


Machine Learning for Android Scareware Detection

Sikha Bagui, University of West Florida, USA*

 <https://orcid.org/0000-0002-1886-4582>

Hunter Brock, University of West Florida, USA

ABSTRACT

With the steady rise in the use of smartphones, specifically Android smartphones, there is an ongoing need to build strong intrusion detection systems to protect ourselves from malicious software attacks. This work focuses on a sub-group of android malware, scareware. The novelty of this work lies in being able to detect the various scareware families individually using a small number of network attributes, determined by a recursive feature elimination process based on information gain. No work has yet been done on analyzing the scareware families individually. Results of this work show that the number of bytes initially sent back and forth, packet size, amount of time between flows and flow duration are the most important attributes that would be needed to classify a scareware attack. Three classifiers, Decision Tree, Naïve Bayes, and OneR, were used for classification. The highest average classification accuracy (79.5%) was achieved by the Decision Tree classifier with a minimum of 44 attributes.

KEYWORDS

Android Malware, Decision Tree Classification, Information Gain, Intrusion Detection Systems, Malware Detection, Scareware

1. INTRODUCTION

Internet traffic on mobile devices has steadily increased over the last few years. In 2018, 85% of the mobile smartphone market share was held by android devices¹. In 2019, 98% of the internet users were mobile users². Google's android operating system, currently leading the mobile market³ (Alzaylaee, et al., 2020; Mutton & Badhani, 2019; Grampurohit, et al., 2014), is predicted to continue to have a dramatic increase in the market with around 1.5 billion android-based devices shipped by 2021 (Alzaylaee, et al., 2020). This adoption and popularity of smartphones has greatly stimulated the spread of mobile malware, especially on popular android platforms. There was an average loss of \$13.0 million due to malware or malware related attacks⁴ in 2018. Malware has seen an increase of +11% from 2018 to 2019⁴. Worldwide mobile app downloads in 2017 were 197 billion and are expected to reach 352.9 billion in 2021⁵. Being open source, Android faces additional challenges with malware infected apps (Alzaylaee et al., 2020). Hence malware detection has become one of the most important factors in the security of smartphones (Lashkari et al., 2018).

DOI: 10.4018/JITR.298326

*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Malware is any software maliciously designed to attack vulnerable services. Many vulnerable services allow malware to infect insecure accessible systems automatically (Priya et al., 2016). Other malicious attackers are able to lure victims into deliberately executing malicious code on their machines (Priya et al., 2016), giving away personal, financial, as well as healthcare data. There are many different types of malware: viruses, worms, trojans, spyware, ransomware, scareware, bots, and rootkits. This paper focuses on Scareware. Scareware is a form of malware which poses a perception of threat in order to manipulate users into buying or installing unwanted software. Mainly used to steal data, it displays frightening screens to show that your device is under attack and uses fake versions of system problem messages and virus alerts, claiming to be an antivirus solution³. A user is often tempted into installing malware without any awareness and the malware steals the users' personal information. Thus, building intrusion detection systems for the detection of malware is critical to protecting smartphone users.

The novelty of this paper lies in developing a scareware detection system using network data from the android platform (Pendlebury et al., 2018). Several papers have looked at classifying malware data in general (Alzaylaee et al., 2020; Mutton & Bhadhani, 2017; Grampurohit et al., 2014; Lashkari et al., 2018; Wu et al., 2014; Li et al., 2016; Arshad et al., 2016; Chavan et al., 2019; Kapratwar et al., 2017), but none have looked at scareware attacks in particular. Moreover, none of the works have analyzed each of the individual scareware families.

Machine learning (ML) has become a standard tool for malware detection in the academic security community, and is used to identify attacks with respectable accuracy. This paper uses three classifiers, Decision Tree, Naïve Bayes and OneR, to classify existing android scareware families using a minimal number of attributes. An Information Gain based recursive feature elimination process is used to determine the features used in classification. The novelty of this work lies in being able to detect the various scareware families using a smaller number of network attributes with respectable accuracy. The top ten attributes contributing to the classification of each scareware family are also presented.

The rest of this paper is organized as follows. Section two presents work related to malware and scareware classification; section three describes the dataset used in this work; section four presents the methodology used in this work; section five presents the results and section six presents the conclusion.

2. RELATED WORKS

There is a significant amount of work on malware classification using different approaches, but very few works have focused on scareware in particular.

