Android Malware Detection using Machine Learning Classifiers

Ajay Bandi, Lunduk Sherpa, and Sai Manideep Allu
{ajay, s524425, s542406}@nwmissouri.edu

Northwest Missouri State University, Maryville MO, 64468, USA

Abstract. Android has changed what mobile can do. The population of android mobile users has been increasing, and the number continues to grow as the number of mobile users increases. Primarily, the applications used in android devices are distributed via Google Play store. Using these applications. These applications are required to meet several criteria in order to be distributed via play store. However, attackers sometimes find their way to compromise the devices running android operating system and steal users sensitive information. Malicious software or Malware are softwares designed to extract data from the users by compromising their devices for financial gain or other reasons. In this study, Machine learning algorithms has been used to detect android malware. Using CIC-AndMal2017[8] dataset[1], random sampling was done to extract a balanced dataset. Feature engineering was performed to obtain the most significant features. Machine learning algorithms were trained using the balanced dataset with selected features. Using "Label" as the target variable all the models were trained and again using "Family" as the target variable. A maximum accuracy of approximately 99% was obtained using Random Forest in both cases.

Keywords: Android Malware detection · Malware · Machine Learning Algorithms · Feature Selection

1 Introduction

Nearly half of the global population use smartphones[2]. Users tend to store private information in their smartphones. The security of users information requires a top priority. Despite all the measures used to keep the information secure, malware attacks are still prevalent. There are numerous types of malware and techniques to inject them in the target system. Some of the most common and dangerous types of malware are Adware, Ransomware, Scareware, etc. Depending on their purpose and method of delivery they vary from one another. The android security sometimes fail to detect malicious software and are distributed

[1] https://www.unb.ca/cic/datasets/andmal2017.html
[2] https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

through the official play store. Users information have been found to be compromised through some programs distributed through the play store [4]. Android, being one of the most popular smartphone operating system [6] has gained many attackers attention. Thousands of malware applications have been discovered in recent years. With every update in the security features, the attacks get even more sophisticated and troublesome to detect. Machine learning algorithms can be trained to formalize the principles hidden in the dataset used to train the model. A properly trained model can reason the features of previously unseen dataset [7]. Based on the approaches taken by the model used, each model has a different capacity and problems they are best suited for. In this study, we train machine learning algorithms using CICAndMal 2017 dataset to detect android malware. The model classifies the malware into one of the four types of malware included in our dataset namely, Adware, Scareware, Ransomware and SMSmalware. The paper is organized as follows: Section 2 includes the related work where we discuss various approaches taken for android malware detection. Section 3 includes the dataset description. Section 4 discusses the procedure used in this study. Section 5 describes the results and findings of our study. Section 6 discussed the limitations and future work.In Section 7 we present the conclusion of our study.

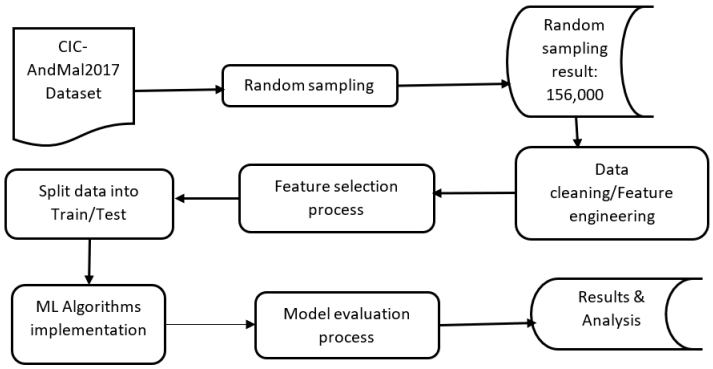

Fig. 1. Project Framework

2 Related Work

Using edge computing and traffic clustering, a new method for detecting Android malware was introduced in [5] . The authors began by sending traffic from Android devices to an edge server. Second, the edge server sends mobile traffic content features (such as plaintext extracted from HTTP flows) and traffic behavior (such as packet intervals) to the cloud platform. Finally, they used a

formula to calculate the similarities between applications and clusters in order to automatically detect malware. They used the TF-IDF algorithm and cosine similarity as similarity methods. They used 400 android applications to test their method. Because of privacy concerns, the data set was not made public online.The final average accuracy for their model was 96.9

The author of [2] used a deep learning framework based on the Long Short Term Memory (LSTM) to detect ransomware malware. Two Enhanced Android Malware Detection and Family Classification Using Conversation-level... This study chose 609 categories from the CICAndMal2017 dataset: benign and ransomware. They also used 8 feature selection algorithms, such as Chi-Square and information gain, to select the top 19 flow-level features. Their model's accuracy, recall, and F1-score results were all 97 percent.

