E , Graphic Era Hill University, Dehradun, India..

Kiran Aswal, Department of CS&E , Graphic Era Hill University, Dehradun, India.

- Prepare and pre-process the labeled real-world dataset, which would contain the traces of different types of Adwares.
- Identify the optimal set of features, which better describes the unique behavior of Adwares.
- Train the machine learning algorithms from the pre-processed dataset with benign and malicious instances.
- Investigate the performance of the machine learning algorithms for binary and multi-class classification of the Adwares instances.
- Evaluate the performance of the different ML algorithms for the optimal set of features.

Adware is generally referred to as a type of malware that displays unwanted advertisements or pop-ups on your computer or mobile device. It is Potentially Unwanted Application (PUA) and is commonly activated without the knowledge of the user when the mobile users install legitimate applications that are bundled with Adwares. These Potentially Unwanted Application (PUA), usually are malicious and may infect the system by hijacking the browser and installing other malwares, such as viruses and Trojans. These infected systems may be controlled by the cyber attacker from a unknown source and are used to compromise more systems. These attacks may cause a security threat for the safety of the system, privacy of the information, and the human life. Some of the malware from the Adware family, which are considered in this paper are;

Table 1: Adware samples considered in the study

1. ADWARE:ANDROID/DOWGIN	
Aliases:	Adware:android/dowgin, Android.Adware.Dowgin, Adware:android/dowgin.[variant]!gen
2. Adware/Ewind!Android	
Aliases:	Adware.AndroidOS.Ewind PUP/Android.Ewind.336663 Adware.AndroidOS.Adir.A!c
3. Adware/Feiwo!Android	
Aliases:	ADWARE/ANDR.Feiwo.D.Gen Trojan.Android.Ewind.ebksbm PUA.AndroidOS.Feiwo
4. Adware/Gooligan!Android	
Aliases:	PUP/Gooligan!Android
5. Android.Kemoge	
Aliases:	cc.taosha.beautify.easylocker cc.taosha.toolbox.shareit Android.Gorpo.A Trojan:Kemoge
6. Android Adware Koodous	
Aliases:	Android/Adware.Koodous Adware.AndroidOS!Koodous

In this paper, we investigate the efficiency of ML algorithms for the binary and multi-class classification of Adwares. In the next section, the methodology used to conduct the experiment is discussed; in section III, we discuss the pre-processing of the dataset and the experimental setup for training and testing of machine learning algorithms; in section IV, we present and analyze the outcomes of the experiments, and section V contains the conclusion of the paper.

II. METHODOLOGY USED

The methodology used in this paper is given below.
Step 1: Two datasets, one for binary classification and another for multi-class classification, are prepared from raw malware traces. These datasets are then preprocessed using python 3.5 and the non-significant features, which contain zero values for all instances, are removed.
Step 2: Random forest algorithm is used to reduce the dimensionality of the datasets.
Step 3: The datasets are then normalized using the MinMax normalization method.
Step 4: The normalized dataset is divided into two half. 60% of the dataset is used to train the ML model, whereas the remaining 40% is reserved for the testing of the trained model.
Step 5: The machine learning algorithms are tested in two different scenarios. Scenario A; for binary classification, and Scenario B; for multi-class classification. Each scenario is further tested for a reduced set of features. The performance is then validated by using a 10-fold cross-validation method.
Step 6: Results are collected, analyzed and the conclusion is derived.

III. PREPROCESSING OF THE DATASET

In this section, we attempt to identify the optimal set of features, which better describes the unique behavior of the Adwares. The Adware datasets are prepared, for binary and multi-class classification both, by extracting the malware traces form the Android malware dataset CICMalAnal2017 [13]. To avoid the change in runtime behavior of the advanced malware that can detect the sandbox environment, the network and system behavior are recorded by executing both malware and benign applications on real smartphones. The different Adware classes that are considered in the multi-class classification dataset are mentioned in *table 1*.