2.1 Related work on Malware Classification

A study by Liu et al. (2020) presents a comprehensive review of Android malware detection approaches based on ML. Zhou and Jiang (2012) present a detailed description of the different kinds of malware, but unfortunately this paper is slightly dated, hence most of the malware families have changed. Mutton and Bhadhani (2017) is a more up-to-date work that lists the features extracted for the purpose of malware detection as well as various machine learning algorithms being used for malware detection.

The study by Kapratwar et al. (2017) applied ML techniques to analyze the effectiveness of particular static and dynamic features for detecting android malware. There are other works that looked at static features. The study by Grampurohit et al. (2014) used permissions and API level information from apps as the features to detect malicious applications. The study by Li et al. (2016) used a new feature vector extracted from the Android Manifest file, which combined permission and component information of an android application. The study by Li et al. (2016) used the Naive Bayes classifier. Chavan et al. (2019)'s work used permissions requested by an application for binary as well as multi classification using a wide variety of machine learning techniques, including Decision Trees, Random Forest, Support Vector Machines, Logistic model trees, AdaBoost, and Artificial Neural Networks. The study by Chavan et al. (2019) determined that permissions are a strong feature

and that by careful feature engineering, the number of features needed for highly accurate detection and classification can be reduced.

Other works looked at dynamic features. Alzaylaee et al. (2020) proposed DL-Droid, a deep learning system to detect malicious android applications through dynamic analysis using stateful input generation. This study achieved up to 97.8% detection rate with dynamic features only and 99.6% detection rate with dynamic plus static features, outperforming many of the other machine learning studies.

Others took a historical perspective. Wu et al. (2014) performed malware detection using runtime logs of an android application. Arshad et al. (2016) analyzed the android malware's penetration techniques.

Lashkari et al. (2018) proposed a systematic approach to generate android malware datasets using real smartphones instead of emulators and hence developed a new dataset, CICAndMal2017. Priya et al. (2016) performed binary malware detection, malware category classification, and malware family characterization by building, training, and evaluating their model via three common ML algorithms, Random Forest, K Nearest Neighbor, and Decision Trees.

Souri and Hosseini (2018) also presented a very comprehensive overview of malware detection approaches for both behavioral as well as signature-based malware.

Gopalakrishnan et al. (2020) looked at models for malware detection using diverse data mining techniques including Decision Trees and Random Forest.

2.2 Related Works on Scareware classification

Very few works have been directed to scareware classification in particular. Shahzad and Lavesson (2011) present a scareware detection method that learns patterns in extracted variable length opcode sequences derived from instruction sequences of binary files. Seifert et al. (2013) proposed an image-based scareware detection technique to identify web-based scareware attacks.

3. THE DATASET

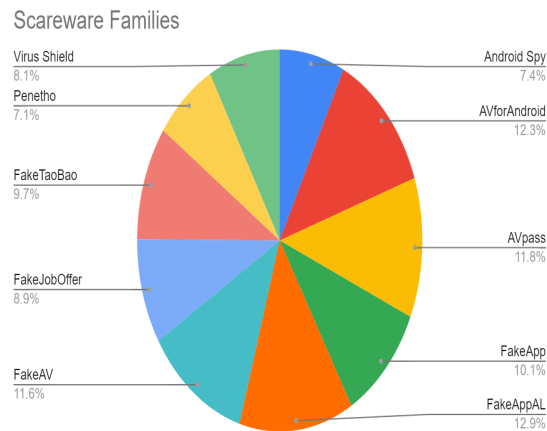
Lashkari et al. (2018) evaluated several publicly available android malware datasets from 2012 to 2017 and presented their shortcomings. These datasets lacked real-phone installation and user interaction scenarios, and contained insufficiently diverse categories and families and failed to balance the number of malware samples to benign samples (Lashkari et al., 2018). None of these datasets captured all the varieties of malware behavioral features, including continuous features (API calls, system calls, logs, network traffic) and discrete features (memory dump, permissions, memory usage, battery usage and network usage). In addition, the datasets did not balance the number of malware samples and the number of benign samples during evaluation. Extremely disbalanced datasets may not be a valid proof of work (He & Garcia, 2009; Bagui & Li, 2021). Liu et al. (2020) and the CrowdStrike 2021 Global Threat Report⁶ mention that the normal distribution of benign and malware apps in the real world is 80% to 20% respectively. Hence Lashkari et al. (2018) developed CICAndMal2017.