The author of [3] used a portion of the CICAndMal2017 dataset, selecting one PCAP file for each malware family. Their samples were chosen at random. Two steps were used to extract features from PCAP files. The first step was to create a Java program that used the flowlevel technique to separate network flows. Then, using a Python program, fifteen features were extracted. The minimum size of the sent packet within a flow was one of the features. We used three supervised machine-learning classifiers. K-Nearest Neighbours, Random Forest, and Decision Tree were used as classifiers. They divide instances into two categories: malicious and non-malicious.Following that, malware was classified into three categories: adware, ransomware, and scareware. Recall, precision, and F-score are the three metrics used by the authors. The results show that the Random Forest classifier has the best results for malware-benign classification, with an F-score of 92 percent and precision and recall of 95 percent. The other classifiers accounted for more than 85% of all used metrics. For the chosen measures in formal ware classification, the selected classifiers achieved more than 80% accuracy. Similar to malwarebenign classification, the Random Forest classifier had the best recall, precision, and F-score results by 84 percent.

Based on IG and Chi-Square tests, network traffic features of Android malware are prioritized in [1]. Then, using a proposed algorithm, network traffic features were minimized to improve detection accuracy and shorten the training and testing phases. To rank features, statistical analysis techniques were used. According to the proposed algorithm, 9 out of 22 features are sufficient for higher detection accuracy. Similarly, the study's findings show that it can cut the time spent on training and testing by 50% and 30%, respectively.

3 Datasets

The dataset used in this study was obtained from the University of New Brunswick's website. CIC-AndMal2017 dataset is used to perform our analysis. This dataset contains four different types of malware. Both malware and benign applications were installed on real smartphones to generate the dataset. Three states of data

capturing was defined to acquire a comprehensive view of malware samples [3] viz. Installation, Before restart and After restart.

CICFlowMeter is a network traffic flow generator and analyzer that has been used for anomaly detection and in various cybersecurity datasets. During all three states, 85 features are extracted using CICFlowMeter-V3. The dataset downloaded from the website contained several excel files containing 42 different families of malware including benign records. In order to prepare the data for curation, all the excel files were merged together using R-Studio. An additional feature was included in the dataset so that the dataset contains a separate column for malware type and family. Four different types of malware analyzed in this study are:

1. Adware: These malware bombards the user with pop-up ads. Adware hides in user device and serves ads. Adware can also monitor users behavior online and serve ads accordingly.
2. Ransomware: A form of malware that blocks access to victim's information or computer files and demands ransom to restore access. This could result in loss of sensitive information and can cause costly disruption to operations[4].
3. Scareware: These malware manipulates users by using fake virus warning pop ups and similar other tricks to gain access to users personal information. Scareware makes user believe that their computer is infected by viruses and coax them to install unnecessary softwares.
4. SMSMalware: These malware exploit the victims mobile device via SMS text. Without users consent, this malware can cause several detrimental effects. The attacker can gain access to users personal information. In addition to that, attackers can also change, delete, expose and misuse users data.

4 Data Curation

The raw data obtained directly from the source requires to be processed before performing any analysis. The quality of the data determines how much information can be extracted from the data. We performed random sampling to obtain a balanced dataset from the raw data and using data cleaning tools we cleaned the data for further analysis.

4.1 Random Sampling

The raw dataset was not balanced and contained multiple csv files for each type of malware family. For our analysis, a balanced dataset including all type of malware is required. In order to obtain a balanced dataset, for each malware family, a csv file was created. A total of 42 different csv files were created. Random sampling has been performed on these csv file to obtain a sample of

[3] https://www.unb.ca/cic/datasets/andmal2017.html
[4] https://www.fbi.gov/scams-and-safety/common-scams-and-crimes/ransomware

Table 1. Malware Family

Malware	Family
Adware	Dowgin
	Ewind
	Feiwo
	Gooligan
	Kemoge
	Koodous
	Mobidash
	Selfmite
	Shuanet
	Youmi
Ransomware	Charger
	Jisut
	Koler
	LockerPin
	Simplocker
	Pletor
	PornDroid
	RansomBO
	Svpeng
	WannaLocker
Scareware	AndroidDefender
	AndroidSpy
	AV for Android
	AVpass
	FakeApp
	FakeApp.AL
	FakeAV
	FakeJobOffer
	FakeTaoBao
	Penetho
	VirusShield
SMSmalware	BeanBot
	Biige
	FakeInst
	FakeMart
	FakeNotify
	Jifake
	Mazarbot
	Nandrobox
	Plankton
	SMSsniffer
	Zsone

3,000 records from each family including benign records. Samples obtained after random sampling are merged together to create a balanced dataset containing 156,000 observations.