The malware dataset, which is extracted and processed from [13], contains 681043 instances (71487 BENIGN and 609556 malware instances) and 81 features. The distribution of the malware instances is given in *table 2*.

Some instances in the raw dataset contain null and infinite values. Such instances are eliminated during preprocessing of the dataset. The 12 features contain zero values for all instances, and therefore removed from the datasets.

These non-significant features are 'BwdPSHFlags', 'FwdURGFlags', 'BwdURGFlags', 'RSTFlagCount', 'CWE Flag Count', 'ECE Flag Count', 'Fwd Avg BytesPBulk', 'Fwd Avg PacketsPBulk', 'Fwd Avg Bulk Rate', 'Bwd Avg BytesPBulk', 'Bwd Avg PacketsPBulk', and 'Bwd Avg Bulk Rate'. The number of features of the datasets is reduced from 81 to 69.

Table 2: Samples in the dataset

Class	# Sample
ADWARE_GOOLIGAN	187544
ADWARE_FEIWO	113264
ADWARE_EWIND	86748
ADWARE_DOWGIN	79364
ADWARE_KEMOGE	77542
ADWARE_KOODOUS	65094
BENIGN	71487
Total	**681043**

A. Selection of Normalized feature set

The selection of optimal Feature set is a well defined process of selecting a set of relevant features from a dataset. This process significantly enhance the efficiency of the ML model. It also reduces the training time and the complexity of the model. In our study, the dataset after the initial phase of preprocessing is left with 69 features.

In order to find optimal set of features, the Random Forest algorithm is used that reports the relative importance of each feature in between 0 and 1. The 20% least important features are removed from the dataset. The 13 least importanct features are 'ActiveMin', 'Active Max', 'ActiveMean', 'ActDataPktFwd', 'ACKFlagCount', 'DownPUpRatio', 'PSHFlagCount', 'Idle Std', 'FINFlagCount', 'Protocol', 'Active Std', 'FwdPSHFlags', 'SYNFlagCount'.

The shape of the dataset, after elimination of these non relevant features, becomes, 681043 rows, and 56 Columns.

The features of the dataset are required to be normalized before feeding into the training phase of ML models. Therefore, The MinMax normalization method is used to normalize the features of the dataset into a range from 0 to 1. For a given feature f, whose value is between f_{min} and f_{max}, can be normalized by the following formula,

$$f_{normalize} = \frac{f - f_{min}}{f_{max} - f_{min}} - - - (1)$$

B. Experimental setup for machine-learning model

An Intel i5 machine with 8 GB of RAM with 64 bit version of Ubuntu 18.04 Operating System is selected to implement and train our ML model. The machine is configured for the experiments with Python version 3.6 with the libraries Numpy [14], Scikit-learn [15], and data analysis library Pandas [16]. Logistic Regression (LR) [17], Linear Discriminant Analysis (LDA) [18], K-Nearest Neighbors (KNN) [19], Classification And Regression Trees (CART) [20], and Naive Bayes (NB)[21] machine-learning algorithms are trained and tested for two scenarios. Scenario A for binary classification and Scenario B is for multi-class classification. The ML algorithms are trained using 60% of the dataset and the remaining 40% of the dataset is reserved for the testing purpose. The algorithms are validated by using k-fold cross-validation (k=10). Further, ML models are trained and validated again for Scenario A and B both, with a reduced feature set.

IV. RESULTS AND DISCUSSION

The outcome of the experiments is evaluated by using the following metrics:

Accuracy: This metric reports the overall efficiency of the model, and represents the ratio of the total instances that were correctly predicted by the ML model.

Precision: This metric represents how many positive predictions were present as positive in the dataset.

Recall: This metric represents what percentage of all positive samples were correctly classified as positive by the ML classifier.

F1-Score: It is derived as a harmonic mean of precision and recall.

Macro Average: This metric is calculated for each class separately and then use the unweighted mean of the measured values.

Weighted Average: Unlike Macro F1, this metric use a weighted mean of the measured values. The weight for each class is taken from the total number of instances of that class.