The dataset used in this work, CICAndMal2017 (Lashkari et al., 2018), developed by the Canadian Institute for Cybersecurity (CIC)⁷, University of New Brunswick, Canada, addresses the limitations and shortcomings of previous datasets. There are 10,854 samples collected from android applications, of which 4,354 are malware samples and 6,500 are benign. CIC collected the benign applications from the Google Play store and the malware samples from VirusTotal service⁸ and Contagio security blogs⁹. CIC then installed these on real android smartphones to execute their experiments. They were able to successfully install around 5,000 benign applications and 429 malicious applications. This was considered a success, since the industry standard for analyzing malware is 80% benign and 20% malware. They then extracted the network data using the CICFlowMeter software (Sharafaldin et al., 2018) in short bursts upon installation in the form of PCAP files. The data extracted from these tests contained 80 plus network attributes that consists of statistical features such as flow duration, number

of packets and network protocol. This work focuses on the CSV files that were an abstraction of the data extracted. The malware samples were divided into four main categories: Ransomware, Adware, Scareware and SMS Malware. This paper focuses only on Scareware and its sub categories or families.

The families that make up the scareware attack category are: Virus Shield, Android Spy, Penetho, AVforAndroid, FakeTaoBao, AVpass, FakeJobOffer, FakeApp, FakeAV, FakeAppAL. These families were the actual attacks that were present in the malware samples acquired by CIC⁷. Figure 1 shows the distribution of the families for the scareware attack category.

Figure 1. Distribution of scareware families



4. METHODOLOGY

To address the issue of highly imbalanced data, the datasets used for experimentation were balanced by composing the datasets of roughly 50% benign samples and 50% scareware samples. Data preprocessing was the next necessary step for this dataset, before classification.

The hardware configurations were used in this study were: CPU: i7 8700, RAM: 16gb and SSD: 1Tb. And, a data mining software, Weka (Witten et al., 2016), was used for preprocessing and running the classifiers.

4.1 Preprocessing

For initial preprocessing, all attributes that were specific to the environment when running the samples were removed. These attributes included: Source IP, Destination IP, Flow ID, Source Port, Time stamp and Destination Port. The data was then discretized. The next step in preprocessing involved a recursive feature selection process, performed using Information Gain in Weka.

4.1.2 Feature Selection with Information Gain

Information Gain (IG), based on a measure of entropy, measures the relevance of a feature relative to a given class. Suppose there are m classes. Let S be a set of training samples where the class label is either benign or attack. Let S contain s_i samples of class C_i , where $i = 1 \dots m$. The probability of an arbitrary sample belonging to class C_i is s_i/s , where s is the total number of samples in set S . For classifying a given sample, the expected information is (Han & Kamber, 2012):

$$I(s_1, s_2, \dots, s_m) = -\sum_{i=1}^m \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

A feature A with values $\{a_1, a_2, \dots, a_v\}$ can be used to partition S into the subsets $\{S_1, S_2, \dots, S_v\}$, where S_j contains those samples in S that have value a_j of A . Let S_j contain s_{ij} samples of class C_i . The expected information based on this partitioning by A is known as the entropy of A . The entropy is the weighted average, shown by (Han & Kamber, 2012):

$$E(A) = \sum_{j=1}^v \frac{(s_{1j} + \dots + s_{mj})}{s} I(s_{1j}, \dots, s_{mj})$$

Information Gain obtained by this partitioning on A is defined by (Han & Kamber, 2012):

$$Gain(A) = I(s_1, s_2, \dots, s_m) - E(A)$$

For the CICAndMal2017 dataset⁷, after removing the environmental variables, there were 78 attributes. Information Gain was run on Weka using the remaining 78 attributes for each of the scareware attack families. Weka's results rank the features by information gain and also present the information gain. In the next round, features with very low information gain were eliminated. That is, all attributes with Information Gain of 0.04 and below were eliminated. For most scareware families, this left 68 features, which were then used to run the classifiers.

4.2 Classifiers Used

For this study, two commonly used classifiers, Decision Tree (J48) and Naive Bayes, and one less commonly used classifier, OneR, was used. Each scareware family was run separately on a binary classifier. The distribution of data for each run was approximately fifty-fifty, meaning that each run had almost an equal amount of data from a scareware family as the amount of benign data.

4.2.1 Decision Tree

Decision tree classification has been used to detect malware by Gopalakrishnan et al. (2020). A decision tree is a tree structured representation where each node represents a feature and each link or branch represents a decision. The leaf represents the result. There are multiple decision tree algorithms. In this work, the J48 decision tree algorithm, which is a Java reimplement of the C4.5 algorithm (Han & Kamber, 2012), was used. C4.5 is an extension of Quinlan's earlier ID3 algorithm (Quinlan, 1993).