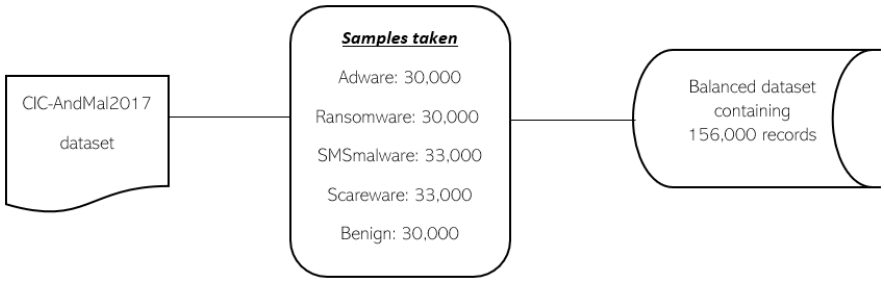

Fig. 2. Sampling framework

4.2 Data Cleaning

Data cleaning is an essential aspect of data science life cycle. For any data to be processed for analysis, it is important to clean the data. Data cleaning is a process of removing unnecessary noise from the data. Prior to data cleaning, dataset usually contain missing values, incomplete or incorrectly formatted data, varying data types, etc.

The dataset obtained after random sampling contained several missing values. Some features required formatting in order to proceed further for analysis. In this study, we performed following processes to clean the dataset:

1. Special characters were removed from the following features:
   (a) Source.IP
   (b) Destination.IP
   (c) Timestamp
2. Rows with missing values were removed.
3. All infinity values were first replaced with the Nan values and all the Nan values were filled with 0.
4. All the features datatype were formatted to Float.
5. The values in the target feature "Label" and "Family" were replaced with numerical values.
6. Single value features were removed from the data frame using variance threshold.
7. Rows containing Nan as the value for "Label" and "Family" were removed.

4.3 Feature Selection

The initial and most important step in our model design should be feature selection. The process of selecting the proper attributes that have a substantial impact on anticipating our output is known as feature selection. The key benefits of feature selection are that it lowers overfitting, improves accuracy, and shortens training time. The techniques used in our work are:

1. Univariate Selection

Univariate Selection Based on the univariate statistical tests, this feature selection selects the features that are most suited. We'll compare all of the attributes to the target variable to see if there's a significant link between them, which is what an analysis of variance, or ANOVA, is all about. For the scoring function, we've chosen f classif.

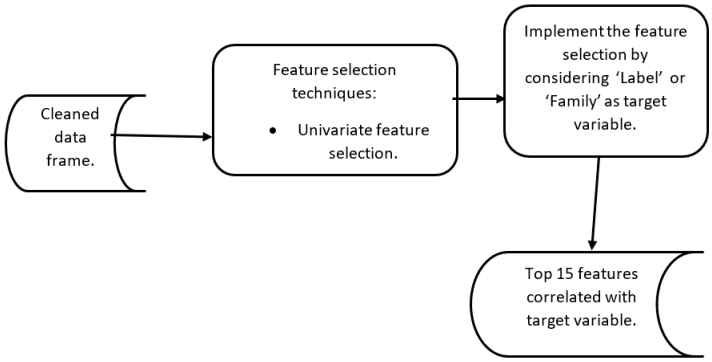

Fig. 3. Feature Selection Framework

5 Procedure

5.1 Machine Learning Algorithms

The machine learning algorithms that we had used to train the datasets are:

1. Decision Tree
2. Random Forest
3. K nearest neighbour
4. Naïve Bayes
5. Stochastic gradient boosting

	Specs	Score
5	Timestamp	27707.365004
4	Protocol	142.148710
28	Fwd.IAT.Max	140.341282
25	Fwd.IAT.Total	129.423231
3	Destination.Port	127.106005
23	Flow.IAT.Max	115.951485
35	Fwd.PSH.Flags	114.141354
49	SYN.Flag.Count	114.141354
2	Destination.IP	104.574873
6	Flow.Duration	99.293089
27	Fwd.IAT.Std	94.692103
81	Idle.Max	93.185653
1	Source.Port	89.198627
79	Idle.Mean	85.681873
22	Flow.IAT.Std	85.226464

Fig. 4. Top 15 features selected by Univariate Selection

1. Decision Tree A decision tree is a decision-making tool that predicts alternative outcomes using a tree-like decision-making paradigm. It covers the consequences of events, resource costs, and decision utility. Decision Trees seem like a flowchart or an algorithm with just conditional control statements.

2. Random Forest Random Forests is an ensemble learning technique for classification, regression, and other operations in machine learning that uses a large number of decision trees during training. They're quick and versatile, and they're a good way to mine high-dimensional data. Random Forest expands the classification and regression techniques we explored earlier using decision trees. The times for random woodland runs are quite quick. They're pretty excellent at dealing with incomplete or incorrect data. On the disadvantage, they can't predict beyond the range of the training data, and they could over-fit data sets that are particularly noisy.

3. K nearest neighbour The K-nearest neighbor (kNN) technique is a straightforward supervised machine learning approach for classification and regression. Using a similar metric, the distance function, kNN keeps track of available inputs and categorizes new data. Statistical estimation and pattern recognition are two of KNN's key uses. The distances between two requests are calculated by the KNN algorithm.

4. Naïve Bayes Naive Bayes is another very effective and commonly used machine learning classifier. The Naive Bayes algorithm family encompasses both supervised and unsupervised learning techniques. The Bayes Theorem is used to create the Naive Bayes classifiers, which are a set of classification algorithms based on the Bayes Theorem. It's a group of algorithms that have a common concept: each pair of features to be categorised is distinct from the others. The most common application of Naive Bayes is to estimate the likelihood of distinct

groups based on a range of attributes. It's largely used for text classification in data mining. If you look at Naive Bayes' applications, you'll find that this family of algorithms is the perfect fit for the projects you've always wanted to do.

5. Stochastic gradient boosting In boosting, individual models are not fully random subsets of data and characteristics. They are developed in order, with cases with inaccurate predictions and significant mistakes receiving more weight. The main premise is that instances that are difficult to correctly anticipate ("difficult" situations) will be focused on during learning so that the model can learn from earlier mistakes. When we train each ensemble on a subset of the training set, we call this Stochastic Gradient Boosting, and it can assist enhance the generalizability of our model.

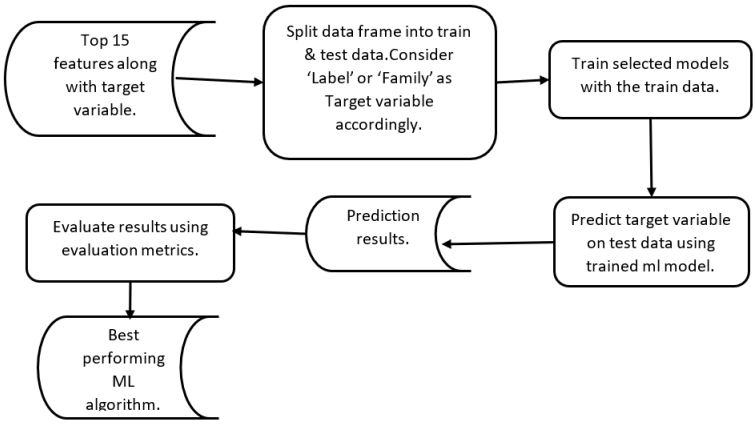

Fig. 5. ML Algorithms Implementation Framework

5.2 Methodology

In our project we had taken the randomly sampled balanced dataset. Later, we had cleaned the dataset and converted all the columns into float data format. Then, we had performed the feature selection techniques to find out the best optimal features to be given as input to the machine learning algorithms and train them. So, now by considering those top 15 features we had implemented all the machine learning algorithms by taking 'Label' feature as the target variable. Then, after evaluating the machine learning algorithms by calculating the 'accuracy','F1-score', 'Precision' 'Recall'. Later, we repeated this whole process on the 'Family' feature which been considered as a target variable. So, finally our machine learning models will calculate first whether what kind of attack it

is or is it benign. Later, it will classify which family does the malware attack belongs too.