The performance of ML models for binary classification of Adware is stored in table 3 to table 7 and for the multi-class classification of six different classes of Adware if stored in table 8 to table 12.

From the results collected, we observed that the accuracy of the Classification And Regression Trees (CART) algorithm is 90.19 %, which outperformed other algorithms for binary classification. Logistic Regression, Linear Discriminant Analysis LDA, K-Nearest Neighbors, and Naive Bayes (NB) reported the accuracy of 79%, 79.28%, 82.03%, and 77.76% respectively. For the reduced feature set (56 features), the algorithms produce almost the same performance. Similarly, for multi-class classification experiments, the accuracy of the Classification And Regression Trees (CART) algorithm is 61.37%, which outperformed other algorithms. Logistic Regression, Linear Discriminant Analysis LDA, K-Nearest Neighbors, and Naive Bayes (NB) reported the accuracy of 18.30%, 31.73%, 38.62%, and17.69% respectively. For the reduced feature set (56 features), these algorithms produced slightly better accuracy, but as compared with the results obtained for binary classification, there is a sharp decline in the performance.

V. CONCLUSION

In this paper, the well known ML algorithms are investigated with the malware dataset, which contain normal as well as Adware instances of six different families. The algorithms are trained and tested for binary and multi-class classification of Adwares. The accuracy of these algorithms is validated using 10-fold cross-validation methods. By using the Random Forest algorithm, an optimal set of 56 features is identified. This optimal set of features significantly improved the training time of the algorithms. From the results discussed in section 4, we can conclude that the CART algorithm outperformed others for binary and multi-class classification. However, there is a scope to improve its accuracy. Hence, a more rigorous study is required to engineer the feature set of the Adware dataset, specially for the multi-class classification scenario.

Table 3: Senario A Classification Report of Logistic Regression (LR)

ML Algorithm	Logistic Regression (LR)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	Accuracy
ADWARE	0.79	1.00	0.88	79 %
BENIGN	0.45	0.00	0.01	
Macro Average	**0.62**	**0.50**	**0.45**	
Weighted Average	**0.72**	**0.79**	**0.70**	
Accuracy with reduced feature set				79.12 %

Table 4: Senario A Classification Report of LDA

ML Algorithm	Linear Discriminant Analysis (LDA)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	Accuracy
ADWARE	0.80	0.99	0.88	79.28 %
BENIGN	0.52	0.02	0.04	
Macro Average	**0.66**	**0.51**	**0.46**	
Weighted Average	**0.74**	**0.79**	**0.71**	
Accuracy with reduced feature set				79.13 %

Table 5: Senario A Classification Report of K-Nearest Neighbors (KNN)

ML Algorithm	K-Nearest Neighbors (KNN)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	Accuracy
ADWARE	0.85	0.94	0.89	82.03 %
BENIGN	0.61	0.38	0.47	
Macro Average	**0.73**	**0.66**	**0.68**	
Weighted Average	**0.80**	**0.82**	**0.80**	
Accuracy with reduced feature set				83 %

Table 6: Senario A Classification Report of CART

ML Algorithm	Classification And Regression Trees (CART)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	Accuracy
ADWARE	0.92	0.97	0.94	90.19 %
BENIGN	0.83	0.66	0.74	
Macro Average	**0.87**	**0.81**	**0.84**	
Weighted Average	**0.90**	**0.90**	**0.90**	
Accuracy with reduced feature set				91.72%

Table 7: Senario A Classification Report of Naive Bayes (NB)

ML Algorithm	Naive Bayes (NB)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	Accuracy
ADWARE	0.79	0.97	0.87	77.76 %
BENIGN	0.27	0.04	0.07	
Macro Average	**0.53**	**0.51**	**0.47**	
Weighted Average	**0.69**	**0.78**	**0.71**	
Accuracy with reduced feature set				77 %

Table 8: Senario B Classification Report of Logistic Regression (LR)