In the decision tree algorithm, Information Gain is used to decide the ordering of the features in the nodes, and the feature with the highest information gain is at the root of a decision tree. Information gain is re-calculated based on the branches, and the feature with the next highest information gain on every branch is chosen as the next root. This process occurs recursively until the tree reaches the leaf node or has one class, attack or benign. Hence, every node and branch prior to the leaf node is associated with a decision function.

The J48 decision tree algorithm was run using Weka using the following parameters: a pruned decision tree with subtreeRaising set to True, 10-fold cross-validation, confidence of 25, and MinNumObj set to 2. Pruning is very important to control overfitting. Weka's J48 Algorithm uses backward pruning, that is, it builds the complete tree and then does the pruning. Backward pruning generally performs better than forward pruning. Confidence is used to control the pruning. Lower confidence values incur more pruning. The confidence in this experimentation was set to 25%.

MinNumObj controls the splitting of the nodes. If a node has few instances, it may not be worth splitting the node. In this case the MinNumObj was set to 2. With SubtreeRaising, the interior nodes of the tree can be pruned. This raises the sub-tree beneath the interior node up one level during pruning, but the tree is better pruned (Witten et al., 2016).

4.2.2 Naïve Bayes

Naive Bayes is a simple, yet effective and commonly-used, probabilistic machine learning classifier¹⁰. Based on the Bayes' Theorem, this classifier predicts class membership probabilities, that is, the probability that a given tuple belongs to a particular class (Han & Kamber, 2012). The Naïve Bayes classifier assumes class conditional independence. This means that the effect of an attribute value on a given class is independent of the values of the other attributes. The Naïve Bayes classifier does not perform as well in the presence of irrelevant (Bermejo et al., 2014) or highly correlated attributes (Inza et al., 2000), hence feature selection is very important for the Naïve Bayes classifier.

4.2.3 OneR

OneR short for "One Rule", is a not very commonly used simple classifier, but has shown to achieve surprisingly high accuracy in many datasets (Nevill-Manning et al., 1995; Holte, 1993). Based on a one-level decision tree¹¹, it has one branch for each value where each branch assigns the most frequent class, and the error rate is determined by the proportion of instances that do not belong to the majority class and their corresponding branch and the attribute with the smallest error rate is chosen^{11, 12}. An attack that is classified accurately by OneR depends very strongly on a single feature. For this experimentation, Weka's default parameters were used to run the OneR.

5. RESULTS

5.1 Evaluation Metrics Used

To evaluate the quality of the classification process, accuracy, precision, recall and the F-measure were used.

Accuracy is the ratio of a model's correct data (TP + TN) to the total data, calculated by:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total No. of Instances} \quad (1)$$

Precision is the positive predictive value, or the percent of attack instances that are truly classified as attacks. Precision is calculated by:

$$\text{TP} / (\text{TP} + \text{FP}) \quad (2)$$

Recall or Attack Detection Rate (ADR) or sensitivity is the effectiveness of a model in identifying an attack. The objective is to get a higher ADR. Recall or ADR is calculated by:

$$\text{TP} / (\text{TP} + \text{FN}) \quad (3)$$

F-measure is the harmonic mean of precision and recall. The higher the F-measure, the more robust the classification model. The F-measure is calculated by:

$$2 * ((\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})) \quad (4)$$

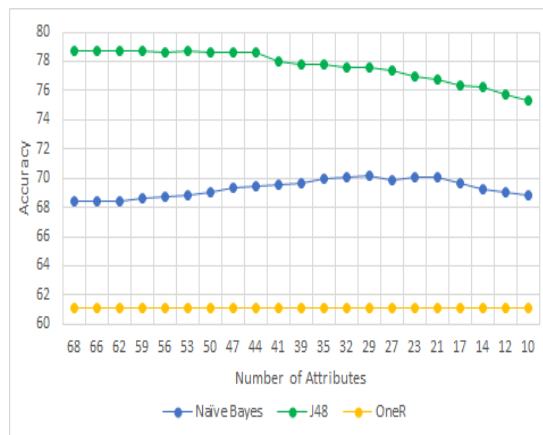
Where TP is true positive, FP is false positive, FN is false negative and TN is true negative.

5.2 Comparing Results of the J48, Naïve Bayes, and OneR classifiers

The classification accuracy, precision, recall and F-measure were determined for 68 attributes down to 10 attributes. A recursive attribute elimination process was used based on information gain. First, 68 attributes for each of the scareware families were run on each of the three classifiers, J48, Naïve Bayes, and OneR, and the accuracy, precision, recall and F-measure recorded. Then, based on information gain, the attribute with lowest information gain was removed and this process was repeated for 67 attributes, then 66 attributes, and so forth. At every point, information gain was used to remove the attribute with the lowest information gain.