5.3 Libraries

The libraries that are used to to implement the ML Algorithms are:

1. Sklearn
2. Matplotlib
3. pandas
4. numpy

6 Results & Findings

The features obtained after univariate feature selection were used to train our models. The dataset was split into train and test data. Using the train set we trained all the models and their corresponding performance was evaluated using measures like accuracy, precision, recall and f1-score. To assess all our data and to ensure all the observation have been used to evaluate our models we perform K-fold cross validation and the average accuracy for all the models was computed.

Table 2. Using "Label" as the target variable

ML Model	Accuracy	Precision	Recall	F1 score
Decision Tree	99.63%	99.63%	99.63%	99.63%
Random Forest	99.54%	99.54%	99.54%	99.54%
K Nearest Neighbors	93.67%	93.67%	93.67%	93.67%
Naïve Bayes	61.80%	61.80%	61.80%	61.80%
Stochastic Gradient Boosting	94.07%	94.07%	94.07%	94.07%

Table 3. Using "Family" as the target variable

ML Model	Accuracy	Precision	Recall	F1 score
Decision Tree	99.01%	99.01%	99.01%	99.01%
Random Forest	99.08%	99.08%	99.08%	99.08%
K Nearest Neighbors	62.00%	62.00%	62.00%	62.00%
Naïve Bayes	44.42%	44.42%	44.420%	44.42%
Stochastic Gradient Boosting	62.25%	62.25%	62.25%	62.25%

Using univariate feature selection, we obtained the top 15 features. Prior to that, we trained our models using all the features. Our models performed better using the top 15 features.

Table 1 shows the results obtained using the "Label" feature as our target variable. Models were trained to classify into five different classifications including benign. We obtained an accuracy of more than 90 % from all the models used except from Naïve Bayes.

Table 2 shows the results obtained using the "Family" features as our target variable. Unlike Table 1, in this case, models were trained to classify into 43 different classifications including benign. Decision Tree and Random Forest performed best of all with an accuracy of approximately 99 %.

7 Limitations

During this work we had faced problems working with the target variables. When, we had trained our ml models with the two target variables 'Label' 'Family' at a time the ml models do get trained with those target variables but we are not successfull to evaluate those multi-class multi-output scenarios. So, this was the major consrtraint we had faced while working on this study. And, also we havent performed one-hot encoding technique for feature scaling due to the large volume of data present in it. Also, due to the large volume of data we havent considered ANN model for this problem.

8 Conclusion

Overall, we had used random sampling to generate 156,000 records balanced dataset. By performing feature selection techniques we came up with the top 15 features that had been given as training dataset to the ml algorithms. After training those ml models and evaluating them we observed that decision tree, random forest stochastic gradient boosting algorithms had performed well with aroun 99.63%, 99.54% 94.076% respectively by considering 'Label' as the target variable. While if take 'Family' feature as the target variable only the decision tree random forest provided better results with 99.01% 99.08% accuracies respectively. So, finally decision tree random forest are better been used in these kind of classification problems.

References

1. Arora, A., Peddoju, S.K.: Minimizing network traffic features for android mobile malware detection. In: Proceedings of the 18th International Conference on Distributed Computing and Networking. pp. 1–10 (2017)
2. Bibi, I., Akhunzada, A., Malik, J., Ahmed, G., Raza, M.: An effective android ransomware detection through multi-factor feature filtration and recurrent neural network. In: 2019 UK/China Emerging Technologies (UCET). pp. 1–4. IEEE (2019)
3. Chen, R., Li, Y., Fang, W.: Android malware identification based on traffic analysis. In: International Conference on Artificial Intelligence and Security. pp. 293–303. Springer (2019)

4. Enck, W., Octeau, D., McDaniel, P.D., Chaudhuri, S.: A study of android application security. In: USENIX security symposium. vol. 2 (2011)
5. He, G., Xu, B., Zhang, L., Zhu, H.: On-device detection of repackaged android malware via traffic clustering. Security and Communication Networks **2020** (2020)
6. Kantar: Kantar worldpanel comtech's smartphone os market share data. (2013,2014)
7. Kaspersky: Machine learning methods for malware detection. In: Machine Learning methods for malware detection. vol. 1
8. Lashkari, A.H., Kadir, A.F.A., Taheri, L., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 International Carnahan Conference on Security Technology (ICCST). pp. 1–7. IEEE (2018)