ML Algorithm	Logistic Regression (LR)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	**Accuracy**
ADWARE_DOWGIN	0.17	0.23	0.20	18.30 %
ADWARE_EWIND	0.18	0.06	0.09	
ADWARE_FEIWO	0.26	0.06	0.10	
ADWARE_GOOLIGAN	0.18	0.18	0.18	
ADWARE_KEMOGE	0.25	0.11	0.15	
ADWARE_KOODOUS	0.15	0.55	0.24	
BENIGN	0.44	0.04	0.08	
Macro Average	**0.23**	**0.18**	**0.15**	
Weighted Average	**0.23**	**0.17**	**0.15**	
Accuracy with reduced feature set				20.38 %

Table 9: Senario B Classification Report of LDA

ML Algorithm	Linear Discriminant Analysis (LDA)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	**Accuracy**
ADWARE_DOWGIN	0.26	0.19	0.22	31.73 %
ADWARE_EWIND	0.29	0.35	0.32	
ADWARE_FEIWO	0.37	0.22	0.28	
ADWARE_GOOLIGAN	0.29	0.42	0.34	
ADWARE_KEMOGE	0.27	0.42	0.33	
ADWARE_KOODOUS	0.15	0.05	0.07	
BENIGN	0.51	0.57	0.54	
Macro Average	**0.31**	**0.32**	**0.30**	
Weighted Average	**0.31**	**0.32**	**0.30**	
Accuracy with reduced feature set				33.21 %

Table 10: Senario B Classification Report of KNN

ML Algorithm	K-Nearest Neighbors (KNN)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	**Accuracy**
ADWARE_DOWGIN	0.28	0.45	0.35	38.62 %
ADWARE_EWIND	0.47	0.55	0.51	
ADWARE_FEIWO	0.40	0.39	0.40	
ADWARE_GOOLIGAN	0.43	0.37	0.40	
ADWARE_KEMOGE	0.38	0.35	0.36	
ADWARE_KOODOUS	0.39	0.27	0.32	
BENIGN	0.57	0.53	0.55	
Macro Average	**0.42**	**0.41**	**0.41**	
Weighted Average	**0.42**	**0.41**	**0.41**	
Accuracy with reduced feature set				40.85 %

Table 11: Senario B Classification Report of CART

ML Algorithm	Classification And Regression Trees (CART)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	**Accuracy**
ADWARE_DOWGIN	0.55	0.60	0.57	61.37 %
ADWARE_EWIND	0.74	0.72	0.73	
ADWARE_FEIWO	0.62	0.62	0.62	
ADWARE_GOOLIGAN	0.69	0.67	0.68	
ADWARE_KEMOGE	0.56	0.58	0.57	
ADWARE_KOODOUS	0.52	0.50	0.51	
BENIGN	0.71	0.68	0.69	
Macro Average	**0.63**	**0.62**	**0.63**	
Weighted Average	**0.63**	**0.63**	**0.63**	
Accuracy with reduced feature set				67.96%

Table 12: Senario B Classification Report of Naive Bayes (NB)

ML Algorithm	Naive Bayes (NB)			
Class / Metrics	**Precision**	**Recall**	**f1-score**	**Accuracy**
ADWARE_DOWGIN	0.36	0.02	0.04	17.69 %
ADWARE_EWIND	0.17	0.82	0.28	
ADWARE_FEIWO	0.29	0.09	0.13	
ADWARE_GOOLIGAN	0.17	0.11	0.13	
ADWARE_KEMOGE	0.26	0.19	0.22	
ADWARE_KOODOUS	0.17	0.01	0.01	
BENIGN	0.43	0.03	0.06	
Macro Average	**0.26**	**0.18**	**0.12**	
Weighted Average	**0.26**	**0.19**	**0.13**	
Accuracy with reduced feature set				17.02 %