Hence, the classifiers were run on each set of attributes for each scareware family after removing one attribute at a time, using an information gain based recursive feature elimination, until the target number of ten attributes. Figure 2 presents the classification accuracy of the average of eleven scareware attack families, run for different numbers of attributes, on each of the three classifiers. On the x-axis a selected number of attributes are shown in order to keep the graph readable. This graph shows the trend of each of the classifiers as well as the behavior of the number of attributes. Figure 2 shows which classifier performed the best on the average, in terms of accuracy.

Figure 2. Average Accuracy of Naïve Bayes, J48 and OneR for various number of attributes



From Figure 2 it can be noted that J48 consistently had the highest classification accuracy, approximately between 79% to 75%, for 68 to 10 attributes respectively. With J48, as the number of attributes decreased, the classification accuracy also went down slightly, but by not very much. The best classification accuracy was at 44 attributes.

Naïve Bayes performed the second best, with a classification accuracy ranging from a little above 68% to 72% for 68 to 10 attributes respectively. For Naïve Bayes, the highest classification accuracy

Table 1. Average precision, recall and f-measure of algorithms

	Average Precision	Average Recall	Average F-Measure
J48	0.7576	0.7573	0.757
Naïve Bayes	0.7101	0.7094	0.7093
OneR	0.6272	0.6264	0.6257

was with about 21 attributes to 37 attributes, but overall, the number of attributes did not have very much of an effect on the classification accuracy.

OneR had the lowest classification accuracy, around 61%, regardless of the number of attributes. Given the fact that OneR performs the best if there is one attribute that contributes highly to the classification, this shows that there is certainly more than one attribute contributing highly to the classification.

Table 1 presents the average precision, average recall and average F-measure of all three classifiers, J48, Naïve Bayes and OneR. This is the average of all eleven scareware families for all attributes, 68 to 10.

From Table 1, it can also be observed that J48 performed the best of the three classifiers, with an average precision of 75.76%, an average recall or attack detection rate of 75.73%, as well as an average F-measure of 75.7%. OneR had the lowest performance of the three classifiers with the average precision, average recall and average F-measure in the low 60%'s. Naïve Bayes performed quite a bit better than OneR, with average precision, average recall and average F-measure in the low 70%'s, but not as well as the J48. Since J48 consistently performed the best of the three classifiers, only the results of J48 are presented moving forward.

Tables 2-11 present the top ten attributes and Information Gain (IG) of each of the scareware families by IG using the J48 classifier. From Table 2 it can be noted that most important attribute for the classification of AndroidDefender is *Init_Win_bytes_forward*, the second most important attribute is *Init_Win_bytes_backward*, the third most important attribute are *Total_Length_of_Bwd_Packets* and *Subflow Bwd_Bytes* (since the latter two attributes have the same information gain), and so forth. From the results of Tables 2-11, it can also be noted that some of attributes are common to more than one family.

Table 2. Top ten attributes of AndroidDefender

AndroidDefender	
Attribute	IG
Init_Win_bytes_forward	0.055
Init_Win_bytes_backward	0.043
Total Length of Bwd Packets	0.038
Subflow Bwd Bytes	0.038
Total Length of Fwd Packets	0.035
Subflow Fwd Bytes	0.035
Max Packet Length	0.033
Fwd Packet Length Max	0.032
Fwd Packet Length Max	0.032

Table 3. Top ten attributes of AndroidSpy

AndroidSpy	
Attribute	IG
Init_Win_bytes_forward	0.086
Init_Win_bytes_backward	0.06
Source Port	0.058
Flow IAT Max	0.057
Flow Duration	0.056
Flow IAT Min	0.053
Max Packet Length	0.047
Flow Packets/s	0.046
Flow IAT Mean	0.046

Table 4. Top ten attributes of AVforAndroid

AVforAndroid	
Attribute	IG
Init_Win_bytes_forward	0.074
Fwd Packet Length Max	0.06
Flow Duration	0.055
Init_Win_bytes_backward	0.054
Flow IAT Max	0.047
Flow IAT Mean	0.043
Max Packet Length	0.041
Flow IAT Min	0.041
Flow Packets/s	0.04

Table 12 presents average Information Gain (for the scareware families), the standard deviation of the Information Gain, and the number of times the attribute occurred within the top ten attributes in each of the scareware families (frequency). It can also be observed that *Init_Win_bytes_forward* was present in all 11 of the scareware families, and had the highest information gain in 7 of those families and had the second highest information gain in 2 of those families. *Flow IAT Max*, *Flow Duration*, and *Max Packet Length* were present in eight of the eleven scareware families. And, each of these attributes had only slight deviation from the mean in Information Gain. *Init_Win_bytes_backward* was present in seven of the eleven scareware families.

Init_Win_bytes_forward is the total number of bytes sent in the initial window in the forward direction. Transmission Control Protocol (TCP) uses a sliding window flow control protocol. In each TCP segment, the receiver specifies, in the receive window field, the amount of additionally received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgment and window update from the receiving host (Lashkari et al., 2018). Hence, *Init_Win_bytes_forward* can be the most important

Table 5. Top ten attributes of AVpass

AVpass	
Attribute	IG
Fwd Packet Length Max	0.071
Flow Duration	0.069
Flow IAT Max	0.067
Flow IAT Min	0.064
Init_Win_bytes_forward	0.063
Flow Packets/s	0.06
Max Packet Length	0.053
Fwd IAT Min	0.053
Flow IAT Mean	0.052

Table 6. Top ten attributes of FakeApp

FakeApp	
Attribute	IG
Source Port	0.052
Init_Win_bytes_forward	0.05
Max Packet Length	0.033
Init_Win_bytes_backward	0.032
Subflow Fwd Bytes	0.026
Total Length of Fwd Packets	0.026
Fwd Packets/s	0.026
Fwd Packet Length Max	0.025
Flow Packets/s	0.025

Table 7. Top ten attributes of FakeAppAL

FakeAppAL	
Attribute	IG
Init_Win_bytes_forward	0.077
Init_Win_bytes_backward	0.057
Flow IAT Max	0.043
Flow Duration	0.043
Max Packet Length	0.041
Fwd IAT Total	0.037
Flow IAT Min	0.037
Fwd IAT Max	0.033
Average Packet Size	0.032

Table 8. Top ten attributes of FakeAV

FakeAV	
<i>Attribute</i>	<i>IG</i>
Init_Win_bytes_forward	0.067
Fwd Packet Length Max	0.065
Flow IAT Min	0.06
Flow IAT Max	0.057
Fwd IAT Min	0.057
Flow Duration	0.056
Fwd Packets/s	0.046
Flow Packets/s	0.046
Max Packet Length	0.044

Table 9. Top ten attributes of FakeJobOffer

FakeJobOffer	
<i>Attribute</i>	<i>IG</i>
Init_Win_bytes_forward	0.07
Init_Win_bytes_backward	0.049
Fwd Header Length2	0.035
Fwd Header Length	0.035
Source Port	0.035
Flow Duration	0.032
Subflow Fwd Bytes	0.03
Total Length of Fwd Packets	0.03
Flow IAT Max	0.029

Table 10. Top ten attributes of Penetho

Penetho	
<i>Attribute</i>	<i>IG</i>
Init_Win_bytes_forward	0.104
Flow IAT Min	0.081
Source Port	0.076
Flow IAT Max	0.073
Flow Duration	0.072
Fwd IAT Min	0.07
Fwd Packets/s	0.068
Flow Packets/s	0.061
Flow IAT Mean	0.058

Table 11. Top ten attributes of VirusShield

VirusShield	
Attribute	IG
Flow IAT Max	0.086
Init_Win_bytes_forward	0.07
Source Port	0.068
Init_Win_bytes_backward	0.056
Flow Duration	0.055
Fwd Header Length	0.052
Fwd Header Length2	0.052
Fwd IAT Min	0.051
Bwd IAT Max	0.048

Table 12. Top ten attributes of the scareware families

Attribute Name	Average Information Gain	σ	Frequency
Init_Win_bytes_forward	0.072	0.015	11
Flow IAT Max	0.057	0.018	8
Flow Duration	0.055	0.013	8
Max Packet Length	0.042	0.007	8
Init_Win_bytes_backward	0.0501	0.01	7
Flow Packets/s	0.046	0.013	6
Flow IAT Min	0.056	0.016	6
Fwd Packet Length Max	0.0506	0.021	5
Source Port	0.0578	0.016	5

attribute. Much of the data being sent is using over the 65,535 bytes, an indication that many bytes can be received before an acknowledgement, and in return it receives another byte size to expect, which is *init_win_bytes_backward*. With this information it can be speculated that the malicious software is most likely going to request the largest size and it will receive the same response byte size every time. With it requesting the same size, it could be inferred that *init_win_bytes_forward* and *init_win_bytes_backward* can be used to detect malware because every time the max window bytes are requested, it expects the same window size.