REFERENCES

1. "Statcounter Global Stats," 2018. https://gs.statcounter.com/os-market-share/mobile/worldwide.
2. "Statista." https://www.statista.com.
3. "Quick Heal Annual Threat Report," 2018, [Online]. Available: http://dlupdate.quickheal.com/documents/others/Quick_Heal_Annual_Threat_Report_2017.pdf.
4. M. Yousefi-Azar, L. G. C. Hamey, V. Varadharajan, and S. Chen, "Malytics: A Malware Detection Scheme," *IEEE Access*, vol. 6, pp. 49418–49431, 2018, doi: 10.1109/ACCESS.2018.2864871.
5. G. K. Rajbahadur, A. J. Malton, A. Walenstein, and A. E. Hassan, "A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety," in *Proceedings of IEEE Intelligent Vehicles Symposium*, 2018, pp. 421–426, doi: 10.1109/IVS.2018.8500383.
6. A. Ray and A. Nath, "Introduction to Malware and Malware Analysis: A brief overview," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 4, no. 10, pp. 22–30, 2016.
7. R. Sihwail, K. Omar, and K. A. Zainol Ariffin, "A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 4–2, p. 1662, Sep. 2018, doi: 10.18517/ijaseit.8.4-2.6827.
8. M. Imran, M. T. Afzal, and M. A. Qadir, "A comparison of feature extraction techniques for malware analysis," *TURKISH Journal Of Electrical Engineering & Computer Sciences*, vol. 25, pp. 1173–1183, 2017, doi: 10.3906/elk-1601-189.
9. M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016, doi: 10.1016/j.jnca.2015.11.016.
10. S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," in *Procedia Computer Science*, 2015, vol. 60, no. 1, pp. 708–713, doi: 10.1016/j.procs.2015.08.220.
11. R. Laxhammar, "Anomaly Detection," in *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications*, 2014.
12. T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine Learning for Anomaly Detection and Categorization in Multi-Cloud Environments," in *Proceedings - 4th IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2017*, 2017, pp. 97–103, doi: 10.1109/CSCloud.2017.15.
13. A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," 2018, doi: 10.1109/CCST.2018.8585560.
14. NumPy, "NumPy - Package for scientific computing with Python," 2018. http://www.numpy.org/.
15. F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 2011.
16. Panda, "Python Data Analysis Library." https://pandas.pydata.org/.
17. L. J. Davis and K. P. Offord, "Logistic regression," in *Emerging Issues and Methods in Personality Assessment*, 2013, pp. 273–283.
18. P. Xanthopoulos, P. M. Pardalos, and T. B. Trafalis, "Linear Discriminant Analysis," in *Robust Data Mining, Springer*, 2013, pp. 27–33.
19. P. Cunningham and S. J. Delany, "K -Nearest Neighbour Classifiers," *Multiple Classifier Systems*, vol. 8, no. 34, pp. 1601–1612, 2007, doi: 10.1016/S0031-3203(00)00099-6.
20. L. Breiman, "Bagging predictors," *Machine Learning*, vol. 2, no. 24, pp. 123–140, 1996, doi: 10.1007/bf00058655.
21. P. Kaviani and S. Dhotre, "Short Survey on Naive Bayes Algorithm," *International Journal of Advance Engineering and Research Development*, vol. 4, no. 11, pp. 607–611, 2017, doi: 10.21090/ijaerd.40826.

AUTHORS PROFILE

Dinesh C Dobhal MCA, PhD (Computer Application), He is an Associate Professor in Department of CS&E, Graphic Era University, Dehradun. He is having 10 year of experience in teaching and research. His area of research includes MANETs, Sensor Networks, Network Security, and Machine Learning.

Purushottam Das M.Tech (CS&E), PhD (Pursuing). He is Assistant Professor in Department CS&E, Graphic Era Hill University, Dehradun. His area of research includes Wireless Network, Image Processing, and Deep learning.

Kiran Aswal, M. Tech (CS&E), PhD (Pursuing) She is pursuing her PhD in Computer Science. Her area of interest includes Network Security, Machine Learning, and Deep Learning.