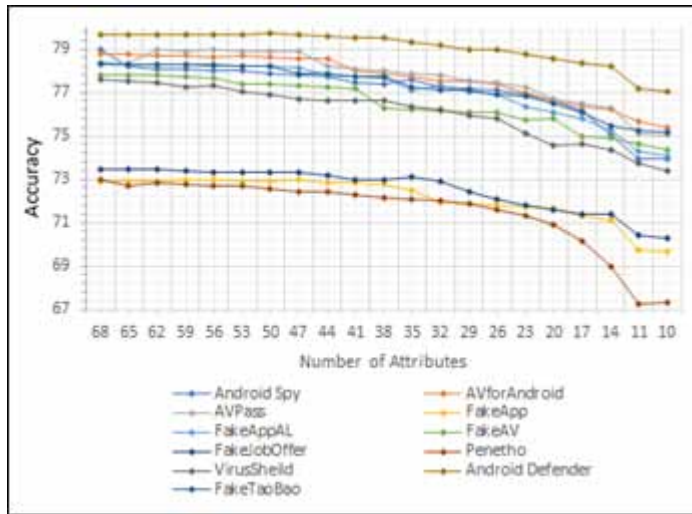
Flow IAT Max is the maximum value of the inter-arrival time of the flow. This is the maximum amount of time between two packets sent in the flow (Lashkari et al., 2018).

Flow Duration is the duration of the flow or how long it took to complete (Lashkari et al., 2018).

Max Packet Length is the maximum length of the packets registered in the flow (both forward and backward directions) (Lashkari et al., 2018).

Figure 3 presents the classification accuracy for 68 to 10 attributes respectively for each of the scareware families using the J48 classifier. Overall, Android Defender had the highest accuracy, between 77% to 80%, and Penetho had the lowest accuracy, between 67% and 73%. FakeApp was also pretty close to Penetho, so FakeApp and Penetho can be considered the hardest to detect.

Figure 3. Accuracy for the Scareware families using J48



6. CONCLUSION

This paper used an information gain based recursive feature elimination process to determine the optimal number of attributes needed for classification for scareware families of attacks. Of the three classifiers used, J48 also had the highest average precision (75.7%), recall (75.7%) and F1 score (75.7%).

The results show that number of bytes initially sent back and forth, packet size, the amount of time between flows and flow duration are the most important attributes that would be needed to classify a scareware attack. Specifically, the *Init_Win_bytes_forward* attribute appeared to be the most important attribute, present within the top ten attributes for all the scareware attack families. Although the top ten attributes for each scareware attack family were presented, the best classification results appeared to be for 44 attributes for the J48 classifier at 79.5% (as can be seen in Figure 2). Reducing the attributes below 44, for the J48 algorithm, reduced the classification accuracy slightly, but not by very much.

From this study, it can also be generalized that scareware families are difficult to detect. Of the scareware families, Android Defender had the highest classification accuracy (79.5% with 44 attributes) while Penetho had the lowest classification accuracy (the highest accuracy for Penetho was at 73% with 66 attributes), so the highest classification accuracies were between 79.5% and 73%. For our future plans, we aim to do more studies to see how these classification rates can be improved.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.

REFERENCES

- Alzaylaee, M. K., Yerima, S. Y., & Sezer, S. (2020). DL-Droid: Deep Learning Based Malware Detection Using Real Devices. *Computers & Security*, 89, 1–11. doi:10.1016/j.cose.2019.101663
- Arshad, S., Shah, M.A., Khan, A., & Ahmed, M. (2016). Android Malware Detection and Protection: A Survey. *International Journal of Advanced Computer Science and Applications*, 7(2). 10.14569/IJACSA.2016.070262
- Bagui, S., & Li, K. (2021). Resampling Imbalanced Data for Network Intrusion Detection Datasets. *Journal of Big Data*, 8(6), 1–41. doi:10.1186/s40537-020-00390-x
- Bermejo, P., Gamez, J. A., & Puerta, J. M. (2014). Speeding up incremental wrapper feature subset selection, with Naïve Bayes Classifier. *Knowledge-Based Systems*, 5, 140–147. doi:10.1016/j.knosys.2013.10.016
- Chavan, N., Troia, F. D., & Stamp, M. (2019). A Comparative Analysis of Android Malware. *3rd International Workshop on Formal Methods for Security Engineering (ForSE), in conjunction with the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*. arXiv:1904.00735
- Gopalakrishnan, P., Narayanan, R. S., Kamath, R., & Ramani, A. (2020). Analyzing Diverse Data Mining Techniques to Detect the Malware based on Signature. *Test Engineering and Management*, 83, 17717–17724.
- Grampurohit, V., Kumar, V., Rawat, S., & Rawat, S. (2014). Category Based Malware Detection for Android. In J. L. Mauri, S. M. Thampi, D. B. Rawat, & D. Jin (Eds.), *Security in Computing and Communications. In SSCC 2014. Communications in Computer and Information Science* (Vol. 467). Springer. doi:10.1007/978-3-662-44966-0_23
- Han, J., & Kamber, M. (2012). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers.
- He, H., & Garcia, E. A. (2009). Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 1263–1284. doi:10.1109/TKDE.2008.239
- Holte, R. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11(1), 63–91. doi:10.1023/A:1022631118932
- Inza, I., Larranga, P., Etxeberria, R., & Sierra, B. (2000). Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence*, 123(1-2), 157–184. doi:10.1016/S0004-3702(00)00052-7
- Kapratwar, A., Troia, F. D., & Stamp, M. (2017). Static and Dynamic Analysis of Android Malware. *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, 653-662. doi:10.5220/0006256706530662
- Lashkari, A. H., Kadir, A. F., Taheri, L., & Ghorbani, A. (2018). Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification. *2018 International Carnahan Conference on Security Technology (ICCST)*, 1-7. doi:10.1109/CCST.2018.8585560
- Li, X., Liu, J., Huo, Y., Zhang, R., & Yao, Y. (2016). An Android malware detection method based on AndroidManifest file. *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 239-243, doi:10.1109/CCIS.2016.7790261
- Liu, K., Xu, S., Xu, G., Zhang, M., Sun, D., & Haifeng, L. (2020). A Review of Android Malware Detection Approaches based on Machine Learning. *IEEE Access*. 10.1109/ACCESS.2020.3006143
- Mutton, S. K., & Badhani, S. (2017). Android Malware Detection: State of the Art. *International Journal of Information Technology*, 9(1), 111–117. doi:10.1007/s41870-017-0010-2
- Nevill-Manning, C., Holmes, G., & Witten, I. H. (1995). The Development of Holte's IR classifier. *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, 239. doi:10.1109/ANNES.1995.499480
- Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., & Cavallaro, L. (2018). *TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time*. arXiv:1807.07838v4
- Priya, A., Singh, K., Tiwari, R., & Vishwaividyalaya, B. (2016). A Review on Malware Analysis by using an Approach of Machine Learning Techniques. *International Journal Online of Sports Technology and Human Engineering*, 3(5), 5.

- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc.
- Seifert, C., Stokes, J. W., Colcernian, C., Platt, J. C., & Lu, L. (2013). Robust scareware image detection. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2920-2924, doi:10.1109/ICASSP.2013.6638192
- Shahzad, R. K., & Lavesson, N. (2011). Detecting Scareware by Mining Variable Length Instruction Sequences. *Information Security in South Africa*, 1-8. 10.1109/ISSA.2011.6027523
- Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)*, 108-116. doi:10.5220/0006639801080116
- Souri, A., & Hosseini, R. (2018). A State-of-the-art survey of malware detection approaches using data mining techniques. *Hum. Cent. Comput. Inf. Sci.*, 8(1), 3. doi:10.1186/s13673-018-0125-x
- Witten, I. H., Eibe, F., Mark, A. H., & Christopher, J. P. (2016). The WEKA Workbench. Online Appendix: Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann Publishers Inc.
- Wu, W-C., & Hung, S-H. (2014). DroidDolphin: A Dynamic Android Malware Detection Framework using Big Data and Machine Learning. *RACS*, 247-252.
- Zhou, Y., & Jiang, X. (2012). Dissecting Android Malware: Characterization and Evolution. *IEEE Symposium on Security and Privacy*, 95-109.

Sikha Bagui is Professor and Askew Fellow in the Department of Computer Science, at The University West Florida, Pensacola, Florida. Dr. Bagui is active in publishing peer reviewed journal articles in the areas of database design, data mining, BigData and Big Data analytics, and machine learning. Dr. Bagui has worked on funded as well unfunded research projects and has over 85 peer reviewed publications. She has also co-authored several books on database and SQL. Bagui also serves as Associate Editor and is on the editorial board of several journals.

Hunter Brocker completed his BS in Computer Science at The University of West Florida.